



Bachelorarbeit

im Studiengang Audiovisuelle Medien

Performance Evaluation of Evotis within a Visual Effects Environment

vorgelegt von Tim Klink

an der Hochschule der Medien Stuttgart

am 13. August 2018

zur Erlangung des akademischen Grades eines Bachelor of Engineering

Erst-Prüfer: Prof. Katja Schmid (Hochschule der Medien)

Zweit-Prüfer: David Harter (Scanline VFX GmbH)

SCANLINE VFX

GO GHOST

CHAO2GROUP

Eidesstattliche Erklärung


„Hiermit versichere ich, Tim Klink, an Eides Statt, dass ich die vorliegende Bachelorarbeit mit dem Titel: „Performance Evaluation of Evotis within a Visual Effects Environment“

selbstständig und ohne fremde Hilfe verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen der Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken entnommen wurden, sind in jedem Fall unter Angabe der Quelle kenntlich gemacht. Die Arbeit ist noch nicht veröffentlicht oder in anderer Form als Prüfungsleistung vorgelegt worden.

Ich habe die Bedeutung der eidesstattlichen Versicherung und die prüfungsrechtlichen Folgen (§26 Abs. 2 Bachelor-SPO (6 Semester), § 23 Abs. 2 Bachelor-SPO (7 Semester) bzw. § 19 Abs. 2 Master-SPO der HdM) sowie die strafrechtlichen Folgen (gem. § 156 StGB) einer unrichtigen oder unvollständigen eidesstattlichen Versicherung zur Kenntnis genommen.“

Fellbach, 13.08.2018

Ort, Datum



Unterschrift

Abstract

Due to the rise of visual effects in film, TV, commercials, games and VR over the last decade and increased competition within the VFX industry it is integral for any major VFX company to always lead through innovations that streamline their production.

However, the last big change to intermediate file types came with deep image technology, invented 18 years ago. To determine whether Evotis, a new rendering-sample based non-uniform image technology, has the potential to be the next improvement to the intermediate workflow several performance tests were conducted, benefits and disadvantages discussed, and potential improvements proposed.

Kurzfassung

Wegen der durch Filme, Fernsehproduktionen, Werbung, Spiele und VR stetig gestiegenen Nachfrage nach visuellen Effekten und der dadurch gestiegenen Konkurrenz ist es für große VFX Firmen von grundlegender Bedeutung stets durch das Nutzen der neuesten Innovationen ihre Arbeitsweise zu verbessern.

Jedoch ist die letzte bedeutende Änderung der als „Intermediate“ genutzten Bildtypen, die Erfindung von deep-Bilder, vor 18 Jahren gewesen. Um zu klären ob Evotis, ein rendering-sample basiertes, und nicht gerastertes Bildformat, das Potenzial hat die nächste Stufe in der Entwicklung des „Intermediate Workflows“ zu sein wurden verschiedene Performance Tests durchgeführt, die Vor- und Nachteile genau beleuchtet und mögliche Verbesserungen vorgeschlagen.

Table of Contents

Eidesstattliche Erklärung	i
Abstract	ii
Kurzfassung	ii
Table of Contents.....	iii
List of Figures	v
Glossary of Terms	vii
Acknowledgment.....	vii
1 Target Audience.....	1
2 Introduction	2
3 State of the Art	4
3.1 Flat Images.....	4
3.2 Deep Images	6
3.3 Workflow	8
4 Evotis	11
4.1 Sample Optimization.....	11
4.2 Capabilities and Implementation	14
4.3 Advantages of Evotis	17
4.3.1 Object Isolation	17
4.3.2 ID Mattes	19
4.3.3 Resolution Independence and Resizing	20
4.3.4 Refining Renderings	21

4.4	Disadvantages of Evotis	23
4.5	Future Possibilities.....	24
4.5.1	Precisely Specified Render Areas for Re-rendering.....	24
4.5.2	Viewing Angle Independence	25
4.5.3	Evotis Library	25
5	Performance Evaluation.....	26
5.1	Test Method.....	26
5.2	3D Test Results.....	27
5.2.1	Test Scene 1 – Simple Scene	28
5.2.2	Test Scene 2 – Motion Blur	33
5.2.3	Test Scene 3 – Fur	37
5.2.4	Test Scene 4 – Fur with Motion Blur	39
5.2.5	Summary of 3D Tests.....	41
5.3	2D Test Results.....	41
5.3.1	Initial Testing	42
5.3.2	Results from 2D Test 1	45
5.3.3	Results from 2D Test 2.....	46
5.3.4	Results from 2D Test 3.....	47
5.3.5	Results from 2D Test 4.....	48
6	Conclusion.....	49
7	Further Work	53
8	Bibliography	55

List of Figures

Fig. 3-1 - worlds first digital image (NIST, 2018)	5
Fig. 3-2 - Shadowsling comparison © Twentieth Century Fox Film Cooperation and Weta Digital Ltd. All Rights reserved	7
Fig. 3-3 - simplified workflow	9
Fig. 4-1 - sample optimization options within Evotis interface in Maya.....	12
Fig. 4-2 - sample optimization comparison.....	13
Fig. 4-3 - evoID interface	15
Fig. 4-4 - evoReformat interface	16
Fig. 4-5 - workflow examples	16
Fig. 4-6 - object isolation comparison.....	17
Fig. 4-7 - edge comparison	19
Fig. 4-8 - scale-up comparison.....	21
Fig. 4-9 - noise level comparison	22
Fig. 5-1 - rendering of 3D test scene 1	28
graph 5-1 - initial testing of different Evotis sample optimization options	28
graph 5-2 - zoom-in of last portion of graph 5-1.....	29
graph 5-3 - comparison of Evotis to flat and deep	29
graph 5-4 - numerical and relative time differences of renderings in different resolutions	30
graph 5-5 - file size comparison (log scale)	31
graph 5-6 - file sizes at different resolutions	32
Fig. 5-2 - rendering of 3D test scene 2	33
graph 5-7 - render time of motion blur scene.....	34
graph 5-8 - render time relation throughout resolution range.....	34

graph 5-9 - absolute and relative file sizes at different resolutions.....	35
graph 5-10 - detailed plot of $f(s)$ -values	36
Fig. 5-3 - rendering of 3D test scene 3	37
graph 5-11 - fur rendering results.....	37
graph 5-12 - absolute and relative file size comparison throughout range of resolutions	38
graph 5-13 - render times of long rendering scene	39
Fig. 5-4 - rendering of 3D test scene 4	39
graph 5-14 - absolute and relative file size comparison throughout range of resolutions	40
Table 5-1 - summary of 3D test results	41
graph 5-15 - render time and file size comparison for different Evotis options.....	42
graph 5-16 - performance data of first test without using evoReformat	43
graph 5-17 - performance data when using evoReformat	43
graph 5-18 - performance data of flat and deep rendering	43
graph 5-19 - absolute and relative render time comparison	45
graph 5-20 - absolute and relative render time comparison of 3D test scene 2.....	46
graph 5-21 - absolute and relative render time comparison of 3D test scene 3.....	47
graph 5-22 - absolute and relative render time comparison of 3D test scene 4.....	48

Glossary of Terms

AOV	-	Arbitrary Output Variable
deep	-	deep-data image, an image format that stores the rgba and depth information of every surface or volume point contributing to the value of a pixel within a rasterized 2D pixel grid, possibly hundreds of samples per pixel
flat	-	rasterized pixel-based 2D image with only rgba information per pixel
repo	-	repositioning of an image
rgba	-	color channels of an image (red, green, blue, alpha), can be referred to as a single channel or a combination, i.e r,g,b,a or rgb
VFX	-	Visual Effects
NaN	-	“Not A Number”, damaged non-displayable pixel value

Acknowledgment

This thesis was made possible by Scanline VFX, GoGhost and Chaosgroup. I would like to thank all these companies, and every single person involved in the decision to support me and this thesis. The support I received ranged from the job and help by Scanline, the access to private beta software, not even presented publicly yet, and lots of advice by GoGhost and, last but not least, free access to v-ray to be able to perform these tests by Chaosgroup. Thanks again for this.

1 Target Audience

This thesis is targeted towards visual effects professionals that possess a working understanding of visual effects work in general, the VFX pipeline, common workflows, computer graphics knowledge and the techniques used in the industry. Therefore basic aspects will not always be explained in detail and industry-specific vocabulary will be used, mostly, without separate explanations. A bit of basic vocabulary has been explained in the glossary, for further reference “Visual Effects in a Digital World: A Comprehensive Glossary of over 7,000 Visual Effects Terms” (Goulekas, 2001) can be used.

This work is meant to be an introduction into Evotis and its non-uniform rendering-sample based approach, an evaluation of its production-readiness and its capabilities in general. It can therefore be considered an early evaluative guide for Evotis.

2 Introduction

Due to the ever-increasing need for high-end VFX in movies, commercials and TV, tight deadlines and even tighter budgets, all VFX companies constantly optimize their workflows towards faster turnarounds, while keeping costs low and quality high. One key aspect to stay competitive for any VFX house is the ability to quickly react to changes, possibly last-minute, requested by either a client or a supervisor.

Due to the complexity of modern VFX work, and its pipeline, a change, especially if it needs to be addressed in the 3D department, can take a long time and therefore prolong a facilities turnaround, which in turn depletes the possible margin. The problem of turnaround times and pipeline rigidity will be addressed in *chapter 3.2 Workflow*.

To better cope with change-requests more responsibility and variability is constantly being moved further down the pipeline, mostly from the 3D department into the compositing department. (Okun, et al., 2015) This started with rendering CG elements separately for them to be combined in compositing, only needing to re-render smaller portions of the image if mistakes were found or changes wanted. (Brinkmann, 1999)

The next step was to split the image into different render passes for different surface and light characteristics, like specular, refraction, diffuse, self-illumination and many more, which could be recombined in compositing using simple mathematical operations, but also altered in a more versatile way, greatly speeding up changes, at least to a certain point of variation. (Wright, 2010)

Another improvement in flexibility was achieved with the addition of AOVs (arbitrary output variables), sometimes also called tech passes, which enabled the compositor to

change more aspects of the 3D rendering within his software package by manipulating only certain characteristics of any given rendering. (Brinkmann, 2008)

The latest big technical advancement was the introduction of deep images, a new image type that was capable of storing multiple depth samples per pixel (Lokovic, et al., 2000), and thereby allowing more precise depth-related effects and many other features, which will be discussed in chapter 3.1.2 *Deep Images*.

The question that leading VFX companies have to ask themselves now to stay ahead, as deep is becoming a standard within the industry and even in mid-sized facilities, is: what will be the next big step?

One possible successor technology to deep could be the new Evotis system developed by GoGhost LLC in San Diego, a rendering-sample based non-uniform image format that aims at shifting even more control and flexibility towards the compositing department. The concept behind this system, its advantages and disadvantages will be covered in chapter 4 *Evotis*. (GoGhost, 2018)

To determine whether Evotis is a possible replacement for deep-, or even flat images, multiple performance and file tests, highlighting different key aspects in 3D rendering as well as in compositing, will be executed and evaluated in chapter 5 *Performance Evaluation*.

A look-ahead for needed changes and suggested further developments of Evotis, and a Conclusion will follow in chapter 6 *Conclusion*, with chapter 7 *Further Work* suggesting further research possibilities in the continued testing and implementing of Evotis.

3 State of the Art

In this chapter the two currently used image types within the VFX industry, flats and deeps, as well as a standardized and simplified workflow are shown, while each image types history, advantages and disadvantages are explained. Furthermore the reasons for rigid workflows are described.

Image type refers to the functionality and underlying architecture of an image, rather than its file format, for example: flat is the type, rasterized pixel image is the underlying technology, while .exr¹, .dpx², .jpeg³ or .png⁴ are some of the formats. Another common image type is the vector graphic, using non-rasterized vector information with .eps⁵ or .svg⁶ being two of the many available formats.

3.1 Flat Images

Flats are “a rectangular array of (...) values” (Rosenfeld, 1969). While they have progressed from simple 2 bit integer arrays, to 6, then 8 and now 32bit float arrays, flats on today’s VFX workflow are still true to the early definition of a digital image. Flats are used in every VFX facility for most of the intermediate work, usually as .exr files. They are the backbone of any workflow. (Brinkmann, 2008)

¹ OpenEXR <http://www.openexr.com/>

² Digital Picture Exchange ST 268:2003 - <https://www.smpte.org/>

³ Joint Photographic Expert Group ISO/IEC 10918-1 - <https://jpeg.org/>

⁴ Portable Network Graphics ISO/IEC 15948:2003 - <http://www.libpng.org/pub/png/>

⁵ Encapsulated PostScript

⁶ Scalable Vector Graphics W3C SVG - <https://www.w3.org/Graphics/SVG/>

Flats have been optimized and developed since the first digital image from 1957, shown in *Fig. 3-1*, as this was a flat, a simple pixel-raster image. (NIST, 2018)

An important step forward in the VFX flat workflow for 3D renderings was the addition of a depth channel, usually called Z, Z-depth or depth, enabling the compositor to manipulate any depth-related effects, like focus, haze or heat distortions much more efficiently and precisely. The Z-depth, originally called



Fig. 3-1 - worlds first digital image (NIST, 2018)

Z-Buffer, first invented to determine visibility of

objects in 3D renderings based on the position in depth in 1974 by Wolfgang Straßer (Straßer, 1974) and independently and shortly afterwards by Edwin Catmull (Catmull, 1974), who coined the name Z-Buffer, also marked the beginning of AOVs as we know them today. Though first only an internal calculation step for determining object intersections and visibility, it later became one of the first widely used render layer that wasn't a part of the visible image, and therefore only a tool for the compositor to use, marking one of the first important steps in handing down responsibility towards compositing.

The next important improvement to digital images, towards today's digital compositing, was the invention of the alpha channel in 1979 by Edwin Catmull and Alva Ray Smith (Smith, 1995). Due to the addition of an alpha channel a set of algorithms, defining blending operations, were introduced. (Wallace, 1981) (Porter, et al., 1984).

The last big step, conceptually, was the introduction of render passes, which, unlike AOVs, are part of the visible image. These render passes split the image into different object and light characteristics, for example: diffuse, reflection, refraction, specular and

self-illumination. The result, if these passes are recombined properly, is identical to the regularly rendered image, but due to the separated characteristics it was now possible to alter and manipulated the images in a more refined and controlled way, greatly adding to the flexibility in compositing. (Brinkmann, 2008)

From this point forward more AOVs and render passes were added, multiple images stored in one file, but no conceptual changes were introduced to flat images anymore.

3.2 Deep Images

Even though the base functionality of deep images, multiple depth-related samples per pixel, has been introduced in 2000 (Lokovic, et al., 2000), it still took until 2008 to be first used for feature film compositing at Weta¹, and again till 2012 to spread among most of the global players to be used on selected shots (Seymour, 2014). While most major companies nowadays included deep into their standard pipeline, many mid-sized and small outlets have not changed to include deep in any way, which is due to the big file size and intensive processing needed for a deep workflow to function in a beneficiary and cost-effective way. (Seymour, 2014)

The main practical advantage of deep images and deep compositing lies in holdout generation, especially for atmospheric renderings, as this used to be a time-consuming and difficult task, often requiring split-renderings of atmospherics in front and behind the subject or especially rendered holdouts demanding a dual-incrimination. With deep, and its depth-based merge capabilities, this now is a one-click task, allowing separate incrimination-renderings of subject and atmospherics.

¹ Weta Digital Ltd., Wellington, NZ - <https://www.wetafx.co.nz/>

Deep also allows the interactive creation of ID-mattes, as object related information can be saved in the depth-samples with only minor changes to the rendering set-up, enabling the compositor to create object based mattes.

One of the most widely known and innovative uses of deep compositing is a technique called “*Camera Space Volumetric Shadows*”, usually called “*Shadowsling*” for ease of use, developed at Weta in 2012, which uses deep PantaRay (Pantaleoni, et al., 2010) shadows and deep volume renderings to allow interactively generated volumetric shadows and god rays in compositing, rather than having to render them in a locked position from a 3D software package. (Hanika, et al., 2012) This enabled a much speedier turnaround, as any changes in the look of the shadows could be achieved in compositing, changes in the horses animation did not demand a re-rendering of



Fig. 3-2 - Shadowsling comparison © Twentieth Century Fox Film Cooperation and Weta Digital Ltd. All Rights reserved

the particle simulation, and, as seen in Fig. 3-2 – *Shadowsling comparison*, even a change in the direction of the light could easily be addressed in compositing, allowing for highly increased versatility. As shown in the three sub-images, the look of the dust rendering can be altered to accommodate any light source position possible, while correctly adapting and calculating the shadows, bounce lights, density changes and god rays.

The disadvantages of deep images are the processing-intensive overhead and the big file size, demanding longer render and processing times, compared to flats, thus, if not used in a beneficial and time-saving way, slowing down the workflow.

The performance differences between flats and deeps will be further assessed in chapter *5 Performance Evaluation*.

3.3 Workflow

The workflow, also called pipeline, will be briefly discussed to illustrate the problem of changes needing to be made in the 3D department compared to changes that can be made in compositing, the possible time being saved and the therefore resulting improved turn-around time. To accommodate all the processes that a VFX facility must cope with, the hundreds of shots and thousands of tasks, and still function in an organized way, the pipeline needs to be at the heart of any company, as it determines every procedure of every department, and therefore is one of the main aspects of the turnaround-times, and possible profit margin of the facility (Wright, et al., 2016). Due to its integral part the pipeline also must be dependable, but to be fully dependable at any given time it also can never be bypassed in any way, meaning every step in a process needs to be taken, as every step that follows will depend on it. To illustrate this rigidity the flowchart in *Fig. 3-3* shows a highly simplified workflow of all major steps necessary to accommodate a change that needs to be addressed in the 3D department. This graph was kept as short as possible, thereby neglecting a lot of intermediate steps necessary. The “change in 3D”-step was also kept as one step, as any change-request that goes beyond the shading, lighting and rendering department, e.g. modeling or animation changes, will most likely be out of scope of possibly being changeable in compositing

anyways, with or without Evotis. In most change-request cases, smaller fixes, not radically changed scenes, that need to be addressed in 3D and therefore pass through the whole workflow shown, the longest active-work-time (the time an artist or a computer, will need to work on a single step) usually is 3D rendering. Several hours per frame are the norm for high quality renderings, even on the most powerful render slaves. Added to this the time a 3D artist needs to incorporate the update, the internal 3D review process, the VFX-Supervisor review and the ingestion steps necessary, then this can easily add-up to multiple days. Therefore, as shown before, it is always the goal to bring as much versatility and freedom as far down the pipeline as possible. The time saved by not having to go back all the way to 3D, even if only applicable, hypothetically, for 10% of the shots, will make a difference for any facility.

On a big show it is common for a major company to have multiple hundreds of shots, for example around 700¹, assuming only half of them need 3D work, which in today's films would be quite a low figure. Sticking with the hypothetical 10% and assuming an average of

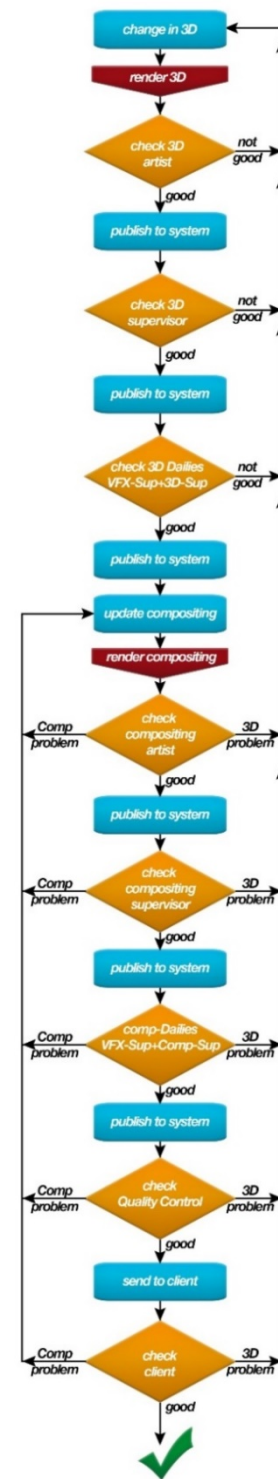


Fig. 3-3 - simplified workflow

¹ Scanlines number of shots on Justice League <https://www.scanlinevfx.com/about>

1.5 days per change request, as discussed earlier, the saved time will still be substantial:

$$\left(\frac{700 \text{ shots}}{2} * 10\%\right) * 1.5d \approx 53 \text{ days}$$

Even with these low assumptions a saving of 53 days is a figure no VFX company can ignore.

4 Evotis

The core concept of Evotis is a render-sample based intermediate image format, rather than a pixel-based one, thus creating new possibilities in the 3D rendering approach and in compositing.

In a pixel-based workflow any Monte-Carlo 3D renderer first generates an adequate number of randomly scattered ray samples to cover every needed pixel sufficiently and then combines those ray samples to calculate the pixels value. (Cook, et al., 1984)

The approach of Evotis is to keep those non-uniform rendering samples rather than the combined and rasterized pixel value and thereby preserving the information more precisely and avoiding any filtering algorithms that would deteriorate the images technical accuracy until later in the pipeline.

Within this chapter Evotis' characteristics, implementation, advantages and disadvantages are discussed in more detail, all based on private beta v1.61 developed by GoGhost. (GoGhost, 2018)

4.1 Sample Optimization

To avoid saving unnecessary samples, two combinable options are available within Evotis: adaptive sampling and resampling.

Adaptive sampling, also used by some 3D renderers for increased speed due to adaptively minimized ray-sample counts, v-ray¹ for example (Chaosgroup, 2018), uses object- and texture-based approximations to determine edges or high-frequency areas within the frame and then uses the resulting map to adaptively adjust the sample count as needed. Thereby reducing the samples saved for plain areas or a black background to the minimum set by the user, and as a result reducing the final file size.

Resampling, on the other hand, allows the user to set the maximum sample count per pixel. Pixels are still used by Evotis as a guideline for the user, and also as a necessity to communicate the desired image proportions, the number of total samples and their spread, to the 3D renderer.

If both options are selected the user can determine the minimum and maximum sample-count per pixel, as seen in *Fig. 4-1*.

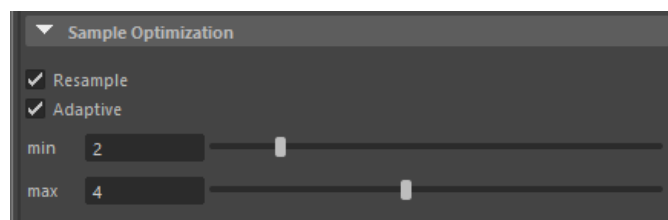


Fig. 4-1 - sample optimization options within Evotis interface in Maya

These values are squared for the output, meaning an entered minimum value of 2 will result in a minimum of 4 samples per pixel in the final rendering. These settings will later be referenced in *chapter 5* as a number pair, e.g. 2-4 for a minimum value of 2 and a maximum value of 4. By setting either of these limits the spatial positioning of the samples within the pixel is changed for every affected pixel to a uniform pattern rather than the non-uniform randomly spread samples generated in the renderer.

The resulting sample behavior is shown in *Fig. 4-2*, a zoomed-in 4x2 pixel segment of a rendering of two planes with solid colors applied.

¹ V-Ray Homepage: <https://www.chaosgroup.com/>

The six labeled sub-images show:

- (a) the resulting pixel image. The hard edge between the two objects gets anti-aliased, resulting in the color bleeding into the white, which in turn means that the two object will not be cleanly separable in compositing without additional edge work, which will be further addresses in chapter 4.3 *Advantages of Evotis*.
- (b) the rendered samples without any reduction method applied, resulting in, on average, 75 samples per pixel.
- (c) the adaptive method with a minimum setting of 2, resulting in a reduced sample count in all pixels not intersecting with the objects edges, but leaving the edge-area sample-counts untouched.
- (d) the resample setting with a maximum setting of 3.
- (e) both settings, adaptive and resample combined (2-3) resulting in reduced sample-counts for both, plain areas and edges, 4 and 9 samples respectively in this case .
- (f) the results of the adaptive setting rendered with v-ray, which, as mentioned before, already uses its own adaptive method for ray-sample generation, resulting in just a single sample per pixel in the flat areas of the two planes.

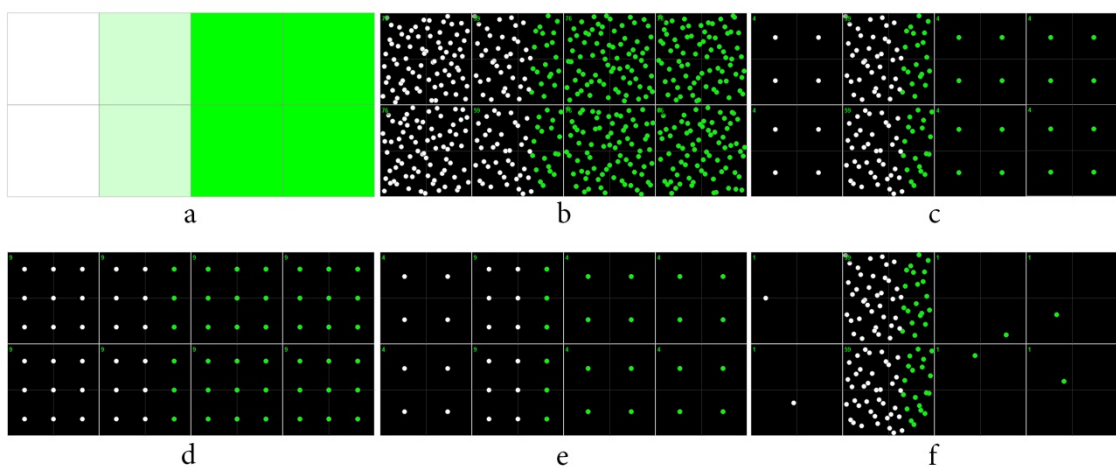


Fig. 4-2 - sample optimization comparison

The shown sample optimization options are a necessity to keep file sizes down to still be able to work with the renderings, but at the same time preserve as much detail as possible, or needed, to take advantage of the features offered by Evotis. The values used are not a representation of any kind, but were chosen for visual clarities sake. The definition of best-practice values will be based on the specific scene, company and use-case, just as it is done with the render settings, these settings will need a constant fine-tuning to balance quality with file size and rendering time.

4.2 Capabilities and Implementation

By preserving the samples, rather than a combined pixel, Evotis postpones the sample combination and area reconstruction into the compositing software, which demands a higher processing effort. This approach is similar to deep images, as they also have to be converted to a flat image within compositing to be handled natively. One problem using this workflow of in-comp conversion is the strongly reduced working-speed of the compositing software, usually leading to pre-renderings, as is common practice, when working with deep files. Another time-affecting aspect to be considered is that Evotis, due to being based on samples rather than pixels, needs to fill-in the areas in between the saved samples. For this Voronoi meshing is used (GoGhost, 2018), another processing intensive step, thereby further reducing the performance. Due to applying the area-reconstruction step during compositing it can also happen, especially when using Evotis' rescaling capabilities, that an insufficient number of samples remain within a given area to properly reconstruct it, leading to flickering.

The 3D part of Evotis is developed as a plug-in, currently only available for Maya¹, which generates a separate .evo file next to the regular flat rendering. Its implementation is based on AOVs or render-elements, depending on the renderer, therefore not demanding any changes to the scene or setup to be activated or deactivated.

For compositing Evotis is, so far, available as a Nuke² plug-in with several gizmos, all designed for basic tasks, as Nuke is unable to handle Evotis files by default. The three important ones, for a basic workflow, are: evoReader, evoID and evoReformat.

The evoReader is used to read evo files into Nuke. Its options are still basic in the tested beta version, as there are no color workflow or metadata options implemented yet.

EvoID enables the user to generate different “Sets” containing objects of the Maya scene. Evotis uses the scene-specific Maya hierarchy and naming convention to automatically generate IDs for any object, which can also be grouped into multiple different “Sets” within the evoID node, to be used

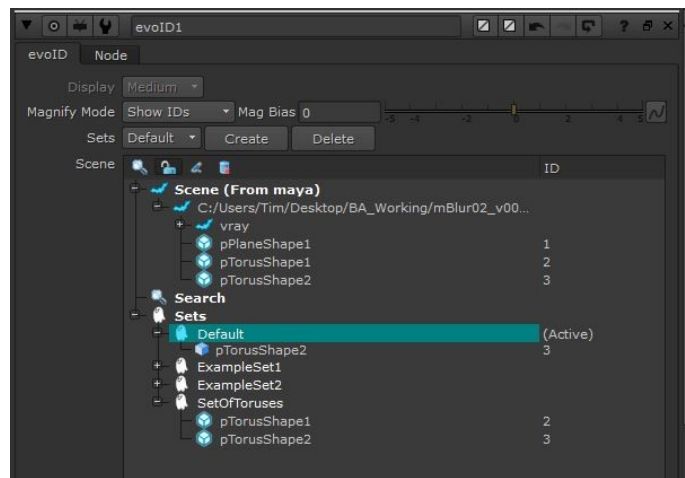


Fig. 4-3 - evoID interface

later for matting or object isolation. The Maya scene hierarchy visible in *Fig. 4-3* is a basic example of only three objects within a scene. In this example three custom sets

¹ Maya Homepage: <https://www.autodesk.de/products/maya/overview>

² Nuke Homepage: <https://www.foxandlorenz.com/products/nuke>

have been created and the “Default” set was modified to only include one torus. Every set created in the evoID node is carried along down the tree.

The evoReformat node is, in its essence, used to reformat the sample-based image to a pixel-based image, for Nuke to continue working on. Due to this being the last sample-using node it offers a lot of options and settings, which are shown in *Fig. 4-4* and mostly self-explanatory.

One feature to point out: within the “Set” dropdown all four sets from before are selectable with the options to either copy the selected set into alpha or RGB, to use as pixel-based mattes later, or to isolate the object and afterwards convert the image to pixels. The most important feature in this node is the dropdown to select

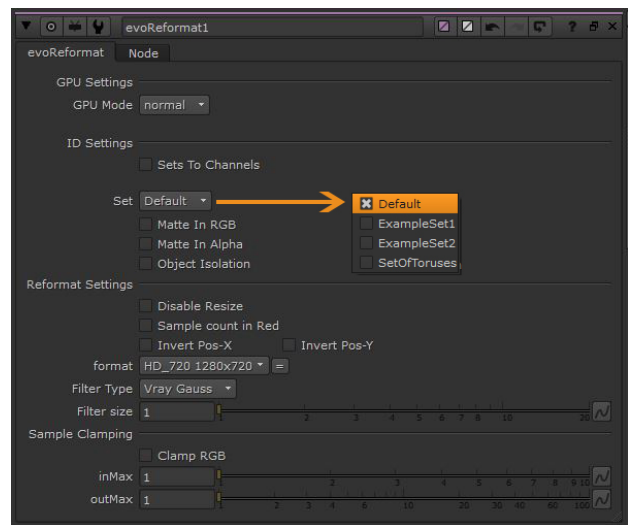


Fig. 4-4 - evoReformat interface

different output formats, as this not only defines the image that will be used down the stream, but also shows one of the great advantages of a non-uniform workflow: resolution independence, which will also be further discussed in chapter 4.3 *Advantages of Evotis*.

Fig. 4-5 shows a brief overview of three possible workflows while working with Evotis within Nuke. The red line indicates the border between the sample-based node-tree and the pixel-based part of the tree. A nice feature, compared to

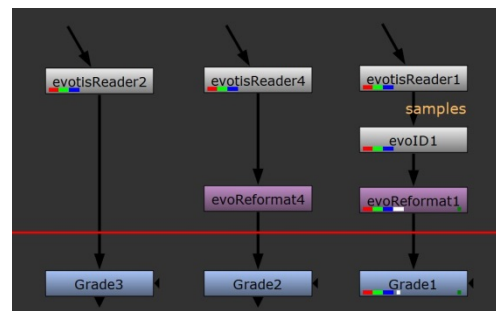


Fig. 4-5 - workflow examples

deep, is the ability to directly attach pixel-based nodes to anywhere in the Evotis section of the tree without needing to attach a separate conversion node.

4.3 Advantages of Evotis

Some of the advantages, improved possibilities and new approaches of a sample-based image type are compared to the applicable flat or deep workflows in the following sub-chapters. Each aspect, object isolation, ID mattes, advanced resizing and the possibility to refine renderings, will be covered in detail, each in a separate sub-chapter.

4.3.1 Object Isolation

One major advantage of the sample-based image type used by Evotis is the improved object separation within Nuke. A clean edge separation within a scene, traditionally done with specifically rendered object IDs and lately by using Cryptomatte (Friedman, et al., 2015), has usually been a problem-introducing task. Whether it's due to anti-aliasing, motion blur or semi-transparencies,

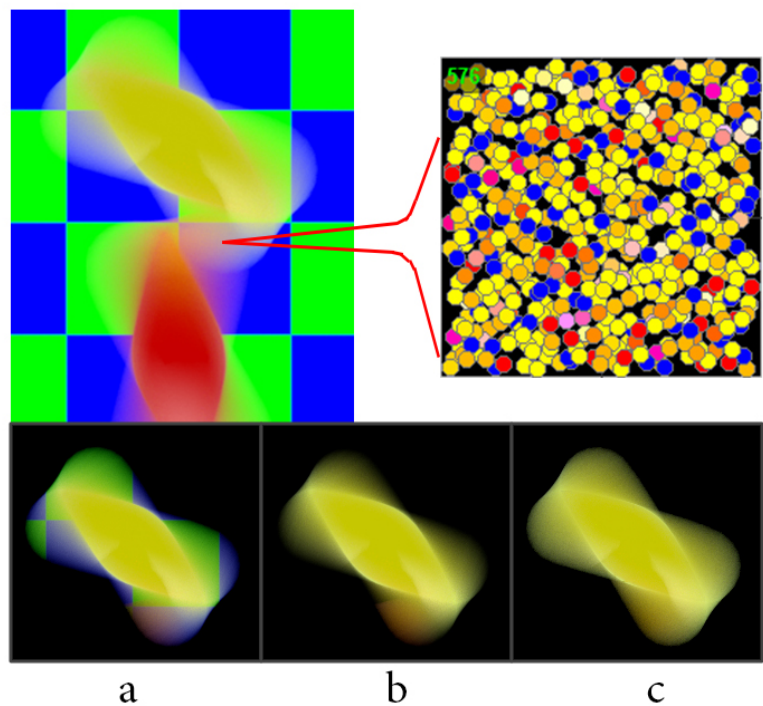


Fig. 4-6 - object isolation comparison

separating an object and extensively changing its appearance usually leads to edge issues, for instance: dark or bright outlines, artifacts, NaN-, infinity-, hot-, dead- or freak-pixels.

To illustrate this, a simple scene, containing two colored spinning toruses and a checkerboard, was setup in Maya. The resulting rendering can be seen in the top-left corner of *Fig. 4-6*. The three sub-images, forming the bottom row, demonstrate three different ways of separating the yellow torus from the background checkerboard and the red torus. In (a) Cryptomatte was used, clearly showing the checkerboard beneath the semi-transparencies due to the motion blur. In (b) a combination of a deepCrop and Cryptomatte was used, already resulting in a better separation, but still with a clearly visible outline of the red torus. Lastly, in (c) Evotis was used, resulting in a clean torus separation, without the visible edge of the red torus. The slight red coloring of the torus is bounce light and therefore is correctly separated. This bounce can also be seen in the top-right corner of *Fig. 4-6*, showing all 576 samples of one pixel in the area of overlaying motion. Besides the expected yellow, red and blue there are also orange, pink and other mixed value samples visible, which are colored like this due to the bounce light in between the objects. This sample square also visualizes clearly why the flat pixel-based approach from (a) was unsuccessful.

Also the outer-edge area retrieved with Evotis is bigger, as even the faintest motion-blurred samples are object-related and therefore retrievable, which will prevent edge issues from occurring during recombination, as can be seen in *Fig. 4-7* in which approach (b) and (c) from *Fig. 4-6* were used to first separate the yellow torus, then recolor it pink and finally recombine it with the background. It is evident, that method (b), deepCrop and Cryptomatte combined, produces a visible outline, resulting from left-out yellow pixels, in the far-out motion blurred areas, whereas method (c), Evotis, produces a solidly recolored torus without any visible outline. This example was, of

course, constructed to provoke this behavior, but similar problems occur in compositing on a daily basis, but are more complex to solve as a simple deepCrop, as used here, will not work in dynamic scenes. This usually means edge work of any kind, which leads to more work for the artist resulting in slower turn-arounds.

The accurate separation of an object also allows for relighting and texture

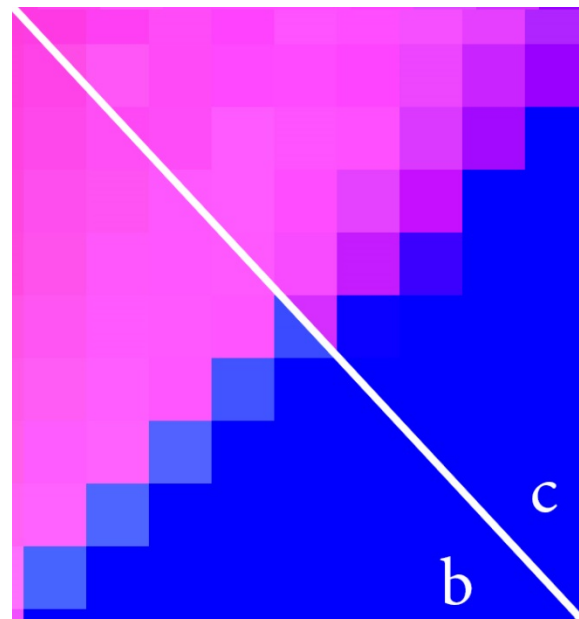


Fig. 4-7 - edge comparison

projection to produce highly improved results, as these operations very often generate a visible outline, due to the strong changes applied to these areas.

4.3.2 ID Mattes

Traditionally ID mattes had to be set up by the 3D artist manually as separate render layers, usually containing a maximum of three mattes per layer, saved in r,g and b respectively, to later be used by the compositing artist. Complex scenes demanding more than three mattes also required multiple render layers, all manually set up by the artist, and often just a guess as to which objects will need mattes.

Evotis offers all mattes that could have been created to the compositing artist by using evoID, as explained previously. As all hierarchical Maya information is preserved, navigating complex scenes to pick specific mattes is easy and structured.

A similar approach to matte generation is used by Cryptomatte and OpenEXR/Id¹, both also allowing the compositing artist to interactively generate mattes, also both using a 3D hierarchy one way or another. While Cryptomatte uses multiple separate render layers, OpenEXR/Id uses a sidecar file and only functions with deep images. Cryptomatte has spread throughout the industry quickly over the last year and will probably become a standard soon. (Friedman, et al., 2015) (Corvazier, et al., 2016)

In conclusion Evotis' approach is not a novel one, although all these were developed roughly at the same time, but as it is sample-based it is more precise and offers a more versatile usage.

4.3.3 Resolution Independence and Resizing

Due to the non-uniform and non-rasterized nature of, and the amount of extra information stored by, Evotis, working independently of source- or delivery resolution opens up a lot of possibilities for new workflow approaches. Additional opportunities for saving rendering time and better preserving image quality, if repos are needed, also arise. Currently Evotis has no set of transformational tools available within the tested version of the Nuke plug-in. The rescaling capabilities on the other hand, are fully supported already, with rescale limitations only defined by the minimum samples saved per pixel, which is a user-definable value. Tripling the image resolution in compositing generates an image as free of quality loss as scaling it by a factor of 1.1, compared to a scaling operation on a rasterized image, which deteriorates in quality with every operation as filtering needs to be applied. To visualize this, an adaptively sample-optimized Evotis rendering, with a resolution of 462x260 is scaled up to full HD, 1920x1080, a rescale by a factor of 16. In *Fig. 4-8* a 500% zoom-in of the resulting scaled

¹ <https://github.com/MercenariesEngineering/openexrid>

up Evotis (a), of a flat rendered natively at full HD (b), and of a flat, 462x260px, scaled up to full HD using the cubic filtering algorithm (c) are shown. As can be seen, there are no apparent visual differences between the scaled up Evotis and the native flat, especially

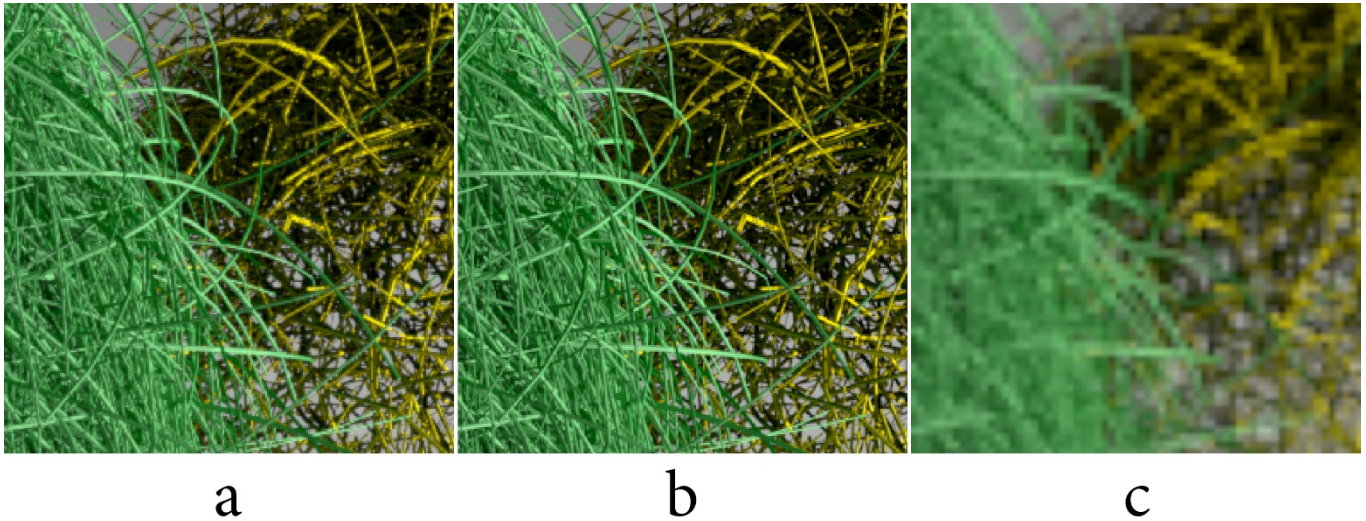


Fig. 4-8 - scale-up comparison

compared to the results of the scaled-up flat rendering (c), which shows a clear decline in image quality due to the rescaling.

4.3.4 Refining Renderings

Evotis enables the compositing artist to append samples to a rendering and thereby refine the original render quality. This in turn allows a rendering artist to refine his previous rendering by re-rendering with another sampling seed resulting in differently positioned samples, thereby correcting areas with artifacts without wasting the previous rendering output and time. This could be a very useful feature for quick last-minute fixes. To determine whether this approach can correct flickering a sequence with different soft light gradients was rendered and afterwards re-rendered with a new seed to be appended. To better visualize the results, the sum of each frames absolute

difference to the sequence average was used, thereby highlighting the strength of noise and flickering present in the sequence:

$$\sum_{k=0}^{x_{SeqLength}} \text{abs} \left(\text{frameValue}(k) - \frac{\sum_{i=0}^{x_{SeqLength}} \text{frameValue}(i)}{x_{SeqLength}} \right)$$



Fig. 4-9 - noise level comparison

The results are shown in a split image in *Fig. 4-9*. The top half is the result of the base renderings summed up differences, while the bottom half is the result after appending new samples. The improved noise level is evident due to the whole image being darker; less difference in between the frames means lower values. This proves that appending new samples can lead to improved noise levels in a rendering, thereby allowing for possibly refining renderings in case a quick fix or last-minute changes are needed.

4.4 Disadvantages of Evotis

Any new technology, even though offering many advantages and advances, has its drawbacks. For VFX, and Evotis as well, these drawbacks are usually hardware related: file size, performance or render time, which will all be covered and tested in detail in this and the following chapters. Furthermore Evotis also lacks deep data support in the current beta version, which is a major problem, as this feature is essential to any possible industry acceptance.

The big conceptual problem of Evotis, at least at this stage, is its inability to preserve and use deep information properly. While it is possible to render a Z-depth AOV on a per sample basis, it is not possible to render two samples behind each other, which means atmospheric renderings are not a viable option at this point. Therefore it is also not possible to use Evotis to generate holdouts or merge correctly in depth. All of this is especially important, as the main reason deep is now as widespread are its advantages in dealing with atmospherics, and combining multiple assets correctly by utilizing the depth information. Therefore, Evotis is, at least with this version, not a suitable competitor for deep, as without depth samples any company would still need to keep deep in their pipeline as well, resulting in three different image types used in parallel, which will not be an option. But as GoGhost has pointed out, this is a renderer specific issue, and has already been solved for other renderers than v-ray, therefore this should be solvable in the future and depth-spread samples could be available, but until now the lack of proper deep information is a knockout argument for Evotis.

Due to all the information stored within an Evotis file one possible, and possibly also very important, disadvantage could be its performance in 3D and 2D as well as the resulting file sizes. Just as the slowed down workflow of deep has kept it out of most

VFX facilities for a long time, a loss in time, be it in rendering or while interacting with it in compositing, is a major drawback to any new technology and might hinder its success severely. As the performance will be of critical importance to Evotis, this will be tested and evaluated in detail in chapter 5 *Performance Evaluation*.

4.5 Future Possibilities

As Evotis is in its early, non-public, beta phase, there is still a lot of room for further improvements and possible features or use-cases to be added. In this chapter three exemplary possibilities, not exceeding already implemented base features, are given: specific re-rendering area selection, viewing angle independence and an Evotis library.

4.5.1 Precisely Specified Render Areas for Re-rendering

To be able to use the possibility to append samples effectively, an array of further option could be implemented. Being able to select the area for re-rendering based on an area of interest, object IDs, shadowed areas, based on the sample count per pixel or a combination of them all will lead to more effectively used render time, as artists can gradually improve the quality dependent on shot needs without losing all previous rendering efforts.

4.5.2 Viewing Angle Independence

As soon as the depth support for Evotis will function properly, generating a point-cloud of samples within Nuke could allow the compositing artist to change the viewing angle of a rendering, to a certain degree at least. While the concept was also proposed for deep multiple times, it never really was used, as the needed reconstruction algorithms to fill the gaps in between the points were missing. But as Evotis is based on area reconstruction, changing viewing angles could become a viable option. This could lead to possibly not needing to create a second rendering for stereo productions, as the shift in perspective is quite low. This could be further optimized by allowing an occlusion rendering option to render overlapped areas to a certain degree, to sufficiently cover the areas needed for correctly generating the parallax. Another possible workflow could be to render and append only the samples needed for parallax, which could be easily accomplished, if the previously mentioned area selection for resampling gets implemented in a versatile enough way, by generating the new samples based on a camera projection map of the hero-eye camera identifying the areas that will be occluded and therefore need further samples rendered.

4.5.3 Evotis Library

Due to the non-uniform nature, and the rescale ability Evotis renderings can be used to build a library that is more versatile due to its resolution independence and the automatic ID function. In combination with the before mentioned probable possibility to adjust viewing angles to a certain degree it allows a compositor to use a single rendering for a multitude of different shots, thereby allowing the library to keep a smaller quantity of files with increased use-cases.

5 Performance Evaluation

As previously mentioned the performance of Evotis, in 3D as well as in 2D, will be a decisive factor for its possible success and acceptance within the industry, as no VFX company can allow loosing time in such a competitive market, therefore several performance monitoring tests are conducted and evaluated in this chapter.

Another important aspect, and therefore also discussed, is the file size, an aspect that has also kept deep away from widespread use at first. Even though the average cost per GB has dropped significantly, from 1.60\$/GB in 2003¹ to 0.03\$/GB today² small file size is still important, as larger files will also lower the performance in compositing significantly by flooding the cache and needing more bandwidth within the local network.

5.1 Test Method

For 3D performance testing Evotis v1.61 private beta was used with four different Autodesk Maya 2017 scenes highlighting different key aspects. Each scene was batch-rendered with Chaosgroups V-Ray for Maya 3.60.04 while the hardware performance was measured using Open Hardware Monitor 0.8.0³ set to two second intervals, with all

¹ <http://www.mkomo.com/cost-per-gigabyte>

² <https://www.alternate.de/Festplatten/SATA/>

³ Open Hardware Monitor Homepage: <https://openhardwaremonitor.org/>

files being read and written to an internal SSD. To determine the rendering times accuracy of the monitored frames at least ten frames of each scene were rendered, and the median rendering time was compared to that of the performance monitoring result. If the deviation exceeded 5% the frame was re-rendered and re-monitored.

For 2D performance testing the resulting flat, deep and Evotis files from the 3D tests were loaded from a local SSD, modified and rendered to the same SSD with the command line renderer of The Foundry's NukeX 10.5 and a custom Python script generating the needed render logs. Each frame was rendered at least 50 times and values given in this chapter are the resulting averages. Evotis v1.61c private beta was used, as this updated version included a few Nuke-related bug-fixes. Performance was measured using Open Hardware Monitor 0.8.0 set to one second intervals.

The resulting .csv files of both, 3D and 2D testing, were cleaned-up and parsed into .xls files to be visualized. For these tests all CPU, GPU and RAM related information was logged.

5.2 3D Test Results

For the 3D tests only the percentile CPU load will be visualized and discussed, as, due to the nature of the test scenes used, neither RAM, GPU, nor disk writing speeds played any role for the final results.

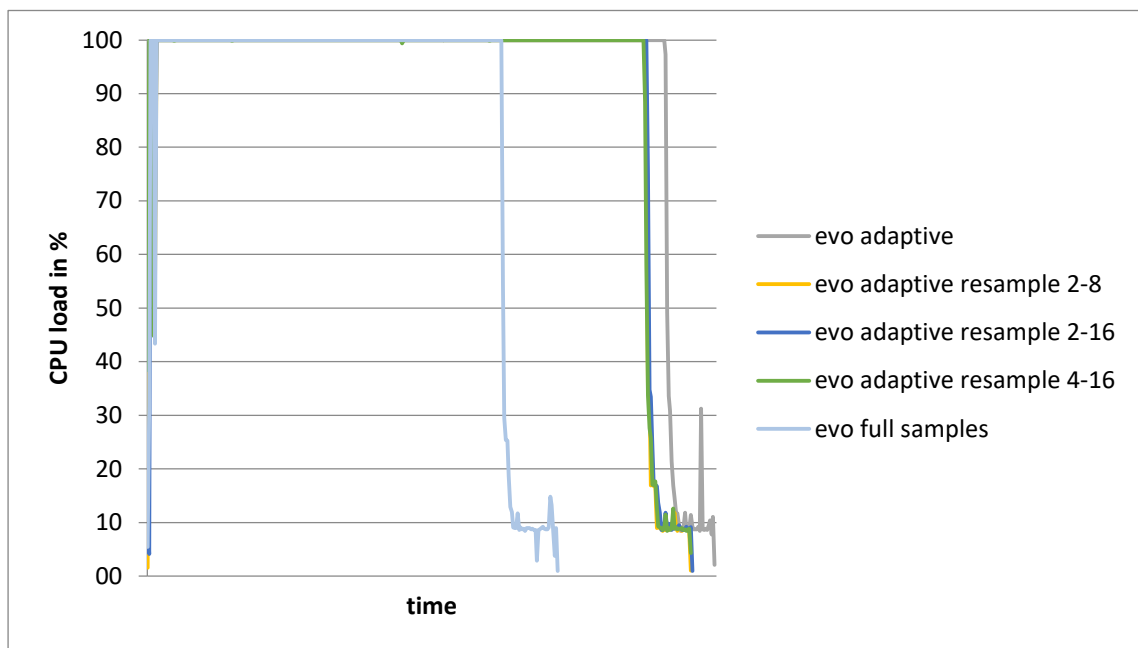
5.2.1 Test Scene 1 – Simple Scene

The initial performance test conducted, using the small-scale hard-surface scene that produces the image seen in *Fig. 5-1*, was supposed to determine the differences of the sample optimization options.

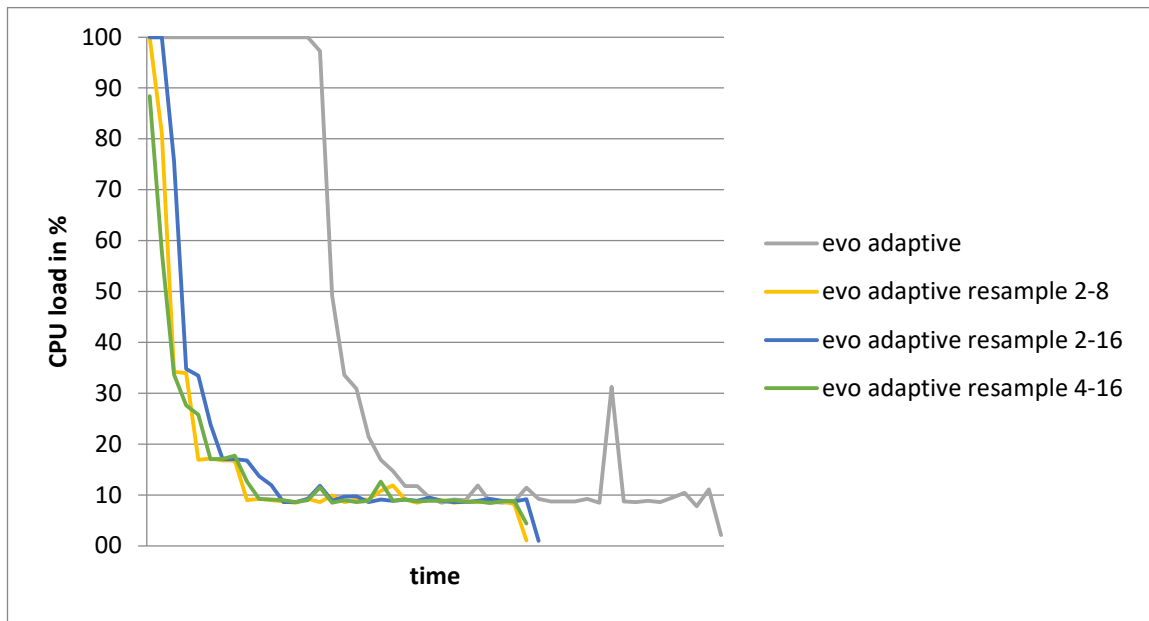


Fig. 5-1 - rendering of 3D test scene 1

As can be seen in *graph 5-1*, and even better in *graph 5-2*, the time differences between the various resampling settings are negligible. As expected, keeping all samples in the files yields the fastest render time, but possibly also the largest files. The resampling yields a slightly shorter rendering time compared to the adaptive method, which was to be expected as well, as the resampling is a threshold-based deletion and repositioning of the remaining samples, rather than the content-dependent approach of the adaptive option.

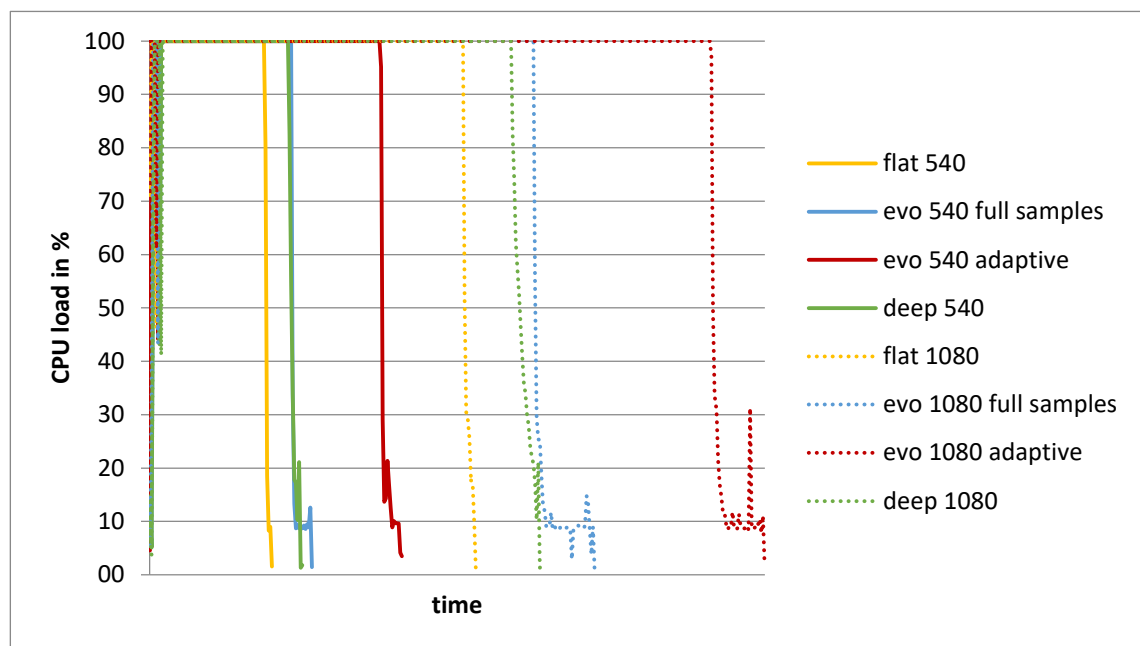


graph 5-1 - initial testing of different Evotis sample optimization options



graph 5-2 - zoom-in of last portion of graph 5-1

For *graph 5-3* the percentile CPU loads of the corresponding flat and deep renderings were added and, for the sake of clarity, only the plots of the full sample Evotis and the adaptive Evotis rendering were kept, as these two indicate the range of possible Evotis rendering times.

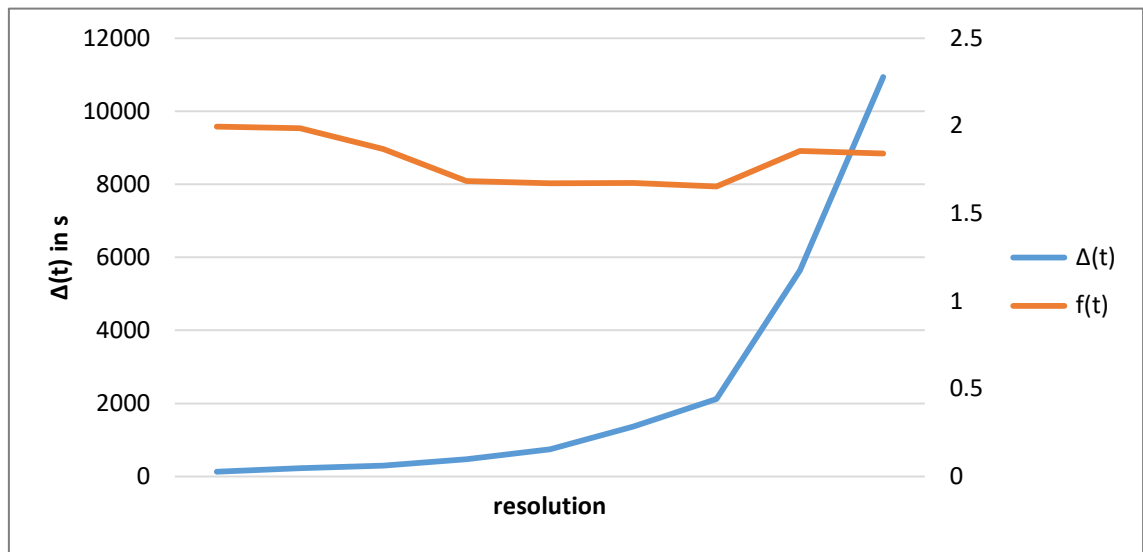


graph 5-3 - comparison of Evotis to flat and deep

The graph clearly shows, that all Evotis renderings take longer than the respective flat rendering, but, the deep rendering and the full sample Evotis rendering produce similar rendering times. This indicates the possibility, that the Evotis rendering samples are all generated at rendering time. After further testing this proved to not be the case and was just coincidental here, as the render time of the Evotis without sample optimization was significantly longer than that of the deep rendering in all other tests conducted.

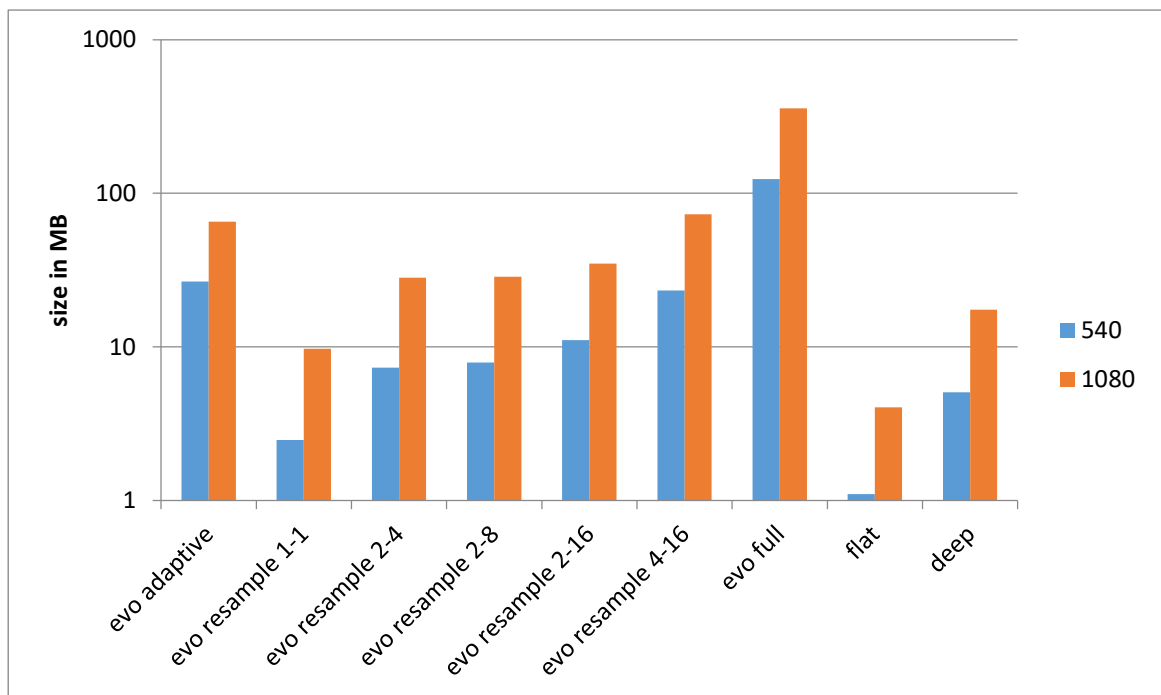
The increased time difference, $\Delta(t)$, between the flat and Evotis renderings, of the half-res and the full-res rendering is also evident in *graph 5-3*. To determine the growth-rate of $\Delta(t)$ the same scene was rendered in nine different resolutions, always doubling vertical resolution with additional half-way steps to avoid 14k+ renderings. The adaptively optimized rendering was used for this test. The results can be seen in *graph 5-4*. While $\Delta(t)$ rises exponentially, in parallel with the amount of pixels, $f(t)$ remains roughly the same, indicating a constant relation between flat and Evotis render times for this test scene.

$$f(t) = \frac{t_{Evotis}}{t_{flat}} \quad \Delta(t) = t_{Evotis} - t_{flat}$$



graph 5-4 - numerical and relative time differences of renderings in different resolutions

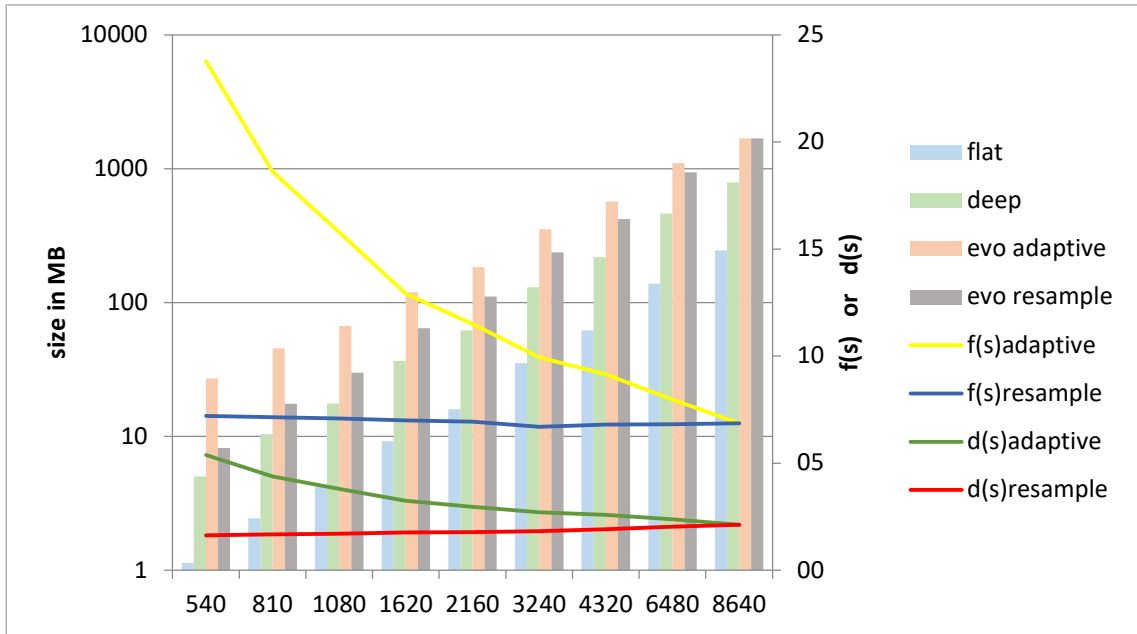
The size of an Evotis file is greatly influenced by the sample optimization options, as they determine the number of samples saved, and thereby the amount of data saved. For these test the same scene was rendered in HD540 and HD1080 to visualize the relation better. It is apparent in *graph 5-5* that a non-optimized Evotis file is unproportionally large in relation to a flat rendering, 113 times the size for the low-res (HD540) rendering and 88 times as big for the full HD rendering, while a resampled-to-minimum version is only 2.25 (540) or 2.4 (1080) times as large. In the lowest-quality settings the Evotis files are smaller than the equivalent deep, although in this setting most of the advantages of the image type are non-existent, therefore it is the most interesting to look at the adaptively optimized and resampled versions with at least a 2-4 setting. The files generated with the adaptive setting are 24 (540) and 16 (1080) times larger than the corresponding flat, while the 2-8 resampled files are only 7.1 (540) and 7.0 (1080) times bigger. These measurments clearly show the importance of sample optimization for Evotis to be usable.



graph 5-5 - file size comparison (log scale)

Another important aspect concerning file size is to determine the relative file sizes behaviour, $f(s)$ and $d(s)$, with increasing resolution.

$$f(s) = \frac{size_{Evotis}}{size_{flat}} \quad d(s) = \frac{size_{Evotis}}{size_{deep}}$$



graph 5-6 - file sizes at different resolutions

Therefore nine different resolutions were rendered each in flat, deep, Evotis adaptive and Evotis adaptive resample 2-8 and the resulting file sizes compared. While the $f(s)$ resample curve roughly stays at a value of 7, slightly declining over the range, as can be seen in *graph 5-6*, $f(s)$ adaptive on the other hand shows a clear decline over the range of resolutions rendered, from 23.8 to 6.9. This indicates that the adaptively optimized renderings cope progressively better with higher resolutions, which is due to the Evotis internal file compression algorithm. The adaptive resample rendering, on the other hand, can maintain a factor of 7 throughout the resolution range. For $d(s)$ adaptive a

similar decline can be seen as with *f(s)adaptive*, although on a smaller scale, while *d(s)resample*, contrary to *f(s)resample*, shows a slight incline in values, from 1.6 to 2.0.

After this initial test none of the following tests will use Evotis full sample renderings, as it is unrealistic for any VFX company to use the full sampled Evotis file, simply due to the enormous file size, instead the adaptively optimized version will be used, as it is the longest render time and thereby functions as a maximum render time indicator, as well as at least one resampled 2-8 version for reference.

5.2.2 Test Scene 2 – Motion Blur

The next test is based on the scene previously used in *chapter 4.3.1* to demonstrate object isolation, shown in *Fig. 5-2*. The goal here was to determine the effect of motion blur and semi-transparencies on the render time.

It is clearly visible in *graph 5-7* that the large amount of samples needed to generate the motion blur, 576 in the example pixel in *chapter 4.3.1*, also prolonged the Evotis render time, as all of those samples needed to be post-processed, resulting in a

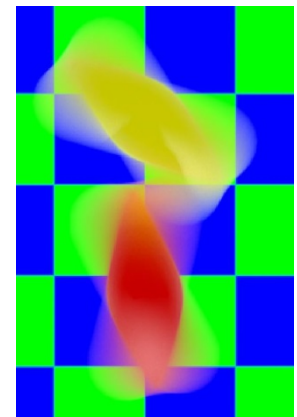
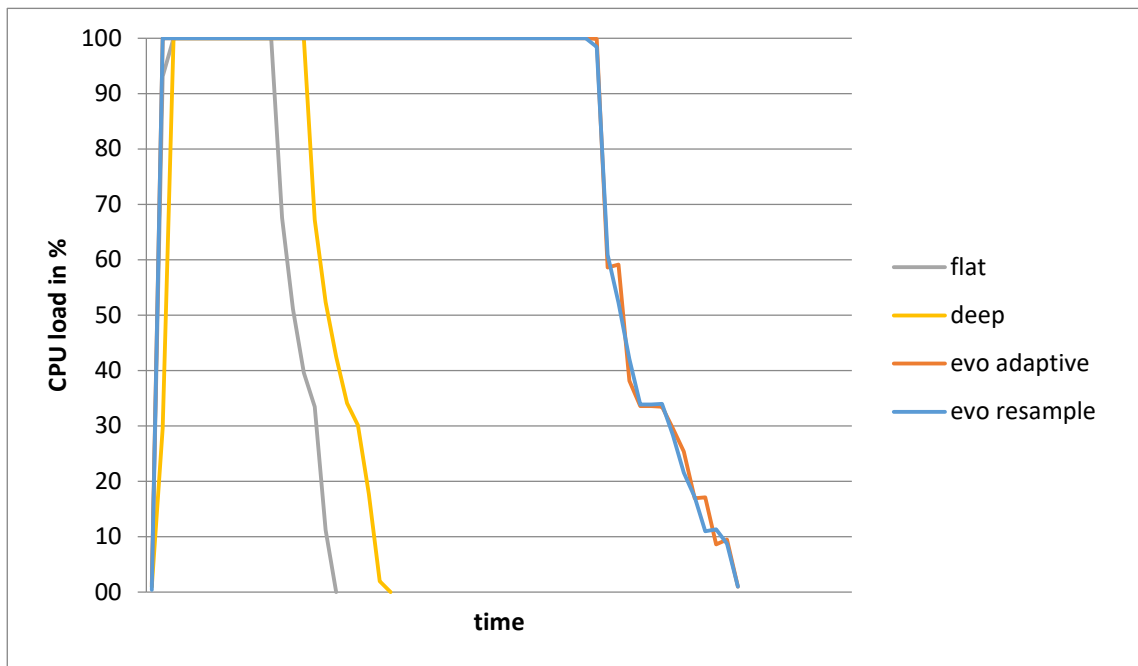


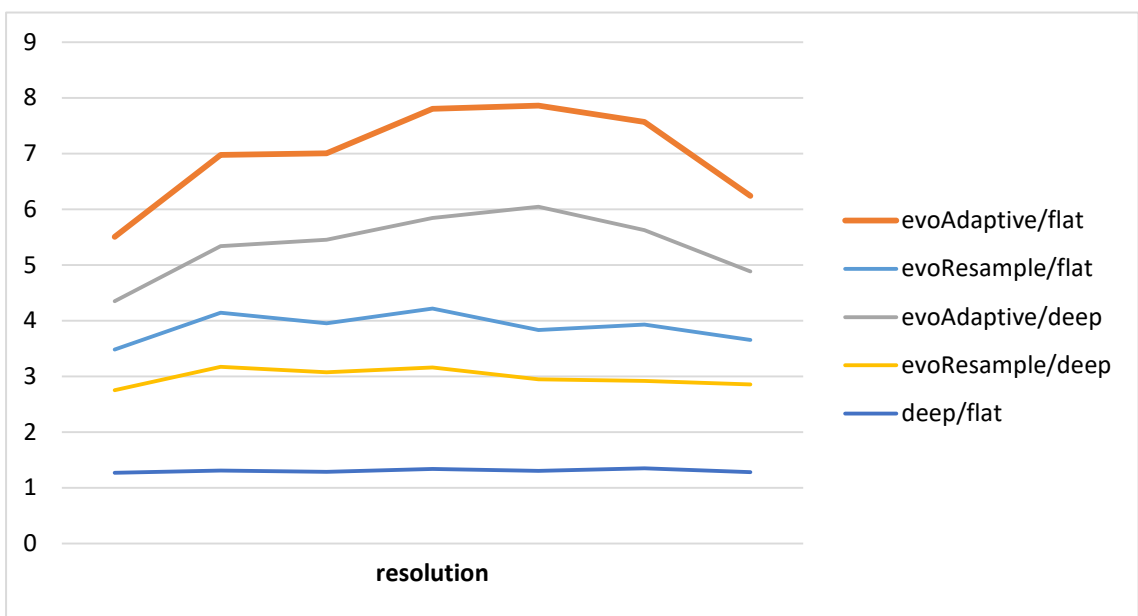
Fig. 5-2 - rendering of 3D test scene 2

render time 3.4 times as long as the flat rendering and 2.5 times as long as the deep rendering.



graph 5-7 - render time of motion blur scene

After determining the constant relation between flat and Evotis rendering times for the first test scene, by rendering the same scene in multiple resolutions, the same procedure, limited to seven different resolutions this time, was repeated for this scene, but with the addition of resampled Evotis renderings and deep renderings. The results can be seen in *graph 5-8*. While a constant relation between the resampled version and the flat is not as



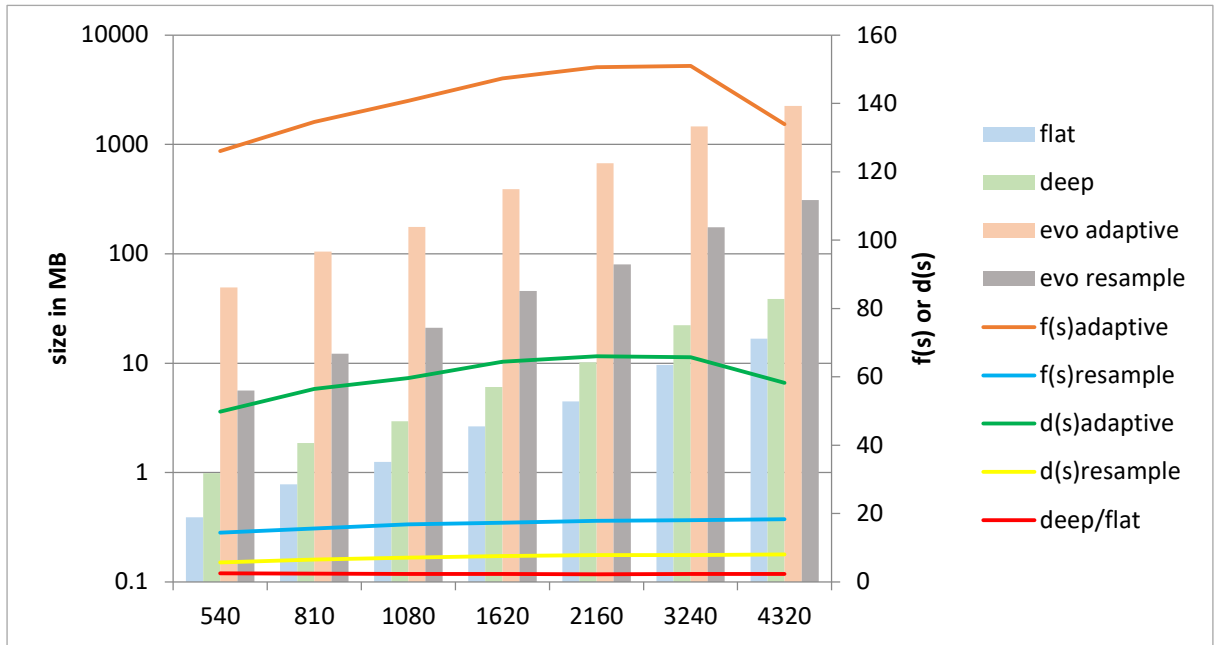
graph 5-8 - render time relation throughout resolution range

clearly visible here, as it was in the first test scene, it can still be considered a constant relation with a value of 3.82, the median over the tested range of resolutions.

This is an increase of roughly 100% compared to the first test scene, which is due to the increased number of samples needed for the motion blur and semi-transparencies, indicating that Evotis render times increase disproportionately compared to flat rendering times, with increasing scene complexity.

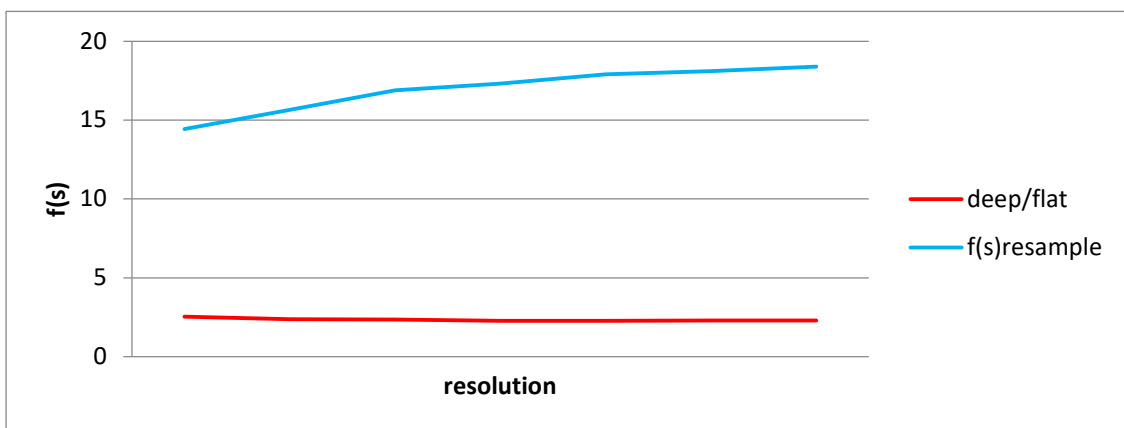
Due to the increased amount of samples needed the file sizes, and the relative file size factors, $f(s)$ and $d(s)$, also increased, as shown in *graph 5-9* and in the more detailed size factor plot in *graph 5-10*.

$$f(s) = \frac{size_{Evotis}}{size_{flat}} \quad d(s) = \frac{size_{Evotis}}{size_{deep}}$$



graph 5-9 - absolute and relative file sizes at different resolutions

A general increase was to be expected, as in a flat only one set of values per pixel gets saved, regardless of its contents, producing, not accounting for compression, content-independent file sizes, whereas Evotis' file sizes greatly depend on the images content. Nevertheless, a file size, on average, 140 times larger, especially for such a simple scene, for the adaptively optimized Evotis renderings, exceeds the scope of possibly being



graph 5-10 - detailed plot of f(s)-values

usable by far. Even the resampled 2-8 version is unlikely to be properly usable, as the files are, on average, 16.9 times as large as the flat rendering.

As a reference, the relation between deep and flat file size is also visualized in the graph, showcasing how disproportionately large the Evotis files are, even compared to deep, which is often criticized for producing files that are too large. Of course the Evotis file can still be reduced, by choosing different settings, a 2-4 resampling will generate a file 8 times larger than the flat and around 3.5 times larger than the deep rendering, a 2-3 resampling is 7.2 times larger than the flat and 3.0 times larger than the deep. Lowering the settings further would not make sense, due to losing the advantages Evotis can offer.

5.2.3 Test Scene 3 – Fur

The third test scene was designed to contain many object edges, to test whether this has any influence on the rendering times and file sizes. To achieve as many edges as possible a scene containing multiple fur elements was used, a rendering of which can be seen in *Fig. 5-3*.

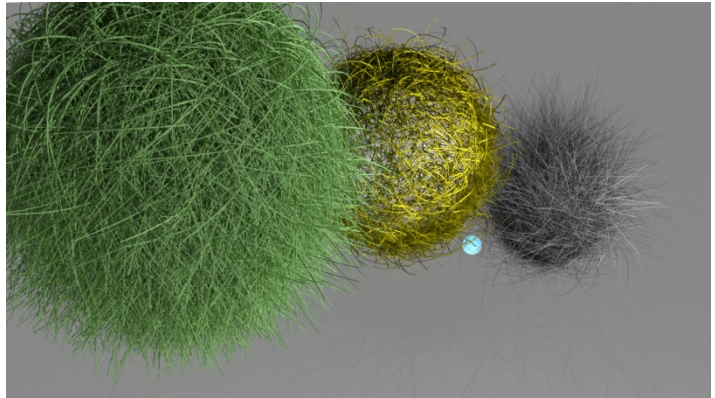
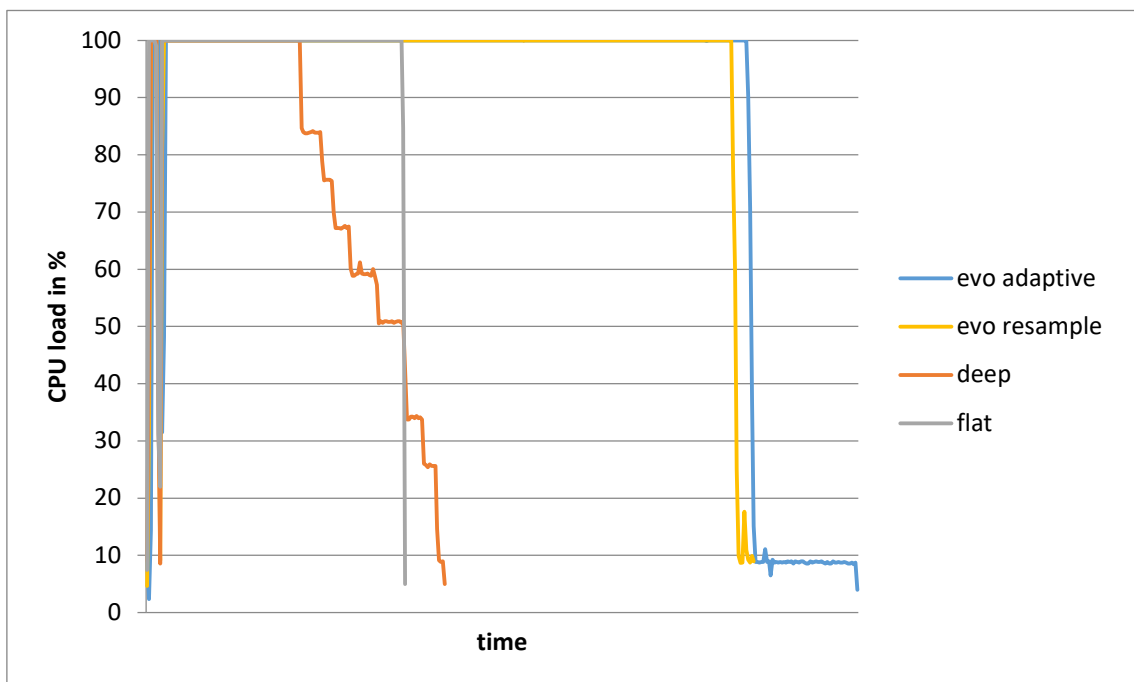


Fig. 5-3 - rendering of 3D test scene 3

For this test scene Evotis with the adaptive optimization option took 2.5 times as long to render as the regular flat and the resample 2-8 version took 2.14 times as long, which is shown in *graph 5-11*. After having shown a constant relation of rendering times in the first two test scenes it can be assumed, that the relation will also be a constant one again.

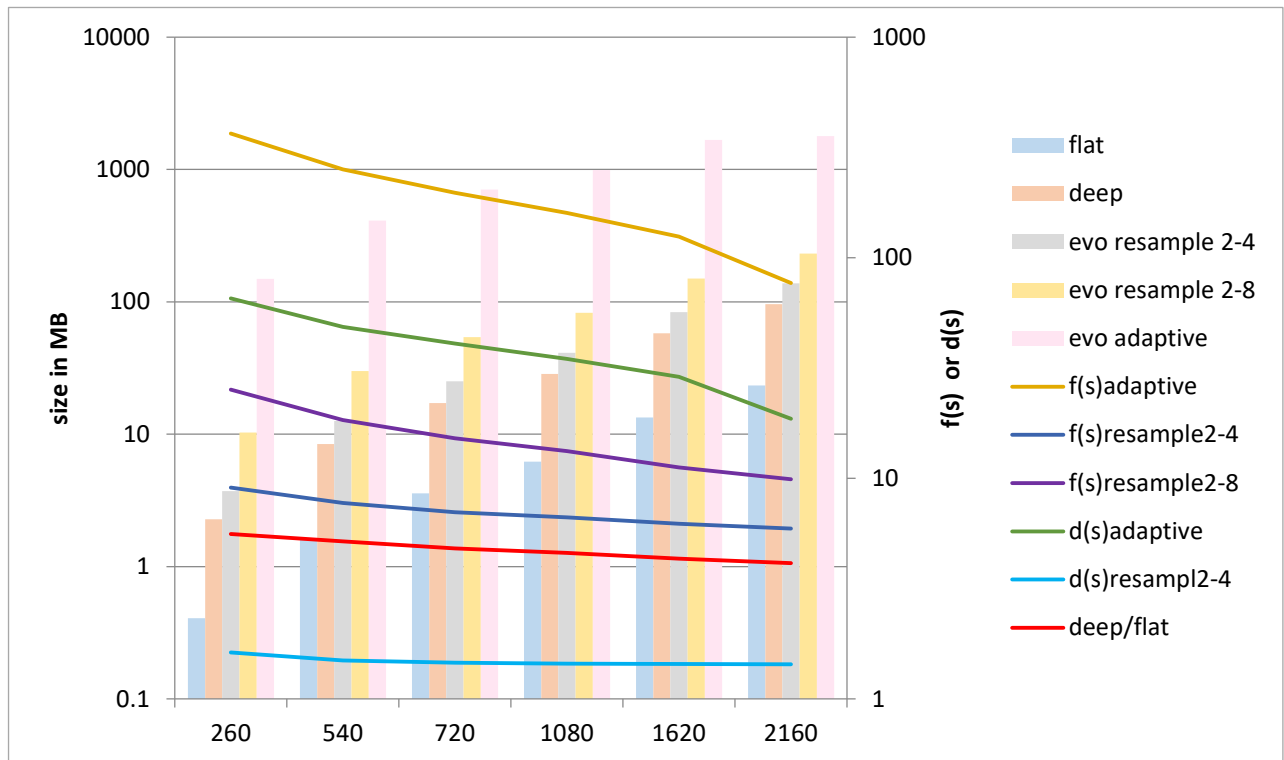


graph 5-11 - fur rendering results

Looking at the file size, the adaptive version was 365 times as large as the flat rendering, the resampled 2-8 version was 25 times as large, and 4.5 times the size of the corresponding deep rendering. The resample 2-4 version was 9.0 times as large as the flat, and 1.6 times as large as the deep.

As with the render time, it can be assumed, that the file sizes of the resampled renderings will also have a constant relation again. To confirm these assumptions a spot check at two higher resolutions was conducted. As expected the relation between render times proved to be a constant again, with an average value of 2.6 for the adaptive/flat and 2.3 for the resample2-8/flat version.

The file size relation, on the other hand, did not prove to be constant, therefore the full range of resolutions was rendered and the results are shown in *graph 5-12*. Due to a reproducible error no renderings with a resolution higher than 2160 were possible, as the beta version crashes the system once the temp file containing the preliminary renderings are loaded for sample conversion, possibly due to sample counts getting too high, therefore the range of resolutions was adjusted accordingly.



graph 5-12 - absolute and relative file size comparison throughout range of resolutions

As can be seen all curves show a decline in values over the tested range, without any possible threshold discernible. This behavior is due to a reduced object-edge/pixel ratio at higher resolution, allowing the adaptive minimum threshold to apply to more pixels as there are not as many object edges present within a single pixel.

5.2.4 Test Scene 4 – Fur with Motion Blur

The last test scene was a long rendering one, due to a higher ray sample count and a more complex scene, fur with motion blur, as shown in *Fig. 5-4*, rather than higher resolution. As can be seen in *graph 5-13* the adaptive Evotis renderings took, 2.57

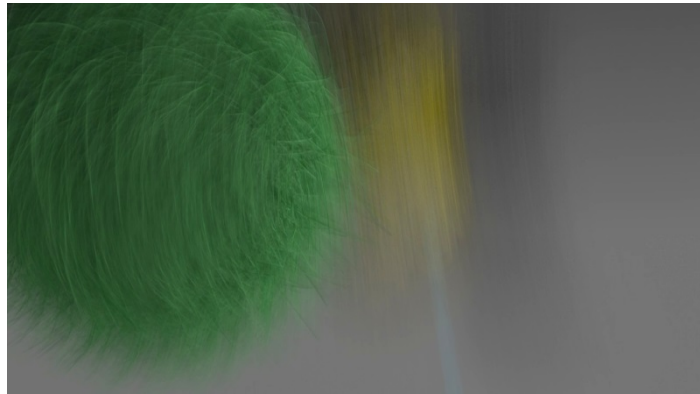
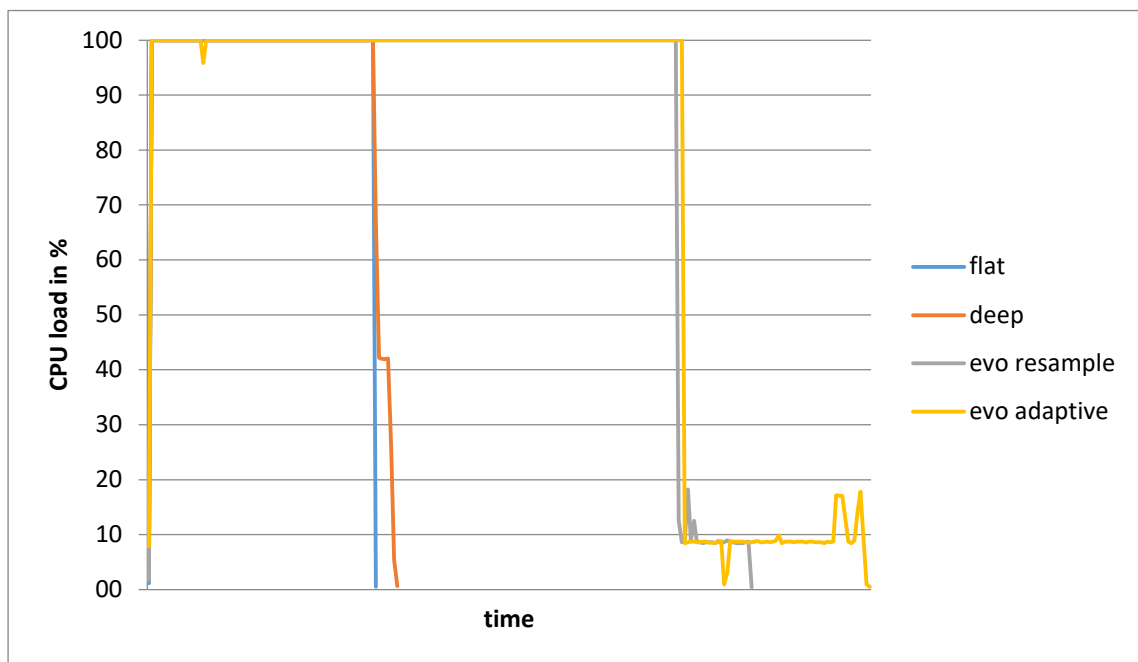


Fig. 5-4 - rendering of 3D test scene 4



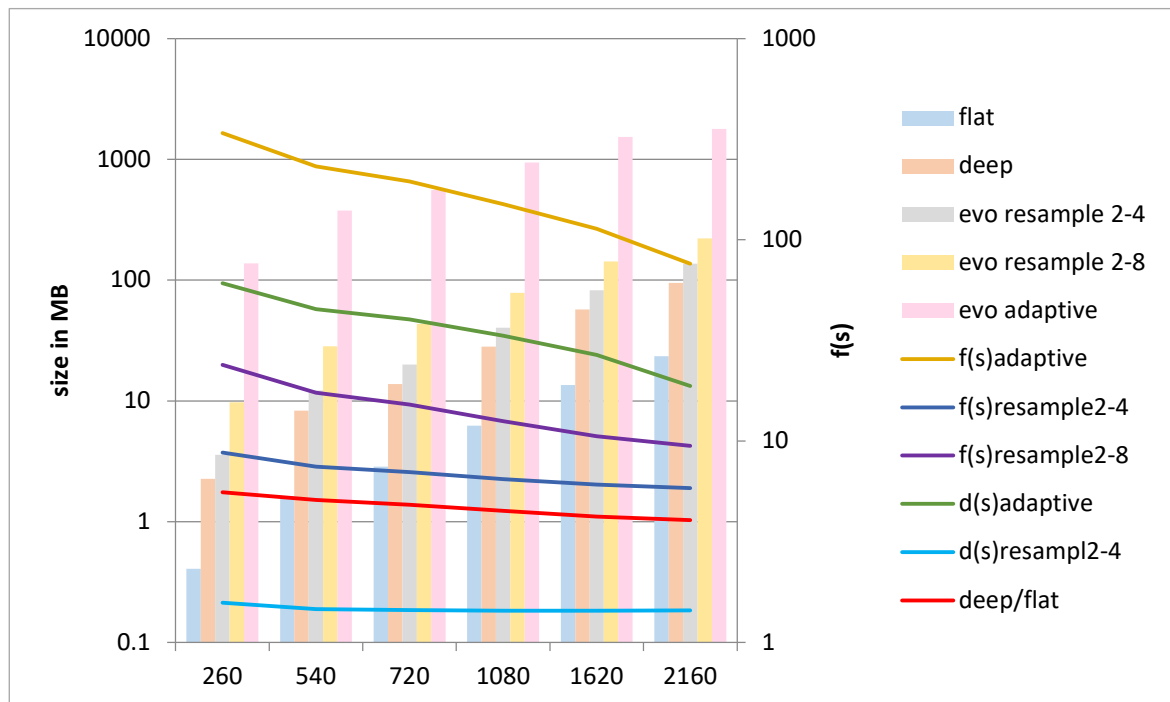
graph 5-13 - render times of long rendering scene

times as long. The constant relation of render times, as proven before, also applies here, as confirmed by two higher resolution spot checks, with a resulting average value of 2.65.

Relative file size has, as expected, increased with the adaptively optimized version being 339 times larger than a regular flat rendering, the resampled 2-8 version 23.9 times larger and the resampled 2-4 version 8.8 times. Compared to the deep rendering, the resampled 2-8 version was 4.3 times larger, and the resampled 2-4 rendering was 1.6 times larger, all these relations measured at low resolution.

After conducting a spot check at higher resolutions it was evident, as in test scene three, that no constant relation could be determined between the resampled renderings and the flat renderings file sizes. Therefore the full set of renderings throughout the shortened resolution range was conducted, as the same system crash during the sample conversion step occurred for these tests as well. The results are shown in *graph 5-14*.

Again, as in test scene three, the graphs show a decline over time, but not a discernible threshold.



graph 5-14 - absolute and relative file size comparison throughout range of resolutions

5.2.5 Summary of 3D Tests

A brief summary of the results obtained throughout the tests performed in the last four chapters is shown in *Table 1*.

	Scene 1	Scene 2	Scene 3	Scene 4
f(t)adaptive	1.8	6.9	2.63	2.65
f(t)resample	1.8	3.82	2.3	2.65
f(s)adaptive	12.9	140.6	195	184
d(s)adaptive	3.28	60	39.5	37.5
f(s)resample	7	16.9	15.5	14.8
d(s)resample	1.8	7.2	3.18	3.07

constant

average

Table 5-1 - summary of 3D test results

It is evident, that the render times, $f(t)$, are mostly of a constant nature, with only the exception of the adaptive render time in test scene 2, which is due to the very simplified geometry and scene setup in combination with strong motion blur. With respect to file size, $f(s)$ and $d(s)$, it is obvious that adaptively optimized renderings always display inconsistent relations to both, flat and to deep renderings. The adaptively resampled renderings on the other hand displayed a constant relation for the first two test scenes, but also shown inconsistent relations in test 3 and 4, due to increased scene complexity.

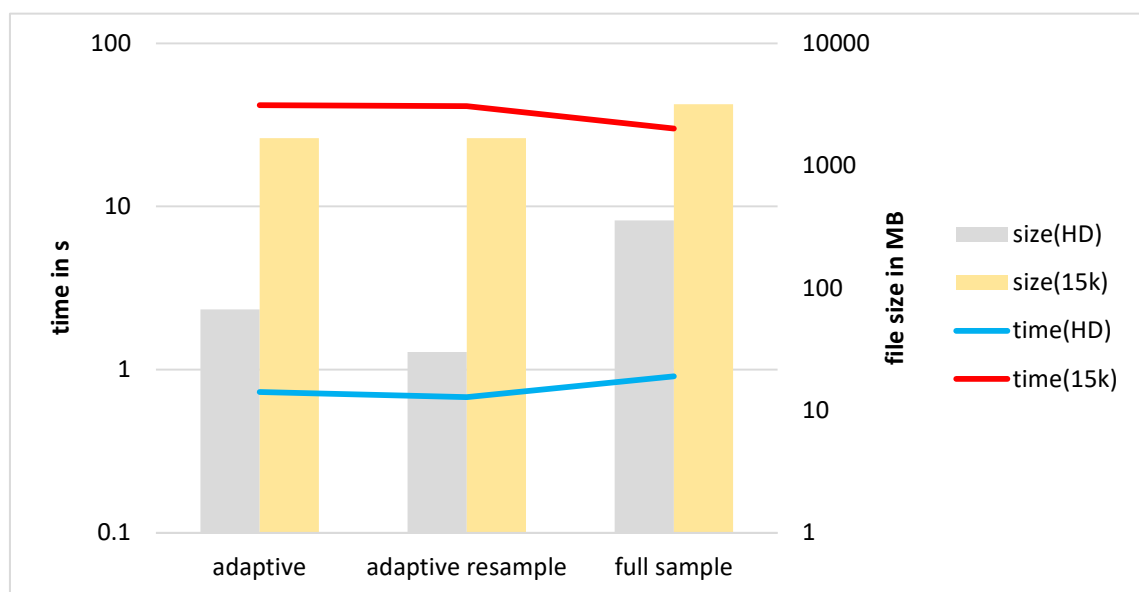
5.3 2D Test Results

For the 2D tests the percentile CPU and GPU loads will be visualized, as hard drive writing speeds, all test were run on a local SSD, and RAM did not contribute in any way. The resulted renderings of all previously utilized test scenes are used for 2D testing in this chapter. The Evotis files will be compared to flats and deeps for their performance

in Nuke, focusing on render times and hardware needs, as all rendered images from Nuke will be flats, therefore the resulting file size is of no significance.

5.3.1 Initial Testing

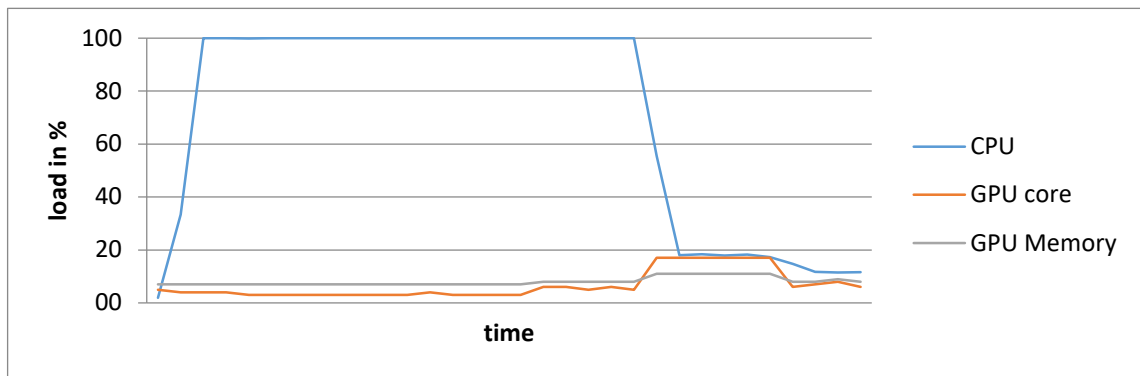
The initial test was conducted to determine whether the differently sample optimized renderings yield different render times in Nuke or utilize the hardware in different ways. Therefore three differently optimized Evotis files, full samples, adaptive optimization and adaptively resampled (2-8), at two different resolutions, HD and 15k, were used. Each one was read into a separate Nuke script from a local SSD, a single grade node was attached and then written to the local SSD again. The results of this test are shown in *graph 5-15*.



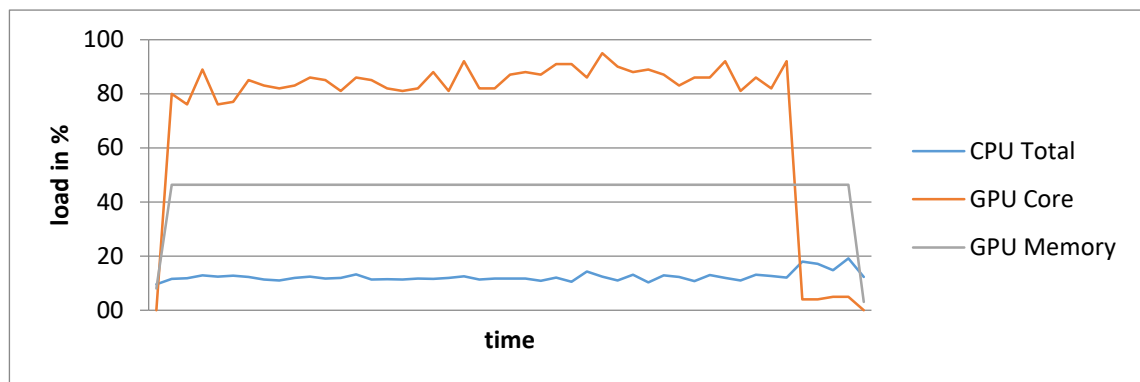
graph 5-15 - render time and file size comparison for different Evotis options

As can be seen, the rendering time needed by any of the three files is roughly equal, even though the file sizes differ significantly, therefore it can be assumed, that the sample

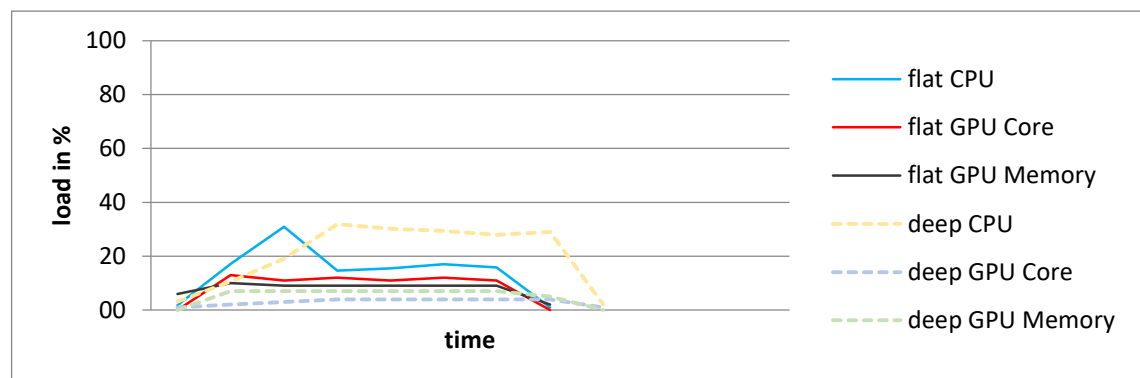
optimization option chosen during the 3D rendering has no influence in the time needed in Nuke. To determine any differences in terms of hardware needs these tests were also performance monitored. By looking at the performance monitoring, shown in *graph 5-16*, it is evident that most of the Evotis rendering is CPU based when attaching a grade node directly to the evoReader.



graph 5-16 - performance data of first test without using evoReformat



graph 5-17 - performance data when using evoReformat



graph 5-18 - performance data of flat and deep rendering

On the other hand, *graph 5-17* shows the performance data when using the *evoReformat* node to convert from samples to pixels within Nuke. As can be seen in the graph, the *evoReformat* node uses GPU rendering, allowing faster parallel processing of the large amounts of samples within the picture, on appropriate graphic cards. Due to an old graphics card in the workstation used for testing, and the lost comparability to flat and deep renderings, GPU rendering will not be tested further, but is important to keep in mind for possible performance gains in other testing environments.

For reference the performance data from the corresponding flat and deep renderings are shown in *graph 5-18*. As can be seen neither flat nor deep uses more than 30% of the available CPU capacity, which is due to the fast native processing of the files within Nuke and the simple change, only one grade, applied in this test.

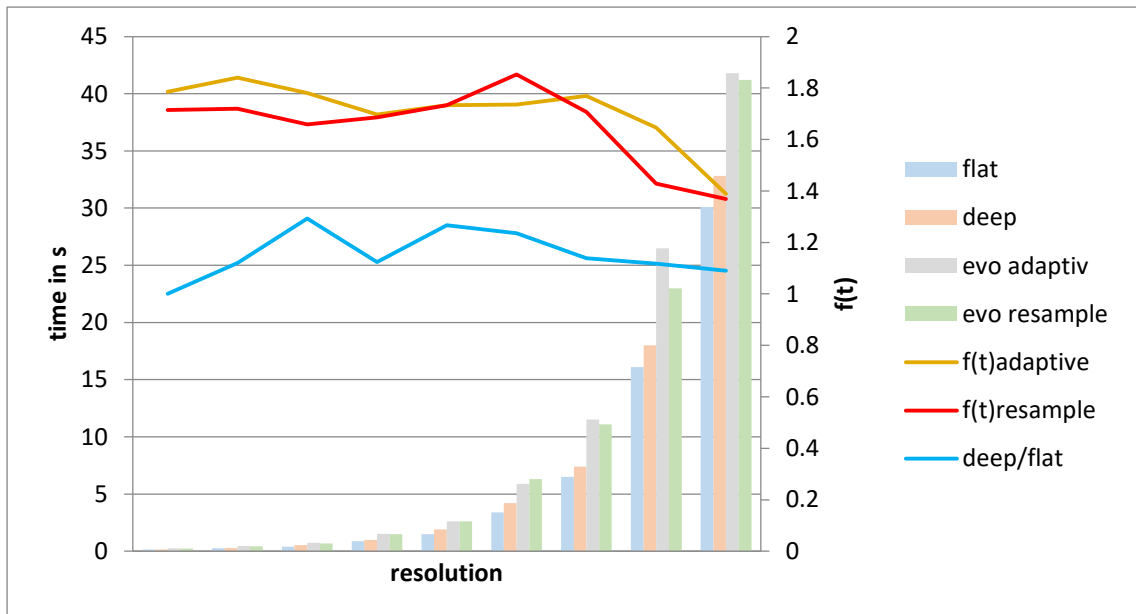
Having determined the similar rendering times in between differently sample optimized renderings and the hardware needs all further tests will be using CPU-based rendering and a maximum of two *Evotis* types.

5.3.2 Results from 2D Test 1

To assess the influence of different resolutions, and therefore file sizes and sample counts, the full range of flats, deeps, adaptive and adaptively resampled Evotis renderings from 3D test scene 1 have been processed with Nuke. The results, shown in *graph 5-19*, indicate a constant relation $f(t)$ between rendering times, even though file size differences are inconsistent.

With an average multiplying factor of 1.67 for the flat render time compared to the adaptive Evotis time, 1.4 times for deep, and a total render times of less than 45 seconds for 15k renderings it can be assumed, that the increased time needed, in this test, for Nuke renderings, are not going to be a factor for the acceptance and use of Evotis within the industry.

$$f(t) = \frac{time_{Evotis}}{time_{flat}}$$

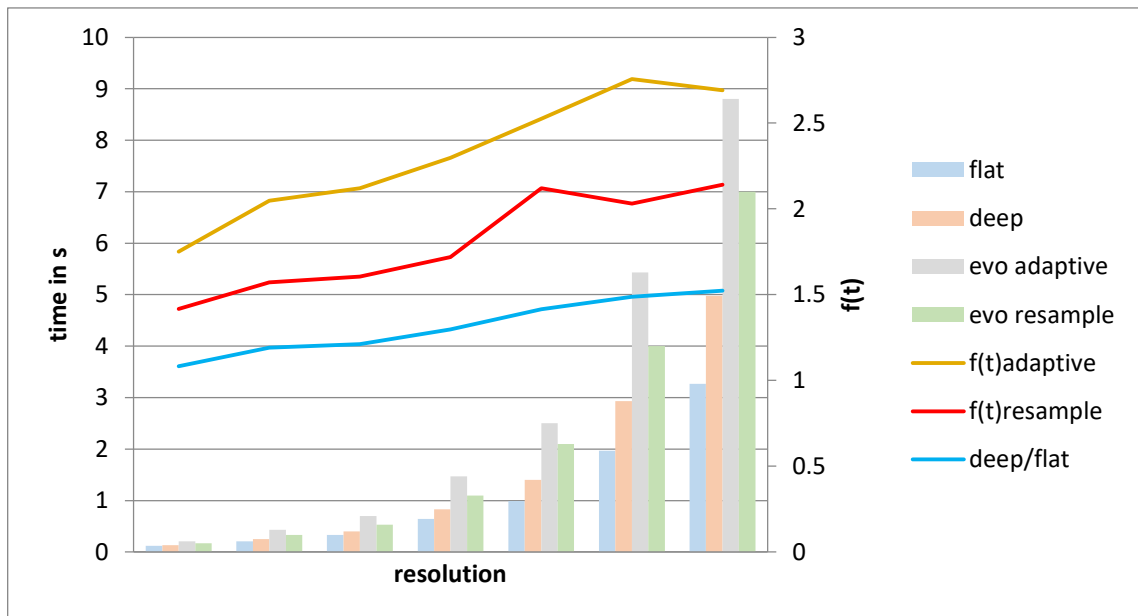


graph 5-19 - absolute and relative render time comparison

To verify these assumptions the same test over the full range of resolutions is performed on the resulted renderings from 3D Test 2, 3 and 4.

5.3.3 Results from 2D Test 2

In *graph 5-20* an increase of the relative time factor throughout the range of resolutions can be observed, indicating that a significantly increased number of samples throughout the image, in this case caused by the semi-transparencies of the motion blurred areas, constantly prolongs the render time. On the other hand, the same increase can be seen

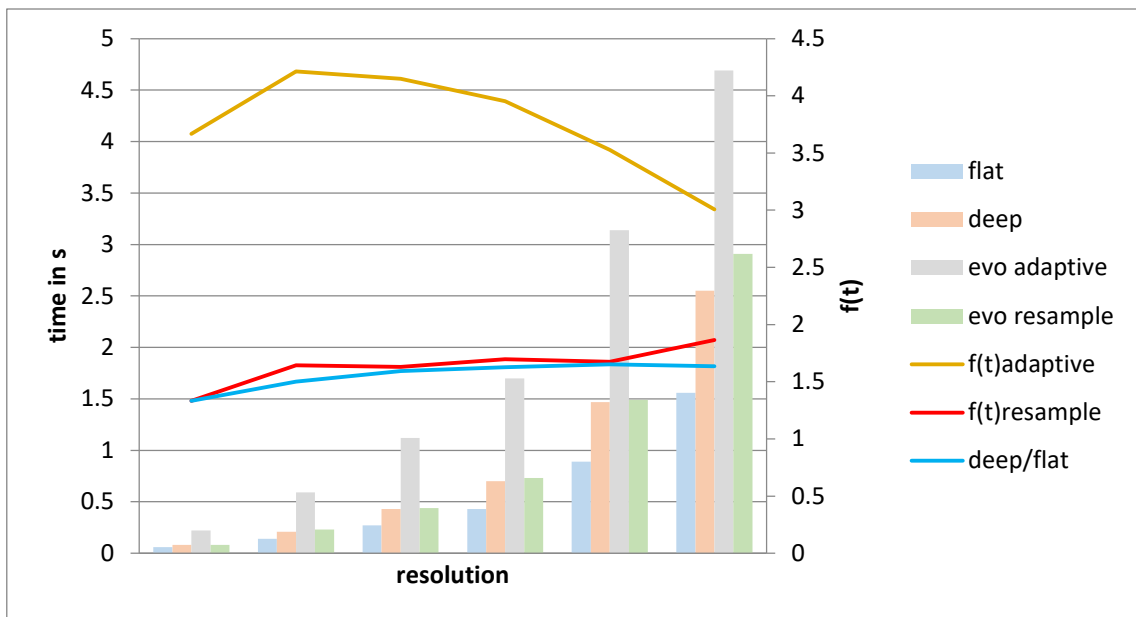


graph 5-20 - absolute and relative render time comparison of 3D test scene 2

in the relation between the deep and flat renderings, as the increase in semi-transparent areas affects the file size and rendering power needed in a similar way. Especially if comparing the deep results to the adaptively resampled ones it is obvious, that an increase of roughly 40%, all occurring within a total time range of less than 10 seconds, should be irrelevant for Evotis' success.

5.3.4 Results from 2D Test 3

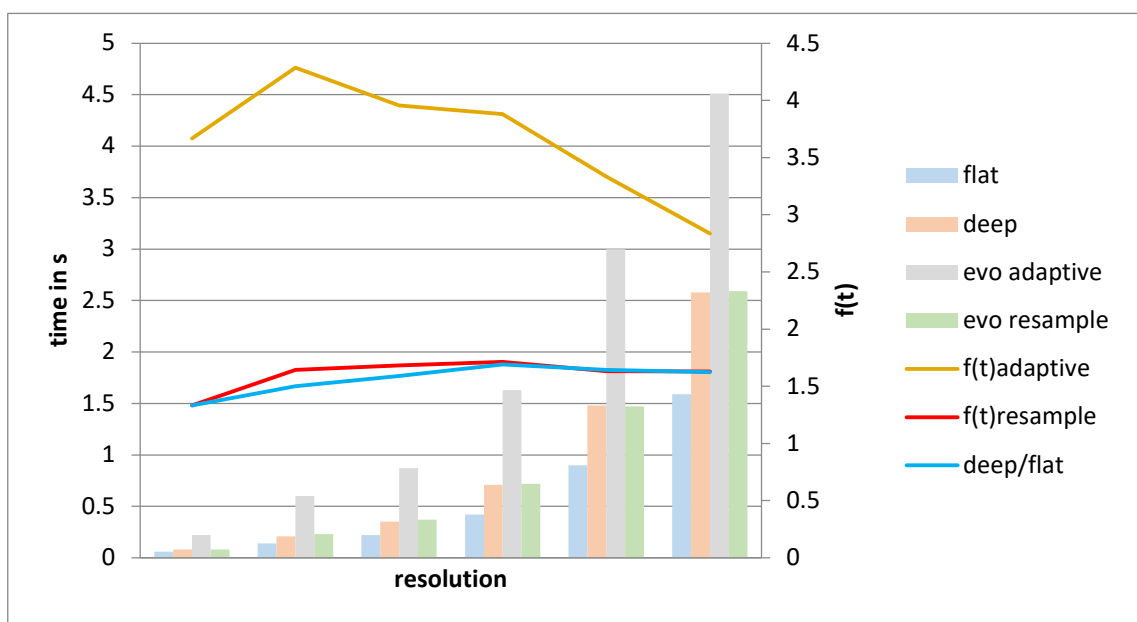
Graph 5-21 shows the results of test 3, multiple fur elements to provoke a large amount of object edges and semi-transparencies. As can clearly be seen the adaptively optimized renderings cope worse at lower resolutions than at higher ones, as indicated by the decline of $f(t)$ adaptive plot over the tested range. This is due to a various edges occupying a single pixel at lower resolutions, compared to fewer for the high-res renderings, as the image is spread over a larger array of pixels. On the other hand the graph also shows, that the deep/flat relation and $f(t)$ resample are highly similar, indicating a faster processing for Evotis, as the file size, as mentioned in *chapter 5.2.1.3*, is 4.5 times as large as the size of the deep file.



graph 5-21 - absolute and relative render time comparison of 3D test scene 3

5.3.5 Results from 2D Test 4

The results visualized in *graph 5-22*, derived from 2D test 4, are highly similar to the results of test 3 shown in *graph 5-21*, as both are based on the same scene, only with the addition of motion blur for test scene 4 to provoke even more samples being generated and saved. Therefore the observations and conclusions drawn are identical as before.



graph 5-22 - absolute and relative render time comparison of 3D test scene 4

6 Conclusion

After all the performance and file size tests described in the previous chapters it is clear, that Evotis, like most new VFX technologies, is very hardware intensive. This was, due to all the extra information preserved, to be expected to a certain degree. But with, on average, tripled 3D render times and significantly increase file sizes, it is questionable whether the discussed advantages will be enough to outweigh the strongly increased hardware needs.

Obviously, at the current stage of development, Evotis is not production-ready, but it is important to remember that Evotis is still in the beta phase, and performance improvements are very likely to happen, as stated by the CEO of GoGhost, Jared Sandrew, they constantly discover new bugs within Evotis, Nuke and the 3D renderers used, that often improve performance. It was also confirmed by GoGhost, that the samples should all be generated at render time, this means a full sample Evotis rendering should not take significantly longer than a flat rendering. The occurring problems were probably caused by an error within the beta version tested, therefore increases in rendering time compared to the test results presented in this thesis can be expected, if the claims from GoGhost are true.

The non-uniform approach chosen for Evotis bears the potential to simplify compositing workflows greatly as soon as there are transformational Nuke tools available. There will no longer be a loss in quality after repositioning was applied, there is no need for concatenation anymore, and no more filtering needs to be used for

transformations at all. Also missing, so far, is an Evotis expression node for Nuke that would allow the user to modify the Evotis files in a more technical way to explore their potential better. Offering these two nodes, of course, is only the beginning if Evotis is supposed to spread throughout the industry many more will have to follow, but with a transform and an expression tool a lot more potential could be explored and should therefore be the priority tools to be developed.

To improve interactive Nuke performance a proxy workflow should also be added, allowing the user to set a percentile or absolute threshold for the number of samples per pixel used while working.

But the main priority for Evotis should be to add deep support. As mentioned before, the ability to better work with atmospherics, creating holdouts and merging in depth were the main reasons deep became accepted as widely, even though it was slow and atmospheric deeps were huge. Therefore GoGhost should focus heavily on bringing depth support to Evotis soon, as this will be a key issue for its success, due to the fact, that replacing deeps is an option, replacing flats is not.

As for the hardware intensity of Evotis improvements need to be made to lower the file size in a more refined way, one possibility would be a sample threshold option that preserves the randomly scattered samples, and the relation between the amounts of different objects samples within a pixel, rather than using a sub-pixel grid, as this will preserve the non-uniform nature of the samples better while also reducing the file size in complex scenes.

More important though, than lowering file sizes, is shortening the render times, as a significantly prolonged render time will not be acceptable for a competitive company, while needing more storage might be. A possibility to improve render times, at least for the short-term, could be to render all Evotis files without any sample optimization,

which would be closer to flat render time, and then optimize within a separate post-render job. This way the accompanying flats get finished faster, allowing the artists to start working while the post-processing of the Evotis samples is still running. Ultimately the Evotis code will need to be improved further, to allow for quicker renders, as tripled render times will most probably not be an option.

On the other hand, the hardware intensity aspects will become less important over the next years, as hardware performance is constantly increasing, cloud computing is becoming a viable option, Athera¹ is launching and other very hardware intensive technologies are emerging, e.g. lightfield or deep learning, or on the breakthrough that will force VFX companies to invest heavily into even higher performing hardware anyways.

Another, not previously discussed, limiting factor for a potential success of Evotis is if and how GoGhost is planning to license it, as no company will be willing to invest time and resources for using Evotis if it is not an open standard, as any development and improvement will be limited to GoGhost only.

In conclusion it is very difficult to predict whether Evotis will be successful and widely accepted in the industry this early in its development. The many advantages, non-uniform images, resolution independence, appending samples and sub-pixel-perfect object separation, as well as the disadvantages, no samples in depth, longer render times, insufficient optimization options and larger files, have all been explained in detail. While including depth sampling will be essential, improving render times and minimizing file size will be important, but not as critical for the short term, 1-2 years, progression. After having included deep support broadening the Nuke support and

¹ Formerly project Elara <https://athera.io/>

developing new techniques and approaches based on a sample workflow, not easily possible with flats, will be decisive, while constantly improving performance.

If this development phase will be successful and Evotis becomes an open standard it could well be possible for Evotis to be an industry-wide replacement for deep within the next 5-7 years, but it will probably never replace flats, just as deeps will never be able to replace flats.

The other question is: will this timeframe be fast enough considering all the movement within the industry at the moment? Possibly a new approach will emerge over the next few years making rendered images as an intermediate obsolete altogether.

7 Further Work

To further determine the usability and performance of Evotis a continued evaluation will be necessary with newer beta version releases. The most important aspects will be the testing of the inevitable integration of deep data, and the handling of all subsequently arising problems with, probably again drastically, increasing file sizes and lowered performance.

Another issue in need of further testing is the rendering time, especially the claimed creation of all samples at render time, as this was not verifiably with this beta version. The claims that the prolonged rendering times were a bug and have been tested with older versions successfully, by GoGhost, need to be confirmed independently and furthermore a new range of rendering time and performance monitoring tests need to be conducted, to verify the shortened time applies to a wide variety of situations.

An additional interesting area for future research are the possibilities for new workflow approaches and techniques possible with Evotis, that were formerly not, or not as easily, achievable in compositing and rendering. Different workflow approaches using the rescaling capabilities could be tested and evaluated to define a range of possible best-practice solutions. The same could be done with an in-depth look into sample optimization and the resulting advantages and disadvantages for image quality, scalability, workability and file size, and cataloging these findings in a guideline for sample optimization.

Furthermore there needs to be in-depth research on the integration of Evotis into a large scale pipeline, distributed rendering, post-job optimizations, color and metadata

workflows, as well as into the development of Nuke gizmos to broaden the usability to postpone the pixel conversion, ideally creating a fully sample based workflow as far as possible.

8 Bibliography

Blinn, Jim. 1998. Dirty Pixels. *Jim Blinn's Corner* . 1998. ISBN: 1-55860-455-3.

Brinkmann, Ron. 1999. The Art and Science of Digital Compositing. s.l. : Academic Press, 1999. ISBN: 0-12-133960-2.

—. **2008.** The Art and Science of Digital Compositing. s.l. : Morgan Kaufmann, 2008. ISBN: 9780123706386.

Catmull, Edwin. 1974. A subdivision algorithm for computer display of curved surfaces. Utah : s.n., 1974.

Chaosgroup. 2018. chaosgroup.com. [Online] 2018. [Cited: July 05, 2018.] <https://docs.chaosgroup.com/display/VRAY3MAX/Adaptive+Sampling>.

Cook, Robert L., Porter, Thomas and Carpenter, Loren. 1984. Distributed ray tracing. *SIGGRAPH '84 Proceedings of the 11th annual conference on Computer graphics and interactive techniques*. New York : ACM, 1984. pp. 137-145. ISBN:0-89791-138-5 DOI:10.1145/800031.808590.

Corvazier, Cyril, Legros, Benjamin and Chikh, Rachid. 2016. OpenEXR/Id isolate any object with a perfect antialiasing. *Proceeding SIGGRAPH '16*. s.l. : ACM, 2016. ISBN: 978-1-4503-4371-8 doi>10.1145/2945078.2945136.

Friedman, Josh and Jones, Andrew C. 2015. Fully automatic ID mattes with support for motion blur and transparency. *Xroads of Discovery*. 2015.

GoGhost. 2018. goghost.com. [Online] 2018. [Cited: May 19, 2018.] <https://www.goghost.com/>.

- Goulekas, Karen E. 2001.** Visual Effects in a Digital World: A Comprehensive Glossary of over 7000 Visual Effects Terms (The Morgan Kaufmann Series in Computer Graphics). s.l. : Morgan Kaufmann Publishers Inc., 2001. ISBN:0122937856 .
- Hanika, Johannes, et al. 2012.** Camera Space Volumetric Shadows. *DigiPro* 12. 2012. pp. 7-14.
- Lokovic, Tom and Veach, Eric. 2000.** Deep Shadow Maps. *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*. s.l. : SIGGRAPH '00, 2000. pp. 385-392. ISBN: 1-58113-208-5 DOI: 10.1145/344779.344958.
- Möller, Michael. 2010.** Open Hardware Monitor. 2010.
- NIST. 2018.** The first digital image. s.l. : National Institute of Standards and Technology, 2018.
- Okun, Jeffrey A., et al. 2015.** The VES handbook of visual effects : industry standard VFX practices and procedures. 2015. ISBN: 978-0-240-82518-2.
- Pantaleoni, Jacopo, et al. 2010.** PantaRay: fast ray-traced occlusion caching of massive scenes. *Proceeding SIGGRAPH '10 ACM SIGGRAPH 2010 papers*. Los Angeles, California : ACM, 2010. ISBN: 978-1-4503-0210-4 doi:10.1145/1833349.1778774.
- Porter, Thomas and Duff, Tom. 1984.** Compositing Digital Images. *Computer Graphics Volume 18, Number 3*. 1984.
- Rosenfeld, Azriel. 1969.** Picture Processing by Computer. *ACM Comput. Surv.* 1969. 0360-0300, pp. 147-176. DOI: 10.1145/356551.356554.
- Seymour, Mike. 2014.** fxguide. [Online] February 27, 2014. [Cited: Mai 26, 2018.] <https://www.fxguide.com/featured/the-art-of-deep-compositing/>.
- Smith, Alva Ray. 1995.** Alpha and the History of Digital Compositing. *Technical Memo* 7. 1995.
- Straßer, Wolfgang. 1974.** Schnelle Kurven- und Flächendarstellung auf grafischen Sichtgeräten. Berlin : s.n., 1974.

Wallace, Bruce A. 1981. Merging and transformation of raster images for cartoon animation. *Computer Graphics Volume 15, Number 3*. 1981.

Wright, Andy, et al. 2016. Large scale VFX pipelines. *Proceedings of the 2016 Symposium on Digital Production*. New York, New York, USA : ACM Press, 2016. ISBN 9781450344296 doi: 10.1145/2947688.2947689.

Wright, Steve. 2010. Digital Compositing for Film and Video. s.l. : Focal Press, 2010. Third Edition. ISBN: 978-0-240-81309-7.