Producing with or without Game Engines

Vincent Broeren

vincent@jagaco.com

Utrecht University
Department of Information and Computing Sciences

Abstract. The decision of producing with or without an engine can have major effects on the game, team and company. Large risks can be made with large rewards afterwards, or game plans can fail miserably by making the wrong choice. In this paper first of all a definition of the term “engine” is deducted. The literature study unfortunately proves that very little scientific research is done in the field of game production and no resource gives a conclusive set of rules regarding the main question. The influential factors regarding the choice are investigated and discussed after which they are reflected on some of the largest and well known game engines. The hypothesis created by the information found during the literature study is tested in a (small) case study. In the case study two companies are questioned about their choices regarding building games with or without an existing engine.
Throughout the whole study a new issue is found, there must be a turning point from which on a company is mature enough to try to build their own engine, because starting developers/companies will have a hard time to try it from the beginning.

1 Introduction

In order to maintain a competitive edge in the video game industry, it is important to continually research new video game techniques and technologies. It is not enough to simply be aware of emerging techniques and technologies but also to make informed decisions about which of these techniques are appropriate given a particular project's requirements and limitations.

Definition of terms
There is no strict definition of the term “engine” or “game engine” related to game development, several sources used different definitions. In this paper the term is intended as a reusable, robust software system used by the game and providing the core technical solutions for the game that is executed at *runtime* while running the game code.

The tooling is not seen as part of the engine itself, because at runtime that code will not be executed as a whole. Tooling does have rooted code in the engine itself and influences the engine in that way, but only for a part of the code and is therefore excluded from the definition.

Objective
In this paper I will consider several factors that influence the choices regarding technologies to be applied in a video game project. These choices include whether the project should use existing game engine technology or whether the game engine technology should be developed in-house. A third option is investigated, building a game without an engine at all. I will try to find out how often larger (triple-A) games are produced without the use of separate engine code, building the game from scratch and tailoring everything build towards the game itself without the effort to make the code reusable for other projects.

One factor not researched is the middleware usage within game companies. In the broader sense of the term, game engines themselves can be described as middleware. But in the context of games the term middleware is often used to refer to subsystems of functionality within a game engine. Some game middleware does only one thing but does it more convincingly or more efficiently than general purpose middleware [1]. Lots of engines have middleware build into them, examples of well know middleware are: HavokPhysics [2] for the physics simulation within the engine, Bink [3] a video codec for videos in games or Speedtree [4] for the generation and rendering of realistic trees in games.

The game engine interfaces the different kinds of middleware so it is also possible to interchange them later on. All choices considering middleware in games production are not researched. Most engines will have middleware build in them, but the engine is treated as a whole in this paper.

This paper focusses on the production of games with or without game engines by larger companies (from A to triple-A companies). The smaller companies/projects are left out of the scope of this paper.

Keywords: Game production, Game engine, Technology choice.

2 Literature study

The first and easiest step to take is to google for some keywords. This is not part of the literature study, but will give an indication of how much is out there about the subject. The first indication was not promising, not much information was found in this way.

Scientific resources
The second step is to repeat the search in scientific databases. I have searched in: *ACM Digital Library*, *Espacenet*, *IEEE Xplore Digital Library*, *"Library and Information Science Abstracts" (LISA)* and the *"Library, Information Science & Technology Abstracts" (LISTA)*. I chose these libraries because I have access to them through my work and because I can search through them simultaneously with a tool. Again alarmingly few results showed up. I found a couple of papers (about four) which sounded interesting, but after downloading them and reading them they didn't cover the question of producing with or without game engines. Almost all papers are about choices to make within the engine itself, but none about the engine as a whole.
There was one paper (found through a reference in the lecture about *reuse*) called *Improving Digital Game Development with Software Product Lines* [24] which introduces Domain-Specific Languages (DSL) for computer games incorporated with Software Product Lines (SPL's). Which can be seen as a high level abstract scripting language build on top of a game engine.

Books
A third source of information are books. And I found a couple of books about game production [5][7] and game engine architecture [10][11][12]. But again none of the books give enough insight to base a choice upon.
The Game Production Handbook [5] is all about the business side of game development and describes decisions needed to be made regarding the technology of the game/engine/tooling, but fails to investigate further into this choice.
Game Engine Architecture [10] is a great book about the internals of a game engine. It gives insights in the architecture behind an engine and gives a lot of pointers in how to build your own. Although many pitfalls of building your own engine are discussed in the book, the actual decision to build your own engine is stepped over. Mainly because the book is about the building of the engine.
Game Engine Gems 1 & 2 [11][12] are in line with the *Game Engine Architecture* book and are a collection of small articles all about game engines. The first book in the series does talk about decisions to make regarding middleware but none about the preliminary choices of using an engine or building it.

Internet resources
There are some websites specialized in game development, and luckily there was some information found on them [6][9].
On the gamedev.net forum one article is famous: "*Write Games, Not Engines*" [6]. This article mainly promotes the making of games instead of a focus on engines. In the quest lectures it was also mentioned that when you decide to make an engine yourself "*you must really know what you are doing*". This is true for beginner to intermediate programmers. Also because they fail to see the option of making games without separate engine code at all. This is also stated in the article. What is interesting to investigate is the turning point of a company when it is mature enough to start thinking of making its own engine. One major part of this maturity level is knowing internally and having experience with existing engines.

Conclusion
During the literature study it became apparent that not much (virtually none) scientific research has been done on this subject. This makes the subject even more interesting from the scientific point of view, but makes it much harder to draw conclusions in this paper.

Preliminary hypotheses
The literature only skims the main question of building with or without a game engine, but it is possible to construct hypotheses from it.
Books about building engines are (logically) biased towards building your own engine [10][11][12]. Only on the internet are opinions found that building your own engine is not always the way to go. This is stated in relation to small companies or students/people beginning with game programming and lacking any experience in the field [6][14]. This is the first hypothesis; bigger companies build their own engines and smaller companies or startups use an existing engine. This hypothesis can be backed up by the fact that building a whole engine from scratch is a project on its own. With an extra project the risks are also larger.
Another hypothesis based on the overview of engines and game companies on the internet [1][13] is that triple A studio's will always separate engine code from game code. This is done because it is too expansive not to. By separating the code the engine part can be reused by sequels or other games made by the company. If this is true, which must be checked in the case study, negates the third option of building larger games with no separate engine code.
The shelf life of a state-of-the-art game engine is short, because the underlying technology is evolving at such a rapid pace. The engine can be seen as the software layer on top of the hardware, and the hardware keeps getting better and better. Not only game consoles as a whole like the PlayStation 4 or the Xbox One, but also for example the graphics cards in PC's [10][13]. Because of this evolution within the industry it's interesting to see how the main question evolves over time. Will it become increasingly more difficult to build your own engine from scratch? This last hypothesis cannot be investigated in this paper because of time constraints and future work will need to address this one.

3 Influential factors

The production of a game starts with a pre-production phase [5]. One of the things to investigate in this phase is the technology to use during the creation of the game. In this phase the human resources are also evaluated and as we will see later, this evaluation can be linked to the choice of technology to use. The choice made in this phase is leading in the path to follow in all other phases of the game production. In the pre-production phase extensive research should be done to base the choice upon. In this chapter the influential factors on this choice will be elaborated on.

Project Management Triangle
All the factors influencing the choice of technology lay within the project management triangle (also referred to as the iron triangle) [21][22][23].

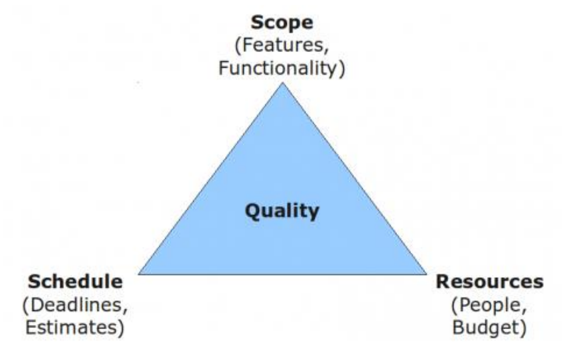

Figure 1 – Project management triangle

The choices lay in this triangle because virtually all operational choices have their place in the triangle. The choice to use or build an engine for the game influences all three corners of the triangle [22].
When a game developing company makes the choice to build their own engine for a game the **scope** of the project broadens because features specific for the engine needs to be added to the planning. Examples of large engine specific features are interfaces, tooling and the asset pipeline [10].
Interfacing in the engine is used to make it more reusable and extendable. This could be left out when building a game with an existing engine or no engine at all.
Tooling consists of all engine related software not directly part of the engine itself or the game codebase. Examples are the level editor, automated build scripts or sound recoding (not recording) programs. These tools are specifically build to tailor the needs of the user and are specific to the game technology.
The asset pipeline is one of the largest features with parts of it integrated in the engine and even in the game. The asset pipeline is the whole management of the 'raw' resources created by the artists in the Digital Content Creation (DCC) programs. Intermediate files are used to translate (multiple) resources into assets. Assets are optimized for speed and can be used directly in the game itself at runtime. This whole process is a large part of an engine and using an existing engine always provides this management chain [10][12].
The **resource** corner in the triangle also changes because people with specific and often low level programming knowledge must be hired to be able to build an engine in-house. Often a project team is formed to work only on the engine code base and related tooling. This team supports the game development team, because all needs for the engine features will come from the game development team [5].

A popular saying to describe what managers think is: *"When one woman takes nine months to deliver a baby, then nine women can do it in one month"*. Related to the above resource growth and setting up a separate engine development team will inevitably mean the **schedule** will expand and deadlines will take longer to achieve. Building the engine and the game at the same time created dependencies between the two which leads to a more difficult planning.

The factors

Taking into regard all literature found about the subject and deducing the information some common factors emerge. The factors summarized below are the ones that are going to be used to evaluate all three options of building a game with an existing engine, building a game and the engine itself and building a game without separating the engine all together. The listed factors below are abstracted terms found throughout the literature but mostly stated in *The Game Production Handbook* [5] and *Game Development Essentials: Game Project Management* [8]. If a term were found in more than four different sources it is stated in the list. Many more factors were found in the literature, but not in enough separate sources to be incorporated in the list.

- Financial factors
- Scope of the project
- Theme and Audience of the project
- Target platform(s)
- Skill-set of available human resources
- Time constraints

Financial factors, determining the financial reach of the company (whether or not backed up by a publisher) will probably be the most influential factor. What is the company prepared to invest? You can invest $1,000,000 in an existing engine or invest many more in the time and personnel to build your own with the possibility of making a lot of extra money monetizing the new engine.
Scope of the project, and ambition of the team/company. Actually building your own engine can be a goal to begin with. Some larger companies did make this decision and succeeded. Examples follow in the next chapter.
Theme and Audience of the project, does the game concept fit an existing engine? A large example is the Grand Theft Auto series. There was no engine that supported an open sandbox world of this foreseen size, so Rockstar made an engine to make that possible. This last statement is reflected in the preliminary conclusions.
Target platforms, what are the platforms the game is targeted for? And is there an engine that covers all those platforms?
Skill-set of available human resources, what can be achieved with the current personnel and is the ambition of the company covered with the knowledge of the current personnel?
Time constraints, as stated before, making your own engine greatly influences the timeframe of the project. Is that a luxury the company has? And is it a risk they want to take?

With the stated factors in mind it is interesting to zoom into some of the larger and well known commercially available game engines. By looking at the details some distinguishing features arise which also needed to be taken into account when making the choice of building a game with an engine.

4 Engine overview

There are many engines out there and to get an indication of the scope of the field some high level overview is given in this chapter.
It is impossible to get a full list of game engines, but a list of the larger and more known engines is possible to make [13][14].
At this moment there are at least:

- 80 larger open-source engines
- 113 proprietary engines

One of the things that becomes apparent is that most engines are written in C/C++ as the main programming language. The used scripting language of the engine differentiates more. There are other programming languages used, but those are more of an exception.
C++ is an older language created in 1979 by Bjarne Stroustrup [15] but is still the main language of choice because of the speed that native code gives over the higher level programming languages. In 2011 the language got the latest large update with the release of the C++11 standard (preceded by the C99 standard from 1999).

Detailed information
Because companies want to make a difference with the technology they build a large variety of engines with all specific selling points exists. It is interesting to see some of those differences, because it gives insight in the technology behind the main question and can even influence the decision towards the main question.
The five larger and more known engines are:

- Unreal Engine (Epic) [16]
- CryENGINE (Crytek) [17]
- Frostbite (EA) [18]
- Source (Valve) [19]
- Unity (Unity3D) [20]

These engines are chosen because of the wide range of requirements they span and because they are used by many games.
All of these engines have some distinguishing features. Unity is biased towards indie developers because of the low costs (around $1,500.-), while the Unreal Engine and CryENGINE are on the top of the list regarding graphical performance. Being at the top means the prizes are also very steep. The Unreal engine is rumored to cost

$1,000,000.- for a full license and is ported towards the most different gaming platforms, from the WindowsPhone to the PlayStation4. The CryENGINE is rumored to cost $1,200,000.- fully licensed and is considered the most advanced engine currently available in the world. The Frostbite engine is an example of an engine only available to game developers affiliated to Electronic Arts (EA). Sony is using the same strategy and makes their engine only available to their own game development studies like Guerilla and Naughty Dog.
What is interesting is that the very high license costs can be a huge hurdle for small companies or indie developers. But during the Game Developers Conference this year, held in San Francisco, [32] both Crytek and Epic announced a major change in their licensing structure. Epic dropped the costs of the Unreal Engine from $1M to $19/month + 5% royalties. This is a cost per seat, meaning that for all personnel using the engine the $19.- must be paid, but still this is a huge drop in costs. Crytek followed dropping the prize from $1.2M to only $10.-/month with no royalties needed. The difference is that Epic also gives the paying developers full excess to the source of the engine, meaning they can change the engine to their liking. Crytek only gives access to the game code part of the engine.
These changes in costs will have an effect on future choices regarding using a cutting edge engine which has been tested in the field to the high risk of making your own engine.

5 Case study

In the case study I wanted to survey three game developers. One small one (<20 developers), one medium company (20-50 developers) and a large one (>50 developers). By getting the three different sizes I could compare the results and draw conclusions between the results and the size of the company.
I chose a survey instead of for example a semi structures interview, so I could compare the answers more easily, but the main reason is that the companies don’t have much time to spend on long interviews, especially the larger companies. Because of the last reason I kept the questions in the survey to a bare minimum of ten questions, so the participant wouldn’t take too long to answer the questions.

Below the questions in the survey are found. They all are about the choice of building an engine and related tooling or middleware use.

1. Is an existing engine used or was a custom one build?
2. Does the engine use middleware (lib's build into the engine)? If so, which ones?
3. Which (support) tools are built besides the core of the engine?
4. Which tool is most often used?
5. Is there a clear separation between engine and game code in your games?
6. What where the main reasons to build a custom engine?
7. Are there plans to monetize the engine?
8. Are you affiliated to a publisher?
9. On which platforms are the games released?
10. In hindsight do you think building a custom engine was worth the effort? If so, why?

Figure 2 - Questions in the survey

Unfortunately I could only contact two companies both in the 'small' range of number of developers. This is mainly because of the time constraints to get into contact with the companies and I started later, because in the beginning we were pursuing a contact (at Nexxus) of my teammate, which during the study quit because of personal reasons. The companies that I could contact are described below.

GB
GB is located in The Netherlands and is the creator of a successful game. The company consists of 17 people working directly on the game or related technologies. My contact was LH, which is a developer at GB. He is working on the game itself and not the engine.
GB has built their own engine because "*its task is very specific for our game, you can't buy that anywhere. It also changes quite frequently*." [27]. They also have built their own tooling used besides the engine itself. The tooling is used to maintain the data used by the game.

JG
"JG is an independent Dutch game developer, dedicated to bring fun experiences to the masses. With a focus on mobile platforms we try to deliver a variety of games, from revamped classics to completely original gameplay."[26].
JG is located in The Netherlands and is the creator of several mobile games, which are planned to be released later this year. The company consists of four people working directly on the game or related technologies.
JG also chose to build their own engine. The main reason is "*that we have the knowhow. Apart from that we feel that this gives us more control over how everything turns out. There is a considerable investment involved initially but we feel we can make up for this in future titles.*" [28]. JG has an extensive tool chain built around the engine. The

tools are used by the artists and game designers to quickly build levels and prototype GUI elements.

Results

The first preliminary result was that only two surveys didn't give enough detailed information from both companies to be able to draw conclusions from it. For that reason I chose to conduct a small semi-structured interview with both my contacts to elaborate some more on some of the given answers in the survey. With LH from GB I've done the interview over the phone, because we both didn't have time to meet in person. Because the interview was small, doing it over the phone gave no problems. RB from JG was interviewed in person, because I see him every week. The interview with RB confirmed some of my assumptions made from his answers in the survey.
The fact that I only have data from two small companies influences the results. No correlation can be found between choices and the size of a company.
What is important to notice is that both small companies did build their own engine for different reasons, this contradicts the preliminary hypothesis that smaller companies don't build their own engine to prevent the bigger risks.
Both companies use middleware in their own engine. Ranging from whole frameworks like MonoGame [29] to PushSharp to communicate between a server and mobile devices [30]. The literature study also uncovered that large engines use a lot of middleware [1].
Also both companies are self-publishing, meaning they are in no way affiliated with a publisher and so have the freedom to make their own choices regarding the development of their games and technology.
Again both companies are happy in hindsight about the choice to build their own engine. This means the choice that was made in the past still works well for them.

6 Conclusion

Because the case study only covered two small companies caution is needed towards drawing conclusions on only these two results. With this in mind some interesting things can be noted though.

"*Triple-A studio's will always separate engine code from game code*" cannot be answered by the case study because only small companies were surveyed, but apparently not only the big companies separate game code from engine code in favor of the reuse, also the smaller companies do that. Even if in the case of GB there is no urge to monetize the engine on its own.

"*Bigger companies build their own engines and smaller companies or startups use an existing engine*" is not backed up by the case study, both companies are small and both build their own engine for different reasons. During this case study the size of the company showed no effect on the choice to build their own engine or not.

While stated that the middleware was not part of this research, it became apparent during the survey and interviews that middleware is used a lot instead of whole engines. In this way the companies can be more flexible in their solutions without the need to build everything from scratch.
The influential factors deducted from the literature which thought to be the main list of reasons to base the main choice upon where not mentioned at all by both companies (even in the interviews). Apparently there are more factors which maybe are more personal that also take a role in making the decision.
Software Product Lines as mentioned in the literature study might be a solution in conjunction with an engine, but maybe only at larger companies, which have multiple game titles in development at the same time.

7 Future work

There is certainly more research possible in the field of game development. The amount of research done in the field shows how immature the industry still is. More research will help the industry to mature faster.
Concerning the main question of producing a game with or without a game engine also much more research is advised. Although it should be noted that it appears to be the case that the choice isn't fully made rationally, but also personal preferences take part in the decision.
It is advised to conduct a larger survey with more companies of different sizes to be able to make a better comparison between the size of the company and the choice to product with an own engine.
Middleware should be included in future research. It might be an intermediate way of reuse conducted in the game industry (instead of reusing the whole engine).
Is there a turning point from which on it's more logical to build your own engine instead of using an existing one? This is a very interesting question which should be answered in future work.
The last hypothesis "*Because of the rapid evolution within the industry it's interesting to see how the main question evolves over time*" needs much more time and data to see if it is true. If so it could be a thread to the industry, meaning the development of ever larger games becomes a larger and larger undertaking in the future with all the risks coming with it.

References

1. *Game Engine wikipedia.* Available at: http://en.wikipedia.org/wiki/Game_engine#Game_middleware (Online; accessed: March 8th, 2014).

2. *Havok Physics.* Available at: http://www.havok.com/products/physics (Online; accessed: March 8th, 2014).

3. *Bink Video - The Video Codec for Games.* Available at: http://www.radgametools.com/bnkmain.htm (Online; accessed: March 8th, 2014).

4. *Speed Tree.* Available at: http://www.speedtree.com/ (Online; accessed: March 8th, 2014).

5. Chandler, H. (2013). *The Game Production Handbook* (3rd edition), Jones and Bartlett Publishers.

6. Josh Petrie (2011) *Write Games, Not Engines*. Available at: http://scientificninja.com/blog/write-games-not-engines (Online; accessed: March 3rd, 2014).

7. Irish, D. (2005). *The Game Producer's Handbook* (1st edition), Cengage Learning.

8. Novak, J. & Hight, J. (2007). *Game Development Essentials: Game Project Management* (3rd edition), Cengage Learning.

9. Cifaldi, F.: *E3 report: The path to creating AAA games*. Available at: http://www.gamasutra.com/view/feature/130722/e3_report_the_path_to_creating_.php (May 2005), (Online; accessed March 4th, 2014).

10. Gregory, J. (2009). *Game Engine Architecture* (1st edition), A K Peters/CRC Press.

11. Lengyel, E. (2010). *Game Engine Gems 1* (1st edition), Jones & Bartlett Publishers.

12. Lengyel, E. (2011). *Game Engine Gems 2* (1st edition), Jones & Bartlett Publishers.

13. *List of game engines*. Available at: http://en.wikipedia.org/wiki/List_of_game_engines (Online; accessed: March 10th, 2014).

14. *GameDev.net.* Available at: http://www.gamedev.net/page/index.html (Online; accessed: March 10th, 2014).

15. *The C++ programming language*. Available at: http://isocpp.org/ (Online; accessed: March 16th, 2014).

16. *Epic - Unreal Engine.* Available at: http://www.unrealengine.com/unreal_engine_4/ (Online; accessed: March 14th, 2014).

17. *Crytek – CryEngine*. Available at: http://www.crytek.com/cryengine/cryengine3/overview (Online; accessed: March 14th, 2014).

18. *EA Digital Illusions CE (DICE) – Frostbite engine*. Available at: http://www.frostbite.com/ (Online; accessed: March 14th, 2014).

19. *Valve – Source*. Available at: http://source.valvesoftware.com/ (Online; accessed: March 14th, 2014).

20. *Unity3D – Unity*. Available at: https://unity3d.com/ (Online; accessed: March 14th, 2014).

21. *Project Management Triangle*. Available at: http://en.wikipedia.org/wiki/Project_management_triangle/ (Online; accessed: April 10th, 2014).

22. *Every Project plan is a triangle*. Available at: http://office.microsoft.com/en-001/project-help/every-project-plan-is-a-triangle-HA001021180.aspx/ (Online; accessed: April 10th, 2014).

23. *Cost, Time, Scope in Project Management*. Available at: http://www.consultants-on-line.com/default.asp?contentID=88/ (Online; accessed: April 12th, 2014).

24. Furtado, A. et al (2011) Improving Digital Game Development With Software Product Lines, IEEE Software 28(5): 30-37

25. *... [Anonymized]*

26. *JG website*. Available at: http://www.jagaco.com/ (Online; accessed: March 12th, 2014)

27. Broeren, V. (April 2014) "*GB used technologies*" Survey.

28. Broeren, V. (April 2014) "*JG used technologies*" Survey.

29. *MonoGame*. Write once, play everywhere. Available at: http://www.monogame.net/ (Online; accessed: April 14th, 2014)

30. *PushSharp. A server-side library for sending Push Notifications to mobile devices.* Available at: https://github.com/Redth/PushSharp/ (Online; accessed: April 14th, 2014)

31. *Engines of Creation: An Overview of Game Engines*. Available at: http://www.gamasutra.com/view/feature/132226/engines_of_creation_an_overview_.php (Online; accessed: April 14th, 2014)

32. *Game Developers Conference*. Available at: http://www.gdconf.com/ (Online; accessed: April 2nd, 2014)