Open access • Proceedings Article • DOI:10.1145/1496984.1497031

The case for research in game engine architecture — Source link

Eike Falk Anderson, Steffen Engel, Peter Comninos, Leigh McLoughlin

Institutions: Coventry University, Bournemouth University

Published on: 03 Nov 2008 - Conference on Future Play

Topics: Game design document, Game design, Game Developer, Game art design and Game testing

The Case for Research in Game Engine Architecture

Eike Falk Anderson[*] Steffen Engel[†] Leigh McLoughlin[‡] Peter Comninos[§]
Interactive Worlds ARG, Coventry University, UK
The National Centre for Computer Animation, Bournemouth University, UK

Abstract

This paper is a call for research in the field of game engine architecture and design, a more comprehensive and thorough understanding of which we consider to be essential for its development. We present a number of key aspects that may help to define the problem space and provide a catalogue of questions that we believe identify areas of interest for future investigation.

1 Introduction

Starting with Spacewar in 1961 [Fleming 2007], computer games have been around for just over four and a half decades. Until about two decades ago, most games were developed by a small team of programmers, if not by individuals, usually written from scratch with very few reusable components. In a young industry that is very much project-based, this does not come as a great surprise; however, as the industry has grown, so has the size of individual projects. This change has necessitated a number of advancements in production techniques, and Garlan and Shaw [Garlan and Shaw 1994] recognised that "one characterization of progress in programming languages and tools has been regular increases in abstraction level – or the conceptual size of software designers building blocks". A similar increase in abstraction level occurred within computer game development towards the end of the 1980s, when the first systems with reusable components appeared. These components are what we would now call game engines.

There appears to be general agreement that game engines are not only useful, but due to the complexity of modern computer games, are actually required for game development. Given this, there exists a surprisingly small body of literature on game engine design. The available research has mainly focussed on game engine subsystems, such as rendering, AI (artificial intelligence) or networking. However, issues regarding the overall architecture of engines, which connects these subsystems, have merely been brushed over.

This lack of literature and research regarding game engine architectures is perplexing. Books on the subject, such as that by Eberly [Eberly 2000], tend to only briefly describe the high-level architecture before plunging straight down to the lowest level and describing how the individual components of the engine are implemented. Often authors present their own architecture as a de facto solution to their specific problem set, without necessarily justifying the decision-making processes that led to their designs, and merely indicating the authors' personal preferences. Such literature offers an excellent source of information for *writing* an engine, but provides little assistance for *designing* one when the requirements are different from the solution described.

We believe that there are a number of points that should be investigated, with regard to the game engine development process. In this paper, we are not trying to answer the questions that need answers, but we would like to invite the academic community as well as the game industry to participate in the discovery of these answers. The aim of this position paper is therefore to present these questions in order to provide a starting point for debate, and to encourage discussion of game engine development and architecture.

2 Research Questions

In this section we present a number of questions, focussed on a range of potential research topics within game and game engine development. The questions that we pose can roughly be categorised into questions of terminology and questions of game engine architecture and design. This is in no way intended to provide an exhaustive list, or even to examine the points in all their detail. Instead it is our hope that we can spark discussions on these and similar topics, and to trigger a more rigorous examination of their effects.

2.1 Terminology: The lack of a 'game development' language.

The games industry is relatively young, and as such it is still lacking a common terminology. This is true within the games industry itself, and between the industry and the emerging games-related academia. However, in order to facilitate the rigorous study of any subject, from both a purely academic and a practical point of view, the ability to communicate effectively is of the utmost importance.

The absence of a formal terminology results in communication at cross purposes, as lamented by Stephens [Stephens 2001], who points out the confusion between 'game engines' and 'rendering engines', clearly demonstrating the need to establish a game development language.

We believe that such a language should encompass a definition of game engines and those engines' components as well as other aspects relating to game development, such as game genre, a now outdated classification of which was presented by Sawyer [Sawyer 1996]. Regarding the technical aspects of game engine architecture, Folmer's attempt to establish a reference architecture [Folmer 2007] can be viewed as a step in the right direction. However, these two topics represent only a small subset of the subject area and, in addition to further investigation of these areas, additional research is required to define the scope of the game development language.

*e-mail: eikea@siggraph.org
†e-mail: sengel@bournemouth.ac.uk
‡e-mail: lmcloughlin@bournemouth.ac.uk
§e-mail: peterc@bournemouth.ac.uk

2.2 What is a Game Engine: Where is the boundary between game and game engine?

As mentioned above, there is disagreement about exactly what a game engine is, with sometimes fundamental differences between definitions. Simpson [Simpson 2002] reports on the confusion between game engines and games themselves, as well as the erroneous description of game engines as the game's component for displaying graphics.

Game engine definitions that are not limited to individual engine components are often very broad and vague, such as "a framework comprised of a collection of different tools, utilities, and interfaces that hide the low-level details of the various tasks that make up a video game" [Sherrod 2007]. The main issue appears to be the question of where the boundary lies between the game engine and the game itself.

There are signs that a common consensus may be emerging for the definition of a game engine, however. More concrete descriptions have been offered, such as that by Lewis and Jacobson [Lewis and Jacobson 2002] that game engines are the "collection of modules of simulation code that do not directly specify the game's behaviour (game logic) or game's environment (level data)". While this undoubtedly brings us closer to an understanding of the boundary between game engine and game logic, this divide is still not clearly defined and many overlaps remain.

As a precursor to an investigation of game engines and their components, research in this area will have to examine computer games to identify the software components which are common and unique among different types of games. This should then lead to the discovery of a clear distinction between game *engine* code and *game* code, helping to establish this boundary.

2.2.1 Toolsets: Is the toolset part of the game engine?

An issue which is becoming increasingly relevant is that of the toolset that must accompany a modern engine. Currently, artists employed to generate game content use mostly off-the-shelf 3D modelling and animation packages, such as Maya or 3D Studio Max. These content creation packages were not, for the most part, designed for the task of creating game assets [Blow 2004], and must be augmented by exporters, world-editors and script-editors that are tailor-made for the game engine. Such tools fill the gap between the asset generation software and the game engine, and often provide an interface between artist and programmer. Without these tools there would be no game. However, an important question that needs to be answered is if such tools should therefore be considered within the scope of the definition of a game engine?

2.3 Game Genre: How do different genres affect the design of a game engine?

The design and definition of a game engine is, currently, intrinsically linked to that of the game that uses the engine. The broad subject of computer games can generally be subdivided into a number of game genres, such as real-time-strategy (RTS) games or first-person-shooters (FPS). Commercially available game engines tend to be more specialised towards the demands faced by games of a particular genre. This allows them to provide genre-specific optimisations, but usually at the expense of flexibility. While such distinctions across genres often highlight the differences between approaches, by focusing on the commonality we should start to gain a perspective and a clearer definition of what a game engine is, and what components it should be comprised of.

We believe that a formal study of engine design with respect to genre could lead to the definition of a game engine that is independent of genre.

2.3.1 Überengine: Is it possible to define a game engine independently of genre?

Currently, game engines are mostly written for specific game genres. Certain optimisations are required for each, and overheads are introduced when trying to come up with a more generic solution. The ability to use a single game engine for any genre of game instead of using a number of genre-specific engines, however, could potentially offer large benefits for the development of games. Apart from financial savings due to the use of only a single system that would have to be maintained, such an 'überengine' would facilitate the transfer of programmers and artists across projects.

There is as yet no taxonomy of game engines that could provide an insight into the commonalities between different engines. While superficially many engines appear to have similar functionality, the question that needs asking is if there is something that could be called a game engine that caters for games of any genre. In other words, if presented with two completely different games from different genres, would a developer recognise the 'engine' in each, especially considering the wide range of games that include the so-called 'serious games' which are often simulations of real-world scenarios? We are convinced that it may be possible to solve this problem, eventually.

2.4 Design Dependencies: How do low-level issues affect top-level design?

Reference architectures, such as those by Doherty [Doherty 2003] or Folmer [Folmer 2007] provide descriptions of top-level design, but rarely investigate the effect of low-level issues on the system architecture as such. It is our belief that the manner in which low-level issues influence architectural design is intrinsically linked to the fact that technology changes at a very fast rate. Such technological changes take place at the low-level, in terms of hardware or software interfaces and capabilities, and can directly affect the evolution of individual or multiple game engine components.

For example, a few years ago rendering was achieved using a fixed-function pipeline, whereas today we use programmable shaders. Initially, one might assume that such changes would be limited to within a single game engine module or component. In the case of shaders, this would obviously be rendering related. However, it is also possible that functionality can be relocated from one part of an engine to another, for example part of an engine's physics calculations could be performed by using shaders.

Another example is the introduction of multiple processors, providing resources which a modern game engine should take advantage of. The restrictions and requirements of these would result in fundamental changes to a game engine's architecture (at some level or other) to transform it into a multi-threaded engine architecture [Tulip et al. 2006].

It is clear that technology changes affect high-level architecture, but it has not yet been established to what extent. Furthermore, the question must be asked if there are any engine design methods that could be employed to minimise the impact of the future introduction of new developments in computer game technology.

2.5 Best Practice: Are there specific design methods or architectural models that are used, or should be used, for the creation of a game engine?

Common sense suggests that top-down design, and bottom-up implementation would provide a solution suitable to the development of game engines. However, this is not necessarily how game engines come into being. Empirical observation suggests that many engines grow and evolve over time. The danger with organically grown projects is that features can spiral out of control, a phenomenon known as 'feature creep', usually because the original goals were poorly defined at the beginning of a project. This has the unfortunate effect that when the implementation finally meets the original requirements, the goals may have changed. The architecture, however, may not be capable of supporting these new requirements, and workarounds, affectionately called 'hacks', must be added. As briefly mentioned above, a large section of the available literature appears to overlook this issue altogether by concentrating solely on the implementation of individual engine components, approaching the subject in terms of micro architecture as opposed to macro architecture.

The question that we ask is whether there is a 'best practice' for the development of game engines that can help eliminate, or at least reduce, these problems?

3 Summary of Research Questions

While it is usually simple to attack specific problems, a scholarly approach would be to pull back to see the bigger picture and to identify trends. This abstraction can then be used to identify gaps in the knowledge of the field and to explore ideas that others may have overlooked. Our task, as academics, should be to look at designs and find commonality, to allow us to try to piece together a 'better design'.

The aim of this call for research is to raise academic interest in the field of game engine architecture and design. Our research agenda could therefore be defined as:

- The establishment of a unified language of game development;
- The identification of software components that are common to all types of computer games;
- The establishment of clear boundaries between game engine (code) and game code;
- The definition of the role of content creation tools in the game development process and as part of game engines;
- The identification of links between game genres and game engine architecture;
- The identification of components that are common to all types of game engine, allowing the definition of a genre-independent reference architecture;
- The investigation of the effects of low-level issues on top-level game engine architecture;
- The identification of a 'best practice' for the development of game engines.

In this paper we have identified several key aspects that may help to define the problem space of game engine architecture and design. We have posed a number of questions that we believe may provide directions for future initiatives to address. A possible next step would be to set up an exploratory meeting of interested parties from industry and academia. This could be followed by the setting up of a committee or working group to coordinate research on this topic, a possible end result being the definition of an open standard for use by all.

References

BLOW, J. 2004. Game development harder than you think. *ACM Queue 1*, 10, 28–37.

DOHERTY, M. 2003. A software architecture for games. *University of the Pacific Department of Computer Science Research and Project Journal 1*, 1.

EBERLY, D. H. 2000. *3D Game Engine Design*. Morgan Kaufmann.

FLEMING, J., 2007. Down the hyper-spatial tube: Spacewar and the birth of digital game culture. Available from: http://www.gamasutra.com.

FOLMER, E. 2007. Component based game development. In *Component-Based Software Engineering*, vol. 4608 of *LNCS*, 66–73.

GARLAN, D., AND SHAW, M. 1994. An introduction to software architecture. Tech. Rep. CMU/SEI-94-TR-21, ESC-TR-94-21, Carnegie Mellon University, Pittsburgh, PA. CMU Software Engineering Institute.

LEWIS, M., AND JACOBSON, J. 2002. Game engines in scientific research. *Communications of the ACM 45*, 1, 27–31.

SAWYER, B. 1996. *The Ultimate Game Developer's Sourcebook*. Coriolis.

SHERROD, A. 2007. *Ultimate 3D Game Engine Design & Architecture*. Charles River Media.

SIMPSON, J., 2002. Game engine anatomy 101. Available from: http://www.extremetech.com.

STEPHENS, N., 2001. "game engine" versus "rendering engine". Letters to the Editor, http://www.gamasutra.com.

TULIP, J., BEKKEMA, J., AND NESBITT, K. 2006. Multi-threaded game engine design. In *Proceedings of the 3rd Australasian Conference on Interactive Entertainment*, 9–14.