IBIMA Publishing

mobile

*International Journal of Interactive Worlds*

*Vol. 2012 (2012), Article ID 418638, 77 minipages.*
*DOI:10.5171/2012.418638*
*www.ibimapublishing.com*

Copyright © 2012 Panagiotis Petridis, Ian Dunwell, David Panzoli, Sylvester Arnab, Aristidis Protopsaltis, Maurice Hendrix and Sara de Freitas. This is an open access article distributed under the Creative Commons Attribution License unported 3.0, which permits unrestricted use, distribution, and reproduction in any medium, provided that original work is properly cited.

# Game Engines Selection Framework for High-Fidelity Serious Applications

**Authors**

**Panagiotis Petridis, Ian Dunwell, Sylvester Arnab, Aristidis Protopsaltis, Maurice Hendrix and Sara de Freitas**

Serious Games Institute, Coventry University Technology Park, Coventry, West Midlands, UK

**David Panzoli**

IRIT-UT1C, Université Toulouse I Capitole, France

## Abstract

Serious games represent the state-of-the-art in the convergence of electronic gaming technologies with instructional design principles and pedagogies. Despite the value of high-fidelity content in engaging learners and providing realistic training environments, building games which deliver high levels of visual and functional realism is a complex, time consuming and expensive process. Therefore, commercial game engines, which provide a development environment and resources to more rapidly create high-fidelity virtual worlds, are increasingly used for serious as well as for entertainment applications. Towards this intention, the authors propose a new framework for the selection of game engines for serious applications and sets out five elements for analysis of engines in order to create a benchmarking approach to the validation of game engine selection. Selection criteria for game engines and the choice of platform for Serious Games are substantially different from entertainment games, as Serious Games have very different objectives, emphases and technical requirements. In particular, the convergence of training simulators with serious games, made possible by increasing hardware rendering capacity is enabling the creation of high-fidelity serious games, which challenge existing instructional approaches.  This paper overviews several game engines that are suitable for high-fidelity serious games, using the proposed framework.

**Keywords**: High Fidelity, Serious Games, Game Engines, Visualization.

## Introduction

The potential for using game engines for serious games has been recognized for years (Seung Seok Noh 2006). In 1997, the National Research Council cited some of the objectives of multi-player games for military simulation: “The underlying technologies that support these objectives address similar requirements: networking, low-cost graphics hardware, human modeling, and computer generated characters.” (National Research Council 1997). The vision of using games for simulation has been realized in a number of military training- oriented games, including America’ s Army and Full Spectrum Command (Wray 2004). As a result, there has been a trend towards the development of more complex, serious games which are informed by both pedagogic and game elements. Whilst it is true that the technical state-of-the-art in serious games mirrors that of leisure games (Anderson 2009), the technical requirements of serious games are frequently more diverse and wide-ranging than their entertainment counterparts. Serious game developers frequently resort to bespoke and proprietary development due to their unique requirements, such as the use of Blitz Games' in-house engine for developing the serious game: Triage Trainer (Jarvis 2009). Such examples of bespoke development and the limited use of off-the-shelf game engines for serious applications highlight the difficulties that exist for engine designers seeking to understand and comprehensively support the needs of instructional design.

The convergence of high fidelity simulators with serious games represents an area with increasing potential to fulfil cognitive learning requirements with a high degree of efficacy, whilst leveraging the advantages of game-based content to motivate and engage users. Modern games are frequently developed on game engines, which can be deployed on personal computers, game consoles, pocket PCs and mobile devices. The popularity of video games, especially among younger people, results in them frequently being perceived as an ideal medium for instructional programmes aimed at hard-to-reach demographics (Malone 1987); however, studies have also shown this demographic responds poorly to low-fidelity games (de Freitas 2009).

The value and impact of realism on learning purely through simulation is well-documented (de Freitas 2006; Jarvis 2009). However, when game elements are introduced, although they have been shown to improve learning in comparison to pure simulation (Mautone 2008), the relationship between simulation and game is less simple to define. Visual fidelity, whilst being a common objective in simulations, is considerably less valued in gaming; Jarvis and de Freitas (Jarvis 2009) demonstrate excessive fidelity to be detrimental to learning and the approaches put forward by learning theorists such as Vygotsky avoid creating a facsimile of reality in favour of internally-consistent abstractions. Visual fidelity is best linked to learning requirements; more is not necessarily always better, particularly from a cognitivist perspective (cognitive overload (Warburton 2008)); therefore, the ability to constrain and control fidelity can be seen as desirable, particularly with respect to composable content.

Hence, whilst visual fidelity is one component of an immersive environment, serious games often seek to use abstraction or non-realistic visual elements. Knez and Niedenthal (Knez 2008) present an interesting factor for consideration through a study linking changes in the affect of users to lighting within a virtual scene. Given that the affect of learners has a noted impact on the efficacy of learning (Bryan 1996), the capacity of a game engine to handle sophisticated lighting approaches is of concern. An increasing recent trend in game engines to transfer lighting calculations to the GPU  has led to a significant variation between engines in their support for shader models and in turn their capacity to perform sophisticated light and shading effects.

In the next section, the researchers consider in more depth the motivation behind the creation of high-fidelity serious games.

Through an analysis of background literature, the researchers are able to highlight common pedagogic elements and hence recognise the subsequent technical implications.

## Background

As  seen before, the technical state-of-the-art in serious games mirrors that of leisure games (Anderson 2009), however, the technical *requirements* of serious games are frequently more diverse and wide ranging than their entertainment counterparts. Serious game developers frequently resort to bespoke and proprietary development due to their unique requirements, such as (Playgen 2010) and (PIXELearning 2010), and difficulties exist for game engine developers in accurately understanding and supporting the needs of instructional design.

Although many serious games have limited visual interactivity, immersion and fidelity, there is an increasing motivation to create serious games that intend to support situative (social and peer-driven) and experiential pedagogies, partially because behaviourist approaches have been shown to be limited (e.g. people learn to play the game, not address learning requirements), whilst cognitive approaches struggle to impart deeper learning in the areas of affect and motivation (Egenfeldt-Nielsen 2005). Furthermore, recent work by Mautone (Mautone 2008)  demonstrates enhanced learning when introducing game elements to a standard flight simulator. Consequently, re-evaluation of simulator approaches to incorporate game and game-like elements places an increasing demand for serious game developers to deliver high-fidelity solutions. Given this motivation to create immersive, high fidelity serious games, an obvious development choice is to utilise game engines, which provide ‘out of the box’ support for state-of-the-art desktop GPU rendering and physics. In the remainder of this section, the researchers  discuss a range of key considerations when selecting the technology behind serious games, supporting effective pedagogy, the learner and their context.

Fritch (Fritsch 2004) et al compared different games engines for large-scale visualization of outdoor environments focusing mainly on the issues of composability. Similarly, Shiratuddin compared various game engines for visualizing large architectural scenes mainly focusing on accessibility and the availability of game engines (Shiratuddin 2007). Following the analysis of all these potential factors,  the methodology used here is defined to include areas, such as audiovisual fidelity, functional fidelity, composability, availability and accessibility, networking and heterogeneity.

## *Visual Fidelity*

High-fidelity in serious games is typically seen as desirable in situations where there is a need to transfer process knowledge learnt within the game to real world situations; and thus the closer the similarity between real and virtual space, the more effective the learning transfer is likely to be. Although a link between learning transfer and verisimilitude of learning activity has been observed particularly in training contexts (Park 2005; Janet L. Grady 2008; Davidovitch 2009)), this link does not necessarily hold true in all game-based learning scenarios; Jarvis and de Freitas (Jarvis 2009) suggest that the level of fidelity required must be mapped onto learning objectives. Furthermore, an over-emphasis upon visual fidelity can mask the complexities of producing verifiable and replicable learning activities and experiences.

The need to engage the learner, and specifically, immersion through high-fidelity content is one such mechanism through which such engagement can be achieved. The concept of immersion is a common one in serious games, although the components that constitute an immersive experience can be more difficult to define. The capacity to immerse learners is a significant consideration, although the means for achieving this immersion can be diverse from highly visual content to less technical approaches such as narrative immersion (Mott 2006). Robertson et al (Robertson 1997) looked specifically at the relationship between user and standard desktop PC as an interface for virtual reality, and compared it to head-tracked systems. Robertson claims “immersion should not be equated with the use of head-mounted displays: mental and emotional immersion does take place, independent of visual or perceptual immersion”, an opinion reinforced by Csikszentmihalyi and Kubey (Csikszentmihalyi 1981). Thus, a further discretisation of the concept of immersion between psychological and perceptual levels is identified. The role consistently is one of ‘drawing in’ the user, such that they experience a perceptual shift between simply viewing the screen and existing within the environment. Breaks in consistency, such as those induced by low frame rate or discontinuities in world content are shown to have a significant negative impact on immersion (Csikszentmihalyi 1981).

With respect to all three of these aspects, there are a number of dimensions in which fidelity must be considered. At a high level, there are many aspects of a game that can be represented with differing levels of fidelity; the narrative, the depth of visual and auditory content, the interaction medium and the behaviour of characters and objects within the game world. Whilst all these concerns must be reflected upon when designing a serious game, in terms of engine selection, a clear distinction exists between affordances for audiovisual and functional fidelity. It is possible to engineer a world, which appears realistic but does not behave in a realistic fashion. Although increasing audiovisual fidelity often implies increased functional fidelity,  as a virtual room fills out with furniture to become more visually realistic, players start expecting furniture to function as it would in the real world. For example, placing a virtual telephone on top of a desk can bring with it a host of potential questions from users expecting to be able to dial out.

The concept of immersion is a common one in serious games, although its definition, and specifically, the components that create an immersive experience can be more difficult to define. The capacity to immerse learners is a significant consideration, although the means for achieving this immersion can be diverse: from visual fidelity to functional fidelity as well as less technical approaches such as narrative immersion. A simple out-of-place texture or inappropriate sound can have catastrophic effects for believability, as metrics of immersion such as the performance indications and cognitive surveys applied by Pausch et al. (Pausch 1997).

## *Accessibility*

Unlike leisure games, target demographics for serious games are often non-game players, with little interest in technology or knowledge of user interfaces. Furthermore, the developers of serious games may be instructional designers wishing to explore a new medium, rather than traditional game developers seeking to develop instructional content. Therefore, the capacity of the engine to support both developers and users with limited expertise is of relevance. Previous Research (Jarvis 2009) has shown that conventional keyboard and mouse interaction in a world with multiple degrees of freedom can prove initially overwhelming for non-gamers. As such, serious games can often require interfaces that deviate from those common to entertainment games, and seek to simplify interactions based upon an understanding of learning requirements.

## *Heterogeneity*

Zyda identified as early as 1995 (Zyda, 1995) three fundamental challenges in multiuser virtual environment design, which he defines as composability, scalability and heterogeneity. These remain substantial challenges within current virtual environment research. With respect to serious games, heterogeneity is of particular concern, since target demographics are frequently ‘non-gamer’, and thus platforms capable of deployment across a wide range of hardware and software platforms are significantly advantageous.

***Composability***

Serious games often seek to model real world locales and situations, or adapt real world data for use in games with minimal development overheads. Composability in this context is used to describe both the reusability of content created within a game engine, and also its capability to import and use data from common or proprietary sources.

Technology evolution often requires the recreation of game content in ever-increasing levels of visual and functional fidelity.. If the scene consists of many high fidelity objects at various distances, it may be possible to adopt a low level of-detail approach (Engel) and use less complex geometry, or even dummy objects (Akenine-Moller T. 2008), to approximate distant objects

(Sander 2006). Alternatively, if only a small sub-section of the world or object is in sight at any one time, it may be possible to hold only these visible parts in memory and then replace them as new parts come into view by applying some form of spatial partitioning (Crassin 2009). Another issue that the designer/developer has to consider arises when attempting to import a high-fidelity model into a game engine, which may not support the geometry or texture formats within the model, or require specific optimisations such as polygon reduction to be rendered in real-time.

Two principal solutions exist: the first is to address the problems on a large-scale through an algorithmic approach that seeks to provide automated conversion between formats and import the model as a whole into the game engine. The second is to select specific components of the model for conversion and progressively convert and integrate them into the game engine by hand. However, algorithms intending to support such an approach face a number of challenges: decomposition of a mesh into multiple levels-of-detail is difficult to optimise since the perceived visual fidelity of the resultant lower-resolution meshes is linked to the perception characteristics of the user, and hence a solution is not only mathematically complex, but must also consider how humans perceive and integrate features of a three-dimensional scene (Treisman 1980). Therefore, developers are commonly forced to select specific components of the model for conversion, and progressively convert and integrate them into the game engine by hand, a task necessary when conversion tools are inadequate or the original data format has insufficient information for the game engine; often the case when models are developed without adequate information on materials, textures, bump maps or levels-of-detail.

Secondly, occlusion culling may only be performed if information on the visibility of polygons is computationally less expensive to obtain than rendering them. Early game engines such as the Quake engine seeks to achieve visibility data prior to run time through pre-processing a visibility matrix and binary partitioning of virtual space. Thirdly, mesh decomposition must be performed in concert with analysis and scaling of texture level of detail to provide a satisfactory visual outcome.

***Networking***

Multi-user elements are often specified at early-stage designs of serious games, since they often affect the nature of the game and its role within a training programme as a whole. User interaction within the game itself is often used to address the difficulties in automating the behaviour of non-player characters in a believable and coherent fashion. In this context, instructors play the role of virtual characters in order to converse and interact with learners in a realistic and adaptive manner. Whilst this can be an effective way of creating believable virtual scenarios, it suffers from limited scalability due to the availability of instructors and practical limits on pupil-tutor ratios.

Support for larger-scale communities and social elements are gaining increasing recognition within the leisure gaming community as a mechanism for increasing uptake and long-term play. As an example, the recent Guitar Hero [35] game wraps a cognitively simple task within a socio-culturally motivating setting, and has consequently proved to be highly successful commercially. Within serious games, social elements often take the form of online communities, and the convergence of games with social networking technologies remains an area of interest.

## Summary of Requirements for Serious Games

As  seen in section 0, the key elements are fidelity, consistency and support for tools for creating immersion and flow. Fidelity can be further subdivided into visual and functional fidelity.

Consistency relates to technical features in this area including the need to load between areas or stream geometry, or whether the engine is standalone or web based. Finally, corresponding features for tools creating immersion and flow include game scripting tools, especially when narrative is a non-linearity (i.e. implementing some form of artificial intelligence or artificial life). This ties in with immersion, but  the term immersion within the framework is avoided due to the current lack of consensus on its definition (Slater 1993), (Slater 2003). Instead the researchers focus on the elements that contribute to immersive experiences.

Additional technical elements include heterogeneity (on which platforms can the engine be deployed ; what hardware requirements; can it scale automatically), accessibility (support for non-standard interfaces and devices; as well as support for standardised interfaces e.g. WASD) and multiuser support, beneficial since in the absence of sophisticated AI, human instructors often play a role in virtual learning experiences, and similarly socio-cultural elements can be key motivators as mentioned above.

## Overview of Modern Game Engines

Modern game engines combine several technologies from the area of computer science such as: graphics, artificial intelligence, network programming, languages and algorithms. Modern computer game engines are robust and extensively tested (Lepouras G. 2005), in terms of the usability and performance, work on off-the-shelf systems (Robillard G. 2003) and can be easily disseminated, for example via online communities

(Burkhard C. Wünsche 2005). A game engine is any tool or a collection of tools that creates an abstraction of hardware, and/or software, for the purpose of simplifying common game development tasks. Many computer game developers support modification of their game environments by releasing level editors, for example to modify the game environment and tools to edit the game behaviour. This allows the reuse of the underlining game engine technology, including 3D rendering, 2D drawing, sound, user input and world physics/dynamics (Lewis M. 2002). For example, users can create new levels, maps and characters, adding them to the game, known as partial conversion or they can create entirely new games by altering the game source engine, known as total conversion (Trenholme 2008). Modern game engines have a modular structure, so that they can be reused into different games (Jacobson J. 2003; Trenholme 2008).

Analysis of a range of current game engines suggests the following modules are common: the Graphics Module, the Physics Module, the Collision Detection Module, the I/O Module, Sound Module, the AI Module and the Network Module. The graphics module is responsible for the generation of the 2D/3D graphics in the environment, including libraries for texture mapping, shadowing, lighting and shader effects. The Physics Module ensures that objects behave according to physical laws, for example objects fall under gravity and glass breaks. The Collision Detection Module is used to ensure that certain actions will occur when two objects collide. Furthermore, the input/output module is responsible for the input and output device which can be integrated into the 3D engine.

One of the most important elements of the creation of serious games is the visual representation of these environments. Although serious games have design goals that are different from those of pure entertainment video games, they can still make use of the wide variety of graphical features and effects that have been developed in recent years. Most game engines provide support for texture mapping, shadowing, lighting and shader effects in their graphics model. Additionally, modern engines frequently include a selection of such effects, which can include more traditional image processing, such as colour correction, film-grain, glow or edge-enhancement, as well as techniques that require additional scene information, such as depth of field and motion blur (Akenine-Moller T. 2008).

An Artificial Intelligence module, often used to create objects or "Non Playing Characters" (NPCs), is able to interact "intelligently" with the player. An important aspect in the creation of realistic scenes is to create in the game environment intelligent behaviours for the inhabitants of the virtual world, which is achieved using Artificial Intelligence (AI) techniques. However, it is important to understand that when the researchers  refer to the AI of virtual entities in game engines, it is not truly AI – at least not in the conventional sense (McCarthy 2007). The techniques applied to computer games are usually a mixture of AI-related methods whose main concern is the creation of a believable illusion of intelligence (Scott 2002), e.g. the behaviour of virtual entities only needs to be believable enough to convey the presence of intelligence and to immerse the human participant in the virtual world. The Network Module is responsible for the multiplayer implementation of the game. As a result, players could cooperate in exploring an area or exchange opinions about certain aspects of a virtual environment, while being located in different areas of the world.

The IO module provides support for different input/output devices. This module provides tools that allow the user to communicate and interact with the game environment. Most game engines provide support for standard input devices such as, joysticks, gamepads and keyboard. Technological improvements and cost reduction in computing power, display and sensor technology have resulted in a widespread use of 3D Input Devices (Fröhlich 2000; Petridis 2005; Mourkoussis 2006). Devices, such as the Nintendo Wii and Playstation controller provide 6 Degrees

Of Freedom interaction that could enhance the user interaction with the environment and increase the immersion of the user.

With such a broad definition, what is referred to as a game engine can vary among developers and where a game engine ends and a game begins is not always a clearly defined line (Trenholme 2008). Game engines should be distinguished from graphics engines that come only with rendering capabilities, and also from Software Developer Kits (SDKs) that aid game development. The reason for this differentiation is that graphics engines impose limitations as to what and how things can be included in a game, whilst SDKs are much more flexible but with narrower focus. For example, Gamebryo is a very flexible proprietary renderer but has no collision detection or physics capabilities, unlike Havoc, which is solely a physics engine. Similar middleware include Criterion's Renderware and Speedtree.

## ***A Framework for Game Engine Selection***

In short,  key elements are defined arising from the background as fidelity subdivided into visual and functional fidelity, consistency, (technical features in this area would include need to load between areas or stream geometry, whether the engine is standalone or web based – basically things that impact on immersion), and support for other tools for creating immersion and flow such as narrative (corresponding features include: game scripting tools), particularly narrative which supports non-linearity (artificial intelligence, artificial life). This ties into immersion; the researchers deliberately avoid "immersion" and the factors that affect immersion within the framework and because it is decided to focus on the elements that contribute to immersive experiences.

Additional technical elements include heterogeneity (on which platforms can the engine be deployed ; what hardware requirements; can it scale automatically), accessibility (support for non-standard interfaces and devices; as well as support for standardised interfaces (e.g. WASD) and multiuser support, beneficial since in the absence of sophisticated AI, human instructors often play a role in virtual learning experiences, and similarly socio-cultural elements can be key motivators as mentioned above.

## Table 1: Framework for Comparing Engines in SG

| | |
|---|---|
| ***Audiovisual Fidelity*** | Rendering |
| | Animation |
| | Sound |
| ***Functional Fidelity*** | Scripting |
| | Supported AI Techniques |
| | Physics |
| ***Composability*** | Import/ Export Content |
| | Developer Toolkits |
| ***Accessibility*** | Learning Curve |
| | Documentation and Support |
| | Licensing |
| | Cost |
| ***Networking*** | Client Server/ Peer–to- peer |
| **Heterogeneity** | Multiplatform Support |

## *Current Game Engines Compared Using the Framework*

At the current time, the researchers have identified over 100 games engines available on the market (WikiPedia 2010). It was decided to compare a subset of this number with the framework in order to gain some baseline data for validating the model. The rationale for the selection of these games engines over others is to meet several identified criteria including: wide usage of engine, availability, modularity and innovative features.

The first game engine under comparative analysis here is the CryENGINE 3. The engine is available for the PC, PS3 and Xbox360. CryENGINE 3 supports development in Microsoft Direct X 9,10 and 11.

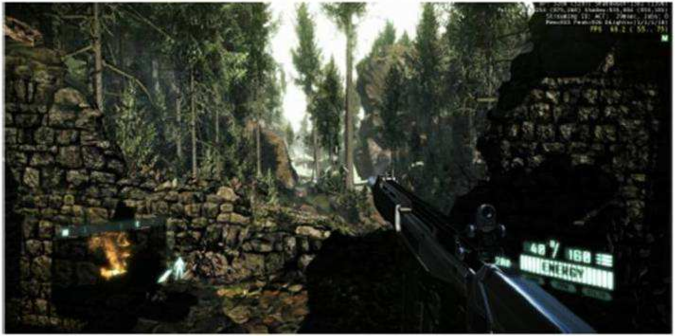


**Figure 1: Screenshot from the CryENGINE3**

The CryENGINE supports a number of features that are useful for creating immersive and realistic serious games, such as real-time world editor, bump mapping, dynamic lights, an integrated multi-threaded physics engine, shaders, shadow support, multi-core support, character animation system, pathfinding, dynamic sounds and interactive music. The engine supports all currently available hardware and it is updated with further hardware support when it becomes available.

The second engine under comparison is Valve's Source Engine. Source Engine provides a number of features such as character animation, advanced AI, real-world physics, shader-based rendering and a highly extensible development environment. The level editor that is used to produce levels for games using the Source Engine is the Valve Hammer Editor, commonly referred to as Hammer. The editor uses brushes, called primitives, to construct a level. The engine also supports particle effects, volumetric smoke and environmental effects such as fog or rain. Typical games that use the Source Engine are Half Life 2(Valve 2010), Dark Messiah of Might and Magic (Ubisoft 2010).

**Figure 2: Screenshot of Half Life 2**

The third engine is the Unreal Engine 3 (Games 2009), which has a complete game development methodology for next-generation consoles and DirectX9-equipped PC's, providing the vast range of core technologies, content creation tools and support infrastructure required by top game developers. The engine supports high performance rendering, advanced animation and high-quality dynamic lighting. Supported effects include a particle system for particles composed of sprites, meshes, lines and beams. The particle system supports various lifetime, texture, movement and collision options. All particle features can be manipulated in real-time in the editor. The texturing features of the engine include a material system that supports alpha-blending, e.g. transparency and blending of multiple layers of textures.

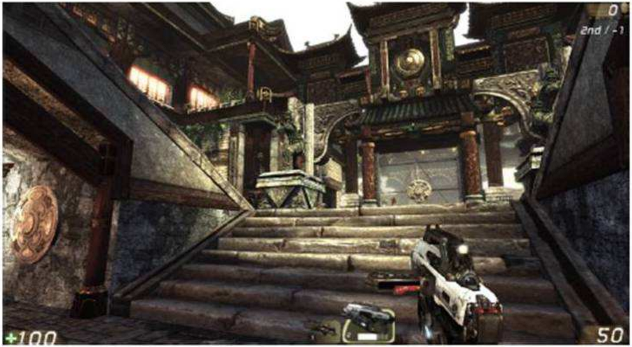
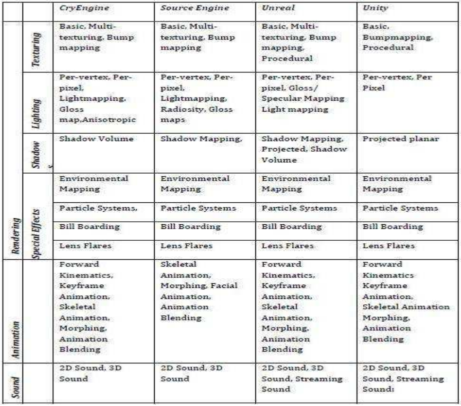


**Figure 3: Screenshot of Unreal Tournament 3**

The final engine under comparison is the Unity engine (Unity 2009), which is a game development tool that allows the developer to create games for different platforms, such as the iPhone, Nintendo Wii, Mac and PC.

CryEngine 3, Unity, Valve's Source Engine and the Unreal engines provide high-fidelity standards and provide support for all the latest technologies of computer graphics as can be seen from Table 2. Using these features such as texturing, lighting, shadows and special effects, the developer has the potential to deliver the same perceptual quality, as the user was present in the real scene. Thus, the player can feel actually present in the real scene being depicted.

## Table 2: Audiovisual Fidelity



| | | *CryEngine* | *Source Engine* | *Unreal* | *Unity* |
|---|---|---|---|---|---|
| *Rendering* | *Texturing* | Basic, Multi-texturing, Bump mapping | Basic, Multi-texturing, Bump mapping | Basic, Multi-texturing, Bump mapping, Procedural | Basic, Bumpmapping, Procedural |
| | *Lighting* | Per-vertex, Per-pixel, Lightmapping, Gloss map,Anisotropic | Per-vertex, Per-pixel, Lightmapping, Radiosity, Gloss maps | Per-vertex, Per-pixel, Gloss/ Specular Mapping Light mapping | Per-vertex, Per Pixel |
| | *Shadows* | Shadow Volume | Shadow Mapping, | Shadow Mapping, Projected, Shadow Volume | Projected planar |
| | *Special Effects* | Environmental Mapping | Environmental Mapping | Environmental Mapping | Environmental Mapping |
| | | Particle Systems, | Particle Systems | Particle Systems | Particle Systems |
| | | Bill Boarding | Bill Boarding | Bill Boarding | Bill Boarding |
| | | Lens Flares | Lens Flares | Lens Flares | Lens Flares |
| *Animation* | | Forward Kinematics, Keyframe Animation, Skeletal Animation, Morphing, Animation Blending | Skeletal Animation, Morphing, Facial Animation, Animation Blending | Forward Kinematics, Keyframe Animation, Skeletal Animation, Morphing, Animation Blending | Forward Kinematics Keyframe Animation, Skeletal Animation Morphing, Animation Blending |
| *Sound* | | 2D Sound, 3D Sound | 2D Sound, 3D Sound | 2D Sound, 3D Sound, Streaming Sound | 2D Sound, 3D Sound, Streaming Sound: |

Another major challenge in the selection of game engines for serious games is functional fidelity. Functional fidelity is closely related to the AI, physics and scripting. All the selected game engines provide support for various AI techniques, such as collision detection and path finding. Additionally, all game engines have integrated their own physics engines, and furthermore, each of the selected game engines has support for scripting languages.

## Table 3: Functional Fidelity

| | | ***CryEngine*** | ***Source*** | ***Unreal*** | ***Unity*** |
|---|---|---|---|---|---|
| ***Scripting*** | Script | Yes | Yes | Yes | Yes |
| | Object Model | Yes | No | No | No |
| ***Supported AI Techniques*** | Collision Detection | Yes | Yes | Yes | Yes |
| | Path Finding | Yes | No | Yes | Yes |
| | Decision Making | Yes | No | Yes | Yes |
| ***Physics*** | Basic Physics | Yes | Yes | Yes | Yes |
| | rigid body | Yes | Yes | Yes | Yes |
| | vehicle dynamics | Yes | Yes | Yes | Yes |

The reusability of content created within a game engine and the capability of importing and using data from common sources should be considered before selecting a game engine. As budgets for creating serious games are usually very limited compared to commercial entertainment games, this becomes an especially important aspect(Protopsaltis 2010; Protopsaltis 2011). Additionally, in serious games, the developers need to have access to the SDK, GDK in order to add different peripheral devices or connect the game engine with learning management systems or with other software APIs. However, from the comparison of the game engines, the researchers  identified that importing a 3D model from CAD software into the supported format of the game engine is a major issue and requires the developer to select specific components of the model for conversion and progressively convert and integrate them into the game engine by hand. Once this obstacle has been overcome, the game engines can create high-fidelity indoor and outdoor scenes occupied with non-player characters in real-time. Table 4 compares the selected engines according to the suggested categories.

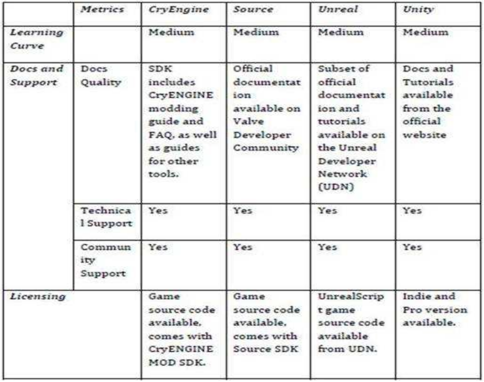
## Table 4: Composability

| | ***Metrics*** | ***CryEngine*** | ***Source*** | ***Unreal*** | ***Unity*** |
|---|---|---|---|---|---|
| ***Import/ Export Content*** | CAD Platforms supported | 3ds max, maya | 3ds max, maya | 3ds max, maya, | 3ds max, maya, |
| | Import Export Limitations | No | No | Yes | No |
| | Content Availability | Small | Large | Large | Mediu m |
| ***Developer Toolkits*** | SDK/GDK | Yes | Yes | Yes | No |

Another major challenge in the selection of game engines for serious games is accessibility; that is, how easy it is to retrieve supporting information about the game engine (see Table 5). CryENGINE provides all the necessary development tools that can be accessed from games that use the engine (i.e. Crysis, FarCry). The engine offers the Sandbox editor, which allows the user to edit levels in real-time. Partial source code and documentation is included with a freely downloadable SDK. Documentations for the Unreal Engine are available through the Epic's Unreal Developer Network (UDN) (Epic Games 2010), which is the official support site for licensees and mod (modifications) developers, providing technical documentation, as well as tutorials, for the Unreal engine and UnrealEd. There are also a number of community websites that provide discussion forums and tutorials (Planet Unreal 2010; Unreal Wiki 2010).

Documentation for the Source SDK is provided on the Valve Developer Community wiki (Valve's Wiki 2010), which provides the most comprehensive guide on using the Source Engine. The Source SDK contains all the necessary tools in order to create a mod and develop game content. The Source SDK also provides a Create a Mod option, which copies the necessary source code and resources to a working directory. A large number of other websites provide their own discussion forums and tutorials, which range from general introductions to the Source SDK tools to more specific tasks (Interpolers.net ; EditLife 2010).

## Table 5: Accesibility



| | *Metrics* | *CryEngine* | *Source* | *Unreal* | *Unity* |
|---|---|---|---|---|---|
| *Learning Curve* | | Medium | Medium | Medium | Medium |
| *Docs and Support* | Docs Quality | SDK includes CryENGINE modding guide and FAQ, as well as guides for other tools. | Official documentation available on Valve Developer Community | Subset of official documentation and tutorials available on the Unreal Developer Network (UDN) | Docs and Tutorials available from the official website |
| | Technical Support | Yes | Yes | Yes | Yes |
| | Community Support | Yes | Yes | Yes | Yes |
| *Licensing* | | Game source code available, comes with CryENGINE MOD SDK. | Game source code available, comes with Source SDK | UnrealScript game source code available from UDN. | Indie and Pro version available. |

The next step in the selection process of the game engine is to focus on the heterogeneity of the engines and their network support (see Table 6). All the selected engines support client-server architectures. However, if the serious game is going to support a large virtual world with hundreds of users, a network supported layer has to be built.

**Table 6: Networking and Heterogeneity**

| | | ***CryEngine*** | ***Source*** | ***Unreal*** | ***Unity*** |
|---|---|---|---|---|---|
| ***Networking*** | *Client-Server,* | Yes | Yes | Yes | Yes |
| | *Peer-to-Peer* | No | No | No | No |
| ***Hetero geneit*** | *Multiplatform* | *Yes* | *Yes* | *Yes* | *Yes* |

Our analysis of four leading entertainment game engines demonstrates that whilst all four provide a sound basis for serious game development, none give specific regard to serious applications. Unsurprisingly perhaps, as none of these games engines have been developed with serious applications in mind;

which in turn may have something to do with the relatively small budgets often found in serious game development. Rather, the emphasis is upon the designer to integrate technology with instructional design, and to this end, the game engine plays the role of an implementation tool, rather than fully supporting the process of serious game development. Future viability of game engines for serious applications will depend on how capable they become at supporting the concepts and core dissimilarities between serious and leisure applications, which includes supporting participation and learner involvement in the development process, integration into intelligent tutoring systems (Dunwell 2011) and support for metadata (Hendrix 2012) in the form of learning objects and repositories. Furthermore, user interactions with serious applications are often diverse and reflected in a wide range of levels of user expertise and expectation, and therefore support for interfaces outside of those common to entertainment games is advantageous.

## Conclusions and Future Work

The creation of a serious game is a complex engineering project that requires technical expertise, as well as a careful balance of game design principles with instructional content. Similarly, the development of generic engines which underpin serious and leisure games is a complicated process that requires time, resources and teamwork. As serious games become more complex, so do the engineering challenges that arise during the development of the game. Hence, the early-stage selection of the optimal engine for development is crucial. This paper presents a selection framework, allowing the developer to select the ideal engine based on the technical requirements of the serious game.

This is the first framework for serious game engine selection currently proposed and tested, and is intended as a starting point for ongoing benchmarking and metrics for supporting serious game engine selection. However, whilst our framework relates overarching technical requirements to a range of modern engines, more research, testing and validation must still be done to relate learning requirements and instructional design principles to these technical features. Ultimately, the design and implementation of effective serious games must be grounded in pedagogy, as well as technology, and therefore future work should address the many issues surrounding the equation of learning requirements to these identified technical features.

Towards this end, future studies will focus upon the analysis of the impact of the various engines and their functionalities on targeted learner groups.

Finally, whilst the researchers sought to identify and evaluate the most widespread subset of game engines within the context of our framework, given the rapid evolution of technology in this area, comparison of additional and emerging game engines may also offer opportunities to further refine and validate the framework. Thus, future work will build upon and further test game engines using this framework method of validation.

## References

Anderson, E. F., McLoughlin, L., Liarokapis, F., Peters, C., Petridis, P. & de Freitas, S. (2009). "Serious Games in Cultural Heritage," 10th VAST International Symposium on Virtual Reality, Archaeology and Cultural Heritage (VAST '09), VAST-STAR, Short and Project Proceedings, Eurographics, Malta, 22-25.

Bryan, T., Mathur, S. & Sullivan, K. (1996). "The Impact of Positive Mood on Learning," *Learning Disability Quarterly* 19(3): 153-162.

Burkhard C. Wünsche, Blazej Kot, Andrew Gits, Robert Amor & John Hosking. (2005). "A Framework for Game Engine Based Visualisations," *Proceedings of Image and Vision Computing New Zealand*.

Crassin, C., Neyret, F., Lefebvre, S. & Eisemann, E. (2009). "Gigavoxels: Ray-Guided Streaming for Efficient and Detailed Voxel Rendering," I3D ’09:Proceedings of the 2009 symposium on Interactive 3D graphics and games (2009). 9: 15-22.

Csikszentmihalyi, M. & Kubey, R. (1981). "Television and the Rest of Life: A Systematic Comparison of Subjective Experience," *Public Opinion Quarterly* 45(3): 17-328.

Davidovitch, L., Parush, A. & Shtub, A. (2009). "The Impact of Functional Fidelity in Simulator-based Learning of Project Management," *International Journal of Engineering Education* 25(2): 333-340.

De Freitas, S. & Jarvis, S. (2006). "A Framework for Developing Serious Games to Meet Learner Needs," The Interservice/Industry Training, Simulation and Education Conference. Florida,USA.

De Freitas, S. & Neumann, T. (2009). "The Use of 'Exploratory Learning' for Supporting Immersive Learning in Virtual Environments," *Computers and Education* 52(2): 343-352.

Dunwell, I., Petridis, P., Arnab, S., Protopsaltis, A., Hendrix, M. & de Freitas, S. (2011). "Blended Game-Based Learning Environments: Extending a Serious Game into a Learning Content Management System," ALICE 2011: International Workshop on Adaptive Learning via Interactive, Collaborative and Emotional approaches, at INCOS 2011: Third International Conference on Networking and Collaborative Systems

EditLife. (2010). Online Resources for Half Life,   Retrieved 21/06/2010, from http://www.editlife.net/

Egenfeldt-Nielsen, S. (2005). "Beyond Edutainment: Exploring the Educational Potential of Computer Games," *Phd thesis, IT-University Copenhagen*.

Engel, W.,  Hoxley,  J., Kornmann, R. & Suni Zink J. Programming Vertex, Geometry, and Pixel Shaders,  line book available at: http://wiki.gamedev.net/, 2008. 9, 11."

Epic Games. (2010). UDN Network Retrieved 21/06/2010, from http://udn.epicgames.com/Main/WebHome.html.

Fritsch, D. & Kada, M. (2004). "Visualization Using Game Engines," *ISPRS commission 5*. Istanbul, Turkey,: 621-625.

Fröhlich, B., Plate, J., Wind, J., Wesche, G. & Gobel, M. (2000). "Cubic-Mouse-Based Interaction in Virtual Environments," *IEEE Computer Graphics and Applications* 20(4): 12-15.

Games, E. (2009). Retrieved 06/11/2009, from http://www.unrealtechnology.com/.

Hendrix, M., Protopsaltis, A., Rolland, C., Dunwell, I., de Freitas, S., Arnab, S., Petridis, P. & Llanas, J. (2012). "Defining a Metadata Schema for Serious Games as Learning Objects," *International Conference on Mobile, Hybrid, and On-line Learning* (eL&mL) IARIA.

Interpolers.net. Resources Available for Source Engine. Retrieved 21/05/2010, from http://www.interlopers.net/.

Jacobson J. (2003). "Using ''CaveUT'' to Build Immersive Displays with the Unreal Tournament Engine and a PC Cluster,*"ACM symposium on interactive 3D graphics*, *ACM Press*.

Janet L. Grady, Rosemary G. Kehrer, Carole E. Trusty, Eileen B. Entin, Elliot E. Entin & Tad T. Brunye (2008). "Learning Nursing Procedures: The Influence of Simulator Fidelity and Student Gender on Teaching Effectiveness," *Journal of Nursing Education* 47(9).

Jarvis, S. & de Freitas, S. (2009). "Evaluation of an Immersive Learning Programme to support Triage Training," Proceedings of the 1st IEEE International Conference in Games and Virtual Worlds for Serious Applications, *IEEE Computer Society*,. Coventry,UK: 117-122.

Knez, I., & Niedenthal, S. (2008). "Lighting in Digital Game Worlds: Effects on Affect and Play Performanc," *Cyberpsychology & Behavior*(11): 129-137.

Lepouras, G. & Vassilakis, C. (2005). "Virtual Museums for all: Employing Game Technology for Edutainment," *Virtual Real*. 8: 96-106.

Lewis M. &  Jacobson, J. (2002). "Game Engines in Scientific Research," *Commun ACM* 45(1): 27-31.

Malone, T. W. & Lepper, M. R. (1987). "Making learning Fun:A Taxonomy of Intrinsic Motivations for Learning," In *Aptitude, learning and instruction*: III. Conative and affective process analyses. F. J. Snow R. E. Erlbaum: 223-253.

Mautone, T., Spiker, V. A. & Karp, M. R.  (2008). "Using Serious Game Technology to Improve Aircrew Training," *In Proc. of I/ITSEC* 2008.

McCarthy, J. (2007). "What is Artificial Intelligence," Available At http://www-formal.stanford.edu/jmc/whatisai/whatisai.html

Moller, T. A.,  Haines, E. & Hoffman, N. (2008). Real-Time Rendering, *A. K. Peters*.

Mott, B. W., McQuiggan, S. W., Lee, S., Lee, S. Y. & Lester, J. C. (2006). "Narrative-Centered Environments for Guided Exploratory Learning," Proceedings of the Agent Based Systems for Human Learning Workshop at the 5th International Joint Conference on Autonomous Agents and Multiagent Systems (ABSHL-2006). Hakodate, Japan.

Mourkoussis, N., Mania, K., Petridis, P., White, M., Rivera, F. M., Pletinckx, D., Troscianko, T. & Hawkes, R. (2006). "An Analysis of the Effect of Technological Fidelity on Perceptual Fidelity," To appear in proceedings of the IEA 2006 (International Ergonomics Association), 16th World Congress on Ergonomics, The Hague, Netherlands.

National Research Council (1997). "Modeling and Simulation: Linking Entertainment & Defense," *National Academy Press*.

Park, G. D., Allen, R. Wade, Rosenthal, Theodore J. & Fiorentino Dary. (2005). "Training Effectiveness: How Does Driving Simulator Fidelity Influence Driver Performance?," *Human Factors and Ergonomics Society Annual Meeting Proceedings, Training*. 5: 2201-2205.

Pausch, R., Proffitt, D., & Williams, G. (1997). "Quantifying Immersion in Virtual Reality," Proceedings of the 24th annual conference on Computer graphics and interactive techniques, *ACM Press*: 13-18.

Petridis, P., White, M., Mourkousis, N., Liarokapis, F., Sifiniotis, M. Basu, A. & Gatzidis, C. (2005). "Exploring and Interacting with Virtual Museums," *CAA* 2005: The World in your eyes, Tomar,Portugal.

PIXELearning. (2010). Pixel Learning: Serious Games and Immersive Simulation for Learning and Development, Retrieved 21/06/2010, from http://www.pixelearning.com/.

lanet Unreal. (2010). Planet Unreal Resource, 21/06/2010, from http://planetunreal.gamespy.com/.

Playgen. (2010). "Make Games and Simulations with PlayGen," Retrieved 21/06//2010, from http://playgen.com/.

Protopsaltis, A., Auneau, L., Dunwell, I., de Freitas, S., Petridis, P., Arnab, S., Scarle, S. & Hendrix, M. (2011). "Scenario-Based Serious Games Repurposing," ACM SIGDOC 29th International Conference on Design of Communication, *ACM*

Protopsaltis, A., Panzoli, D., Dunwell, I. & de Freitas, S. (2010). "Repurposing Serious Games in Health Care Education," *12th Mediterranean Conference on Medical and Biological Engineering and Computing* (MEDICON 2010)

Robertson, G., Czerwinski, M. & Van Dantzich, M. (1997). "Immersion in Desktop Virtual Reality," Proceedings of the 10th annual ACM symposium on User interface software and technology, *ACM Press*.: 11-19.

Robillard, G. Bouchard, S. Fournier, T. & Renaud, P. (2003). "Anxiety and Presence Using VR Immersion: A Comparative Study of the Reactions of Phobic and Non-Phobic Participants in Therapeutic Virtual Environments Derived from Computer Games." *CyberPsychol Behav* 6(5): 467-475.

Sander, P. V. & Mitchell, J. L. (2006). 'Out-of-Core Rendering of Large Meshes with Progressive Buffers,' *ACM SIGGRAPH* 2006: Proceedings of the conference on SIGGRAPH 2006 course notes (2006). 9: 1-18.

Scott, B. (2002). The Illusion of Intelligence, *AI Game Programming Wisdom*, Charles River Media: 16-20.

Seung Seok Noh, Sung Dea Hong & Jin Wan Park (2006). "Using a Game Engine Technique to Produce 3D Entertainment Contents," Proceedings of the 16th International Conference on Artificial Reality and Telexistence--Workshops (ICAT'06).

Shiratuddin, M. F. & Fletcher, D. (2007). "Utilizing 3D Games Development Tool for Architectural Design in a Virtual Environment," 7th International Conference on Construction Applications of Virtual Reality.

Slater, M. (2003). "A Note on Presence Terminology," from www.presence-connect.com.

Slater, M. & Ushoh, M.(1993). 'Representation Systems, Perceptual Position, and Presence in Immersive Virtual Environments,' Presence: Teleoperators and Virtual Environments 2(3): 221-233.

Treisman, A. M. & Gelade, G. (1980). "A Feature-Integration Theory of Attention," *Cognitive Psychology* 12(1): 97-136.

Trenholme, D. & Smith, S. P. (2008). "Computer Game Engines for Developing First-Person Virtual Environments," *Virtual Reality*, 12(3): 181-187.

Ubisoft. (2010). Retrieved 21/06/2010, 2010, from http://darkmessiahgame.uk.ubi.com/.

Unity. (2009). Retrieved 04/11/2009, from http://unity3d.com/.

Unreal Wiki. (2010). Unreal Wiki. Retrieved 21/06/2010, from http://wiki.beyondunreal.com/.

Valve. (2010). Retrieved 21/06/2010, 2010, from http://orange.half-life2.com/.

Valve's Wiki. (2010). "Valve's Source Developer Wiki," Retrieved 21/06/2010, from http://developer.valvesoftware.com/wiki/SDK_Docs.

Warburton, S. (2008). 'Defining a Framework for Teaching Practices Inside Virtual Immersive Environments: the Tension Between Control and Pedagogical Approach,' *Proceedings of RELive '08 Conference*.

WikiPedia. (2010). "Game Engines," Retrieved 21/06/2010, from http://en.wikipedia.org/wiki/List_of_game_engines.

Wray, R. E. Laird, J. E. Nuxoll, A. Stokes, D. & Kerfoot, A. (2004). "Synthetic Adversaries for Urban Combat Training," *P*roceedings of the 2004 Innovative Applications of Artificial Intelligence Conference, San Jose, CA.