Technische Universität Berlin

Computergrafik in Houdini

Oliver Gross *TU Berlin*

TechTalks der Freitagsrunde
Berlin, Dezember 2023

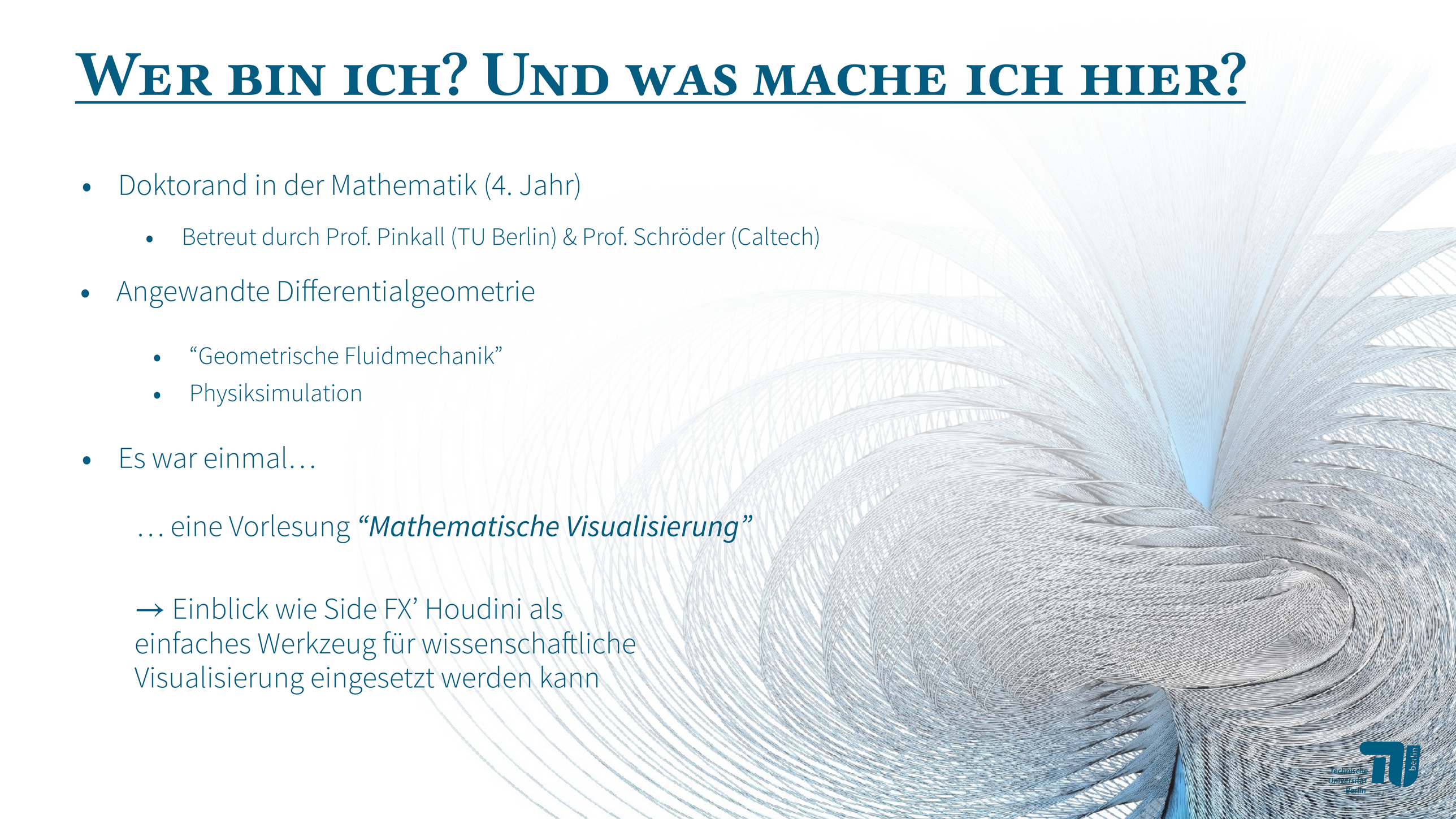
Berlin Mathematical School BMS · SideFX · DGD · SFB TRR 109 Discretization in Geometry and Dynamics

Wer bin ich? Und was mache ich hier?

- Doktorand in der Mathematik (4. Jahr)
  - Betreut durch Prof. Pinkall (TU Berlin) & Prof. Schröder (Caltech)
- Angewandte Differentialgeometrie
  - “Geometrische Fluidmechanik”
  - Physiksimulation
- Es war einmal…

  … eine Vorlesung *“Mathematische Visualisierung”*

  → Einblick wie Side FX’ Houdini als einfaches Werkzeug für wissenschaftliche Visualisierung eingesetzt werden kann



Technische Universität Berlin

Houdini 20 Sneak Peek

Houdini 20 Sneak Peek

Konzept

→ Einblick wie Side FX’ Houdini als einfaches Werkzeug für wissenschaftliche Visualisierung eingesetzt werden kann

- Erstellen von Abbildungen
- “rapid prototyping”
- Rendering
- Simulation

Konzept

→ Einblick wie Side FX’ Houdini als einfaches Werkzeug für wissenschaftliche Visualisierung eingesetzt werden kann

Bemerkung: Es gibt viele Alternativen welche ähnliche Workflows erlauben. Zum Beispiel:

- LibIGL
- Geometry Central
- Polyscope
- Gpytoolbox, Pyddg + Blender
- Penrose…

Konzept

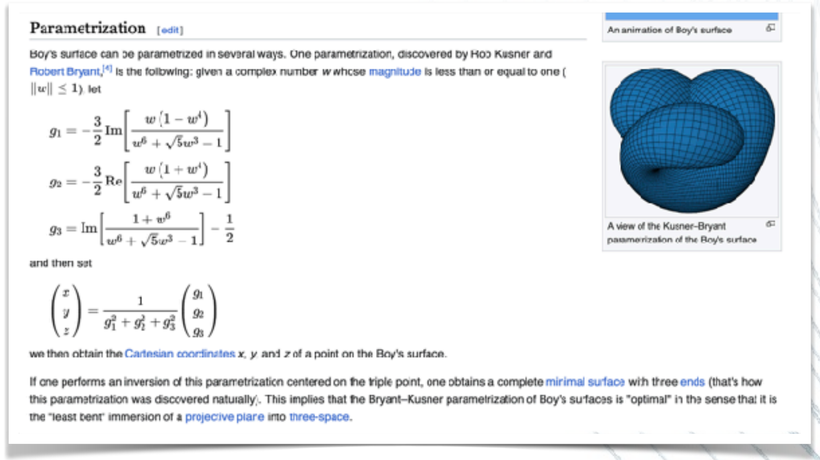
Ziel ist ein grundlegendes Verständnis für das:

- User-Interface und Datenstrukturen
- Erzeugen und Manipulieren von Geometrie
- Rendern von Figuren und Animationen für z.B. Hausarbeiten, oder Vorträge

Konzept

Ziel ist ein grundlegendes Verständnis für das:

- User-Interface und Datenstrukturen
- Erzeugen und Manipulieren von Geometrie
- Rendern von Figuren und Animationen für z.B. Hausarbeiten, oder Vorträge



Parametrization [edit]

Boy's surface can be parametrized in several ways. One parametrization, discovered by Hoo Kusner and Robert Bryant,[4] is the following: given a complex number w whose magnitude is less than or equal to one ($\|w\| \leq 1$), let

$$g_1 = -\frac{3}{2}\operatorname{Im}\left[\frac{w\left(1-w^4\right)}{w^6+\sqrt{5}w^3-1}\right]$$

$$g_2 = -\frac{3}{2}\operatorname{Re}\left[\frac{w\left(1+w^4\right)}{w^6+\sqrt{5}w^3-1}\right]$$

$$g_3 = \operatorname{Im}\left[\frac{1+w^6}{w^6+\sqrt{5}w^3-1}\right] - \frac{1}{2}$$

and then set

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \frac{1}{g_1^2+g_2^2+g_3^2}\begin{pmatrix} g_1 \\ g_2 \\ g_3 \end{pmatrix}$$

we then obtain the Cartesian coordinates x, y, and z of a point on the Boy's surface.

If one performs an inversion of this parametrization centered on the triple point, one obtains a complete minimal surface with three ends (that's how this parametrization was discovered naturally). This implies that the Bryant–Kusner parametrization of Boy's surfaces is "optimal" in the sense that it is the "least bent" immersion of a projective plane into three-space.

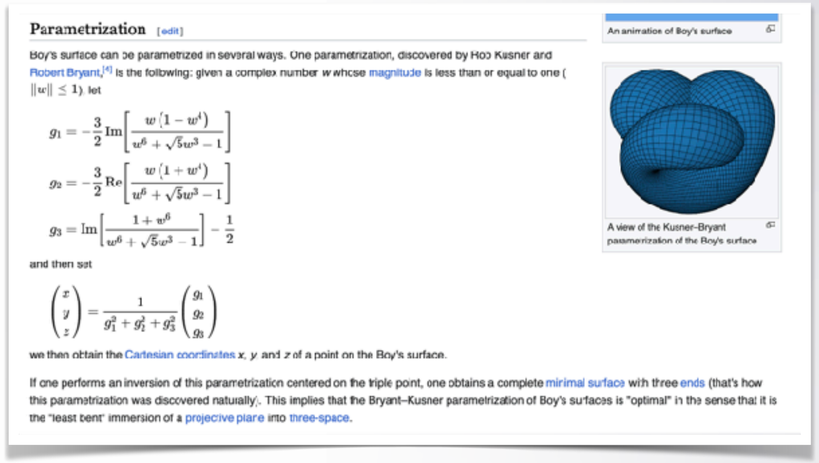
An animation of Boy's surface

A view of the Kusner–Bryant parametrization of the Boy's surface

Konzept

Ziel ist ein grundlegendes Verständnis für das:

- User-Interface und Datenstrukturen
- Erzeugen und Manipulieren von Geometrie
- Rendern von Figuren und Animationen für z.B. Hausarbeiten, oder Vorträge



Parametrization [edit]

Boy's surface can be parametrized in several ways. One parametrization, discovered by Rob Kusner and Robert Bryant,[4] is the following: given a complex number w whose magnitude is less than or equal to one ($\|w\| \leq 1$), let

$$g_1 = -\frac{3}{2}\operatorname{Im}\left[\frac{w\left(1-w^4\right)}{w^6+\sqrt{5}w^3-1}\right]$$

$$g_2 = -\frac{3}{2}\operatorname{Re}\left[\frac{w\left(1+w^4\right)}{w^6+\sqrt{5}w^3-1}\right]$$

$$g_3 = \operatorname{Im}\left[\frac{1+w^6}{w^6+\sqrt{5}w^3-1}\right]-\frac{1}{2}$$

and then set

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \frac{1}{g_1^2+g_2^2+g_3^2}\begin{pmatrix} g_1 \\ g_2 \\ g_3 \end{pmatrix}$$

we then obtain the Cartesian coordinates x, y, and z of a point on the Boy's surface.

If one performs an inversion of this parametrization centered on the triple point, one obtains a complete minimal surface with three ends (that's how this parametrization was discovered naturally). This implies that the Bryant–Kusner parametrization of Boy's surfaces is "optimal" in the sense that it is the "least bent" immersion of a projective plane into three-space.

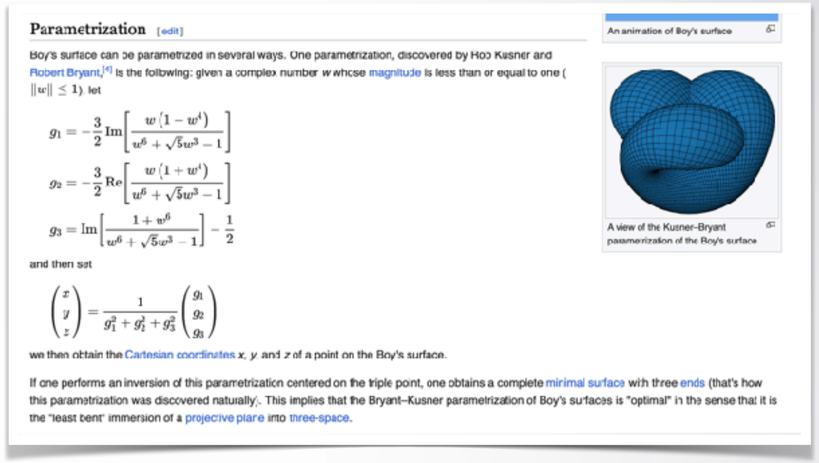
An animation of Boy's surface

A view of the Kusner–Bryant parametrization of the Boy's surface

Konzept

Ziel ist ein grundlegendes Verständnis für das:

- User-Interface und Datenstrukturen
- Erzeugen und Manipulieren von Geometrie
- Rendern von Figuren und Animationen für z.B. Hausarbeiten, oder Vorträge



Parametrization [edit]

Boy's surface can be parametrized in several ways. One parametrization, discovered by Rob Kusner and Robert Bryant,[4] is the following: given a complex number w whose magnitude is less than or equal to one ($\|w\| \leq 1$), let

$$g_1 = -\frac{3}{2}\operatorname{Im}\left[\frac{w\left(1-w^4\right)}{w^6+\sqrt{5}w^3-1}\right]$$

$$g_2 = -\frac{3}{2}\operatorname{Re}\left[\frac{w\left(1+w^4\right)}{w^6+\sqrt{5}w^3-1}\right]$$

$$g_3 = \operatorname{Im}\left[\frac{1+w^6}{w^6+\sqrt{5}w^3-1}\right]-\frac{1}{2}$$

and then set

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \frac{1}{g_1^2+g_2^2+g_3^2}\begin{pmatrix} g_1 \\ g_2 \\ g_3 \end{pmatrix}$$

we then obtain the Cartesian coordinates x, y, and z of a point on the Boy's surface.

If one performs an inversion of this parametrization centered on the triple point, one obtains a complete minimal surface with three ends (that's how this parametrization was discovered naturally). This implies that the Bryant–Kusner parametrization of Boy's surfaces is "optimal" in the sense that it is the "least bent" immersion of a projective plane into three-space.

An animation of Boy's surface

A view of the Kusner–Bryant parametrization of the Boy's surface

Technische Universität Berlin

Erste Schritte

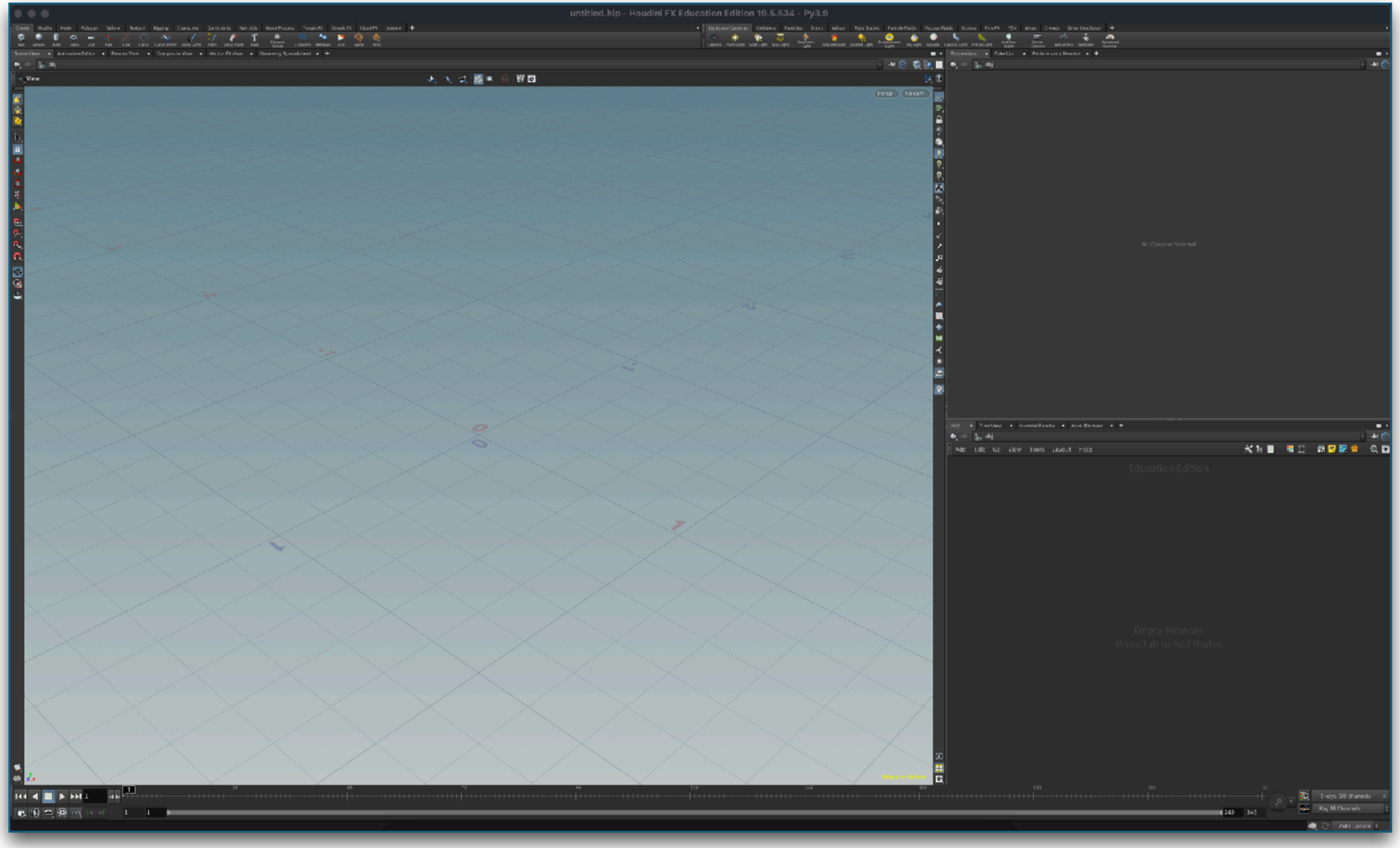
Berlin Mathematical School BMS

SideFX

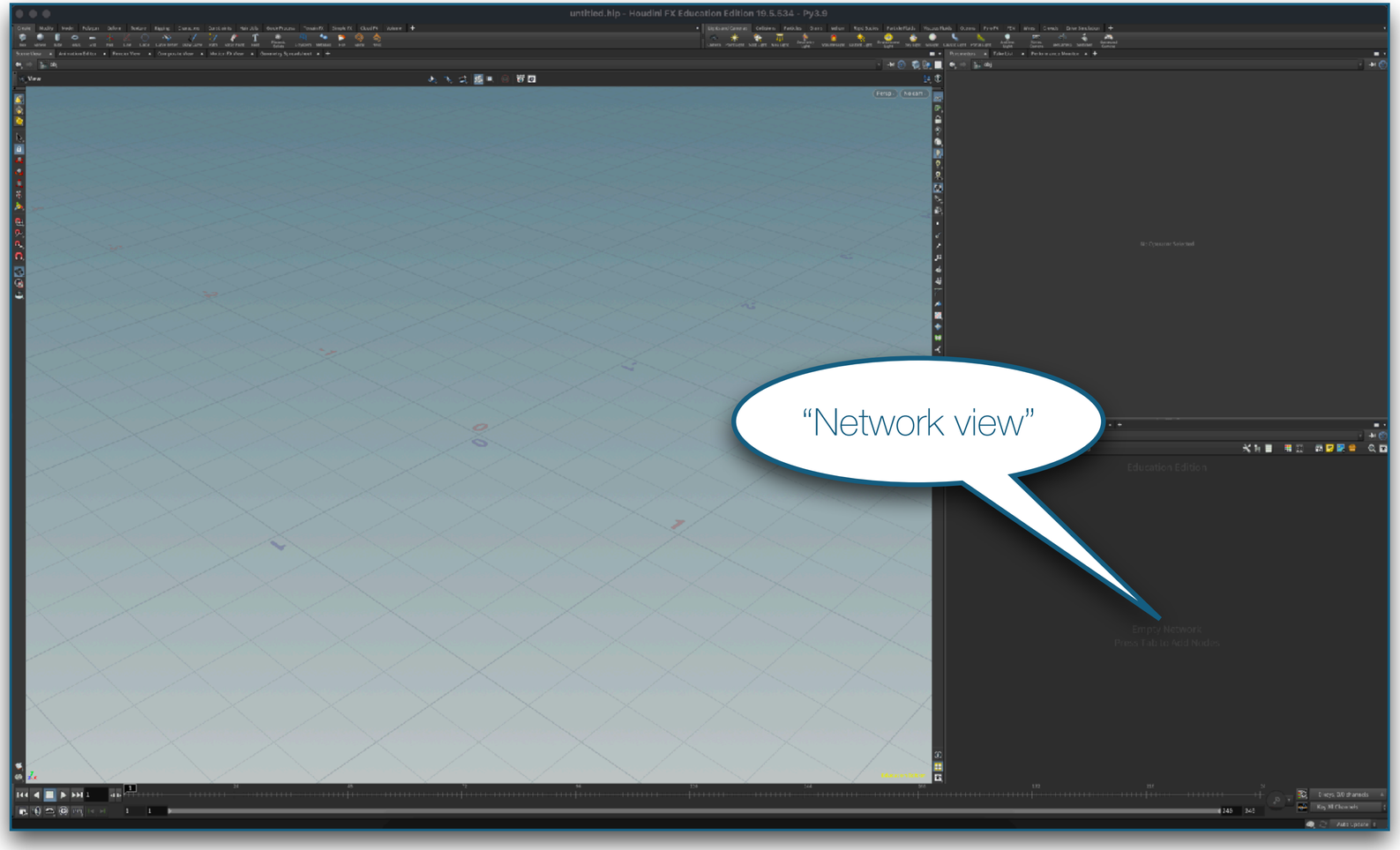
DGD

SFB TRR 109 Discretization in Geometry and Dynamics

Der Startbildschirm

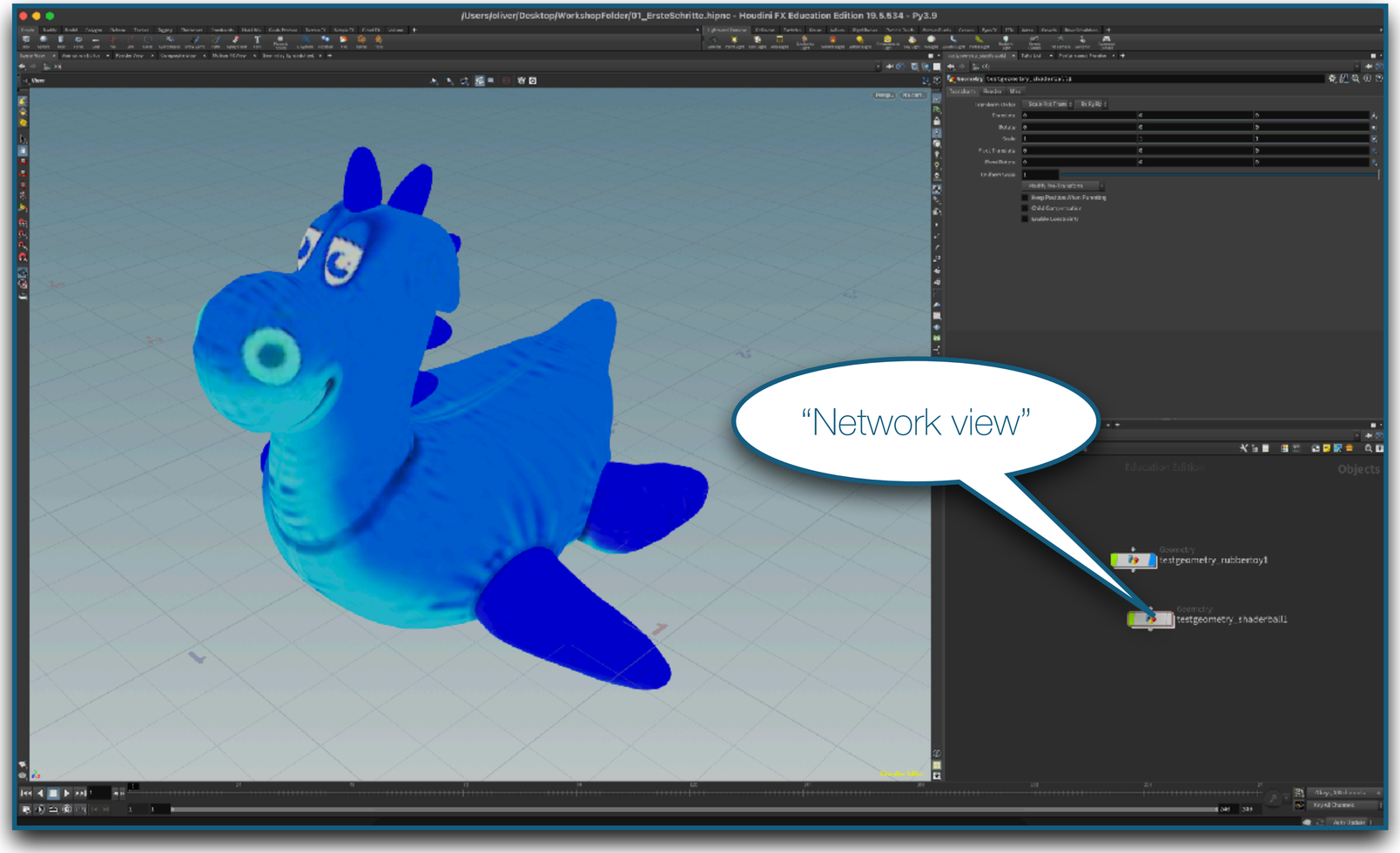


Der Startbildschirm



Der Startbildschirm

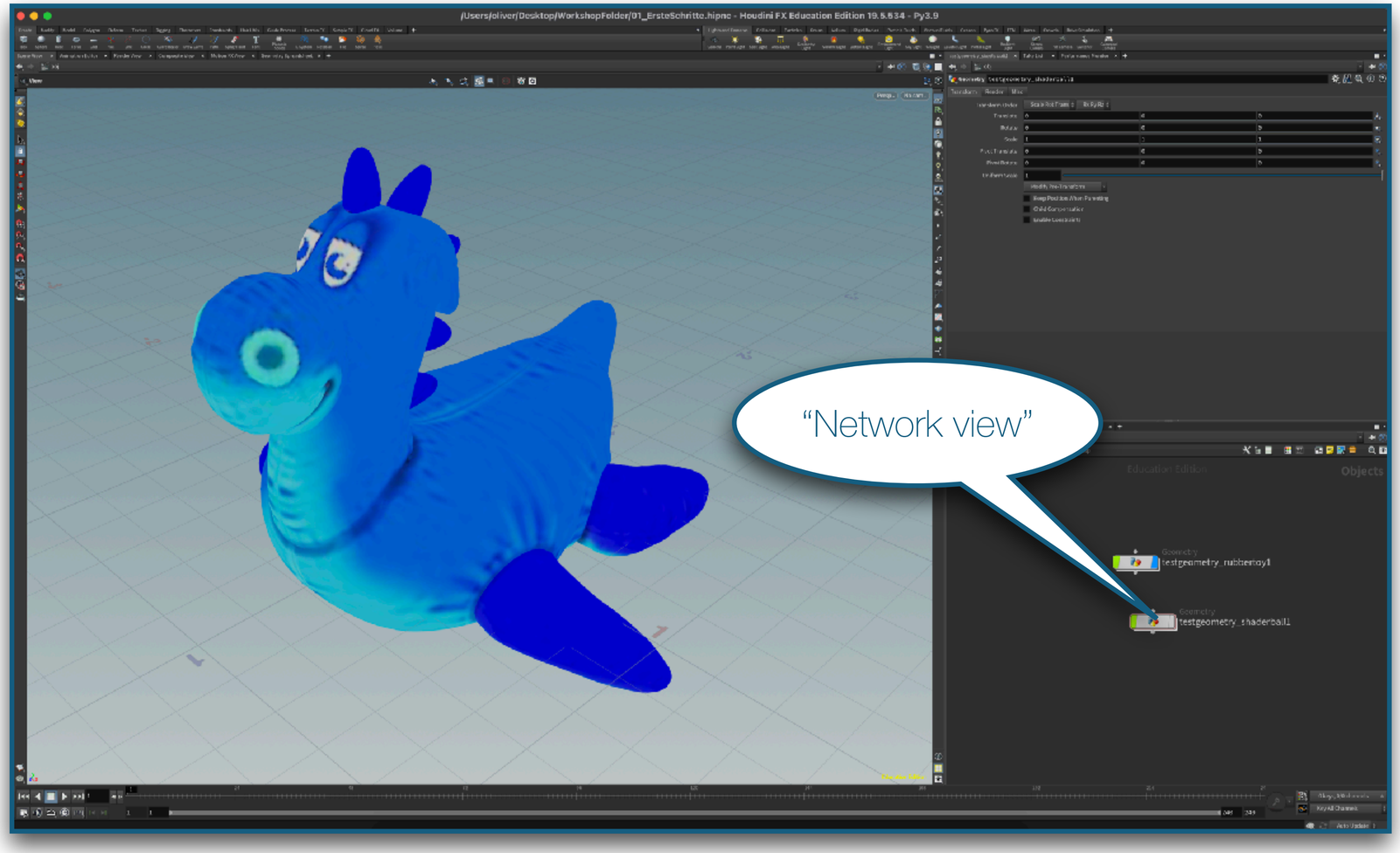
- Übersicht über das file



Der Startbildschirm

- Übersicht über das file

- Später finden wir hier neben Geometrie auch z.B. Licht und Kameras

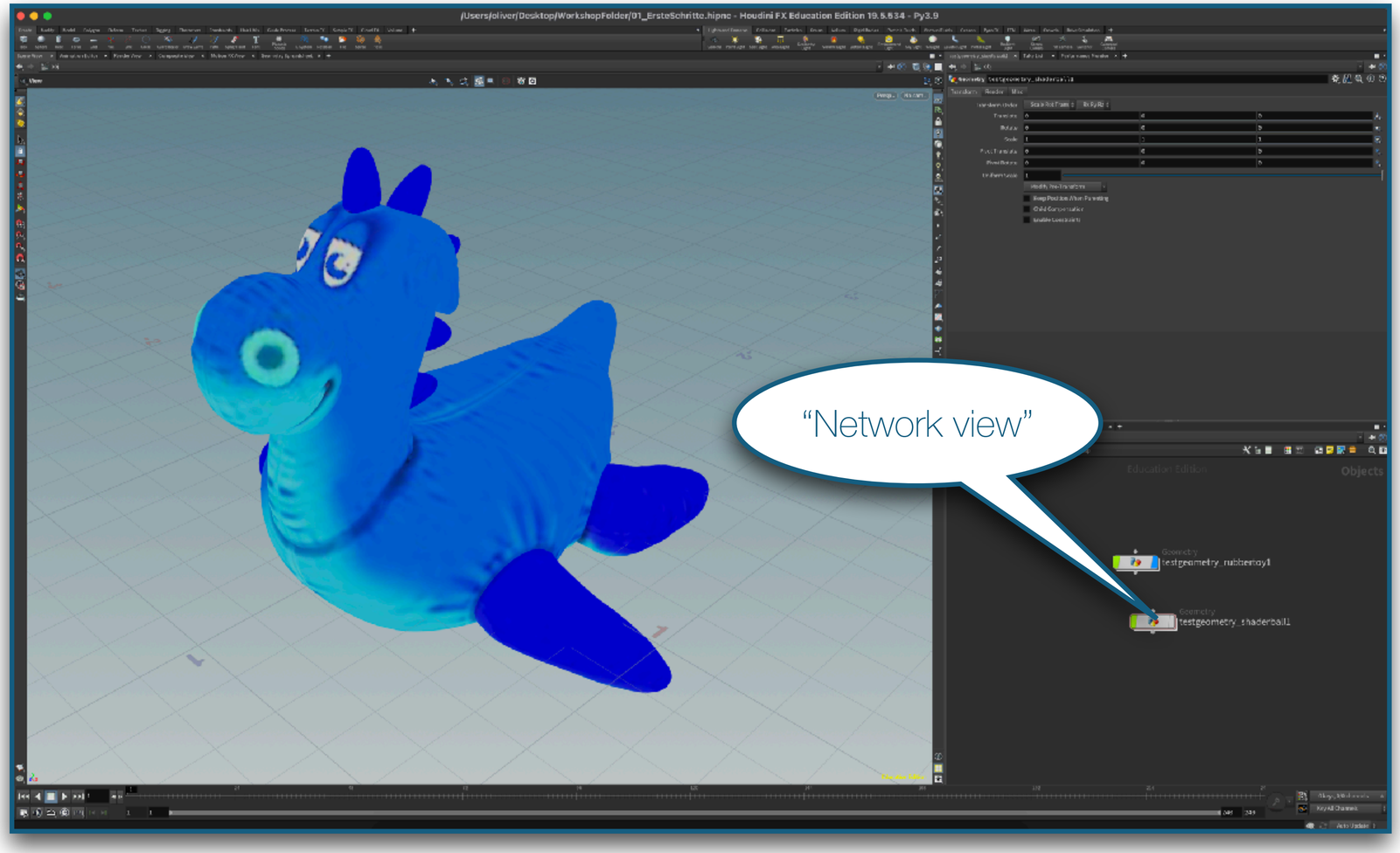


Der Startbildschirm

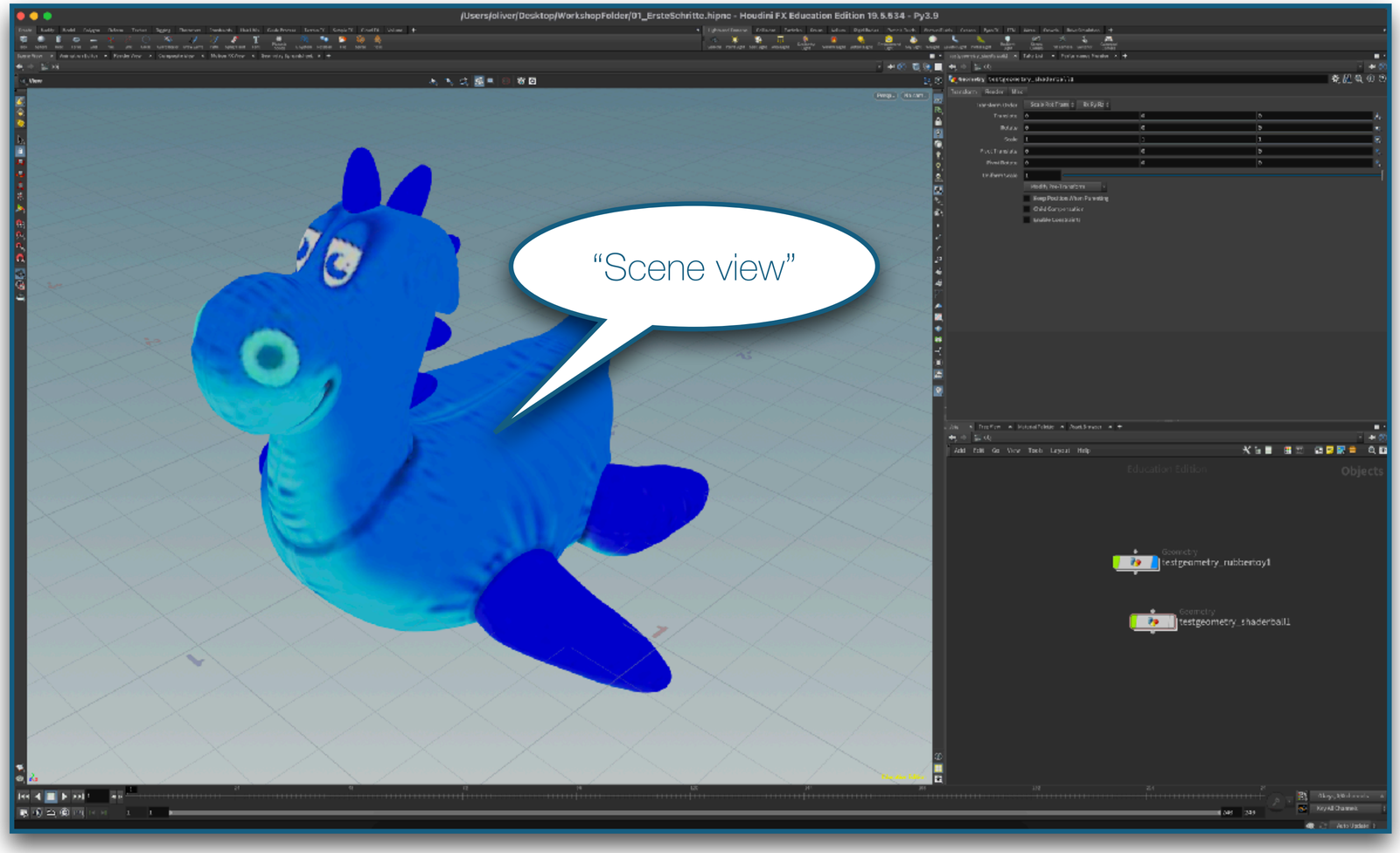
- Übersicht über das file

- Später finden wir hier z.B. Geometrie, Licht und Kameras

- “Geometry” Knoten sind zentraler Ort des Geschehens

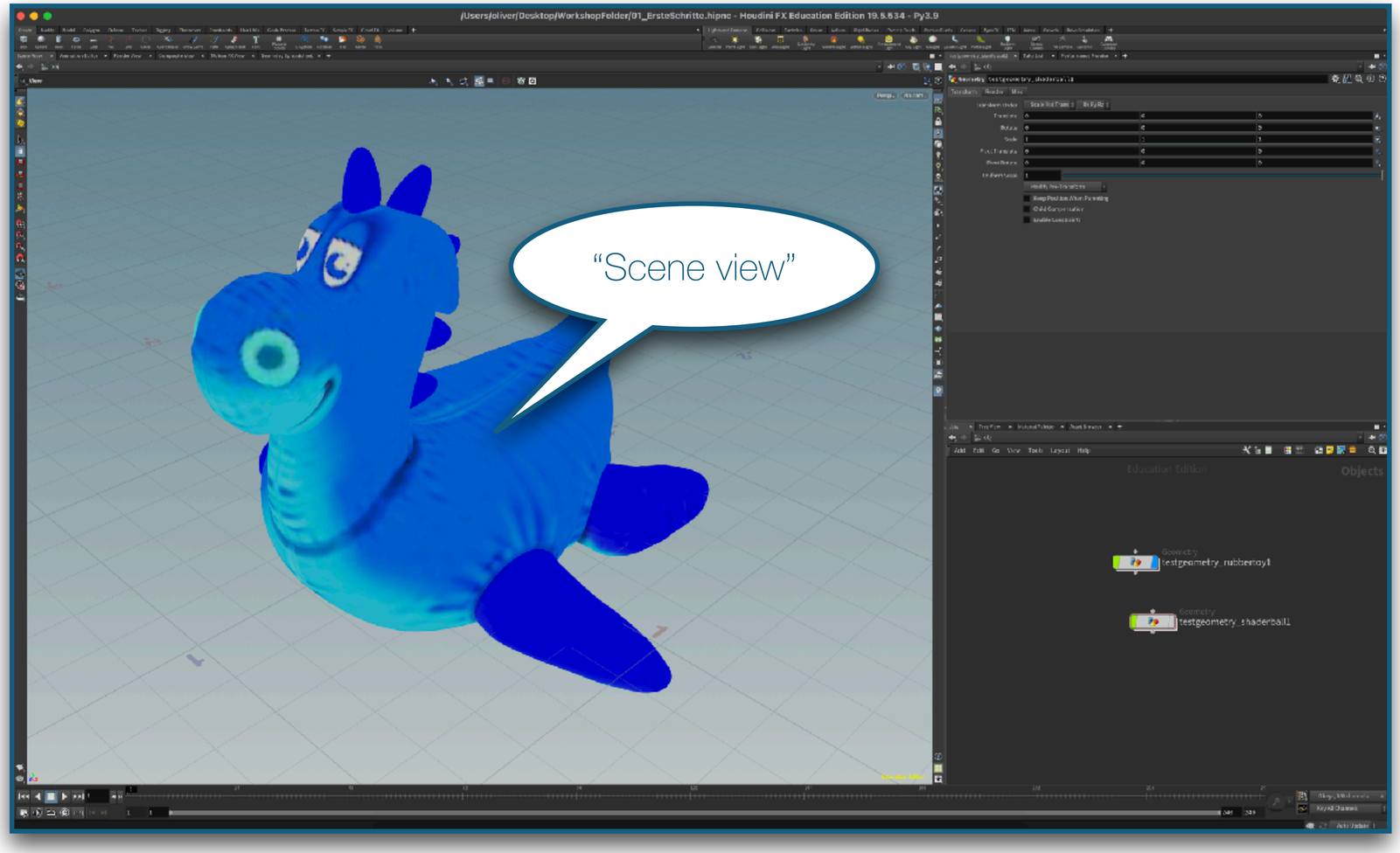


Der Startbildschirm



Der Startbildschirm

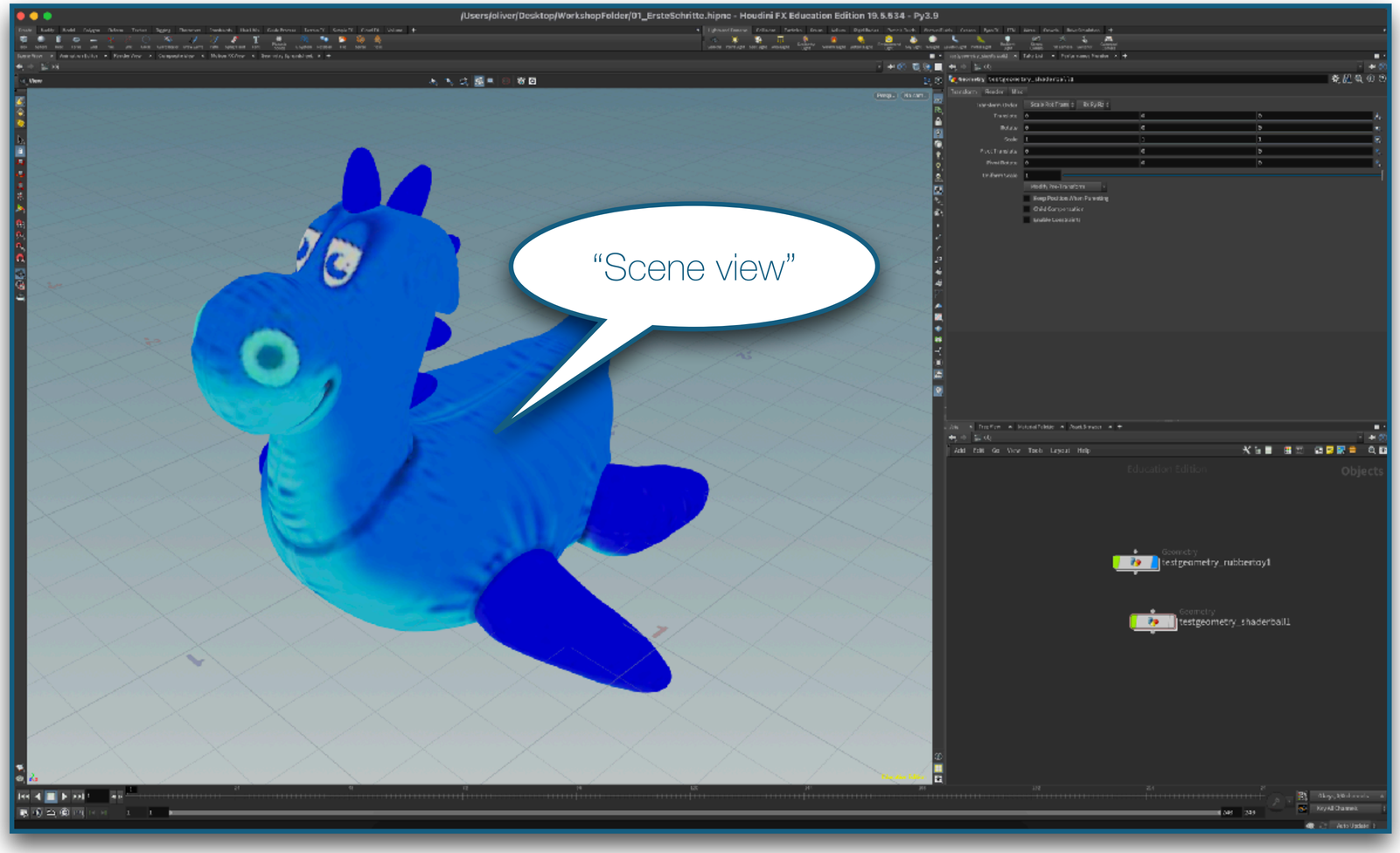
- zeigt ausgewählte Szene



Der Startbildschirm

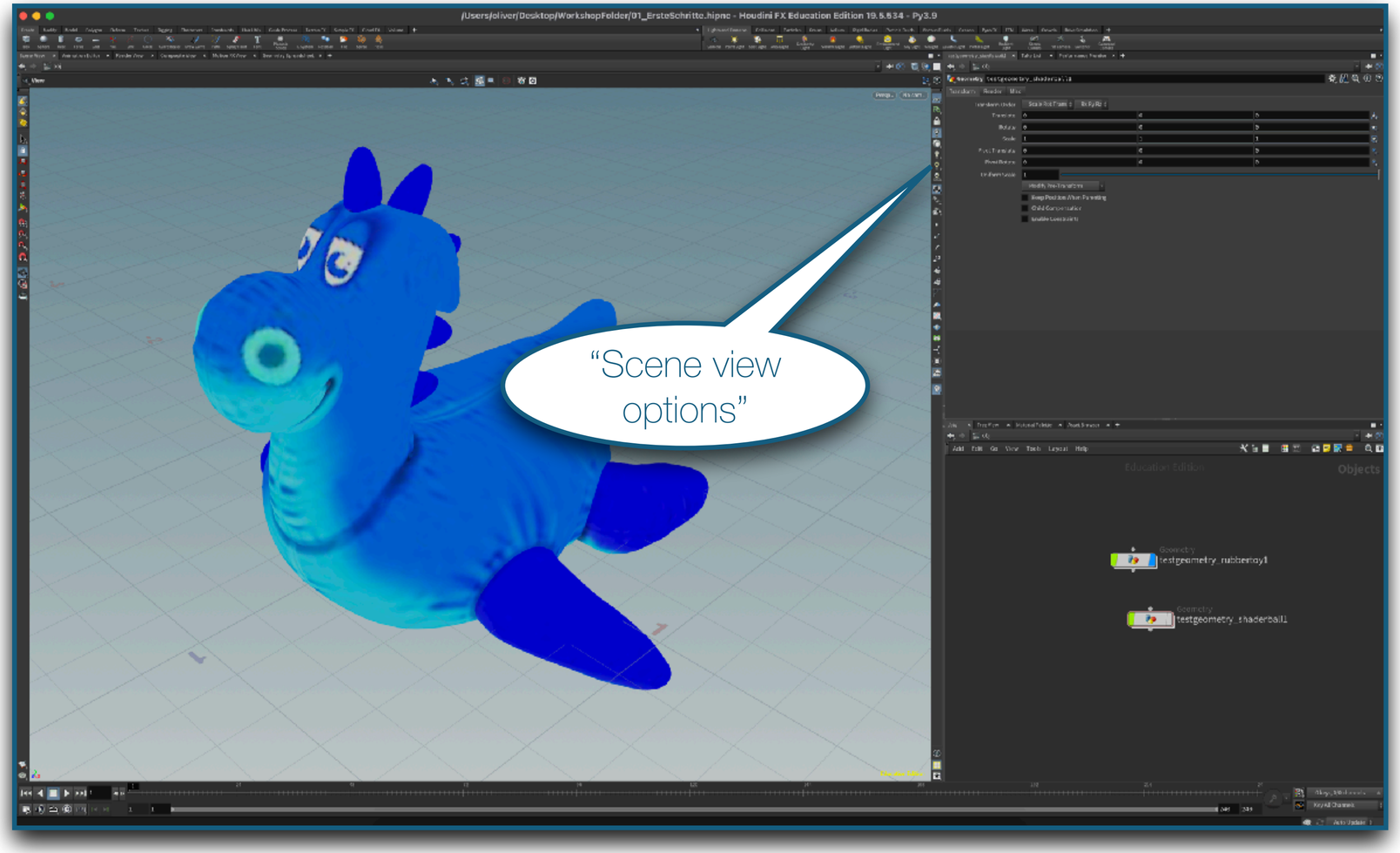
- zeigt ausgewählte Szene

- Auswahl der Szene durch den blauen Marker an den Knoten

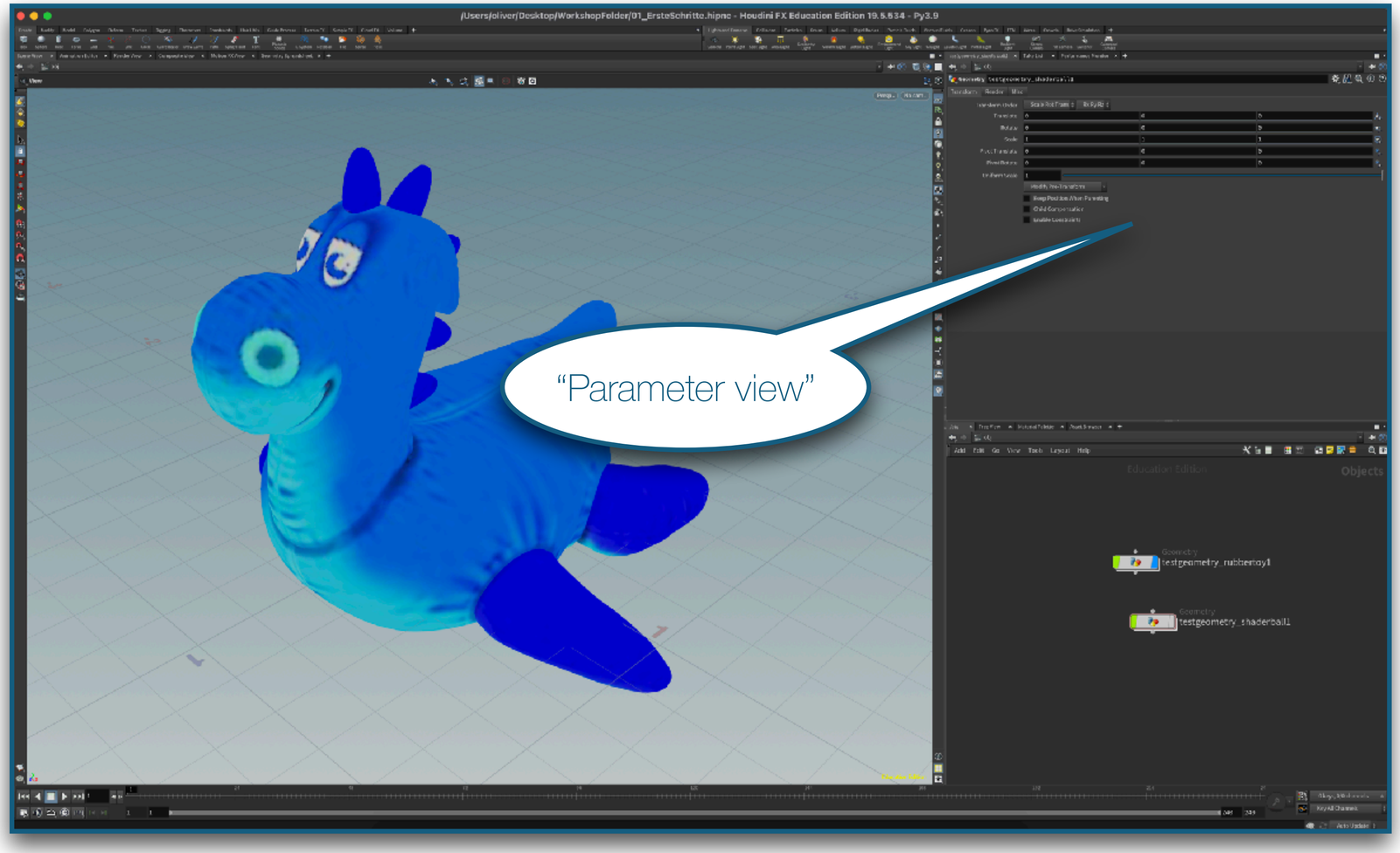


Der Startbildschirm

verschiedene Darstellungsoptionen für den Scene view

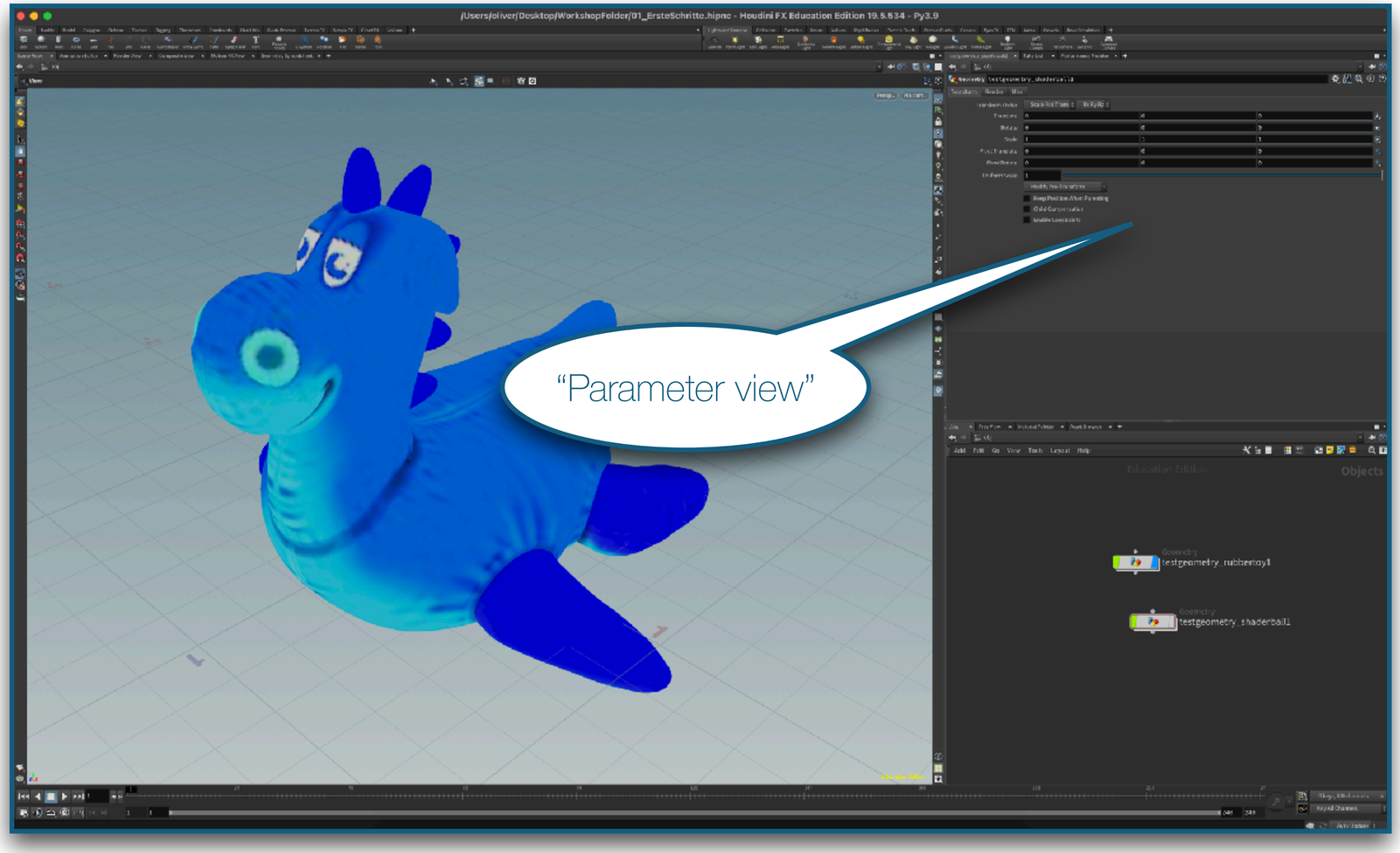


Der Startbildschirm



Der Startbildschirm

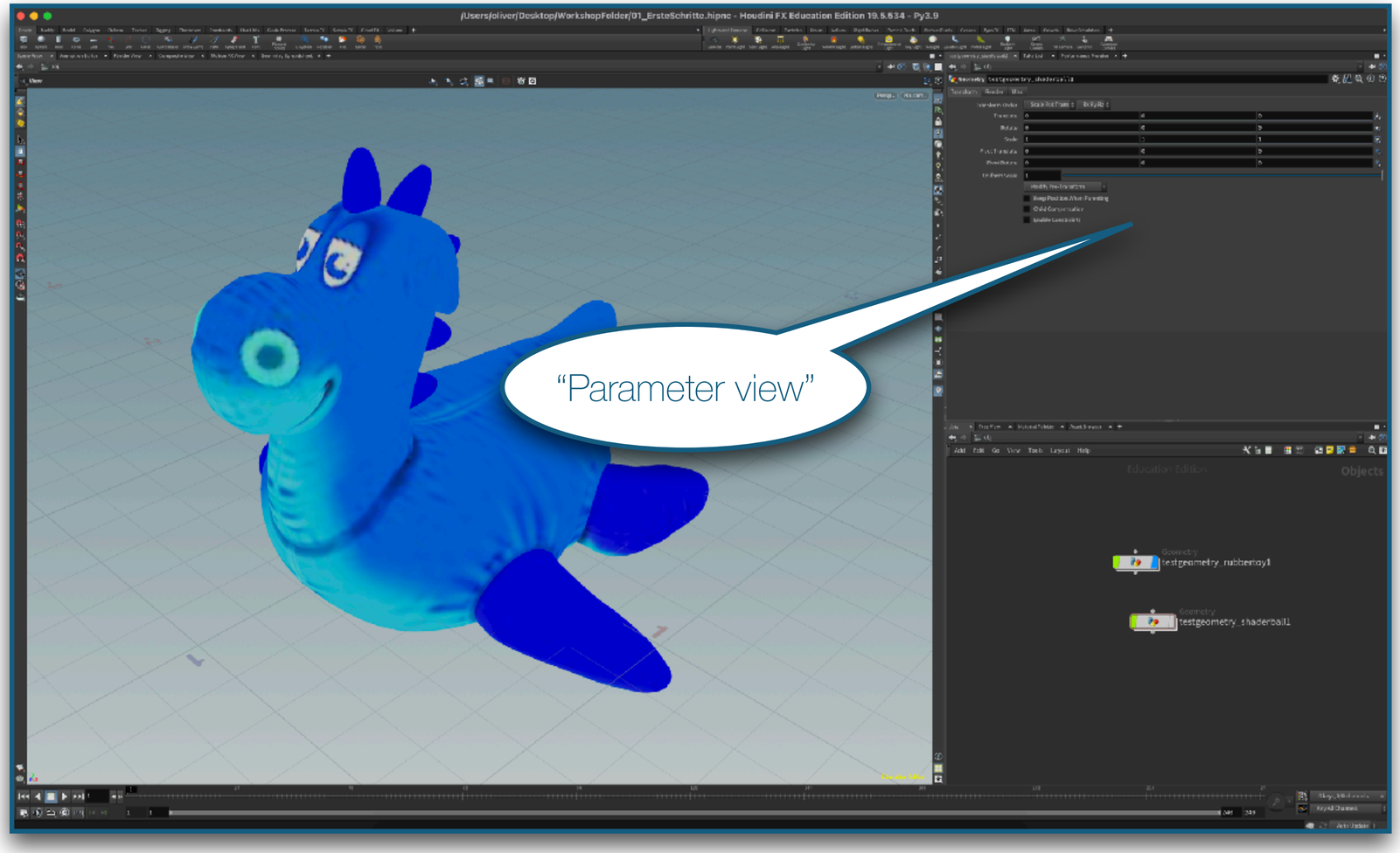
- zeigt “Optionen” eines ausgewählten Knotens



Der Startbildschirm

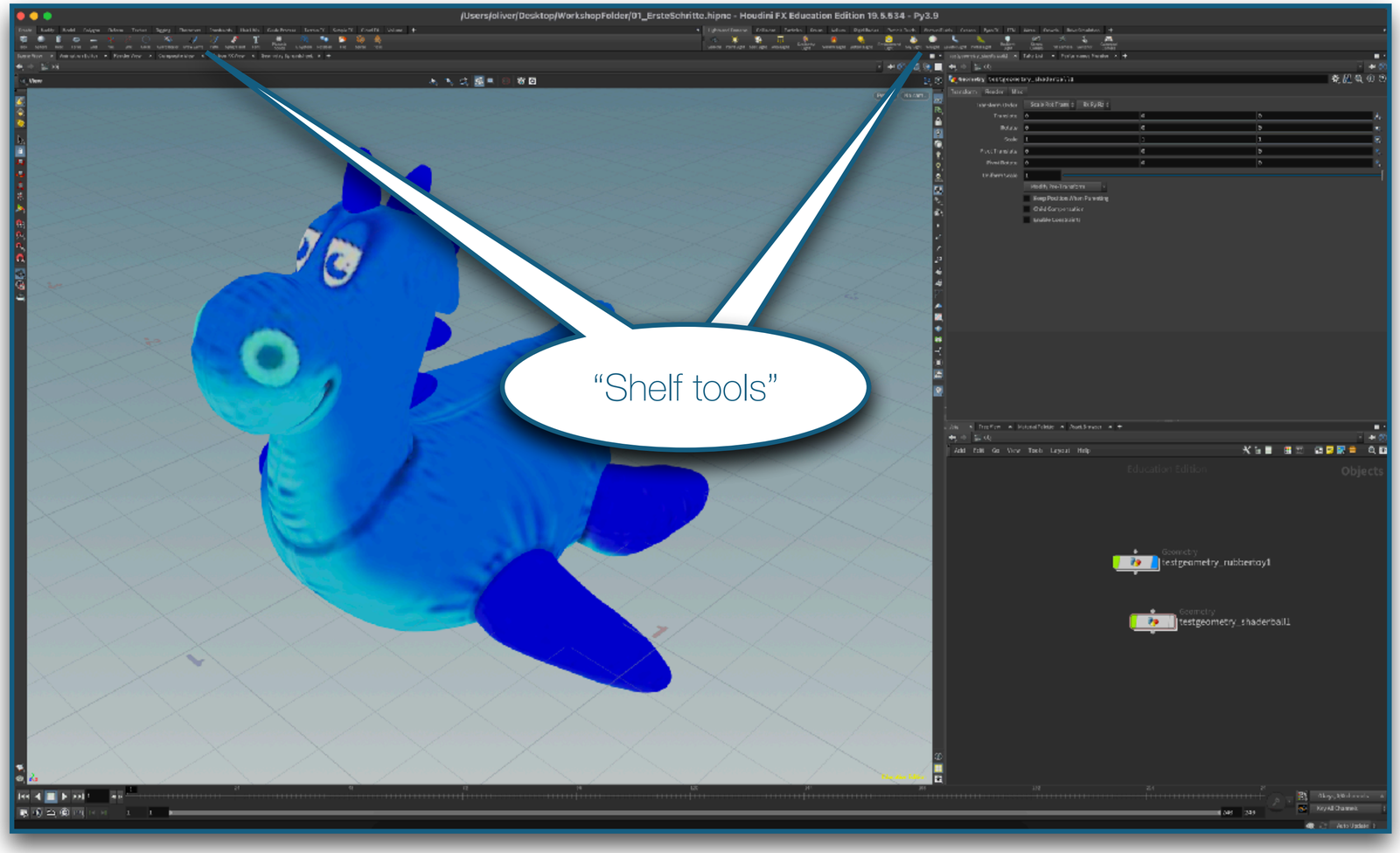
- zeigt “Optionen” eines ausgewählten Knotens

- später erlauben uns “Wrangle”-Knoten erlauben uns lokale Codeschnipsel zu schreiben



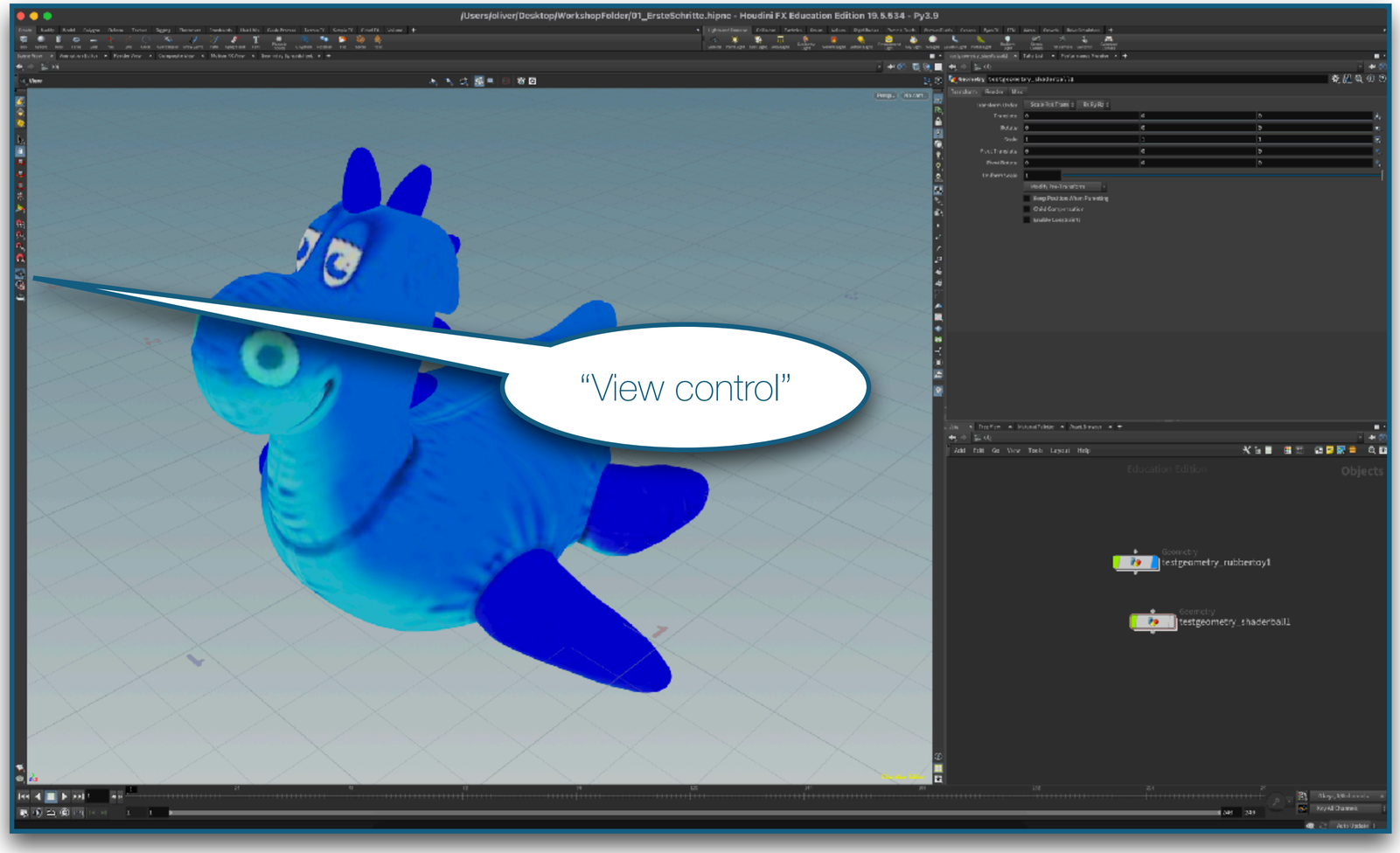
Der Startbildschirm

- Sammlungen von Template-Netwerken



Der Startbildschirm

- Markiere die Kamera um im Szene view zu navigieren

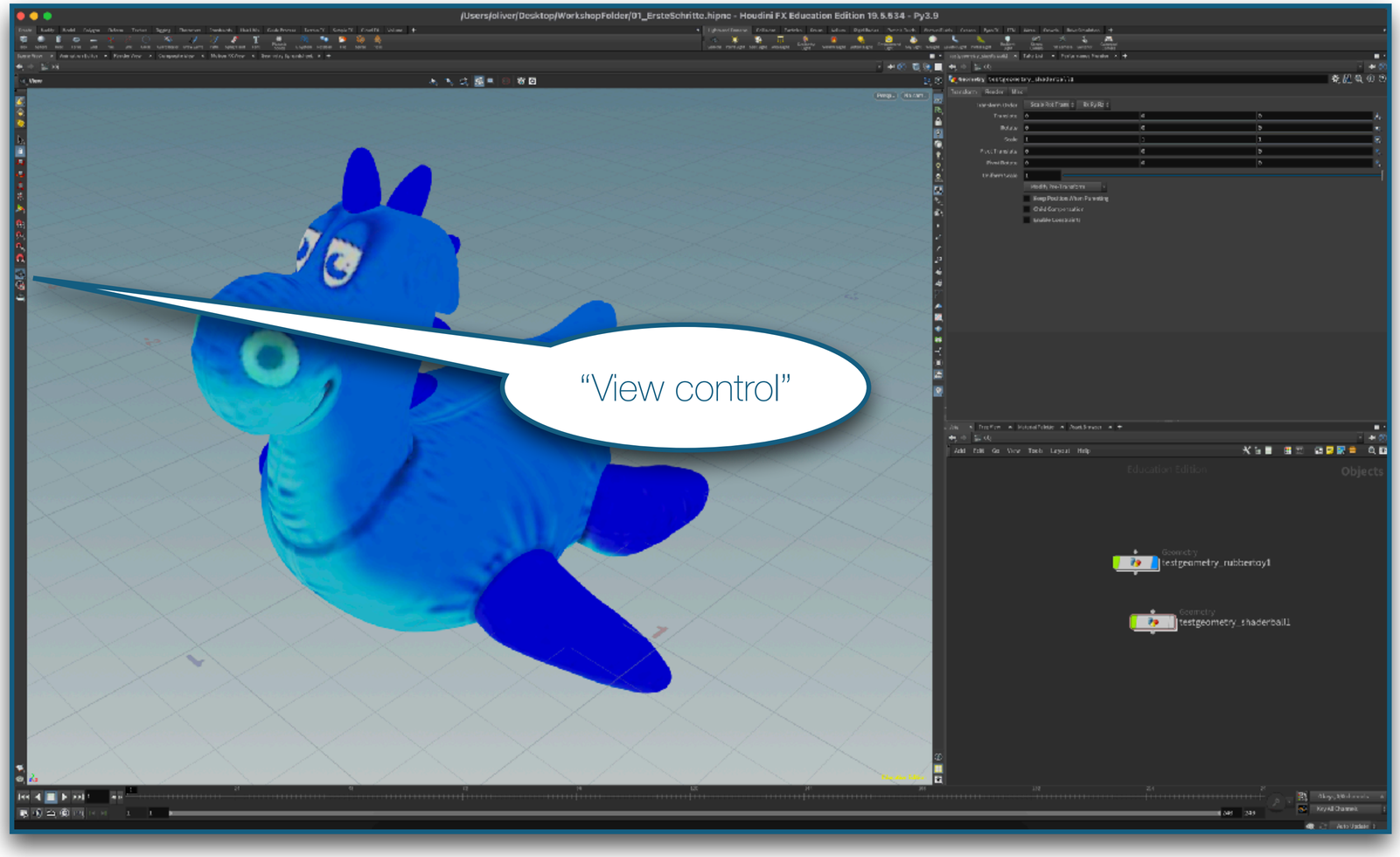


Der Startbildschirm

- Markiere die Kamera um im Szene view zu navigieren

Nützliche Shortcuts:

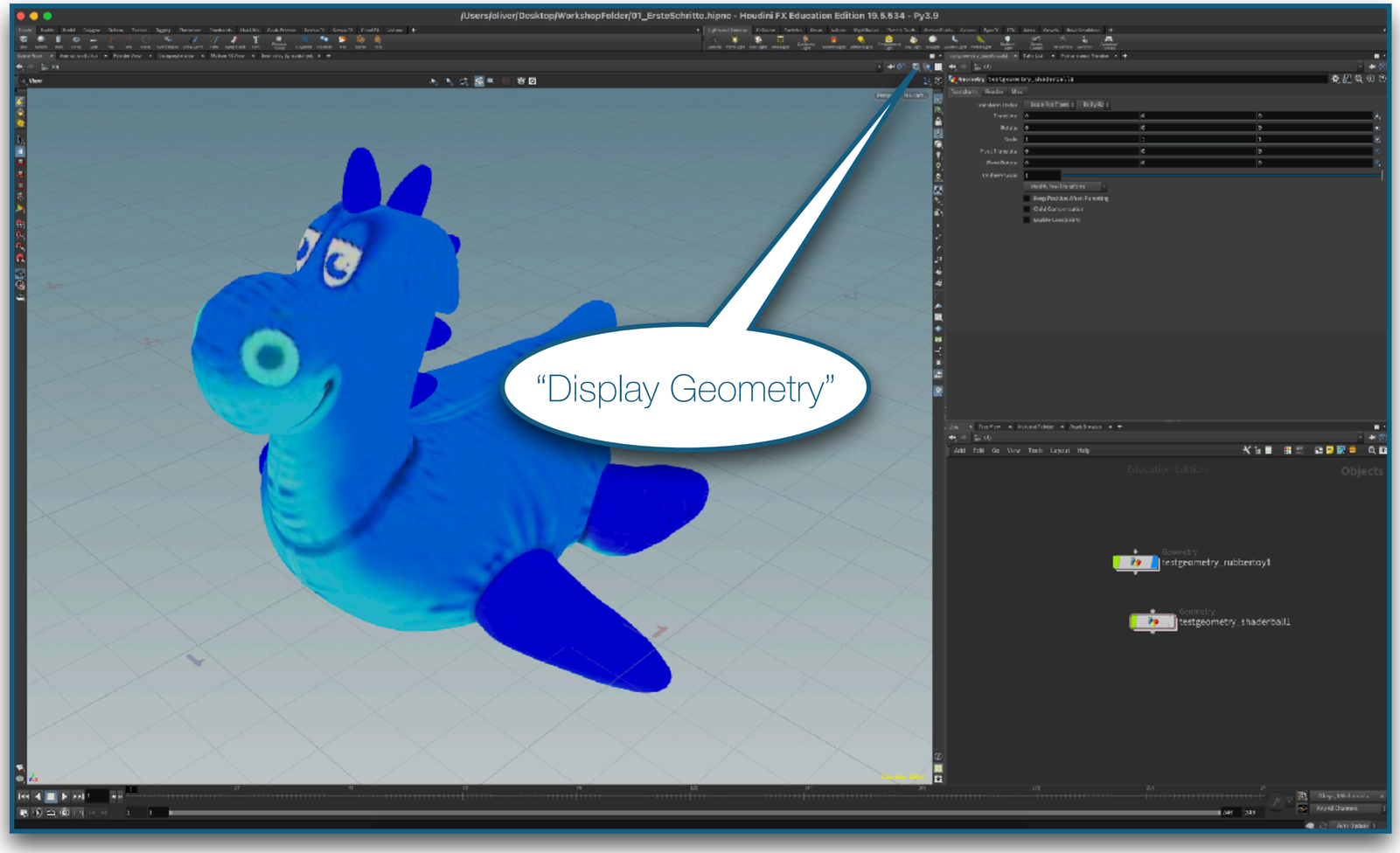
- “H”-Taste zum zentrieren der Szene im Szene view
- “1”, “2”, “3”, “4” für eine orthografische Sicht auf die Szene entlang der jeweiligen Koordinatenachse



Der Startbildschirm

verschiedene Darstellungsoptionen für Geometrien, z.B.

- “smooth shading”
- “flat shading”,
- Jeweils mit oder ohne “wireframe”



Der Startbildschirm

Übung:

a) Erzeuge eine “rubbertoy” und eine “shader ball” Testgeometrie.

b) Nutze die blauen Marker um sie einzeln und zusammen anzuzeigen.

c) Zeige die Geometrie nur als Wireframe an

d) Zeige die Geometrie “smooth shaded” and und vergleiche die “Lighting” Optionen

Alles klar?

Technische Universität Berlin

Berlin Mathematical School BMS

SideFX

DGD

SFB TRR 109 Discretization in Geometry and Dynamics

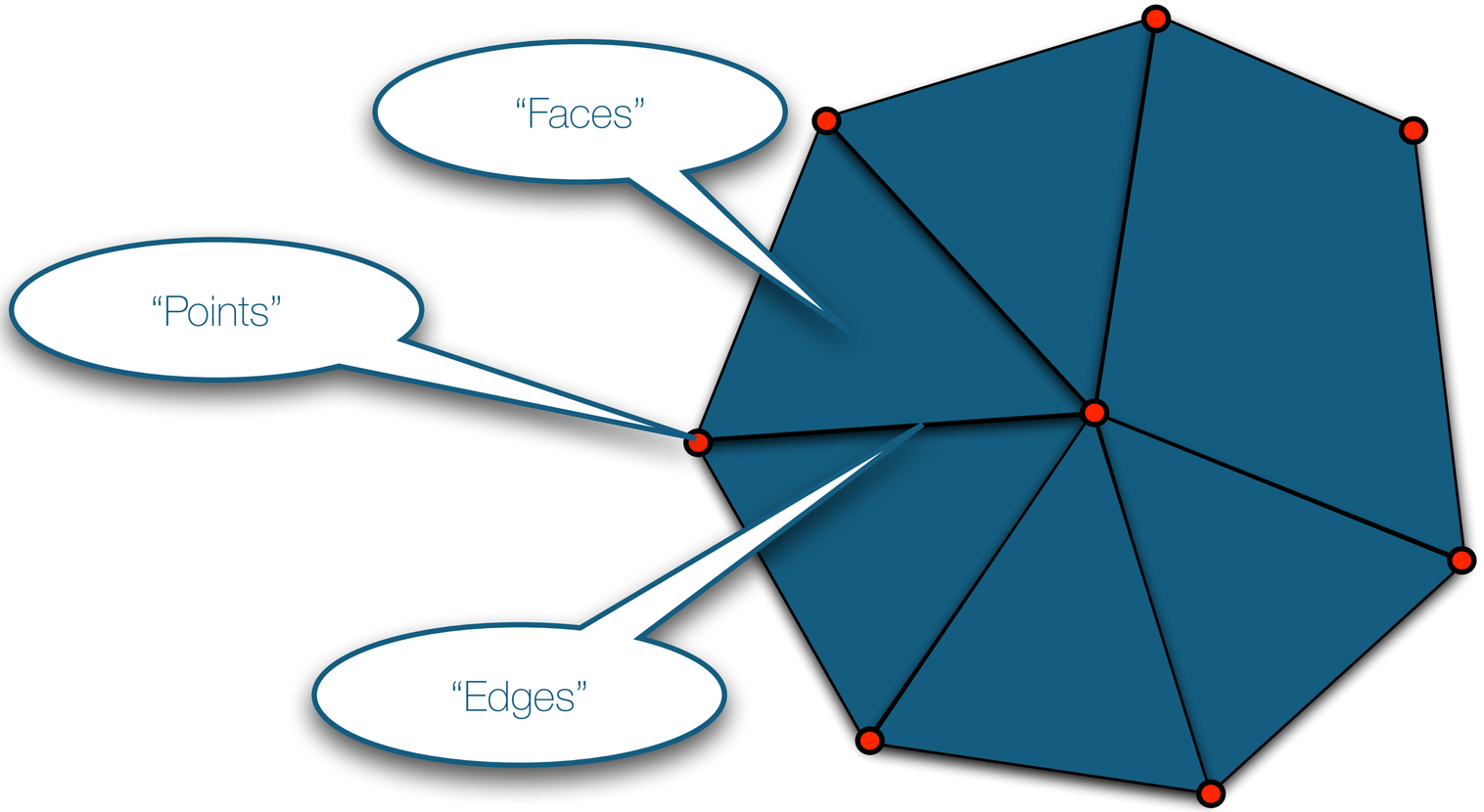
Technische Universität Berlin

Datenstruktur und Attribute

Berlin Mathematical School BMS · SideFX · DGD · SFB TRR 109 Discretization in Geometry and Dynamics

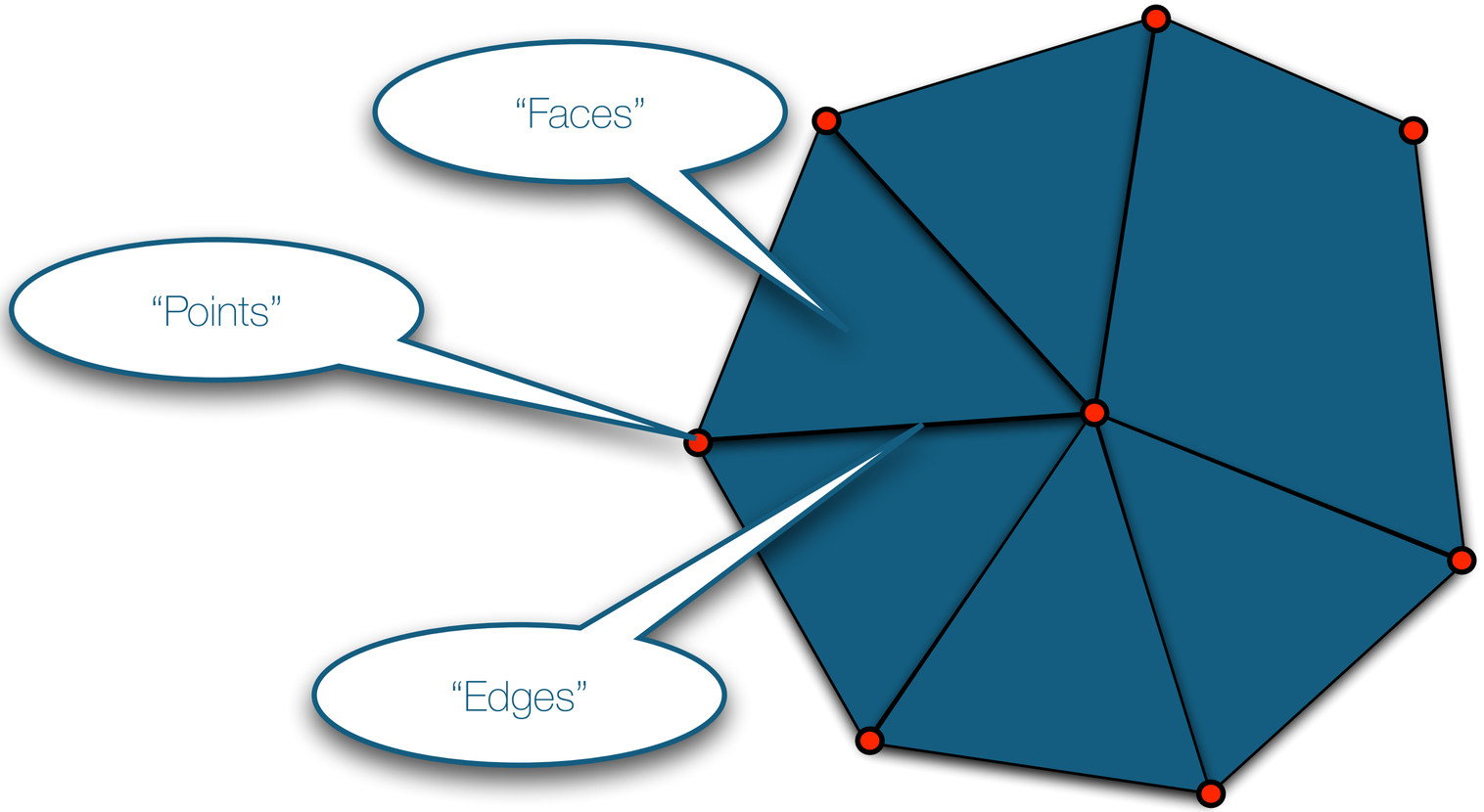
Half-Edge Datenstruktur

- Wir betrachten ein Polygonales mesh



Half-Edge Datenstruktur

- Wir betrachten ein Polygonales mesh

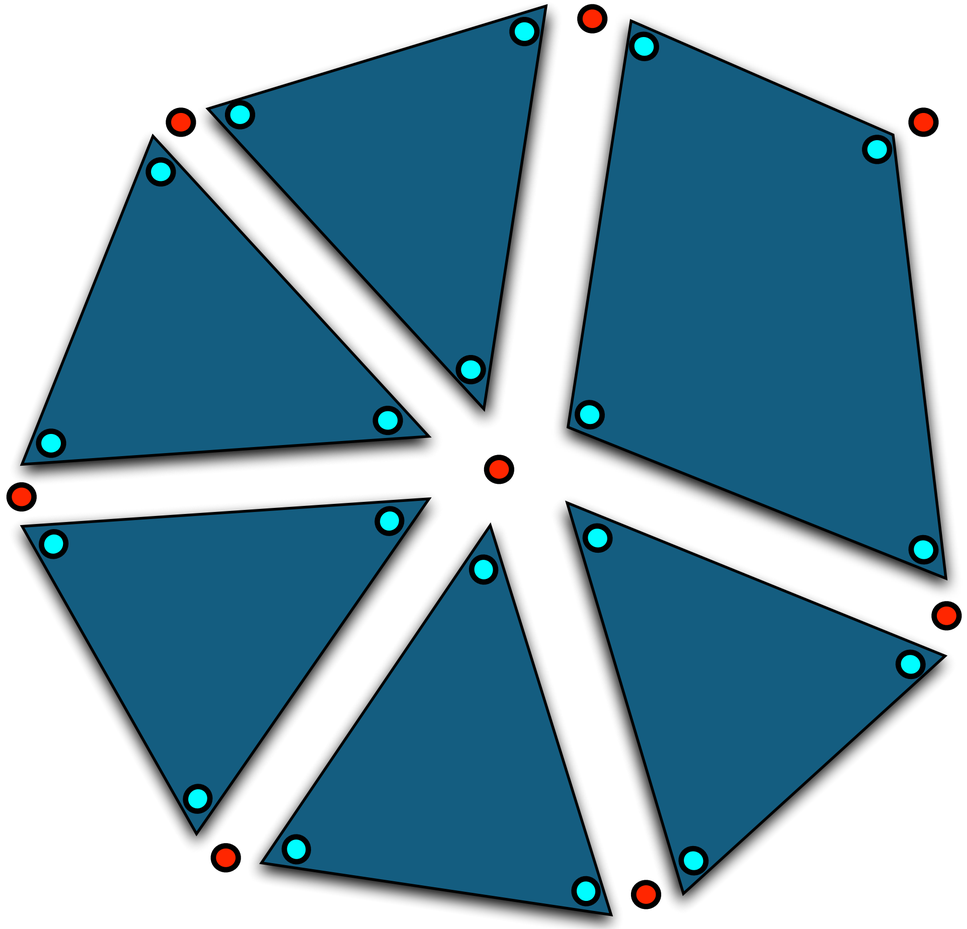


In Houdini’s half-edge Datenstruktur gilt:

Point ≠ Vertex

Half-Edge Datenstruktur

- Wir betrachten ein Polygonales mesh

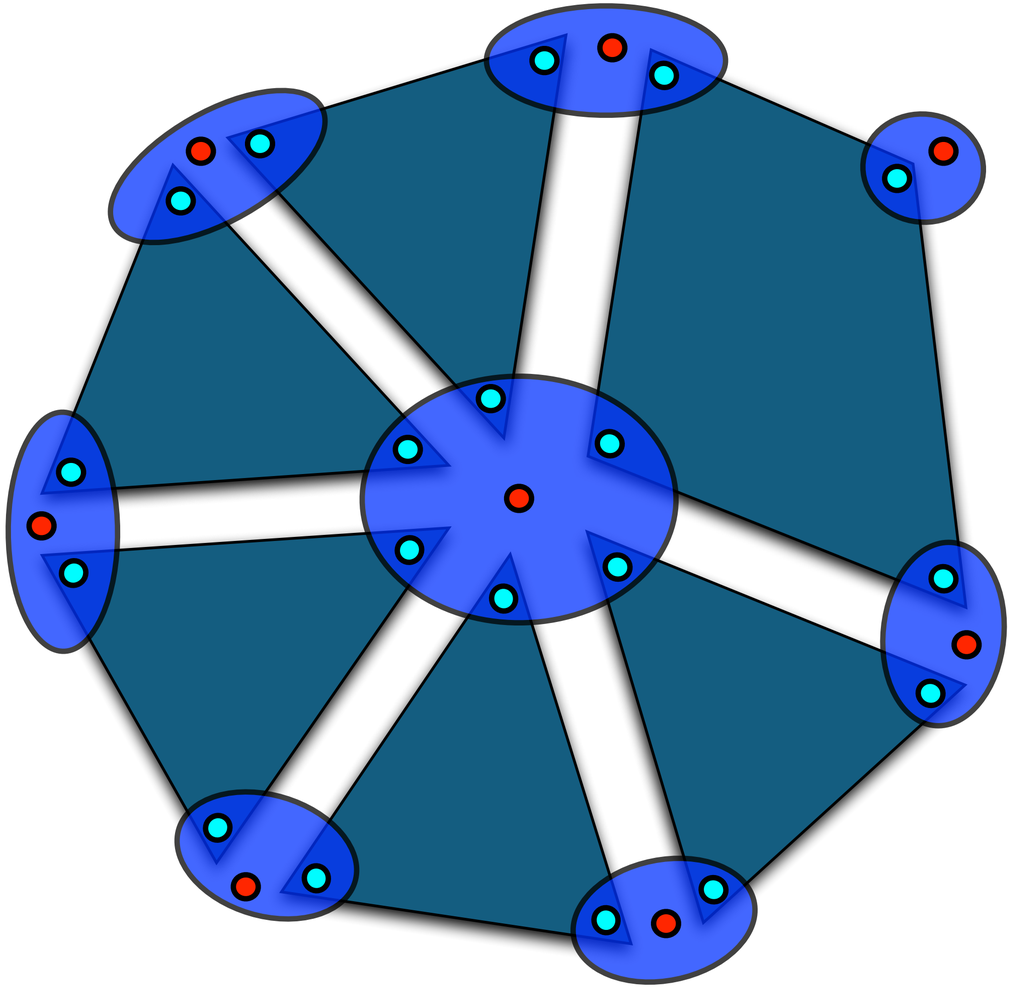


In Houdini's half-edge Datenstruktur gilt:

Point ≠ **Vertex**

Half-Edge Datenstruktur

- Wir betrachten ein Polygonales mesh

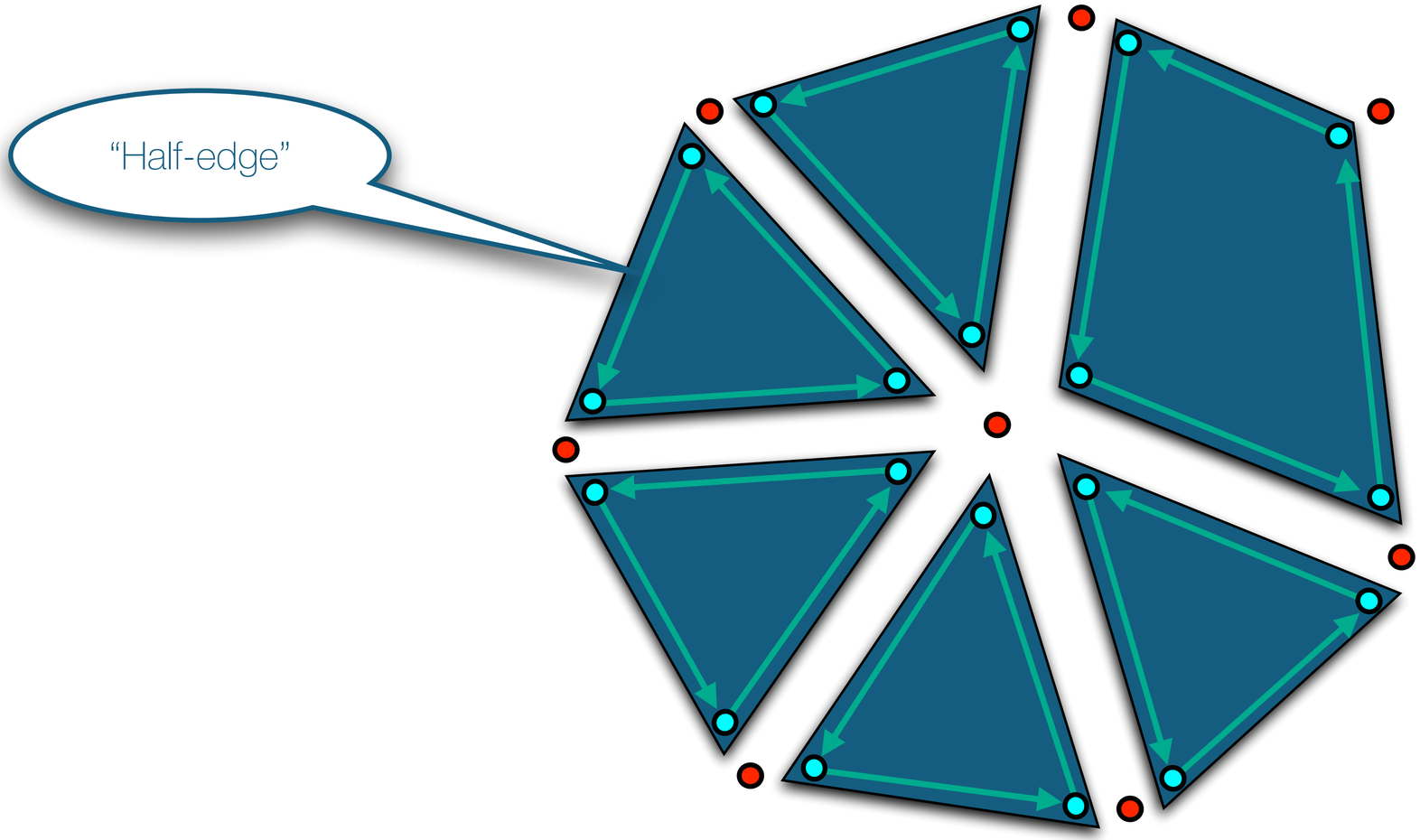


In Houdini's half-edge Datenstruktur gilt:

Point ≠ **Vertex**

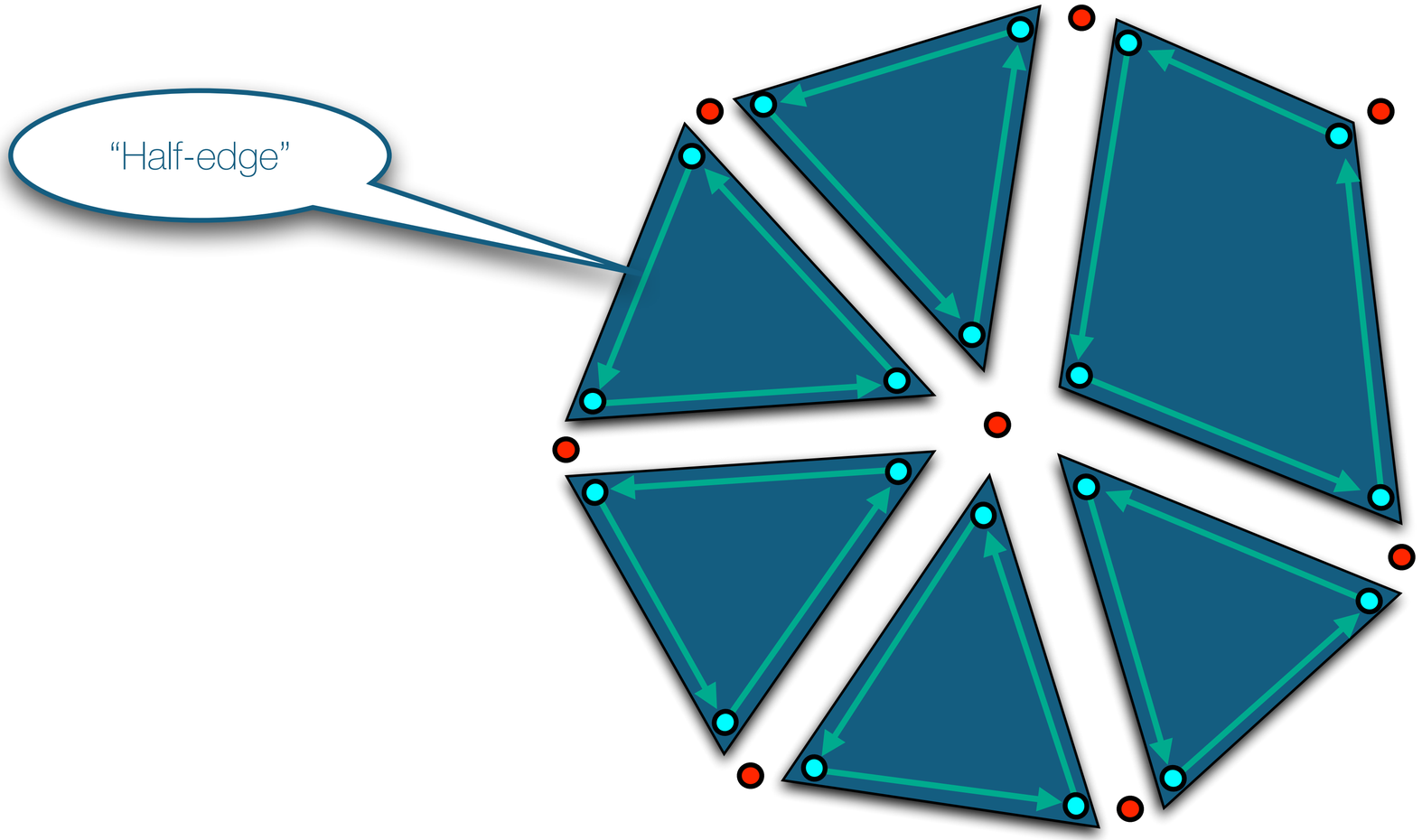
Half-Edge Datenstruktur

- Wir betrachten ein Polygonales mesh



Half-Edge Datenstruktur

- Wir betrachten ein Polygonales mesh

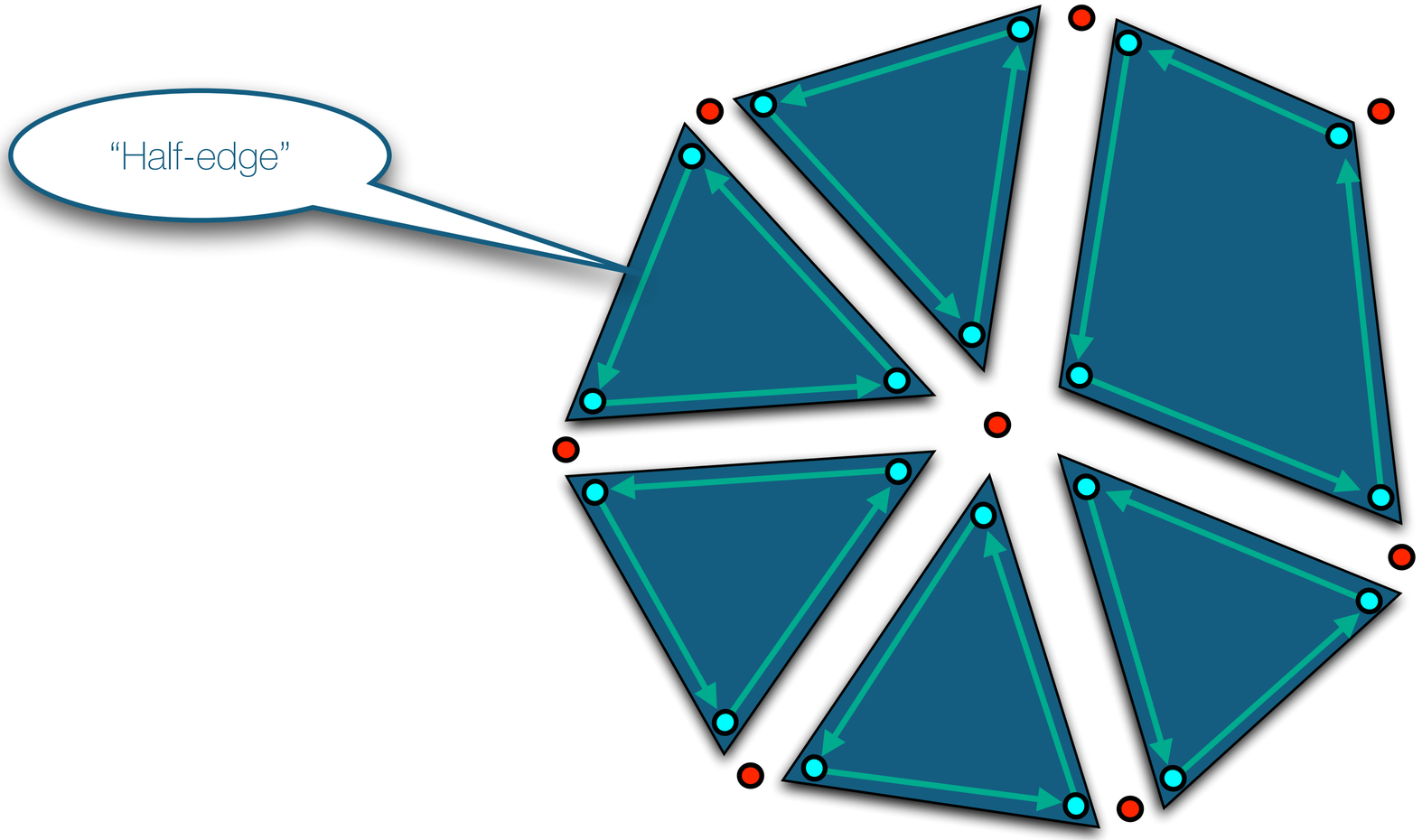


Wir können über folgende Objekte iterieren:

- Faces
- Punkte
- Vertices

Half-Edge Datenstruktur

- Wir betrachten ein Polygonales mesh

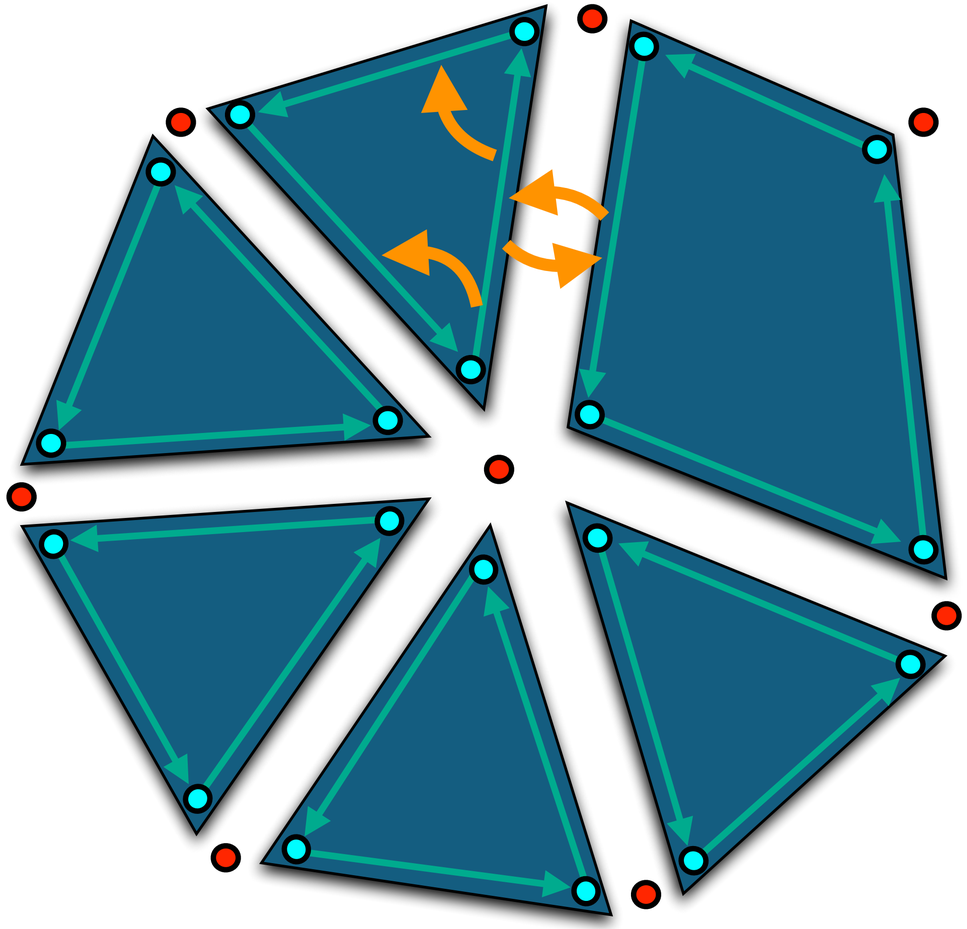


Wir können über folgende Objekte iterieren:

- Faces
- Punkte
- Vertices
- Half-edges sind nicht iterierbar, aber gehören zum source-vertex

Half-Edge Datenstruktur

- Wir betrachten ein Polygonales mesh

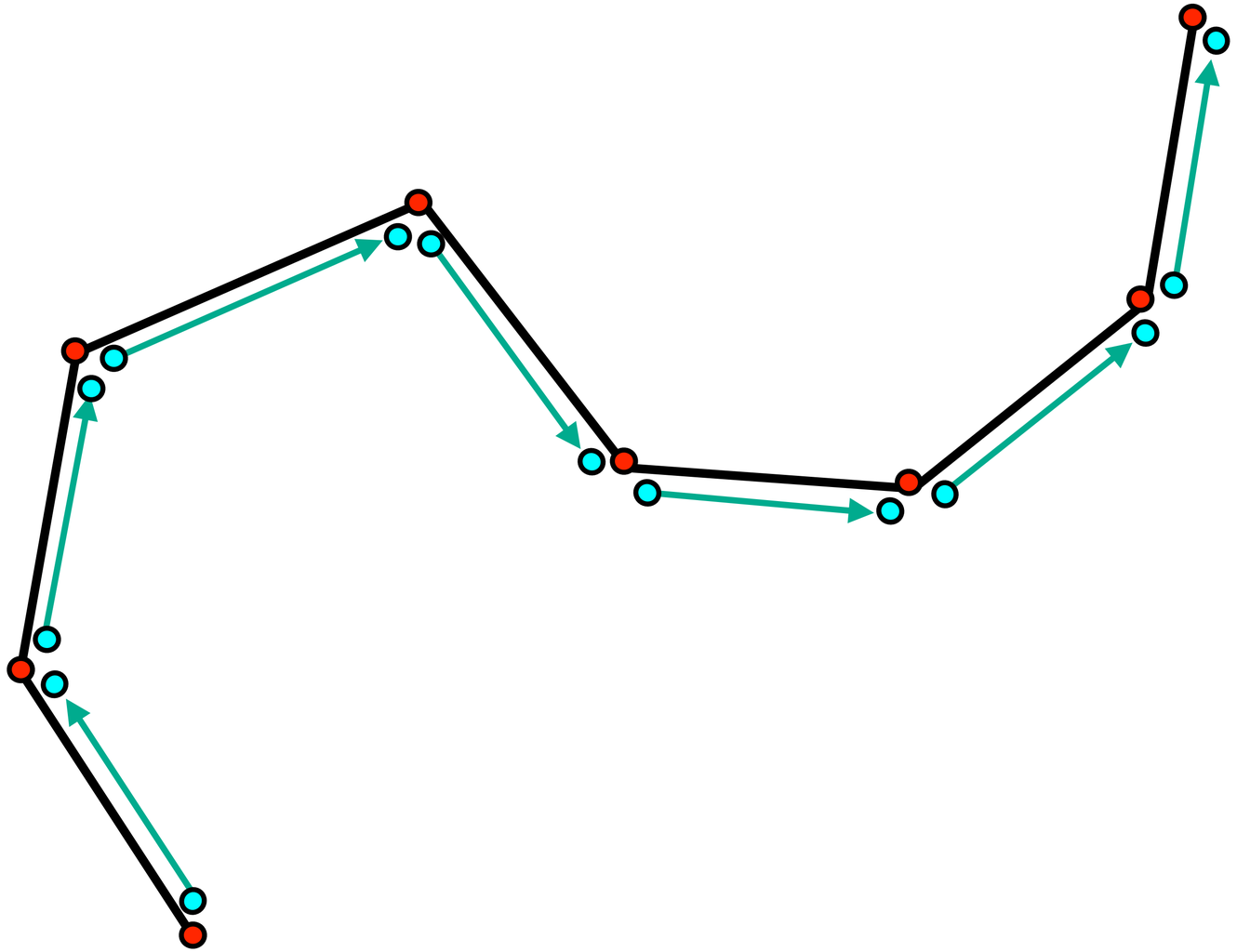


Wir können über folgende Objekte iterieren:

- Faces (Primitives)
- Punkte
- Vertices
- Half-edges sind nicht iterierbar, aber gehören zum source-vertex

Half-Edge Datenstruktur

- Kurven sind “polyline” primitives

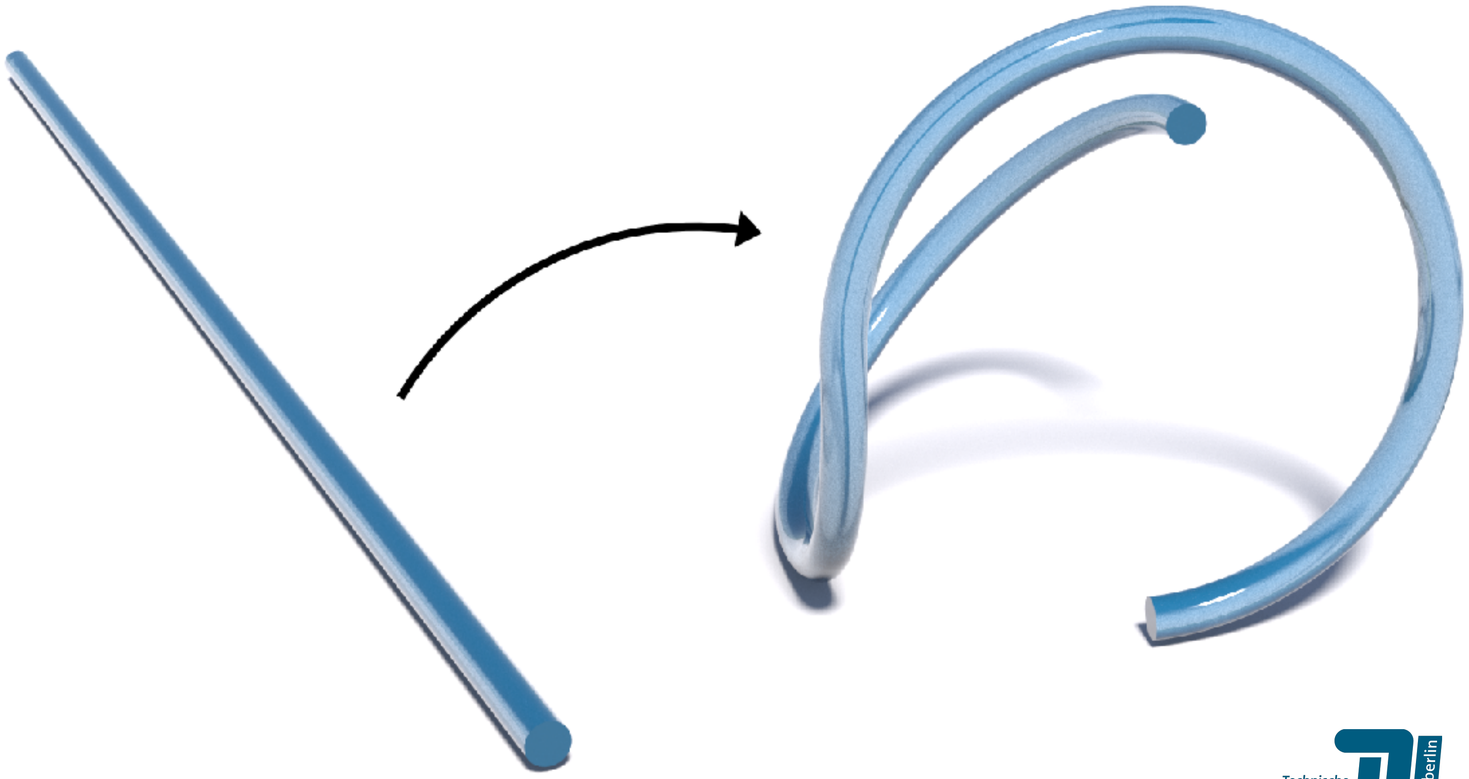


Template Geometrien

Oft starten wir mit dem Erzeuger einer Template Geometrie, z.B.:

- Circle
- Grid
- Line
- Sphere
- Torus
- Tube

→ Später manipulieren wir diese lokal



Template Geometrien

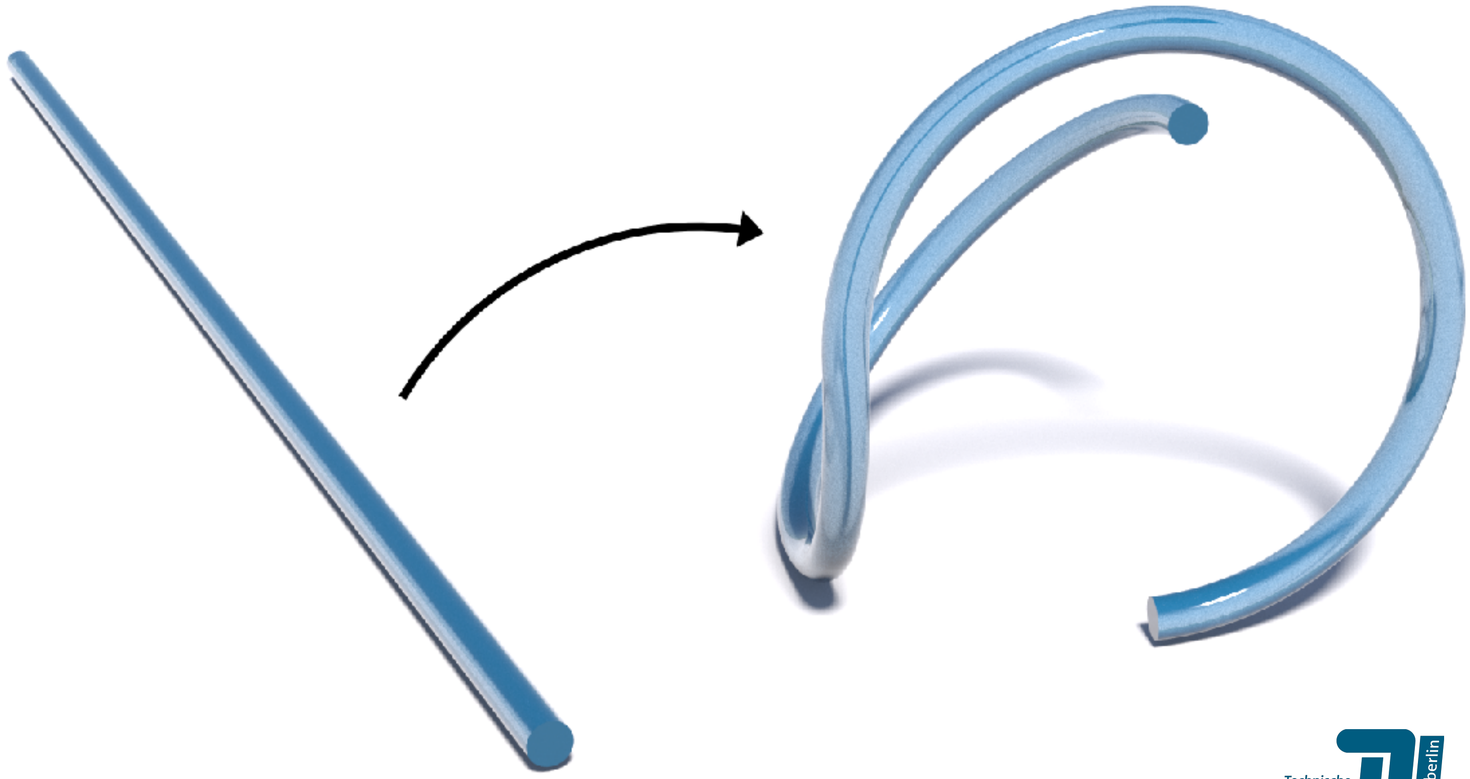
Oft starten wir mit dem Erzeuger einer Template Geometrie, z.B.:

- Circle
- Grid
- Line
- Sphere
- Torus
- Tube

→ Später manipulieren wir diese lokal

Zuerst: ein paar sehr nützliche Knoten

- Resample
- Remesh
- Normal
- Divide
- Polyreduce
- Polywire
- Transform
- Copy Transform
- Copy to Points
- Merge
- ⋮



Template Geometrien

Übung:

a) Erzeuge einen Torus mit Radienverhältnis $\frac{R}{r} = \sqrt{2}$.

b) Nutze einen “Remesh” Knoten um ein Delaunay Mesh zu erzeugen

c) Nutze “Copy to Points” und “Polywire” Knoten für eine “Ball and Stick” Visualisierung

Technische Universität Berlin

Wrangle-Knoten

Berlin Mathematical School BMS

SideFX

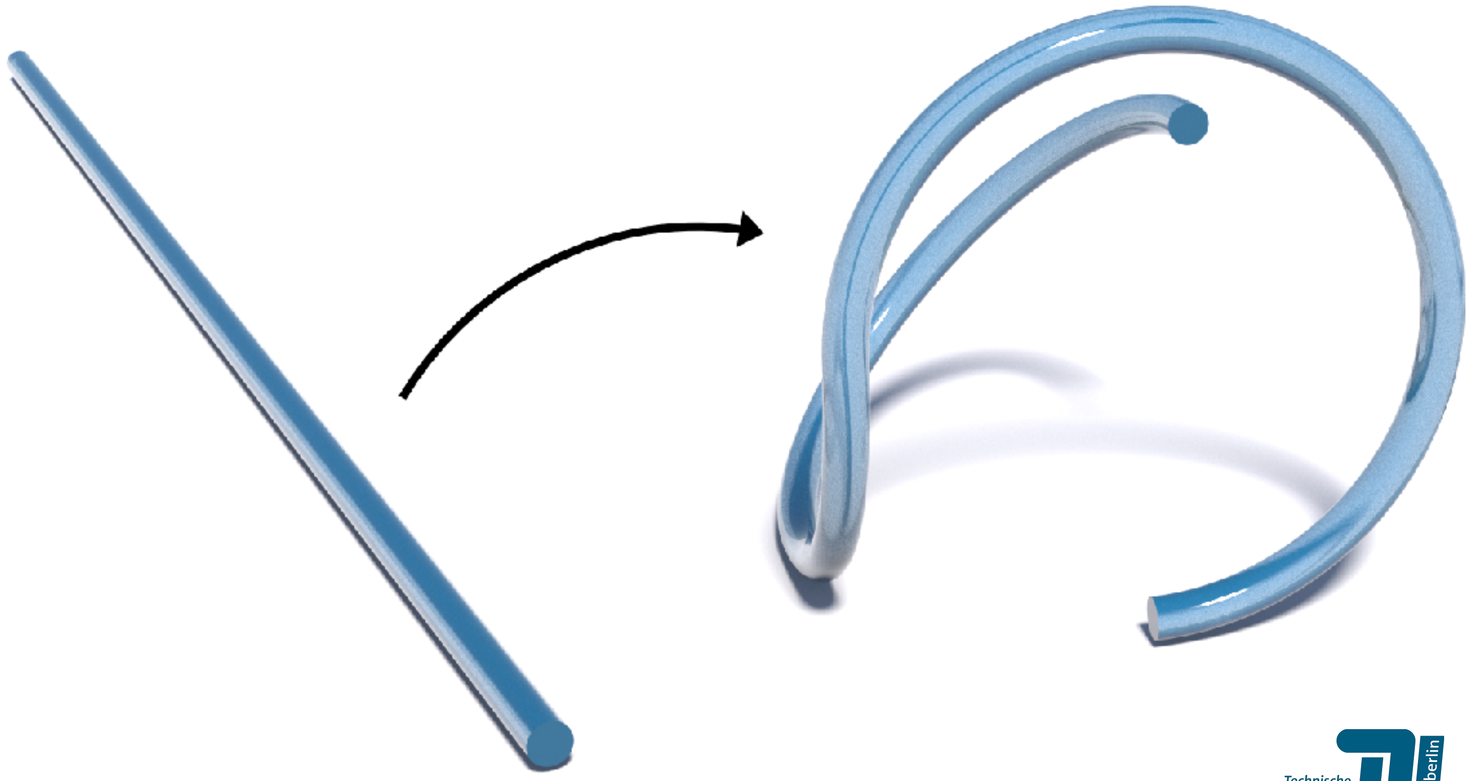
DGD

SFB TRR 109 Discretization in Geometry and Dynamics

Houdini

Der Workflow ist oft nah an “mathematischer Denkweise”

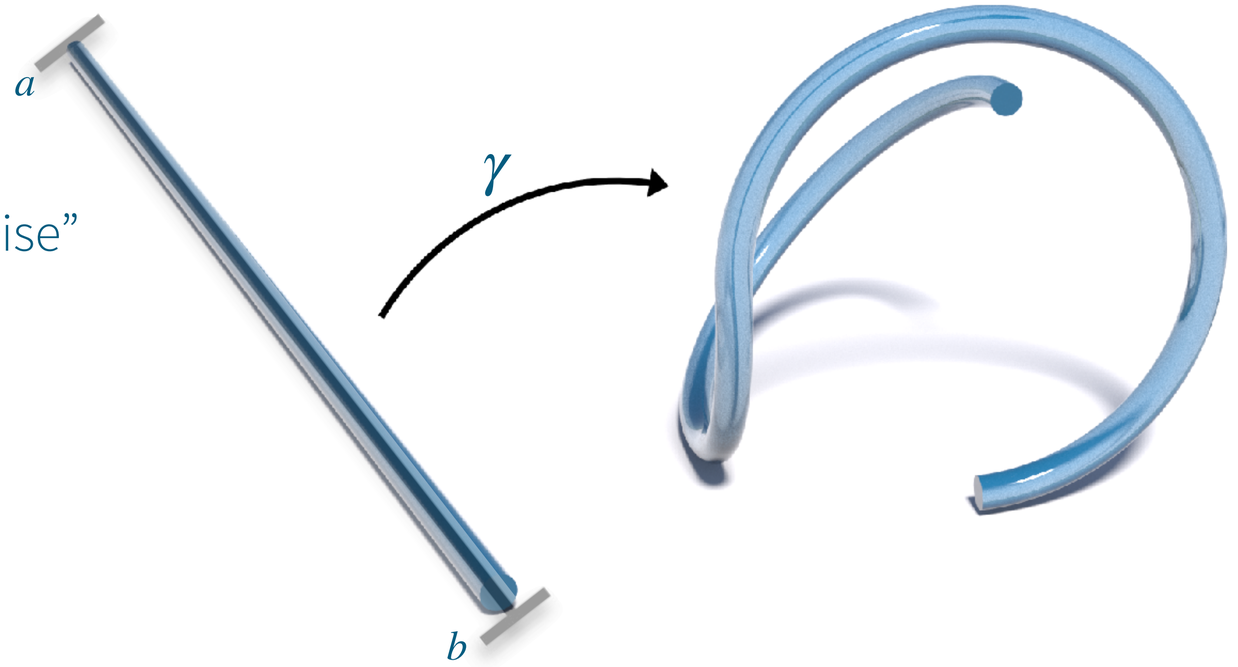
- Erst werden template Geometrien erzeugt
- Abbildungen manipulieren die Geometrie

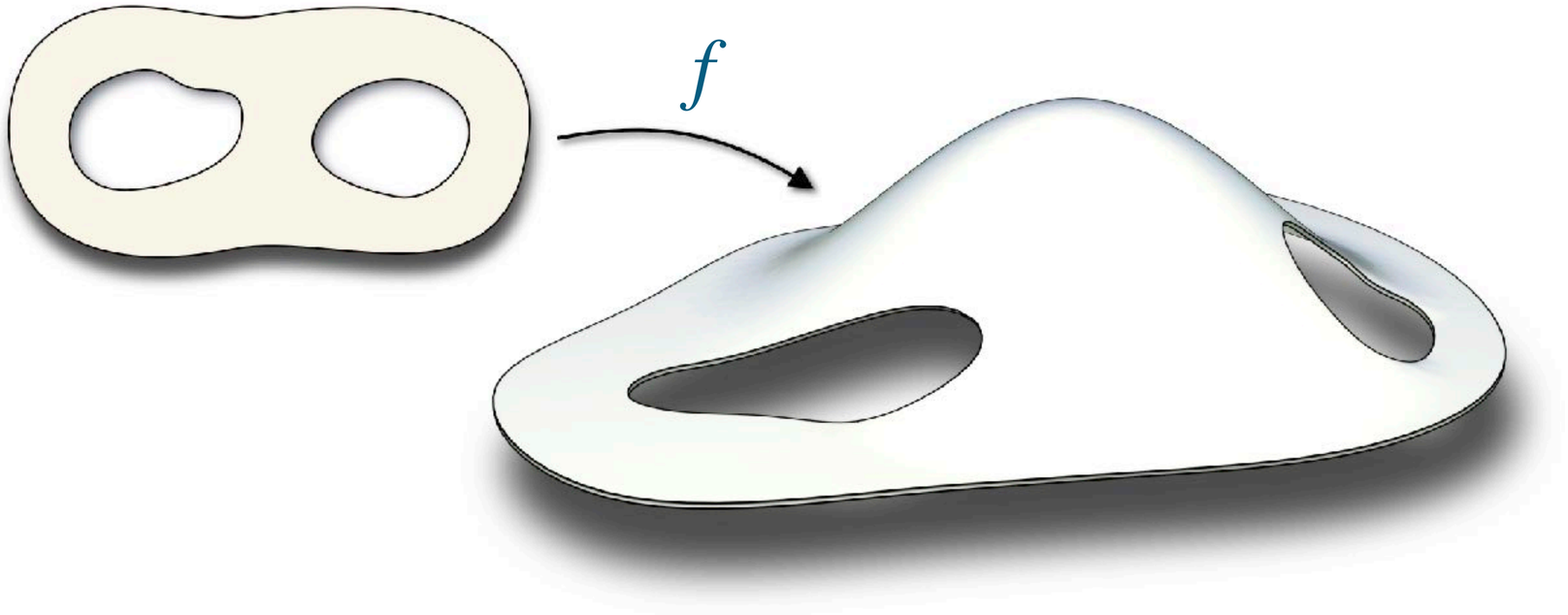


Houdini

Der Workflow ist oft nah an “mathematischer Denkweise”

- Erst werden template Geometrien erzeugt
- Abbildungen manipulieren die Geometrie

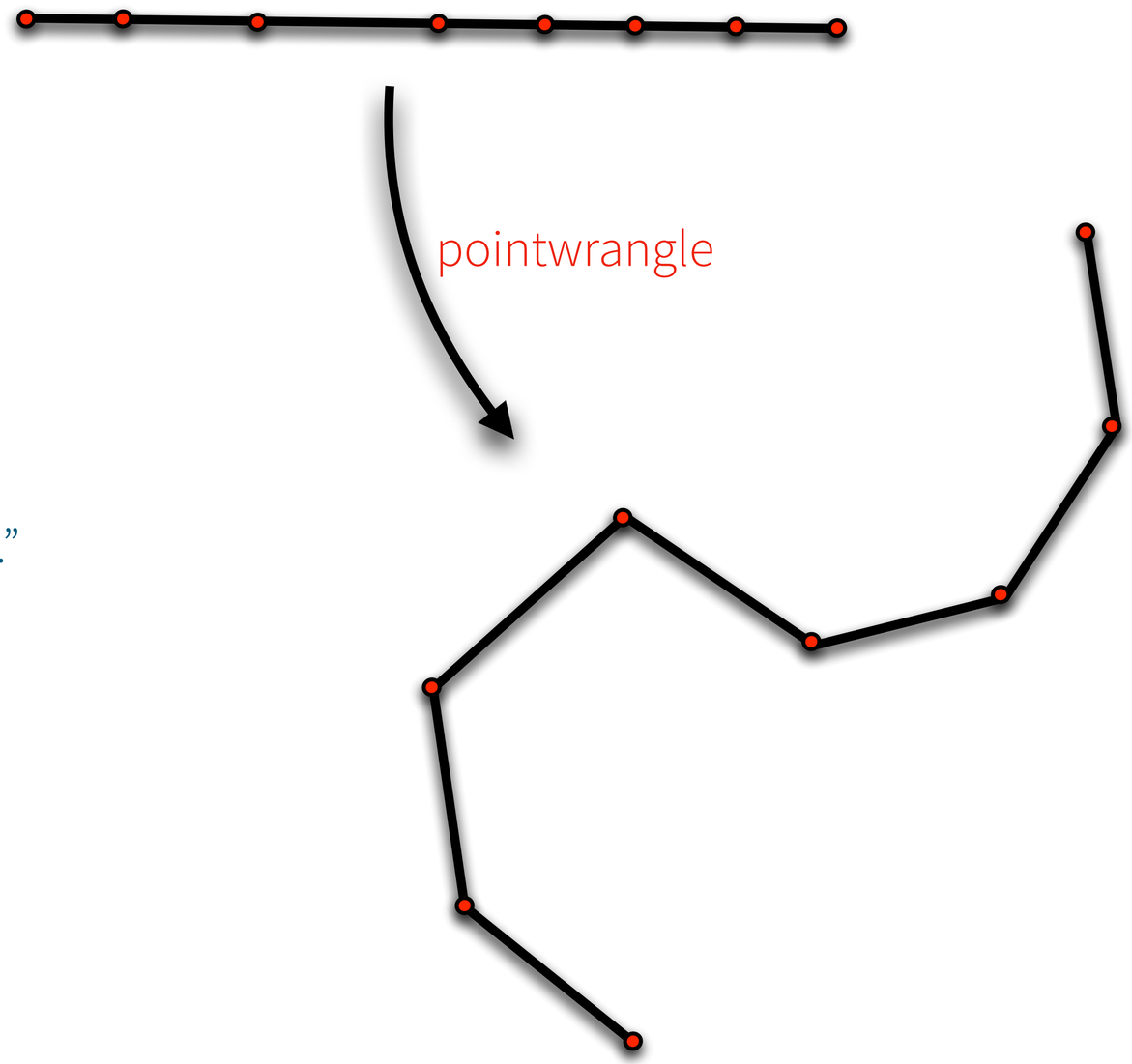




Wrangle Knoten

Mit Hilfe von “Attribute Wrangle” Knoten können wir über Geometrie iterieren

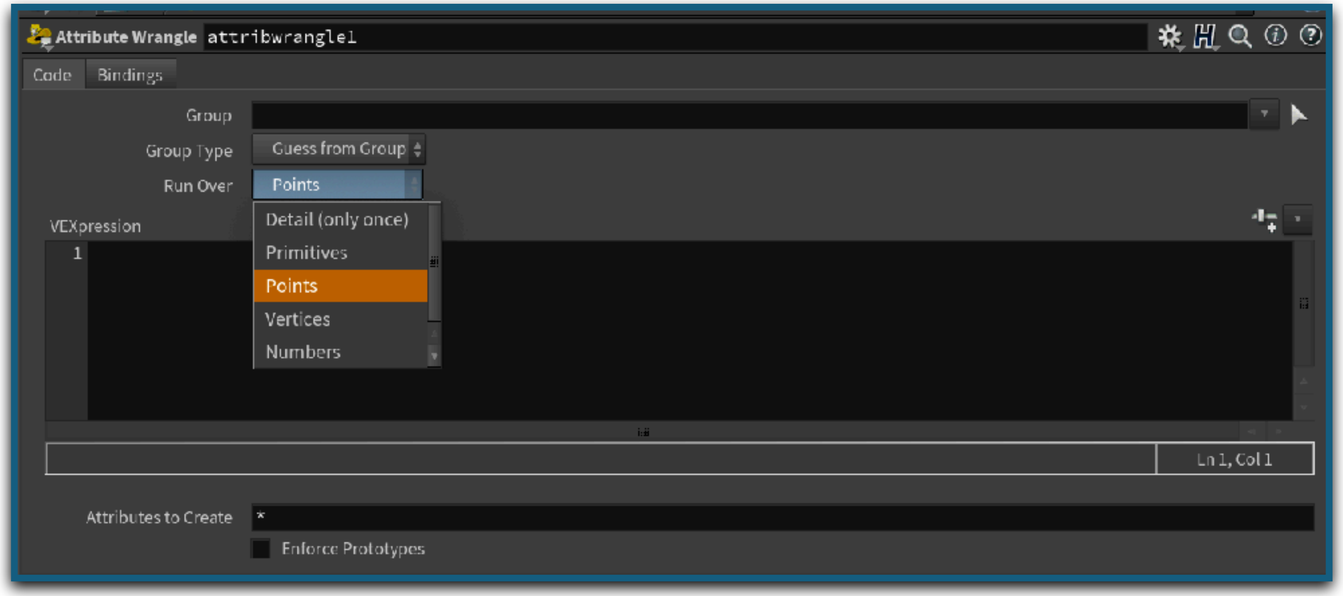
- Detail → globale Operation (kein loop)
- Primitive → “for each primitive in the geometry do: …”
- Point → “for each point in the geometry do: …”
- Vertex → “for each vertex in the geometry do: …”

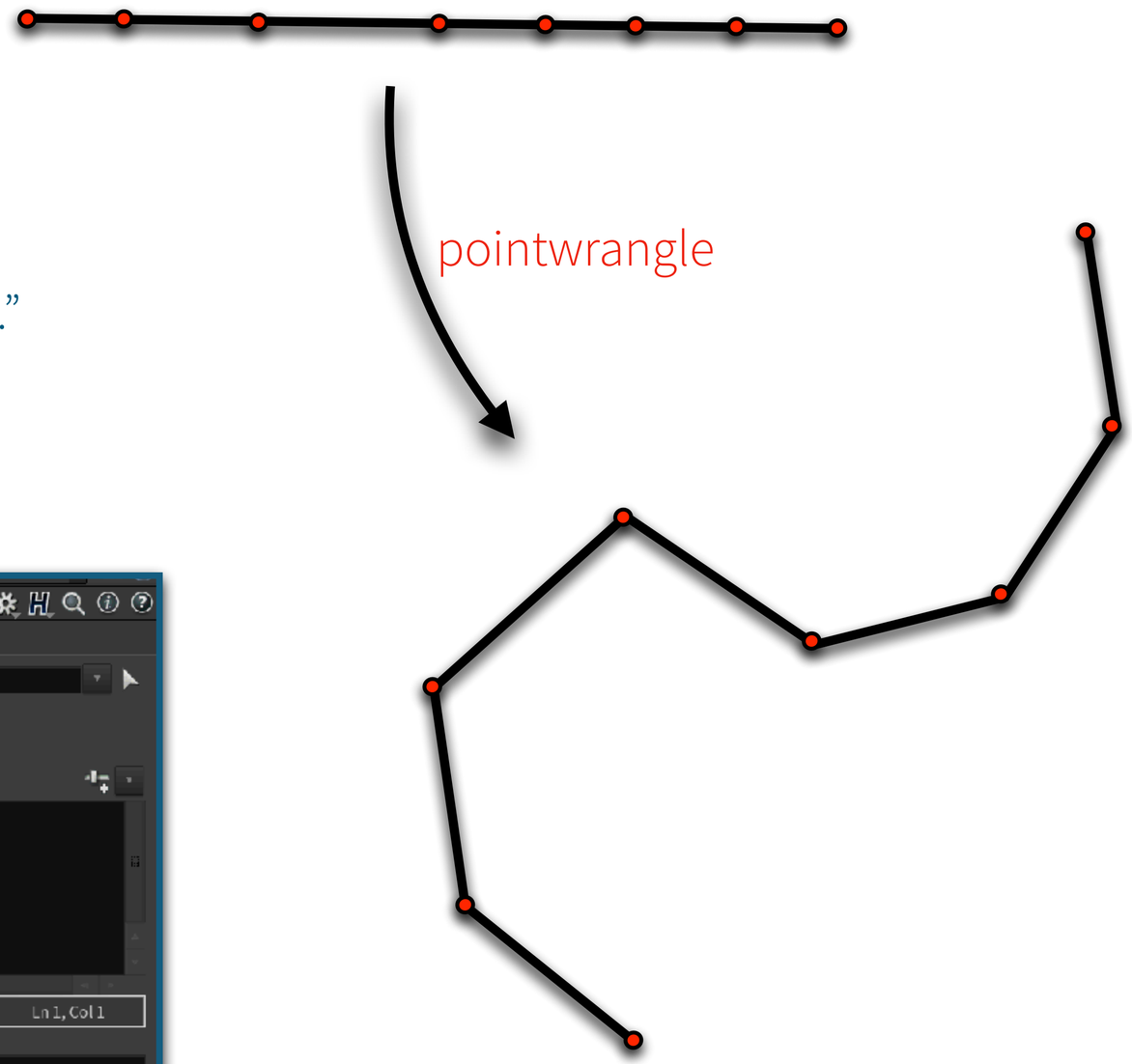


Wrangle Knoten

Mit Hilfe von “Attribute Wrangle” Knoten können wir über Geometrie iterieren

- Detail → globale Operation (kein loop)
- Primitive → “for each primitive in the geometry do: …”
- Point → “for each point in the geometry do: …”
- Vertex → “for each vertex in the geometry do: …”

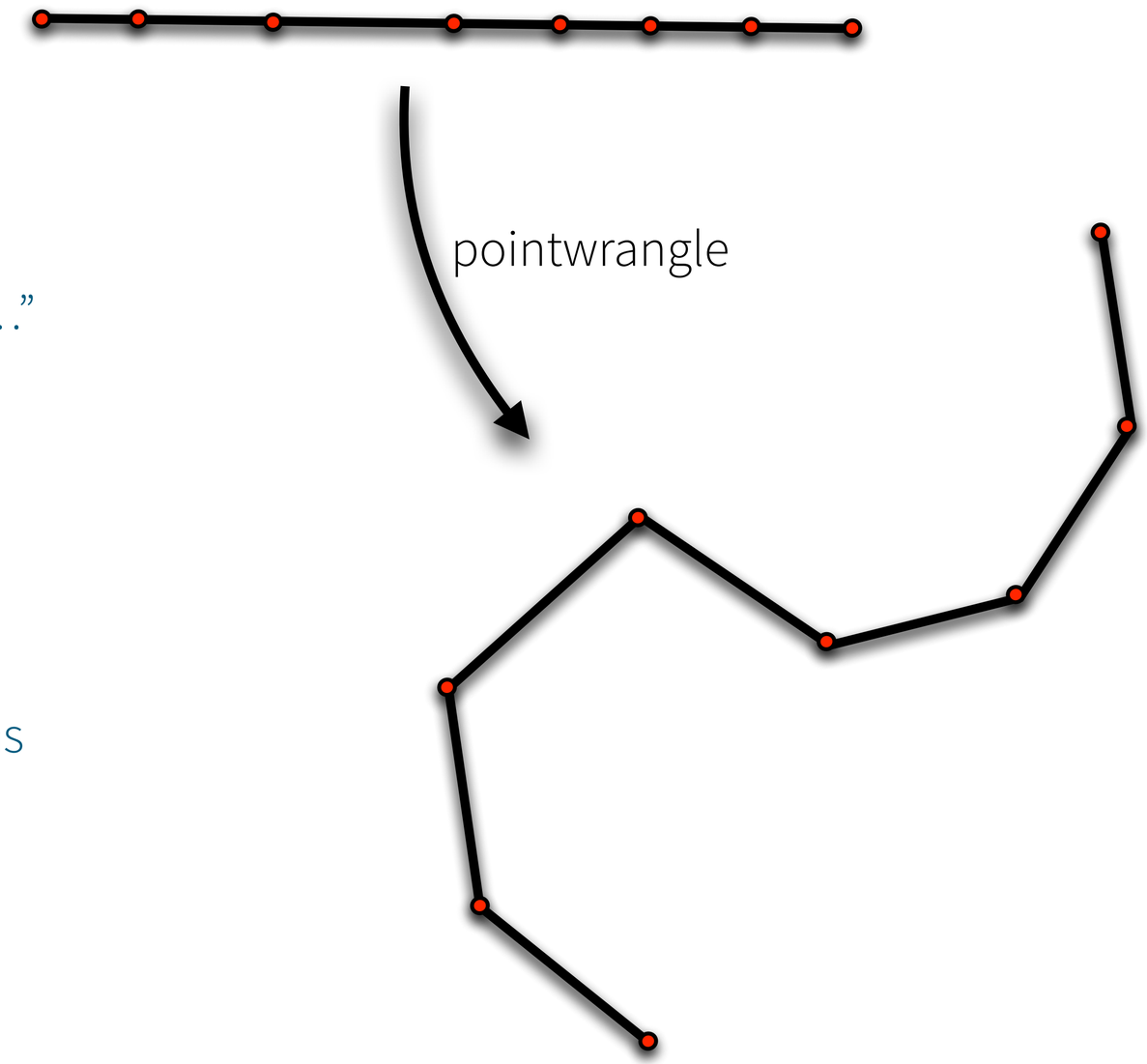




Wrangle Knoten

Mit Hilfe von “Attribute Wrangle” Knoten können wir über Geometrie iterieren

- Detail → globale Operation (kein loop)
- Primitive → “for each primitive in the geometry do: …”
- Point → “for each point in the geometry do: …”
- Vertex → “for each vertex in the geometry do: …”

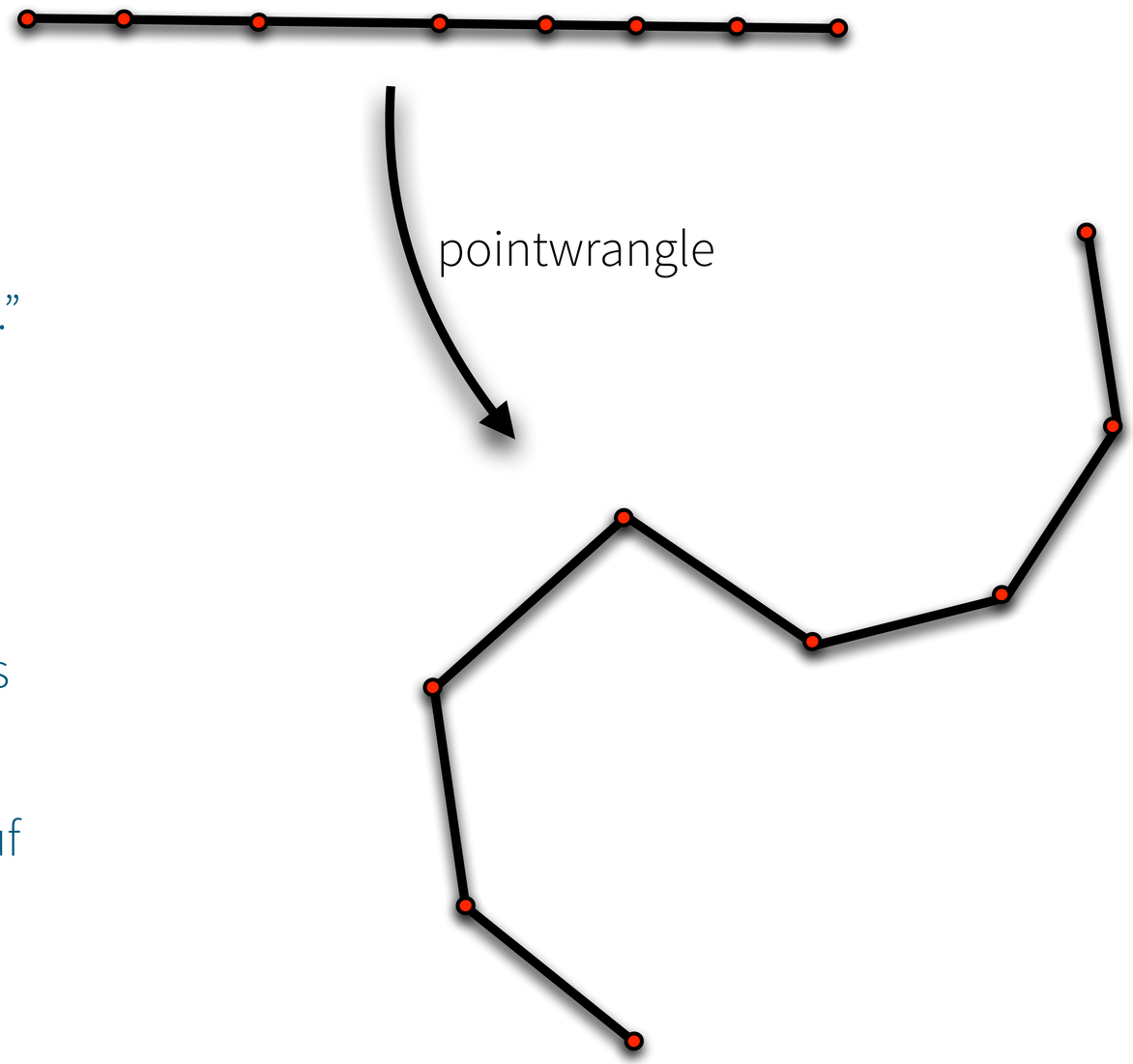


Houdini hat eine eigene Programmiersprache namens *VEX*, die ähnlich wie Java und C++ ist.

Wrangle Knoten

Mit Hilfe von “Attribute Wrangle” Knoten können wir über Geometrie iterieren

- Detail → globale Operation (kein loop)
- Primitive → “for each primitive in the geometry do: …”
- Point → “for each point in the geometry do: …”
- Vertex → “for each vertex in the geometry do: …”



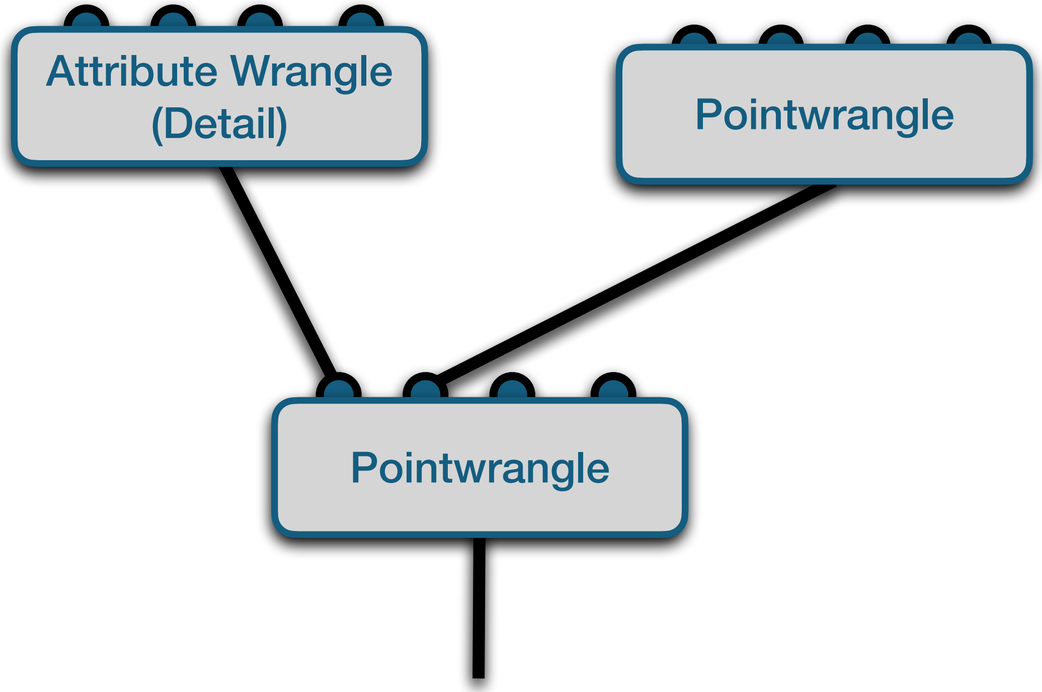
Houdini hat eine eigene Programmiersprache namens *VEX*, die ähnlich wie Java und C++ ist.

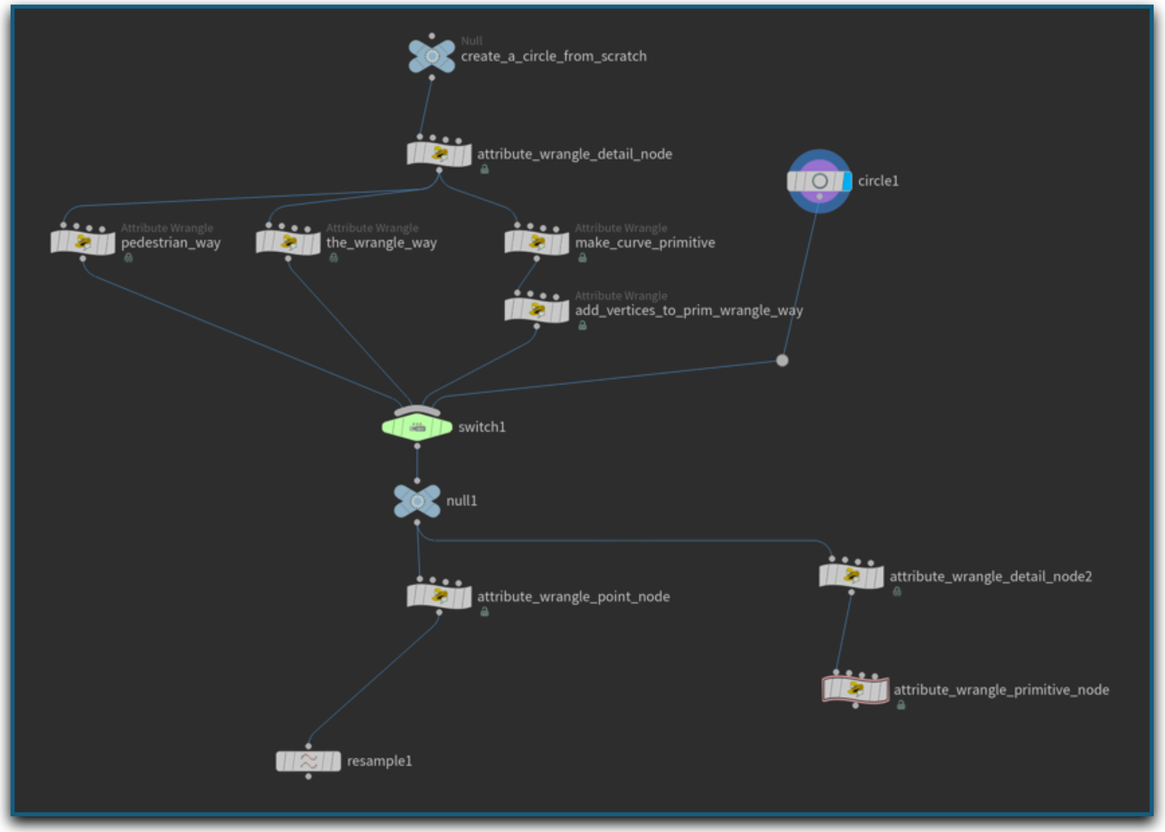
→ für die parallele Verarbeitung von Operationen auf allen Punkten, Primitiven und Vertices optimiert

Wrangle Knoten

Houdini hat eine eigene Programmiersprache namens *VEX*, die ähnlich wie Java und C++ ist.

→ für die parallele Verarbeitung von Operationen auf allen Punkten, Primitiven und Vertices optimiert

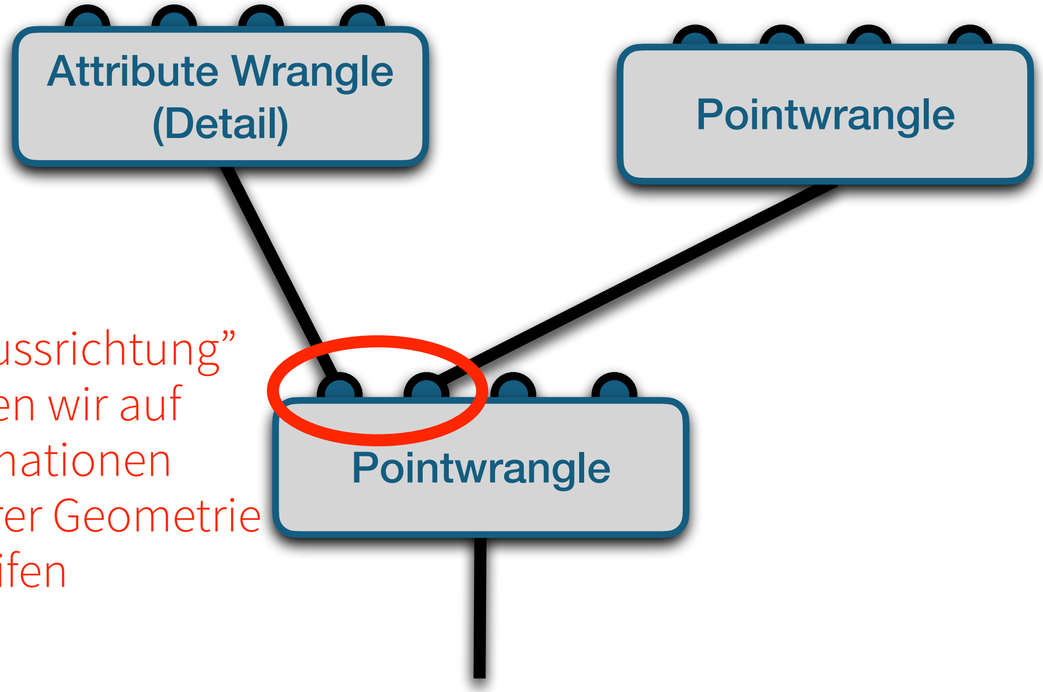




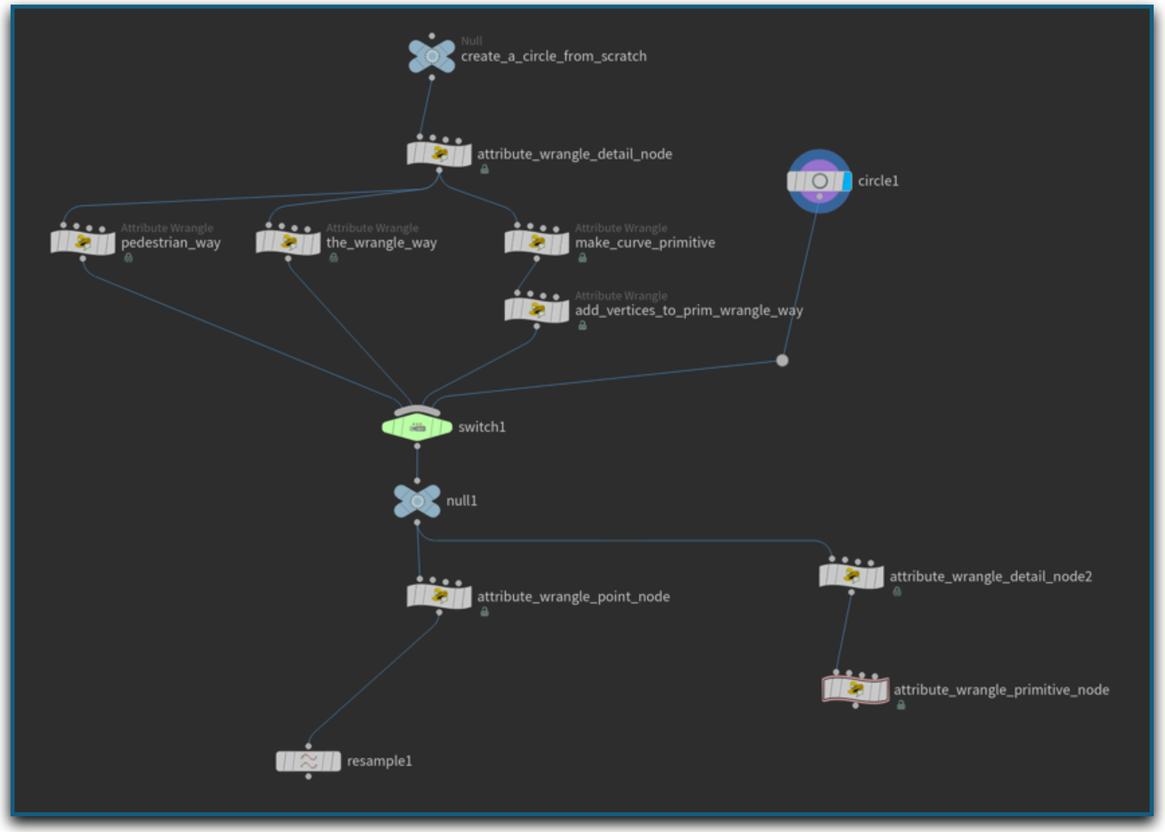
Wrangle Knoten

Houdini hat eine eigene Programmiersprache namens *VEX*, die ähnlich wie Java und C++ ist.

→ für die parallele Verarbeitung von Operationen auf allen Punkten, Primitiven und Vertices optimiert

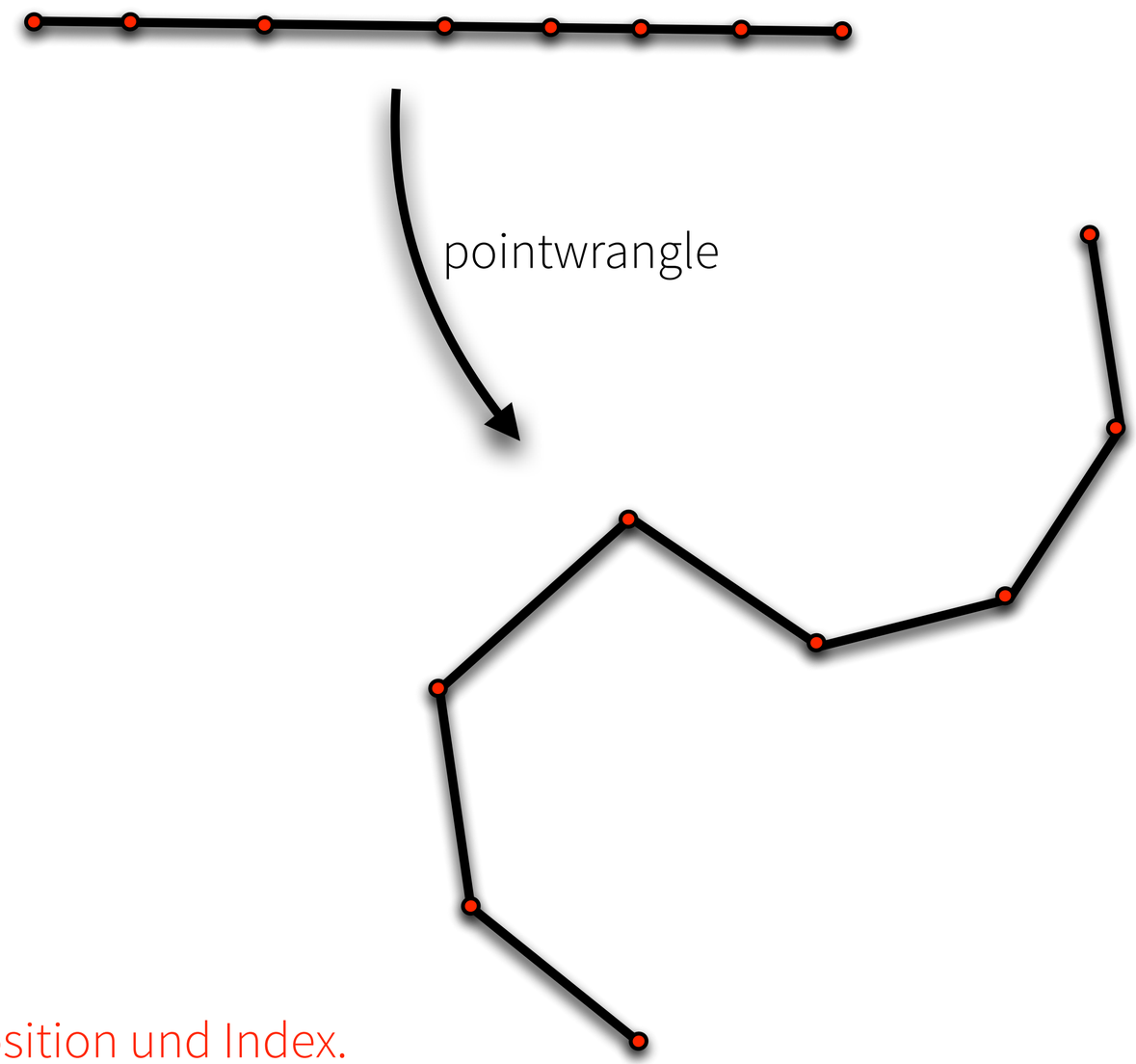


In “Flussrichtung” können wir auf Informationen anderer Geometrie zugreifen



Technische Universität Berlin

Wrangle Knoten



Relevante Informationen werden als Attribute gespeichert, z.B.

- Position **P**
- Normalenvektor **N**
- Farbe **Cd**
- Transparenz **Alpha**
- UV-Texturkoordinate **uv**
- sowie Index, Arrays, Matrizen usw.

Jeder Punkt kennt seine Position und Index.

Wrangle Knoten

Relevante Informationen werden als Attribute gespeichert, z.B.

- Position **P**
- Normalenvektor **N**
- Farbe **Cd**
- Transparenz **Alpha**
- UV-Texturkoordinate **uv**
- sowie Index, Arrays, Matrizen usw.

Attribute werden addressiert über

type@name

Wrangle Knoten

Relevante Informationen werden als Attribute gespeichert, z.B.

- Position **P**
- Normalenvektor **N**
- Farbe **Cd**
- Transparenz **Alpha**
- UV-Texturkoordinate **uv**
- sowie Index, Arrays, Matrizen usw.

Attribute werden addressiert über

type@name

- i@… integer
- f@… float
- u@… 2D vector
- v@… 3D vector
- p@… 4D vector
- 2@… 2D matrix
- 3@… 3D matrix
- 4@… 4D matrix
- s@… string

Wrangle Knoten

Relevante Informationen werden als Attribute gespeichert, z.B.

- Position **P**
- Normalenvektor **N**
- Farbe **Cd**
- Transparenz **Alpha**
- UV-Texturkoordinate **uv**
- sowie Index, Arrays, Matrizen usw.

Attribute werden addressiert über

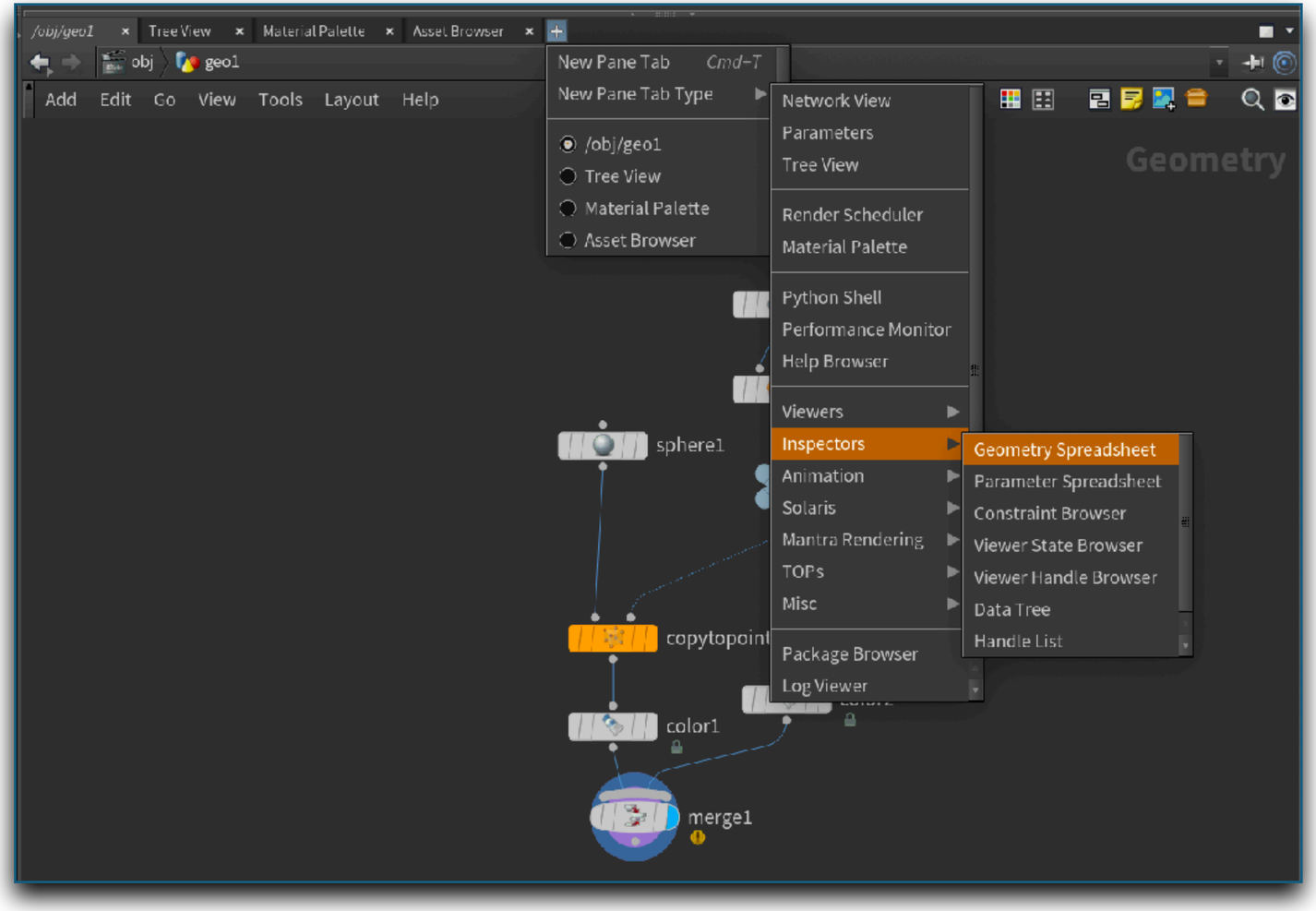
type@name

- i@… integer
- f@… float
- u@… 2D vector
- v@… 3D vector
- p@… 4D vector
- 2@… 2D matrix
- 3@… 3D matrix
- 4@… 4D matrix
- s@… string

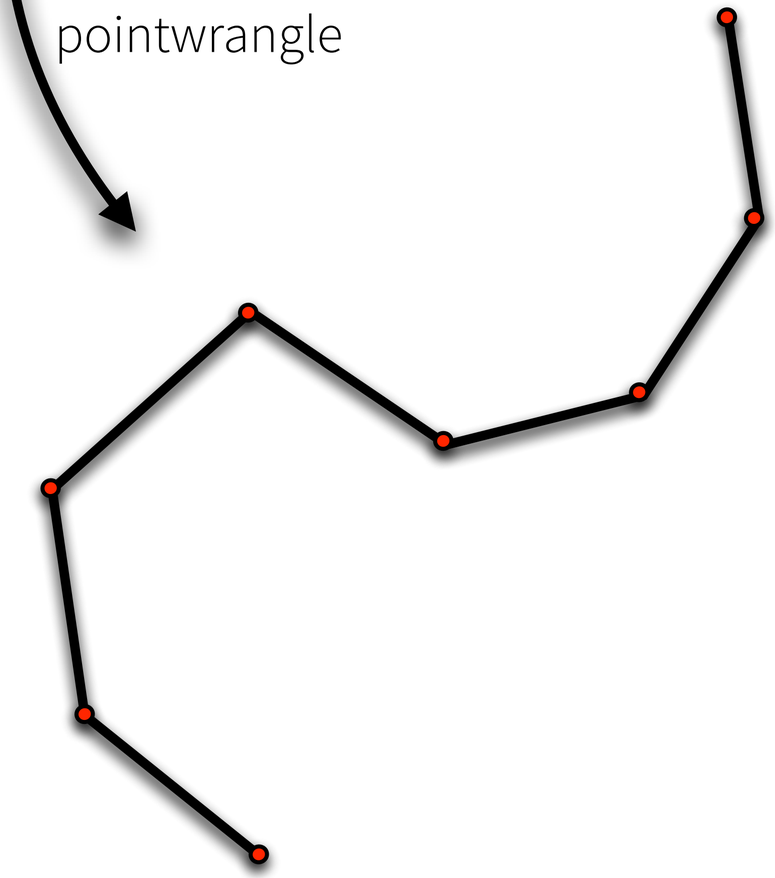
→ array Version über
type[]@name

Wrangle Knoten

Welche Attribute existieren und welche Werte sie haben könnt ihr im “Geometry Spreadsheet” ansehen



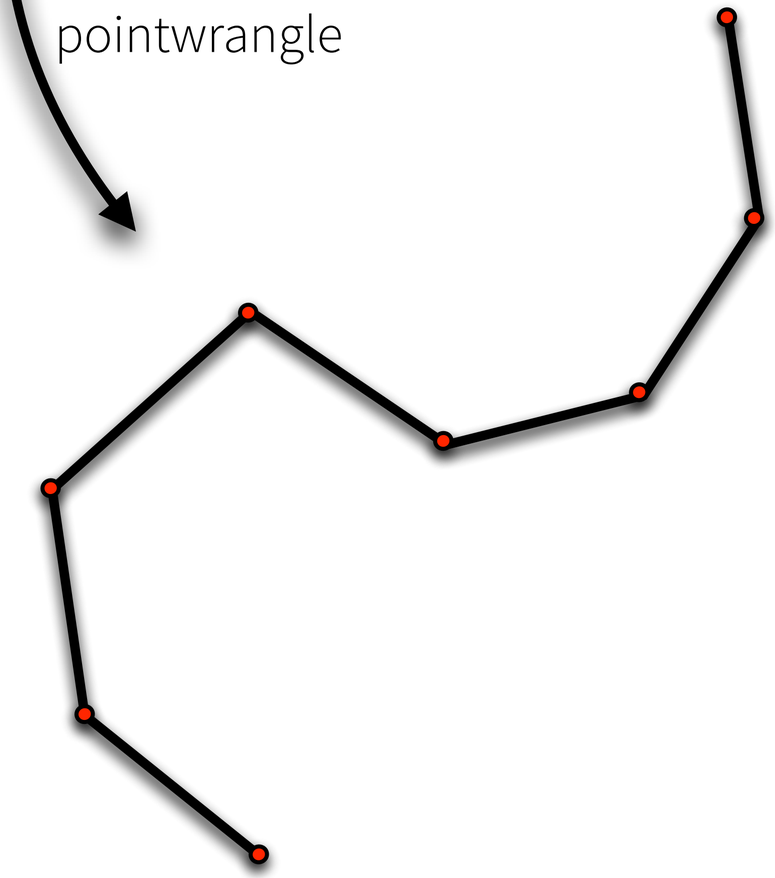
pointwrangle



Wrangle Knoten

Welche Attribute existieren und welche Werte sie haben könnt ihr im “Geometry Spreadsheet” ansehen

pointwrangle



/obj/geo1 | Tree View | Material Palette | Asset Browser | Geometry Spreadsheet

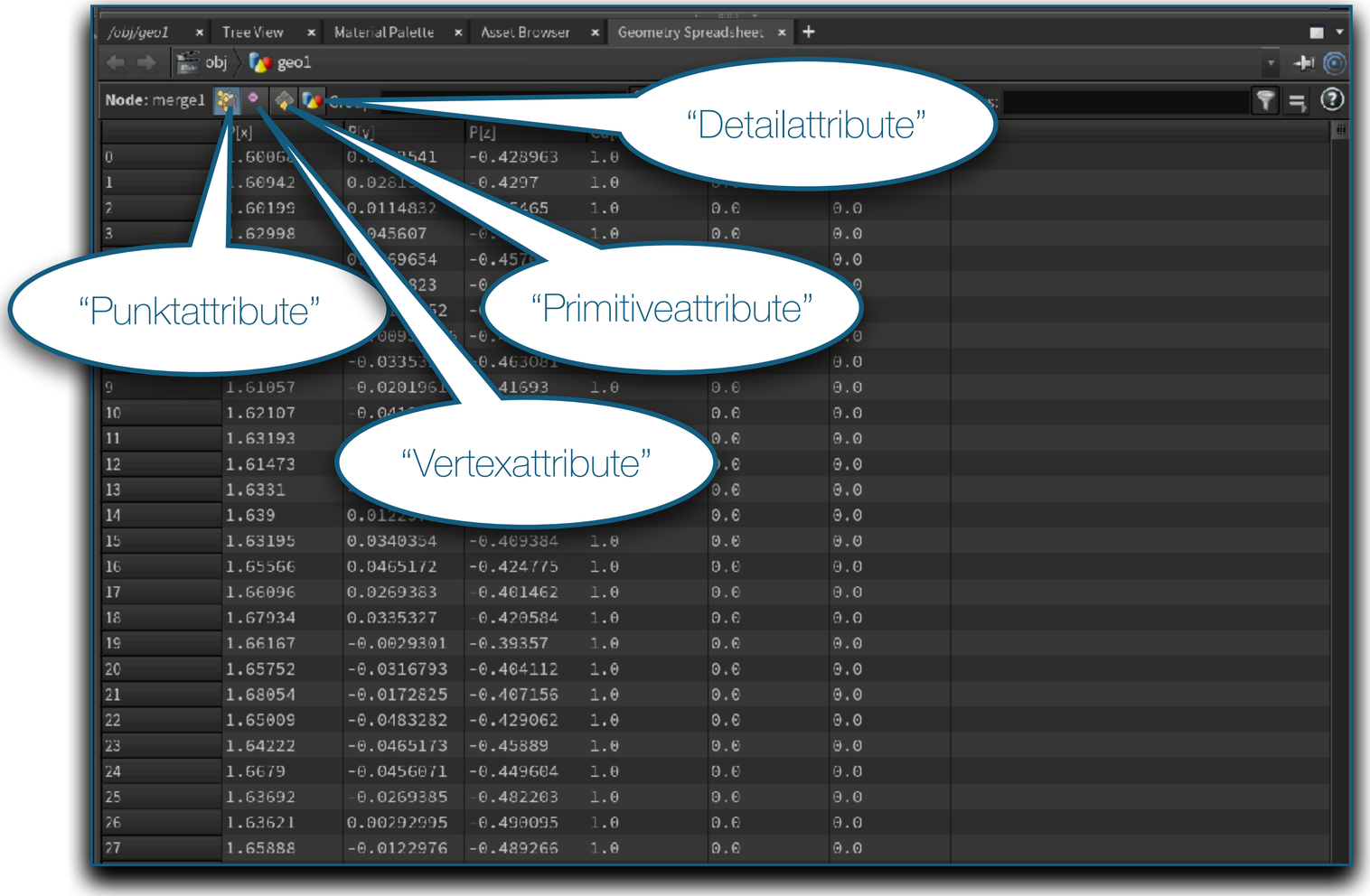
obj / geo1

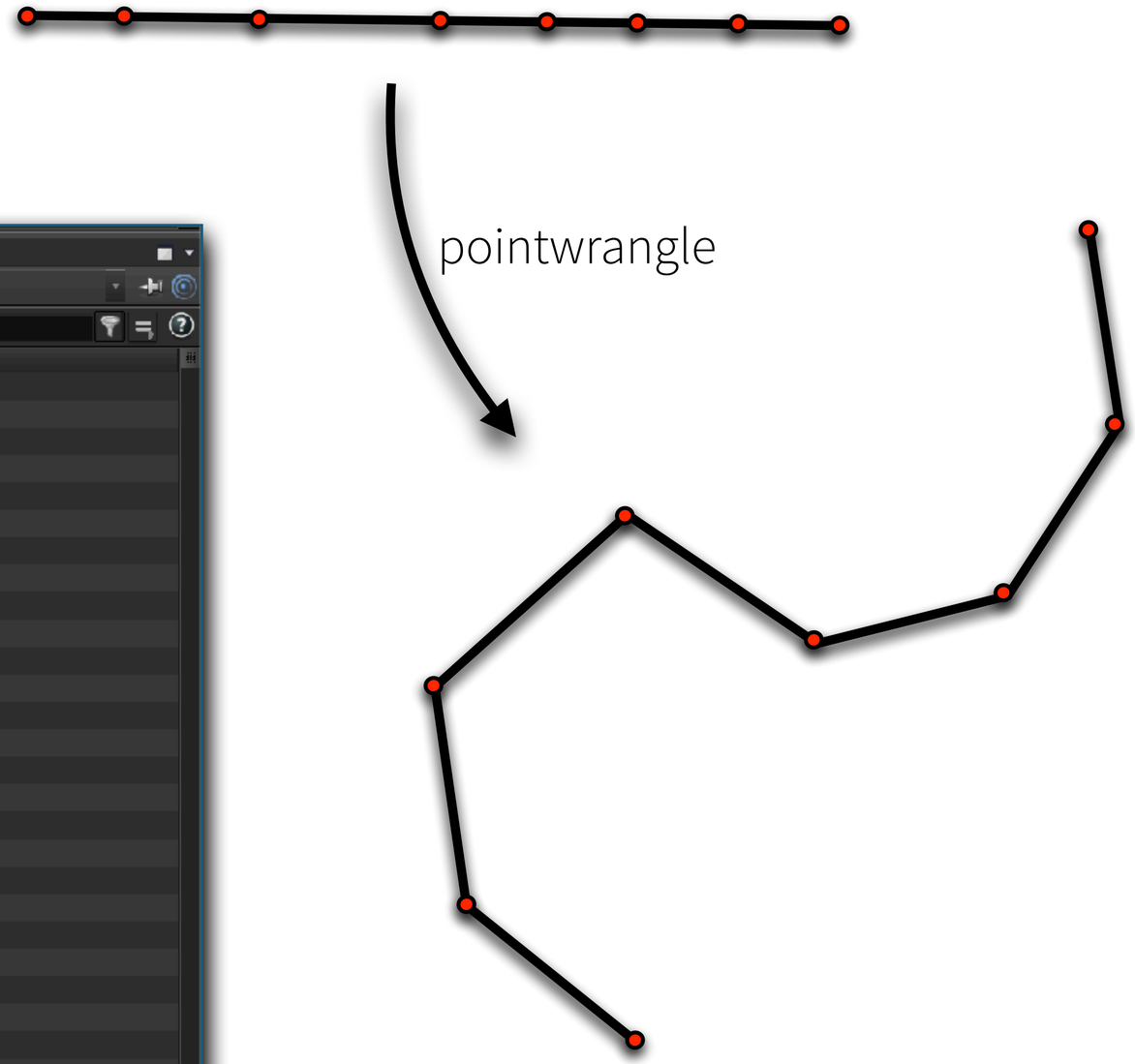
Node: merge1 Group: View Intrinsics Attributes:

	P[x]	P[y]	P[z]	Cd[r]	Cd[g]	Cd[b]
0	1.60068	0.0022541	-0.428963	1.0	0.0	0.0
1	1.60942	0.0281321	-0.4297	1.0	0.0	0.0
2	1.60199	0.0114832	-0.45465	1.0	0.0	0.0
3	1.62998	0.045607	-0.434061	1.0	0.0	0.0
4	1.61922	0.0369654	-0.457646	1.0	0.0	0.0
5	1.61733	0.0172823	-0.476509	1.0	0.0	0.0
6	1.6627	-0.0183852	-0.446757	1.0	0.0	0.0
7	1.61249	-0.00955185	-0.474704	1.0	0.0	0.0
8	1.61854	-0.0335329	-0.463081	1.0	0.0	0.0
9	1.61057	-0.0201961	-0.41693	1.0	0.0	0.0
10	1.62107	-0.0412311	-0.436984	1.0	0.0	0.0
11	1.63193	-0.0366137	-0.412335	1.0	0.0	0.0
12	1.61473	0.0085532	-0.406387	1.0	0.0	0.0
13	1.6331	-0.0142927	-0.396614	1.0	0.0	0.0
14	1.639	0.0122975	-0.394399	1.0	0.0	0.0
15	1.63195	0.0340354	-0.409384	1.0	0.0	0.0
16	1.65566	0.0465172	-0.424775	1.0	0.0	0.0
17	1.66096	0.0269383	-0.401462	1.0	0.0	0.0
18	1.67034	0.0335327	-0.420584	1.0	0.0	0.0
19	1.66167	-0.0029301	-0.39357	1.0	0.0	0.0
20	1.65752	-0.0316793	-0.404112	1.0	0.0	0.0
21	1.68054	-0.0172825	-0.407156	1.0	0.0	0.0
22	1.65009	-0.0483282	-0.429062	1.0	0.0	0.0
23	1.64222	-0.0465173	-0.45889	1.0	0.0	0.0
24	1.6679	-0.0456071	-0.449604	1.0	0.0	0.0
25	1.63692	-0.0269385	-0.482203	1.0	0.0	0.0
26	1.63621	0.00292995	-0.490095	1.0	0.0	0.0
27	1.65888	-0.0122976	-0.489266	1.0	0.0	0.0

Wrangle Knoten

Welche Attribute existieren und welche Werte sie haben könnt ihr im “Geometry Spreadsheet” ansehen





Technische Universität Berlin

Wrangle Knoten

Übung:

a) Erstelle die Geometrie eines “diskreten” Kreises mit 100 Punkten

   (1) “Per Hand” mit jeder Kante als einzelne Kurve

   (2) Mit einem pointwrangle und jeder Kante als einzelne Kurve

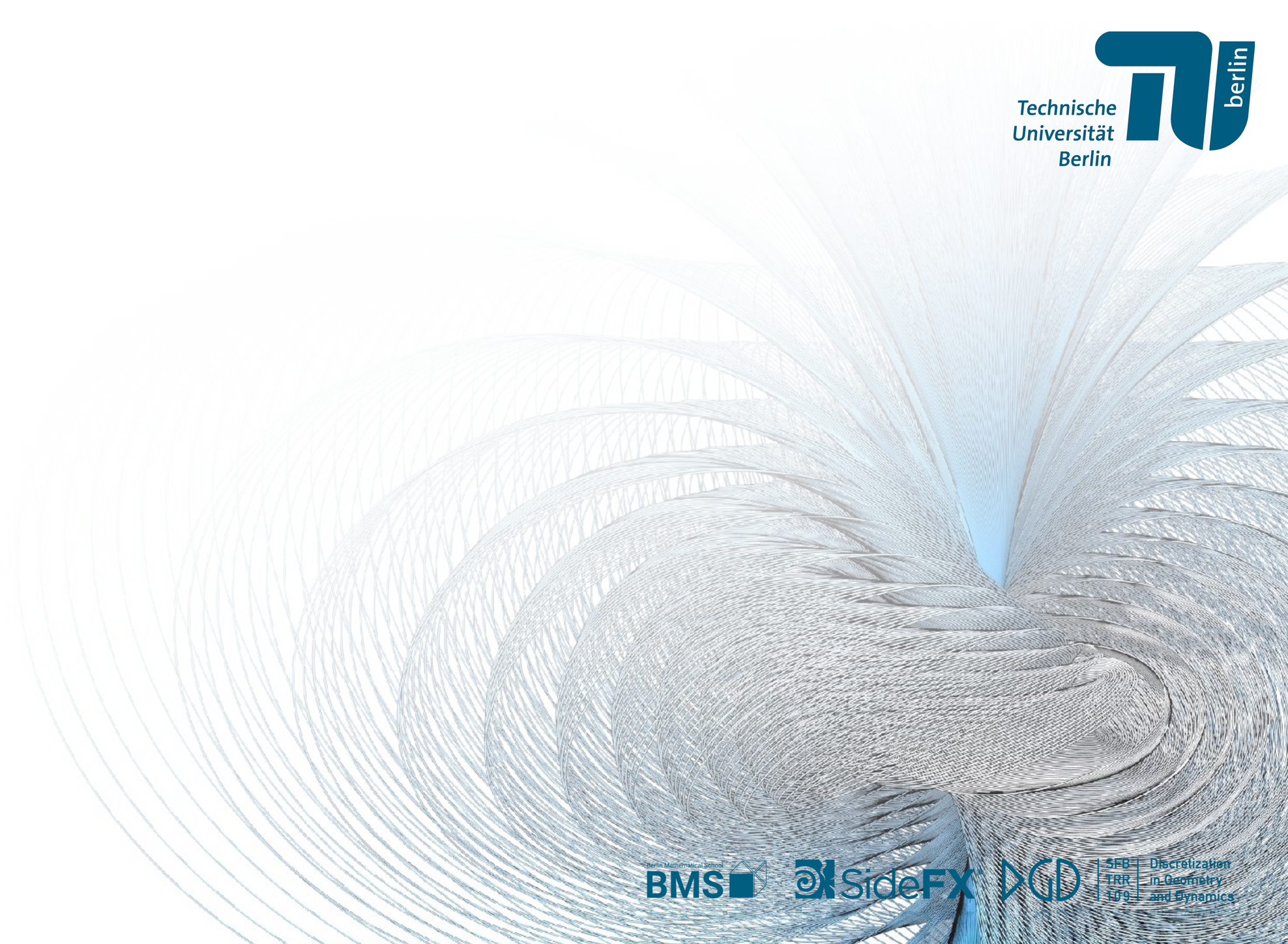
   (3) “Per Hand” mit einer *polyline* welche alle Vertices enthält

   (4) Als template Geometrie

b) Bilde den Kreis über eine Abbildung auf eine andere Form ab

c) Nutze einen *Resample* Knoten um die Kurve zu reparametrisieren

Pause



Technische Universität Berlin

Berlin Mathematical School BMS

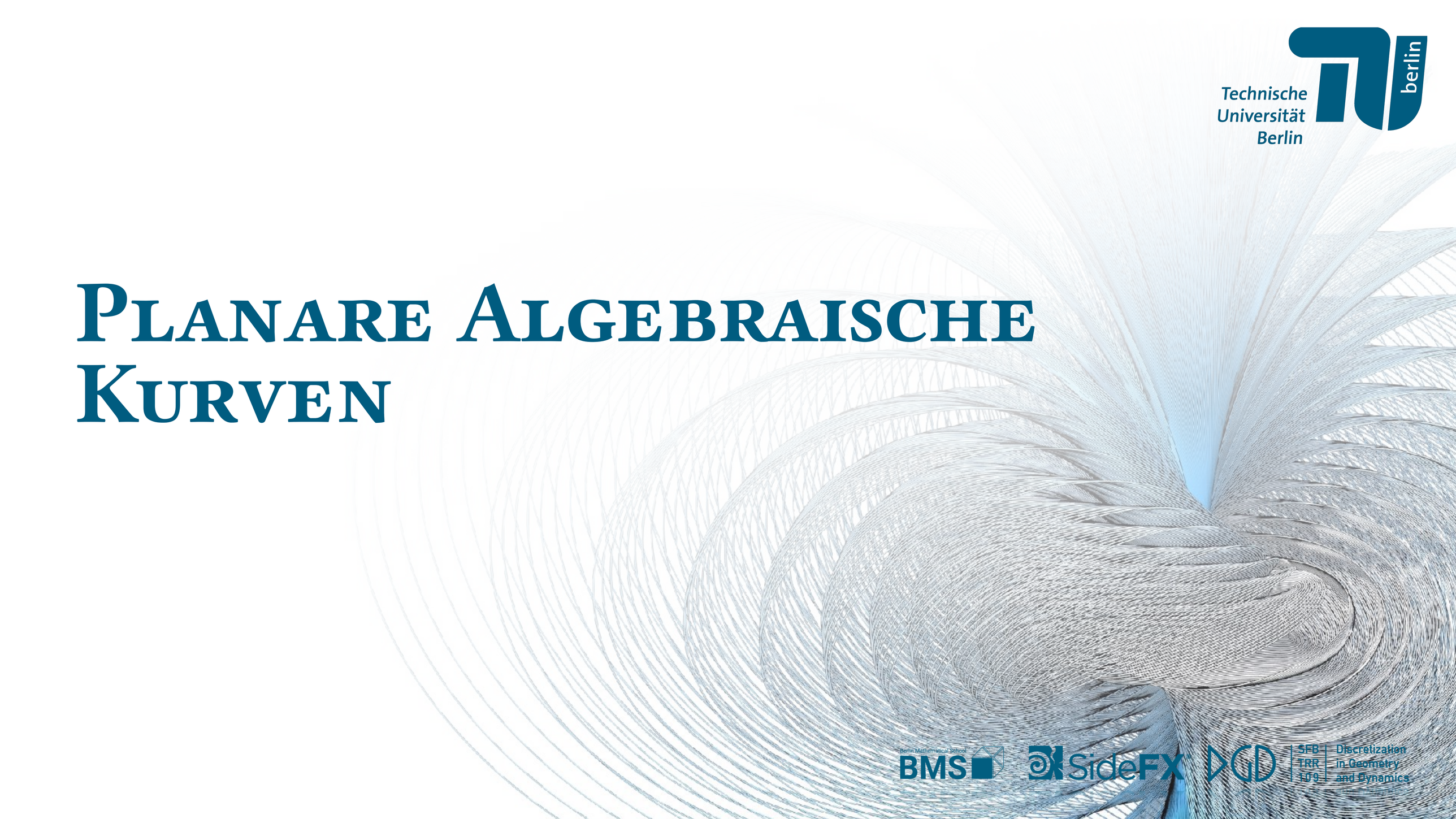
SideFX

DGD

SFB TRR 109 Discretization in Geometry and Dynamics

Technische Universität Berlin

Planare Algebraische Kurven



Berlin Mathematical School BMS

SideFX

DGD

SFB TRR 109 Discretization in Geometry and Dynamics

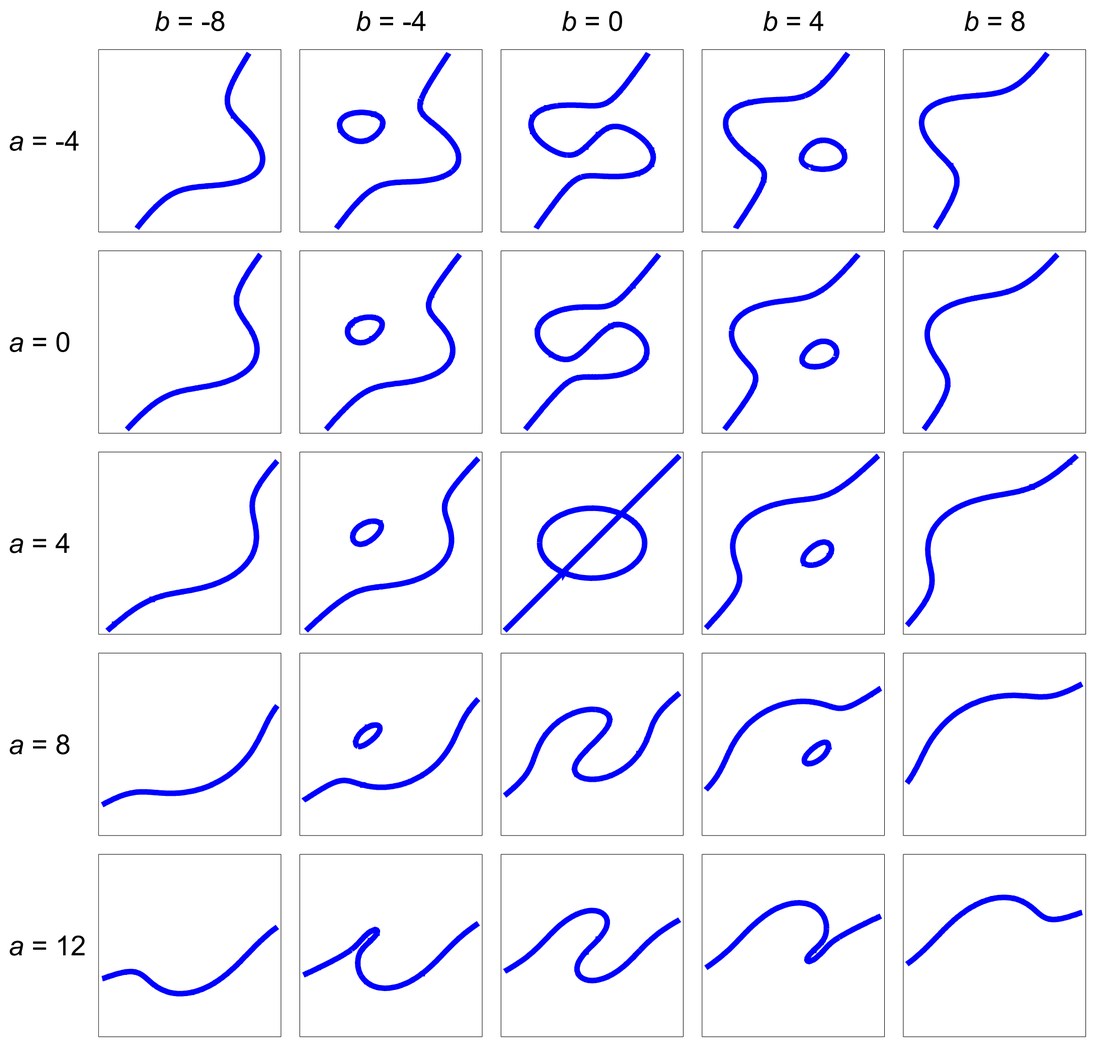
Planare Algebraische Kurven

Planare kubische Kurven sind definiert als die Menge aller Nullstlellen einer kubischen Gleichung

$$F(x, y, z) = 0.$$

In homogenen Kooridinaten $(x : y : z)$, d.h., eines einer Linearkombination aus dem Monomen

$$x^3, y^3, z^3, x^2y, x^2z, y^2x, y^2z, z^2x, z^2y, xyz.$$



Planare Algebraische Kurven

Planare kubische Kurven sind definiert als die Menge aller Nullstlellen einer kubischen Gleichung

$$F(x, y, z) = 0.$$

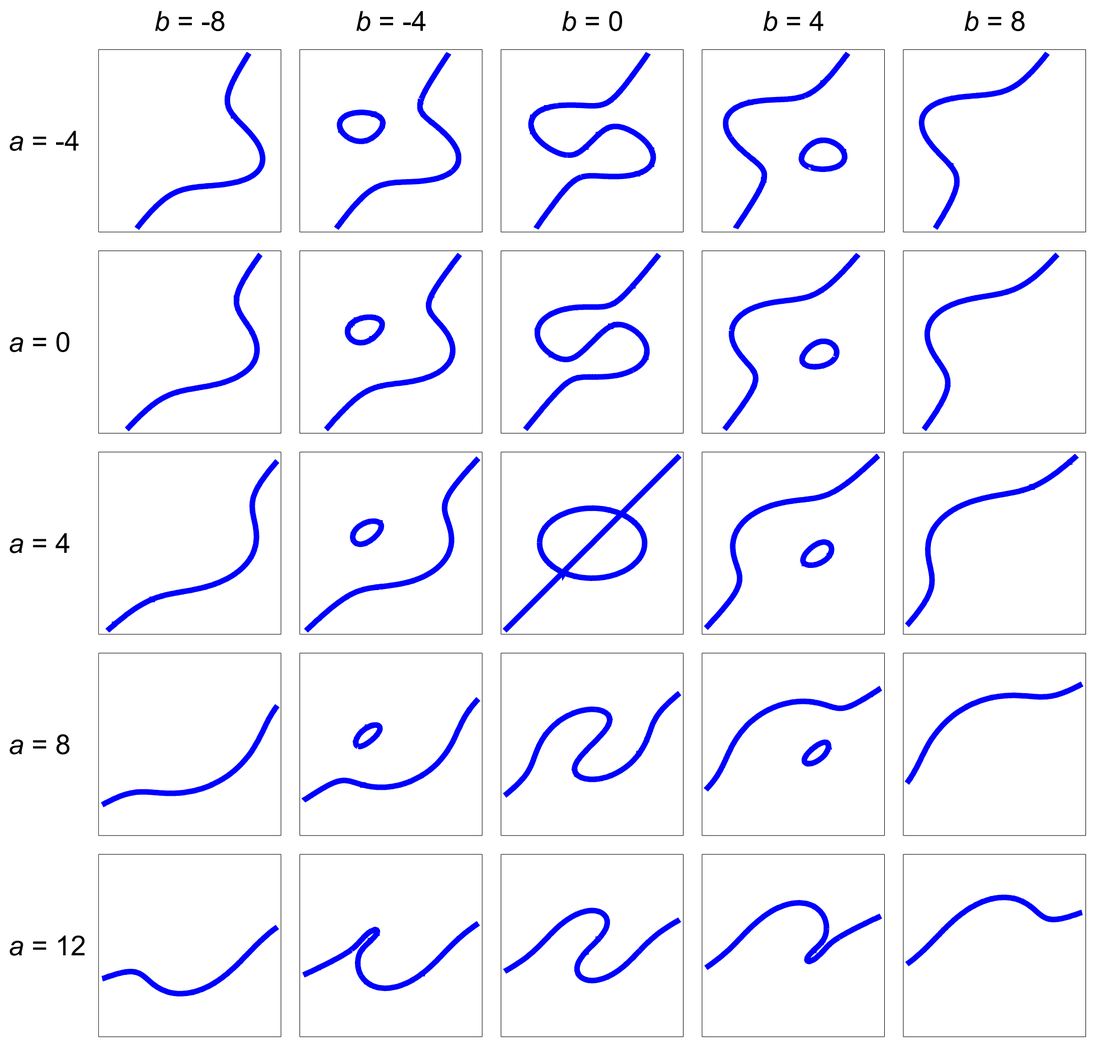
In homogenen Kooridinaten $(x : y : z)$, d.h., eines einer Linearkombination aus dem Monomen

$$x^3, y^3, z^3, x^2y, x^2z, y^2x, y^2z, z^2x, z^2y, xyz.$$

Für $\lambda \neq 0$ gilt also

$$F(\lambda x, \lambda y, \lambda z) = \lambda^3 F(x, y, z).$$

→ Wir normalisieren $z = 1$



Planare Algebraische Kurven

Planare kubische Kurven sind definiert als die Menge aller Nullstlellen einer kubischen Gleichung

$$F(x, y, z) = 0.$$

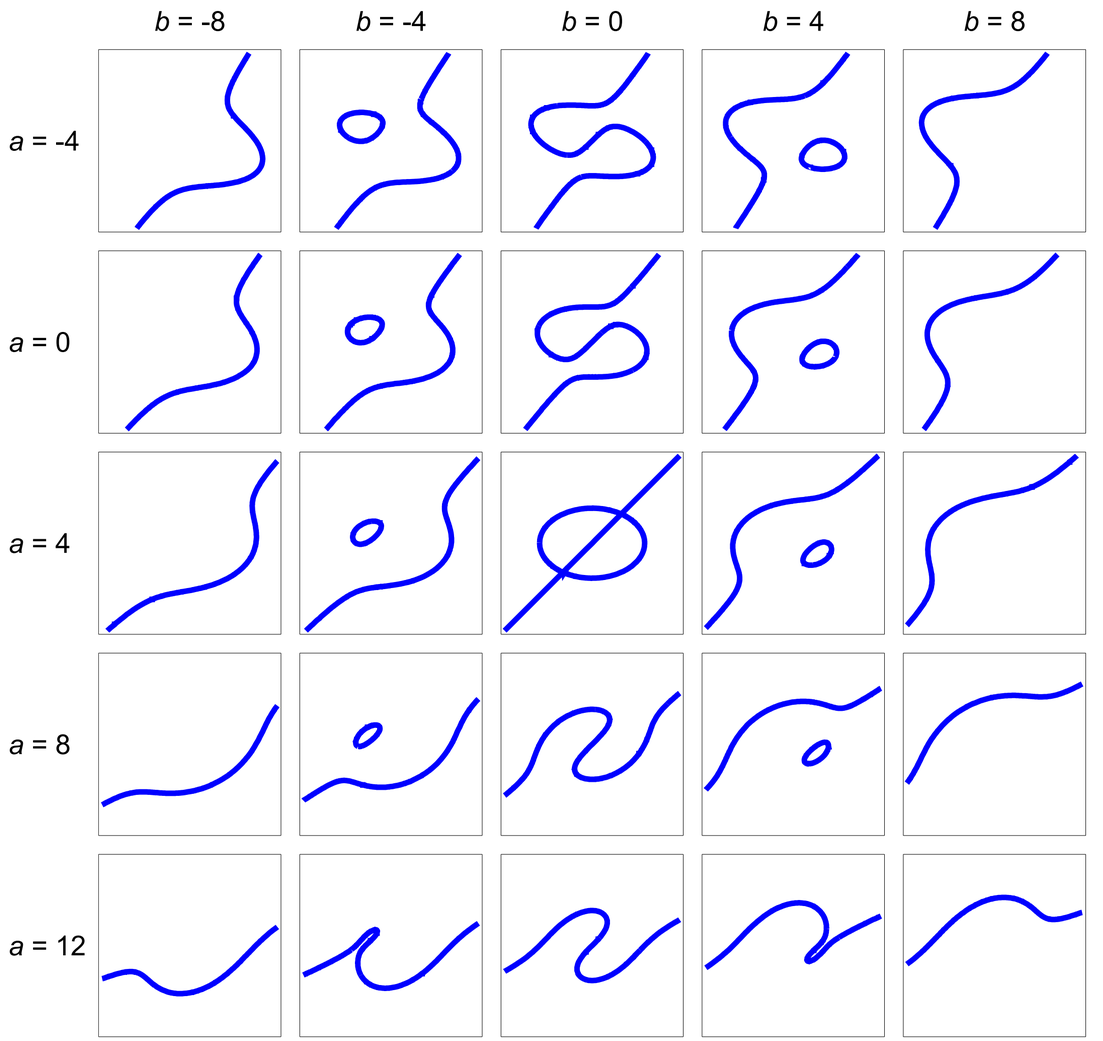
Für $\lambda \neq 0$ gilt also

$$F(\lambda x, \lambda y, \lambda z) = \lambda^3 F(x, y, z).$$

$\rightarrow$ Wir normalisieren $z = 1$

Beispiel: Für $a, b \in \mathbb{R}$

$$F(x, y) = 4x^3 - ax^2y + 9xy^2 - 9y^3 - 36x + 36y + 10b$$



Planare Algebraische Kurven

Aufgabe:

a) Baue ein Houdini Netzwerk welches den Funktionsgraphen von

$$F(x, y) = 4x^3 - ax^2y + 9xy^2 - 9y^3 - 36x + 36y + 10b$$

visualisiert.

b) Visualisiere den die jeweiligen Funktionswerte des Graphen farbig.

c) Visualisiere die planaren algebraischen Kurven welche durch F definiert werden für verschiedene Werte von $a, b \in \mathbb{R}$.

Technische Universität Berlin

Materialien und Rendering

Berlin Mathematical School BMS

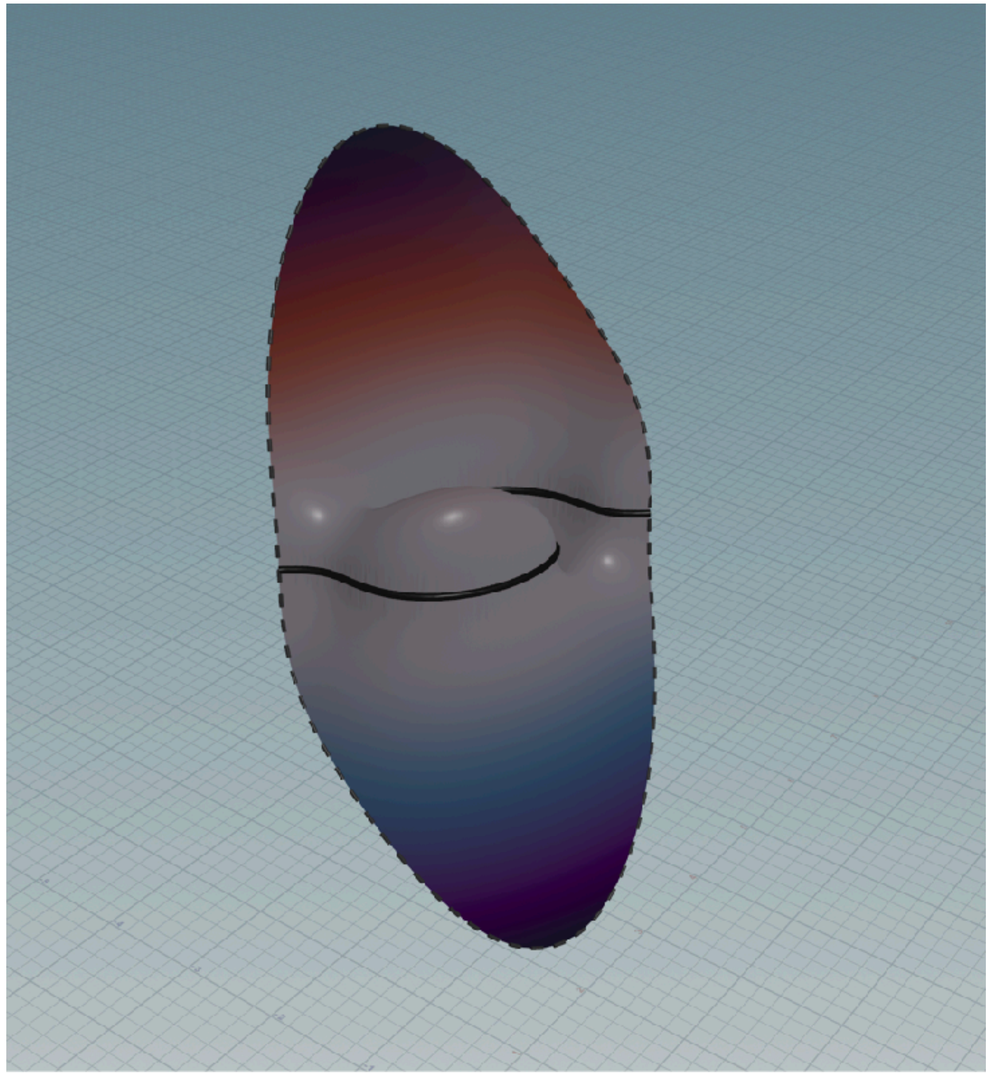
SideFX

DGD

SFB TRR 109 Discretization in Geometry and Dynamics

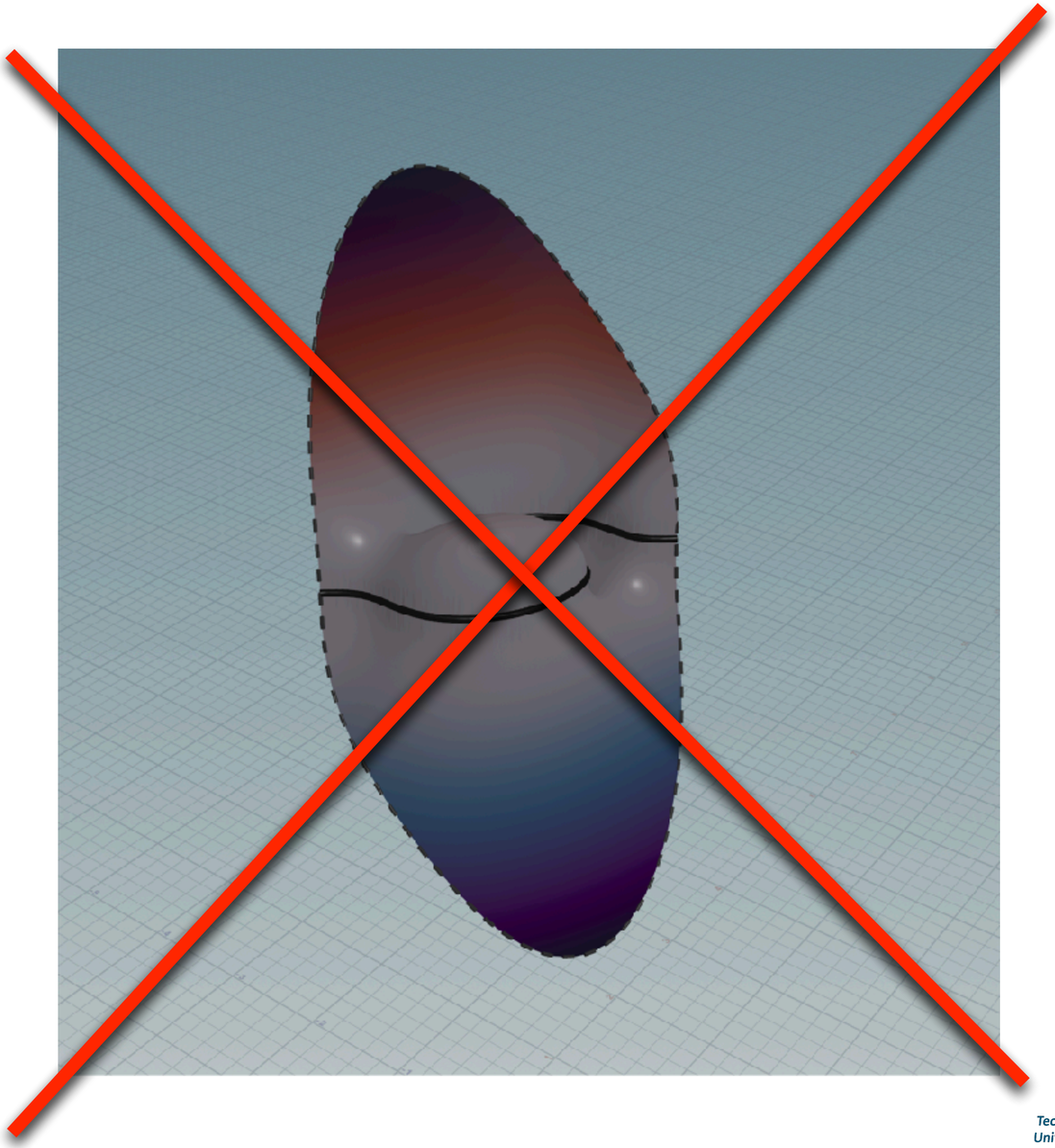
Materialien und Rendering

Unser Ziel ist es die von uns erstellten Visualisierungen als “schöne” Bilder zu rendern.

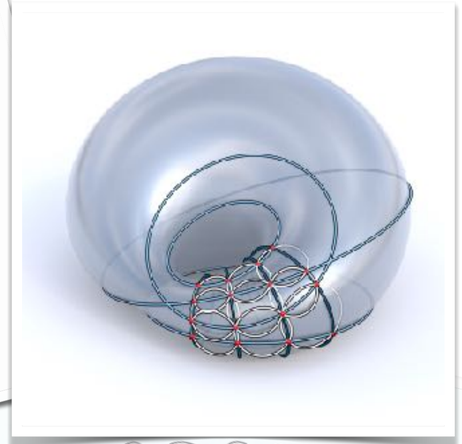
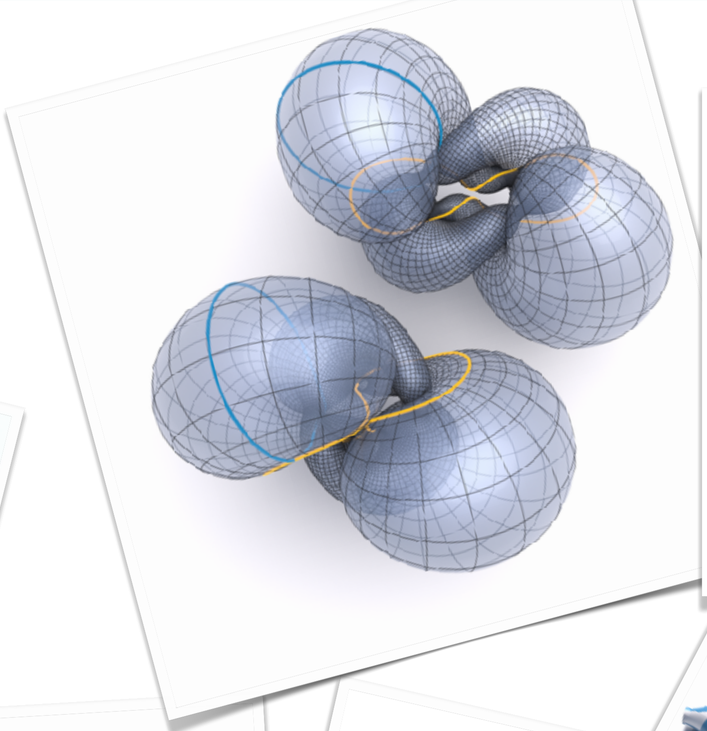
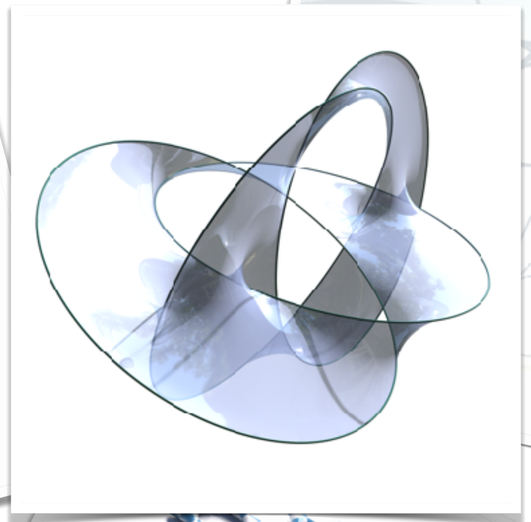
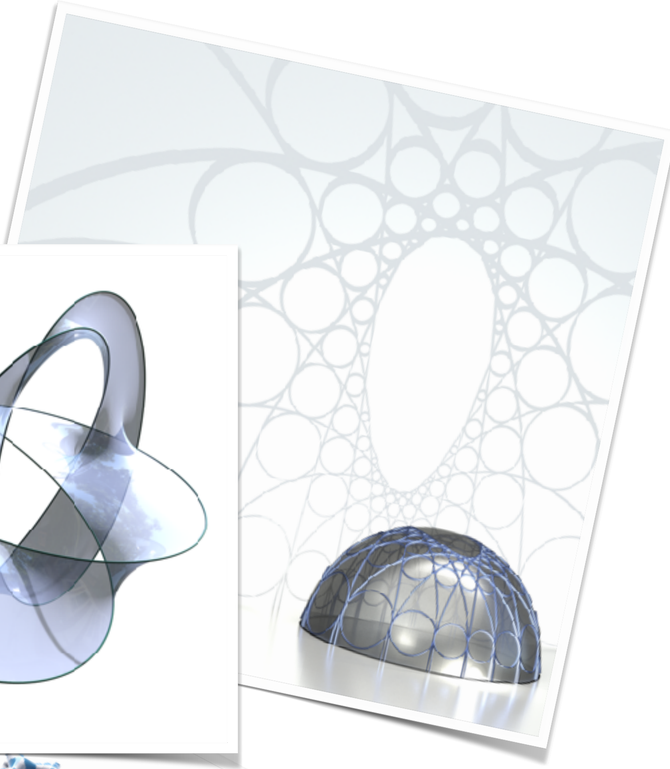
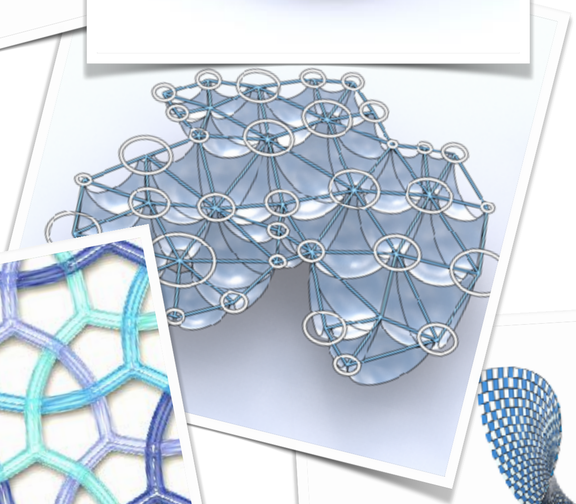
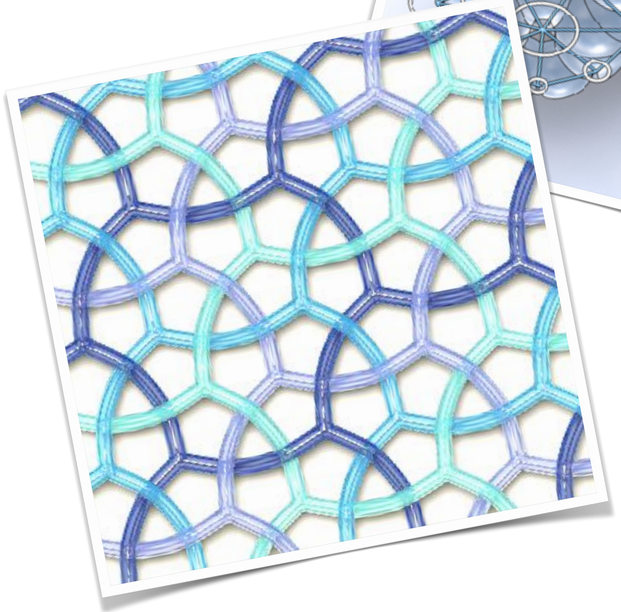
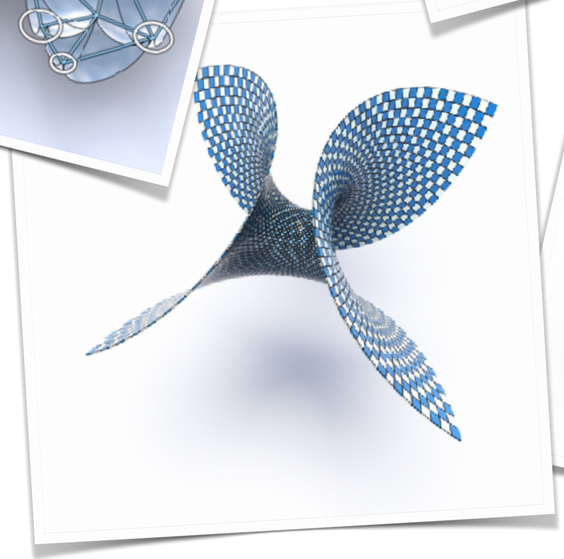


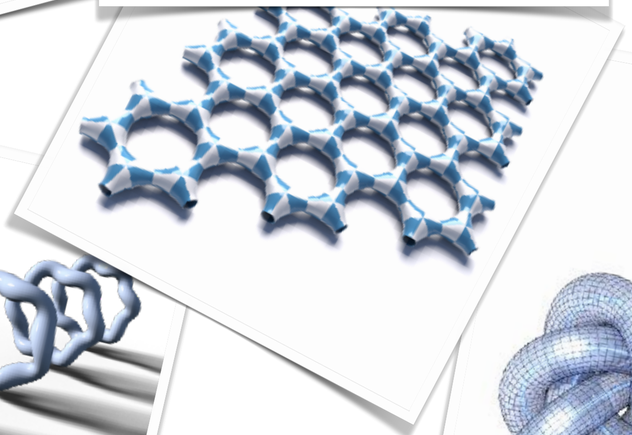
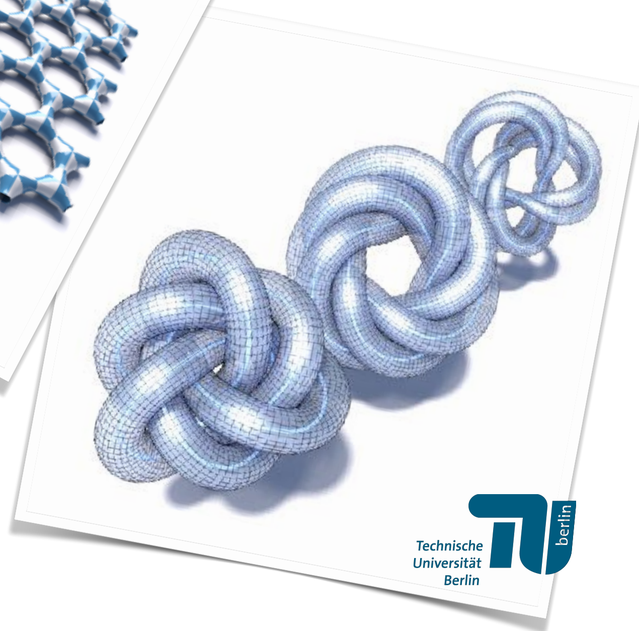
Materialien und Rendering

Unser Ziel ist es die von uns erstellten Visualisierungen als “schöne” Bilder zu rendern.

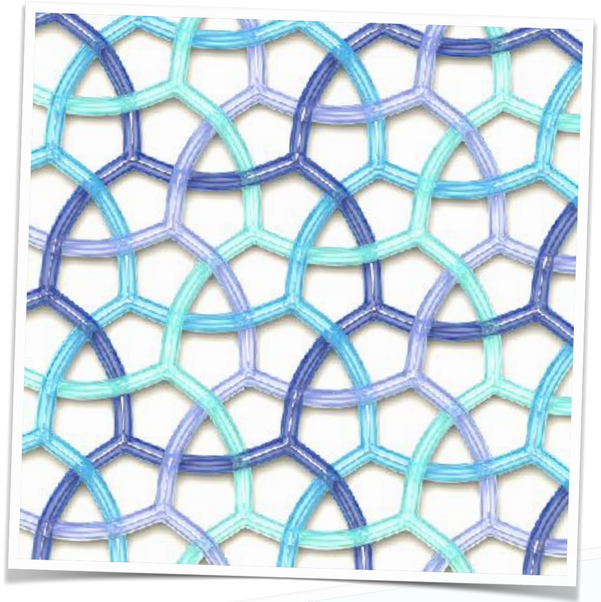


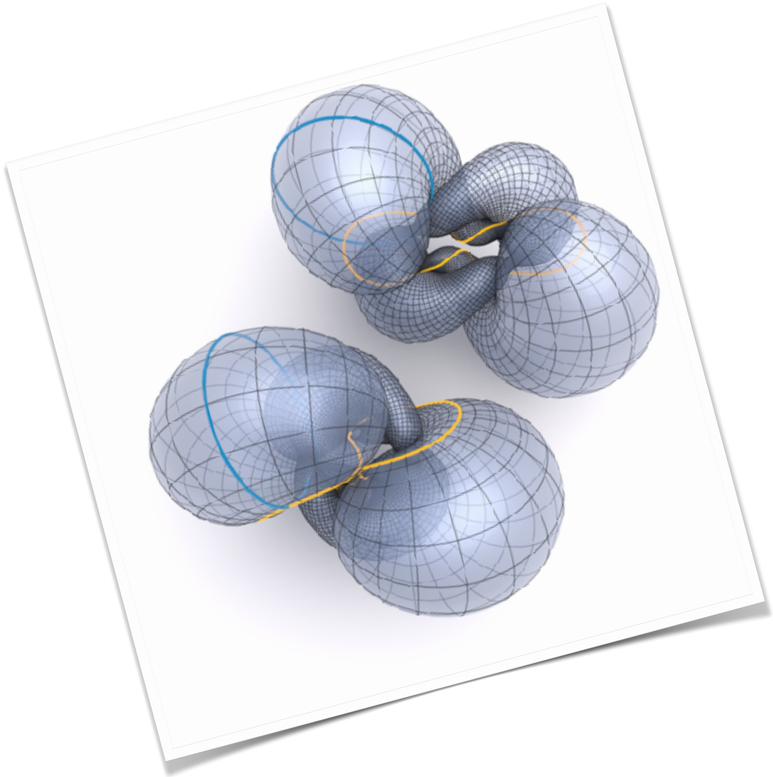
Materialien und Rendering

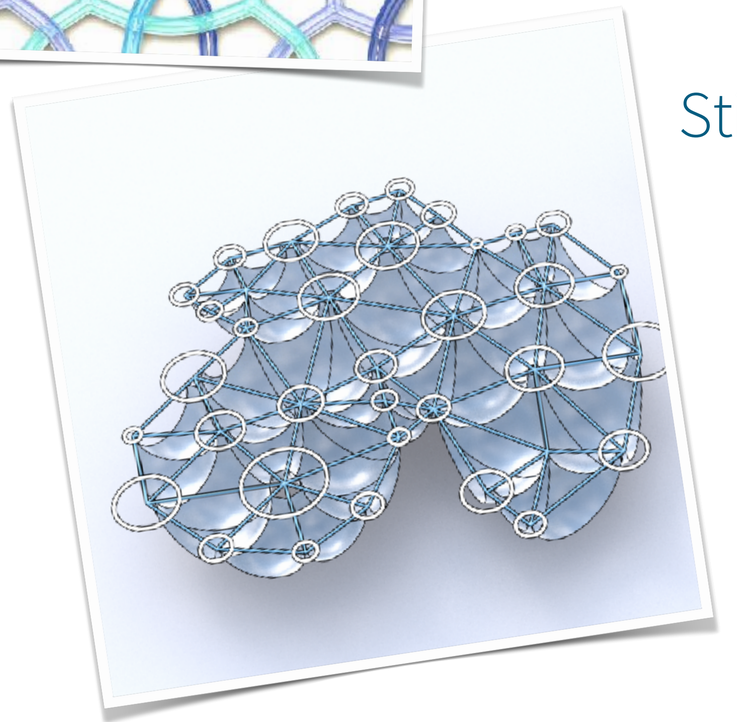



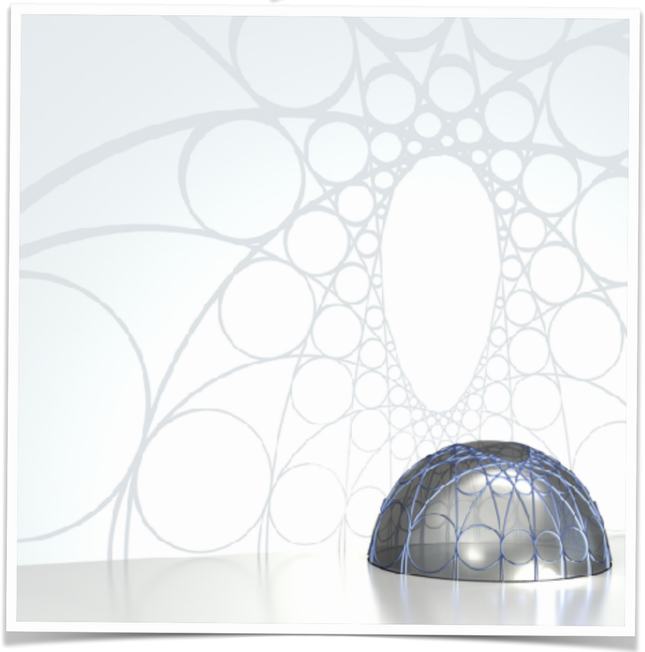
Materialien und Rendering



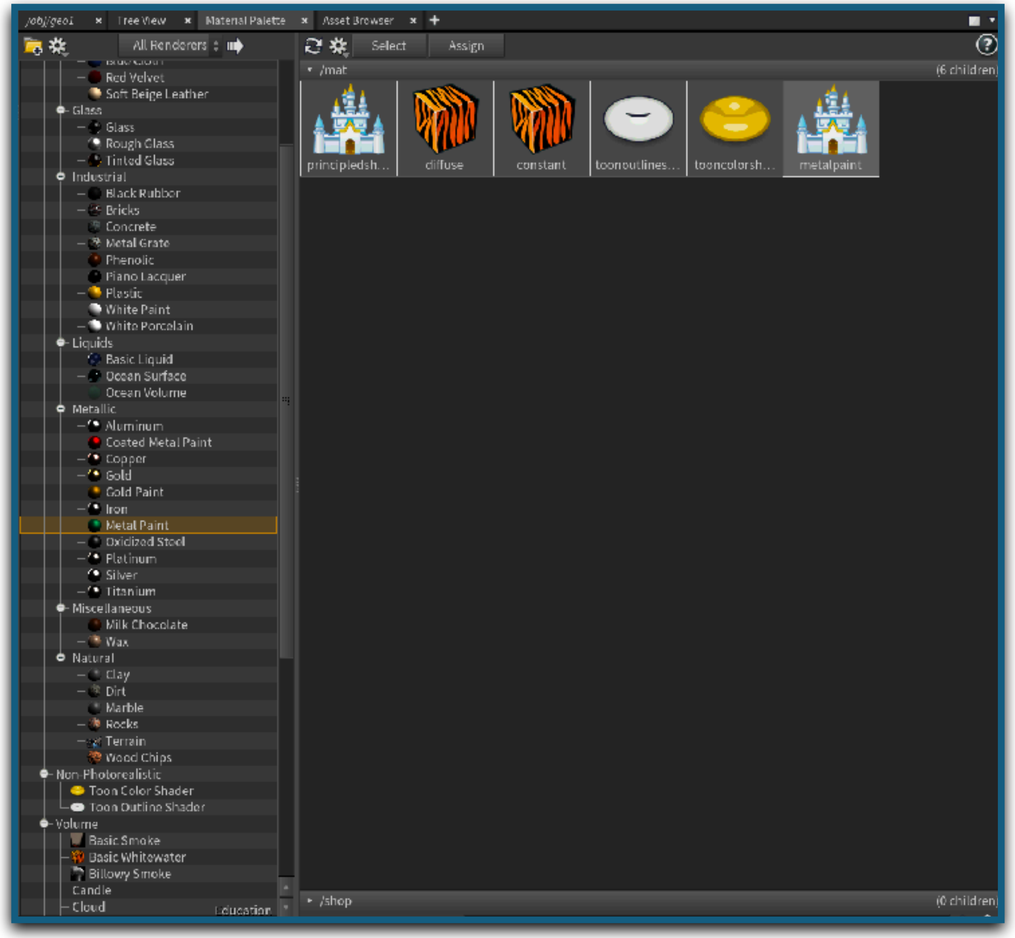


Materialien ermöglichen
uns verschiedene
Stilisierungen von Bildern

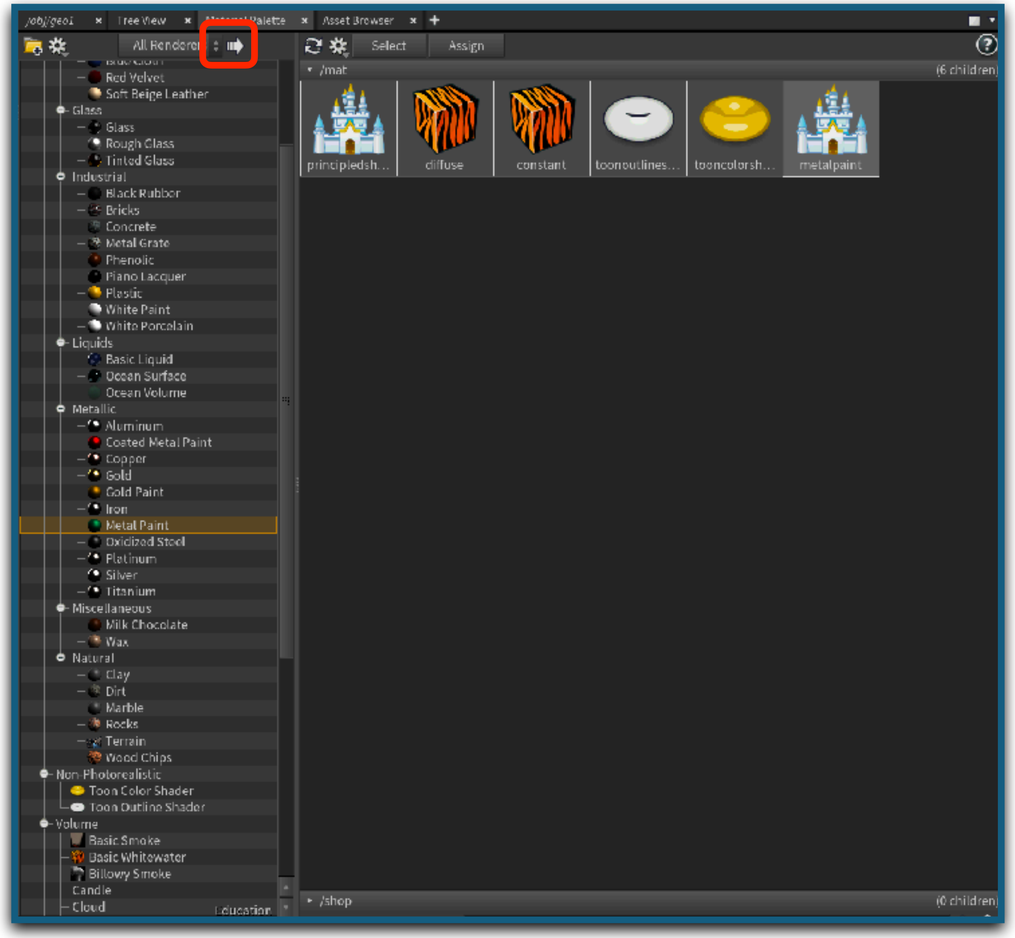




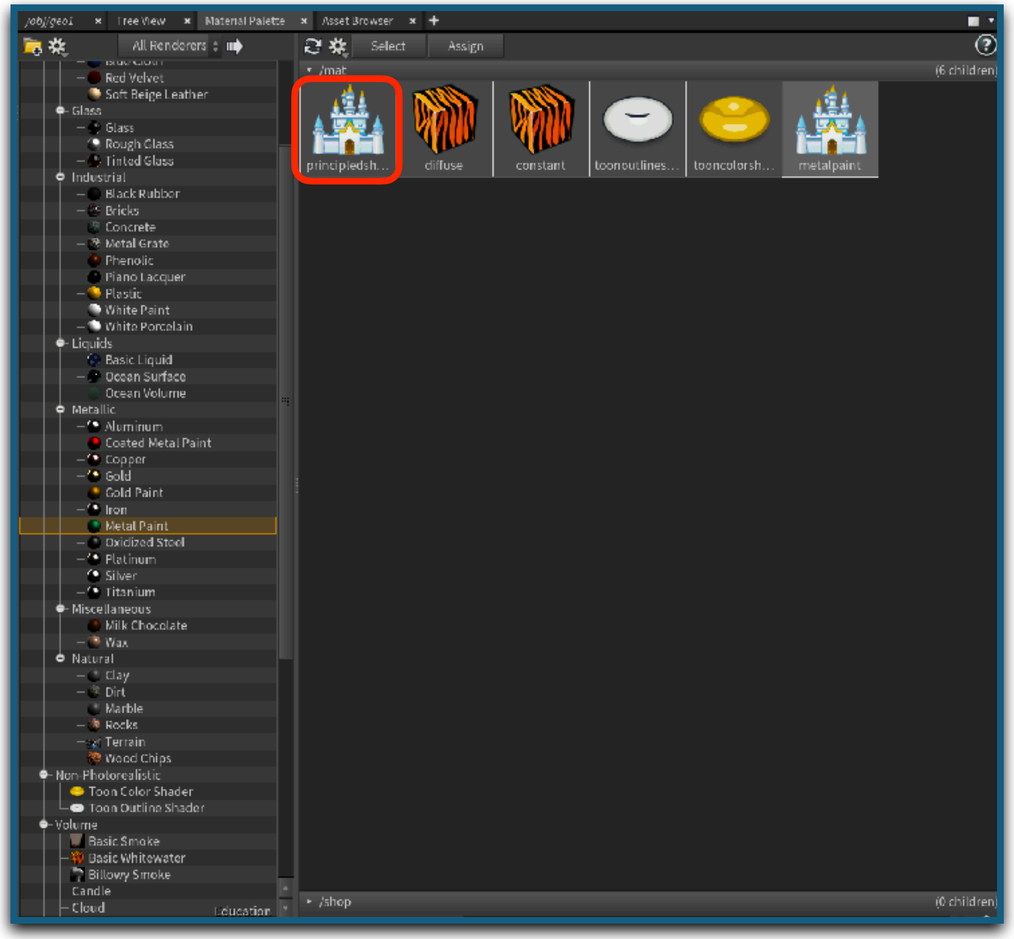
Materialien

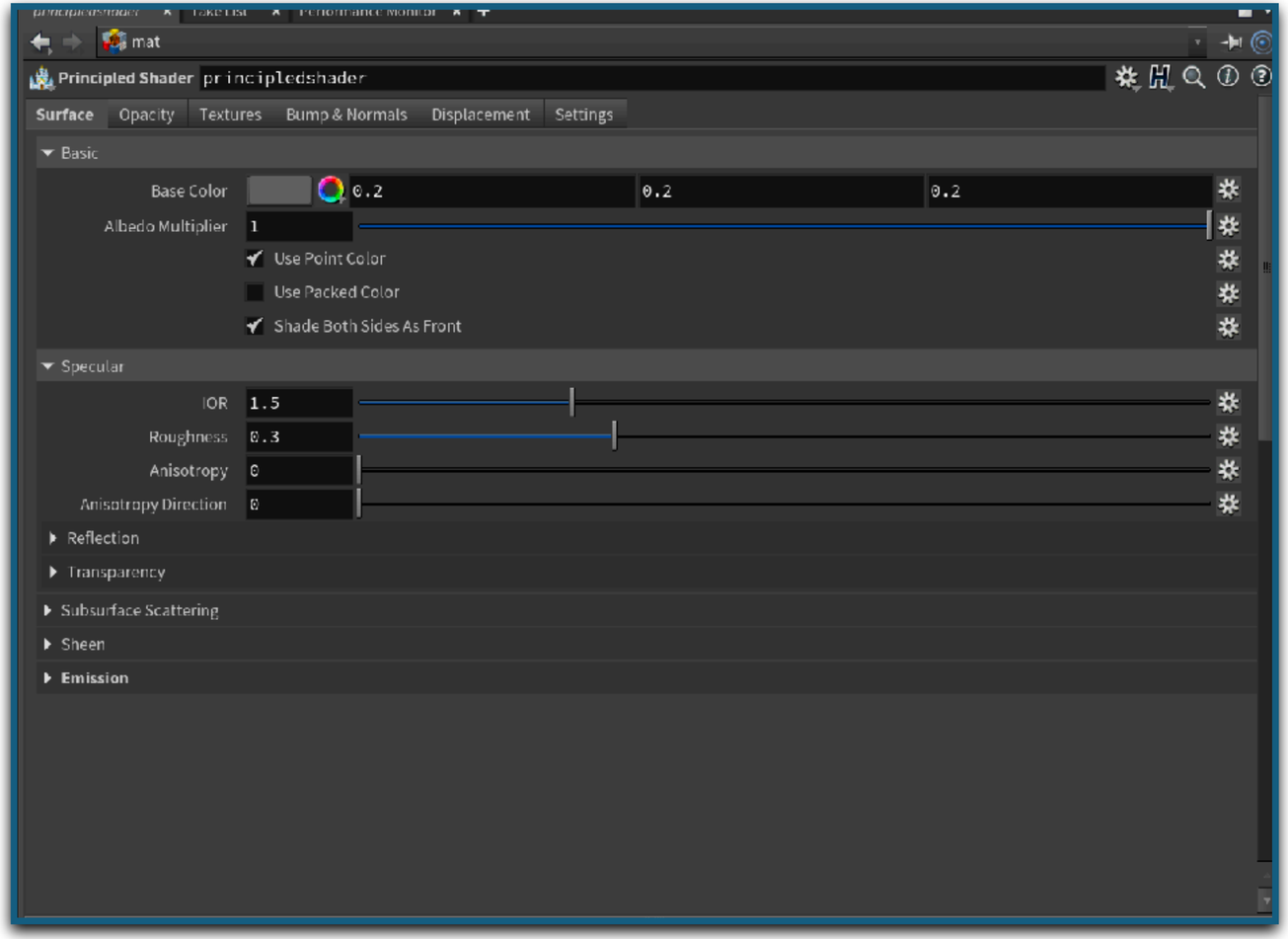


Materialien

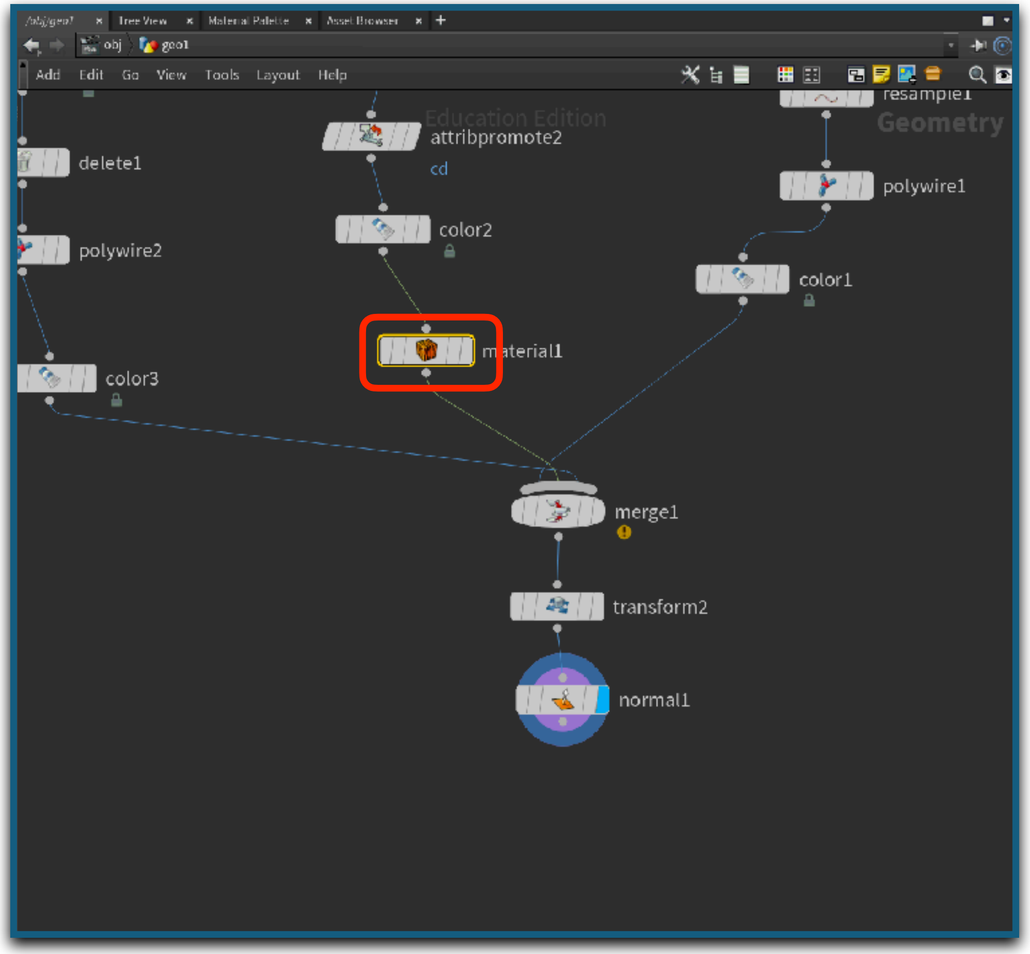


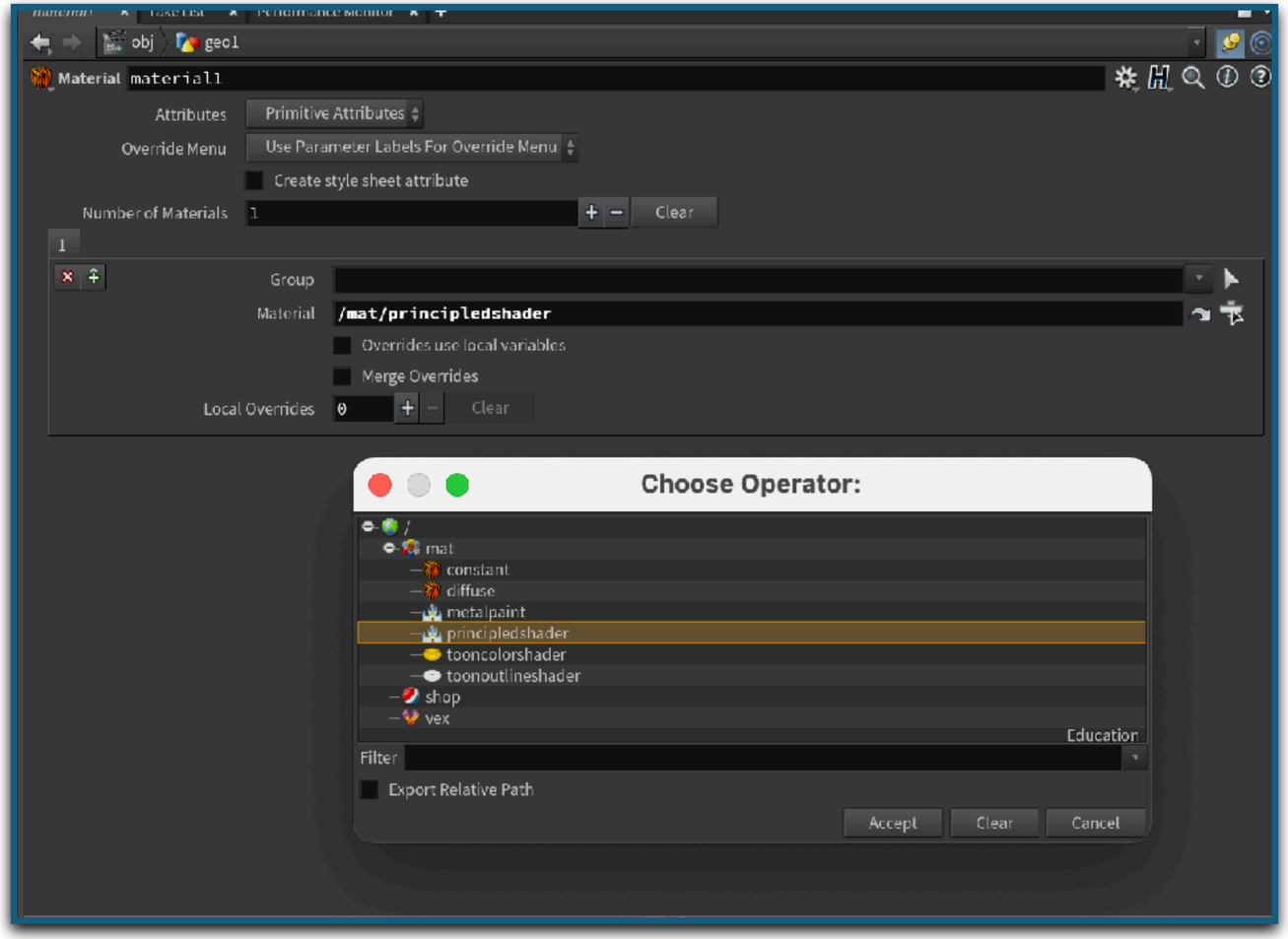
Materialien



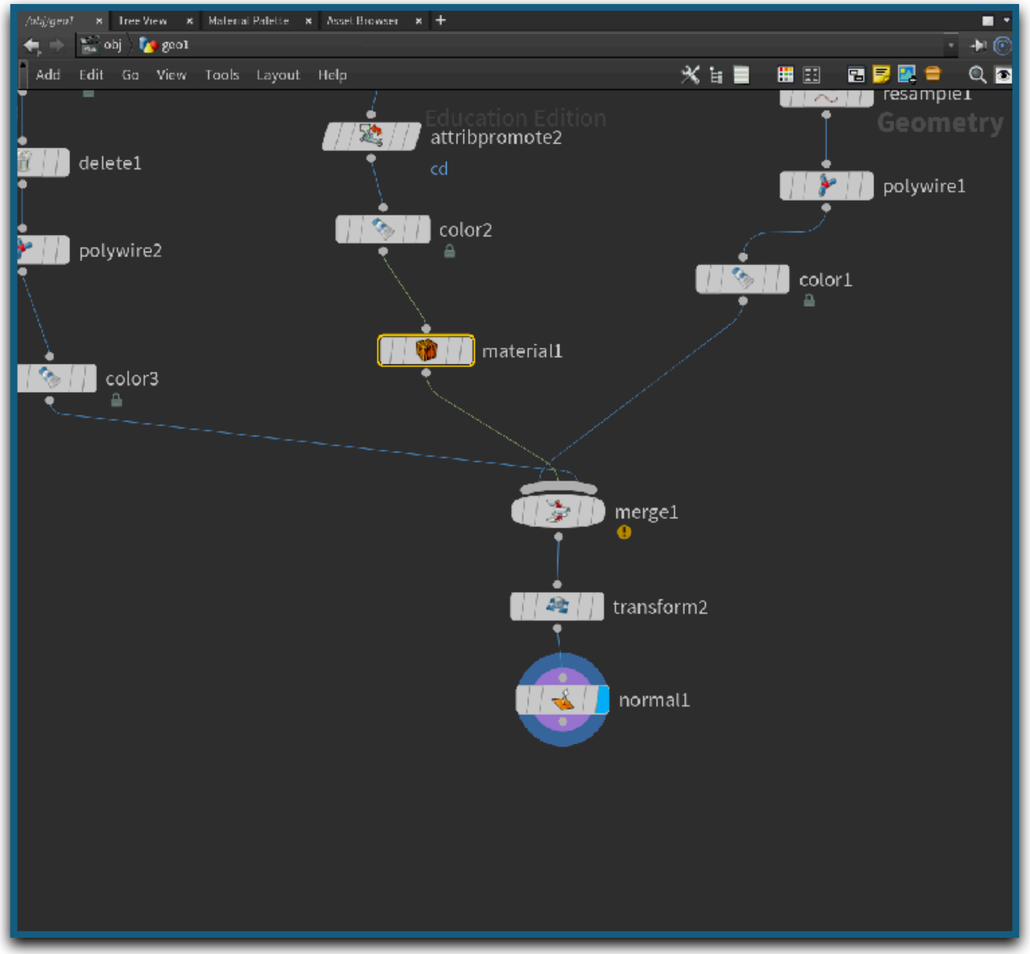


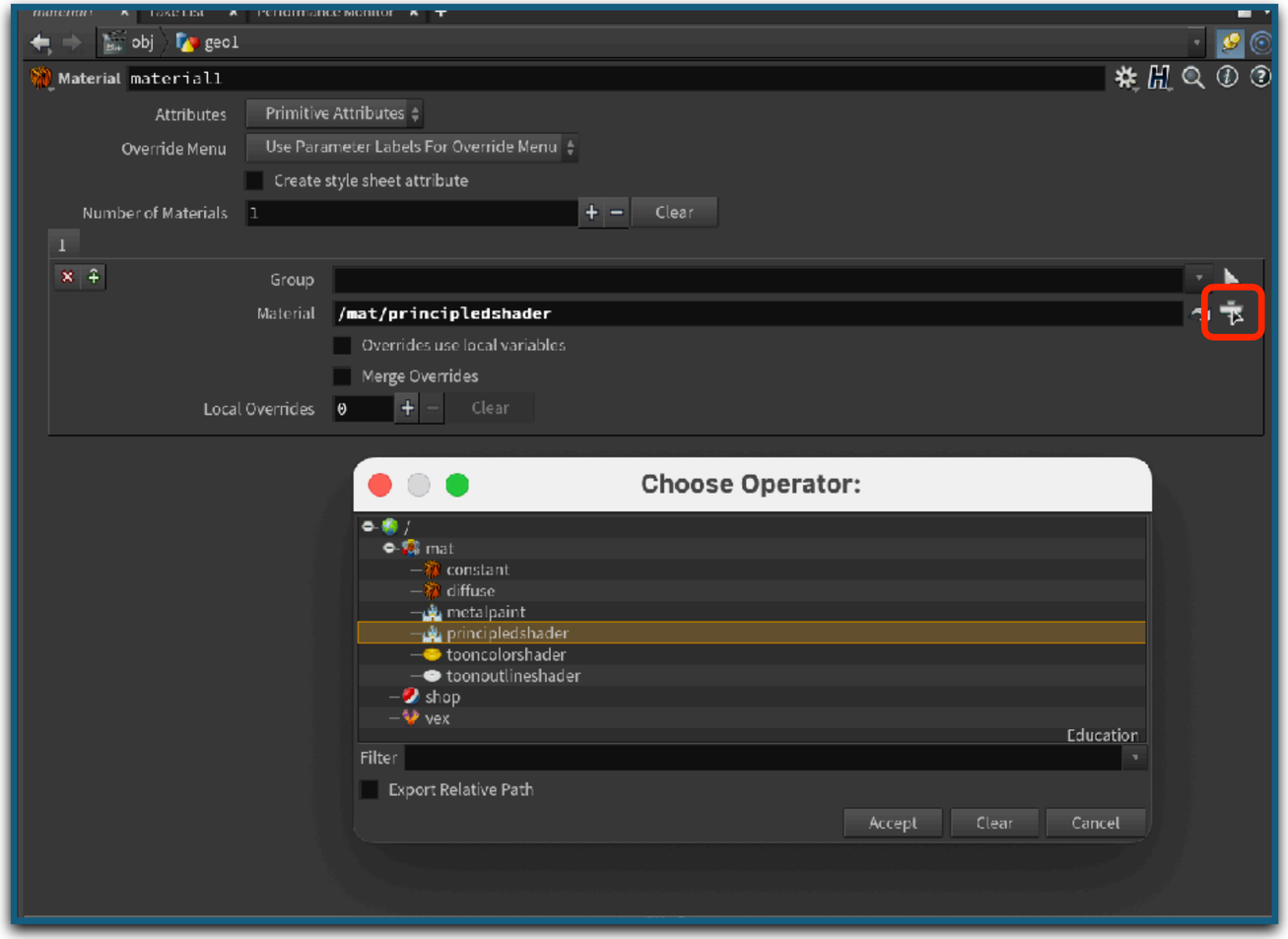
Materialien



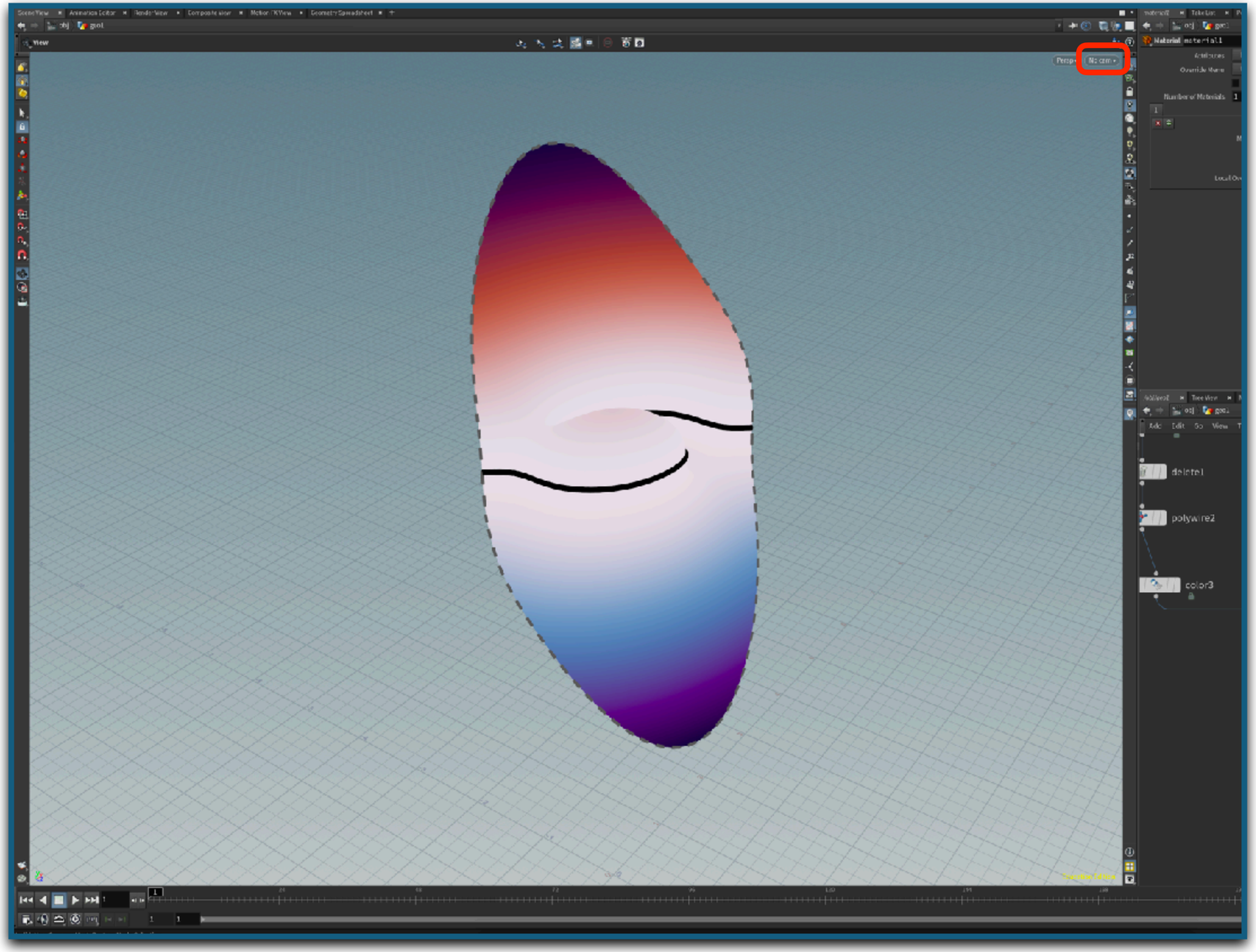


Materialien



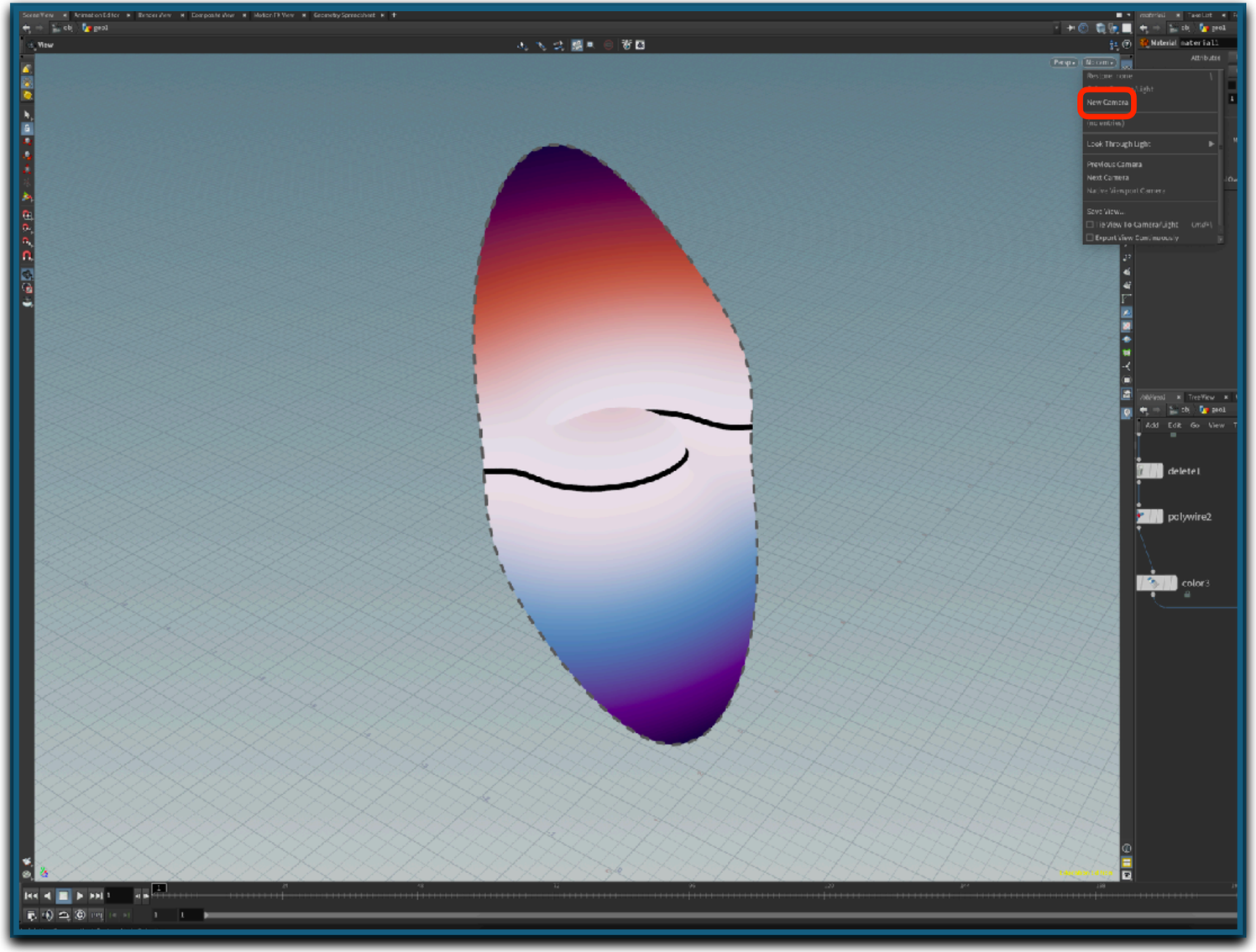


KAMERA



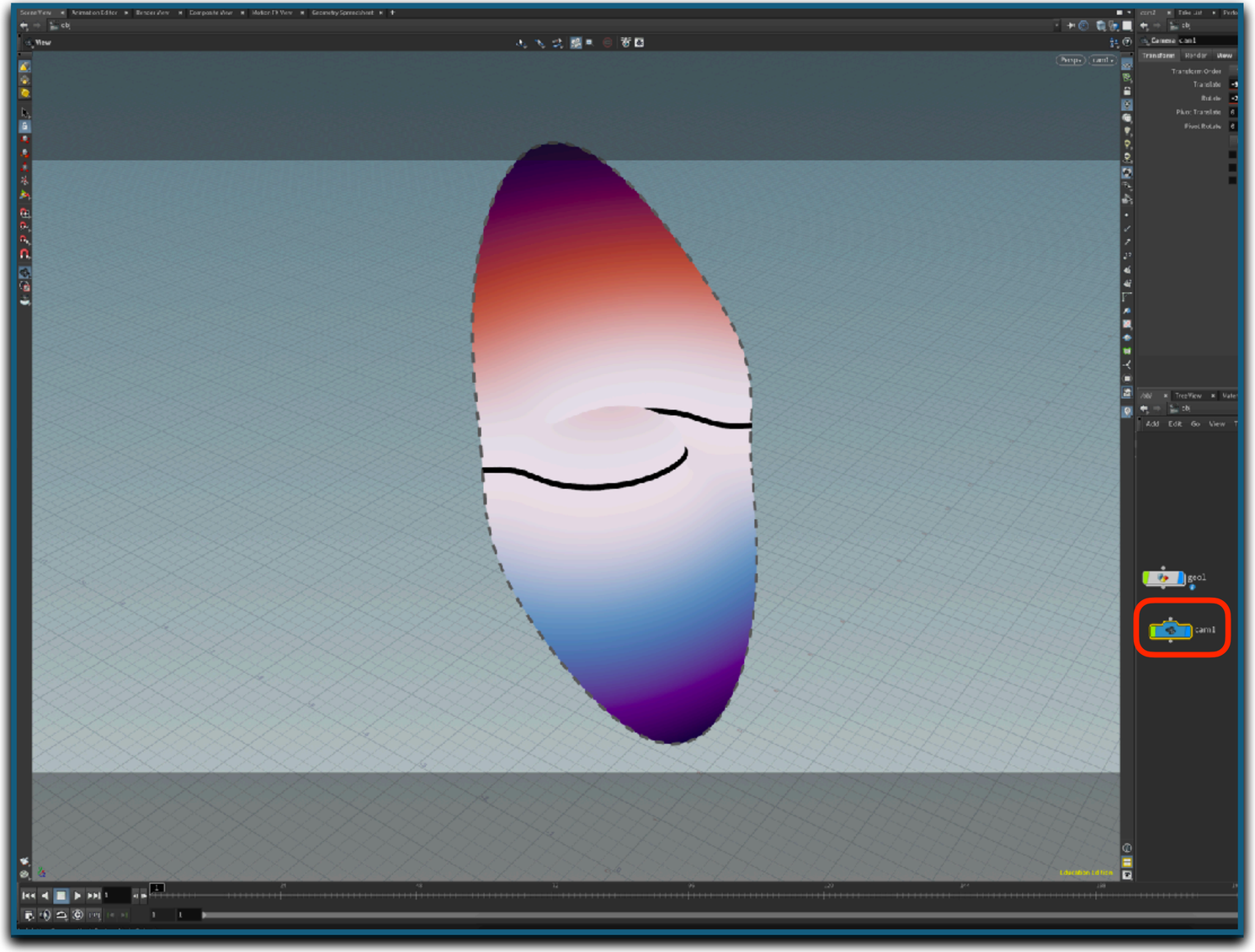
- Kamera Menü

Kamera



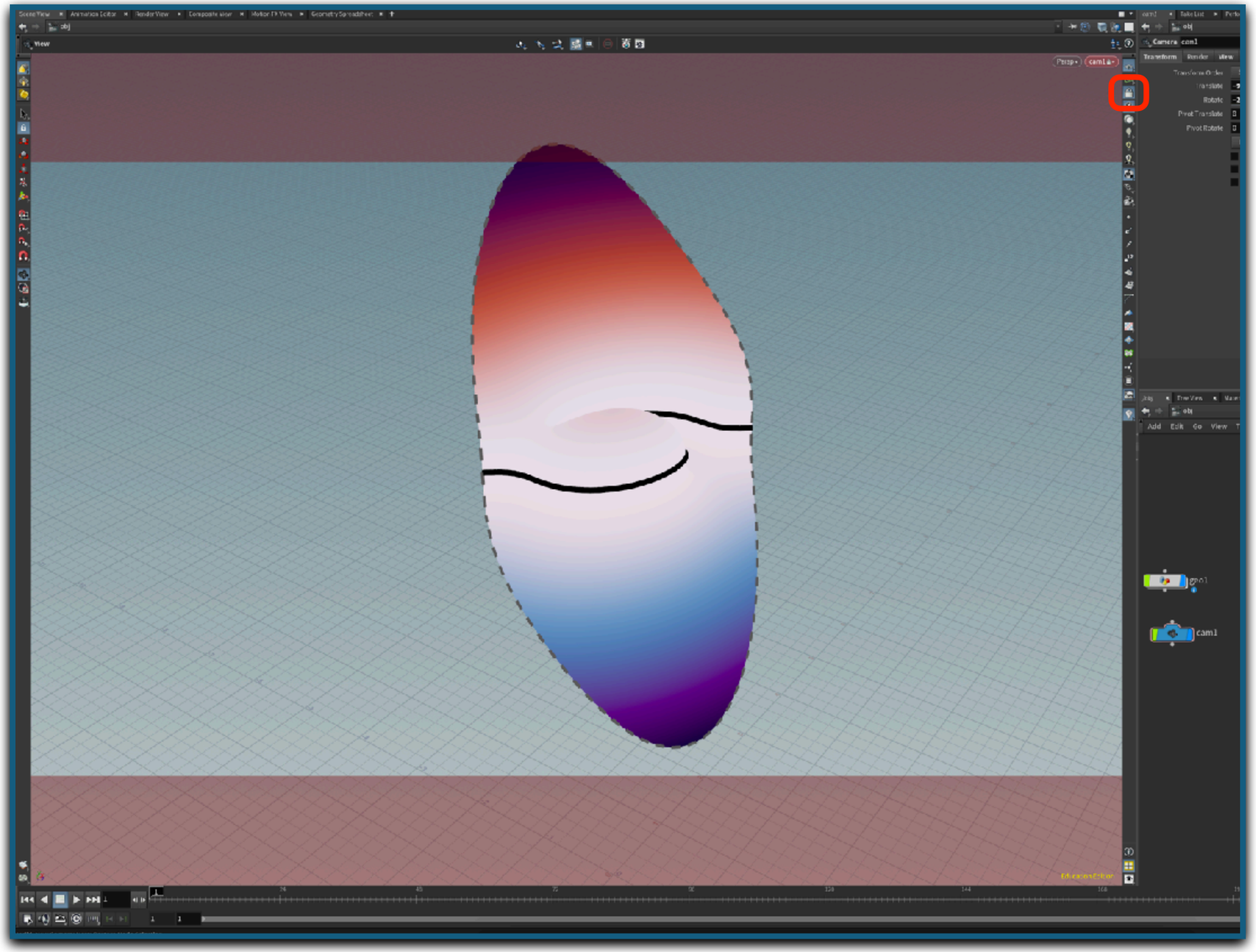
- Kamera Menü
- Neue Kamera erzeugen

KAMERA



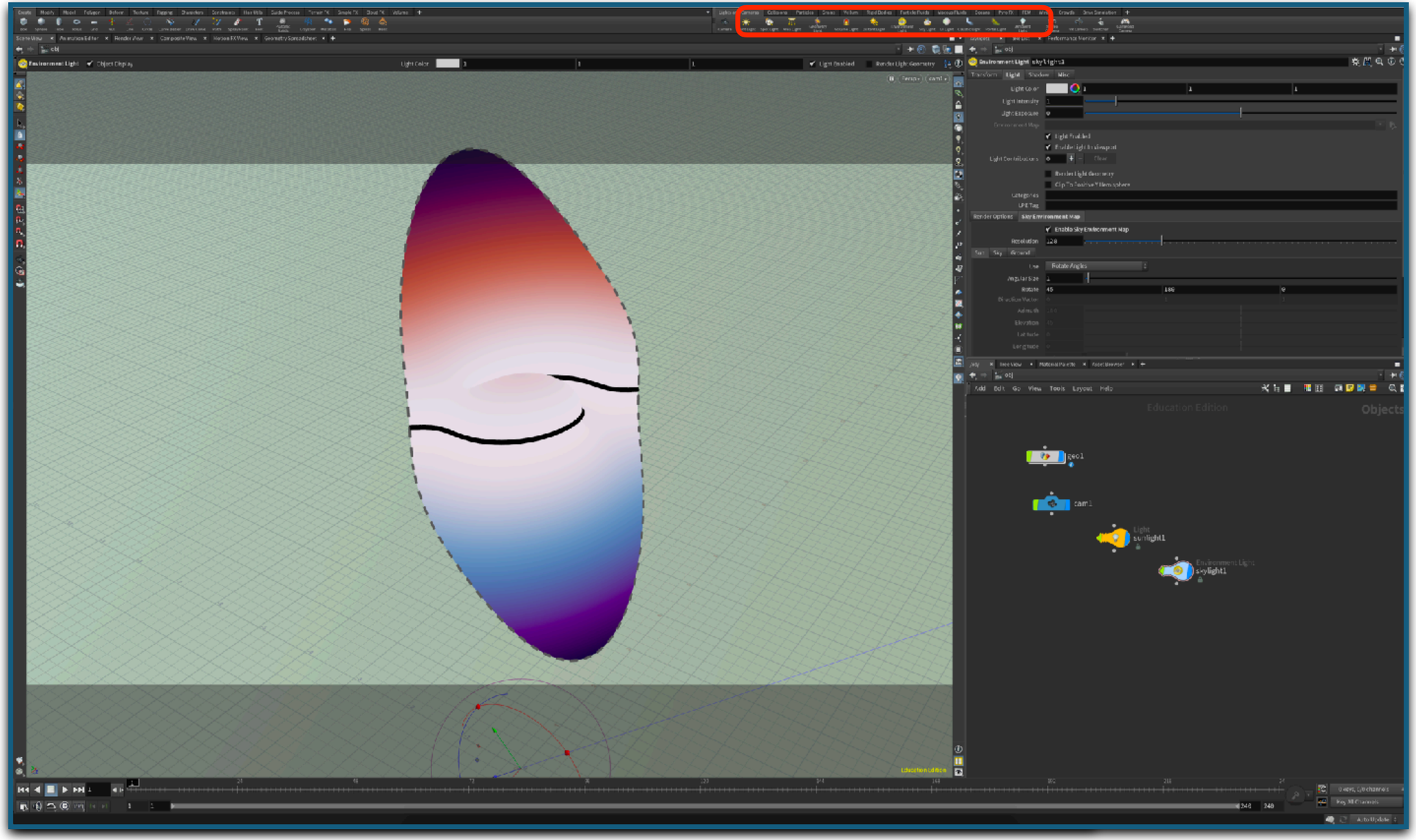
- Kamera Menü
- Neue Kamera erzeugen
- Kamera im Network View

KAMERA

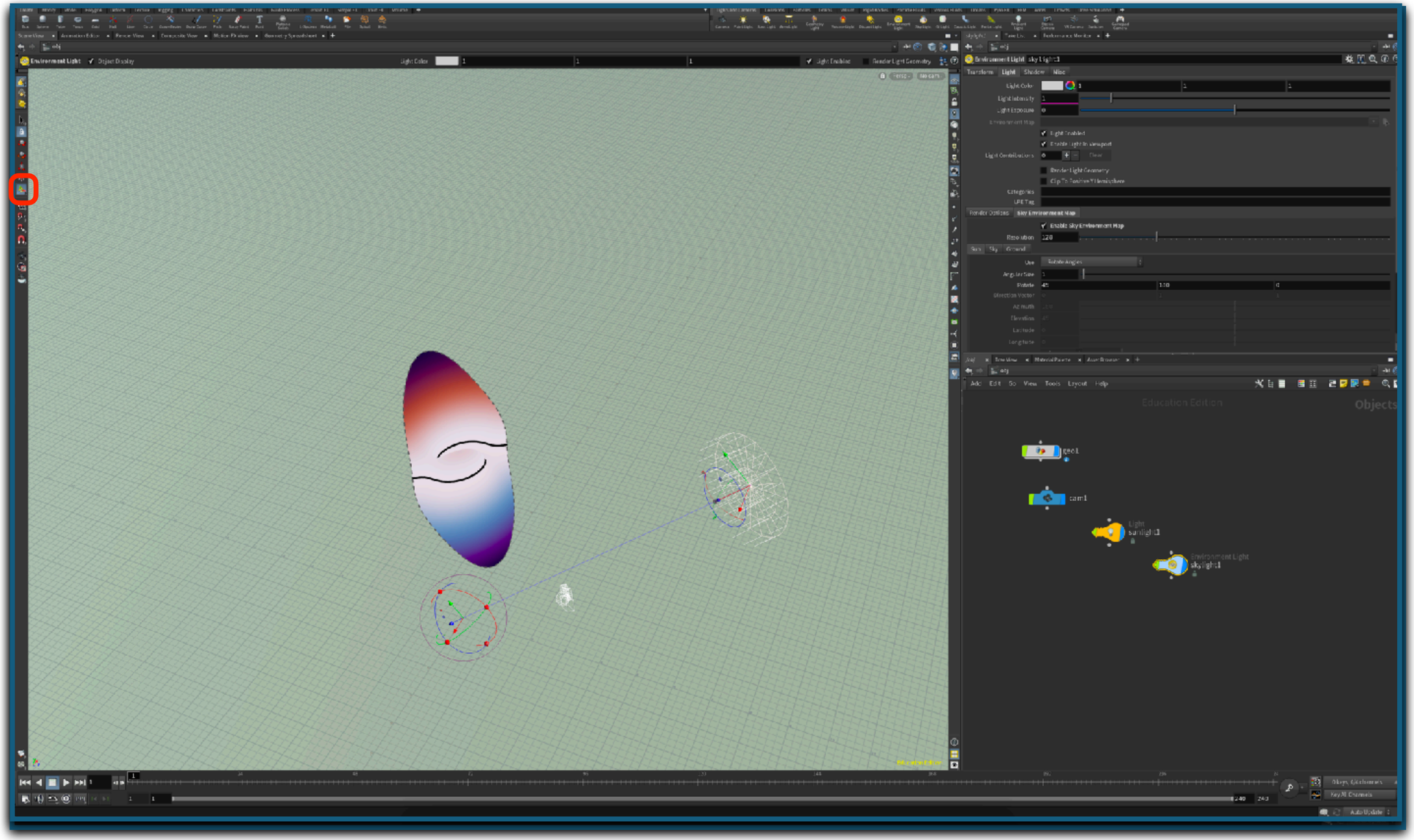


- Kamera Menü
- Neue Kamera erzeugen
- Kamera im Network View
- Neu Positionieren

Licht

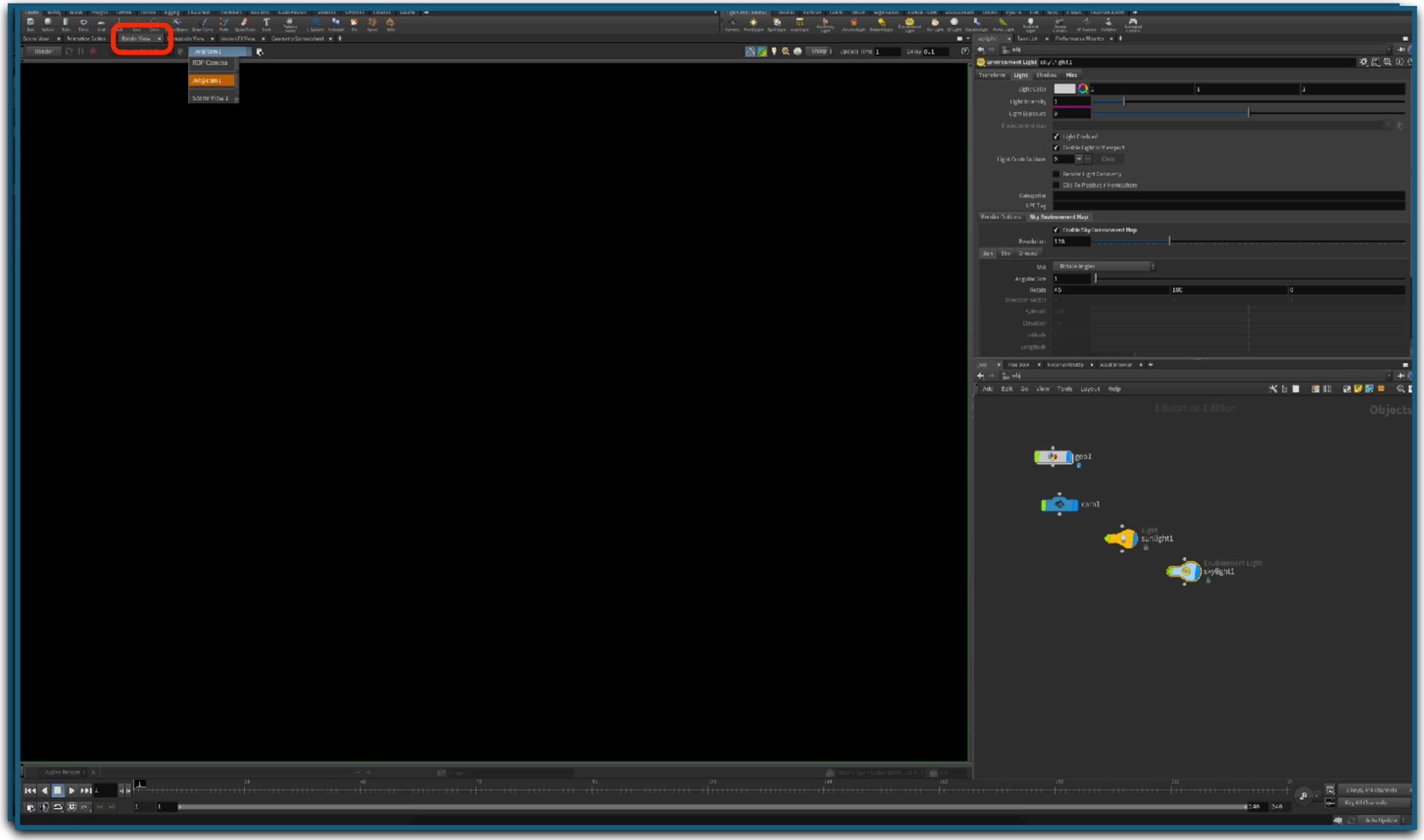


Licht

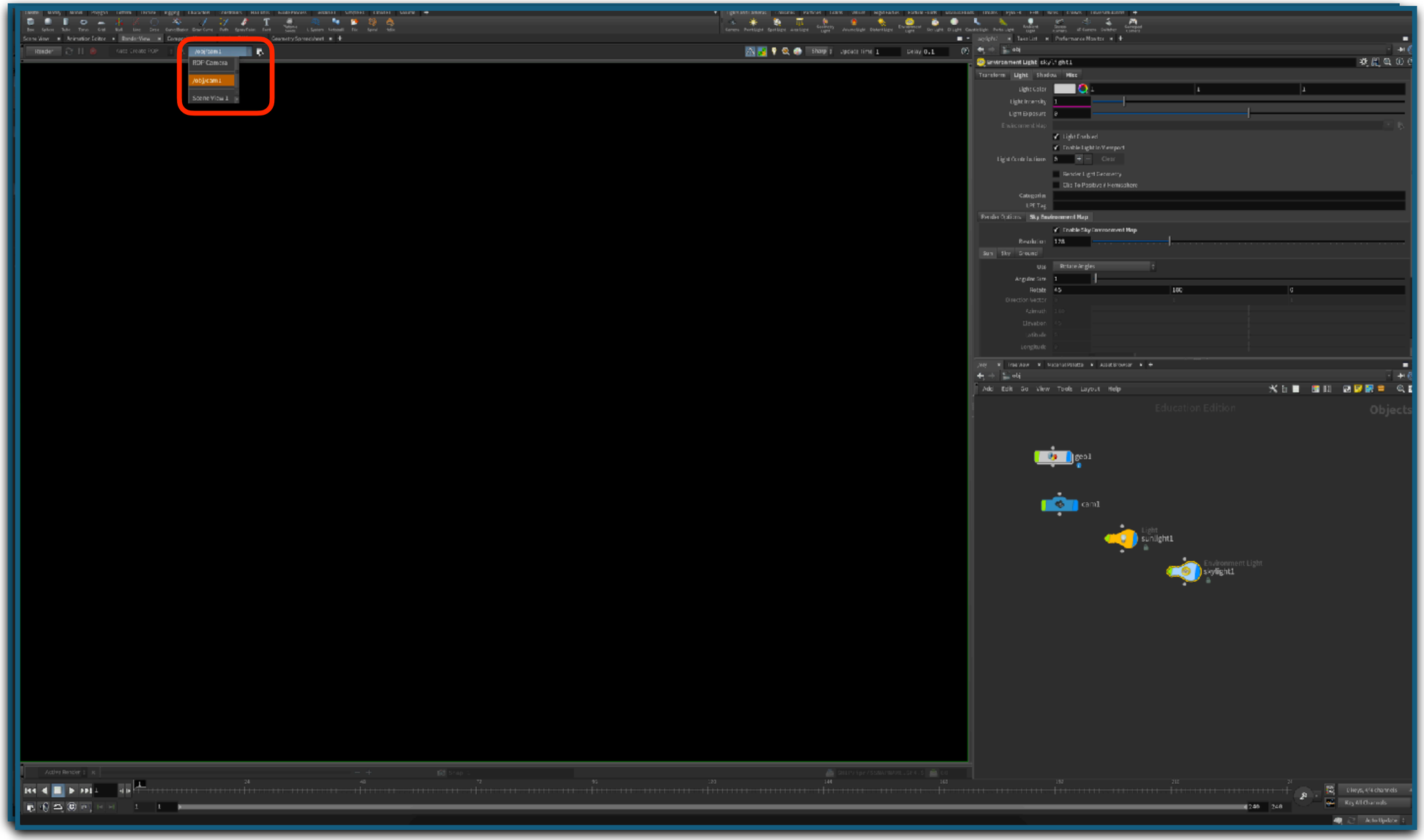


Technische Universität Berlin

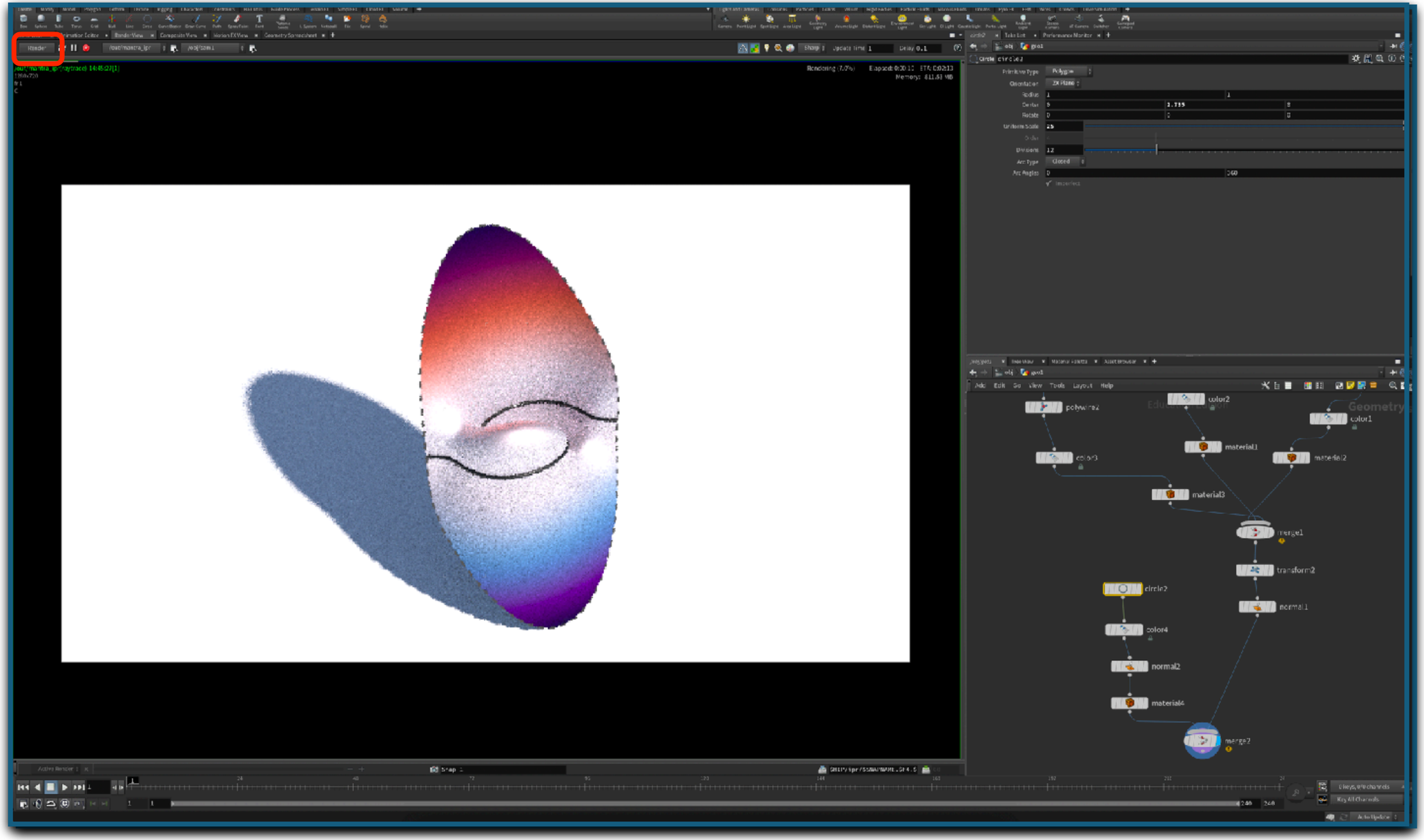
Rendering


Render View
/obj/cam1
Scene View 1
Education Edition
Objects
geo1
cam1
Light
sunlight1
Environment Light
skylight1

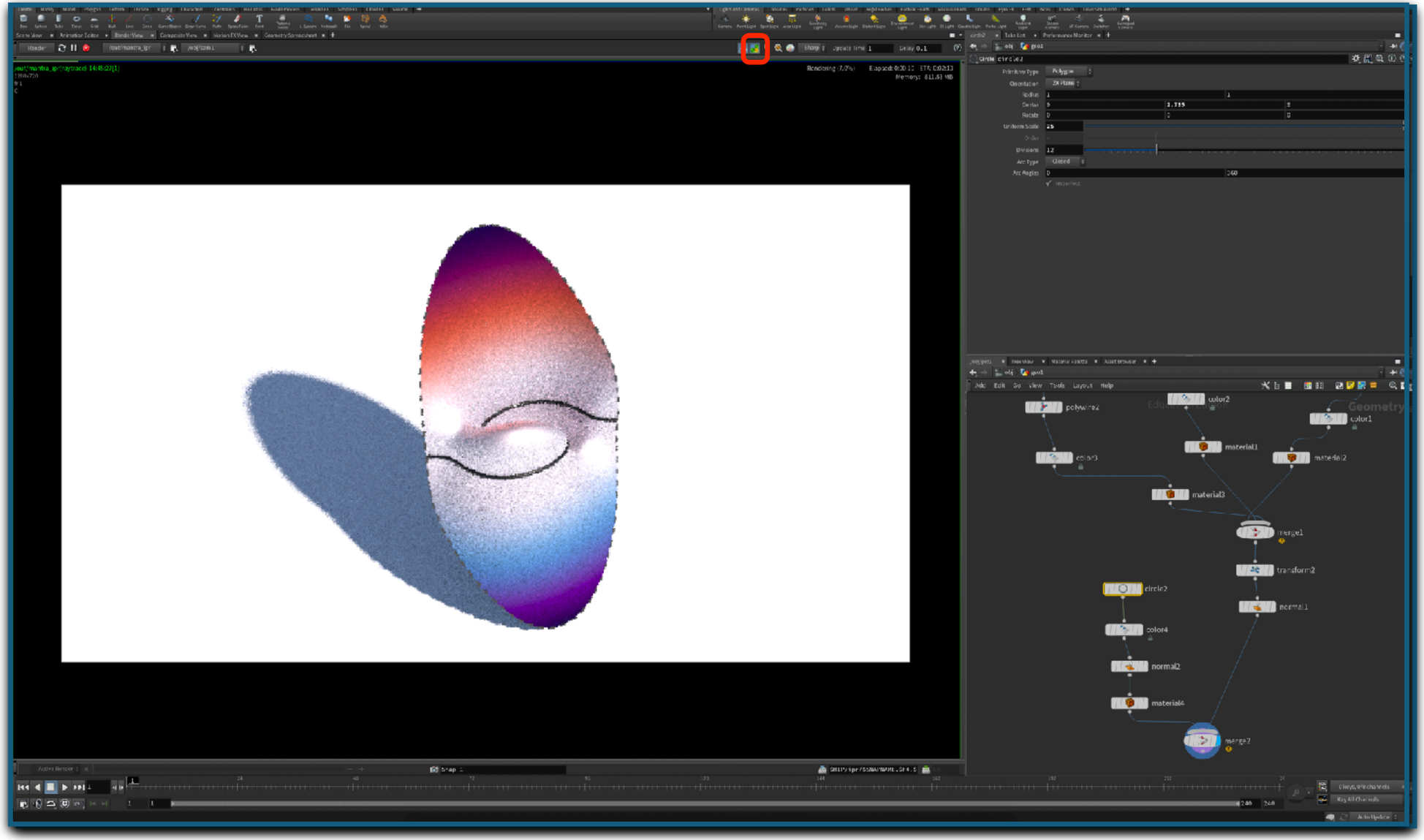
Rendering


/obj/cam1
/obj/cam1
Scene View 1
Education Edition
Objects
geo1
cam1
Light
sunlight1
Environment Light
skylight1

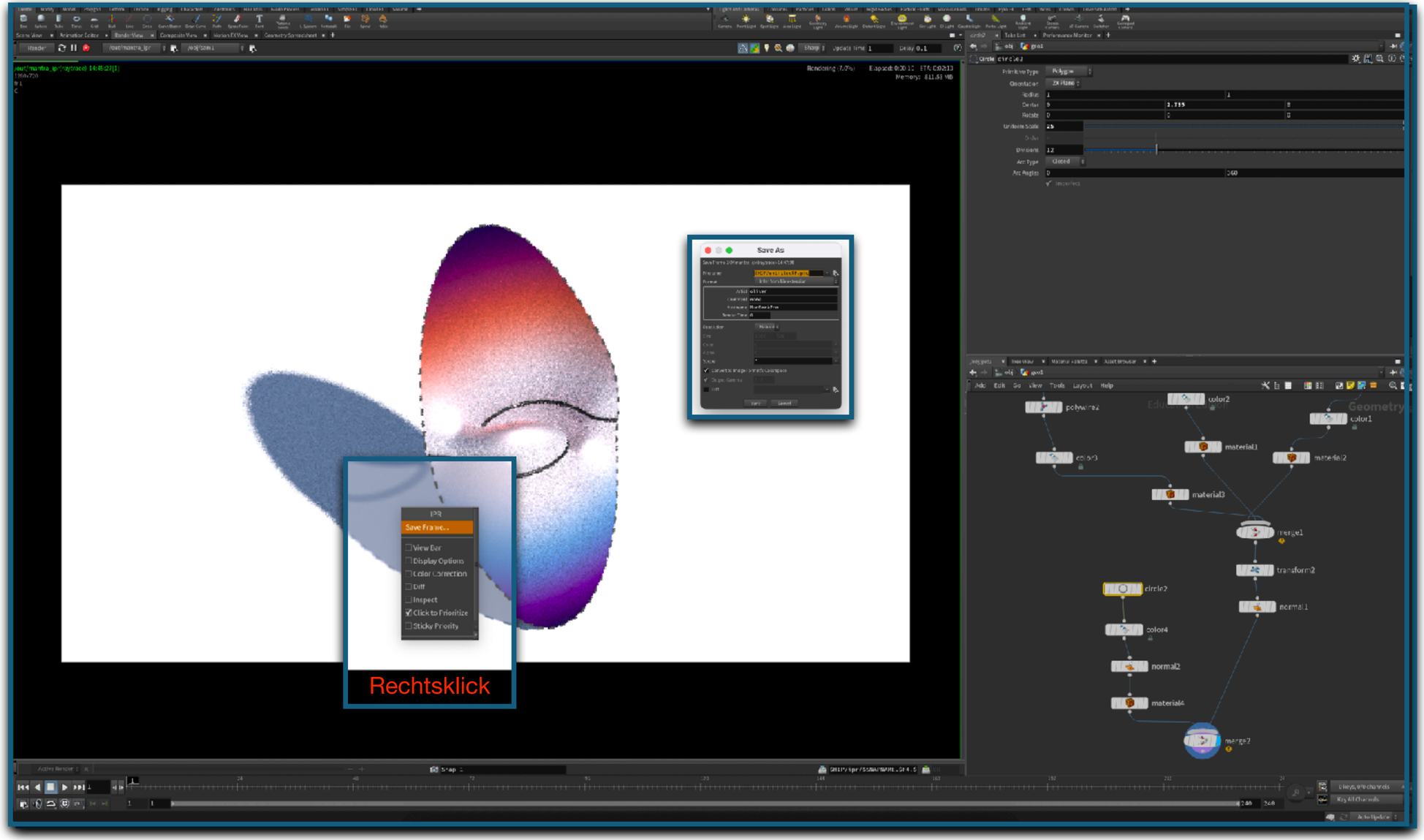
Rendering


color2
polywire2
color1
material1
color3
material2
material3
merge1
transform2
circle2
normal1
color4
normal2
material4
merge2

Rendering


color2
polywire2
color1
material1
color3
material2
material3
merge1
transform2
circle2
normal1
color4
normal2
material4
merge2

Rendering

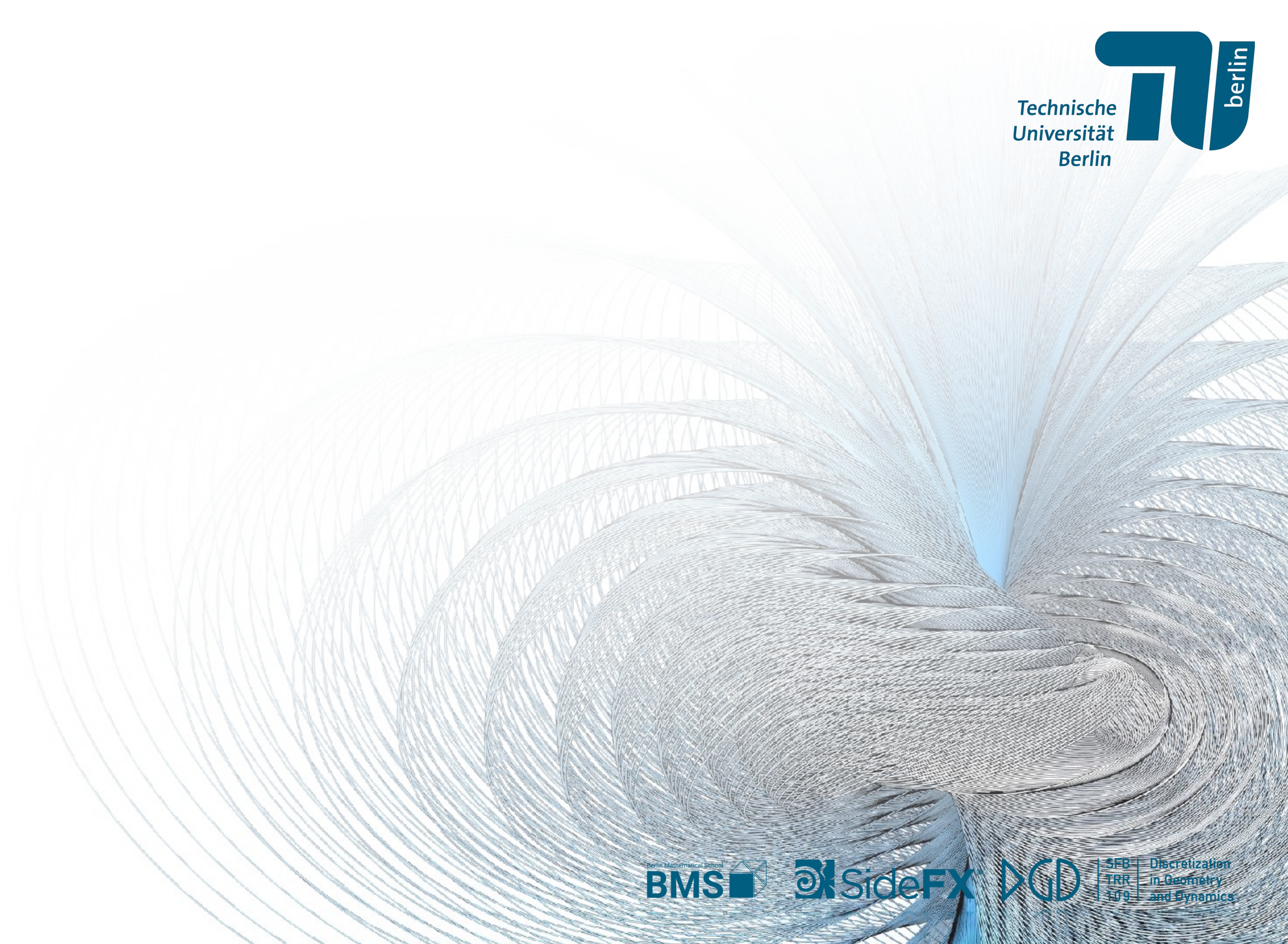

Save As
IPR
Save Frame...
View Bar
Display Options
Color Correction
Diff
Inspect
Click to Prioritize
Sticky Priority
Rechtsklick
color2
color1
material1
material2
material3
merge1
transform2
circle2
normal1
color4
normal2
material4
merge2

Rendering

Übung:

a) Wähle Materialien aus, Rendere eine Szene und speichere das Bild ab.

Pause



Technische Universität Berlin

Berlin Mathematical School BMS

SideFX

DGD

SFB TRR 109 Discretization in Geometry and Dynamics

Technische Universität Berlin

Funktionen

Berlin Mathematical School BMS

SideFX

DGD

SFB TRR 109 Discretization in Geometry and Dynamics

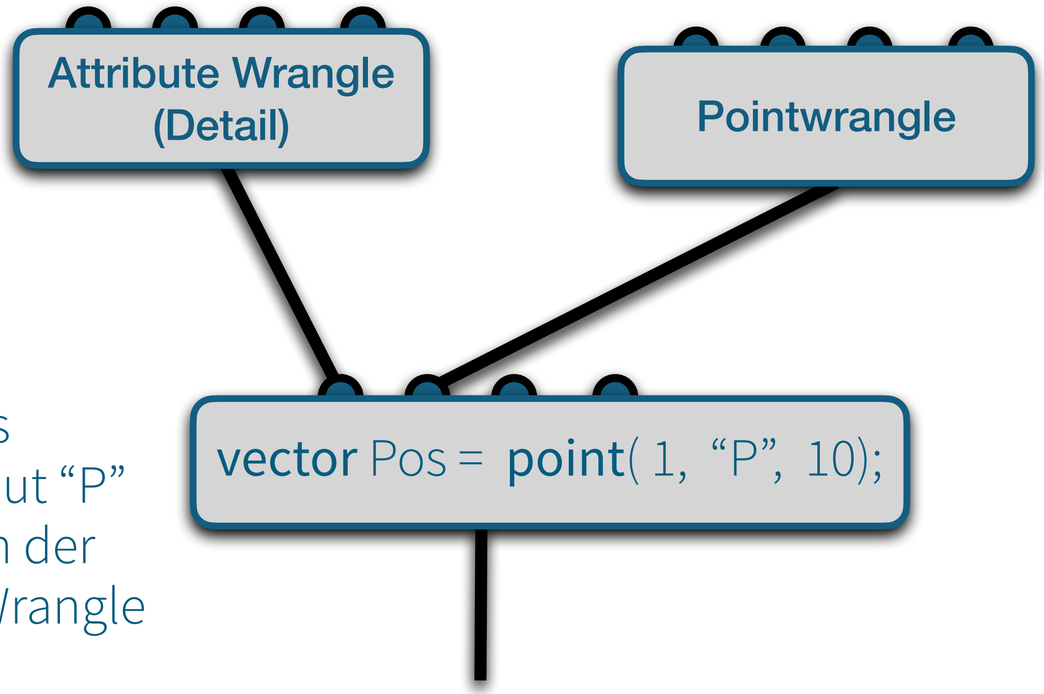
Funktionen

Funktionen folgen einer einfachen (standard) Syntax:

VEX-Funktionen

FunctionName(geohandle, Input1, Input2);

Beispiel:



Gibt uns den Wert des vektorwertigen Punktattribut “P” vom Punkt mit Index 10 in der Geometrie im 1. Input des Wrangle

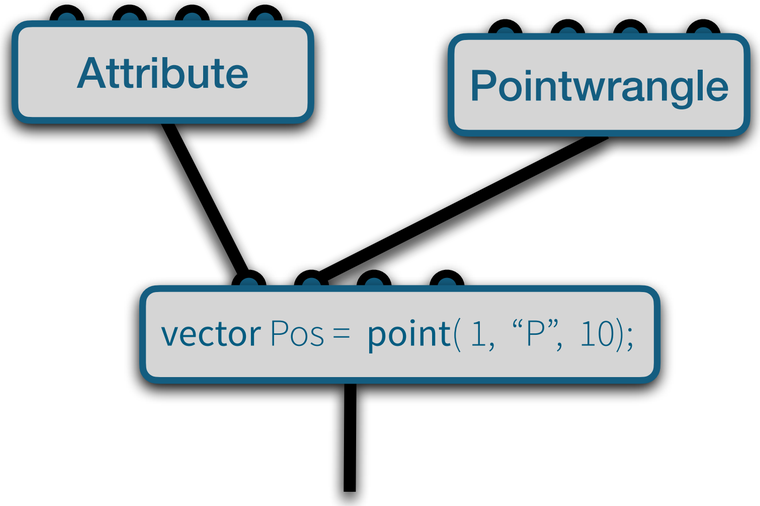
Funktionen

Funktionen folgen einer einfachen (standard) Syntax:

VEX-Funktionen

FunctionName(geohandle, Input1, Input2);

Beispiel:



Eigene Funktionen

```
OutputType ( InputType1 X, Y ; InputType2 c )
{
        VariableType1 Z = X + c*Y ;
        VarialbleType2 W = set(1,0,0) ;
        return Z + W ;
}
```

Funktionen

Übung:

a) Visualisiere die *Mandelbrot Menge*

Hinweis: Vorschriften, Formeln etc. finden wir oft auf Wikipedia.

Funktionen

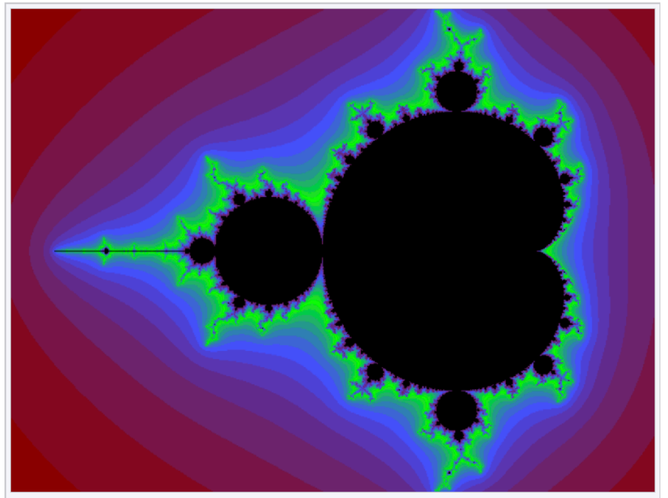
Mandelbrot-Menge

Die **Mandelbrot-Menge**, benannt nach Benoît Mandelbrot, ist die Menge der komplexen Zahlen c, für welche die durch die iterative Vorschrift $z_{n+1} = z_n^2 + c$ mit dem Anfangswert $z_0 = 0$ definierte Folge $z_0, z_1, z_2, z_3, \ldots$ endlich bleibt, d. h. beschränkt ist.

Interpretiert man die Mandelbrot-Menge (eine Teilmenge der Gaußschen Zahlenebenen) als geometrische Figur, so ergibt sie ein Fraktal, das im allgemeinen Sprachgebrauch oft *Apfelmännchen* genannt wird. Bilder berechnet man, indem man jedem Pixel (x, y) eines Bildes eine komplexe Zahl zuordnet $(c = c_0 + a \cdot x + bi \cdot y)$ und beginnend mit $z_0 = 0$ untersucht, ob und wann die Iterationen anfangen, zu „explodieren“. Bleiben die Werte klein, wird das Pixel häufig schwarz gefärbt, kommt es zu einer „Explosion“ der Zahlenwerte, wird die Anzahl der dafür notwendigen Iterationen als Farbe kodiert.

Die ersten mit einem Computer generierten Darstellungen wurden 1978 von Robert W. Brooks und Peter Matelski vorgestellt.[1] 1980 veröffentlichte Benoît Mandelbrot eine Arbeit über das Thema.[2] Später wurde sie von Adrien Douady und John Hamal Hubbard in einer Reihe grundlegender mathematischer Arbeiten systematisch untersucht.[3] Die mathematischen Grundlagen dafür wurden bereits 1905 von dem französischen Mathematiker Pierre Fatou erarbeitet.

Inhaltsverzeichnis [Verbergen]



Technische Universität Berlin

Copy To Points Knoten

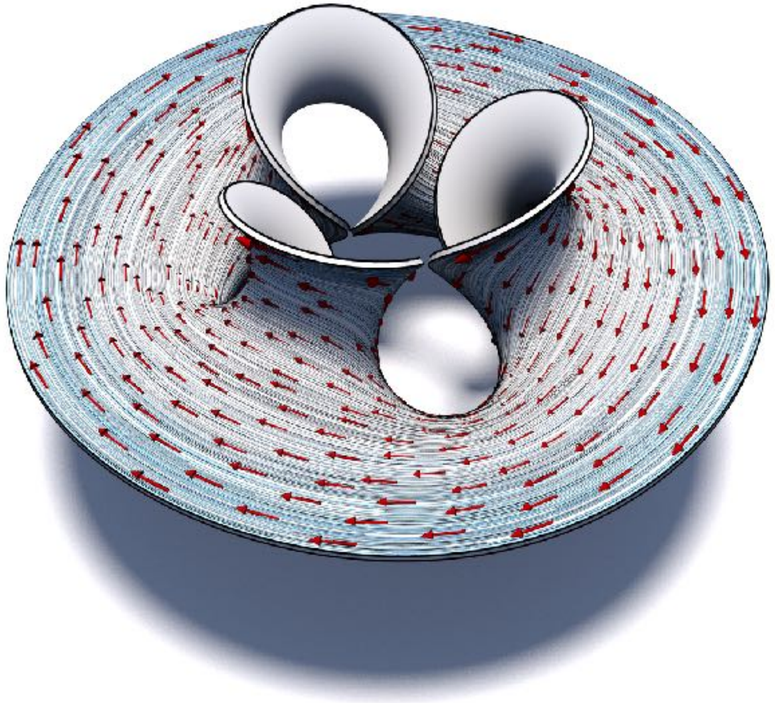
Berlin Mathematical School BMS

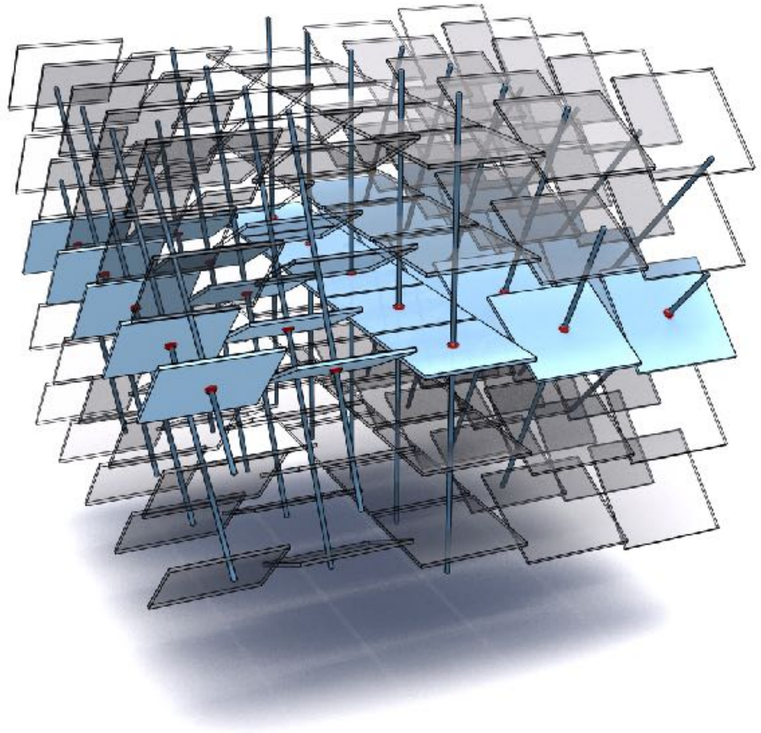
SideFX

DGD

SFB TRR 109 Discretization in Geometry and Dynamics

Wie machen wir solche Abbildungen?





Rotationen in Houdini

Rotationen werden in Houdini am besten durch Quaternionen beschrieben.

$$\mathbb{H} = \mathbb{R} \oplus i\mathbb{R} \oplus j\mathbb{R} \oplus k\mathbb{R} \quad \text{mit} \quad i^2 = j^2 = k^2 = -1$$

Eine Rotation um den Winkel α um die Rotationsachse $X \in S^2 \subset \mathbb{R}^3$ ist gegeben durch

$$Y \mapsto \bar{q}Yq \qquad \text{wobei} \qquad q = \sin(\frac{\alpha}{2}) + \cos(\frac{\alpha}{2})X.$$

Im Gegensatz zu komplexen Zahlen kann Houdini natürlicher Weise mit Quaternionen umgehen.

Rotationen in Houdini

Eine Rotation um den Winkel α um die Rotationsachse $X \in S^2 \subset \mathbb{R}^3$ ist gegeben durch

$$Y \mapsto \bar{q}Yq \qquad \text{wobei} \qquad q = \sin(\frac{\alpha}{2}) + \cos(\frac{\alpha}{2})X.$$

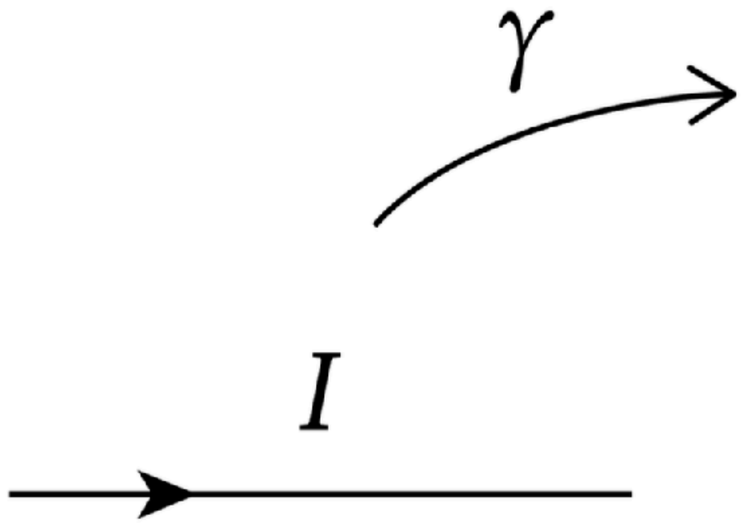
Im Gegensatz zu komplexen Zahlen kann Houdini natürlicher Weise mit Quaternionen umgehen.

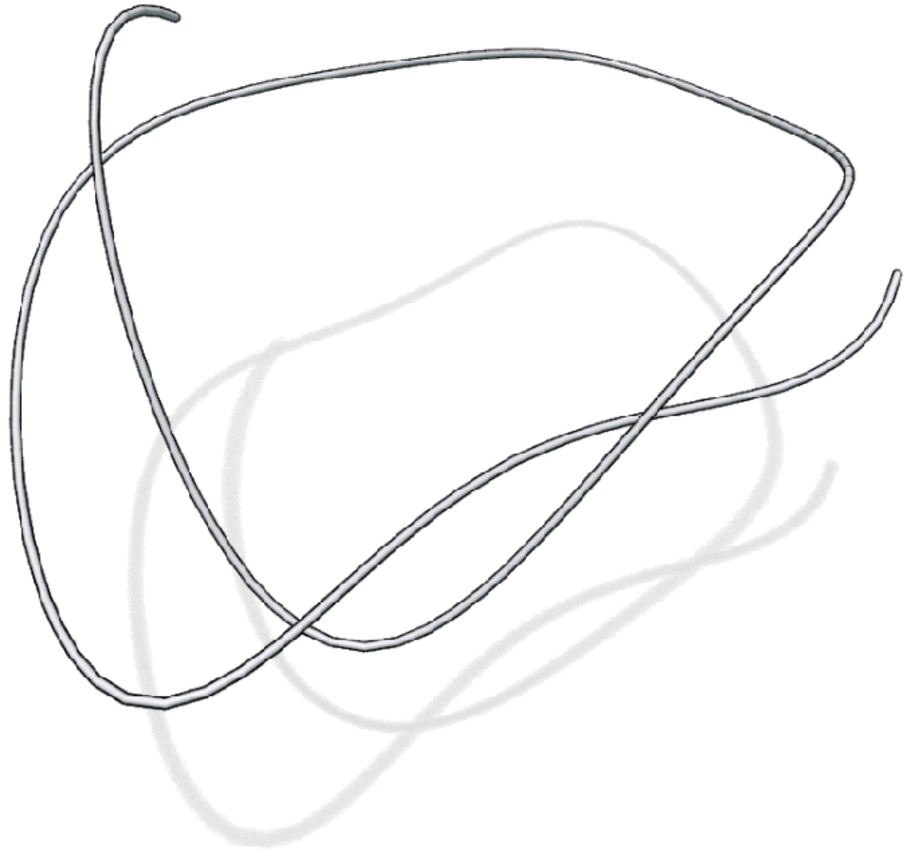
Bequeme Funktionen sind z.B.:

- quaternion$(\alpha, X) \mapsto q$
- qrotate$(q, X) \mapsto \bar{q}Xq$
- vector4 $q =$ dihedral(X, Y) mit $Y = \bar{q}Xq$
  - Das Punktattribut **“p@orient”** wird vom “Copy to Points”-Knoten benutzt

Frenet-Rahmen für Raumkurven

Eine Raumkurve ist eine Abbildung $\gamma\colon I \to \mathbb{R}^3$



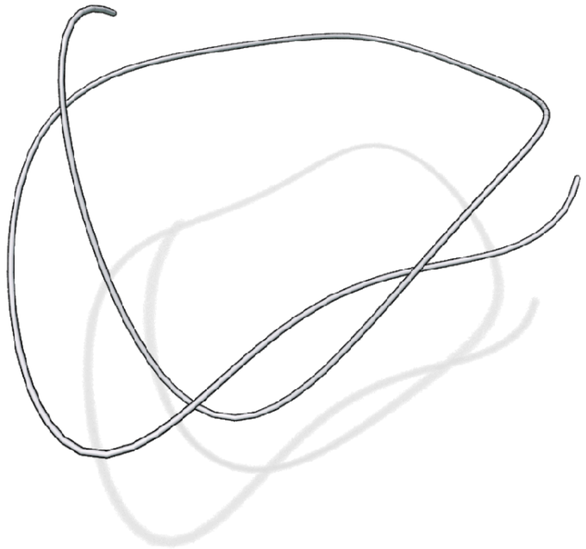


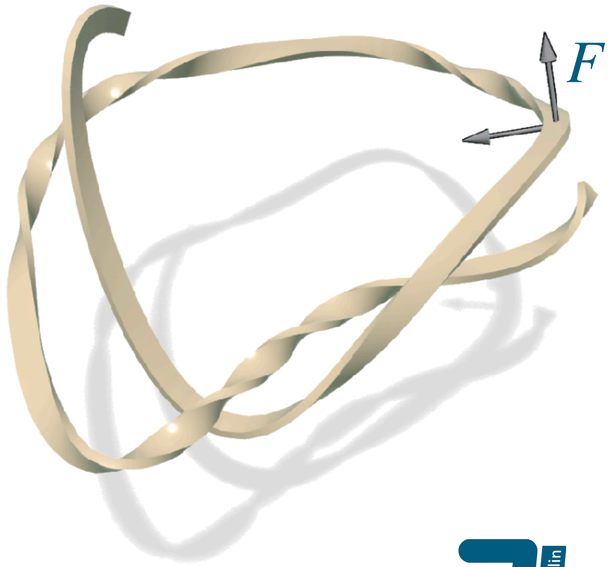
Frenet-Rahmen für Raumkurven

Eine Raumkurve ist eine Abbildung $\gamma\colon I \to \mathbb{R}^3$

Eine *gerahmte Kurve* ist eine Raumkurve zusammen mit einem *Rahmen*, d.h. eine Abbildung

$$F\colon I \to \mathrm{SO}(3)$$



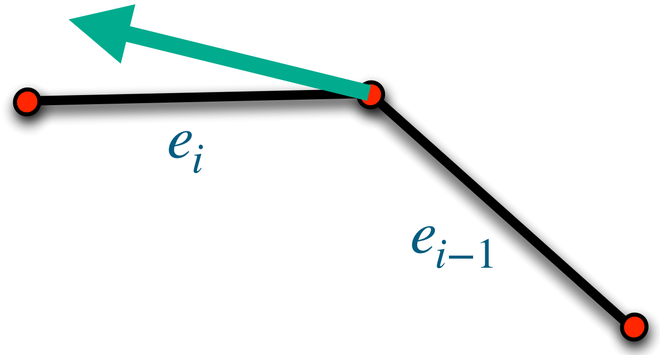


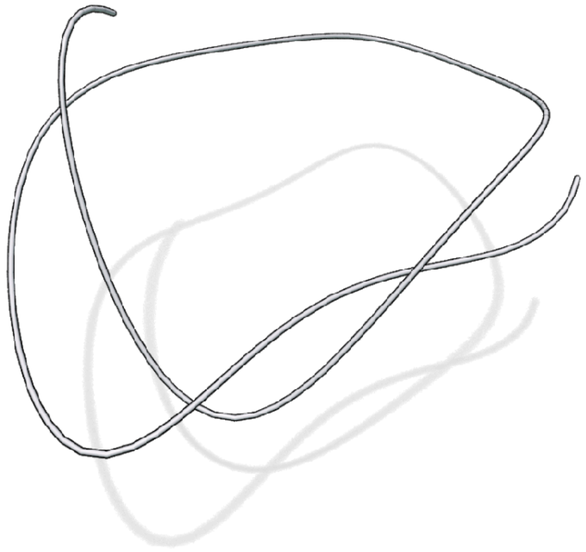
Frenet-Rahmen für Raumkurven

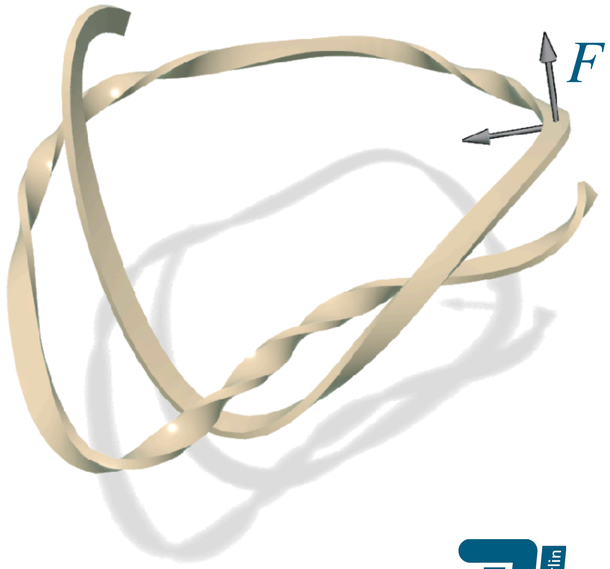
Eine Raumkurve ist eine Abbildung $\gamma\colon I \to \mathbb{R}^3$

Ein spezieller Rahmen $(T, N, B)\colon I \to \mathrm{SO}(3)$
is definiert durch

$$T_i := \text{normalize}\left(\frac{e_i}{|e_i|} + \frac{e_{i-1}}{|e_{i-1}|}\right)$$







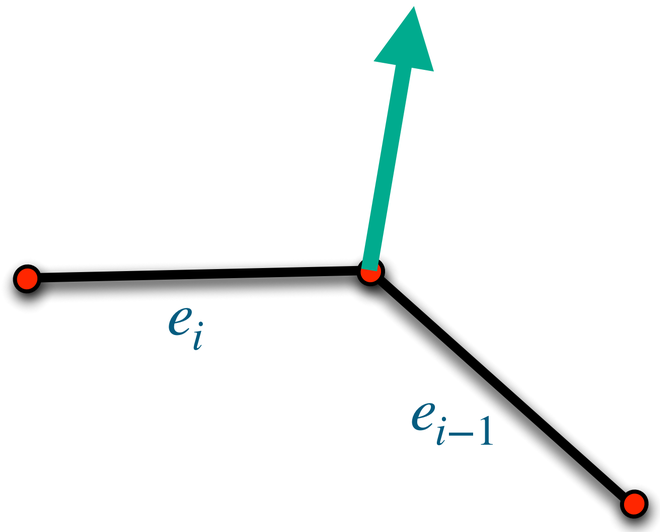
Frenet-Rahmen für Raumkurven

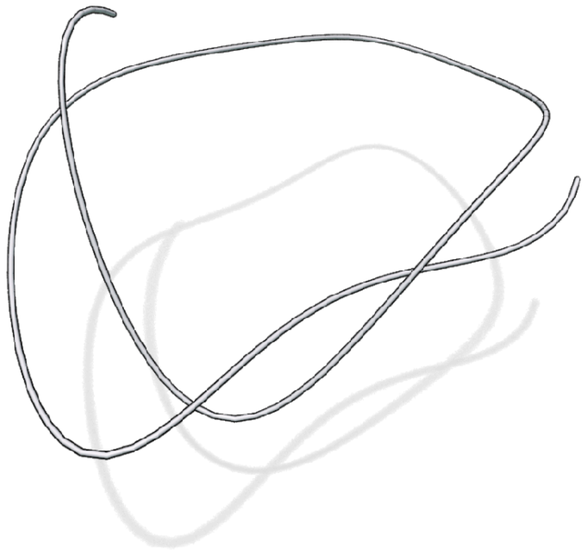
Eine Raumkurve ist eine Abbildung $\gamma\colon I \to \mathbb{R}^3$

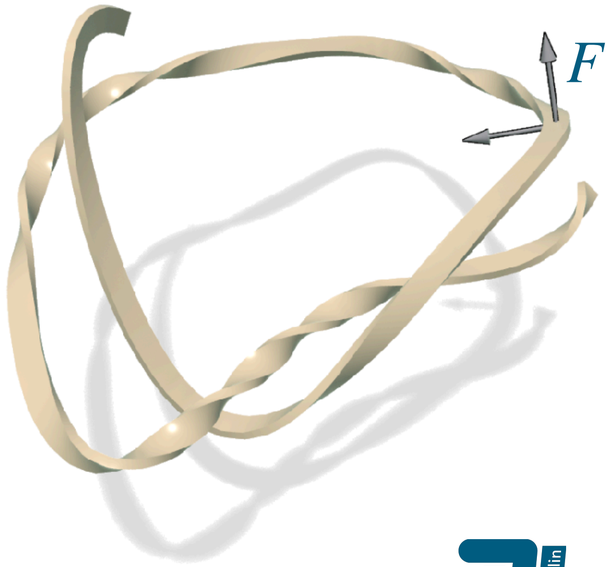
Ein spezieller Rahmen $(T, N, B)\colon I \to \mathrm{SO}(3)$
is definiert durch

$$T_i := \text{normalize}\left(\frac{e_i}{|e_i|} + \frac{e_{i-1}}{|e_{i-1}|}\right)$$

$$B_i := \text{normalize}\left(\frac{e_i}{|e_i|} \times \frac{e_{i-1}}{|e_{i-1}|}\right)$$







Frenet-Rahmen für Raumkurven

Eine Raumkurve ist eine Abbildung $\gamma\colon I \to \mathbb{R}^3$

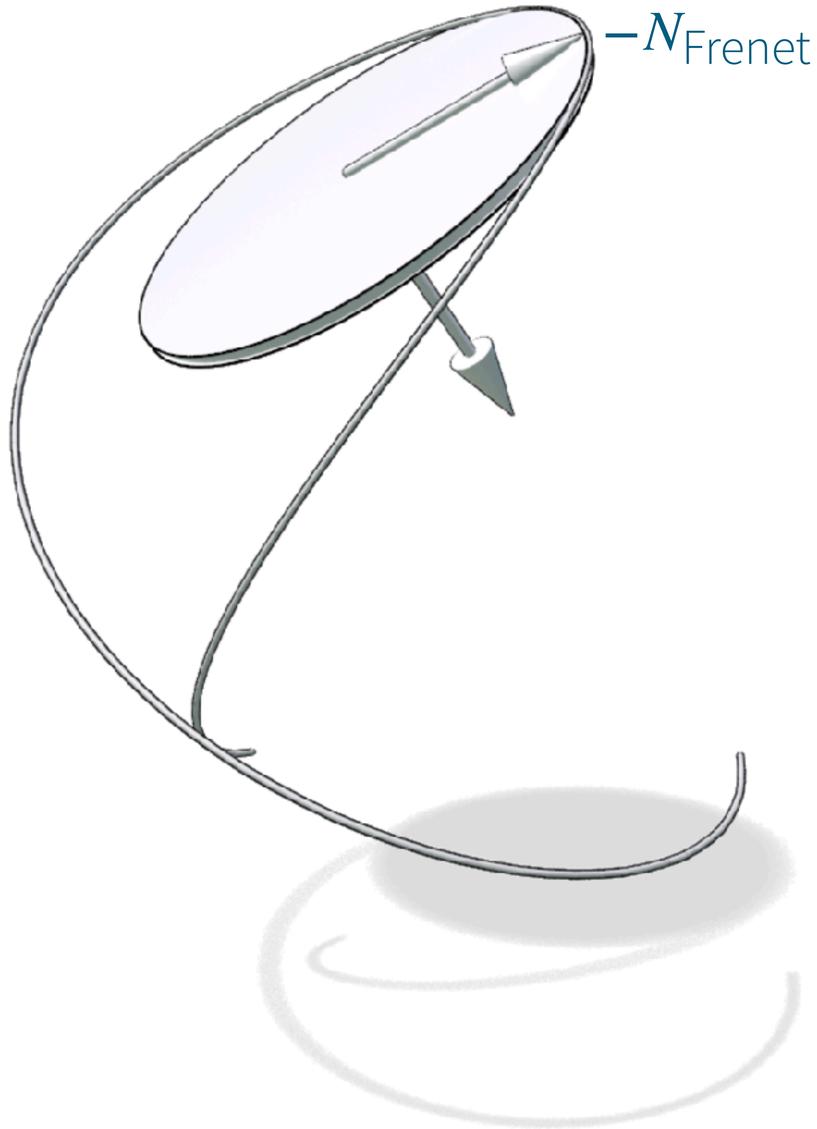
Ein spezieller Rahmen $(T, N, B)\colon I \to \mathrm{SO}(3)$
is definiert durch

$$T_i := \text{normalize}\left(\frac{e_i}{|e_i|} + \frac{e_{i-1}}{|e_{i-1}|}\right)$$

$$B_i := \text{normalize}\left(\frac{e_i}{|e_i|} \times \frac{e_{i-1}}{|e_{i-1}|}\right)$$

N_i definiert durch $B_i = T_i \times N_i$

Frenet-Rahmen



Frenet-Rahmen für Raumkurven

Eine Raumkurve ist eine Abbildung $\gamma\colon I \to \mathbb{R}^3$

Ein spezieller Rahmen $(T, N, B)\colon I \to \mathrm{SO}(3)$
is definiert durch

$T_i := \text{normalize}(\frac{e_i}{|e_i|} + \frac{e_{i-1}}{|e_{i-1}|})$

$B_i := \text{normalize}(\frac{e_i}{|e_i|} \times \frac{e_{i-1}}{|e_{i-1}|})$

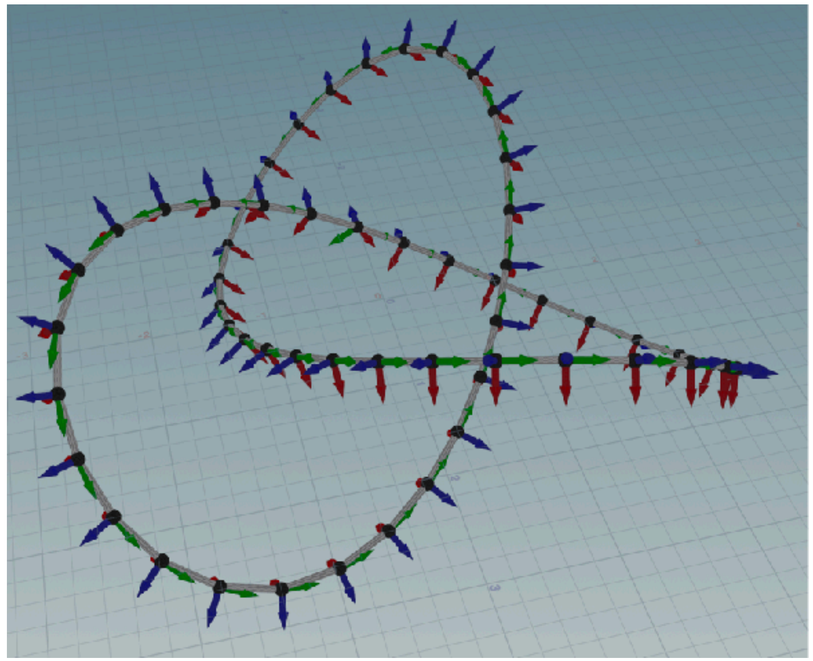
N_i definiert durch $B_i = T_i \times N_i$

Frenet-Rahmen

Rendering

Übung:

a) Visualisiere den Frenet Rahmen einer geschlossenen Raumkurve mit Hilfe von einem *“Copy to Points”*-Knoten und den Attributen *p@orient* und *f@pscale*



Technische Universität Berlin

Animationen

Berlin Mathematical School BMS

SideFX

DGD

SFB TRR 109 Discretization in Geometry and Dynamics

Animationen

Eine Frame Abhängigkeit der aktuellen Szene kann für Animationen genutzt werden:

Animationen

Eine Frame Abhängigkeit der aktuellen Szene kann für Animationen genutzt werden:

Animationen

Eine Frame Abhängigkeit der aktuellen Szene kann für Animationen genutzt werden:

- **Keyframes**

→ über **option + Click** in den jeweiligen parameter

Animationen

Eine Frame Abhängigkeit der aktuellen Szene kann für Animationen genutzt werden:

- **Keyframes**

→ über **option + Click** in den jeweiligen parameter

Animationen

Eine Frame Abhängigkeit der aktuellen Szene kann für Animationen genutzt werden:

- **Keyframes**
- **Hscript**

→ über **$F** in Parameter Variablen im Parameter view

Animationen

Eine Frame Abhängigkeit der aktuellen Szene kann für Animationen genutzt werden:

- **Keyframes**
- **Hscript**

→ über **$F** in Parameter Variablen im Parameter view

Animationen

Eine Frame Abhängigkeit der aktuellen Szene kann für Animationen genutzt werden:

- Keyframes
- Hscript
- VEX

→ über **@Frame** in VEX-code

Animationen

Eine Frame Abhängigkeit der aktuellen Szene kann für Animationen genutzt werden:

- Keyframes
- Hscript
- VEX

→ über **@Frame** in VEX-code

Animationen

Eine Frame Abhängigkeit der aktuellen Szene kann für Animationen genutzt werden:

- Keyframes
- Hscript
- VEX
- Solver Knoten

Animationen

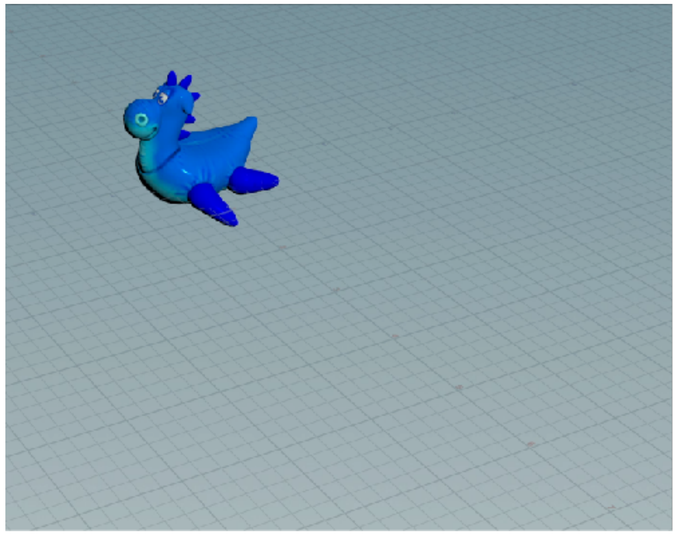
Eine Frame Abhängigkeit der aktuellen Szene kann für Animationen genutzt werden:

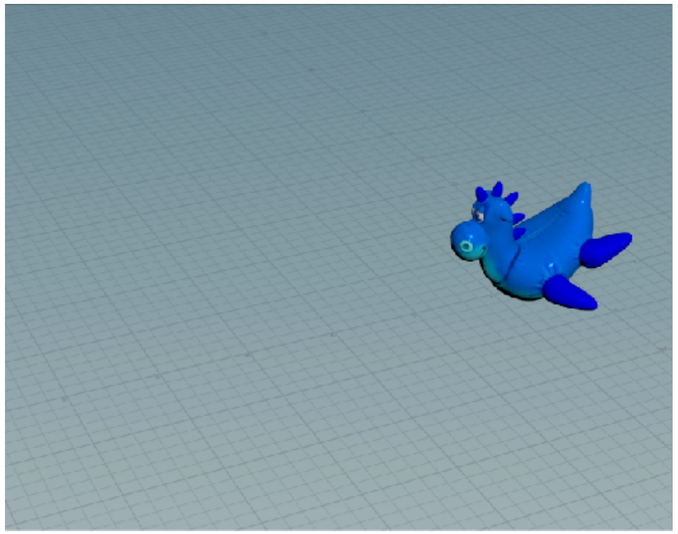
- Keyframes
- Hscript
- VEX
- Solver Knoten

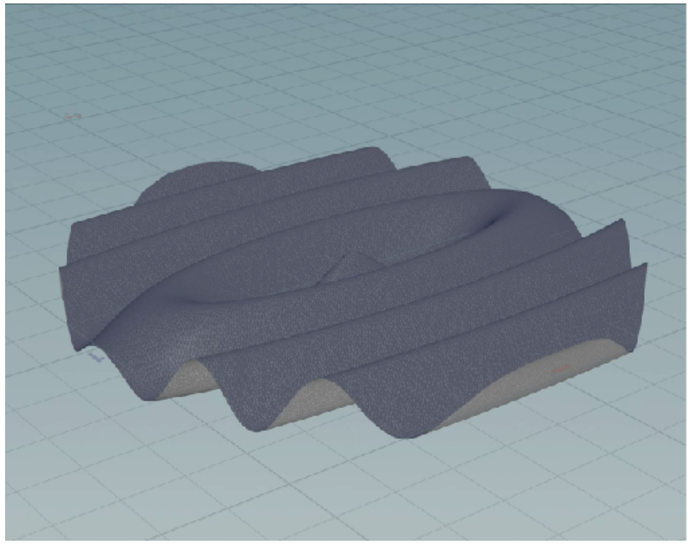
Animation

Aufgabe:

a) Animiere folgende Animationen



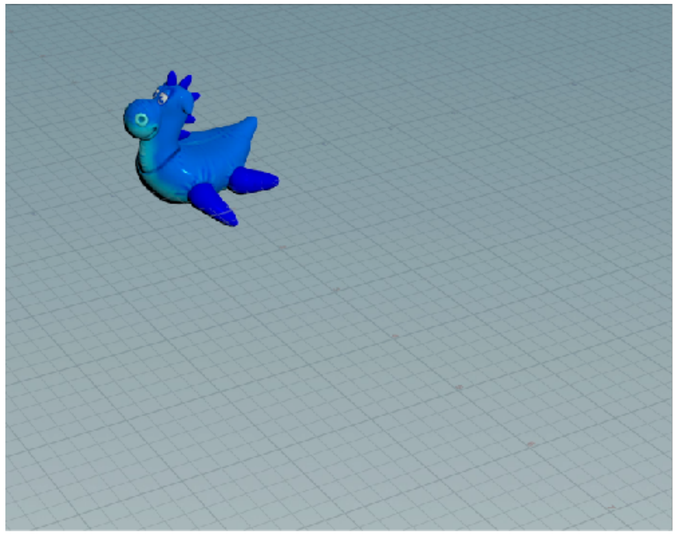


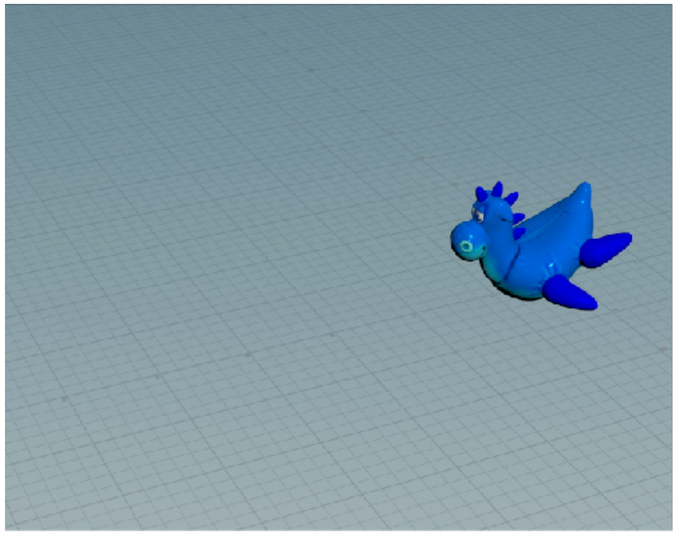


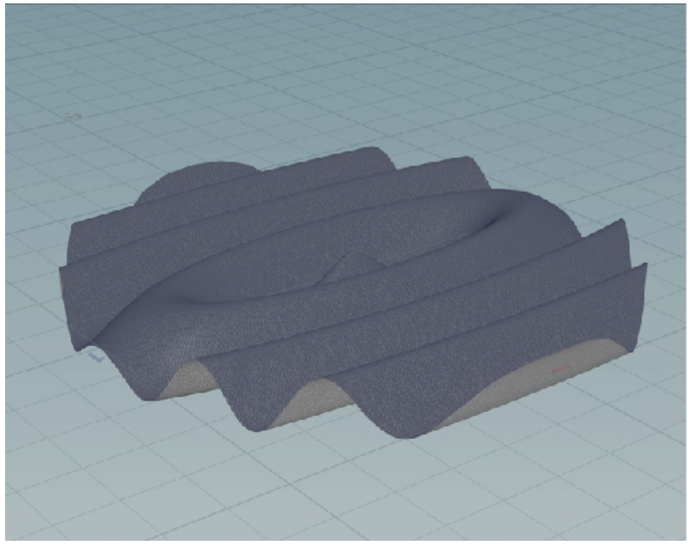
Animation

Aufgabe:

a) Animiere folgende Animationen





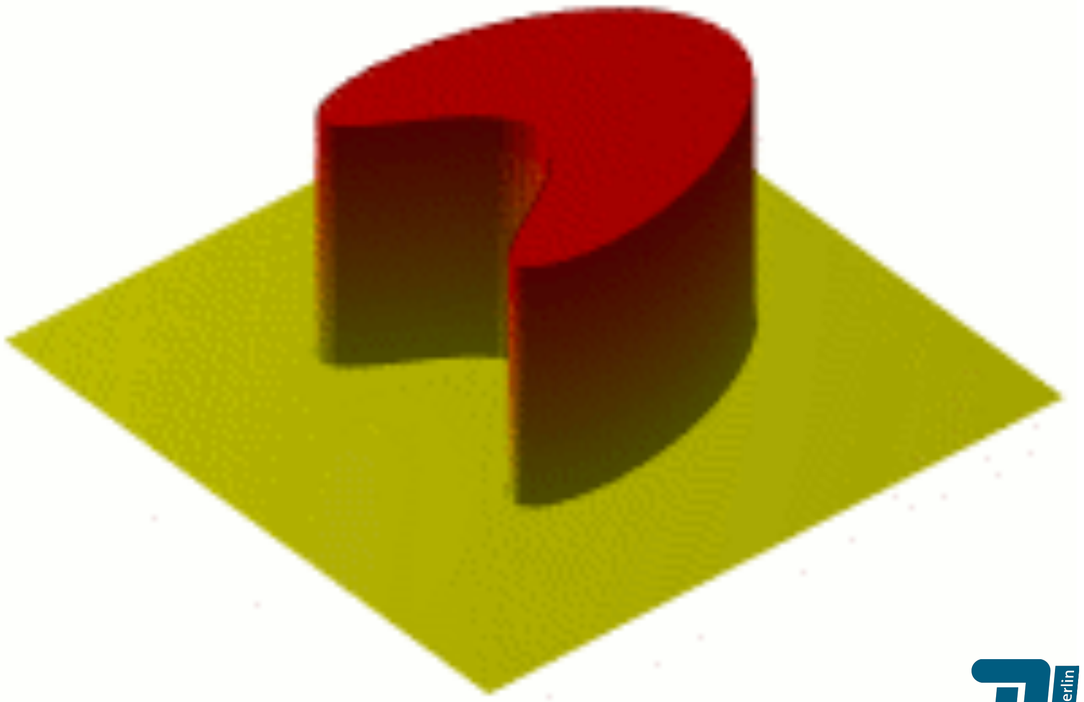


Wärmeleitungsgleichung

Die Wärmeleitungsgleichung is gegeben durch

$$\frac{\partial}{\partial t}u = \rho\,\Delta u,$$

wobei für ein $u\colon [a,b] \to \mathbb{R}$ ist $\Delta u = \frac{\partial^2}{\partial x^2}u$.

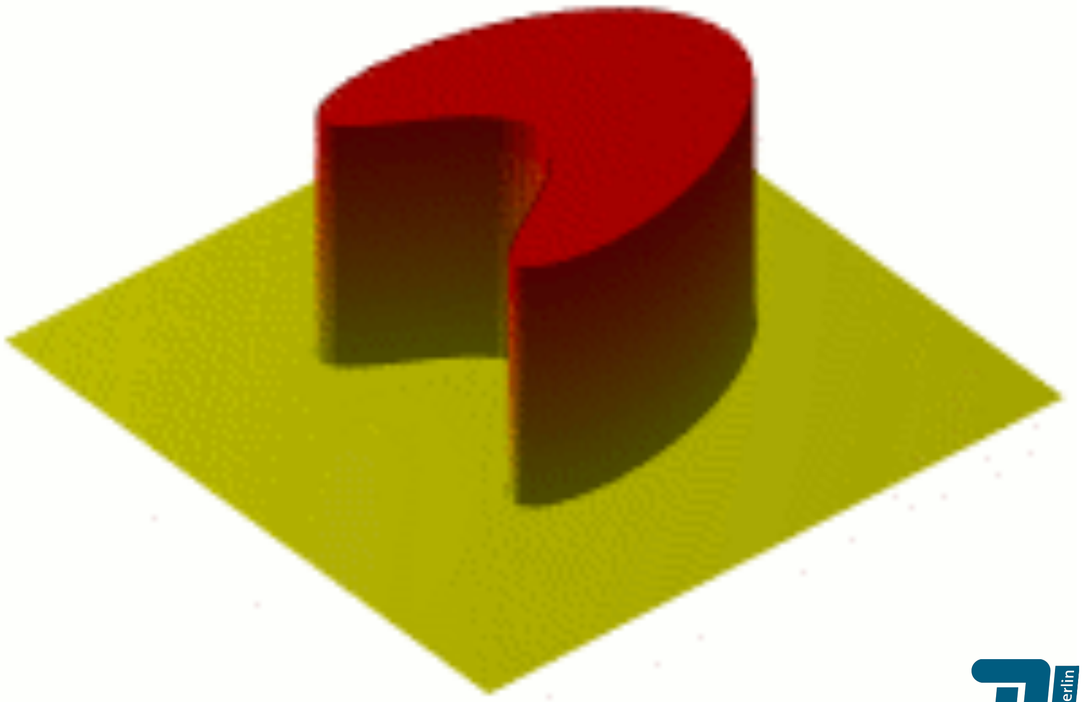


Wärmeleitungsgleichung

Die Wärmeleitungsgleichung is gegeben durch

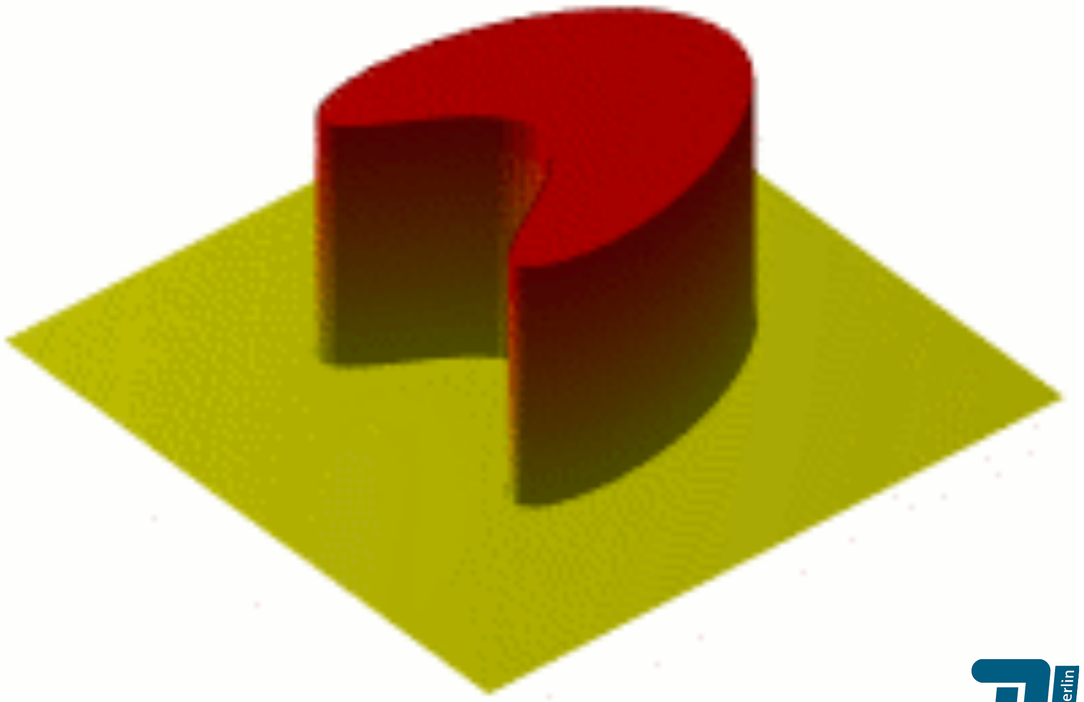
$$\frac{\partial}{\partial t}u = \rho\,\Delta u,$$

wobei für ein $u\colon [a,b] \to \mathbb{R}$ ist $\Delta u = \frac{\partial^2}{\partial x^2}u$.



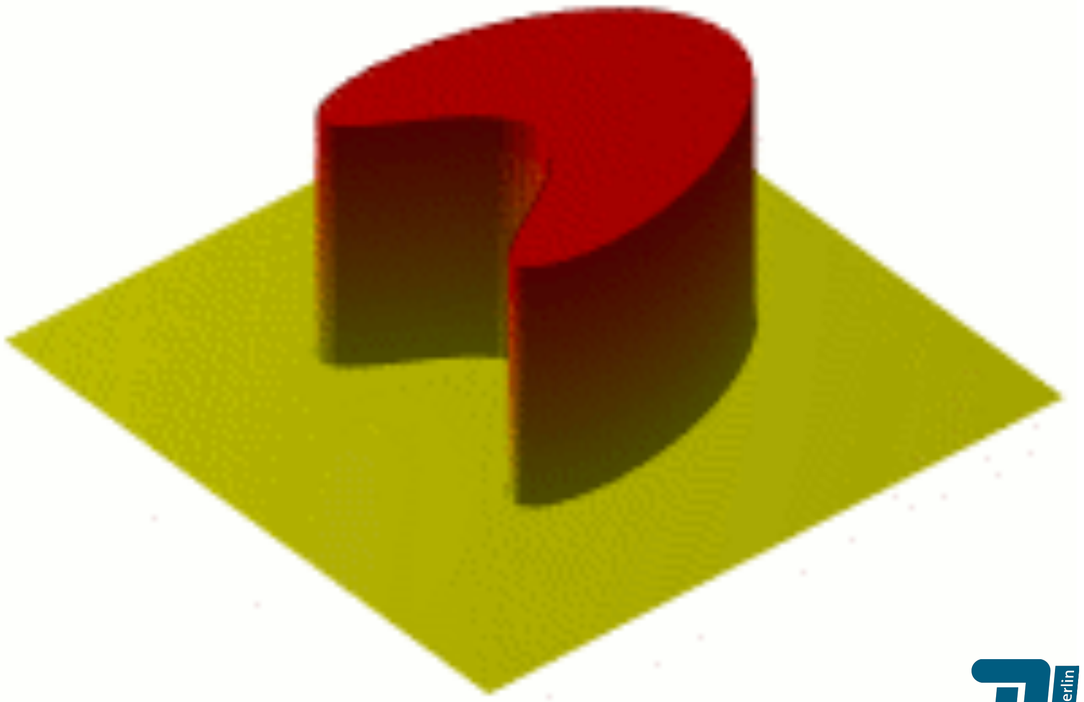
Wärmeleitungsgleichung

$$\frac{\partial}{\partial t}u = \rho \frac{\partial^2}{\partial x^2}u$$



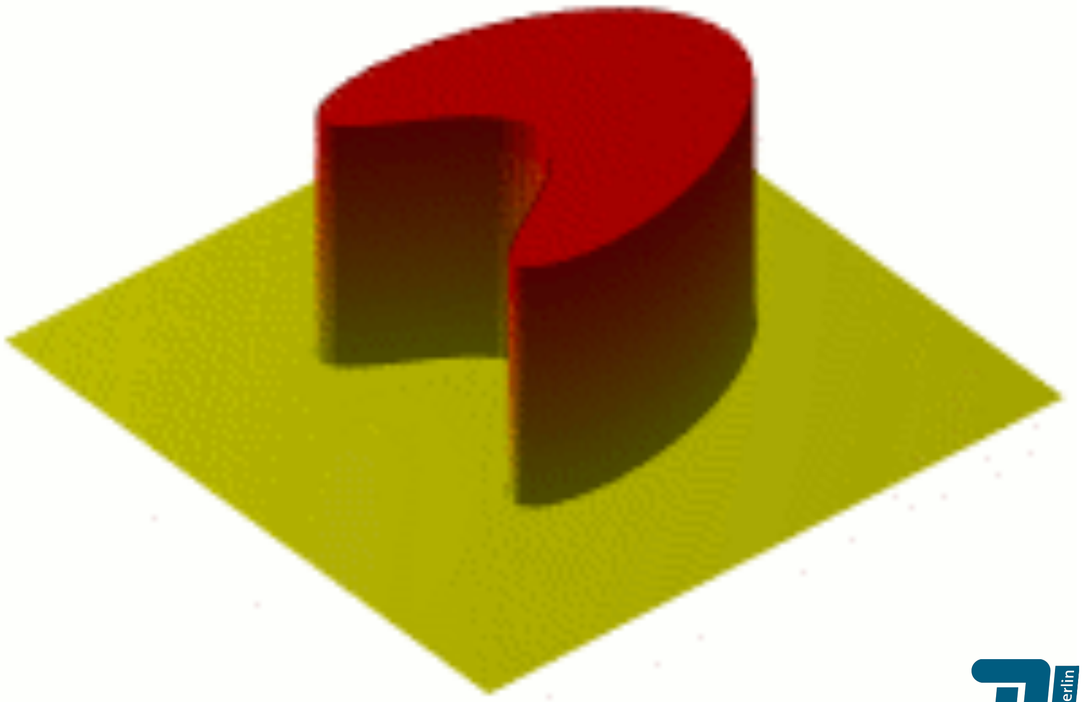
Wärmeleitungsgleichung

$$\frac{\partial}{\partial t}u = \rho \frac{\partial^2}{\partial x^2}u$$



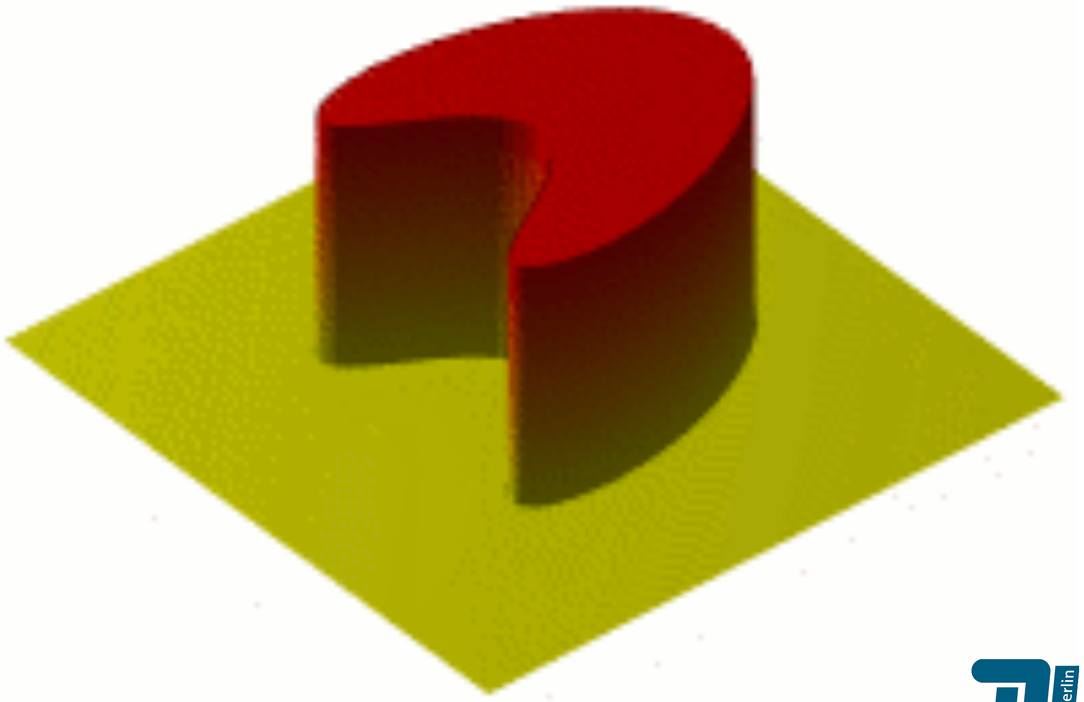
Wärmeleitungsgleichung

$$\frac{u_x^{t+1} - u_x^t}{\Delta T} = \rho \frac{\partial^2}{\partial x^2} u$$



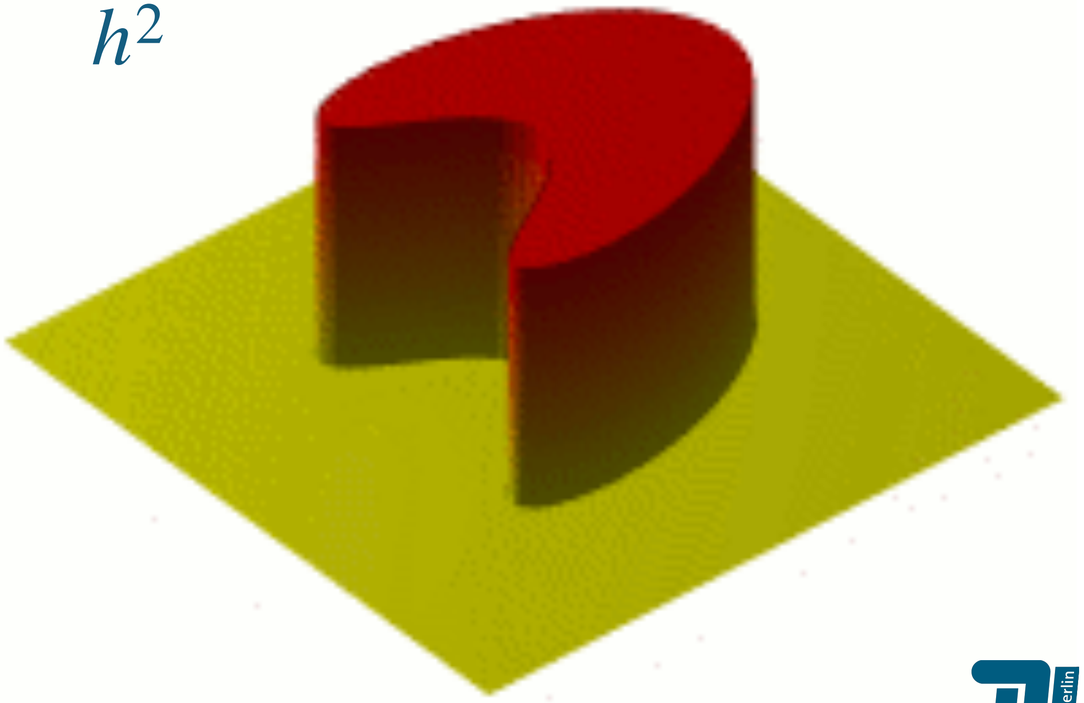
Wärmeleitungsgleichung

$$\frac{u_x^{t+1} - u_x^t}{\Delta T} = \rho \frac{\partial^2}{\partial x^2} u$$



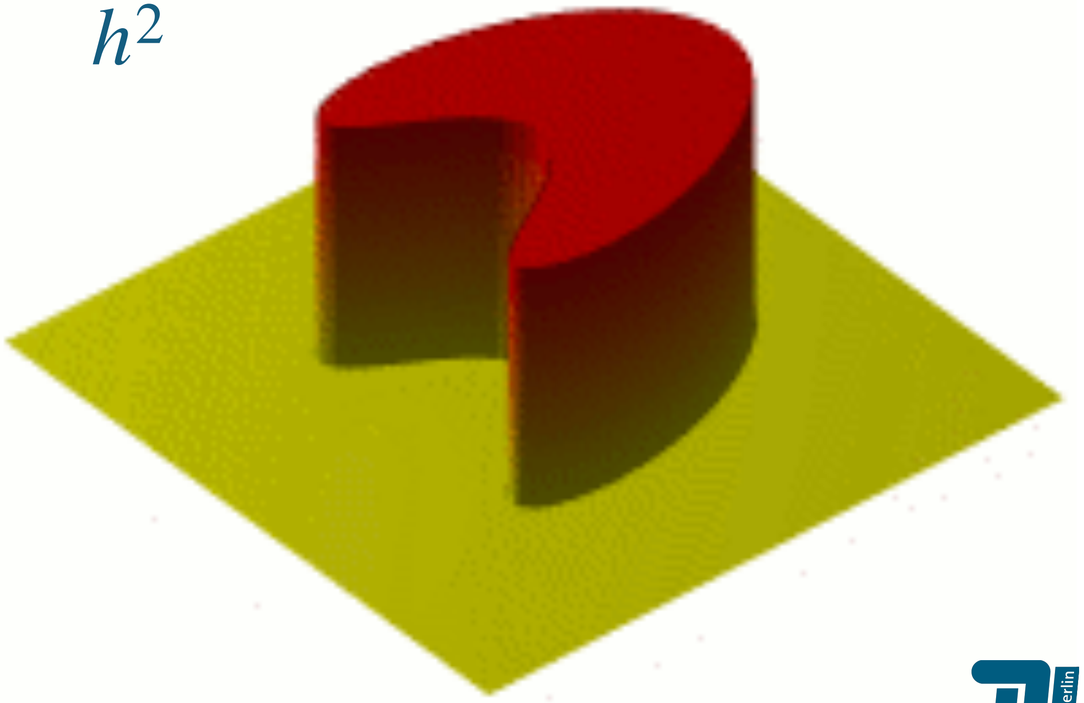
Wärmeleitungsgleichung

$$\frac{u_x^{t+1} - u_x^t}{\Delta T} = \rho \frac{u_{x+1}^t - 2u_x^t + u_{x-1}^t}{h^2}$$



Wärmeleitungsgleichung

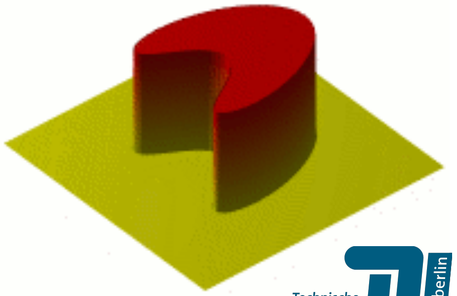
$$\frac{u_x^{t+1} - u_x^t}{\Delta T} = \rho \frac{u_{x+1}^t - 2u_x^t + u_{x-1}^t}{h^2}$$



Wärmeleitungsgleichung

$$\frac{u_x^{t+1} - u_x^t}{\Delta T} = \rho \frac{u_{x+1}^t - 2u_x^t + u_{x-1}^t}{h^2}$$

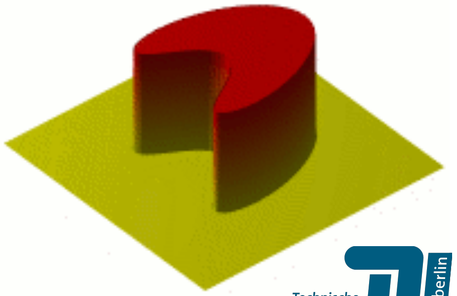
$$u_x^{t+1} = u_x^t + \frac{\rho \Delta T}{h^2}(u_{x+1}^t - 2u_x^t + u_{x-1}^t)$$



Wärmeleitungsgleichung

$$\frac{u_x^{t+1} - u_x^t}{\Delta T} = \rho \frac{u_{x+1}^t - 2u_x^t + u_{x-1}^t}{h^2}$$

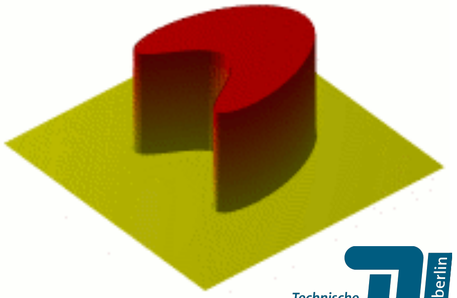
$$u_x^{t+1} = u_x^t + \frac{\rho \Delta T}{h^2}(u_{x+1}^t - 2u_x^t + u_{x-1}^t)$$



Wärmeleitungsgleichung

$$u_x^{t+1} = u_x^t + k(u_{x+1}^t - 2u_x^t + u_{x-1}^t)$$

"Forward Euler"- Methode mit Schrittweite $k > 0$.

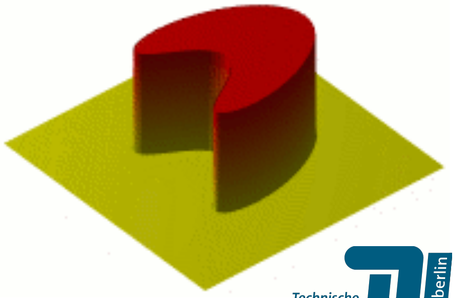


Technische Universität Berlin

Wärmeleitungsgleichung

$$u_x^{t+1} = u_x^t + k(u_{x+1}^t - 2u_x^t + u_{x-1}^t)$$

"Forward Euler"- Methode mit Schrittweite $k > 0$.



Technische Universität Berlin

Animation

Übung:

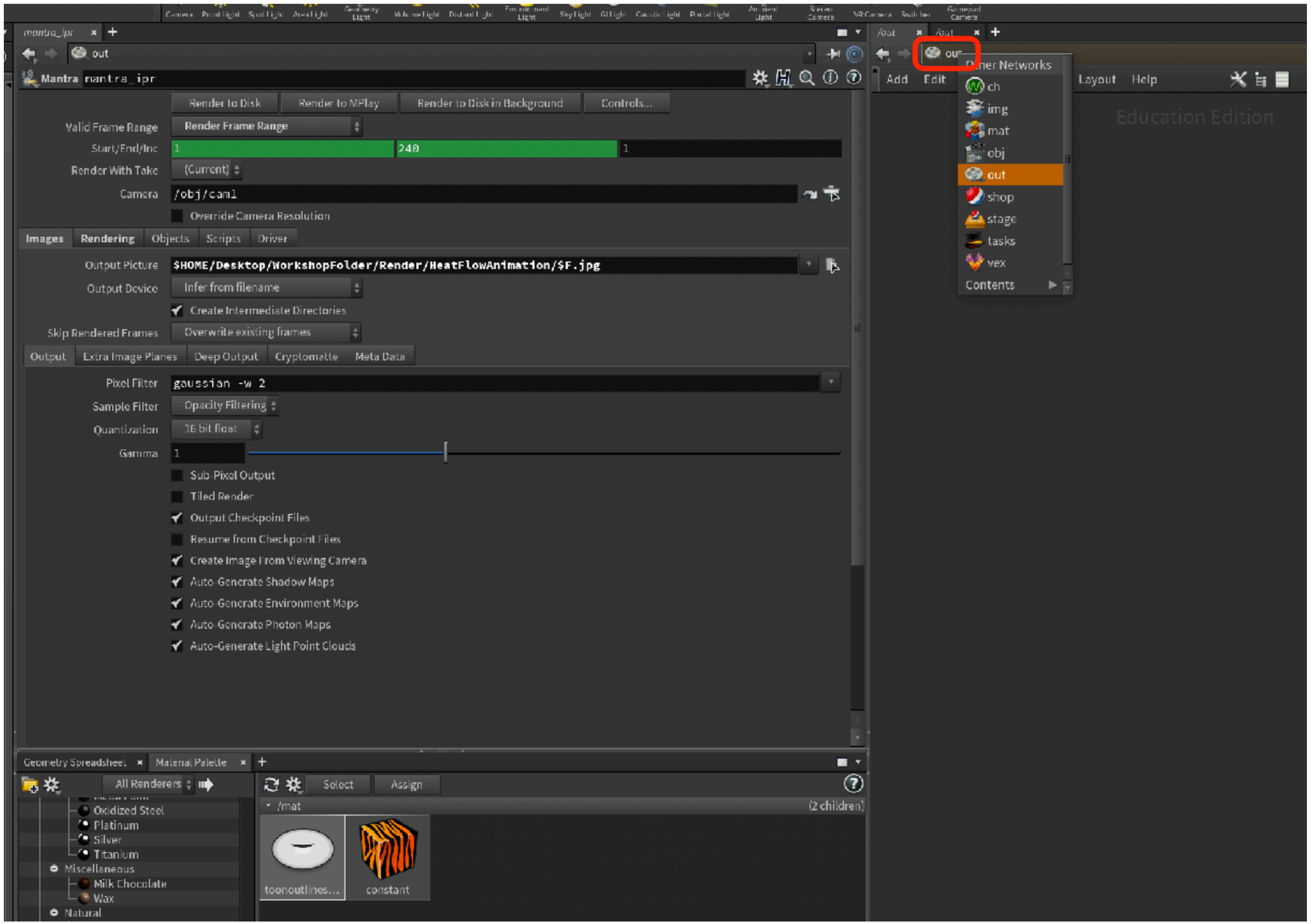
a) Nutze einen “Solver”-Knoten um forward-Euler heat flow in 1-D zu visualisieren.

   (1) Mit freier Randbedingung

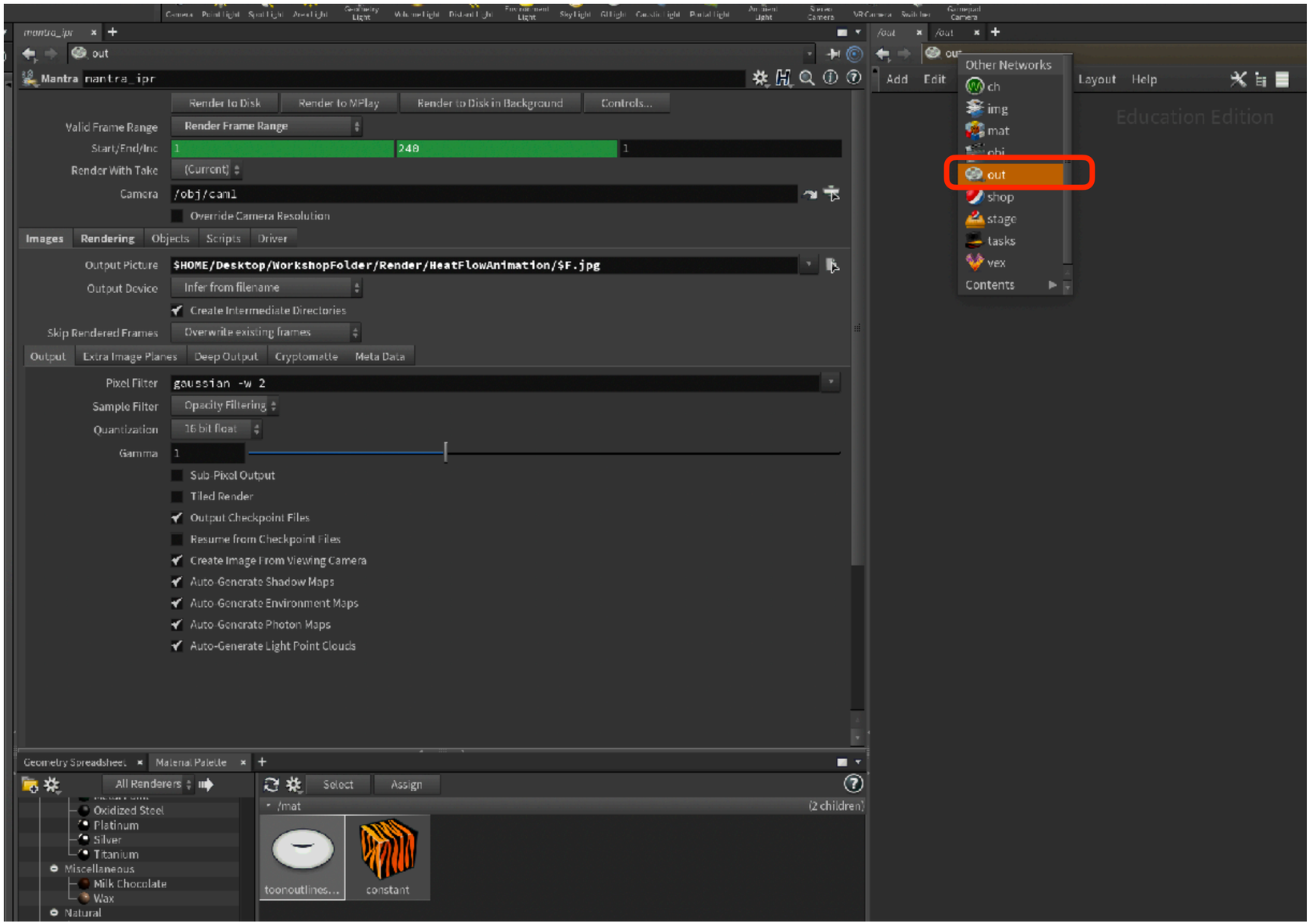
   (2) Mit festgehaltener Randbedingung

   (3) Mit Periodischer Randbedingung

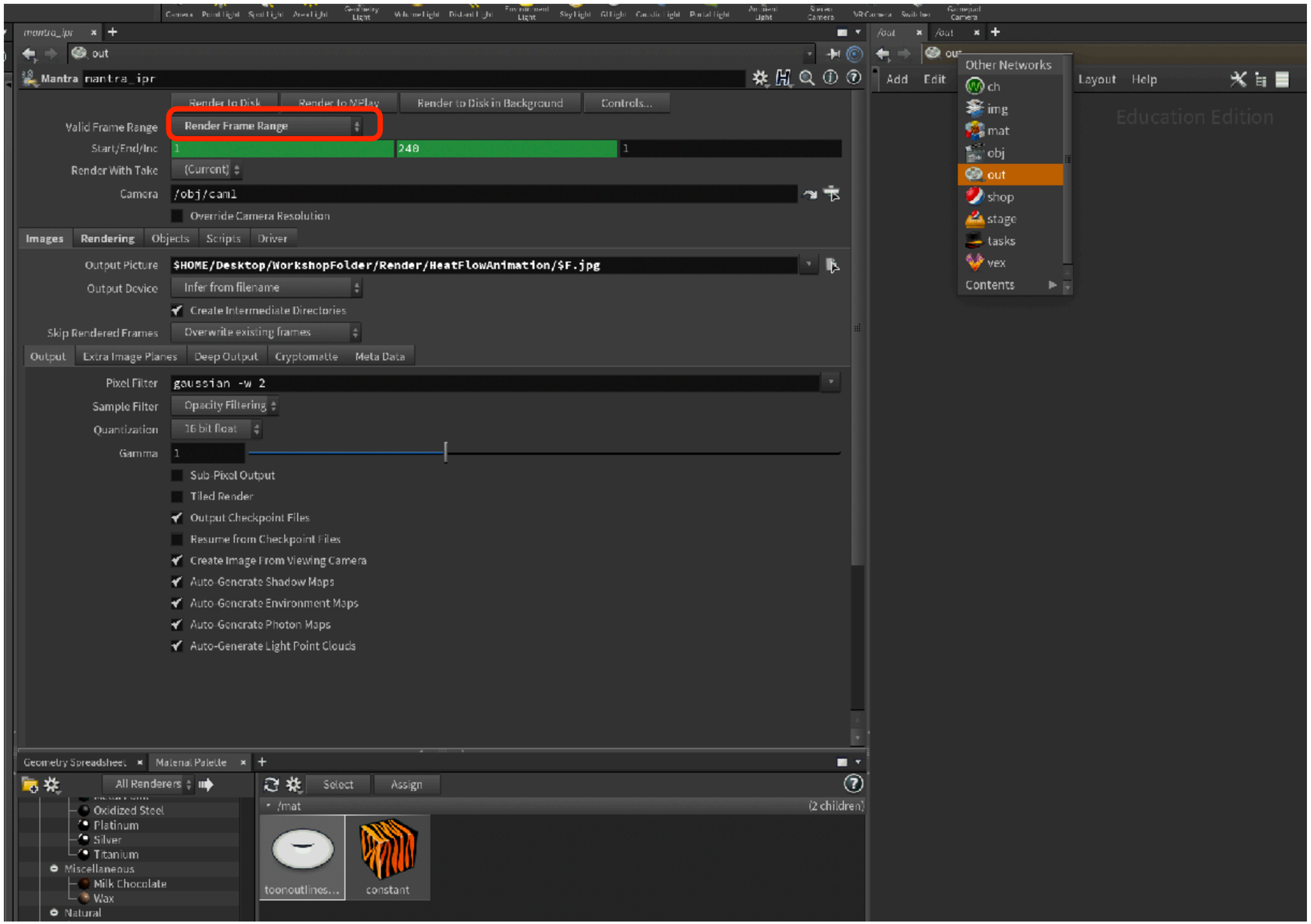
Animationen Rendern



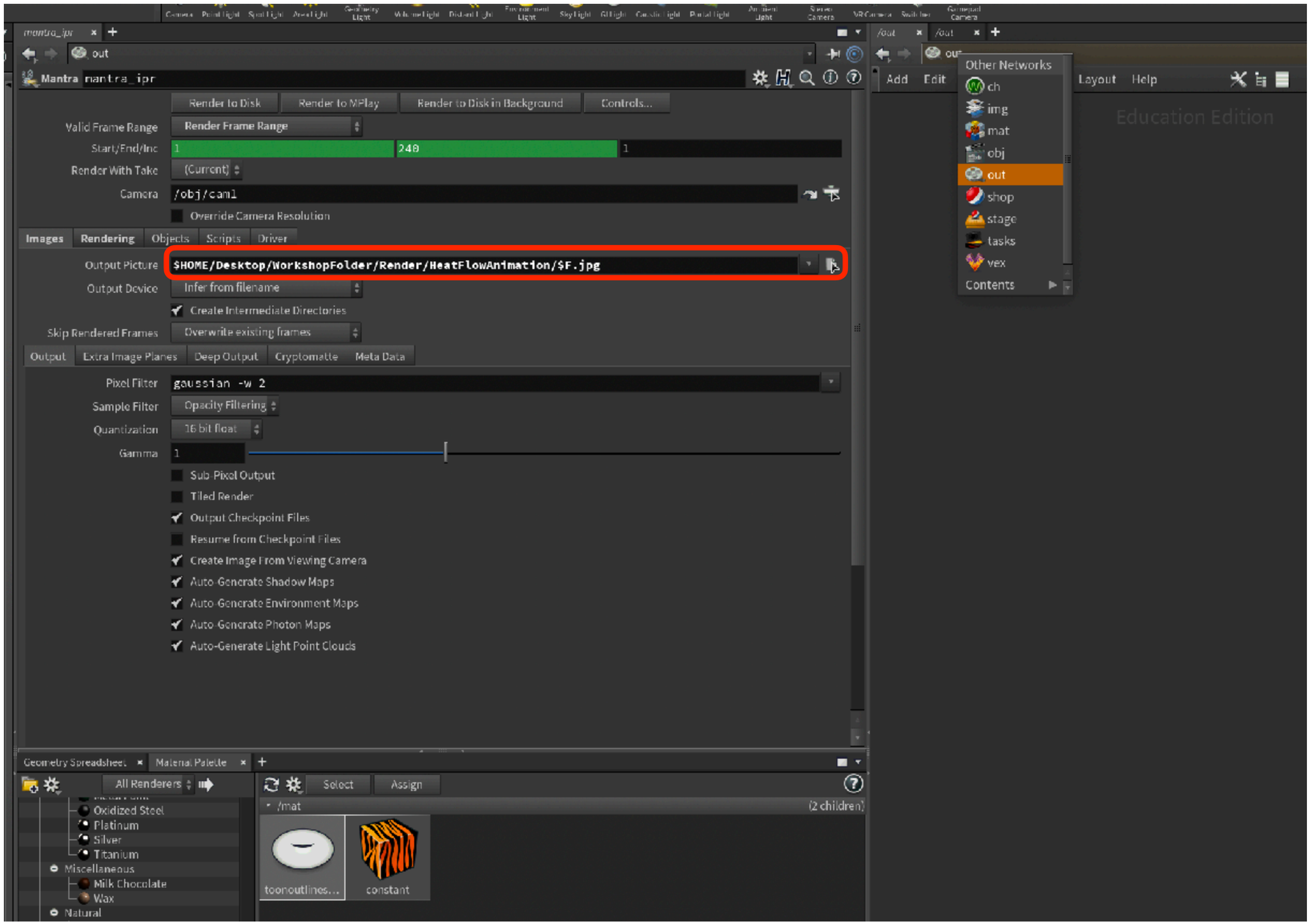
Animationen Rendern



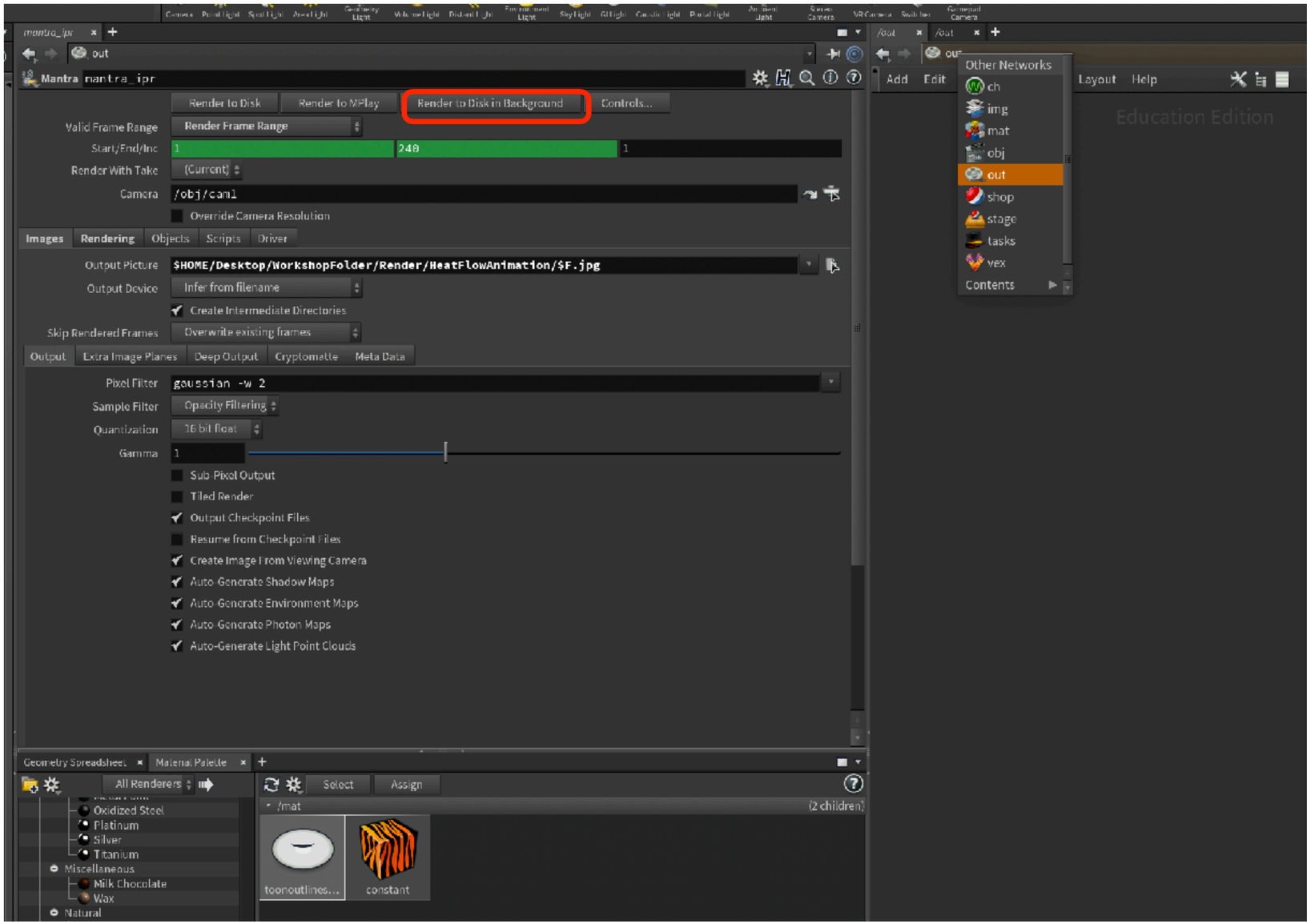
Animationen Rendern



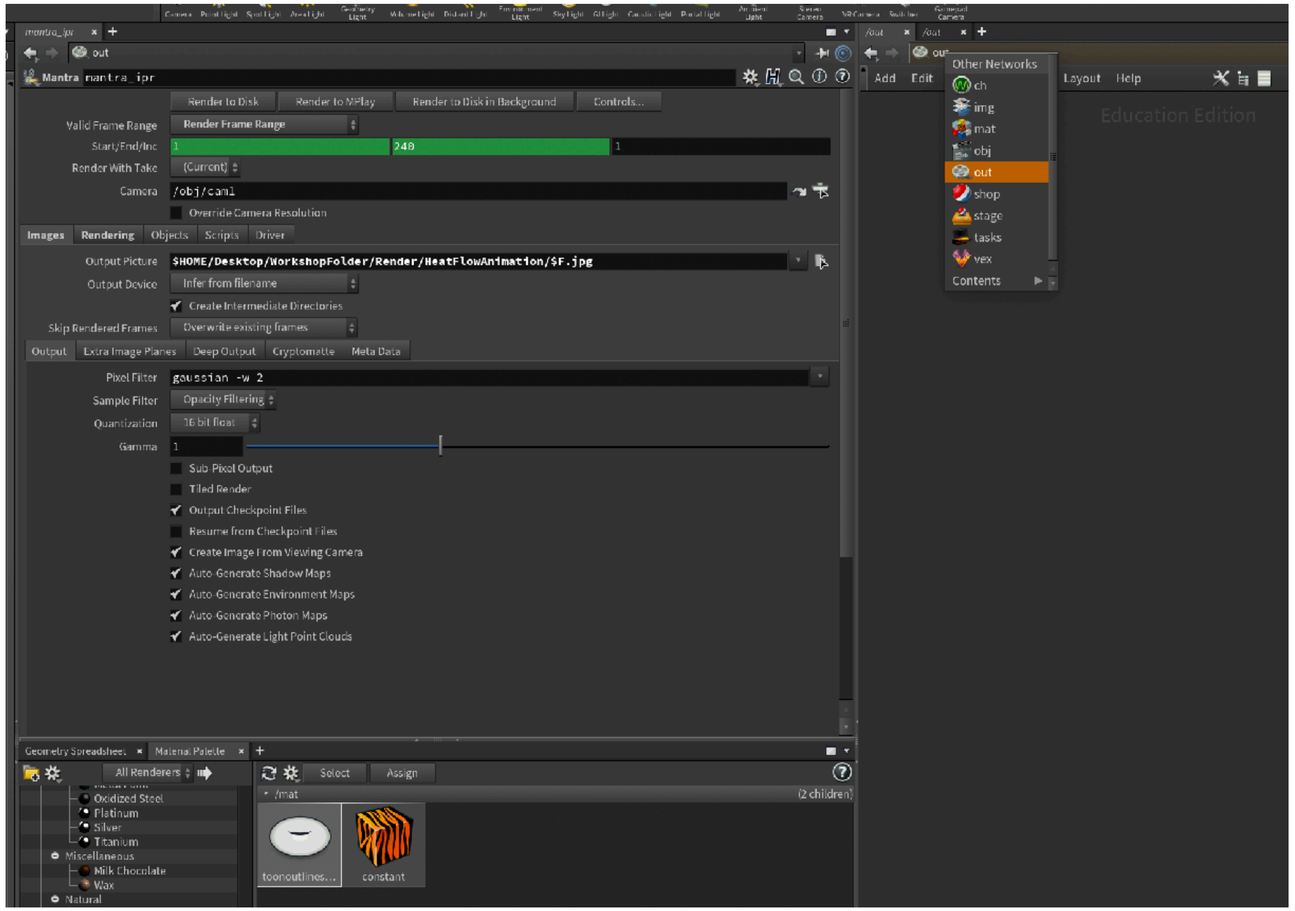
Animationen Rendern

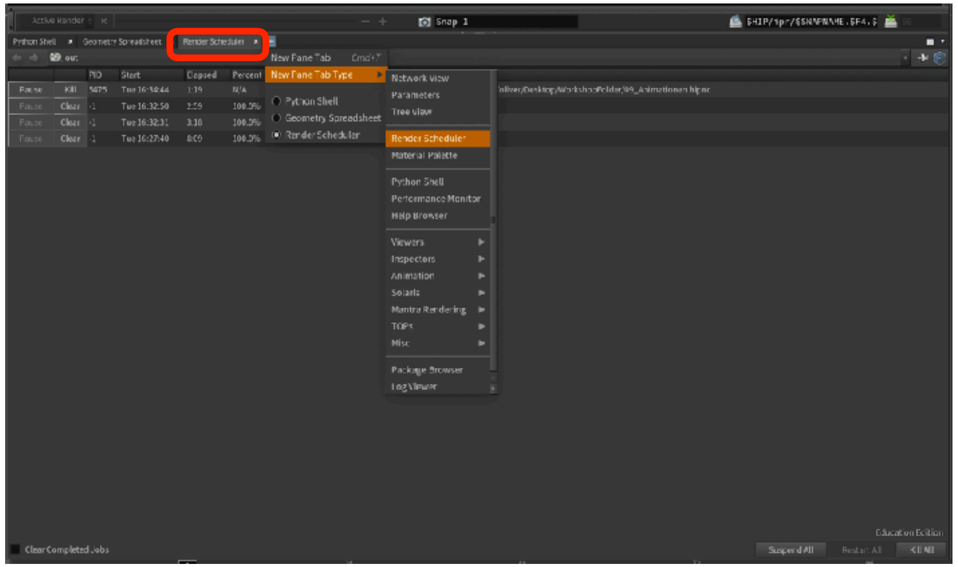


Animationen Rendern



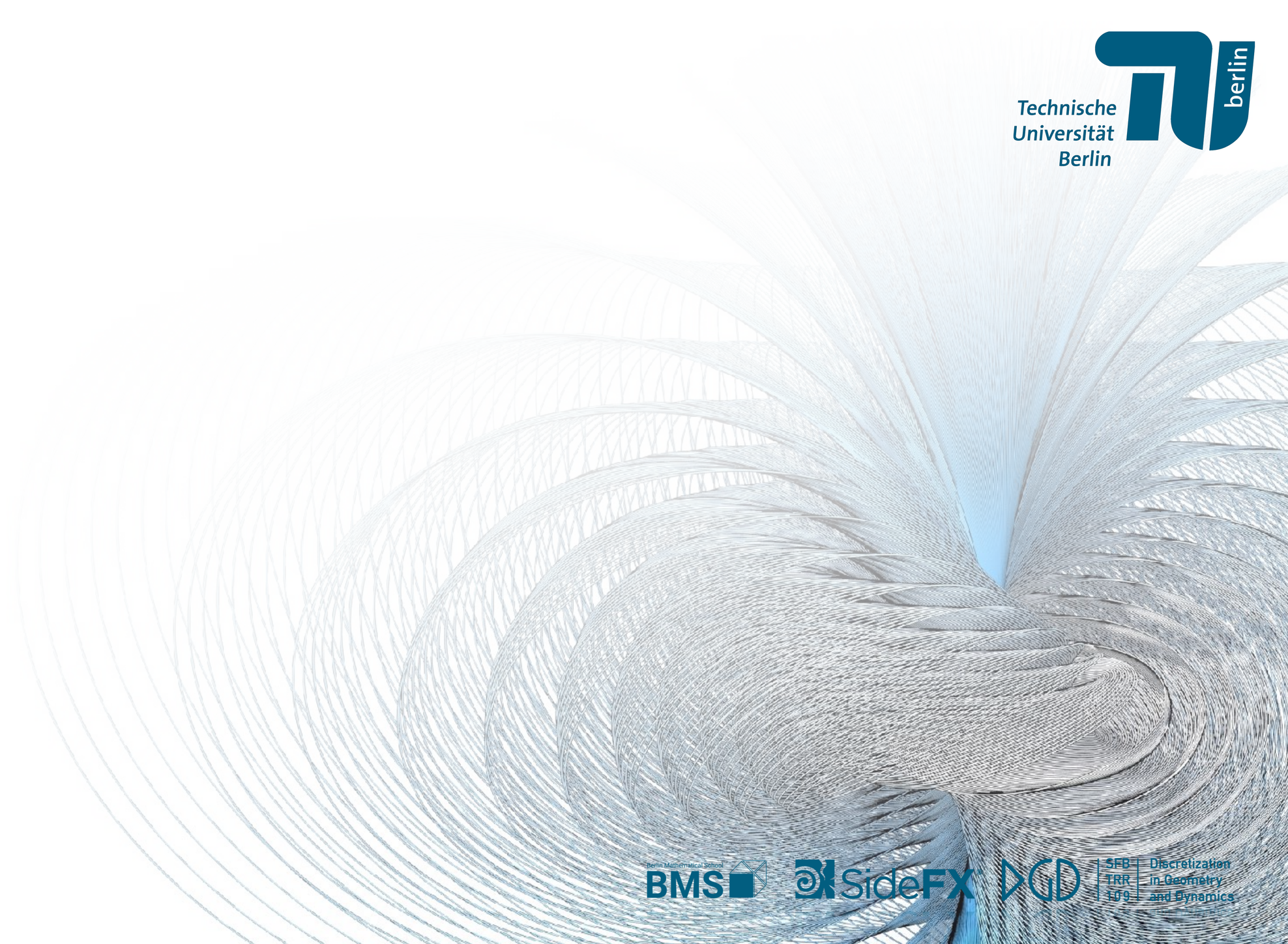
Animationen Rendern





Technische Universität Berlin

Pause



Technische Universität Berlin

Berlin Mathematical School BMS

SideFX

DGD

SFB TRR 109 Discretization in Geometry and Dynamics

Volumen

Technische Universität Berlin

Berlin Mathematical School BMS

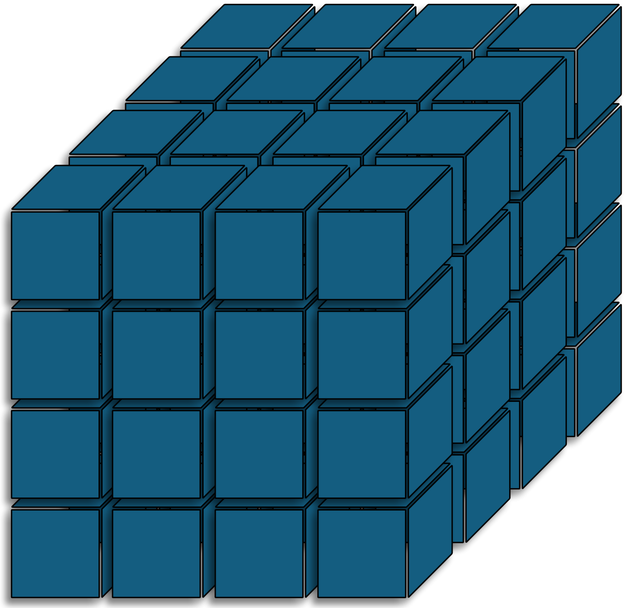
SideFX

DGD

SFB TRR 109 Discretization in Geometry and Dynamics

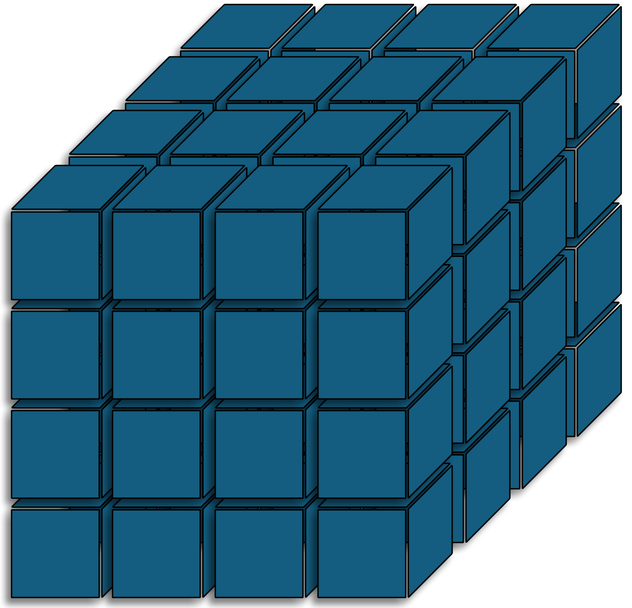
Volumen

- Für volumetrische Visualisierungen können wir “Houdini Volumes” benutzen.



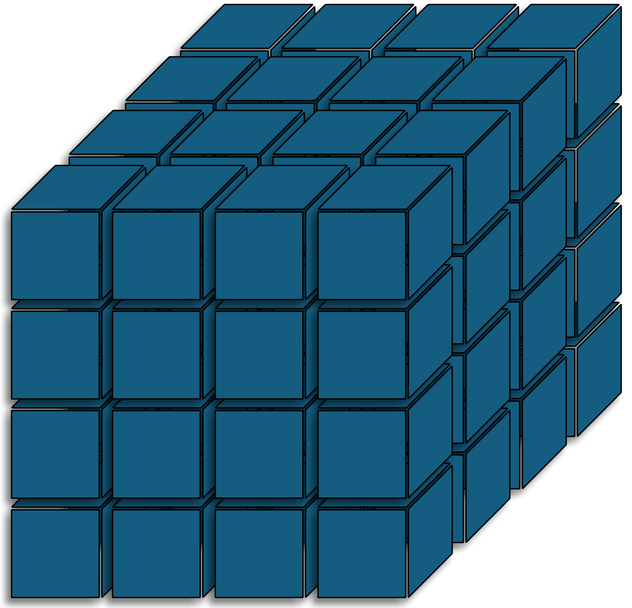
Volumen

- Für volumetrische Visualisierungen können wir “Houdini Volumes” benutzen.
    - Sie repräsentieren ein 3D-grid bestehend aus “Voxeln”.



Volumen

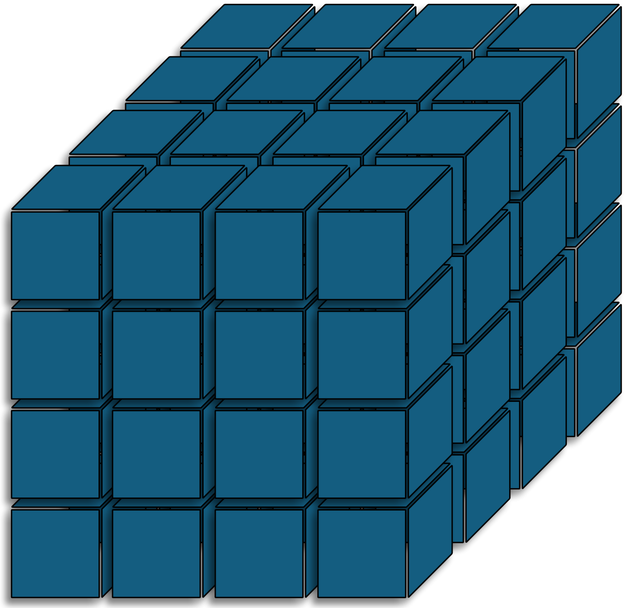
- Für volumetrische Visualisierungen können wir “Houdini Volumes” benutzen.
    - Sie repräsentieren ein 3D-grid bestehend aus “Voxeln”.
    - Es wird unterschieden zwischen Skalar- und Vektor-wertigen Volumen



Volumen

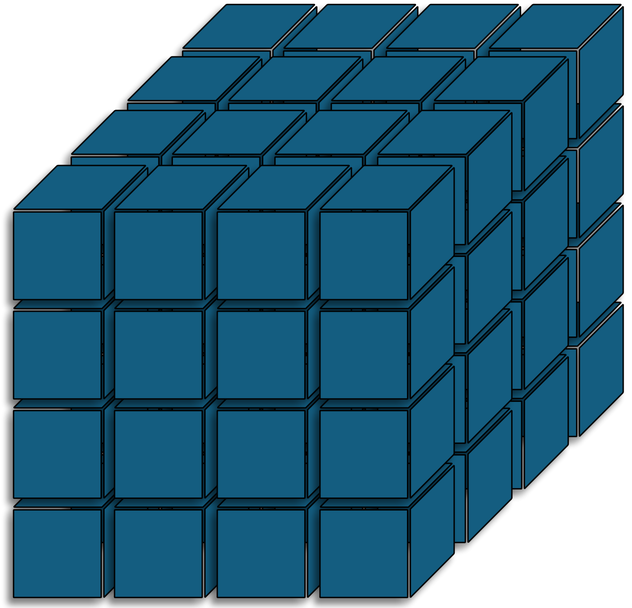
- Für volumetrische Visualisierungen können wir “Houdini Volumes” benutzen.

- Wir iterieren mit einem “volumewrangle” (for each voxel do:…)



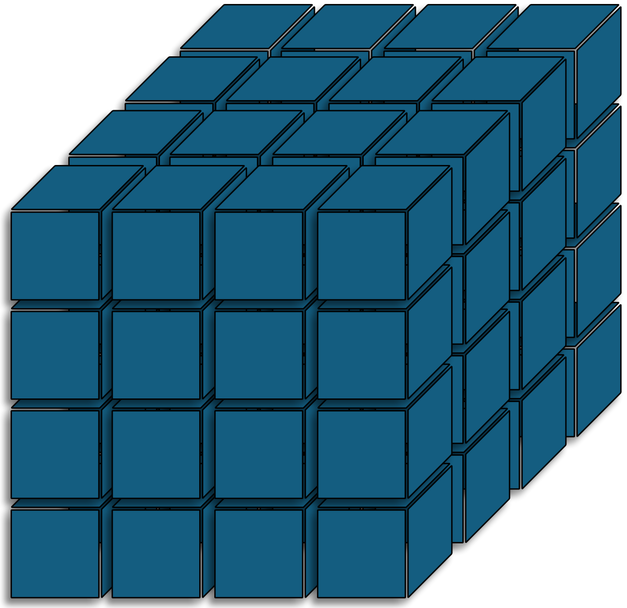
Volumen

- Für volumetrische Visualisierungen können wir “Houdini Volumes” benutzen.
- Wir iterieren mit einem “volumewrangle” (for each voxel do:…)
- Visualisierung von Vektorfeldern durch “volume trail” oder “volume slice” Knoten

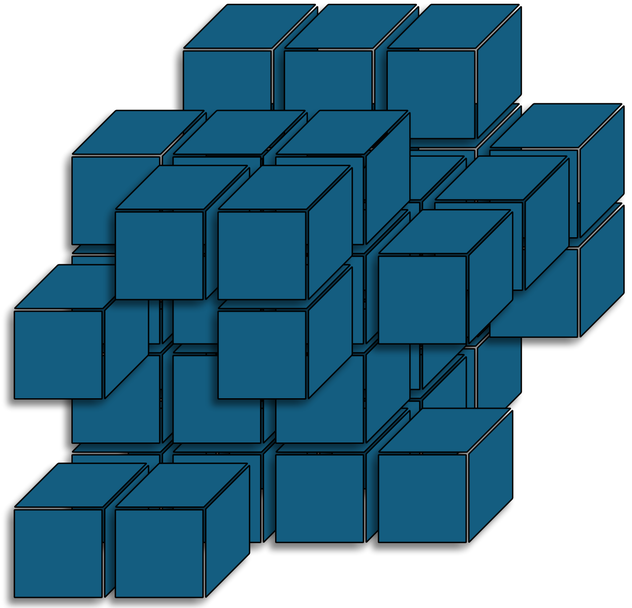


Volumen

- Für volumetrische Visualisierungen können wir “Houdini Volumes” benutzen.



- Effizientere Alternative: VDB-Volumes (etwas komplexer)



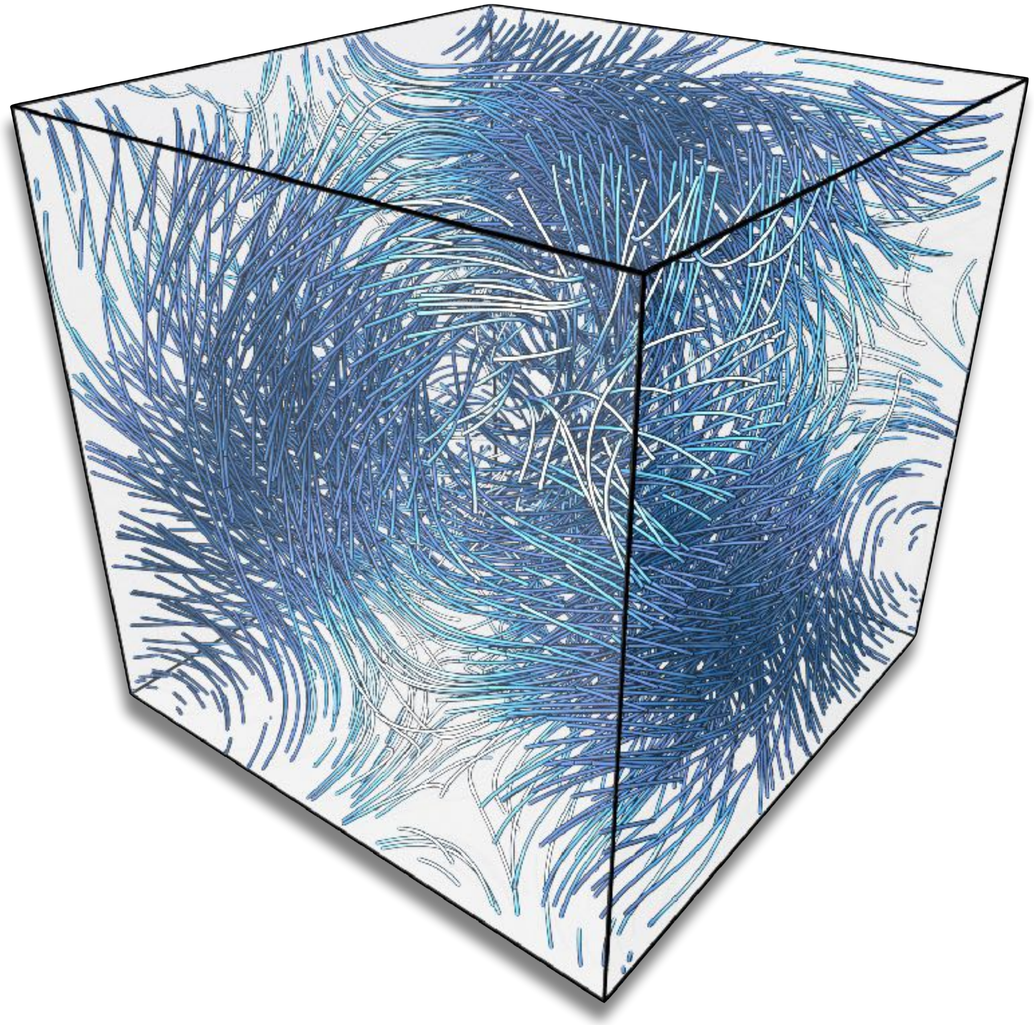
Animation

Übung:

a) Visualisiere den “ABC-Flow” gegeben durch

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} A\sin(z) + C\cos(y) \\ B\sin(x) + A\cos(z) \\ C\sin(y) + B\cos(x) \end{bmatrix}$$

für freie Parameter $A, B, C \in [0,1]$.



Technische Universität Berlin

Wie geht es weiter?

Berlin Mathematical School BMS

SideFX

DGD

SFB TRR 109 Discretization in Geometry and Dynamics

Wie geht es weiter?

In der begrenzten Zeit kratzen wir natürlich nur nur an der Oberfläche.

Nicht angesprochen haben wir z.B.:

- “Sinnvolle Projekte”

Wie geht es weiter?

In der begrenzten Zeit kratzen wir natürlich nur nur an der Oberfläche.

Nicht angesprochen haben wir z.B.:

- “Sinnvolle Projekte”
- Python-Knoten → SciPy

Wie geht es weiter?

In der begrenzten Zeit kratzen wir natürlich nur nur an der Oberfläche.

Nicht angesprochen haben wir z.B.:

- “Sinnvolle Projekte”
- Python-Knoten → SciPy

Weiteres Übungsmaterial findet ihr z.B. hier:

- **Mathematische Visualisierung TU Berlin** (einfach googeln)

Wie geht es weiter?

In der begrenzten Zeit kratzen wir natürlich nur nur an der Oberfläche.

Nicht angesprochen haben wir z.B.:

- “Sinnvolle Projekte”
- Python-Knoten → SciPy

Weiteres Übungsmaterial findet ihr z.B. hier:

- **Mathematische Visualisierung TU Berlin** (einfach googeln)
- **Houdini Tech-Blog** (http://wordpress.discretization.de/houdini/)

Aber was ist möglich?

Wir haben gesehen, dass Houdini für professionelle Zwecke z.B. in der Computergrafik genutzt wird.

- Genutzt für Code bei z.B. ACM SIGGRAPH
  - Geometry Processing
  - Physics / Fluid Simulation
  - Rendering

Filament Based Plasma

Filament Based Plasma

Motion from Shape Change

Oliver Gross
TU Berlin

Yousuf Soliman
Caltech

Marcel Padilla
TU Berlin

Felix Knöppel
TU Berlin

Ulrich Pinkall
TU Berlin

Peter Schröder
Caltech

Motion from Shape Change

Oliver Gross
TU Berlin

Yousuf Soliman
Caltech

Marcel Padilla
TU Berlin

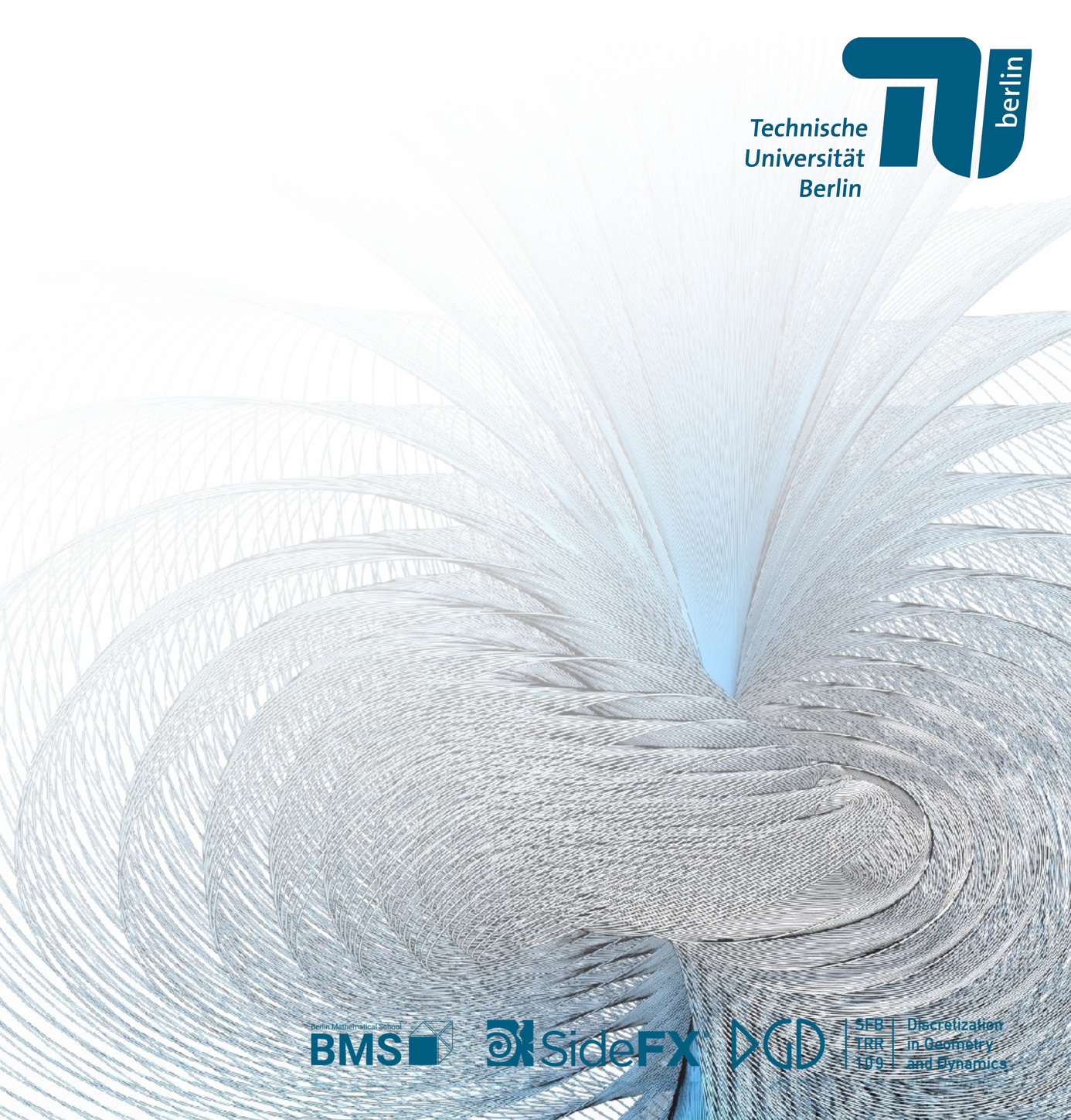
Felix Knöppel
TU Berlin

Ulrich Pinkall
TU Berlin

Peter Schröder
Caltech

Technische Universität Berlin

Vielen Dank!



ogross@math.tu-berlin.de

olligross.github.io

Berlin Mathematical School BMS

SideFX

DGD

SFB TRR 109 Discretization in Geometry and Dynamics