2019
4
ISSN 1433-2620 > B 43362 >> 23. Jahrgang >>> www.digitalproduction.com
Publiziert von DETAIL Business Information GmbH

Deutschland € 17,90
Österreich € 19,–
Schweiz sfr 23,–

DIGITAL PRODUCTION

DIGITAL PRODUCTION

MAGAZIN FÜR DIGITALE MEDIENPRODUKTION

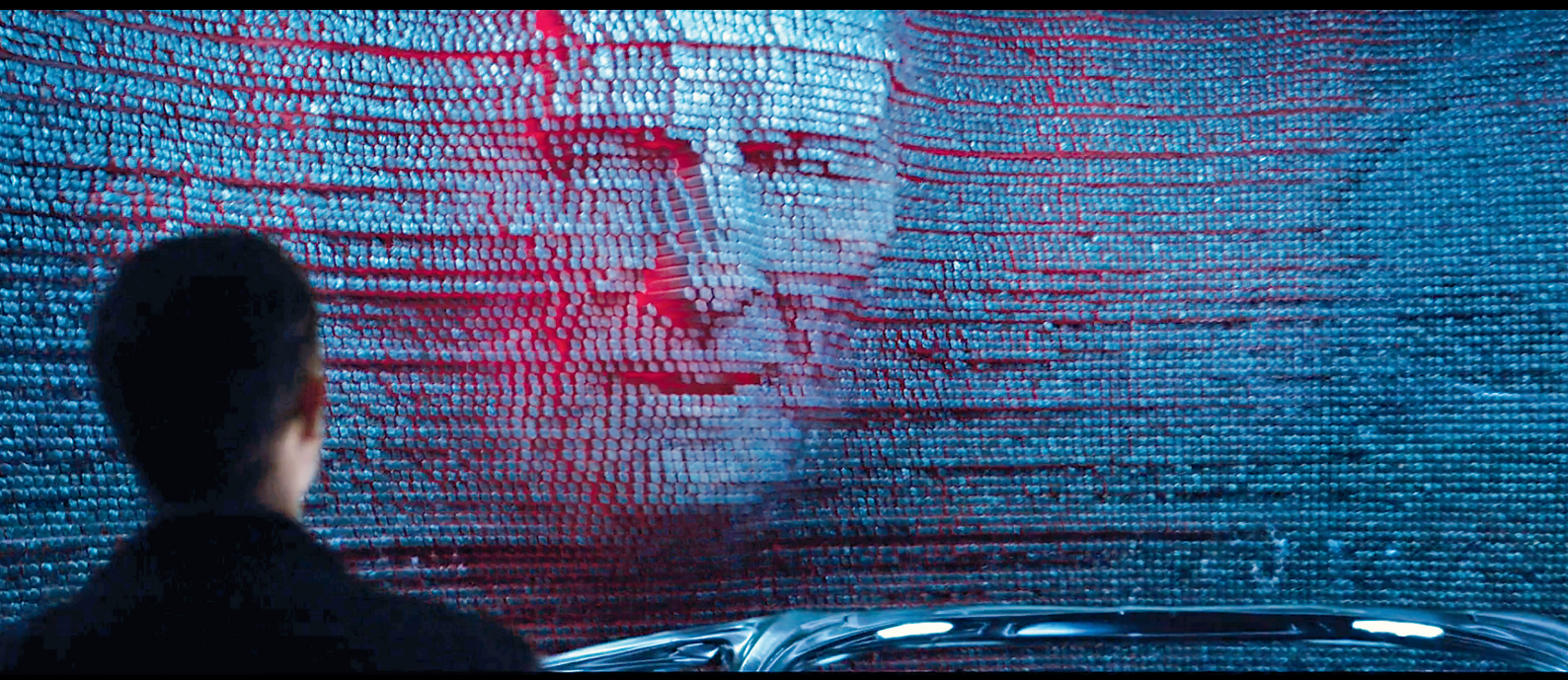
JULI | AUGUST 04:2019

Hardware
Keys, Screens, Storage und mehr im Fokus

Praxis
ZBrush, X-Particles, PBR, OSL, Tyflow, Flame & Resolve

und vieles mehr!
Renderman, Open Timeline IO, Weta, Open Movies ...

4 194336 217907 04



Das Shading der Pinwand im Power Rangers-Film stellt sich als schwierig heraus. Die Lösung: Die Geometrien der einzelnen Nagelköpfe wurden per OSL individuell angepasst, um dadurch die Glanzlichter kontrollieren zu können.

Mehr als nur Shading –
Was mit OSL alles möglich ist

Fast jede große Render Engine, die auf Path Tracing basiert – dazu gehören z.B. Arnold und Cycles –, unterstützt inzwischen auch die Open Shading Language (OSL – Artikel in DP 04:18) für die Definition von Shadern und prozeduralen Materialien. Bei Image Engine wird OSL aber für deutlich mehr als den ursprünglichen Einsatzzweck verwendet, z.B. für Compositing, die Generierung von Volumen via OpenVBD oder zur prozeduralen Modellierung und Animation.

von Gottfried Hofmann

Image Engine arbeitet dabei nicht im stillen Kämmerlein. Vielmehr kann man die so entstandenen Werkzeuge direkt im Look-Dev-Progamm Gaffer ausprobieren. Die Software wurde in einigen großen Hollywood-Produktionen wie „Thor: Ragnarok" oder „Fantastische Tierwesen: Grindelwalds Verbrechen" eingesetzt, ist Open Source und kann für Linux sowie Mac OS heruntergeladen werden.

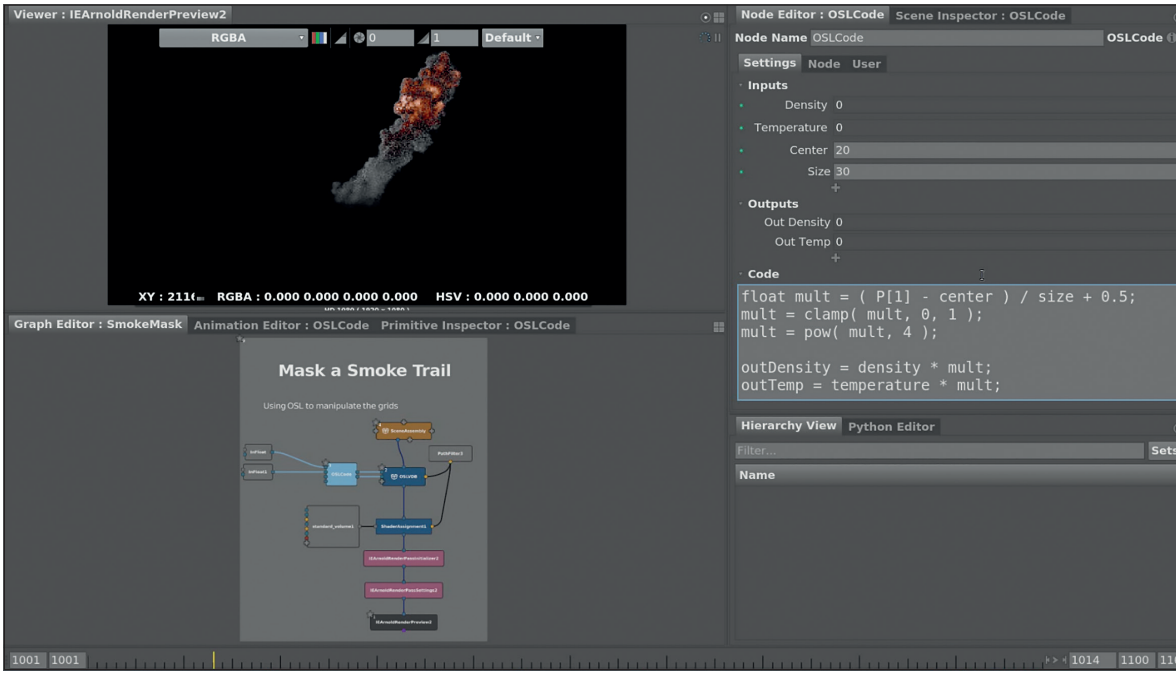
Nicht noch eine Sprache

Aber wie kam Image Engine auf die Idee, OSL für so viele Anwendungsbereiche einzusetzen, wenn es doch im Kern eine Shader-Sprache ist?

Mit der gleichen Idee haben auch die Entwickler von Blender bereits gespielt, und das aus dem gleichen Hauptgrund: nicht zu viele Sprachen ins Paket lassen. Bei Blender ist der Gedanke, OSL für Compositing einzusetzen, da es bereits in Cycles unterstützt wird und Blender-Nutzer keine weitere Programmiersprache erlernen müssen. Bisher wurde der Plan noch nicht umgesetzt, bei Image Engine hingegen ist man schon viel weiter gegangen.

C++, Python, OSL

Im Kern ist Gaffer in C++ geschrieben, für das Interface und als allgemeine Skriptsprache wird Python verwendet. Für das Shading wird zudem OSL unterstützt. Nun war aber die Frage, welche Sprache man für Dinge wie prozedurale Modellierung und Animation sowie Compositing verwenden sollte. Für Ersteres hätte sich z.B. SeExpr von Disney angeboten, für Letzteres OpenFX. Beides hat in der VFX-Branche eine gewisse Verbreitung. Aber es wären auch gleich zwei neue Sprachen hinzugekommen, die Team-Mitglieder hätten erlernen müssten. Und es hätte die Kommunikation über Team-Grenzen hinweg schwieriger gemacht, da eben nicht jeder alle diese Sprachen beherrscht.



OpenVDB durch OSL – In diesem Beispiel wird OSL genutzt, um einen Teil einer Rauchsimulation wegzumaskieren. Auch hier finden die Operationen wieder direkt auf den Daten und nicht erst beim Shading bzw. Rendern statt.

Diverse Fliegen mit einer Klappe

OSL hingegen war für all diese Einsatzgebiete prinzipiell ebenfalls geeignet, es mussten nur geringe Anpassungen vorgenommen werden, die von Image Engine übrigens Upstream in das OSL-Projekt eingepflegt wurden. Sprich auch in anderen Programmen kann jetzt der Einsatz von OSL ausgeweitet werden, das Fundament dafür wurde schon errichtet.

Dank OSL schlägt man bei Image Engine mehrere Fliegen mit einer Klappe, und Mitarbeiter fast aller Departments können ihre kleinen Helferlein nun mit der gleichen Sprache realisieren.

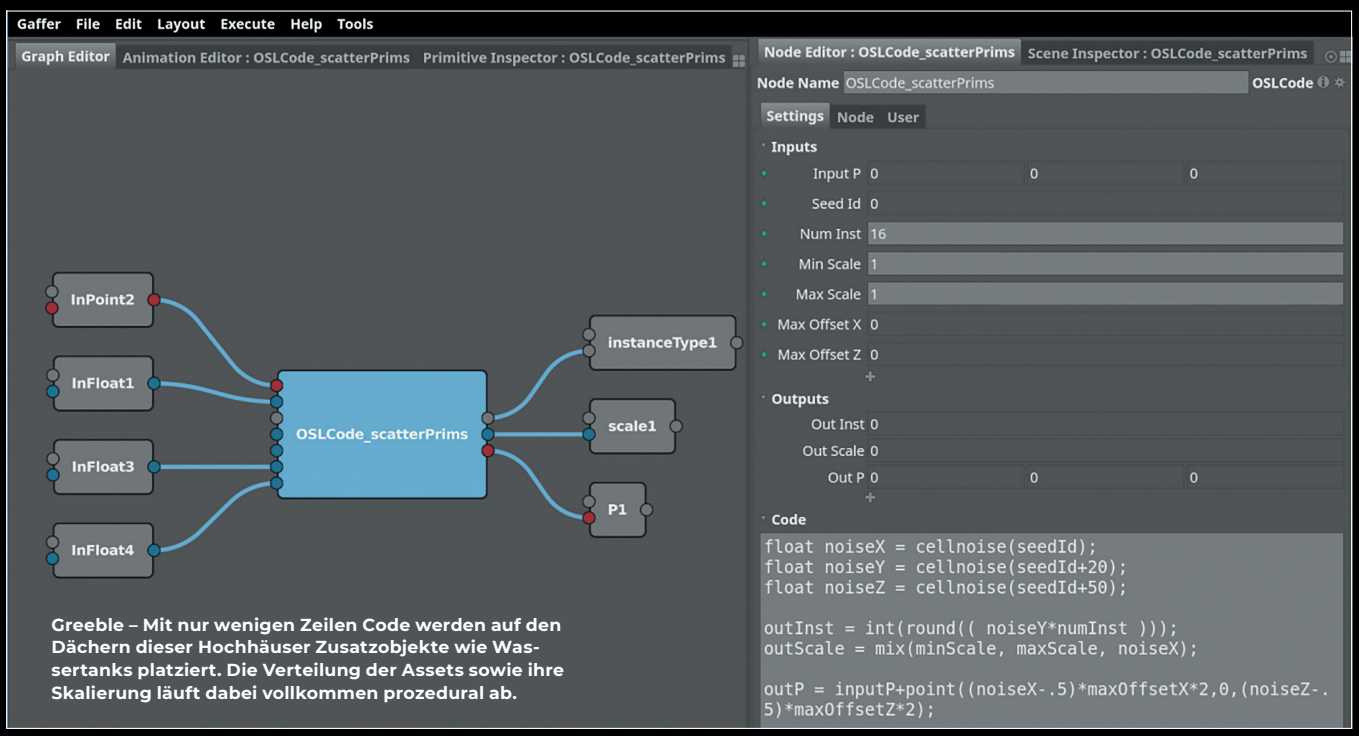
Dabei kam es durchaus schon vor, dass Features, die auf der To-do-Liste von Gaffer standen und eigentlich low-level über C++ implementiert werden sollten, von Mitarbeitern via OSL erstellt wurden.

Ein Beispiel dafür ist Camera Frustum Culling, das Objekte, die sich nicht im Sichtfeld der Kamera befinden, automatisch ausblendet. Realisiert wurde es mittels 12 Zeilen OSL-Code während einer Produktion und arbeitet dabei ausreichend performant.

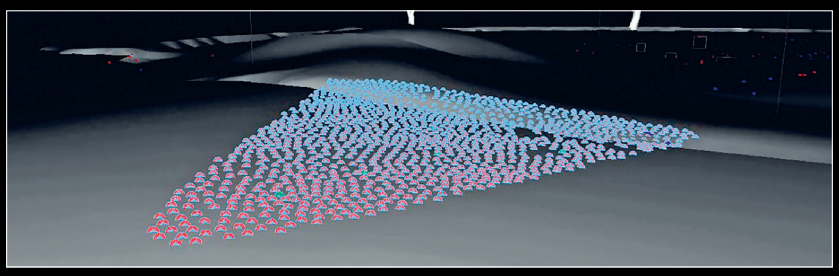
Kein Ersatz für Python?

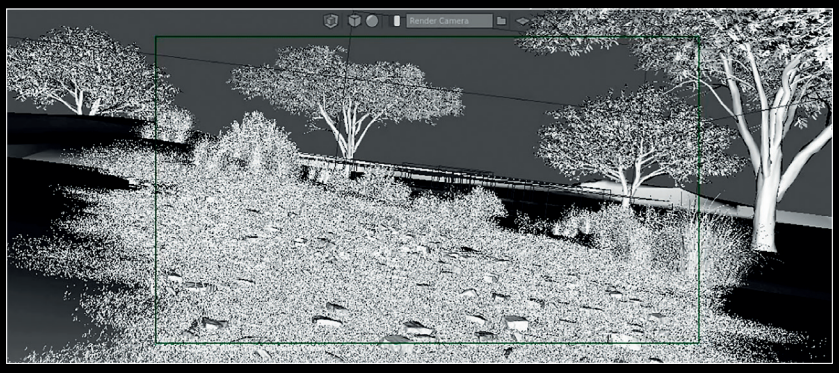
Lediglich im Bereich Skripting wird OSL bei Image Engine noch nicht wie erwartet angenommen, und das obwohl die Performance deutlich besser ist.

Python hat mit seiner großen Auswahl an Paketen und Modulen z.B. für den Zugriff auf Dateien und Datenbanken allerdings auch einige handfeste Vorteile, die in OSL wohl nicht so schnell Einzug halten werden.



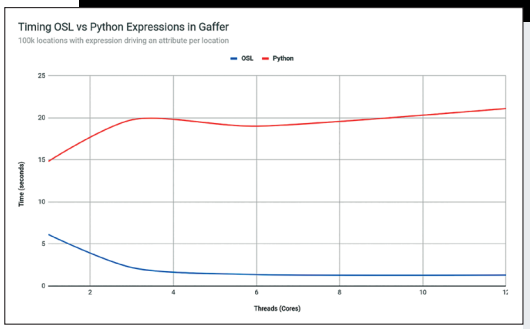
Greeble – Mit nur wenigen Zeilen Code werden auf den Dächern dieser Hochhäuser Zusatzobjekte wie Wassertanks platziert. Die Verteilung der Assets sowie ihre Skalierung läuft dabei vollkommen prozedural ab.





So sieht das Resultat der Frustum-Culling-Funktion in Gaffer aus. Da OSL in Gaffer nicht erst zur Shading-Zeit eingesetzt wird, können die Daten nicht nur gut visualisiert, sondern auch gleich weiterverarbeitet werden. In diesem Beispiel wurden nur in der sichtbaren Region Gräser und Steine verteilt.

In der finalen Animation merkt man nichts davon, dass das Rendering auf das Sichtfeld beschränkt wurde.



OSL zeigt gegenüber Python deutliche Performance-Vorteile und skaliert besser bei Multithreading.

Fazit

Image Engine hat gezeigt, dass OSL für mehr als nur Shading eingesetzt werden kann. Die Implementierung in Gaffer ist Open Source und damit für jedermann nutz- und einsehbar. Vielleicht folgen ja weitere Studios und Programme diesem Beispiel, was OSL einen zweiten Frühling bescheren könnte. Man munkelt zudem, dass Gaffer bald auf Windows portiert werden wird. **› ei**

Interview

Interview with Andrew Kaufman from Image Engine on the usage of OSL for non-shading tasks in the open-source lookdev-tool Gaffer.

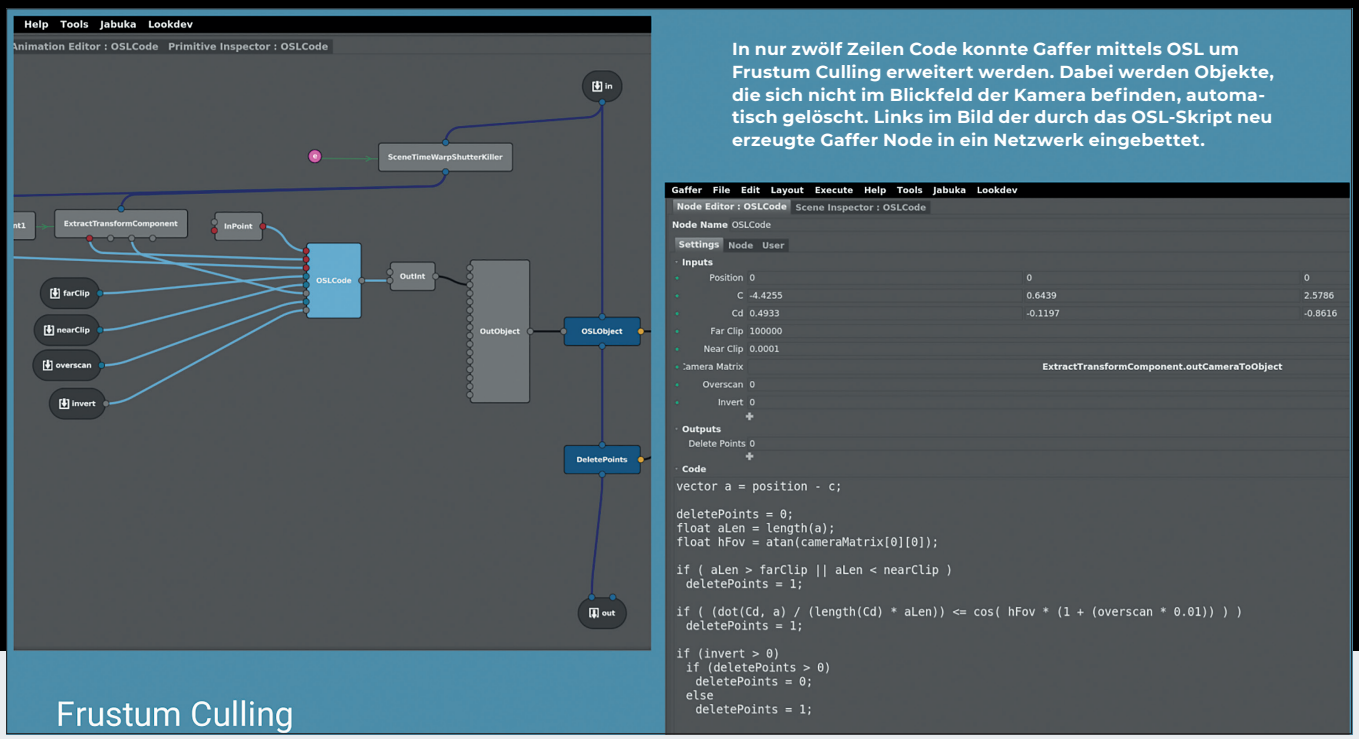
DP: We are here at FMX 2019 with Andrew Kaufman from Image Engine. Andrew, what was the rationale for choosing OSL for things like compositing, procedural modeling, Python scripting, and generating volumes as OpenVDB grids and what were your experiences?
Andrew Kaufman: One of the core philosophies in Gaffer is that we want to keep the number of languages minimal. We use C++ for all our core computation for performance reasons. Python is used for all the glue code and a lot of the UI. We needed a language that sits between the two, so we tried to see how far we can push OSL. We were already using it for shading and wanted to see whether or not we could use it for some live data manipulation in Gaffer, rather than introducing another language. One of the things we found out after that experiment proved successful, is that there are really good benefits to cross-training our employees in this language. We end up with people in all departments knowing OSL for one reason or another. Maybe you are in FX and you are learning OSL because you want to use it in your instancing setup or maybe you are in lookdev and you know OSL very well because you are writing shaders all the time. Maybe you are in a department that is in between and you need a little bit of OSL for your QC render template. As a result, we end up with all these artists who know this language, which is useful for fast prototyping and actually seems fast enough for final results in most cases.

DP: What were your experiences with the users, how did they pick it up?
Andrew Kaufman: A lot of people were able to adopt it fairly quickly. For example, people who come from an Arnold or Renderman background would be familiar enough with OSL to be able to pick it up easily. People with a background in Houdini are used to VEX, so it is a kind of natural transition. You have to learn the new language in that case, but it is similar enough that you don't have to spend a lot of time doing so.

DP: Did you find any boundary cases where OSL does not cut it?
Andrew Kaufman: So far, we have not. If we did discover boundaries where OSL was not working, we optimized how we were using OSL inside Gaffer. There is a lot of C++ you have to write in order to use OSL in the first place. So we optimized that code path. In the end, we always have the fall-back option to write a pure C++ node if we need to.

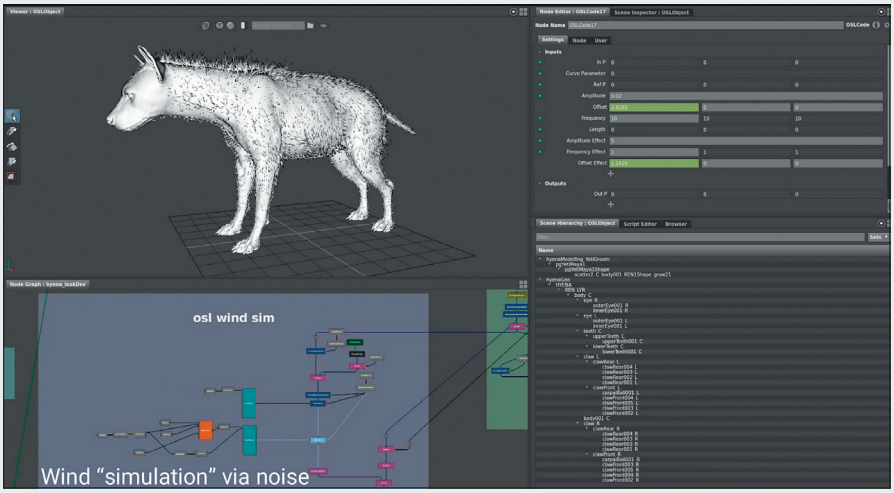
DP: OSL is not ported to work on GPU yet. That is probably not a problem at Image Engine because you are not using GPUs that much.
Andrew Kaufman: The master branch of OSL is getting there in terms of GPU support. It would be really cool to see what kind of performance gains we can get from that, but for now it has not felt necessary.

DP: How did OSL come along as a Python replacement?
Andrew Kaufmann: It works very well as a Python replacement in terms of performance but we really have not had much success incorporating it in production. Most artists still prefer Python over OSL. It lacks complex data structures and Python has a lot of useful modules that are not available in OSL.



In nur zwölf Zeilen Code konnte Gaffer mittels OSL um Frustum Culling erweitert werden. Dabei werden Objekte, die sich nicht im Blickfeld der Kamera befinden, automatisch gelöscht. Links im Bild der durch das OSL-Skript neu erzeugte Gaffer Node in ein Netzwerk eingebettet.

Frustum Culling



In Gaffer können über OSL sogar Haare animiert werden. Das Fell dieser Hyänen wurde mittels einer Noise-Funktion und einigen weiteren Anpassungen leicht in Bewegung versetzt, es musste keine dedizierte Windsimulation durchgeführt werden.

For example, if you want to interface with your asset management database or read a JSON file, you cannot do that in OSL. That is enough of a hindrance to prevent people from switching. We also do not have enough documentation yet on why it is beneficial.

DP: **Using OSL to deform geometry is an obvious use case, but there probably is a lot of work to do to support it in a DCC. How difficult was it for you to plug it into all those parts of the pipeline?**

Andrew Kaufman: I do not have actual stats on how much development time went into it. I know we did have to make a couple of pull requests back to OSL in order to get it running fast enough in those scenarios. We needed some extra queries to be able to short-circuit things. The code is on Github. If you take a look into the GafferOSL module, you can walk yourself through it. It certainly is complex if you are coming to it for the first time but it shouldn’t be overwhelmingly difficult for anyone who is writing a DCC.

DP: **You did all the groundwork, so it is easier for those who want to follow up.**

Andrew Kaufman: That is one of the advantages of open-source software, you end up with proper reference implementation that actually works, not just some white paper that might be missing crucial details.

DP: **Someone is writing a Cycles integration into Gaffer.**

Andrew Kaufman: The Gaffer-Cycles project is a good example of extending Gaffer on your own. We have an extension system, not a plug-in because there is no plug-in API you have to map to. Just write your libraries and get them on the right path. The extension system helps you do that. The Gaffer-Cycles project will presumably be considered stable at some point. The developer set it up as an extension so that you can download it separately from Gaffer, set a single environment variable and be good to go. Someone is working on porting Gaffer to Windows as well. That project can’t be done as an extension, because it requires core changes, but I know a lot of people are looking forward to using Gaffer on Windows. So hopefully that effort will be incorporated back into the official releases later this year.

Gottfried Hofmann ist Diplom-Informatiker und bietet seit mehreren Jahren professionellen Support sowie Schulungen für die freie 3D-Software Blender an. Als freischaffender Autor schreibt er für Fach- und Computerzeitschriften. Er hat zahlreiche Blender-Tutorials verfasst, u.a. für CG Tuts+ und CG Cookie. Weiterhin betreibt er die Website www.BlenderDiplom.com, auf der Blender-Tutorials in deutscher und englischer Sprache zur Verfügung stehen und Schulungen gebucht werden können, und hilft bei der Organisation von BlenderDay und Blender Summer School in Mannheim.