Aalborg University Copenhagen
Department of Medialogy

Aalborg University Copenhagen
Lautrupvang 15, 2750 Ballerup, Denmark

Semester coordinator: Stefania Serafin
Secretary: Lisbeth Nykjær
Phone: 9940 2471
lny@create.aau.dk
https://internal.media.aau.dk/

Semester: MED 10

Title: Perceived quality of smoke effects in virtual environments, using fluid systems and imposters.

Project Period: Spring 2011

Semester Theme: Master thesis

Supervisor: Rama Hoetzlein.

Created by:

Morten Flyvholm Iversen

This project revolves around the use of imposters as replacement of real particle or fluid system in interactive environments. This is done to maximize frame rates. During the test a set of complex imposters which changes viewing angle of the actual animation will be compared to a normal imposter that is constantly viewed from the same angle. This will be to test whether or not the difference in orientation is a major factor in the perceived quality of the effect.

However as no test has been conducted yet, the results cannot be presented. This report is still a work in progress and is thus incomplete.

Copies: 3

Pages: 30

Finished: May 27th - 2011

Content

Abstract

Se ″den gode opgave″ s. 204.

Readers Guide

Is in introduction, should it be alone here instead?

Introduction & Motivation

The inspiration and motivation for this project comes from the ever raging battle between hardware producers and video game developers. The graphical quality of video games are increasing rapidly, and as a result of that, hardware producers are constantly releasing new products which contain the newest technology, thus enabling the users to draw full benefit of these new graphical wonders. However, always being up to date, and never having to settle with settings below the video games absolute maximum is a rather expensive hobby, this i have experienced myself first hand. The question which is interesting to this problematic is easily described as one of the common phrases used at medialogy; *what is nice to have, and what is need to have*.

Game producers have a large variety of methods for creating graphical spectacles in their productions. They also possess a lot of methods for scaling these graphical spectacles, either showing them in their fullest quality, a decreased quality or as an illusion which mimics some of the graphical spectacles but in reality is something completely different, dependant on the capabilities of the user's game station.

One of the more new features which have been added to games is the ability to process an effect with a real-time renderer. This means that the effect is not created beforehand but is processed directly when it happens, thus being more unique and responsive to the game environment, but also a lot heavier on the processing unit of the game station. Connecting this to the phrase described above, an interesting comparison arises. Is a real-time rendering of an effect really that much more desirable than one created beforehand compared to the added stress to the game station which easily can result in a drop in frame rate within the game? As an eager consumer of video games myself, I have through my interactions on the internet come across a large variation of opinions on this matter, and often these things can result in a lot more than just graphical annoyance. Other results such as personal irritation, game stations overheating or drop in performance by the user within the game are not uncommon descriptions of consequences of wrong graphical settings. This happens because some users simply swear only to use the best settings provided within the game, without accepting the limitations of their game station.

There are a number of theoretical solutions to this. However, no game developers nor hardware producers are not interesting in limiting themselves to accommodate the user's financial capabilities of upgrading, so the issue must lie somewhere in the users mindset.

I will in this project attempt to highlight what technologies within games that are currently pushing the boundaries of both the user's perception and also the hardware producers. What effects are most riveting and stunning to the user, and what effects are easily replaced with something more acceptable towards older hardware without the user actually noticing? After finding the most optimal effect to recreate and perform a test on, I will try to recreate a scenario where multiple versions of the same effect will occur, and then let a group of users experience this variation and observe their reaction. This observation will be the foundation of a statistical analysis

between various graphical effects and their fallout towards the user, and finally answering the question; are the newest graphical settings really a necessity for the user, or are they simply only being used because the user is aware of their availability, thus something letting the user handicap him or herself as a result of this.

Readers Guide

It is advised that the reader has followed a course in image processing or 3D animation at Aalborg University, or at least a similar course at another research facility. As well as the reader should have a small amount of knowledge about video games and terms used within that field.

Project Angle

In this project it has been chosen that the analytical angle will not be that of a hardware tester, performance analyst, game critic or anything of the likes. Those are not in the interest of the problem, and analysis will solely be performed with the angle of the user's perception towards video game graphics and effects.

Pre-analysis

FIX ALL PLACES WHERE YOU REFER TO SMOKE AND FIRE AS THE SOLUTION, INTRODUCTION SWEARS TO EFFECTS, NOT TO SMOKE AND FIRE!!

There will be three stages to this chapter. Initially the overall problematic will be described along with any possible hypothesis. Following up on that, the overall problem will then be dissected and the various elements of it will be examined. First a series of explanations, definitions and relevant theories regarding each of the elements will be researched in order to gain the sufficient knowledge, ensuring that there are no loose ends or gaps in the necessary knowledge needed before really engaging in a solution to the problem presented. Finally, an investigation towards any relevant or similar researches will be conducted. Any results found from researches that are examining somewhat the same area will always have an interest, and can help this project come to a conclusion more easily.

All this is done to for two purposes, two gain sufficient overall knowledge of the problem so that each element can be isolated and evaluated for its relevance towards a possible solution. Secondly, this is to be used as relevance and knowledge for delimitation purposes. The point is to get the overall problem narrowed down to something tangible which does not span over too many variables or different angles.

The Problem

In modern computer graphics both in movies and in games, there exist various ways of creating and portraying effects to the viewer, all which have strengths and weaknesses that makes them more or less appealing towards the various platforms in which they can be used. One of the major issues which separate these methods is the computers performance capabilities. Some techniques can provide stunning effects, but result in a very heavy load on the hardware used to produce it.

Some very common effects in both games and movies are effects such as fire and smoke. This is however where the problematic starts to arise. Smoke and fire are often used very extensively, and thus becomes an expensive thing to produce, both economically for when creating real fire and smoke in movies, but perhaps more interestingly when used as a visual effect in both movies and games. There exist solutions such as particle and fluids systems which can simulate very realistic replications of these effects, but they are computationally heavy and often not suitable for real-time rendering. So the question is; are those systems the best solution to create these effects, or are there any solid alternatives? Movies like Lord of the Rings 1-3, which are as of 2011 still considered fairly state of the art are known to use other options such as imposters and sprites which mimic real 3D effects, but on the contrary are consisting of simple 2D images which are much easier to render and produce. (## ref til forklaring) They do this even though they are not running them real-time. So why do they do that, are real 3D effects not more desirable in every situation, especially when within your money and hardware range? So the final question used as the initiator for this pre-analysis is:

To what degree can the creation of virtual effects such as smoke or fire be simplified while still maintaining the same graphical satisfaction towards the viewer? (REMEMBER TO ANSWER SOMEWHERE WHY FIRE AND SMOKE)

As *Thomas Akenine-Möller, Eric Haines* and *Naty Hoffman* explains in their book *Real-Time Rendering*:

> *"There is no single correct way to render a scene. Each rendering method is an approximation of reality, at least if that is the goal."* (1)

This is important to keep in mind. Some methods might provide a much higher possibility of creating realism or perhaps quality, but that does not make the method more correct. A selection of methods for creating effects will be briefly explained in chapter (## REF).

Hypothesis

As a supplementing basis for this analysis, it is the belief that certain effects can, within certain boundaries, be changed to a similar effect that uses a more hardware tolerant method, and still provide the same amount of perceived quality for the viewer.

Methods for Creating and Portraying Visual Effects

There are a few different ways of creating realistic effects, followed by a few different ways of portraying them inside a production. The two main methods have been categorized by me as real 3D renders, and image based renders. In this chapter these ways of creating effects will be examined, along with how they can be portrayed inside the virtual environment.

Real 3D Rendering

The most obvious way to create effects can be compared to the most obvious way of creating a regular 3D model. Models are created in 3D using polygons, and effects such as fire, smoke and the likes are created using fluid or particle systems which have inbuilt mathematical formulas that tells the fluid or particles how to behave, so that is mimics the real world effects as much as possible. This is done so that the effects are created in a live 3D render, making it a very versatile use, and enables the fluid or particle system to interact easily with other objects or forces within the scene.

The upside of this method is that the systems are very dynamic, responsive and adaptive towards the environment that they are put into, and the forces which they are affected by, if any. Another positive element of these types of renders are that they can be viewed from any angle or orientation and still look credible or maintain the same quality. The downside of this method is that it can be computationally heavy on the hardware. If the systems become too complex or too detailed, the computation time becomes proportionally higher, making the render and simulation time (##reference til analysis hvor jeg beskriver det her) of the simulation very long. This is especially a problem for games where these effects have to be rendered in real-time. *Thomas Akenine-Möller, Eric Haines* and *Naty Hoffman* also describes this in their book called *Real-Time*

Rendering, though they talk about polygon models and not particle or fluid systems, however the problematic is the same. They write:

> *"Modeling surfaces with polygons is often the most straightforward way to approach the problem of portraying objects in a scene. Polygons are good up to a point, however. Image-based rendering (IBR) has become a paradigm of its own. As its name proclaims, images are the primary data used for this type of rendering. A great advantage of representing an object with an image is that the rendering cost is proportional to the number of pixels rendered, and not to, say, the number of vertices in a geometrical model."* (1)

In what they write here the number of polygons and vertices can be compared to the amount of fluid or particles released in a system. The description here is actually a possible solution to the problematic which is described in the introduction of this report, or more accurately, a way to minimize the problem by using alternative methods of portraying these effects. One possible method of doing this is what is referred to as IBR.

Image-Based Rendering

Heung-Yeung Shum, Shing-Chow Chan and *Sing Bing Khan* writes as the introduction for their report titled *A Review of Image-Based Rendering Techniques*:

> *"Image-based rendering (IBR) refers to a collection of techniques and representations that allow 3D scenes and objects to be visualized in a realistic way without full 3D model reconstruction. IBR uses images as the primary substrate."* (2)

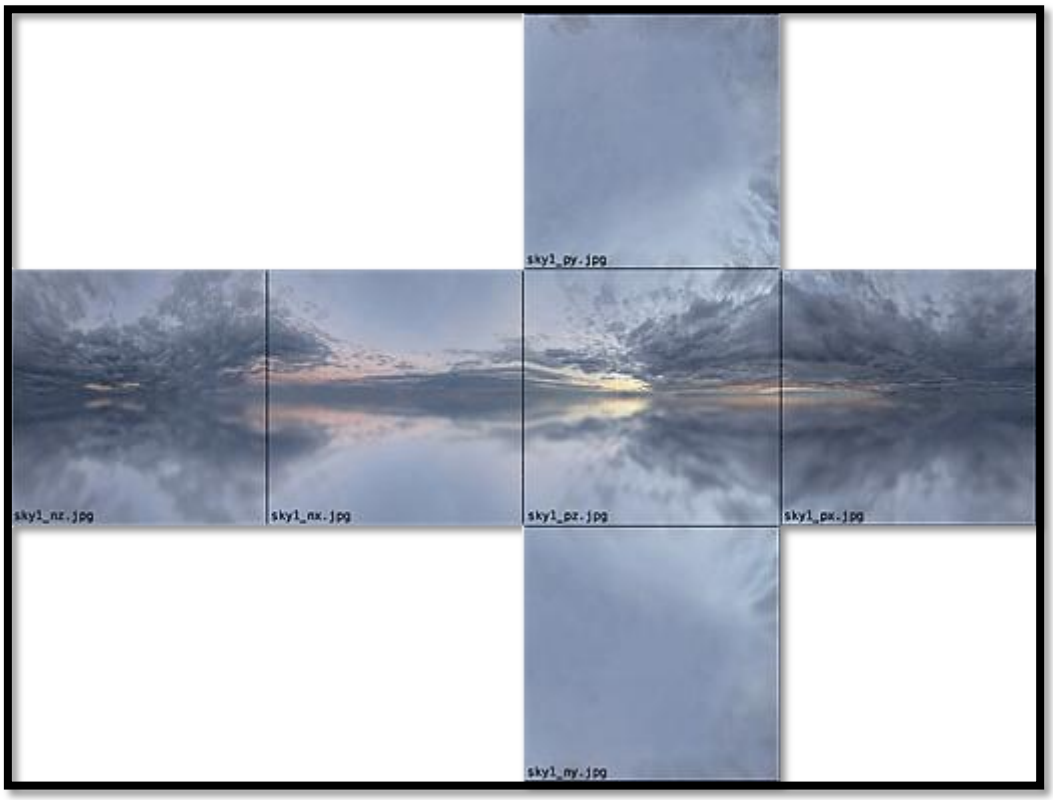
This description is very accurate and does not only describe what their book is about, but provides a good description of what IBR is about in general. There exist a vast amount of methods within the boundaries of IBR, but the overall idea is that objects are replaced with images, in one form or another. These images then imitate either real world effects or 3D effects, but portray then on a 2D image. Specific methods within this category can be skyboxes, sprites, billboards and imposters. All of these are more or less the same, but with small variations of their functionality. In the following section a small walkthrough of their functionality will be described in order to determine the relevance of each method to the solution of this projects problem.

Skyboxes

Skyboxes are not as much a method for displaying effects as it is a method for portraying overall environment maps. Skyboxes take advantage of distant objects lack of movement when the viewer is moving around. *Thomas Akenine-Möller, Eric Haines* and *Naty Hoffman* describe a scenario in their book based on mountains being viewed from a distance:

> *"... A distant mountain itself does not normally look appreciably different if you move a meter, or even a thousand meters. It may be blocked from view by nearby objects as you move, but take away those objects and the mountain itself looks the same."* (1)

Environment maps can be used for various things, but the focus here is on its ability to act as the environment of a scene by using a texture mapped on an object surrounding the scene. The use of skyboxes becomes relevant when you need to portray a very distant environment, this can be anything and can even include effects or the likes, and the only limitation is that it is something which is supposed to be far away. Very distant objects does not suffer the *parallax (##)* effect. The parallax effect only occurs on objects which change depth and orientation when we move past or around them. Distant objects are only affected by this in a very limited sense, which enables the use of skyboxes.



Figur 1 - http://www.cafu.de/wiki/_media/textures:skytut_1s.gif

Figur 1 shows what a potential skybox could look like. It is consisting of six squared images taken in a panoramic fashion, making each of them work as one side of a squared box in which a sky scenario will take place. The images are connected such in a way that makes it hard to notice the edges of the box when it is wrapped with textures.

Skyboxes is not particularly a way to portray effects and is not described as a way to portray effects, however it is important to notice that the possibility of doing that is there, and it is a viable way of portraying effects with less render time as opposed to real 3D renders. Effects can be put into the skybox's texture; this becomes especially valid when the amount of time where the skybox

is viewed at a time is limited, so that the change in the visual effect is close to zero, or if the skybox's texture is somehow animated to portray the effect.

Sprites

When talking about sprites we come closer to a render type which is much more specific towards the possibility of portraying effects. A sprite is basically a 2D image which can be placed anywhere on the screen, image a mouse curser or a dialogue box within a game. Sprites are also not limited to being squared as parts of the image in most render engines can be portrayed as transparent.

Figur 2 - http://www.geekologie.com/2008/06/23/sprites-12.jpg

Figur 2 shows a possible use of sprites; here the sprites are placed in a real-world scenario. Sprites can contain Z-depth values and as such can be placed behind other objects if the sprites Z-depth value is larger than the scene objects value, as they are done in the picture in Figur 2.

Sprites can also be created as images portrayed on a polygon surface. These sprites have more uses and can be skewed, enlarged and transformed accordingly to what it is supposed to portray. Sprites can also be animated by using creating a number of sprites portrayed in succession. This is where things get really interesting, let's return to the smoke and fire idea. Portraying a small distant fire or smoke cloud can easily be portrayed through an animated sprite, especially if it is out of reach to the user, and only can be viewed from that distance. This scenario changes when the viewer actually can move closer or around the sprite and becomes a bit more problematic. The sprite then needs to align itself towards the viewer in order to maintain the same appearance. However if the sprite is portraying a 3D objects, the illusion fades as the user moves around the objects or effect, but the appearance does not change accordingly. This is especially obvious when viewing objects or effects with a clear perspective, like a cube, a house or similar.

Billboards

Billboards or billboarding is not very different from using sprites and consist more or less of the same technique. However billboards take use of polygonal content to portray their images. The basic principle of billboarding is to take a polygonal plane and plant a texture on it, much like a sprite, but what makes the difference is that billboards change their alignment so that the plane always faces the viewer or the camera. This is called billboarding. (1) *Thomas Akenine-Möller, Eric Haines* and *Naty Hoffman* lists a number of objects and effects which are often created with billboards, particularly they mention the use of smoke and fire, as I did earlier in this report. They write:

> *"Billboarding, combined with alpha texturing and animation, can be used to represent many phenomena that do not have smooth solid surfaces. Vegetation, especially grass is an excellent candidate for this type of technique. Smoke, fire, fog, explosions, energy shields, vapor trails, and clouds are just a few of the objects that can be represented by these techniques."*

In Figur 3 a number of billboards portraying a fire can be seen, this is taken from the game *Counter-Strike* from Valve. The fire looks great and realistic; however it is easy to notice that these are billboards, especially if you notice the clipping between the fire and the wooden box in the right side of the screen.

Figur 3 - http://www.igniq.com/reviews/explosion%20fire230904.jpg

Screen-Aligned Billboard

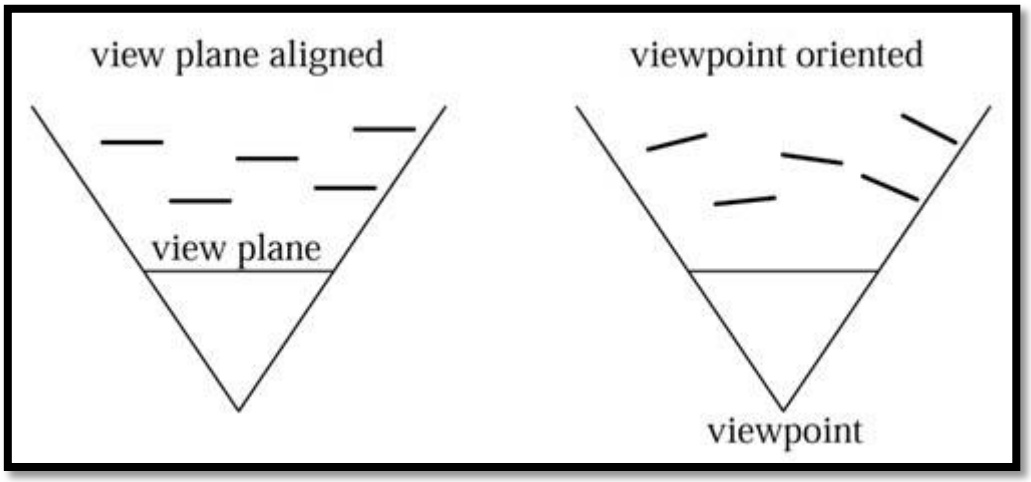
The most common billboard is a *screen-aligned billboard*. This type of billboard is a lot like the standard two-dimensional sprite; here the image is always parallel to the screen. This means that the direction from the billboard to the camera is constant, as well as the up direction is constant, as

they are both defined by the placement of the camera in the scene. The last direction is determined by finding the cross product between the two known directions or vectors as they are called.

This is a good way to show elements like text, heads up displays (HUD) or other elements which needs to be constantly facing the screen, while also maintaining the same up direction. (1) The reason for this is that these types of elements are not presenting objects in respect for the worlds up direction but with respect for the cameras up direction.

World-Oriented billboard

World-oriented billboards are different from screen-aligned billboards only because of the up direction in the billboards general orientation. Here the objects are rendered with respect to the worlds up direction and not the cameras up direction as screen-aligned billboards are. This is basically the only difference; these images still aligns themselves towards the camera but the up direction of the billboard remains constant towards the world and not towards the camera, often used for objects like clouds or lens flares. A small downside of using world-orientated billboards is that when viewing sprites that are of a little larger size the object can be skewed when aligned with the view plane. A solution to this is to make the direction vector point towards the viewer's position, so that all sprites are aligned towards the viewpoint.



Figur 4 - http://www.flipcode.com/archives/billalgn.jpg

Figur 4 shows an image from the book *Real-Time Rendering* which perfectly portrays this issue. When the sprites are small, this issue can partly be ignored as the skewing then becomes harder to notice, however when the sprites are above a certain size the viewpoint oriented solution has to be implemented. (1) The difference between these two types of billboards are not noticeable when the camera moves around the objects, moves up or down or moves closer or further away, but only noticeable when the camera rolls around the view axis.

Axial Billboard

Axial billboards are different from world-oriented billboards and screen-aligned billboards (also sometimes referred to as point-billboards), as they are only able to rotate around some fixed angles, while the last angle is locked in relation to the world. This is a good technique to portray

objects such as grass, trees, general vegetation, and other cylindrical objects or similar which can be found in a scene.

Figur 5 - http://imageshack.us/photo/my-images/441/wowscrnshot030708221112jo0.jpg/sr=1

Figur 5 shows a screenshot from the game *World or Warcraft* released by *Blizzard Entertainment*. (3) In this game both the grass and the crowns of the trees are consisting of axial billboards. However the grass rotates around the Y-axis to constantly face the viewer, the tree crowns do not, and they are fixed in both Z and Y direction, but are consisting of two or more billboards which are crossing each other.

Imposters

Billboards are often referred to as imposters, however imposters are slightly different. An imposter consists of real 3D objects which is rendered on an off-screen image, and then saved as a texture which is a placed on a billboard within the game; this is done to improve performance.

Imposters also have downsides though, one of them is when the orientation or distance changes, either the illusion breaks because of the different viewing angle, or the illusion breaks because of the pixels growing too large when the imposters comes too close to the viewer. Imposters can circumvent this by setting a threshold, like a maximum angle or maximum distance in which the viewer can move, and once this threshold is passed, the imposters are re-rendered.

Imposters can also be pre-rendered and cached, this may be classified as a billboard but that is not of importance. What is important is to recognize this method as being very clever. A pool if

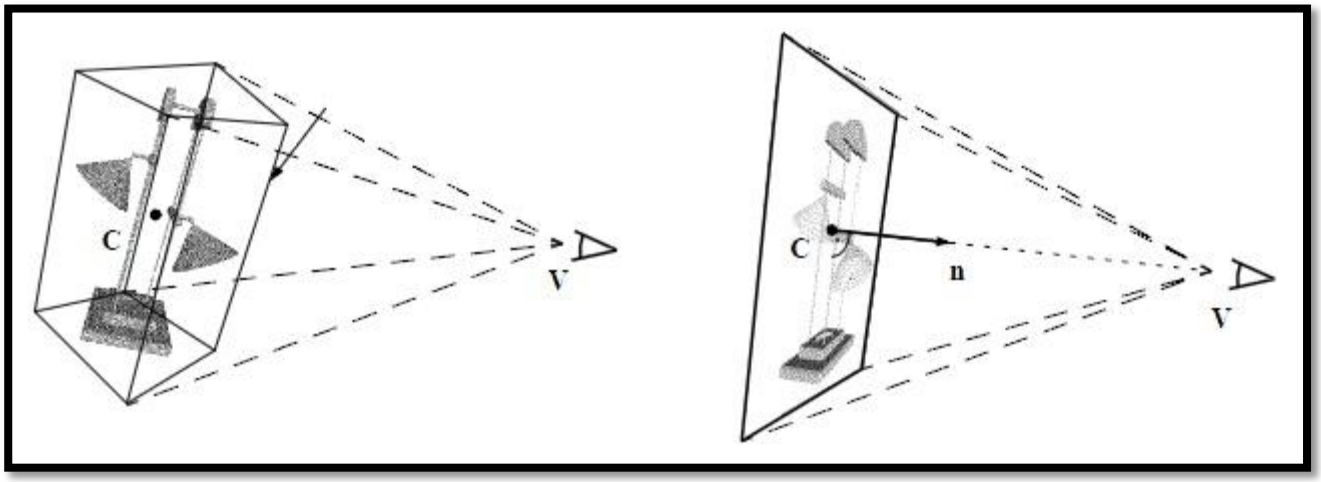
different angles of the same animation are then rendered, and the billboard or imposter is then portrayed differently, depending on which angle or distance it is viewed from. The billboard or imposter simply contains all methods. This might be larger in data size, but a lot faster in frame rates.

Related Studies

Here you are going to describe all the similar researches that has been conducted, both regarding imposters vs. 3d, but also the studies of creating realistic fire no matter what the method has been, examine what they do and what issues they describe, use this to extract the most interesting issues and use them to define the final problem after all of this has been explained and concluded in the end.

A Three Dimensional Image Cache for Virtual Reality

This report might not be exactly categorized as a related study, but it was one of the first studies to introduce the concept of imposters, as because of that it has been found interesting and somewhat relevant as a knowledge database for this report. It was written by *Gernot Schaufler* and *Wolfgang Stürzlinger* in 1995, (4) and presents the use of imposters to maximize frame rates in virtual environments. They present impostors and 3D virtual caches as a possible solution to this problem, which is directly linking to the problem in this report, and thus an interesting read.



Figur 6 – Screenshot from (4)

Figur 6 shows an image from their report, here a visual interpretation of the concept is presented and afterwards carefully explained both conceptually, but also mathematically and technically, as how to create, use and implement this method.

The use of Imposters in Interactive 3D Graphics Systems

In a report by *Kenneth Rohde Christiansen* called *The use of Imposters in Interactive 3D Graphics Systems*, (5) he wishes to examine the use of imposters to maintain frame rates. He bases his analysis on the concept created by *Gernot Schaufler & Wolfgang Stürzlinger* in their report called *A*

Three Dimensional Image Cache for Virtual Reality. In this report they introduce what they refer to as *dynamically generated impostors*, which is exactly what has been described earlier in this report.

In *Kenneth Rohde Christiansen's* report, he examines whether or not it is viable to use imposters for rendering distant occurrences of a 3D object, while close occurrences are rendered in real 3D, and whether or not this difference is noticeable.

The basis for his study reminds a lot of the basis and problematic described for this project, and is therefore interesting. He describes the process of his report as:

> *"In this paper we have looked at a method for speeding up the rendering of complex scenes in order to improve the frame-rate and thereby the user experience. The idea was to use so-called imposters and reuse these over a series of frames. The imposters were re-generated when the diverged too much from the original object due to the user/camera having turned more than a given degree."* (5)

And then concludes his report by saying:

> *"In order to look at the usability of the method we did a proof-of-concept version and conclude that the negative visual impact is neglectable. Dynamic generated imposters are an easy-to-implement method with almost no visual impact that with a good implementation has the ability to speed up the rendering of complex interactive 3D environments."* (5)

There are some differences between his study and this study, one of them is the fact that he is talking about 3D objects, while this report focuses on effects, however his study is very interesting and is overall strengthening the hypothesis which has been introduced in this report despite the differences.

Animated Impostors for Real-time Display of Numerous Virtual Humans

This report was published not long after *Schaufler* and *Stürzlinger* published theirs. It is published by *Amaury Aubel, Ronan Boulic & Daniel Thalmann.* (6) Their study is not about whether or not the presence of impostors is distinguishable compared to the use of real 3D models, but whether or not it is a viable method for improving the frame rate of virtual environments containing a large amount of the same objects. This might not be within the same problem range as this report, however it is directly related to the same field of study, and this angle is equally important to include when gathering relevant knowledge through related studies.

What can be concluded from this study is their ability to use imposters for an obvious improvement of the frame rate, while at the same time maintaining a certain standard of the graphical quality. Finally, it is important to notice how they acknowledge *Schaufler's* and *Stürzlinger's* work and studies as one of the elements of their report.

Real-Time Cloud Rendering

This is a report written by *Mark J. Harris & Anselmo Lastra.* (7) They present a method where a particle system is used to generate a cloud like effect within a game. This is an interesting study as

it introduces a particle system to generate a virtual effect, but also stumbles upon the problem which can arise when particle systems are becoming too heavy on the hardware, which forces them to find an alternate solution.

The concept of this report is to use particle systems to create a cluster of smaller billboards which as the end product looks like a somewhat realistic cloud. The particles are pre-generated, which means that within the game, only the shading of the clouds is rendered. A problem arises when their scene becomes too complex, and the frame rate starts to suffer. The decision to implement imposters is then taken, and they put up two solutions: either to use pre-computed imposters which contain images of the object generated from multiple viewports, or to use dynamically generated imposters which are then generated and rendered when only when needed. The latter is chosen.

They result is very positive, and even in scene where the imposters has to be redrawn every frame, the frame rate and performance is still better than without the imposters. This is an interesting report, both because of their great results, but also because they discuss and reflect on several different uses of imposters in their argumentation, before implementing a solution.

Real-Time Tree Rendering

This report is written by *I. Remolar, C. Rebollo, M. Chover & J. Ribelles*. (8) It focuses on real-time rendering of trees and vegetation, and is based on a problem that arises when complex models contains too many polygons and thus slows down the system, limiting the frame rate. They focus their research on methods which allows them interactively either to lower the resolution of the polygon models, depending on the distance, or to completely replace the model with an dynamically generated imposter. They implement a clever use of both, but particularly their use of imposters is interesting. They create a movable imposter, which gradually moves through the polygon model, thus replacing more and more of the actually model with the imposter, the further away from the object you get. Again, a study is shown where the use of imposters can improve the frame rate, and in this case, with great focus on not reducing the quality of the scene.

Spherical Billboards and their Application to Rendering Explosions

This report is written by *Tamás Umenhoffer, László Szirmay-Kalos & Gábor Szijártó*. (9) It does not share any of the same specific problems as this report; however it is an interesting read as it clearly shows billboards obvious ability to portray effects such as fire and smoke, and does it well. It also presents a more modern way of using billboards. They have developed a new method which uses multiple billboards on particles to render a volume billboard that provides a better graphical finish than using regular billboards. This is particularly useful as it shows that using billboards or imposters is not an outdated technique and can be altered and used in various ways to portray great visuals.

Realistic Real-Time Rain Rendering

It is written by *Pierre Rousseau, Vincent Jolivet and Djamchid Ghazanfarpour.* (10) Their report strays a little further away from what the initial problem in this report is about. It is included as a source however, as it shows a very clever use of imposters when creating realistic rain for real-time

rendering. Their technique is to model and use raindrops within the virtual environment, but to create the refractive and reflective characteristics of the raindrops as if they were really in the scene; they use an off-screen imposter which has a texture image of the scene to generate that. The alternative would be to raytrace each drop within the actual scene, which is computationally heavy, and not possible in real-time.

SOME OF THIS WILL BE MOVED TO ANALYSIS I THINK.

- Talk about Particle rendered billboards, and fully rendered billboards. The difference is important and interesting to my project.
- Include earlier document about various changeable elements and factors that can be altered in the production, use it for the final problem.

(REFERENCE TO FIRE IN THE VULCAN DEMO)

Changeable Elements

When creating an object or an effect within a scene, there are various elements that are also eligible to be altered. The goal of this section is to list and find the elements which are contributing to the biggest difference in how effects in graphics are viewed, but also (and perhaps more importantly) to isolate a few elements that are going to be the focus of this project. This is in order to keep the varying elements to a minimum so that the precision of the results can be maximized.

Lighting

Lighting is one of the obvious elements which can easily change the whole appearance of a 3D scene, and changing the variables to turn a bright scene into a dark and dim lighted scene is very easy.

Light and bright

A very bright scene is also a very revealing scene. Small effects, textures and other elements are all lit up and are easy to be seen, so concealing an effect or an object in a much lit scene can be difficult, however it is very effective when it is the desired result that the viewer is supposed to notice these elements. However in a lit scene objects which emit light themselves can blend in with the environment very well, fires, light bulbs and the likes do not contribute with much when the scene is bright and well lit.

Dark and dim

A dark scene does, naturally, the opposite of what a very bright scene does. It limits the level of detail which can be seen on various objects in the scene, and conceals other elements simply by the lack of light. However illuminating elements and generally elements which emit or manipulate light in the scene are much more important to the scene, and easily take up a lot more focus when the scene is low on light.

Movement/viewport

Generally this means how the objects are viewed when taking direction, speed of the camera and rotation into account. Objects viewed from a static camera can more easily be replaced with imposters or billboards as they are only required to portray the object from one angle and in one type of render. However if the camera moves, or the object moves, the objects animation should change accordingly to accommodate for this change.

Theme

The theme of the setting might sound like something which does not exactly translate to any importance of how 3D effects, animations or objects are viewed. However if the theme of the scene is a highly fictive fantasy theme, it might be more credible to present some slightly skew or awkward animations as the amount of realism required to create belief might be lower in these type of settings, compared to a scene striving for real-world realism.

Game/Movie

The discussion about game versus movie which is being proposed here is not to be confused with the amount of realism when comparing computer game effects with those in a movie. It is regarding the amount of control the viewer has over the scene, and if the ability to control the game yourself proposes any significant change in how effects are viewed, compared to viewing a pre-recorded scene from the same game, as a movie. This also takes the changes of viewport into account, when using a movie instead of a game, you can predefine the viewport render angles and that way circumvent some of the difficulties these can create when using a real-time renderer.

Scenery

This element leans very heavily up against the element of theme, but is still slightly different. Instead of choosing a specific theme to follow, certain scenery is used. Choosing a forest scene compared to for example a harbor or a factory does make a difference in how the user views objects. However one of the more significant parts of scenery which can make a great deal of difference towards the user is when dynamic scenes are used. A fixed billboard of an effect can be useful in a fixed scene, but if the elements in the scene moves around or are broken up in some way, it would seem necessary to create a dynamic effect that accommodates for that, so that the illusion is not too obvious. A lot of the newer games take advantage of technologies which incorporate gravity and breakable objects, such as the PhysX technology, (11) thus this is becoming more and more relevant. An example of the uses of the PhysX technology can be found in the game *Crysis 2: Be the Weapon* from *EA Games*. (12)

Quality

Quality does not necessarily propose any difference between for example imposters or a real 3D animations, however if the quality is low, and the effect looses credibility, it suggestively does the same damage as if a flaw was noticed in the presentation of the effect. This element might not be a candidate for a testing variance; however the quality should be high enough to avoid this pitfall. So when testing there is no confusion that the results of the test are from the elements which are being altered, and not as a cause of poor quality settings in the render.

Distance

The last element which is being highlighted is distance. Right now many games use LOD (Level of Detail) to minimize render times, and thus show low quality render in the distance, and high quality renders when portrayed close to the camera. This is very effective and can be used both for imposters as well as real 3D objects, and can even be combined (## REF TO RELATED WORK SECTION). Imagine imposters being shown at distance where the angle and size of the object changes very slowly and then 3D real animations will be shown when the camera is close to the object.

Selection

Using all these variables in a test is not desirable; it creates a very large amount of possible variations, and makes it extremely difficult to pinpoint which change is causing what result. So delimitation is a necessity in order to be able to receive proper test results. Additionally it needs to be clarified that several of these elements overlap, such as theme and scenery, and movement/viewport and distance, so even though the desire to keep the variables to a minimum is desired, they quickly grow to an undesired amount.

Delimitation

Here the rendering methods, along with the related researches will be concluded and summarized. All interesting points make up a whole, which is narrowed down to the most basic interesting points, which then lays the foundation for the final problem, which should be a lot more specific.

Final Problem

Here the final problem is described as narrow and specific as possible. And then the elements which need to be further analyzed within this problem are highlighted, leading the way for the upcoming in depth analysis.

Analysis

- Specification of problem: billboard clusters, particle billboards or fully rendered billboards. Pre-rendered or dynamically rendered billboards.
- Simulation techniques: Fluid, particle, others?
- Fluid algorithms and types (grid, particle, etc.)
- Rendering techniques
- Programs and utilities for creation
- Definitions: Quality

Design

INCOMPLETE

Implementation

INCOMPLETE

Test Methodology

INCOMPLETE

Test

INCOMPLETE

Results

Conclusion

I have chosen not to include any sections beyond the pre-analysis as those are far from complete and would not benefit at all.

Discussion

Perspective

Future Work

References

Bibliography

1. **Akenine-Möller, Tomas, Haines, Eric and Hoffman, Naty.** *Real-Time Rendering.* s.l. : AK Peters, 2008.

2. **Schaufler, Gernot and Stürzlinger, Wolfgang.** *A Three Dimensional Image Cache for Virtual Reality.* Linz : s.n., 1995.

3. **Christiansen, Kenneth Rohde.** *The use of Imposters in Interactive 3D Graphics Systems.* Groningen : s.n.

4. **Aubel, Amaury, Boulic, Ronan and Thalmann, Daniel.** *Animated Impostors for Real-time Display of.* Lausanne : s.n.

5. **Harris, Mark J. and Lastra, Anselmo.** *Real-Time Cloud Rendering.* Chapel Hill : EUROGRAPHICS, 2001.

6. **Remolar, I., et al.** *Real-Time Tree Rendering.* Castellón : Springer-Verlag, 2004.

7. **Umenhoffer, Tamás, Szirmay-Kalos, László and Szijártó, Gábor.** *Spherical Billboards and their Application to Rendering Explosions.* Budapest : s.n.

8. **Rousseau, Pierre, Jolivet, Vincent and Ghazanfarpour, Djamchid.** *Realistic Real-Time Rain Rendering.* s.l. : Computers & Graphics, 2006.

9. **nVidia Corporation.** nVidia PhysX. *nVidia Web site.* [Online] 2011. http://www.nvidia.com/object/physx_new.html.

10. **Electronic Arts Inc.** Electronic Arts Crysis 2 front page. *Electronic Arts Web site.* [Online] 2011. http://www.ea.com/crysis-2.

11. **Blizzard Entertainment Inc.** World of Warcraft front page. *World of Warcraft Web site.* [Online] 2011. http://eu.battle.net/wow/en/.

12. **Heung-Yeung, Shum and Kang, Sing Bing.** *A Review of Image-based Rendering Techniques.* s.l. : Microsoft.

Appendix