# Comparison of SAGE, Gamebryo and Unreal Engine 3

**Christopher Pedersen**

**IT-University of Copenhagen**

E-mail: gammabyte@gmail.com/gammabyte@itu.dk

*What is the difference between using engines designed to fit certain game types?*

*The following tries to evaluate the features of three different game engines and how they are used in relation to each other.*

## INTRODUCTION

The engines I have chosen for the comparison is based primarily on the different style of games created on them. Personally I worked with the Unreal Engine last semester and it will represent the first person shooter element of this comparison.

Below you will see the three engines I have chosen:

1. SAGE
2. Gamebryo[1]
3. Unreal Engine 3[2]

The description of each engine will be followed by a comparison between them and the difference that define their usage in different areas.

## ENGINE #1 – SAGE

The SAGE (Strategy Action Game Engine) engine has primarily been used to create games in the genre of RTS (Real-Time Strategy), such as Red Alert and BFME2 (Battle for Middle Earth 2). SAGE has been developed by Westwood and is now continually developed by EA (Electronic Arts).[3]

One of the features of the SAGE engine is that it has support for dynamic lighting which lets it render light sources on the fly, basically letting objects and elements in the game seem more realistic as the light sources change the way we see the surfaces. This means that objects are now lighted in accordance to moving light sources and in some cases screen position to allow for objects to be lighted in a more realistic fashion.[4]

As an extra feature of lighting the engine also now uses HDR (High Dynamic Range) rendering which allows for more rich lighting effects on a stage. Specifically the HDR allows for elements of either very dark or light areas to be darker than dark and lighter than light. This essentially means that whenever an area of viewing is supposed to be brighter than a white value, such as when a sun reflects upon a glossy surface, the surrounding areas will show a glow that represents a brighter than white value.[5]

As many other engines the SAGE engines also employs anti-aliasing. This means that the engine makes sure that images or other signals that are distorted due to alterations in the dimensions of the element, which aren't correctly shown on a given medium, will be slightly blurred to take into account the aliasing the occurs in the "sameness" of the surrounding elements.[6] The rendering in sense has problems sometimes

[1] http://www.emergent.net/en/Products/Gamebryo/

[2] http://www.unrealtechnology.com/

[3] http://en.wikipedia.org/wiki/SAGE_engine & http://www.cncgeneralsworld.com/generals/game-engine.aspx

[4] http://gamestudies.org/0701/articles/elnasr_niedenthal_knez_almeida_zupko

[5] http://en.wikipedia.org/wiki/High_dynamic_range_rendering

[6] http://lunaloca.com/tutorials/antialiasing/

making distinctions between elements of, for example a texture that has sections that look alike to other sections because of down scaling or a greater viewing distance. When showing the results of aliasing, fine lines in the distance will possibly be shown to the user as a mash of colored pixels, in no apparent grouped order.

The SAGE engine described here does not currently support D10 (DirectX 10)[7] and therefore does not support any of the features that lie herein, such as geometry shader, Shader model 4.0, Rapid Occlusion Culling and Instancing 2.0.

The lack of support for geometry shader means that the engine will not be able to create primitives as needed. As the geometry shader sits between the vertex shader and the pixel shader, it will be possible to take primitive points and create more primitives to be able to smooth off edges before passing information on to the pixel shader. The creation of vertices is performed on the GPU to try and remove some of the transport time of data.[8]

Furthermore the lack of D10 means that ROC (Rapid Occlusion Culling) is not done. This means that when objects to be rendered are selected along the pipeline, the objects that are entirely eclipsed by other objects may be selected to be omitted for rendering. This allows a lighter load on the pipeline.[9]

### ENGINE #2 – GAMEBRYO

Gamebryo is a 3d engine which has been used to create a variety of games. Bigger games include titles such as Elder Scrolls IV: Oblivion and Fallout 3 have been created in Gamebryo, but also simulation games such as Railroad Tycoon. As such, the goal of the Gamebryo engine is to allow the users to create almost any type of game they want and not focus too much on making the engine genre specific.

The engine itself it mostly modifiable because it is supposed to be a widely genre unspecific engine and as such the developer could implement whatever he wants, however Gamebryo does come with some standard features.

The engine itself is a set of C++ libraries which developers can use to create what they need, and in that sense also remove or add to the content of the libraries as needed. The engine has support for the following platforms:

- PC
- Xbox 360
- Playstation 3
- Nintendo WII

As standard, Gamebryo comes with a Terrain Editor to lighten the load of creating worlds within space. Gamebryo also comes with a load of tools and plugins that allow users to quickly create concept game mechanics, new types of features and smaller or larger game prototypes.

One of the great tools coming with the engine is an analysis tool that allows the users to see how much time is used on rendering elements, how much is culled and how many polygons rendered for each object. This allows for optimizing on the developers side.

The culling technique that is default in Gamebryo is hierarchical frustum culling, which basically means that rendered objects do not include those which are not in the viewing perspective. However since Gamebryo is highly customizable, it is possible to just implement your own culling methods, which might suit better for the game you are currently developing.[10]

The shadow mechanism that standard comes with Gamebryo is SM (Shadow Mapping). SM uses a technique where it stores all of the depths of every surface from its own point of view; thereafter it draws the shadow map when the scene is rendered according

---

[7] http://www.firingsquad.com/matrix/blog.asp/62074/300

[8] http://www.extremetech.com/article2/0,1697,1982034,00.asp

[9] http://en.wikipedia.org/wiki/Hidden_surface_determination

[10] http://www.beyond3d.com/content/interviews/10/2

to the z-depth. More precisely, this information is stored in a depth map that is then drawn with the rest of the object at the camera view angle.[11] This technique, as opposed to SV (Shadow Volumes), should behave faster in many instances but create less accurate shadowing.

SV would create shadow polygons to be rendered as part of the shadow world and thus could create more stackup on the GPU for rendering. This is typically done by projecting lines through every vertex on the objects to be shadowed. These lines form a space that is individually colored in the correct shadow color.[12]

Gamebryo comes with support for a list of special effects features, such as billboarding, decals, environment mapping, explosion, lens flares, particle system, sky, fire and fog. [13] The particle system in Gamebryo supports tools for full import of particles from selected 3d artist programs such as Maya and 3dsMax.

Billboarding allows the game creators to cut down the amount of polygons rendered on screen by replacing objects with flat textures, thus improving performance significantly in some cases.[14] This feature can be used when objects are farther away from the player than other closer and important elements. It allows greater performance in the sense that very few details are to be rendered because of this, but gameplay, reality and the wow effect is not so greatly reduced because of the large distance between the player and the object. In a sense lower detail is allowed for places where players will most likely and usually not focus their attention on.

The environment mapping, although not specifically the type mentioned, probably uses a spherical or cube reflection mapping. This means that all objects that are supposed to give off a reflection, such as a mirror,

window, car or a spoon lying on a table, there is created either a cube or something of the like a skybox on the reflecting object. This texture will then twist and turn in accordance with the viewport and therefore will come close to showing a real reflection of the surroundings. However the object in question looking at the element is not mirrored in it.[15]

## ENGINE #3 – UNREAL ENGINE 3

The UE3 (Unreal Engine 3) is a highly modular engine that allows the creator to pull and push many features that he wants to either add or remove. This means that functionality is highly focused into classes that will allow any scripter to overwrite or add content as he wishes. The difference between Gamebryo and UE3 in modifiability can mostly be seen in the sense that UE3 has its own script language, USL (Unreal Script Language), and modifiability can be done through this entirely, however since the UE3 engine is also written in C++, and this code is available to the buying developer, the engine can also be modified for more varied use.

Although UE3 was created as a background for the game Unreal 3, there really should be no limit as to what kind of game you can create if you have the time. The engine is however widely used to create FPS (First Person Shooter) games, which include titles such as BioShock, Gears of War and Mass Effect.

In the terms of platforms, the UE3 supports a great deal, it supports Windows, Linux, Mac OS, Dreamcast, Xbox, Xbox 360, PS 2, PS 3 and Nintendo Wii.. The fact that UE3 has been written in C++ to support all these underlying platforms, makes the used unreal scripting language that more valuable as a mediator between the operating systems.[16]

As part of the ways of creating levels and worlds for the engine, UE3 contain an editor for creating worlds in a visual appearance. This editor has many features to
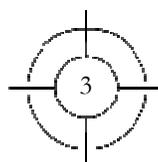
[11] http://en.wikipedia.org/wiki/Shadow_mapping

[12] http://en.wikipedia.org/wiki/Shadow_volume

[13] http://www.devmaster.net/engines/engine_details.php?id=32

[14] http://www.lighthouse3d.com/opengl/billboarding/

[15] http://en.wikipedia.org/wiki/Environment_mapping

[16] http://en.wikipedia.org/wiki/Unreal_Engine

allow non programmers to change many features, also allowing for event handling with the drag and drop feature called Kismet. Some even believe, with all the extra features in the Unreal Editor, that there soon will be no need for programmers.

The UE3 also supports HDR rendering which is described under the SAGE engine.

The engine also uses a technique called Anisotropic, which is used to remove aliasing much like anti-aliasing. However the blurred effects of aliasing is not so evident in anisotropic in more extreme angles, it is however a bit more expensive in computing power.

As opposed to Gamebryo UE3 actually has a possibility of using either SM or SV to create shadows from objects.

UE3 also comes with a rich fan of collision possibilities as it supports interaction with elements as standard and has ragdoll features, plus features to support physics of all objects including vehicles.

Unlike Gamebryo, UE3 comes with standard support for VL (Volumetric Light) that is used to create light beams also called God Rays. It allows for light beams to be projected through space to show a richer picture. This is done mainly by creating a volume for light to pass through with a property, such as fog for creating the rays.

Since UE3 is originally developed for Unreal, the engine also comes with a lot of AI possibilities such as path finding and finite state mechanics are already implemented, this would allow at least easier production of a new FPS game. This finite state machine can be altered easily through the USL to accommodate for new behaviour such as feigning death when hit points are below a certain number. It would also be possible to implement more complex method of AI, such as fuzzy logic, through the USL and the finite state class hierarchy.

### COMPARISON

Amongst all the different game engines some a primarily developed to create a certain game type, others that have a broader perspective are more modular and customizable than others. This fact shows in some of the features that are available in the certain engines.

The SAGE engine, that was primarily used to create RTS games, has features that enable it to create great looking graphics at a distance view of slight sideways or above. Since players become more focused upon graphics, the newer SAGE engine supports the HDR rendering that allows for the richer diversity in colors. The lack of D10 is not that important for the current game types as RTS is usually played from above and things like ROC is not so much needed, but as player always say they focus on gameplay, there are in fact very few old games we would play because of the horrendous graphics. The distance at which most RTS games are played also means that the use for geometry shader might not be as important as many other game engines. Players would most likely not need more rounded off shapes since objects that are used in most futuristic RTS games, such as tanks, would be squarely shaped.

The funny thing about the Gamebryo engine is that it does not contain that many features that are specifically useful to any one game type. It seems apparent that all the features of Gamebryo are created in a way that makes it possible for developers to use some features if they want a certain kind of game, like an RPG, and if they want a FPS they can assemble things on stubs to get the elements they are missing from the default engine.

This also means that the engine has features that are more light weight, such as SM and Billboarding.

In essence developers are encouraged to fill out the blanks of the engine, whilst the manufacturer of course writes the API for the most common uses, such as making the engine easily run on multiple platforms. The broadness of this engine is also signified by the titles created on it, which include RPG, Tycoon, RTS games and other genres.

It is more evident however that UE3 is more than equipped to create FPS games, since this is what it was initially created for. However the sheer amount of tools and helpful gadgets that follow this engine allow developers to create more than FPS games if they so choose.

All the features that are present for an FPS game, such as AI, HDR rendering and Anisotropic are used to create a more immersive player experience, and since eye-candy always is a popular feature, this is almost necessary in a game engine that wants to create good looking games, mostly inside the same genre but also has a different look and feel.

It is quite interesting to look at the purpose of the game types when seeing what capabilities the engines have been implemented with. For example the SAGE engine will not need to render as many different and largely detailed shapes (as in detailed FPS games), however it will need to handle smaller objects that comes in tens or hundreds at the same time but have the same shape. It just goes to show that while you would be able to make an RTS game in UE3, many features will likely be overdone with regards to what needs to be implemented, and more unspecific engines such as Gamebryo could just as well be used.

It is also important to note that more is not always best. Some target audiences that favour a certain gametype may not have the appropriate power to run a simple game in their genre that has unnecessary beautiful flowering or great looking grass, especially when the game goal is all about having 1.500 tanks on screen shooting at each other.