Paper

History of Game Development Options in Game Studies

ゲーム研究におけるゲーム開発の選択肢の歴史

BRANDSE Michael ブランセ マイケル

Digital Hollywood University, Assistant Professor
デジタルハリウッド大学 助教

Game Development has historically been a complicated field, drawing from a wide variety of specializations, including but not limited to programming, graphic design and interaction design. However, this need for many high level skills has meant that game research has been difficult, as researchers either don' t have the skills, time or budget for game development. This has limited game research primarily to studies using existing developed games, which is particularly problematic for research fields that depend on comparative research. In recent years however, the situation has markedly improved through 4 key developments; financial accessibility, an increased knowledge base, the forming of the game development middleware industry as well as the rise of visual scripting. In this paper, we explore these key developments as well as their history, and how these can potentially impact game research.

キーワード：Game Research Methodology, Game Development, Interaction Design

1. Introduction

1.1 The Closed Nature of the Game Industry

The game development industry has historically been very tight lipped about what kind of design processes they use. It' s also relatively uncommon for game developers to be active readers of research [1]. This means that game research and game development are very much disconnected, which is made worse by the fact that research is a slow process, whereas the game industry needs to keep innovating at high speed or risk being left behind by their competitors.

While this has gotten better over the years and game developers are sharing more development knowledge with the public [2][3], generally the type of knowledge they share is concrete in nature. For instance, specific graphics techniques and algorithms, as well as technology used to optimize game rendering are relatively easy to present since they have measurable effects. However, more "fuzzy" knowledge, like game design methods, are still very rare and often either kept behind closed doors, or only showcased at events with relatively high barriers of entry, such as the annual Game Developers Conference. Even then, we cannot expect that developers will share the latest innovations, as was evidenced by research we ourselves have conducted. In previous affordance design for games we conducted [4] we presented a model for challenge encounters in games, in which we likened these encounters in games to doors and puzzles, with the doors needed to be opened with a key gained through solving the puzzle, in order for the player to progress. In the Game Developers Conference 2018, D. Shaver and R. Yang from Naughty Dog presented something remarkably similar about level design guidance principles they called gates and valves [5], where the gates resembled the doors we defined in our own research and the valves took the role of opening those doors.

There are a number of books by experienced game developers that teach about game design methods, but these too are relatively rare [6][7], meaning we generally cannot count on game developers to make their results and expertise available to a wider audience. Therefore, our best chance of making game design methodology more known to wider audience relies on game design researchers to create a knowledge base in regards to game design methods.

1.2 The Problematic Nature of Game Development

Unfortunately, depending on game design researchers to conduct research on game design methods is problematic in its own right. The reason for this is that in order to conduct research into design methodology, one would need to be well-versed in game development. This is difficult, since game development is a multi-faceted development process; to create a game, one needs a combination of skills from fields such as programming, graphic design, animation, sound design, interaction design, and many more. This is a level of knowledge game researchers generally don' t have. Another option would be to hire developers to make up for the lacking knowledge, but since game development is not a set process, even experienced developers will have trouble properly estimating when their game will be effective, or "fun."

Even on the fourth entry in a series that Naughty Dog had been making for a decade, it was still impossible to plot out just how long everything would take. "The problem is, you couldn' t task creativity," said Bruce Straley. "You can' t task fun." [8]

Furthermore, research budgets generally don' t approach the size of even medium sized games, with even medium sized game development project budgets potentially being several millions of dollars large.

For that reason, researchers who specialize on game research will often-times base their research on existing games [9][10], as it' s much less time consuming and much more economically viable to analyze existing games rather than build the games by yourself. This has limits of course, as researchers will be depending on existing content they generally cannot modify, meaning that comparative research (especially when the researcher has specific goals) is hard to conduct.

2. Changes in Game Development Options

Fortunately, especially in the last decade there have been a number of developments, both financially and technologically, that has significantly lowered the barrier of entry to game development. We have observed in particular four key developments that contributed significantly to the lowering of the barrier of entry.

1. Financial accessibility.
2. Game development knowledge base.
3. Game development middleware.
4. Visual scripting systems.

In the following chapters, we will go into more detail in what these four key developments are and how they have impacted game developers.

3 Financial Accessibility

Game development is primarily conducted using specialized tools called game engines. These game engines are responsible for calculating and rendering the real-time content to screen and oftentimes offer a variety of tools to make otherwise tedious tasks more manageable and less time consuming.

Traditionally, game development companies would make their own proprietary game engine which would then be used for their game projects. Examples of these are the Frostbite engine [11], developed by DICE and used by Electronic Arts, as well as the Luminous Engine [12], developed and used by Square Enix as well as its subsidiary Luminous Productions. Game engines are generally be unavailable to the general public, like the examples mentioned above, but sometimes companies decide to allow licensing of their engine. Examples of this is the Quake Engine, by ID Software and the Unreal Engine, developed by Epic Games.

However, licensing a game engine doesn' t come cheap. While information on the price of licensing is scarce, Gestalt from Eurogamer stated that a license to the Quake 3 engine [13] could be around $500,000 per game project, excluding royalties on the sales of said project. This is of course unfeasible for researchers, as this only concerns the game engine itself and not yet any contents created with it.

There were cheaper options, but generally these options didn' t offer very robust solutions or had very limited support. One example of this is a game engine called Virtools [14], which was released in 1993. The software was only used in a small number of regions in Europe and thus had a very limited support base. The features were also very bare, meaning that researchers would have to have in-depth knowledge of various game development processes in order to make effective content with it, such as the manual creation of light maps. Interestingly however, Virtools did offer a visual scripting solution, something that other game engines would only start implementing much later. The limited support however meant that Virtools eventually had to be discontinued in 2009.

A major change in the traditional game engine licensing model occurred in 2004, when Unity Technologies created a game engine with the name Unity [15]. Rather than aiming for game developers developing high budget productions, Unity Technologies instead opted to specifically target independent developers with much smaller budgets [16]. For the first 10 years of the software, Unity was sold as is, with prices ranging from $199 to $1499 [17], prices that were a massive departure from the usual licensing pricing. From 2009 onwards, Unity started to offer free versions of their software as well [17], before changing entirely to a subscription based service in 2016 [18].

Shortly after Unity announced their decision to make one version of Unity free to use in 2009, Epic Games also released a game engine that was free to use under the name Unreal Development Kit (UDK) [19]. The Unreal Development Kit was based on the Unreal 3 engine technology, and allowed users to make content for the PC or IOS platform. While the Unreal 3 engine was technically also made available for modding game content through games like Gears of War for PC [20], the release of a completely free UDK was the first time that Epic Games released the engine for free by itself. While the software itself was free to use for non-commercial purposes, if a developer decided to sell their creation, they had to pay Epic Games a one-time fee of $99 and 25% royalties on any profit they made over $50,000 [21].

In 2014, Epic Games released the Unreal 4 engine and initially charged users a monthly subscription fee of $19 [22]. However, the subscription didn' t only give users access to the Unreal 4 engine, it also gave them access to the engine' s source code, which was unheard of before that. Now users could not only use a fully featured game engine for a small charge, they could modify the source code as well. While this required extensive knowledge of C++, it did mean that users could customize the engine for their own goals, research included. Finally, one year later Epic Games decided to remove the monthly subscription fee as well and make the Unreal 4 engine free to use [23], a model they have continued to use even with the release of an early access version of Unreal 5 engine [24].

Other game engine developers also followed suit, with the CryEngine [25] becoming free to use in 2018, after having adopted a “pay what you want” model for a number of years. At QuakeCon 2005, John Carmack announced that the id Tech 3 source code (formerly known as the Quake 3 Arena engine) would be made freely available under the GNU General Public License v2.0 [26].

4 Game Development Knowledge Base

Initially, a large reason for why game development was hard to get into is because knowledge regarding game development was hard to come by. This didn't only regard research into design methodology or knowledge on concrete technologies like particular graphical effects, like we discussed earlier, but also practical knowledge dealing with how to create 3D content for real-time use or how to program a game.

Since game engines were initially only available to major game developers, it meant that a knowledge base that could be accessed by all was simply not a necessity. After all, there was very little use in explaining the usage of the Quake 3 engine, when said engine could cost $500,000 to license.

When Unity Technologies targeted small to medium teams with their Unity engine, it meant that even people who were simply making games for a hobby could get into game development. When Unreal 4 became free to use as well, game engines experienced a surge in popularity. This meant that knowledge of how to use the engines effectively also became more common, for both premium [27][28] as well as free options (through the use of personal blogs, Youtube videos, and so on).

With the engines gaining in popularity, the companies responsible for creating these engines also started organizing conventions specifically meant to dissertate information regarding their own technologies. In 2017 Unity Technologies created the Unity Unite convention [29]. Epic Games created the Unreal Fest [30], an event that allowed for the sharing of Unreal engine techniques and technologies.

In past research we conducted into the immersive qualities of interactive environments [36], we had to prepare a number of stimuli in the form of game environments. To gather the impressions that players had of these environments, we conducted a questionnaire. When the experiment was conducted, we were relying on Unreal 3 engine technology, whose source code was inaccessible. This meant that in order to write out data from the game to external text files, we had to rely on a technique called “DLL bind” [31], which required knowledge of DLL creation as well as linking the created DLLs to UnrealScript, which was Unreal 3 engine’s programming language. Unfortunately, material on how to create a DLL that could write out text files was hard to find, and due to our limited programming ability we had to create an alternate experiment flow where we constantly interrupted subjects to fill in questionnaires through external hardware. In follow up research presently being conducted, we used the Unreal 4 engine instead. We made another attempt at writing out text files from within the game, to create an experiment with a smoother flow. Due to the popularity of the engine, it was relatively easy to find a tutorial on this subject [32] and we were able to create a solution that enabled us to conduct questionnaires from within the game environment. This would not have been possible without the Unreal 4 engine being popular enough to have many people write tutorials for it.

5 Game Development Middleware

Middleware services for 3DCG and game development have been around for a while, and come in two forms; the sale of premade assets that the costumer can include in their projects or tools that help make the development process easier.

5.1 Middleware to ease development time

Initially, game development was a very time consuming process, as assets made for real time usage had to be optimized, while trying to keep an acceptable level of detail, or even outright faking the presence of detail. For instance, a game ready 3D model would require a number of textures to render correctly. For games developed for the Playstation 3 (2006) and the XBOX360 (2005), a wide variety of textures were required to visualize a 3D model. A base color texture to take care of the colors of the model, a specular map to deal with specular data, a normal map to generate fake normal data on a 3D model to make it feel that models contained more details than they actually did, alpha textures to make parts of the model invisible, as well as emissive textures to create the illusion that certain parts of the model glowed or otherwise emitted light. The Playstation 4 (2013) and the XBOX One (2013) were capable of rendering more advanced rendering models, and during this age PBR (Physically Based Rendering) [33] became the primary means of rendering textures on 3D models. However, this new model introduced the need for additional textures to simulate the difference between conductive and dielectric materials, as well as textures to simulate the overall roughness and thus how light would reflect off the surface if it hit.

Traditionally, all these assets would have to be made by hand, requiring complicated setups to calculate all these various maps using a combination of high and low definition models. Some textures were required to be painted or modified by hand. To reduce the burden of rendering multiple objects at the same time, game developers would have to create various LOD (Level of Detail) models, models that were swapped to models with lower polygon counts to increase the performance. A wider variety of techniques for real time graphics optimization existed, but all these would have to be prepared by hand as well. This meant that even for the average experiment, the researcher would have to spend a lot of time just to prepare one single stimuli.

This started to change around 2000, when a variety of middleware developers were born. One well-known example is a company called Havok, responsible for a physics calculation software development kit also known as Havok Physics [34]. Since its founding, Havok Physics has been used in more than 600 games. Another famous example is Speedtree [35], developed by Interactive Data Visualization, software that was meant to quickly and efficiently generate various kinds of foliage, a particular type of asset that would normally require a lot of time to create. InstaLOD [36], developed by Crunchbase in 2016, would quickly generate level of detail meshes, so that the developers would not have to bother themselves. It was not just limited to making specific development processes easier. In 2003, Allegorithmic was founded and would eventually release Substance Designer and Substance Painter, two software suits that would make texture generation for real time usage

a lot easier. For instance, in Substance Designer designers would create a procedural texture using visual scripting, which could then easily be rendered out as metallic textures (to determine conductive and dielectric conditions), roughness textures, normal textures and so on, whereas normally a developer would have to make these one by one. Even game engines started incorporating tools into their engine. The Unreal 3 engine, for instance, had in-engine tools to generate level of detail meshes or to generate UV coordinates for light map textures, textures that were used to create cheap lighting and shading performance wise. The Unreal 5 engine even offers a direct link to Quixel's models [37], a company who specializes in the creation of photo-scanned 3D models and textures and was acquired by Epic Games in 2019.

5.2 Pre-made assets as middleware

Middleware software meant to speed up the development process is not the only major change in the field of middleware for game development. Another one is the sale of premade assets, which users can immediately use in their projects without having to create it themselves. One well-known example of a service selling assets to customers is Turbosquid, a site that sells 3D models and has done so since 2000. While initially Turbosquid's offerings were primarily meant for 3DCG (meaning they sold high definition models that would have difficulty being rendered in a real time application), currently they also have a number of so-called "low poly" models on sale, which can be rendered in real time applications without problems. Another well-known example is textures.com and gametextures.com, the former offering a wide variety of textures for various uses and the latter primarily focusing on textures that can be used within real time applications. While textures.com was initially completely free to use, currently they use a credit system where users can download only a set of textures per day, and offer a number of premium plans where users get more credits to use on a monthly basis. There are a variety of other services, like FreeSound.org, which focusses on sound effects that can be downloaded at no charge and Mixamo, a supplier focusing on 3D character models and animations.

In 2010 Unity Technology introduced the asset store, an online marketplace where developers could sell real time content specifically meant to speed up development time for other developers. This was the first time that content specifically developed for a game engine was sold through a marketplace. The asset store quickly became popular and now boasts over 70,000 asset packs. The asset store's success meant that Epic Games also created an online marketplace when they released the Unreal 4 engine, simply called the Unreal Engine Marketplace. The Unreal Engine Marketplace currently offers over 19000 asset packs for use with the Unreal 4 engine, across a wide variety of categories.

In our own experience, we have found that the development of real-time assets became much more effective when using existing asset packs. In past research [38] we set out to define the game world as a narrative tool, and to validate the model we developed, we measured the immersion rates in digital game environments. To validate our narrative model, we needed to use game environments that increased in detail, so we could measure the effects of environments with and without the narrative elements in our model. Due to this, the development of a variety of custom environments was a necessity. However, since all assets had to be created by hand, it took us 6 months to design and develop a total of two environments (Figure 1).

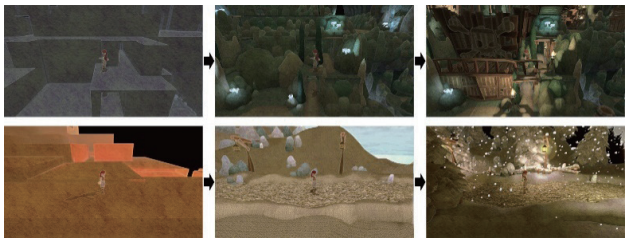

Figure 1: Environment Stimuli

For currently ongoing research, we decided to expand on that research by measuring the effect of level of detail in the immersion rates of users, as well as measure how well interactive environments are capable of conveying narrative. Like the prior research, this research also heavily depended on custom made environments, as once again we needed to increase the level of detail for each environment to be able to compare the results of each environment. We developed a total of 6 different core environments and subdivided this into 3 sets with increasing level of detail, giving us a total of 18 stimuli. Using a number premade assets packs and functionality [39] [40] [41] [42] [43] [44] [45] [46] [47] [48] [49] [50], we managed to reduce our total development time to 2 months for these 6 core environments (Figure 2). The increased number of stimuli meant that we could conduct more accurate experiments on the subject of immersion levels in digital interactive environments as well.

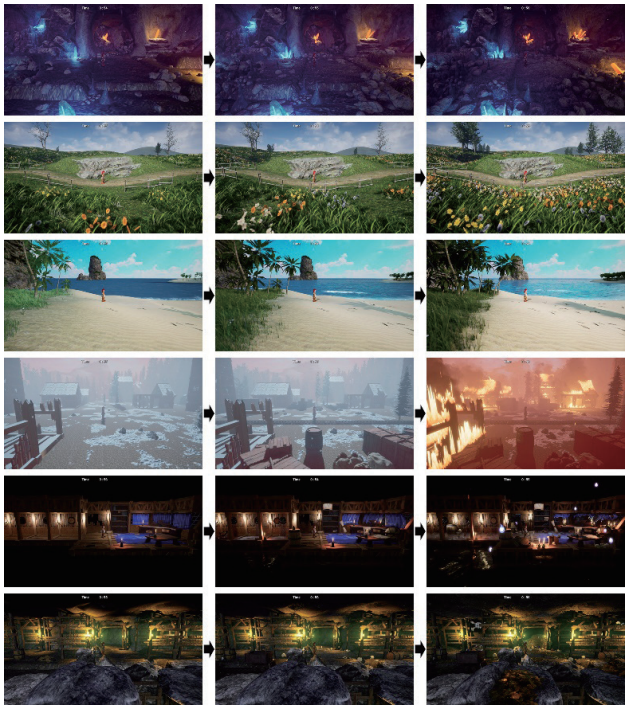

Figure 2: Environment Stimuli for continuation experiment

6 Visual Scripting Systems

The development of games requires a deep knowledge of programming, from the programming of game-play, to the programming of real-time graphics to render out the game to the screen. Especially for big budget game productions, this need for programming is big enough that programming is often divided into multiple specializations. This has meant up until now that unless a developer had programming skills, a developer would be unable to create digital games. While this barrier has not entirely been resolved yet, it has gotten more accessible through the use of a concept called "visual scripting."

Visual scripting is not an entirely new concept. The first experiments with Visual Scripting were already being conducted in 1960-1970, in the form of scripting languages like Pygmalion [51] and GRaIL (Graphical Input Language) [52]. However, it wasn' t until Microsoft developed Visual Basic in 1991 that visual scripting started to become more popular. While the current image of visual scripting differs quite a bit from programming using an IDE (Integrated Development Environment) that Visual Basic helped pioneer [53], Visual Basic was responsible for making programming more visual.

However, for the longest time, any other efforts to create visual scripting was largely constrained to the field of education. For instance, a visual programming language engineered at MIT with the name Scratch [54][55], was primarily meant to teach younger students how to think logically.

In recent years, various developers of game engines have started adding visual scripting features to their engines. A potential reason for this development could be the ever increasing scope and cost of game development. One calculation by Raph Koster states that from 1985 to 2005, the cost of games has multiplied by 22 times [56]. With the cost of development ever increasing, it would make sense for developers to introduce visual scripting tools so they could involve non-programmers in programming heavy development processes to stream-line production.

In 2007, Epic Games released the Unreal 3 engine, which introduced visual scripting in the form of two editors. One of the editors was called Kismet and the other was a material editor, one of the earliest examples of a shader development tool that made use of visual scripting. While Kismet was not exactly user friendly, it would eventually become the basis of a system called Blueprint in the Unreal 4 engine, a tool that would come to encompass a major part of the engine. Still, despite Kismet being a visual scripting tool, it was primarily meant to script simple occurrences and events in game environments, and wasn' t meant to be used to create entire games with. As such, programming was still a necessity for development.

Still, around this time, other engines that either added visual scripting solutions or were developed around a visual scripting tool also started to be released. The open-source Godot game engine [57] was released in 2014, includes visual scripting and is often lauded for its ease of use. Crytec incorporated a visual scripting tool by the name of Schematyc into their game engine by the name of CryEngine in 2016. Unity Technologies as well, acquired the developer of the Bolt visual scripting middleware, and incorporated the tool into their engine in 2021 [58]. Other software as well have started embracing visual scripting. The Substance Designer software allows for procedural texture generation creation using a visual scripting interface, and a variety of 3D software suits (like 3D Studio Max and Maya) now have visual scripting systems for shader creation. Houdini goes a step further, and has a visual scripting system for non-shader related functionality as well (like 3D model creation), and presents it as a selling point for their software [59].
In our own experience, we found that the existence of visual scripting systems helped us in 2 significant ways.

1. We were able to significantly speed up our prototyping process, by using a combination of the default templates present within the Unreal Engine, templates that take the form of a basic game. Due to this combination, we were able to get a game working in less than 2 hours of development time.

2. We were able to integrate questionnaires into the game environment with relative ease using the visual scripting tools present within the Unreal Engine (Figure 3). In our prior research [38], we had no way of integrating questionnaires into the game prototype, meaning that we had to interrupt participants of our research constantly by asking them to fill in questionnaires. The visual scripting system of Unreal Engine allowed us to easily create a user interface that recorded the user input. With testing methods integrated directly into the game content, not only did we not have to interrupt the participants anymore, additional testing venues like online testing also became more viable and secure.

With the rise of visual scripting systems and how robust they have become as development tools, it has become a lot easier even for non-programmers to make functional games.

Figure 3: Questionnaire conducted from within the game prototype itself. The questionnaire system was entirely developed using "blueprint," Unreal Engine's visual scripting system.

7. Conclusion

Game development is a uniquely time consuming and costly activity, requiring an in-depth knowledge of a wide variety of developments fields, including but not limited to

programming, visual design and interaction design. As a game researcher, this meant that one had to spend a long time developing the material required for the research, rely on hired help, or instead analyze games that has already been developed, all options that carried significant drawbacks.

In recent years however the situation has markedly improved through 4 key developments; game engines and software required for game development have become more financially accessible, the accessible knowledge base on game development has dramatically increased, a market has formed around middleware and assets specifically meant for speeding up game development and finally, the increasing popularity of visual scripting has made it so that even non-programmers are now capable of creating functional games.

In our own experience, we have found that these four developments have had a tremendous positive impact on our game research. Through the decreased required development time, and the increased options for development, we were not only able to develop much more content in a much smaller timeframe, but we were able to create additional functionality that allowed us to conduct questionnaires and other testing from within the game environment itself.

Taking all of this into account, we conclude that research options for game research has expanded dramatically, which is particularly positive for the field of game design methodology research, which before was relatively hard and time consuming to do.

References

[1] Mata Haggis-Burridge et al, et al, ridge et al, nt, we conclude thHorizon 2020 Research and Innovation Programme – Grant Agreement No 732332 (2018)

[2] VPRO: t Agreement No 732332No ation Programme – G"V(Accessed 2021-07-10)

[3] Carlos Gonzalez-Ochoa: arlos Gonzalez-Ochoa:Uncharted 4"ncharted 42016 Advances in Real-Time Rendering in Games

[4] Michael Brandse: "The Shape of Challenge - Using Affordance Design to Create Challenge within Games" Human Computer Interaction International (2017)

[5] David Shafer, Robert Yang: "Level Design Workshop: Invisible Intuition: Blockmesh and Lighting Tips to Guide Players and Set the Mood"Game Developers Conference (2018) https://www.gdcvault.com/play/1025360/Level-Design-Workshop-Invisible-Intuition (Accessed 2021-07-13)

[6] Phil Co: iLevel Design for Gamesom/New Riders (2006).

[7] Tekinbaş, Katie Salen, and Eric Zimmerman: mRules of play: game design fundamentals.ules of play: game design fun

[8] Jason Schreier, hBlood, sweat, and pixels: the triumphant, turbulent stories behind how video games are madele-Intuition (Accessed 202, pp 54.

[9] Constantino Oliva: "The Musical Ludo Mix of Taiko no Tatsujin" Transactions of the Digital Games Research Association (2021), Vol. 5 No 2, pp. 131-160

[10] Dalila Forni: "Horizon Zero Dawn: The Educational Influence of Video Games in Counteracting Gender Stereotypes" Transactions of the Digital Games Research Association (2019), Vol. 5 No. 1, pp. 77-105.

[11] Electronic Arts: "Frostbite Engine – the most adopted platform for game development" https://www.ea.com/frostbite (Accessed 2021-07-20)

[12] ComicBook: "Square Enix Shows Off Incredible, Cutting-Edge Tech Demo" https://comicbook.com/gaming/news/square-enix-back-stage-tech-demo-luminous-engine/ (Accessed 2021-07-20)

[13] Eurogamer: "The Engine Licensing Game - The ups and downs of licensing a game engine" https://www.eurogamer.net/articles/engines (Accessed 2021-07-20)

[14] 3DVIA: "3DVIA Virtools – Demo Showcase" https://www.3ds.com/fileadmin/PRODUCTS/3DVIA/3DVIAVirtools/demoshowcase/html/index.html (Accessed 2021-07-20)

[15] Unity Technologies: "Unity Real-Time Development Platform ¦ 3D, 2D, VR/AR Engine" https://unity.com/ (Accessed 2021-07-20)

[16] TechCrunch: "How Unity built the world' s most popular game engine" https://techcrunch.com/2019/10/17/how-unity-built-the-worlds-most-popular-game-engine/ (Accessed 2021-07-20)

[17] Unity Technologies: "A Free Unity?"https://blog.unity.com/technology/a-free-unity (Accessed 2021-07-20)

[18] Unity Technologies: "Evolution of our products and pricing" https://blog.unity.com/community/evolution-of-our-products-and-pricing (Accessed 2021-07-20)

[19] IGN: "Epic Games Announces Unreal Development Kit, Powered by Unreal Engine 3"https://www.ign.com/articles/2009/11/05/epic-games-announces-unreal-development-kit-powered-by-unreal-engine-3 (Accessed 2021-07-20)

[20] Epic Games: "UDK Gears Mod Home" https://docs.unrealengine.com/udk/Three/GearsModHome.html (Accessed 2021-07-20)

[21] Epic Games: "UDK Licensing Resources" https://www.unrealengine.com/en-US/previous-versions/udk-licensing-resources (Accessed 2021-07-20)

[22] Engadget: "Epic Games' Unreal Engine 4 now available by subscription for $19, headed to OS X and more" https://www.engadget.com/2014-03-19-unreal-engine-4-available.html (Accessed 2021-07-20)

[23] Epic Games: "If You Love Something, Set It Free" https://www.unrealengine.com/en-US/blog/ue4-is-free (Accessed 2021-07-20)

[24] Epic Games: "Unreal Engine 5" https://www.unrealengine.com/en-US/unreal-engine-5 (Accessed 2021-07-20)

[25] Crytec: "CryEngine ¦ the Complete Solution for next generation game development by CryTec" https://www.cryengine.com/ (Accessed 2021-07-20)

[26] BitTech: "Quake 3 source code released" https://bit-tech.net/news/gaming/quake3_source_code/1/ (Accessed 2021-07-20)

[27] PluralSight: "Technology Skills for Business" https://www.pluralsight.com/ (Accessed 2021-07-27)

[28] Udemy: "Online Courses – Learn Anything, on Your Schedule" https://www.udemy.com/ (Accessed 2021-07-27)

[29] Unity Technologies: "The Most Popular Event for Game, Animation & Industrial Developers ¦ AR/VR Conference"https://unity.com/events/unite (Accessed

2021-07-27)
[30] Epic Games: “Unreal Fest Europe 2020” https://www.unrealengine.com/en-US/events/unreal-fest-europe-2020 (Accessed 2021-07-27)
[31] Epic Games: “Calling DLLs from UnrealScript (DLLBind)” https://docs.unrealengine.com/udk/Three/DLLBind.html (accessed 2021-08-03)
[32] Youtube: “UE4 / Unreal Engine 4 / Saving to Text File (CSV) C++” https://www.youtube.com/watch?v=uZPzTN5Debc (Accessed 2021-08-03)
[33] Marmoset: “Physically-based Rendering, and you can too!” https://marmoset.co/posts/physically-based-rendering-and-you-can-too/ (Accessed 2021-07-30)
[34] Havok: “Havok Physics” https://www.havok.com/havok-physics/ (Accessed 2021-07-30)
[35] Interactive Data Visualization: “3D Vegetation Modelling and Middleware” https://store.speedtree.com/ (Accessed 2021-07-30)
[36] Crunchbase: “Everything you need for the production and automatic optimization of 3D content” https://instalod.com/ (Accessed 2021-07-30)
[37] Quixel: “Bridge is now a part of Unreal Engine 5 Early Access” https://quixel.com/blog/2021/5/26/bridge-is-now-a-part-of-unreal-engine-5-early-access (Accessed 2021-07-30)
[38] Michael Brandse, Kiyoshi Tomimatsu: “Immersion Levels in Digital Interactive Environments” Kansei Engineering and Emotion Research International (2014)
[39] Unreal Engine Marketplace: “Brushify - Tropical Pack” https://www.unrealengine.com/marketplace/en-US/product/brushify-tropical-pack (Accessed 2020-7-30)
[40] Unreal Engine Marketplace: “Brushify - Arctic Pack” https://www.unrealengine.com/marketplace/en-US/product/brushify-arctic-pack (Accessed 2020-7-30)
[41] Unreal Engine Marketplace: “Crystal Mines - Scene and Assets” https://www.unrealengine.com/marketplace/en-US/product/crystal-mines-scene-and-assets (Accessed 2020-7-30)
[42] Unreal Engine Marketplace: “Environment Set” https://www.unrealengine.com/marketplace/en-US/product/environment-set (Accessed 2020-7-30)
[43] Unreal Engine Marketplace: “Flowers and Plants Nature Pack” https://www.unrealengine.com/marketplace/en-US/product/flowers-and-plants-nature-pack (Accessed 2020-7-30)
[44] Unreal Engine Marketplace: “Infinity Blade: Grass Lands” https://www.unrealengine.com/marketplace/en-US/product/infinity-blade-plain-lands (Accessed 2020-7-30)
[45] Unreal Engine Marketplace: “Low Poly Snow Forest” https://www.unrealengine.com/marketplace/en-US/product/low-poly-snow-forest (Accessed 2020-7-30)
[46] Unreal Engine Marketplace: “Meadow - Environment Set” https://www.unrealengine.com/marketplace/en-US/product/meadow-environment-set (Accessed 2020-7-30)
[47] Unreal Engine Marketplace: “Old Mine Tunnel & Caves” https://www.unrealengine.com/marketplace/en-US/product/old-mine-tunnel-caves (Accessed 2020-7-30)
[48] Unreal Engine Marketplace: “Spring Landscape” https://www.unrealengine.com/marketplace/en-US/product/spring-landscape (Accessed 2020-7-30)
[49] Unreal Engine Marketplace: “Stone Boulders” https://www.unrealengine.com/marketplace/en-US/product/stone-boulders (Accessed 2020-7-30)
[50] Unreal Engine Marketplace: “SHADERSOURCE - Tropical Ocean Tool” https://www.unrealengine.com/marketplace/en-US/product/beach-wave-water (Accessed 2020-7-30)
[51] David Canfield-Smith: “Pygmalion: A Creative Programming Environment.” Unpublished doctoral dissertation (1975)
[52] T.O. Ellis, J. F. Heafner, W. L. Sibley: “The Grail Project: An Experiment in Man-Machine Communications.” Santa Monica, California, RAND Corporation (1969)
[53] Mendix: “The History of Visual Development Environments: Imagine There’s no IDEs. It’s Difficult if You Try.” https://www.mendix.com/blog/the-history-of-visual-development-environments-imagine-theres-no-ides-its-difficult-if-you-try/ (Accessed 2020-7-30)
[54] Bubble: “A History of Visual Programming: From Basic to Bubble” https://bubble.io/blog/visual-programming/ (Accessed 2020-7-30)
[55] MIT: “Create stories, games, and animations Share with others around the world” https://scratch.mit.edu/ (Accessed 2020-7-30)
[56] Raph Koster: “The cost of games” https://www.raphkoster.com/2018/01/17/the-cost-of-games/ (Accessed 2020-7-30)
[57] Godot Engine: “Free Open Source 2D and 3D Game Engine” https://godotengine.org/ (Accessed 2020-7-30)
[58] Unity Technologies: “Bolt visual scripting is now included in all Unity plans” https://blog.unity.com/news/bolt-visual-scripting-is-now-included-in-all-unity-plans (Accessed 2020-7-30)
[59] Houdini: “3D Procedural Software for Film, TV & Gamedev” https://www.sidefx.com/products/houdini/ (Accessed 2020-7-30)