Methods And Technologies To Advance The Rendering And Compositing Process To Use A Given Timeframe More Efficiently (For Small Production Teams)

How Can Technologies Like Deep Compositing Help To Start The Look Development Process As Early On As Possible

Research Project
Justus Schmidt
747009

Supervisor: Martin Streit

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

Abstract

Deep data opens up the possibility to save not only 2D, but 3D pixels, leading to a rendered image which can be edited in 3D, giving the option to insert additional objects and effects without the need to create holdout mattes, as this method is not layer-based but relies on the depth values per pixel.
This allows for a much more flexible, non-destructive and procedural workflow. The rendering process in a production can start as early as the camera position and lighting is set, meaning that additional assets can always be added later on without any problems and worrying about holdout mattes. Also, volumetric objects can be rendered separately without having to consider the intersecting objects, and changes can be made after rendering. This workflow is handy for small teams with tight deadlines, as the options for rendering on the last second are limited. Only the storage needs to be more capable, as deep data image files take up substantially more space than traditional 2D images, which also leads to a slower workflow which comes with working with large files. Deep Compositing is not a faster rendering approach, but a more flexible one.

Motivation

Producing an animated short or commercial within a given timeframe and a small team usually comes along with numerous challenges, especially with only a small budget available. The jobs have to be evenly assigned for the most efficient workflow, as the deadlines are tight. Thus, everything has to go fast and each step has to be well planned and executed. Particularly rendering, as one of the last steps in the 3D pipeline, takes up plenty of time, if the render jobs are not handled by multiple computers (render farm). Therefore, small productions oftentimes lead to plan to render as early on as possible. [1] To render a scene, everything (assets, textures, materials, etc) needs to be finished. If something is not finished in time, either the render time gets postponed or the asset will not make it into the final shot. [23]
Thus, rendering pretty much can be seen as the deadline regarding the 3D production. To add e.g. an asset later on usually would take up a lot of additional compositing work. If one render did not work for some reason or something needs to be changed later on, this might not be possible as changes take time and the tight time schedule does not allow for that.
As rendering oftentimes gets postponed for many reasons, the time available for final Look Development and compositing gets less and less. This, of course, would have an impact on the final look of the movie.

Deep Compositing is a technology which sustains the 3D data in the rendering, which promises the workflow to be more flexible, as every asset in the scene could essentially be treated as a 3D object and not a 2D image. [2] This would allow for a both procedural and flexible workflow, as every object could be handled separately and the scene could be simply put together in compositing, with the option to move assets around and exchange objects, or add objects later on.
Can Deep Compositing, therefore, come in advantage when utilized in a small production team? As one could render early on and simply add objects which were not finished before, thus more time would be spent on Look Development

[1] "Finding freedom through deep compositing techniques", Foundry, Retrieved December 07, 2018.
[2] Heinen, Patrick "Deep image compositing in a modern visual effects pipeline", Stuttgart Media University, Berlin, March 18th 2013
[3] Interview with Sebastian Metz, "WeCanDance", November 2018

and one would not have to wait for all assets to be finished, which would save time.

Definitions

On the following pages, terms and abbreviations will be used for ease of use, which will be briefly explained.
Deep images, *Deep data* or *Deep pixel images* will be referring to the files and workflow of Deep Compositing, therefore images which do not only have 2D but 3D pixel information.
2D images or *traditional* images refer to the counterpart, images which do not have a 3D value incorporated into them.
3D workspace means the generally work in the 3D environment, with deep data and polygonal objects.

State Of The Art

Traditional 2D Workflow

In a small production team with little budget, rendering is a bottleneck.
To save precious time, the production is shot-based. Meaning that not all assets are produced first, but the priority rather lays on finishing up shot by shot, this way not all shots has to be rendered in the end, but rendering can start fairly early in production, while the rest of the team still works on additional assets and models[4].
Once the first 3D scene has been finished building up, all the assets are in place and materials are applied, the overall look development process begins with the next pipeline step, Lighting. [5]
As it is not possible yet to render complex scenes in real time, the ighting artist oftentimes has to work with a rough estimate of what it will look like later on.

[4] Interview with Sebastian Metz, "WeCanDance", November 2018
[5] Pellacini, Fabio "the 3D production pipeline", 2009

In a smaller production, single frames for testing purposes are rendered and already edited in a compositing software [6]. As the final Look Development takes place in the compositing suite, where the shot can be tweaked, and post effects are applied.
Before rendering, however, the 3D scene needs to be split up in render layers and passes.
As one would like to have as much creative freedom in the compositing suite and in the composting process, prior to rendering it is usual to logically split up the 3D scene for the best flexibility, so ideally each aspect of the shot could be treated and tweaked separately later on and objects could be replaced or re-rendered.
Therefore, characters are usually split up from the environment and rendered on a separate layer.
Of course, if there are objects in front of the character, they need to be taken into account, otherwise, the character cannot correctly be layered on top.
Thus, the occluding object has to rendered separately as well.
In some occasions, however, that is not possible. E.g., if the occluding object is volumetric.

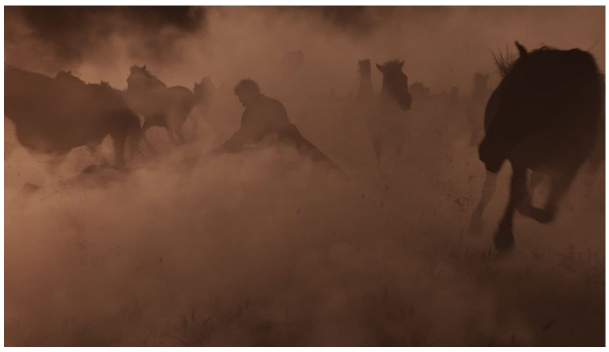


Fig. 01:
Smoke Holdout Matte for the movie "Vampire Hunter" by Weta Digital.
(Failes, Ian, "Vampire Hunter: two killer sequences", 1. July 2012, https://www.fxguide.com/featured/vampire-hunter-two-killer-sequences/)

If the character is placed in smoke, for example, one can not simply render the smoke separately. Instead, a so-called holdout matte needs to be created. The smoke is rendered, but the character is included in the render having a matte shader assigned. This way, the character can be placed inside the smoke.[7]
Using that method, the character and the smoke are basically linked to each other. If changes are made to the character, the smoke also needs to be

[6] Interview with Sebastian Metz, "WeCanDance", November 2018
[7] "Finding freedom through deep compositing techniques". Foundry. Retrieved December 07, 2018.

re-rendered. “While this works, it can be incredibly limiting as you are committing to the placement of elements and lighting without seeing the elements in the context of the entire shot.”[4]

In addition to render layers, passes (AOVs) are rendered, which are layers rendered alongside and oftentimes saved inside the same EXR. These can be object IDs, a Z-Depth pass or separate reflection, light or diffuse passes.
In compositing, render layers and passes are used to alter the image in 2D.
If a rendering fails for some reason, the image has to be either rerendered, at least parts of it. This also applies for changes that are made after the rendering.

Deep Workflow

In the traditional 2D workflow, up to the point of rendering, everything is worked on in 3D space. When an image is rendered, the 3D information is converted into pixels and therefore, in the traditional workflow, the 3D information is lost.
As a 2D image (at least most of the time), is what is desired in the end, this step, of course, makes sense.
But as there is one last production step still to be done, compositing has to work with the 2D information only. But as the most and final look development of the image is defined in this process [8], it would seem like there is a missed opportunity to only work in 2D.

Deep Compositing, on the other hand, is a technique which uses a format to store not only the 2D pixel information.
“With deep data, every pixel contains multiple samples at different depths, with each sample holding colour data, camera-relative depth, and opacity information on a per-pixel level.” [9]
Regarding the example with the smoke mentioned earlier, the smoke and the character could be rendered separately and then be merged in the compositing software thanks to the 3D Information, allowing the compositor work in 3D space instead of 2D. “So you can render the smoke without worrying about any

[8] Interview with Sebastian Metz, “WeCanDance”, November 2018

[9]“Finding freedom through deep compositing techniques”. Foundry. Retrieved December 07, 2018.

elements that might be inserted into it, you just insert them in the comp and it just magically works." [7] As one is working in 3D compositing space, one can not only stack the rendered objects together but also move them around in space. "This has a tremendous practical use beyond volumetrics, too, as it's useful in nearly any situation where you have many overlapping elements at varying Z depths that you want to combine in comp. Achieving this without deep compositing requires hold-outs that need to be merged in the correct order and can't be changed without re-rendering. Deep Compositing also gives compositors more creative choices, says Christy Anzelmo, Senior Commercial Product Manager at Foundry. Essentially, Deep Compositing lets you delay more decisions, providing compositors with more options to achieve a seamless, natural-looking composite and influence the look of their shots.
Bigger choices are being made at the point of comp far more frequently than ever before, particularly as modern VFX productions now lean on compositing as part of the creative process from the very beginning, rather than as a step at the end of the pipeline. The flexibility deep compositing affords is an enticing draw for more studios, now that the technology and processing power has caught up to make handling the extra data a far smoother affair. [...] Adding assets was also really simple - we saved a lot of time since we never had to deal with holdouts, says Blur CG Supervisor Sebastien Chort. We were able to just render each asset under the correct lighting condition and drop it into the comp." [10]

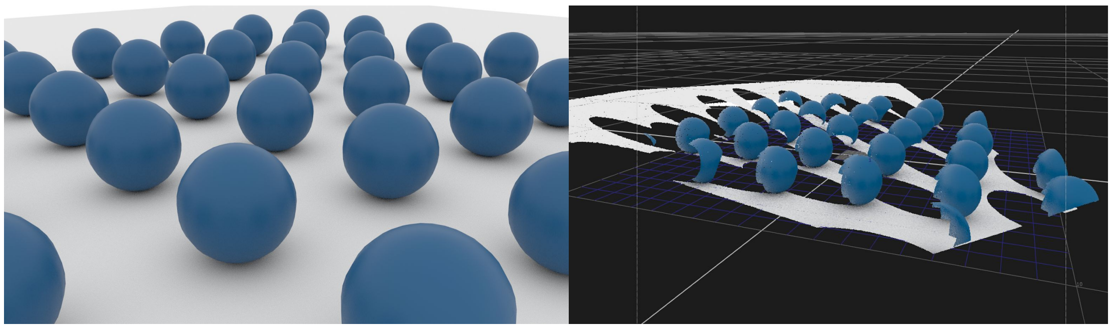


Fig 02 and 03 :

The rendered image in 2D (left) and Deep Data visualized as a 3D Pointcloud inside Nukes 3D Space (right)

[10] "Finding freedom through deep compositing techniques". Foundry. Retrieved December 07, 2018.

Approach

In the following, it is going to be investigated how and how far Deep Compositing can help to ease the production pipeline of small teams in order to relax the Look Development process by additionally providing as much creative freedom as possible. Also, it would increase the flexibility of rendering, which hopefully relaxes the time schedule at the end of the production.

File Size And Render Times For Deep Data Images

When it comes to the use of Deep Compositing in small production teams, one first has to look at the possibilities a small team has got. Due to a presumably smaller budget and less workforce, the capabilities are limited in comparison to a larger company.
So, in which way is Deep Compositing even worth being implemented in a smaller pipeline?

Let's first take a look at the storage which is needed to store a Deep data image. As a Deep data image contains consistently more information than a traditional image, the file should be much larger. A Deep image contains multiple values per pixel. How many, however, completely depends on the content which is in the image. For a volumetric object stored in the image, much more samples are saved per pixel, as it is no hard surface material and there needs to be (colour) information per stored depth sample. Vice versa, for a scene which only contains hard surface objects, the number of samples is much less.
The standard data format for a deep image is Open EXR 2.0. In the following, the data structure of that format is explained.

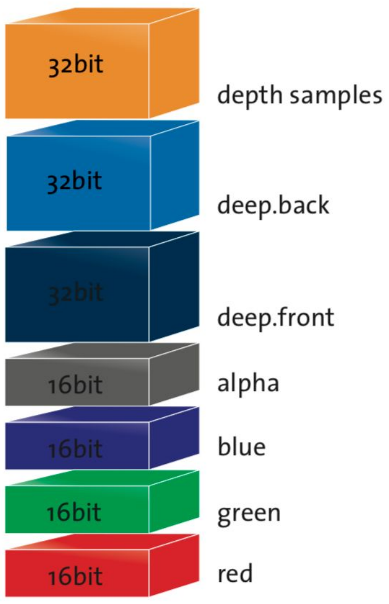
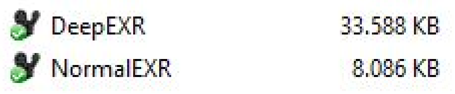


Fig 04 and 05: Bit depth of Deep EXR elements (source: Patrick Heinen, top), and actual and general file size difference between traditional and deep images (bottom)



DeepEXR	33.588 KB
NormalEXR	8.086 KB

"With no additional information – in other words: with a single sample per pixel – a deep image will have an increased file size of 96 bit per pixel compared to a traditional image. This is 32 bit each for deep.front, deep.back as well as the sample count. With every additional depth sample the file size increases by another 128bit, 64 for rgba and 64 for deep. This means that a 16 bit deep EXR file will have double the size of an rgba 16 bit EXR. This seems drastic but is comparable to an additional layer in a multilayer EXR file. A further increase of the file size is then directly proportional to the sample count."[11]

Additional samples can come not only from volumetric objects, but far more common are semi-transparent pixels produced by motion blur or anti-aliased edges.

[11] Heinen, Patrick "Deep image compositing in a modern visual effects pipeline", Stuttgart Media University, Berlin, March 18th 2013

As the number of saved samples could be infinite, depending on the object that is rendered, the overall number of stored samples can be capped.
Also, it is advised to store additional layers and AOVs in the same EXR and not in single files, as the depth information would only be stored once, saving memory.
When it comes to render times, the time needed to render a deep image EXR is only increased by the time it needs to store the image on the hard drive.

Performance While Compositing

As working with Deep data images happens inside the compositing suite, the performance is a tremendous factor. If it turns out, that it is not possible to do compositing in almost real time, as it would be for non-Deep images, it would also mean that one would have to consider in what way implementing Deep would be worth it and improve a pipeline, especially if time is the biggest limiting resource.
In comparison to rendering, in compositing it is paramount to work as fast as possible. So not only is the CPU performance important but memory and network. "Image data has to be accessed at a very high frequency and caching it to disk or memory helps a lot to speed this up." [9]
As Patrick Heinen examines in his paper "Deep image compositing in a modern visual effects pipeline", "As the amount of data that has to be moved poses a big problem for connection speeds, working from local drives should give better performance than working from a network share. [...]it becomes clear that high transfer speeds are important as long as the processing is fast. But if the processing takes long, the IO operations have to wait and the processor becomes the bottleneck.[...]The impact of deep images on compositing are apparent. Due to the elevated amount of data that needs to be loaded, cached and processed, processing times rise and make a real-time feedback nearly impossible. To allow for a

smooth work concepts like proxies or regions of interest should be considered.“ However, that was published in 2013. Nuke by Foundry, the compositing tool in which Deep Compositing has been implemented since 2010 [12], has since been updated to better handle Deep data.
Foundry claims that, also due to the improvement in processing power over the last years, Deep Compositing will find its way to mid-size studios and grow rapidly[13].
Nuke has got Graphics Card Acceleration and Solid State Drives have been becoming more and more affordable over the last years, promising a huge speed upgrade in comparison to classic hard drives. [14]

Working In Nuke 3D Space

Currently, the only commercial compositing tool to handle deep data in its entirety is Nuke by the company Foundry. Fusion by Blackmagic Design is also able to work with Deep data, but does not have as many tools as Nuke. [15] So this paper is focussing on Nuke.
It allows to read and write Deep data and offers a variety of nodes to incorporate Deep into the workflow.
It is however not possible to work in Deep and traditional, as Deep data first needs to be converted to be then processed in the traditional 2D way.

The following statements are based on the experience with working with Deep data in Nuke.

The compositing software Nuke has got the feature, next to the classical 2D viewport, to work in true 3D space. In addition to image planes, it is also possible

[12] Seymour, Mike “The art of deep compositing”, fxGuide, February 27, 2014
[13] ”Finding freedom through deep compositing techniques”, Foundry, Retrieved December 07, 2018.
[14] Kerekes, Zsolt,”Charting the Rise of the SSD Market”, StorageSearch.com, Retrieved January 12th, 2018.
[15]Andre, Pascal, April 29. 2016, https://render.otoy.com/forum/viewtopic.php?f=27&t=53775, Retrieved January 12th, 2018.

to load up polygonal assets or even an entire Maya scene. One can even add lights and shaders, so building up a 3D scene in Nuke is possible, in theory. Nevertheless, most of the time this feature is used to insert in effects like smoke, volumetric lighting or particles.
Using (animated) textures on 3D polygonal planes which are layered not in 2D, but located in the rendered environment 3D space, due to the imported camera data, and can, therefore, be coherent with the correct camera movement, as it is essentially part of the 3D scene. If the plane is occluded by geometry, the relevant geometry can be loaded in as well and given a matte shader to create a holdout matte for the texture plane. To get from the 3D viewport to the 2D workspace within Nuke, the 3D scene needs to be "rendered" out. In the traditional workflow, this rendering would then be layered onto the previous 2D rendered frames from the 3D application.

For rendering, Nuke has got a few options available. The nodes "ScanlineRender", "Rayrender" and Pixars Renderman renderer are built in, but "Octane" is also available as a plugin. While most of the renderengines support shaders and even raytracing, they are only capable of giving out traditional 2D data at the moment. Only "ScanlineRender", which does only support "flat" shading, renders out Deep data. So at the moment, the possibilities in that regard are limited. [16]

In combination with Deep rendering data, the compositing work for 3D data gets heavily reduced, as additional 3D data could simply be merged with the rendered scene with no other geometry needed for occlusion. Within the Deep Workflow, every object is placed "in" the actual 3D scene and not layered on top of it.

This easy workflow makes it possible to only load up the corresponding camera per scene. With the camera rendering the 3D Scene in Nuke, which contains all additional geometry and particle systems, all is done on a per shot level.
Look Development and compositing is therefore handled on an environment or scene level, and not much needs to be worked on for each shot.

[16]Nuke Documentation, https://learn.foundry.com/nuke/content/learn_nuke.html, Retrieved January 12th, 2018.

For example, given a scene which takes place in an old building. On the compositing stage, volumetric lights (“god rays”) in the form of textured polygonal objects are placed within a 3D Nuke scene, as well as a Nuke particle system for a dust effect.
Every shot can now be rendered inside the 3D program. In Nuke, the Deep render data is imported, as well as the camera data of the shot. Due to the correct alignment in Nuke, all the effects are in the correct place and therefore less time is spent on recreating and adjusting effects for each shot.

Working With Deep Compositing In Regard Of Flexibility

While Deep Compositing has been standard in VFX pipelines of some of the largest studios like Weta, ILM or Blur Studios [17], it has yet to find its way into smaller companies and studios. As explained earlier, Foundry expects a large growth in usage of Deep Data.
Since smaller companies and teams do usually not have a complex pipeline system which controls their data, so switching to a new workflow approach should generally speaking be easier for smaller productions.

The big advantage of Deep image compositing is, of course, the flexible and sandbox style workflow, meaning that objects can not only be rendered separately very easily and just put together in post, also it is possible to later move Deep objects around as in any other 3D viewport. One can translate, rotate and scale them. Also, it is possible to cut off/mask objects in 3D, meaning that objects could be removed from the scene by using the 3D data.
Having a scene build up entirely in 3D, without the need of a complicated network of hold-out mattes and masking nodes also means that all post effects continue to work, when the Deep data is altered, exchanged or moved.

[17] ”Finding freedom through deep compositing techniques”, Foundry,, Retrieved December 07, 2018.

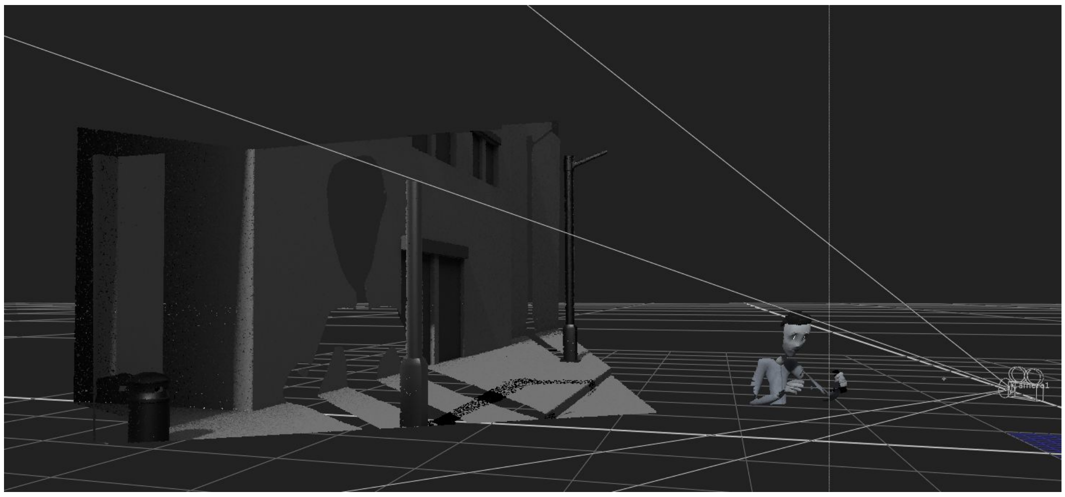


Fig 06 . Nuke can display a deep rendered image in 3D space, using a pointcloud. This way one can see how the image is saved. The environment behind the character, for example, shows a cut-out in shape of the character, since the pixels behind the character have not been rendered. If the polygonal scene was to be loaded into Nuke as well, it would be congruently.

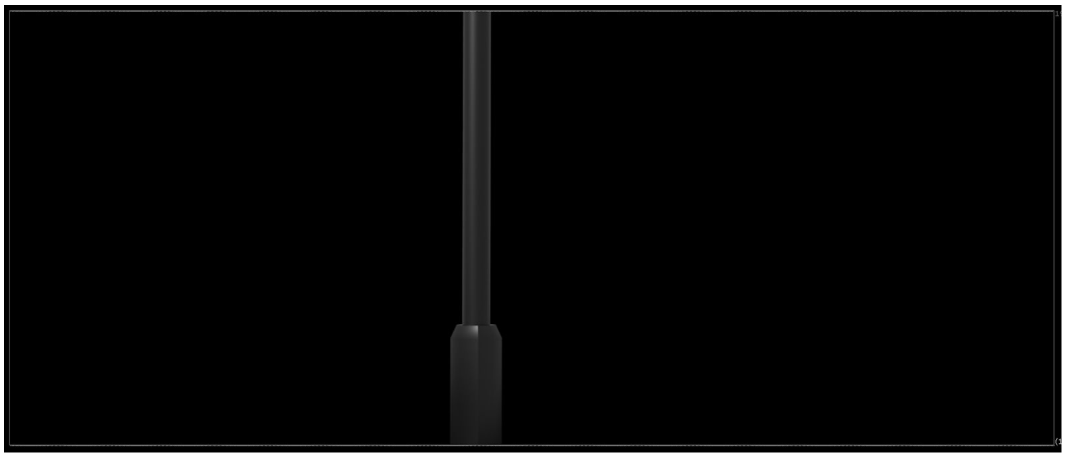
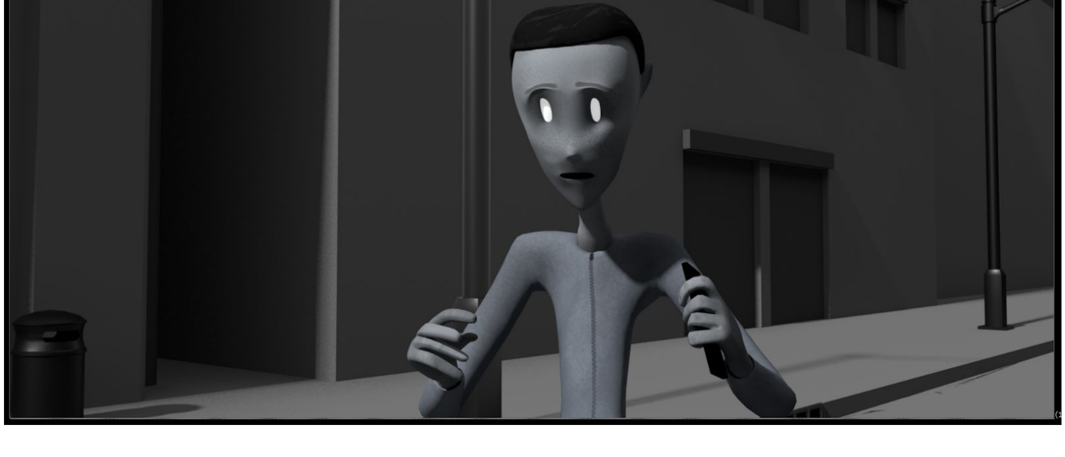


Fig 07 and 08. The street lantern, rendered separately can easily be merged into the scene as shown below. The character is occluding the pixels of the lantern behind him, as the characters pixels are in front of the lantern when seen in 3D space.



But Deep Compositing also has its limits, when the objects have to interact with each other. For example, when there is a rendered environment and the character is rendered in separately. In the compositing stage, one would want the character to be of course placed inside that environment. To fit in, the character has to interact with the environment at least on a Lighting level. There need to be shadows of the character somewhere on the environment floor, also the character is affected by the light rays reflected by the environment, meaning global illumination.
As both are dependant on each other, rendering both, of course, needs to be set up correctly in the 3D software of choice. The shadows are usually rendered as a separate file.
In the traditional pipeline, both objects are now layered on top of each other. Deep data, however, enables additional options. As explained earlier, objects can be moved around. So in what way would it be possible to change the position of the character?
Since it is dependant on the overall lighting as explained earlier, this is exactly how much it can be altered. While Global Illumination is usually not as noticeable when slightly off, the shadows, of course, have to react to the shape of the environment. As the shadow information is "baked in", currently there is no way to have them change on runtime within the compositing software.

There are a few things possible at the moment, however.
Marco Romeo, for example, has developed a script for Nuke, which enables Screen Space Ambient Occlusion for Deep Compositing. It compares two separately rendered deep data files and generates a 2D map to be layered on top of the plate.
This can come in handy when an asset would be added to a scene later, without rendering out a shadow map for that object as it would usually work, one can use that script to have the object appear to be inside the scene. [18]

[18]Romeo, Marco, "Nuke Screen Space Ambient Occlusion for Deep Compositing, " https://vimeo.com/94854706, Retrieved January 12th, 2018.

Also, within Nuke it is possible to convert deep data in geometry and back. Although of course, it is easier to just load in the geometry from the 3d program, which can then help relight the scene to some degree and create different effects.

One renderpass which is particularly important in the traditional flat composting workflow is the Z-Depth channel. It stores the depth of the scene using a greyscale image from an in-camera perspective. Using this pass enables to work with depth information without having 3D pixels, but rather use the black and white values to distinguish the depth differences per pixel.
This depth data is used to create post effects like depth of field. Roughly explained, one can choose one greyscale value of the z-depth channel to be in focus, while the other pixels on the coloured image get blurred accordingly.
Another post-effect using z-depth is to add fog or mist to the 2D plate.
This method of using z-depth, however, comes with a few downsides. First, the pass is not anti-aliased, meaning that the edges around the objects, especially when fine and complex shapes like fur, for example, are displayed, artefacts can easily be seen.

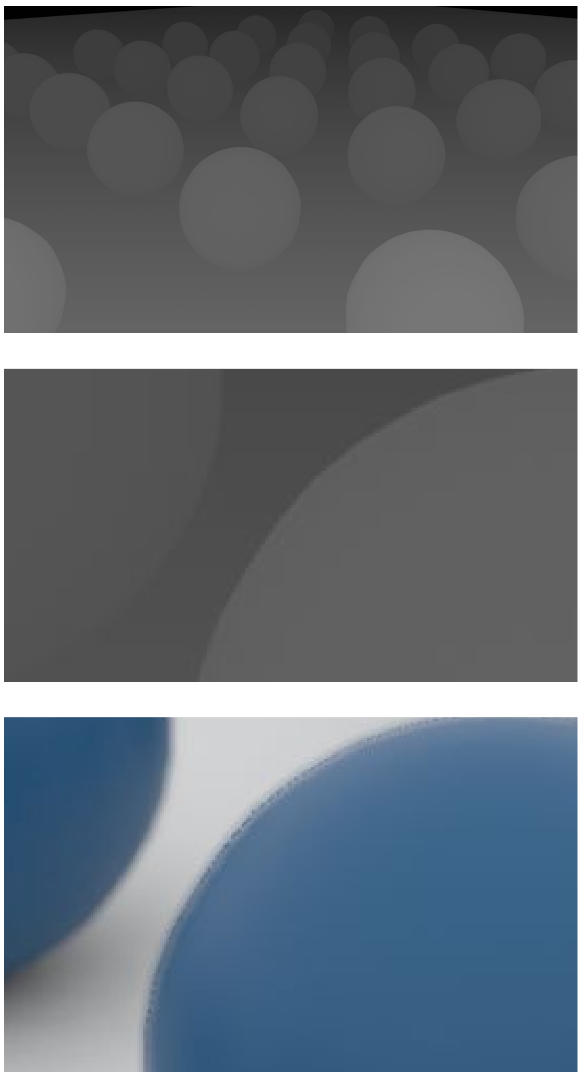


Fig 09,10,11: The traditional way to have access to depth data is to render the Z-Depth channel (top). However, as the result is not anti-aliased (middle), this can easily result in edge artefacts, e.g. when used for a depth of field effect (bottom).

Another problem occurs when rendering non-opaque objects like smoke or transparent objects, as it is not possible to save multiple depth values per pixel as it is for Deep data. Instead, every pixel can only have one colour value, making it impossible to display e.g. motion blur as well as explained earlier.
All this can be solved by using Deep data instead. For depth of field, there are solutions like “Bokeh” by Peregrine Labs, which allows creating a shallow DOF effect by using Deep data, which promises a realistic and natural blur, as real 3D data is used. Therefore, edge artefacts are not possible, and motion blur rendered in-camera does not produce any problems.

But there is one more thing.
When rendering objects separately and put them together in post, one can not simply go ahead and merge the Z-Depth. Instead, it would need to be layered in the same order as the objects would be, meaning it would require much more effort than simply use DeepMerge Nodes.

Within the deep workflow, having the option to add objects later on and sustain all post effects due to the “procedural” system to simply merge objects together shows that deep compositing can indeed be a huge timesaver.
This, of course, is an enormous benefit when it comes to rendering, as the image could be split up and larger sections could be rendered when they are ready, not-finished assets and characters could simply be added later. That relieves the render time in the end, as rendering and lookdev can already start by the time the camera position is set. Pretty much anything else can change later on.
As the workspace is 3D, all rendered objects/layers can be merged and still keep the correct three-dimensional information. Meaning that the Environment can be rendered separately without any masks of foreground objects. The Character can be merged into that environment and is occluded by all the correct parts.
On the other hand, one has to take into consideration, that more time overall would be spent rendering, as areas would be rendered which would be otherwise occluded using the traditional workflow.

Regarding Look Development, it, therefore, is possible to start the otherwise oftentimes rushed Look Development process early on, as compositing can be shifted to an earlier time frame since it can be started when not everything has to be rendered or even finished. Practically speaking, the environment does not have to be completely modelled and textured for example, because it is possible to work with rough placeholders within Nuke instead. Of course, this procedure is not limited to the deep workflow as placeholders and test frames of course can be rendered and worked within 2D as well. But Deep Compositing enables a truly non-destructive workflow, as the environment could simply be replaced by the newer one, once it has been rendered, keeping all the effects, particle systems, etc in 3D space and do not have to worry about masks and hold-out mattes. If objects would change, no matter if they would be in the fore - or background, the scene would be easily replaceable. This, of course, is applicable for all assets and objects which are needed in the scene, as long as they are not dependant on each other.
For example, if the shadow of the character is visible on the environment (as it most usually would be), the shadow, of course, would have to be re-rendered as long as the objects interacting with the shadow had changed.
In another example, a character model had not been finished but all the environment and assets, including the characters hair and cloth simulation, have been rendered and composited. For the character effects like hair and cloth, a version of the character has been used which did not have every feature yet. The finished character can now easily be merged into the compositing scene, once been rendered, and it works, if the model did not change dramatically in comparison to the earlier version of the character. Added displacement maps for example or additional sculps, which would have been rendered into the hold-out mattes otherwise, can now be merged without a problem.
Regarding the concern of a slow working speed due to heavy data, it was discovered that it is completely tolerable and does not affect the overall working speed for a relatively “usual” amount of layers (5-10).

Future Outlook

Now that it is explained what is currently doable with Deep data, it is interesting to imagine what could be possible in the future. Or better, in which direction Deep data could develop to further enhance the compositing workflow and beyond.
As finalizing the image more and more shifts towards compositing, enabling to change everything on the last second of the production, we can take a look at how this approach could be possibly improved. Furthermore, as Deep data could be used for much more than just compositing, In the following, it will be examined how the technology could be used in combination with other media, to further expand possible applications.

Deep Masks

Currently, as explained above, it is possible to crop or mask out Deep data using 3D gizmos. Meaning that within Nukes 3D space, one can define the region of interest using a 3D cube. For example, if one has got a natural environment rendered in Deep, but would only want to have one tree to be used in the composition, a squarish box could be drawn around the tree to only have it masked. The same procedure could be applied if one would like to get rid of the tree, only that it would be masked out (leaving a hole in the ground).
At the moment, this method works to some degree, if the object stands on its own and is not intersecting with other objects, or if the object could roughly fit inside a squarish box. The method is, however, not very flexible, as most of the time objects stand near each other or do not fit easily inside a squarish shape.
For that it would be useful to have the option to mask or crop out Deep data with polygonal objects, meaning for example that if one would like to mask out the previously mentioned tree, the actual 3D model of that tree could be used to mask it out, comparing the 3D pixel data of the deep image to the polygonal object.
Of course, that system also would not be flawless, as it does not incorporate motion blur, depth of field or volumetrics. But it would mean that the compositing artist would have way more flexibility to work with the 3D scene.

Additional Deep Nodes (DeepBlur etc)

Currently, there is a specific set of Deep Nodes in Nuke, developed for the use with Deep data and oftentimes comparable to their 2D counterpart. What "Read" does for 2D files, "DeepRead" has to offer for Deep data.
Nonetheless, there is only a certain number of Deep nodes available which already offer a valuable base to work with. Also, not every 2D node could be translated to 3D, as it would not make sense in some cases.
However, expanding the list of Deep nodes certainly would be useful in the future.
For example, there currently only is a node which blurs pixels on a 2D basis.
A "DeepBlur" node as an alternative to the traditional blur node, for example, would mean a smoother workflow within the Deep data pipeline.
Blurring 3D pixels, of course, would mean horrendous computing power, as every pixel would need to be blurred not only in 2 but 3 directions.
However, for the future, it would mean that within Nuke, one would be able to stay in the Deep data stream longer and would not have switch back and forth between 2D and 3D.

Deep Motion Blur

Other than depth of field, motion blur is very hard to "fake" in compositing. It is possible to render a motionvector pass out of the 3D package, which allows adding a motion blur effect in Nuke for "simple" movements. However, when two objects intersect for example [19], this system fails and the blur does not work.
So it has been industry standard to do the motion blur "in camera", meaning it will be rendered inside the 3d package into the footage.
This, however, does of course not provide a flexible workflow, not to mention that renderimes are much larger this way.
Deep data could help to solve this problem[17], as the motionvector pass could be rendered in 3D, which could give a motion path for each pixel in 3D rather than 2D, meaning that each object could be differentiated by the compositing software.

[19] Heinen, Patrick "Deep image compositing in a modern visual effects pipeline", Stuttgart Media University, Berlin, March 18th 2013

Another solution could be provided by the 3D objects themselves. As it is possible to load in entire Maya scenes including animations, the motion data could be extracted from there and then be applied, even to a 2d plate.

The only problem could exist from having too few pixels. As Vectorblur would blur the moving object and have it less opaque, due to the more transparent blurred pixels, the background behind the object would be visible. As there is no colour information, it would be rendered black. If it is not possible to render the objects separately, a solution could be a rendered-in "overlap" of pixels, which could be easily done with deep images. [20] In the rendering process, more pixels would be rendered than are seen by the camera. This approach could come in handy later on.

Deep Relighting

Relighting describes the process of changing the light of an image after it has been rendered. Naturally, the light is dependant on the materials of the objects and therefore it is hardly possible to change a lot in compositing. There is however the approach to separate each light into different passes/layers so that in compositing it can be combined in different ways. To relight the scene in Nuke, it is possible to use the geometry of the scene and a nuke light to create an additional lighting pass to add into the formula.
This method does not work with volumetric or motion blurred objects, as well as any other semi-transparent pixel. If it was possible to use the available Deep data instead of hard surface polygonal geometry, this problem would be fixed, as the light would be calculated on a per-pixel basis.

Also, the material information could be stored per pixel, if it was highly reflective, matte, which colour it would have in which scenario, etc. This way, even the shading could be calculated inside of Deep data, meaning that light passes could be exchanged entirely with new ones, giving the compositing artist even more control over the final image.

[20] Heinen, Patrick "Deep image compositing in a modern visual effects pipeline", Stuttgart Media University, Berlin, March 18th 2013

3D Shadows

Currently, when moving an object in the Deep workspace, it can be translated and rotated to wherever one would like. If however, there is a shadow map rendered and therefore attached to the object, it of course glitches into other objects and does not interfere correctly with the surrounding elements.
In combination with Deep relighting, having some sort of intelligent shadow system would enable way more compositing options and the true option to fully change the 3D scene in post.
If the 3D data of the Deep images would be more incorporated, it also would be possible to have some sort of global illumination to be calculated using the Deep image. In doing so, additional, non rendered objects could be added easily without complex colour correction. If the Deep data would work in combination with the physical based render engines inside of Nuke, it would open up endless possibilities to have physically based, Nuke rendered objects inside the Deep data scene.

Improving The Workflow Approach

Currently, when working in Deep, there is no visual feedback. Inside the Deep workflow, one works inside the node tree and does not have any insight on the 3D aspect of the scene, unless the Deep data stream is connected to a visual "DeepToPoints" node, which converts the Deep data into a visible pointcloud inside of Nukes 3D workspace. When it comes to the actual editing and merging of the Deep data, one has to work "blind" and only with nodes, without seeing the scene in 3D, but only the 2D outcome. This makes working with Deep data very counterintuitive, as there is no traditional viewport with direct feedback. Also, to move Deep data "objects", there is no gizmo, one rather has to change the values inside of the "DeepTransform" node.
If deep data would be handled much more like objects which can be added to a scene, rather than nodes which can be visualized using pointclouds, it would get more similar to a real 3D program and workflow, as the interface operations would be more intuitive.

Additional Pointcloud Support (Save 3D Objects Rather Than Pixels)

To separate objects from one another, in the traditional workflow object IDs are used. These are passes which store a colour value for each object so that later each object could be treated separately in the compositing software. Colour values, however, come along with the same problem as the motionpass mentioned before, as the data for a 3D scene is stored inside of a 2D image.
The logical conclusion would be to store these object IDs within the Deep data, so each object can really be separated without any problems.
This would mean a huge benefit for the compositing artist.

However, if incorporated in the 3D pipeline rather than 2D, an idea is to store the data in objects, not in colours.
If any pixel would have an additional value specifying which object it belongs to inside the 3d package and 3d scene, there would not be any need for object IDs, a system which was invented for 2D images.
If Deep data would be treated like a 3D scene, and each object would be treatable as an object (from the users point of view, of course, Deep data still exists as pixels), this would also mean that most render layers would be neglectable, saving up working hours and providing more flexibility to the overall workflow.

Usage In Combination With Live Action Footage

So-called Lightfield cameras essentially produce Deep data for real footage, which could revolutionize the film industry in the future, as technologies like green screens would then be redundant.
Currently, Lightfield cameras are still under development. A Lightfield camera system uses a set up of different cameras which capture the scene from different angles. This way, depth of field and camera angles can be adjusted later on.[21]
Merging Deep rendered imagery and Lightfield data could mean a smooth and seamless workflow between digital and live action footage, provided that the camera movement of the live action footage is tracked or recorded, so both plates could exist in the same 3D space.

[21] Dr. Keinert, Joachim, “Lichtfeld”, Fraunhofer ILS, Retrieved January 12th, 2018.

3D Stereo And Virtual Reality

To produce a film in stereoscopic 3D, two images have to be rendered. One for the left, and one for the right eye. With two cameras, this way enough parallax can be created to have the brain see three dimensional. That, of course, means double the render time, double the post production.
What if it was possible to shift the stereoscopic process to compositing?
Since Deep data can display the scene in 3D, it would be possible to have two cameras inside of Nuke to render the Deep image from two different angles.
Of course, as mentioned above, this would mean to have a certain overlap to be rendered in, to prevent black areas which have no pixel data as they have not been rendered.
Another problem would exist when it comes to reflections and refractions, as they are camera-dependant and would vary on the camera's position.
Having two cameras in Nuke with reflections and refractions basically baked in would mean that it would look fake and flat to the eye. There would have to be another solution, where only the reflections are rendered separately in the 3D package, or even real-time, and then be layered on top of the existing two plates.
For applications in the field of virtual reality, this could also come in handy. As for movies shown in 360 degrees, the viewer using a VR headset has the option to see the film in 360, but there is not the possibility to have a parallax effect since the perspective is rendered in the video file.
With Deep data, this could be fixed to some degree, providing enough computing power to have animated point clouds or Deep pixels displayed in real-time.

Conclusion

Deep Compositing and Deep image data offer to work in a more modern approach, leaving methods like hold-out mattes and masks behind, as the 3D workflow and data of the rendered 3D scene is kept intact. One has the option to work in 3D compositing space, which solves a lot of common problems like z-depth and anti-aliasing,.
Therefore, it is easy for the compositing artist to change or add assets later on, as the node based, procedural system of Nuke can be fully used to work non destructively on all layers. Assets can be easily exchanged new geometry added and treated separately in 3D space, which is the same as in the 3D scene. This way, compositing and Look Development can start earlier in the production, without the need to wait for all finished assets or characters. This workflow allows for a much smoother transition between rendering and compositing and relaxes the pipeline.
Also, compositing is not as rushed as it would be in most productions with a tight deadline.

Previously, there were some downsides to this method. The storage needed to save a Deep image file is substantially larger, which not only means more capable storage but also slows down the workflow, as one has to handle and work with bigger files.
Since in small teams, most of the work would be done locally and SSDs have been becoming cheaper, this bottleneck, however, is about to resolve itself.

References

1. ”Finding freedom through deep compositing techniques”, Foundry, https://www.foundry.com/industries/film-television/freedom-with-deep-compositing, Retrieved December 07, 2018.

2. Heinen, Patrick “Deep image compositing in a modern visual effects pipeline”, http://patrickheinen.com/docs/DeepCompositingInVFX_PatrickHeinen_2013.pdf, Stuttgart Media University, Berlin, March 18th 2013, Retrieved January 14th, 2018.

3. Interview with Sebastian Metz, “WeCanDance”, November 2018

4. Seymour, Mike “The art of deep compositing”, fxGuide, https://www.fxguide.com/featured/the-art-of-deep-compositing/, February 27, 2014, Retrieved January 12th, 2018.

5. Pellacini, Fabio “the 3D production pipeline”, 2009, Pellacini, Fabio “the 3D production pipeline”, http://pellacini.di.uniroma1.it/teaching/projects10/lectures/01_pipeline.pdf, 2009, Retrieved January 12th, 2018.

6. Ostasheva, Alexandra “DIGITAL COMPOSITING IN THE VFX PIPELINE”, May 2012, https://www.theseus.fi/bitstream/handle/10024/94432/DIGITAL%20COMPOSITING%20IN%20THE%20VFX%20PIPELINE.pdf?sequence=2, Retrieved January 14th, 2018.

7. Failes, Ian, “Vampire Hunter: two killer sequences”, 1. July 2012,https://www.fxguide.com/featured/vampire-hunter-two-killer-sequences/, Retrieved January 14th, 2018.

8. Nuke Documentation, https://learn.foundry.com/nuke/content/learn_nuke.html, Retrieved January 12th, 2018.

9. Dr. Keinert, Joachim, “Lichtfeld”, Fraunhofer ILS, Retrieved January 12th, 2018, https://www.iis.fraunhofer.de/en/ff/amm/for/forschbewegtbildtechn/lichtfeld.html, Retrieved January 12th, 2018.

10. Romeo, Marco, “Nuke Screen Space Ambient Occlusion for Deep Compositing, ” https://vimeo.com/94854706, Retrieved January 12th, 2018.

11. Kerekes, Zsolt,”Charting the Rise of the SSD Market”, StorageSearch.com, http://www.storagesearch.com/chartingtheriseofssds.html, Retrieved January 12th, 2018.

12. Andre, Pascal, April 29. 2016, https://render.otoy.com/forum/viewtopic.php?f=27&t=53775, Retrieved January 12th, 2018.