Claudius Dokumentation

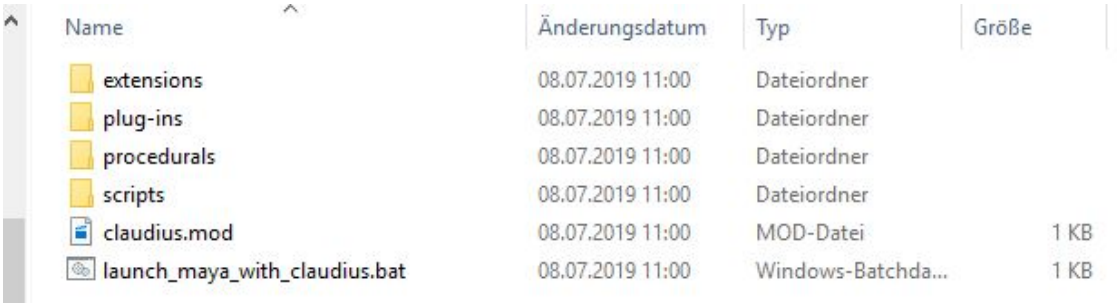
Jan Honsbrok
Bachelor Medieninformatik, 734992
Technical School of Film and Animation 3, SoSe 2019

Was ist Claudius?

Claudius ist ein C++ Plugin, um statische Punktwolken, wie sie von 3D Laserscannern produziert werden, in Maya importieren, im Maya Viewport darstellen und mit Arnold rendern zu können. Claudius wurde für Windows und Maya 2018 entwickelt.
Claudius stellt einen Maya FileTranslator zur Verfügung, womit Punktwolken in gewohnter Art und Weise über den Menüeintrag “File -> Import” in Maya importiert werden können. Derzeit werden ausschließlich PTS Dateien unterstützt. In Maya wird eine Punktwolke durch einen ClaudiusVisualizer Node repräsentiert. Dieser Node sorgt für die Darstellung in Viewport und der User kann mit ihm in gewohnter Weise wie mit jedem anderen Maya Node interagieren.
Um Punktwolken mit Arnold rendern zu können, wurde ein Arnold Procedural implementiert. Dieses Procedural liest zur Renderzeit die Punktwolke ein und übersetzt sie in ein Arnold Partikelsystem. Damit der User wie gewohnt in Maya einfach auf den “Render” Button klicken kann, wurde ein Plugin für MtoA, das Arnold Plugin für Maya, erstellt. Dieses Plugin sorgt beim Übersetzen der Maya Szene für Arnold dafür, dass ein ClaudiusVisualizer Node in ein entsprechendes Arnold Procedural übersetzt wird.

Claudius ausführen

Claudius wird als ZIP-Archiv ausgeliefert. Es genügt, dass ZIP-Archiv zu entpacken und die launch_maya_with_claudius.bat auszuführen.



Erwartetes Verhalten von Claudius

- Das Plugin lässt sich fehlerfrei laden. Siehe Fehlermeldung Specified procedure not found
- Über File -> Import lässt sich eine PTS Datei importieren

- Die importierte Punktwolke wird im Viewport angezeigt. Standardmäßig hat die Punktwolke keine Farbe
- Ändert man das Particle File Attribut auf eine andere Datei, wird die neue Punktwolke geladen
- “Render with Color” kann aktiviert werden und die Punktwolke wird mit Farbe im Viewport dargestellt und gerendert
- “Render With Every Nth” kann auf einen höheren Wert gestellt werden, die im Viewport dargestellte Punktzahl verringert sich. Im Arnold Rendering werden nach wie vor alle Punkte gerendert

Super impatient building guide

```
set MTOA_LOCATION=<yout_mtoa_location>
set ARNOLD_HOME=<yout_arnold_location>
mkdir build
cd build
call "C:\Program Files (x86)\Microsoft Visual Studio
14.0\VC\vcvarsall.bat" x64
cmake -G "Visual Studio 14 2015 Win64" -DMAYA_VERSION=2018 ..
cmake --build . --config Release
```

Voraussetzungen

Cmake

Als Build System wird CMake verwendet. Siehe CMake. Cmake kann hier heruntergeladen werden: https://cmake.org/download/. Es wird mindestens Version 3.1 benötigt. CMake muss im PATH verfügbar sein. Test:

```
cmake --version
```

gibt etwas ähnliches wie

```
cmake version 3.14.3

CMake suite maintained and supported by Kitware (kitware.com/cmake).
```

aus

Visual Studio für C++

Da Claudius in C++ implementiert ist, wird ein C++ Compiler benötigt. Dabei muss darauf geachtet werden, dass richtige Compilerversion verwendet wird. Welche Visual Studio Version zu verwenden ist, hängt von der Maya Version ab:

Maya Version	**Visual Studio Version**

2019	Visual Studio 2015 Update 3
2018	Visual Studio 2015 Update 3
2017	Visual Studio 2012 Update 4

Welche Visual Studio Version zu verwenden ist, erfährt man andernfalls in der Dokumentation von Maya:
https://help.autodesk.com/view/MAYAUL/2016/ENU/?guid=__files_Setting_up_your_build_env_Windows_env_32bit_and_64bit_htm
In der Maya Dokumentation benötigt man Zeit, bis man findet, was man sucht. Es lohnt sich sicherlich, die Seite https://around-the-corner.typepad.com/ zu besuchen. Dort werden nach jedem neuen Maya Release zeitnah die Compiler Versionen für alle Plattformen aufgelistet.

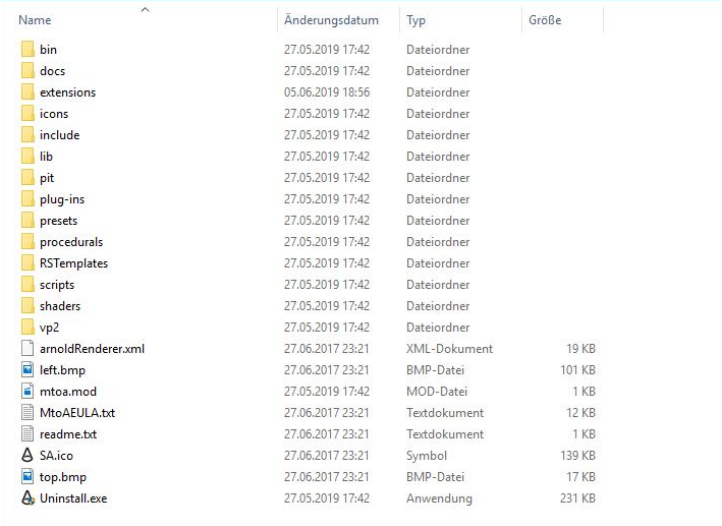
Maya Devkit

Da Claudius über ein Maya Plugin verfügt, wird auch das Maya Devkit benötigt. Das Devkit wird nicht mit Maya mitgeliefert und muss separat heruntergeladen werden. Derzeit sind die Devkit Versionen unter folgendem Link zu finden:
https://www.autodesk.com/developer-network/platform-technologies/maya. Dies kann sich für spätere Maya Versionen jedoch ändern. Es wird das Devkit für die jeweilige Maya Version benötigt, für die kompiliert werden soll. Das Devkit wird als ZIP-Archiv ausgeliefert. Aus dem entpackten ZIP-Archiv müssen die Ordner cmake, include und lib nach C:\Program Files\Autodesk\Maya<Maya Versionsnummer> kopiert werden

MtoA

MtoA steht für “Maya to Arnold” und ist der interne Name des Maya Plugins für Arnold. MtoA wird bei Maya Versionen ab 2017 standardmäßig mitgeliefert und muss bei der Installation lediglich aktiviert werden. Damit MtoA während des Builds gefunden werden kann, muss die Umgebungsvariable MTOA_LOCATION gesetzt werden.

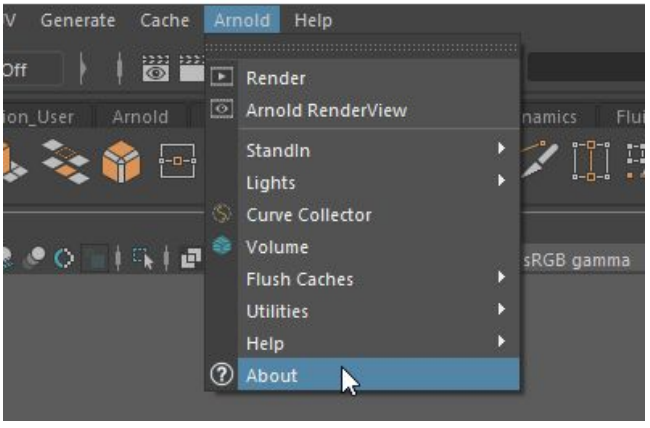
So sollte der Ordner aussehen, auf den MTOA_LOCATION gesetzt wird :

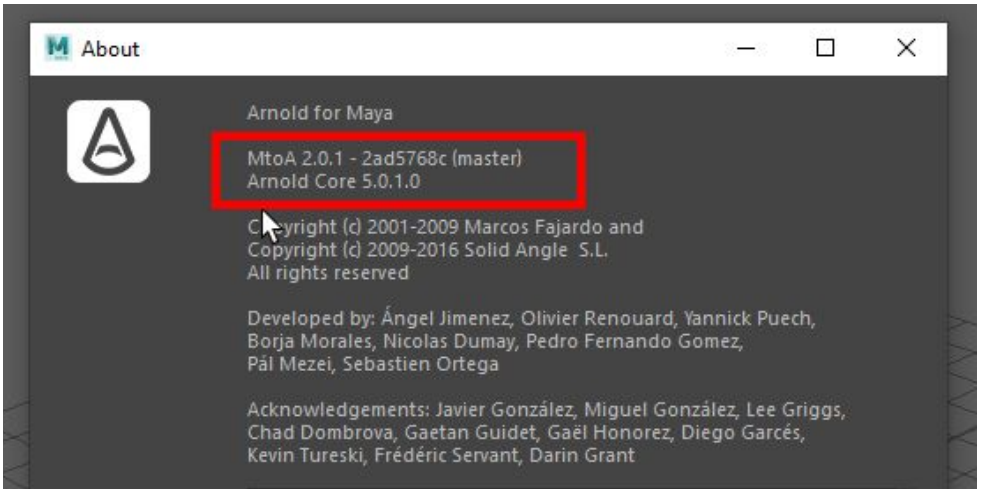


Name	Änderungsdatum	Typ	Größe
bin	27.05.2019 17:42	Dateiordner	
docs	27.05.2019 17:42	Dateiordner	
extensions	05.06.2019 18:56	Dateiordner	
icons	27.05.2019 17:42	Dateiordner	
include	27.05.2019 17:42	Dateiordner	
lib	27.05.2019 17:42	Dateiordner	
pit	27.05.2019 17:42	Dateiordner	
plug-ins	27.05.2019 17:42	Dateiordner	
presets	27.05.2019 17:42	Dateiordner	
procedurals	27.05.2019 17:42	Dateiordner	
RSTemplates	27.05.2019 17:42	Dateiordner	
scripts	27.05.2019 17:42	Dateiordner	
shaders	27.05.2019 17:42	Dateiordner	
vp2	27.05.2019 17:42	Dateiordner	
arnoldRenderer.xml	27.06.2017 23:21	XML-Dokument	19 KB
left.bmp	27.06.2017 23:21	BMP-Datei	101 KB
mtoa.mod	27.05.2019 17:42	MOD-Datei	1 KB
MtoAEULA.txt	27.06.2017 23:21	Textdokument	12 KB
readme.txt	27.06.2017 23:21	Textdokument	1 KB
SA.ico	27.06.2017 23:21	Symbol	139 KB
top.bmp	27.06.2017 23:21	BMP-Datei	17 KB
Uninstall.exe	27.05.2019 17:42	Anwendung	231 KB

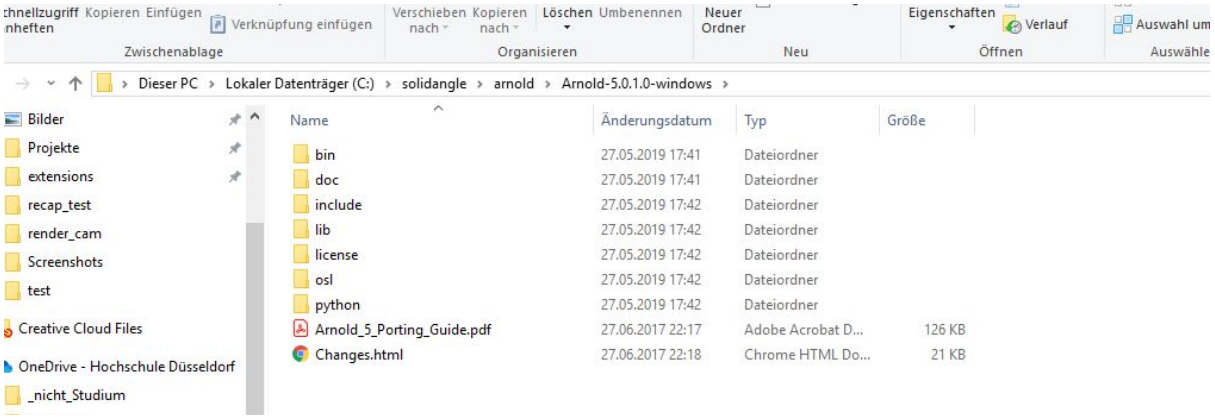
Arnold SDK

Claudius enthält ein Procedural für den Arnold Renderer, weshalb das Arnold SDK benötigt wird. Das Arnold SDK kann hier heruntergeladen werden: https://www.arnoldrenderer.com/arnold/download/. Zum Download wird ein Autodesk Account benötigt. Die Arnold Version muss mit der von MtoA verwendeten Arnold Version übereinstimmen. Die von MtoA verwendete Arnold Version kann hier eingesehen werden:





Damit das Arnold SDK während des Builds gefunden werden kann, muss die Umgebungsvariable ARNOLD_HOME gesetzt werden. So sollte der Ordner aussehen, auf den ARNOLD_HOME gesetzt wird:



7-Zip

Am Ende des Build Prozesses wird automatisch ein ZIP-Archiv generiert, um die Verteilung des Plugins zu erleichtern. Zur Erstellung des ZIP-Archivs wird 7-Zip benötigt. 7-Zip ist hier als Download erhätlich: https://www.7-zip.de/ . Es wird erwartet, dass 7-Zip in den Standard Installationspfad nach C:/Program Files/7-Zip installiert wird.

Python (optional)

Python wird nicht benötigt, um Claudius zu kompilieren. Das Release-Helfer-Skript release.py sowie bumpversion sind jedoch in Python geschrieben. Um mit der VFX Reference Platform 2019 kompatibel zu sein, wird Python 2.7 verwendet. 2020 endet der Support für Python 2.7 und sollte danach nicht mehr verwendet werden!

Kompilieren

Eine neue Shell wird im Claudius Projektverzeichnis gestartet.
Um Claudius zu kompilieren, sind folgende Schritte erforderlich:

1. Umgebungsvariablen aufsetzen (ggf. optional)
2. Visual Studio korrekt konfigurieren
3. Cmake aufrufen
4. Kompilieren
5. Testen / ausführen

Umgebungsvariablen aufsetzen (ggf. optional)

Die Umgebungsvariablen können entweder in der Shell Session oder auf Betriebssystemebene definiert werden. Sollte Claudius häufig kompiliert werden, so ist es ratsam, die Umgebungsvariablen über das Betriebssystem zu setzen. Andernfalls können die Umgebungsvariablen wie folgt gesetzt werden:

```
set MTOA_LOCATION=<yout_mtoa_location>
set ARNOLD_HOME=<yout_arnold_location>
```

Visual Studio korrekt konfigurieren

Claudius wird als 64Bit Library kompiliert. Damit Visual Studio weiß, dass in 64 Bit gearbeitet werden soll, muss eine Batch Datei aufgerufen werden, die Visual Studio korrekt konfiguriert:

```
call "C:\Program Files (x86)\Microsoft Visual Studio
14.0\VC\vcvarsall.bat" x64
```

Cmake aufrufen

Cmake ist ein Cross Platform Build System Generator. Cmake kann basierend auf einer Cmake Projektdefinition native Dateien für eine Vielzahl von Build Systemen generieren, wie z.B. GNU Makefiles, Ninja o.ä.. Dazu wird ein sogenannter Generator verwendet. Für Claudius wird der Visual Studio Generator verwendet, der ein Visual Studio Projekt generiert. Um Dateien, die während des Build Prozesses erstellt werden, von den eigentlichen Projektdateien zu trennen, wird ein sogenannter Out-of-Source Build generiert. Dazu wird ein neuer Ordner namens build erstellt und cmake in diesem Ordner aufgerufen. Cmake wird nun alle Dateien im Ordner build ablegen.
Cmake werden folgende Informationen übergeben:

```
cmake -G "Visual Studio 14 2015 Win64" -DMAYA_VERSION=2018 ..
```

Mit -G "Visual Studio 14 2015 Win64" wird Cmake mitgeteilt, dass der Visual Studio Generator für Version 2015 verwendet werden soll.
Mit -DMAYA_VERSION=2018 teilen wir CMake mit, dass für Maya 2018 kompiliert werden soll. Wird für eine andere Maya Version kompiliert, so ist die Versionsnummer entsprechend anzupassen.
Mit .. teilen wir CMake mit, dass der Source Code ein Ordnerlevel höher liegt.

Kompilieren

Mit

```
cmake --build . --config Release
```

wird der Build gestartet. Es ist wichtig, dass ein Release Build erstellt wird. Debug Builds sind nur auf dem aktuellen Computer lauffähig, da sie zusätzliche Informationen zu Debugging enthalten.

Testen / Ausführen

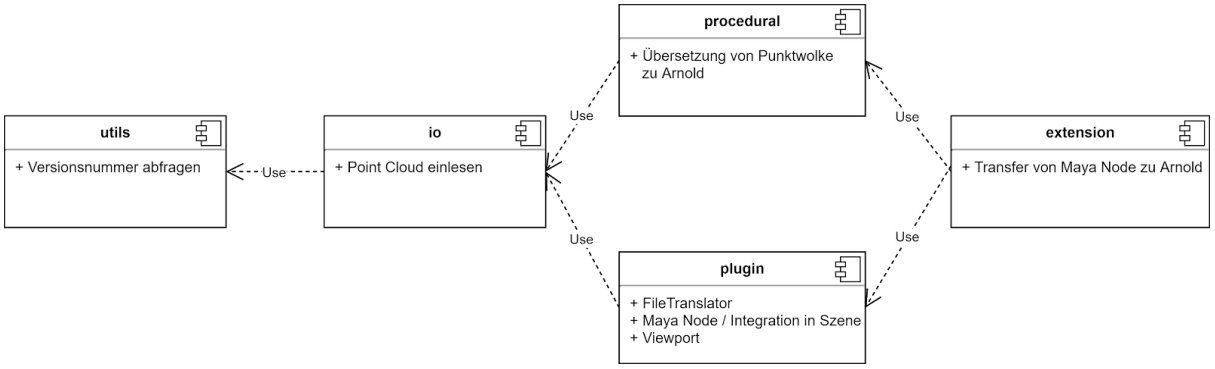
Ist der Build fehlerlos beendet worden, so befindet sich in buil/src/plugin/Release ein Ordner namens claudius-<Claudius Version>-Maya-<Maya Version>-Windows, der ein übliches Claudius Deployment enthält. Siehe auch: Plugin ausführen

Alle Schritte zusammengefasst

```
set MTOA_LOCATION=<yout_mtoa_location>
set ARNOLD_HOME=<yout_arnold_location>
call "C:\Program Files (x86)\Microsoft Visual Studio
14.0\VC\vcvarsall.bat" x64
mkdir build
cd build
cmake -G "Visual Studio 14 2015 Win64" -DMAYA_VERSION=2018 ..
cmake --build . --config Release
```

Komponenten

Claudius ist in mehrere Komponenten unterteilt:



utils

Die utils Komponente stellt Hilfsfunktionen zur Verfügung, die für mehr als eine Komponente interessant sind. Derzeit ist dies ausschließlich die aktuelle Version von Claudius. Diese ist der aktuelle Git Tag und wird von CMake zur Compile-Zeit aus Git ausgelesen.

io

Die io Komponente stellt Funktionalitäten zum Einlesen von Point Cloud Dateien zur Verfügung. Einzelne Dateiformate werden dabei in ein abstraktes Interface verpackt, sodass weitere Dateiformate einfach hinzugefügt werden können. Siehe Weitere Point Cloud Dateiformate hinzufügen. Innerhalb von Claudius werden Punktwolken durch den ParticleContainer repräsentiert, der ebenfalls in der io Komponente enthalten ist. Es keine externe Abhängigkeiten für die io Komponente.
Für die io Komponente existieren auch Tests. Allerdings sind diese Tests lediglich “Smoke Tests”, d.h. sie garantieren nicht, dass das ganze Programm richtig funktioniert, sondern lediglich, dass nichts grundsätzlich falsch ist.

plugin

Die plugin Komponente stellt Funktionalitäten in Maya zur Verfügung. Dazu gehören:

- ein FileTranslator, um Punktwolken in Maya importieren zu können. Maya stellt hierzu das abstrakte Interface MPxFileTranslator zur Verfügung, das von Claudius implementiert wird.
- ein eigener Maya Node ClaudiusVisualizer, der als MPxLocatorNode implementiert ist. Er speichert alle Daten, die der Nutzer ändern kann, wie Transformation, Pfad zur Point Cloud Datei, Farbe an/aus, Level of Detail
- ein DrawOverride für den ClaudiusVisualizer Node zur Darstellung im Viewport. Mit Einführung des Viewport 2.0 kann die alte draw() Methode eines Nodes nicht mehr zur Viewport Darstellung verwendet werden. Stattdessen implementiert und registriert man dazu das Interface MPxDrawOverride

Innerhalb der plugin.cpp werden all diese Einzelkomponenten registriert.
Die plugin Komponente hat nur eine externe Abhängigkeit zum Maya Devkit.

procedural

Die procedural Komponente ermöglicht es, eine Punktwolkendatei in Arnold rendern zu können. Dazu wurde ein Arnold Procedural implementiert. Ein Procedural nimmt verschiedene Parameter entgegen und erzeugt damit zur Renderzeit Geometrie, die gerendert werden kann. Werden Arnold Szenen als .ass Datei gespeichert, bietet dies den Vorteil, dass nicht Gigabytes an Punktwolken dupliziert werden müssen, sondern dass Procedural referenziert einfach die Punktwolke und liest sie zur Renderzeit ein. Siehe https://docs.arnoldrenderer.com/display/A5AFMUG/Custom+Procedurals. Möchte man eine höhere Performance erreichen, kann man Procedurals immer “expanded” abspeichern, d.h. das Procedural wird bereits beim Speichern der Arnold Szene evaluiert. In Fall von Claudius nimmt das Procedural einen Dateipfad entgegen, liest die im Datei ein und generiert daraus ein Arnold Partikelsystem. Zusätzlich werden die Farbinformationen der Punktwolke als Arnold User Data hinterlegt, sodass sie später vom Shader abgerufen werden können.
Die procedural Komponente hat als externe Abhängigkeit nur das Arnold SDK.

extension

Die extension Komponente ermöglicht es, eine Maya Szene, die einen ClaudiusVisualizer Node enthält, in Maya mit Knopfdruck auf “rendern” rendern zu können. Dies wird durch eine Extension für den MtoA Translator realisiert.
Wird eine Szene mit Arnold in Maya gerendert, so greift Arnold beim Rendern nicht auf Mayas Datenstrukturen zurück, sondern auf seine eigenen. Dazu muss vor dem Rendering die Mayaszene in eine Arnoldszene übersetzt werden. Dies übernimmt der MtoA Translator. Damit der Translator weiß, wie er mit einem ClaudiusVisualizer Node umgehen soll, gibt es die Erweiterung für den MtoA Translator. Beim Translation Vorgang wird für jeden ClaudiusVisualizer Node in der Szene ein Arnold Partikelsystem erstellt. Zusätzlich wird diesem Partikelsystem ein Arnold StandartSurface Shader zugewiesen. Dieser Shader wird auch richtig eingestellt, sodass er die Partikelfarbe als Arnold User Data ausliest.
Die procedural Komponente hat als externe Abhängigkeit MtoA, das Arnold SDK und Maya.

Ordnerstruktur

externals

Externe Libraries, die als Header-Only Libraries verteilt werden. Sie benötigen keine besonderen Build Instruktionen.

modules

Enthält CMake modules, um Maya, das Arnold SDK und MtoA korrekt in das Projekt zu integrieren.

src

Enthält den Source Code für die einzelnen Komponenten

test

Simple Tests für die io Komponente

Deployment als Maya Module

Claudius sollte im Projekt von Kommilitonen eingesetzt werden können. Dies beeinflusst das Deployment erheblich. Folgende Gesichtspunkte wurden bei der Wahl des Deployments berücksichtigt:

- Neue Versionen sollten schnell und einfach verteilt werden können
- Claudius soll auf den Rechnern im CG-Pool laufen. Darum kommt ein Installer nicht in Frage, da so ein Abhängigkeit zum CG-Pool Personal bestünde
- Es wird nicht nur eine einzelne .mll geben, sondern auch mehrere DLLs für Arnold / MtoA sowie ggf. mehrere Skriptdateien
- Es muss für einen Artist sehr einfach sein, Maya mit Claudius in der richtigen Konfiguration zu starten.

Maya und Arnold erlauben eine Konfiguration über Umgebungsvariablen. So können Plguins für Maya, Arnold und MtoA einfach registriert werden. Die Ordner , die die entsprechenden DLL-Dateien enthalten, müssen nur in den entsprechenden Umgebungsvariablen stehen. Es gibt eine Batch-Datei, in der diese Umgebungsvariablen gesetzt werden und Maya anschließend gestartet wird. Um das Setzen der Umgebungsvariablen zu vereinfachen, wurde ein Maya Module angelegt. Ein Maya Module besteht aus einer .mod-Datei, in der Umgebungsvariablen relativ zur .mod-Datei definiert werden können (siehe Maya Module Dokumentation). Maya muss nur über die MAYA_MODULE_PATH Umgebungsvariable mitgeteilt werden, wo die .mod-Datei liegt. Alles weitere wird automatisch von Maya übernommen.
Ein neues Claudius Release kann wie folgt deployed werden:

1. Das fertige ZIP-Archiv wird hochgeladen und der Link wird den Kursteilnehmern zur Verfügung gestellt
2. Die Kursteilnehmer laden das ZIP-Archiv herunter und entpacken es

3. Die Kursteilnehmer könne Maya mit Claudius in der korrekten Konfiguration mit nur einem Doppelklick starten

Unterstützte Point Cloud Dateiformate

Derzeit wird lediglich das PTS Dateiformat in der hier beschriebenen Ausführung unterstützt. Der Grund liegt darin, dass PTS-Dateien sehr einfach strukturierte Textdateien sind. Für die Anwendungszwecke im Projekt war dies gut genug. Claudius kann aber einfach mit weiteren Punktwolkenformaten erweitert werden (siehe Weitere Point Cloud Dateiformate hinzufügen).

Unterstütztes PTS Format

Für das PTS Dateiformat gibt es keine einheitliche Spezifikation. Claudius unterstützt **ausschließlich** PTS Dateien, auf die folgendes zutrifft:

- die erste Zeile muss die Anzahl aller Punkte enthalten
- Farben werden in RGB als drei Integer gespeichert, wobei 0,0,0 schwarz und 1,1,1 weiß bedeutet.
- die ersten drei Werte sind x, y, z Koordinaten eines Punktes in einem linkshändigen Koordinatensystem, in dem Z die Up-Achse darstellt
- neben der Position kann ein Punkt folgende weiteren Informationen enthalten: Farbe, Remission und Quality. Diese Zusatzinformationen dürfen in folgenden Reihenfolgen auftreten:
    - 1. Position und RGB
    - 2. Position, Remission und RGB
    - 3. Position, Remission, Qualität und RGB
- Claudius stellt die verwendete Kombination anhand der Anzahl der Einzelwerte fest
- alle Punkte müssen über dieselbe Anzahl von Informationen verfügen

Weitere Point Cloud Dateiformate hinzufügen

Um Support für ein weiteres Point Cloud Dateiformat zu Claudius hinzuzufügen, muss lediglich das Interface ParticleReader implementiert werden. Zusätzlich muss der neue Reader in ParticleReaderFactory::createParticleReader registriert werden.

CMake

Als Build System wird CMake verwendet. CMake ermöglicht es, ein Projekt in einzelne Komponenten zu zerlegen, die wiederum voneinander abhängen können. So können die einzelnen Komponenten möglichst wiederverwendbar gehalten werden. CMake ermöglicht es zudem, das Projekt auf allen drei großen Betriebssystemen zu kompilieren. Claudius wurde zwar nur unter Windows entwickelt und getestet, aber grundsätzlich steht einer Portierung auf andere Plattformen somit nichts im Wege.
CMake wurde ausgewählt, weil es sich sowohl generell für C++ Projekte etabliert hat und für CG Software Projekte ebenfalls das vorherrschende Build System ist. So existieren bereits viele Ressourcen, um CG Software Projekte mit CMake zu entwickeln. Beispielsweise

existieren viele CMake Pakete, um um CG Libraries & SDKs in Projekte zu integrieren, beispielsweise für Maya, das Arnold SDK und MtoA.

C++ Version

Der C++ Standard definiert verschiedene Versionen der Sprache. Um mit der VFX Reference Platform 2019 kompatibel zu sein, ist Claudius in C++ 14 implementiert.

VFX Reference Platform

Aus Performancegründen werden alle performance kritischen Anwendungen für die Computergrafik in C++ implementiert. Zudem werden die Anwendungen durch Plugins von Drittanbietern oder Animationsstudios erweitert. Um eine größtmögliche Kompatibilität zwischen den einzelnen Anwendungen zu schaffen und den Entwicklungsaufwand gering zu halten, hat man die VFX Reference Platform eingeführt. Die VFX Reference Platform definiert für jedes Jahr, welche C++ Version, welche Python Version und welche Version der wichtigsten Libraries verwendet werden soll. Die VFX Reference Platform ist unter https://vfxplatform.com/ öffentlich einsehbar

Bumpversion

Bumpversion ist ein kleines in Python geschriebenes Tool, um ein Projekt mit einfachen Befehlen basierend auf Semantic Versioning hochzuversionieren. Um ein beispielsweise ein Major Versionsupgrade durchzuführen, genügt es

```
bumpversion major
```

auf der Konsole auszuführen. Bumpversion erstellt automatisch einen neuen Git Commit mit einer passenden Commit Message und erzeugt automatisch einen neuen Git Tag. Bumpversion ist nicht zwingend erforderlich, aber sehr praktisch. Bumpversion kann einfach über de Python Paket Manager pip installiert werden:

```
pip install bumpversion
```

Es ist keine spezielle Version von Bumpversion erforderlich

Release Script

release.py ist ein kleines Helferskript, dass überprüft, ob der aktuelle Git Commit ein Tag besitzt. Falls ja, kopiert es das erzeugte ZIP-Archiv in einen Ordner.

Häufige Fehler

Specified procedure not found

Maya Plugins müssen für jede Maya Version gesondert kompiliert werden. Dieser Fehler tritt auf, wenn ein Plugin geladen, welches für eine andere Maya Version entwickelt wurde, als die Version in der es geladen wird.

Die eingegebene Zeile ist zu lang. Syntaxfehler.

Einfach ein neues Terminal starten und noch einmal probieren

Wrong architecture

Visual Studio kompiliert immer mit 32 Bit, wenn vorher nicht

```
"C:\Program Files (x86)\Microsoft Visual Studio 14.0\VC\vcvarsall.bat"
x64
```

aufgerufen wurde. **Achtung**: CMake muss erneut aufgerufen werden.