

KRONOS

# BENUTZERHANDBUCH



Handbuch zum  
Studienprojekt 2015/16  
“Satellitengestützte  
Terrainvisualisierung zur  
Umweltbeobachtung und  
Präsentation relevanter  
Observationen”



# Inhaltsverzeichnis - Benutzerhandbuch Kronos

<b>Start der Software "Kronos"</b>	<b>6</b>
Start unter Windows	6
Start unter Linux	6
Aufbau von Kronos	6
Laden von Plugins	6
<b>Erstellen der Datanbank</b>	<b>8</b>
<b>Visualisierung der Wettbewerbsdaten des SciVis-Contest 2014</b>	<b>9</b>
Darstellung der MIPAS Datensätze	9
Darstellung der AIRS Datensätze	10
Darstellung der CLaMS Datensätze	13
<b>Visualisierung von Flugdaten</b>	<b>15</b>
Filtern nach Airlines	16
Filtern nach Flughäfen	16
Filtern nach Fluglänge	16
Visualisierung der Flugrouten durch Geodäten	17
Detailgrad (Level of detail)	17
Bogenhöhe (Arc Size)	17
Berechnungstiefe (Limit calculation depth)	17
Mögliche Fehlerquellen	19
Technische Details	19
<b>Visualisierung von Klima- und Wetterdaten zur Analyse der Flugsicherheit und Aerosoltransport</b>	<b>20</b>
Darstellung von Temperatur	20
Visualisierung der Temperaturdaten als Heatmap	20
Filterung nach Minimal- und Maximaltemperatur	20
Zeitliche Aggregation von Temperaturdaten	21
Zeitliche Interpolation von Temperaturdaten	21
Darstellung von Niederschlag	21
Visualisierung der Niederschlagsdaten als Heatmap	21
Filterung nach Niederschlagstypen	22
Zeitliche Aggregation von Niederschlagsdaten	22
Zeitliche Interpolation von Niederschlagsdaten	22
Darstellung von Wind	23



Visualisierung der Windgeschwindigkeiten als Heatmap	23
Filterung nach Windgeschwindigkeit	23
Zeitliche Aggregation von Winddaten	24
Zeitliche Interpolation von Winddaten	24
Flow-Visualisierung von Winddaten	24
<b>Erstellung einer Terrainvisualisierung der Erde</b>	<b>26</b>
Globus und Flachkarte	26
Manuelle Einstellung der Überhöhung	27
<b>Ortssuche</b>	<b>29</b>
Mögliche Fehlerquellen	29
<b>Ortsdarstellung</b>	<b>30</b>
<b>Einbinden geolokalizierter Twitter Daten</b>	<b>31</b>
Filtern nach Autor	31
Filtern nach Schlüsselwörtern und Hashtags	31
Filtern nach Anzahl der Retweets (Kriterium für Relevanz eines Tweets)	32
Visualisierung als Heatmap	32
Anzeigen der Twitternachrichten / Detailansicht	32
Anwendungsbeispiel: Heatmap zu einem bestimmten Hashtag generieren	34
<b>Zeitliche Interpolation</b>	<b>36</b>
<b>Räumliche Interpolation</b>	<b>37</b>
<b>Zeitliche Aggregation</b>	<b>38</b>
<b>Transformationsfilter</b>	<b>39</b>
Häufige Fehlerquellen	39
<b>Datenübergreifende Filteroptionen</b>	<b>40</b>
Data Point Density Reduction Filter	40
Heatmap Density Filter	40
Terrain Height Filter	41
<b>Anbindung an die Powerwall</b>	<b>42</b>
<b>Einrichten von dynamischen Zoomstufen</b>	<b>47</b>
<b>Datenformate und Datenkonvertierungen</b>	<b>49</b>
Das Kronos JSON-Format	49
Nicht temporale Daten	49
Temporale Daten	50
Flugdaten	51
kJSON Format für Flüge (nicht temporal)	52
Städtedaten	52



kJSON Format für Städte (nicht temporal)	52
Wetterdaten	53
kJSON Format für Niederschlag (temporal)	53
kJSON Format für Temperatur (temporal)	53
kJSON Format für Wind (temporal)	54
kJSON Format für Bewölkung (temporal)	54
Twitterdaten	55
kJSON Format für Tweets (temporal)	55
Küstenlinien und Ländergrenzen	55
Erweiterung durch neue Datenquellen	56
<b>Visualisierung von mehreren Datensätzen</b>	<b>57</b>



# Start der Software "Kronos"

## Start unter Windows

Nach der Installation mithilfe des Installers (siehe Installationsanleitung) kann Kronos ganz einfach über einen Doppelklick auf die kronos.exe, die sich im Installationspfad befindet, gestartet werden.

Bei einem eigenen Build befindet sich standardmäßig unter Buildpfad/src/<Buildmodus> die entsprechende exe, die ebenfalls mit einem Doppelklick ausgeführt werden kann.

## Start unter Linux

./Kronos

## Aufbau von Kronos

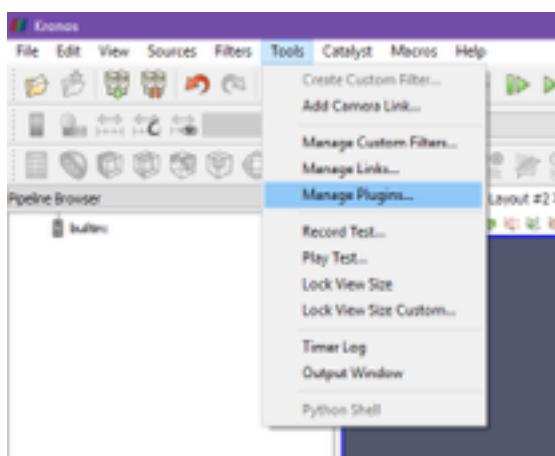
Kronos basiert auf der Open-Source Software "Paraview" von KitWare. Wenn Sie sich bereits in ParaView eingearbeitet haben, werden Sie sofort intuitiv die neuen Funktionen von Kronos erkennen und bedienen können, da die Modulare Struktur von Kronos konsistent auf Paraview aufbaut und diese logisch erweitert.

Kronos kann im Großen und Ganzen in vier Bereiche eingeteilt werden:

1. Die Menüleiste enthält wichtige Werkzeuge zur Ausrichtung der Daten.
2. Der Pipeline-Browser zeigt alle geladenen Daten und Filter an und ermöglicht es diese umzuordnen und neue Filter anzuwenden.
3. Im Visualisierung-Fenster findet die eigentliche Visualisierung statt. Hier ist der Globus zu sehen, um den sich alles dreht, sowie alle weiteren Daten.
4. Weitere modulare Ansichten, die je nach Bedarf hinzugeschaltet werden können (beispielsweise Animation View oder Color Map Editor). Diese Ansichten können unter "View" ein- und ausgeblendet werden, indem das entsprechende Häkchen gesetzt wird.

## Laden von Plugins

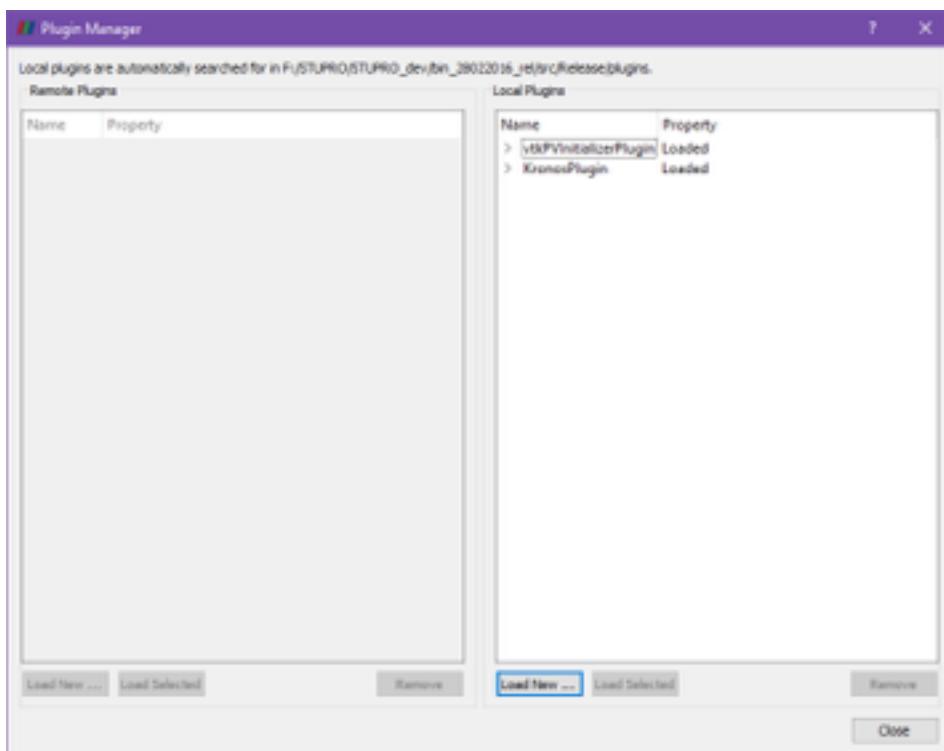
Um andere Plugins als das standardmäßige "KronosPlugin" in Kronos zu laden, kann man in der Menüleiste auf den Punkt "Tools > Manage Plugins..." gehen, um das Fenster zur Pluginverwaltung zu öffnen.



Öffnen des "Manage Plugins"-Fensters



Hat sich dann folgendes Fenster geöffnet, kann man über den Button "Load New ..." den Computer nach entsprechenden "\*.dll", "\*.dylib" oder anderen Bibliotheksformaten durchsuchen und diese in Kronos laden.



Die Übersicht des Pluginmanagers

---

Plugins, die bei ParaView standardmäßig enthalten sind, sind bei Kronos alle im Installationspfad im Ordner "paraview\_plugins" zu finden.



# Erstellen der Datenbank

Bitte beachten Sie hierbei: PostgreSQL muss installiert sein (siehe Installationsanleitung). Unter Linux und Mac OS das Skript "init-table.sh" ausführen (in "STUPRO/scripts/sql"). Ist der postgres-User nicht der Superuser (befugt Datenbanken zu erstellen), dann das Skript mit "init-table.sh --SUPERUSERNAME" ausführen.

Unter Windows muss zuerst sichergestellt sein, dass alle ausführbaren Dateien von PostgreSQL in der globalen "PATH"-Variable zu finden sind. Sobald dies der Fall ist, kann die Datei "windows-init-db.bat" via Doppelklick oder über die Kommandozeile ausgeführt werden.

Daraufhin erfolgen drei Passwortabfragen; die erste und zweite Abfrage verlangen das Passwort des postgres-Benutzers, welches beim Installieren von PostgreSQL angelegt wurde, die Dritte unser Passwort, "weloveparaview".

Hinweis: Steht im "Find-Places" Widget in dem Suchfeld der Hinweis "Failed to open database connection" liegt das häufig daran, dass der database-server lokal nicht gestartet wurde.



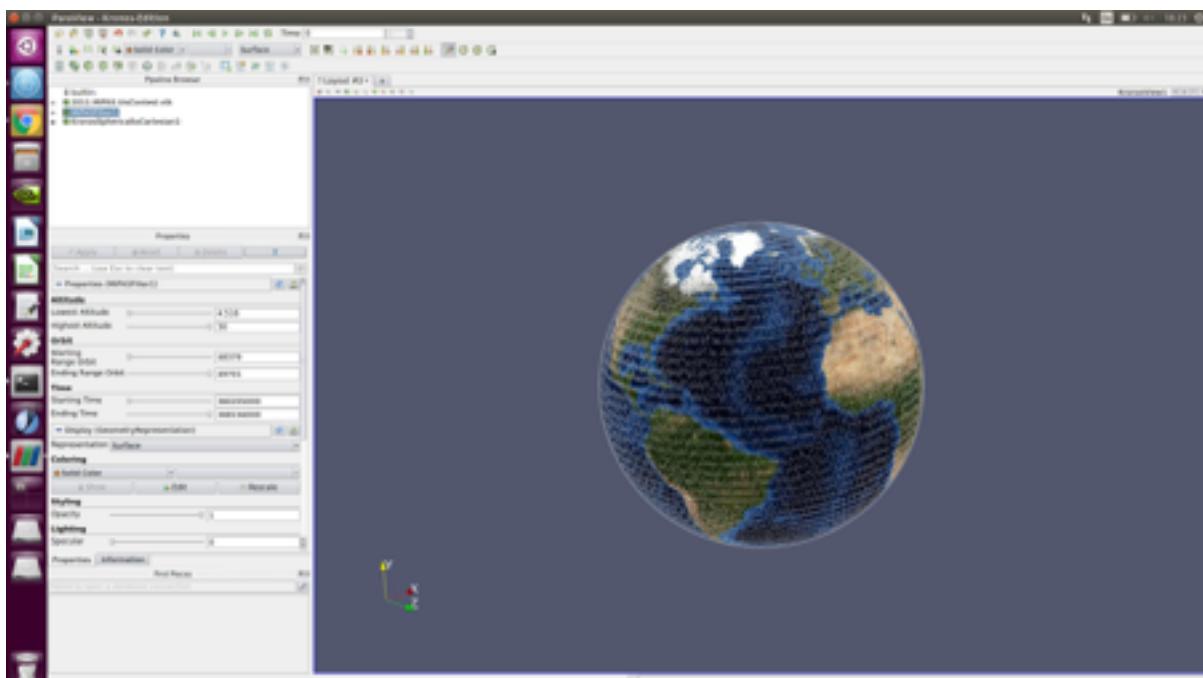
# Visualisierung der Wettbewerbsdaten des SciVis-Contest 2014

Zur erfolgreichen Darstellung der Wettbewerbsdaten müssen folgende Filter betrachtet werden: Der MIPASFilter zur Filterung der MIPAS-Daten, der AIRSFilter zur Darstellung der AIRS-Daten, sowie der CLaMSFilter.

Für die Interpolation der SciVis-Daten kann der interne Shader von Paraview genutzt werden, da dieser ein Mesh automatisch interpoliert. Mit dem Delaunay2D Filter kann ganz einfach aus den vorhandenen Daten ein Mesh erstellt werden. Möchte man Durchschnittswerte erhalten, so nimmt man den temporal statistics Filter um die Daten noch genauer anpassen zu können. Dabei sollten die oben genannten Filter nur auf die im Filternamen angegebenen Datensätze angewendet werden. Es wird nicht garantiert, dass ein MIPASFilter auch für AIRS-Datensätze funktioniert oder der AIRSFilter mit CLaMS-Datensätzen.

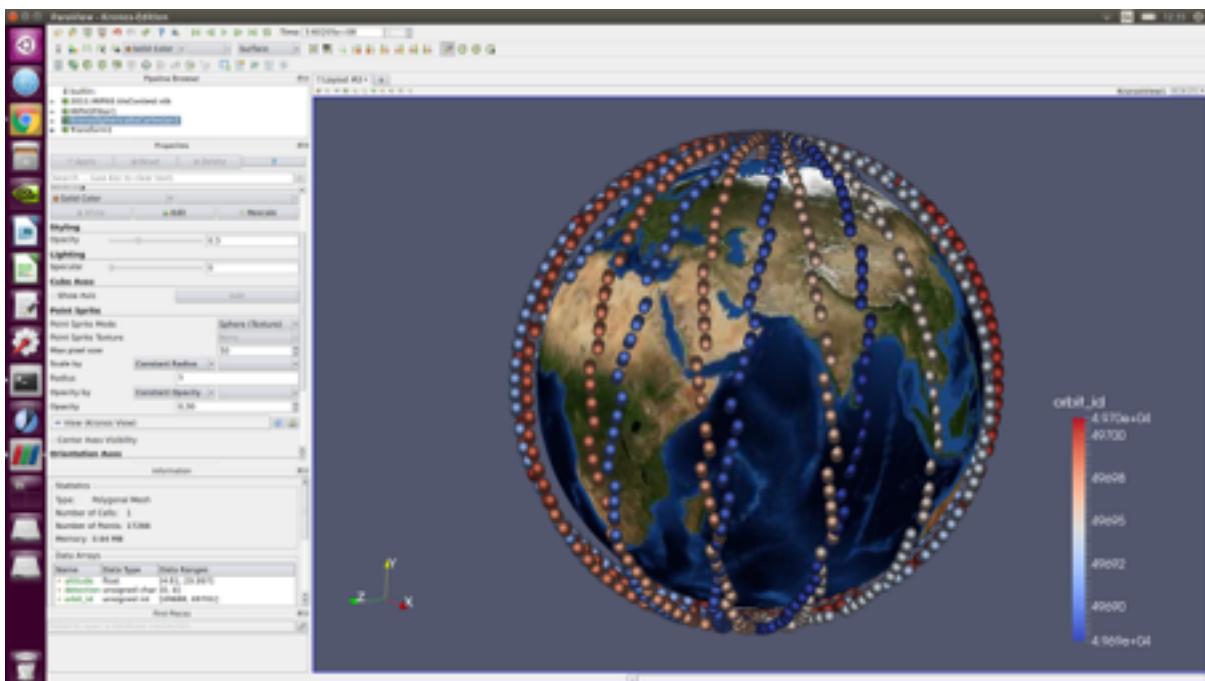
## Darstellung der MIPAS Datensätze

Zum Darstellen der MIPAS Datensätze wird der MIPASFilter verwendet. Bevor man beim MIPASFilter auf "Apply" drückt, sollte man rechts neben den Minimal- und Maximalwerten auf den rotierenden Pfeil drücken. So werden alle für diesen Datensatz relevanten Minimal- und Maximalwerte automatisch eingestellt. Der Filter ermöglicht es über die Zeit, die Orbit\_id, die Detektionen, Profile sowie über die Höhe zu filtern. Dabei kann der Nutzer mit Slidern den Maximal- und Minimalwert einstellen. Zur besseren Darstellung der MIPAS-Daten werden Point Sprites verwendet. Dazu muss der Nutzer beachten, dass folgendes Plugin vorhanden ist: PointSprite\_Plugin. Dieses ist im Plugin Manager zu finden. Eine Anleitung, wie Plugins geladen werden, wird im Abschnitt "Laden von Plugins" erklärt. Die folgenden Bilder zeigen die MIPAS-Daten auf dem Globus und der 2,5D Karte als auch mit Point Sprites auf dem Globus.

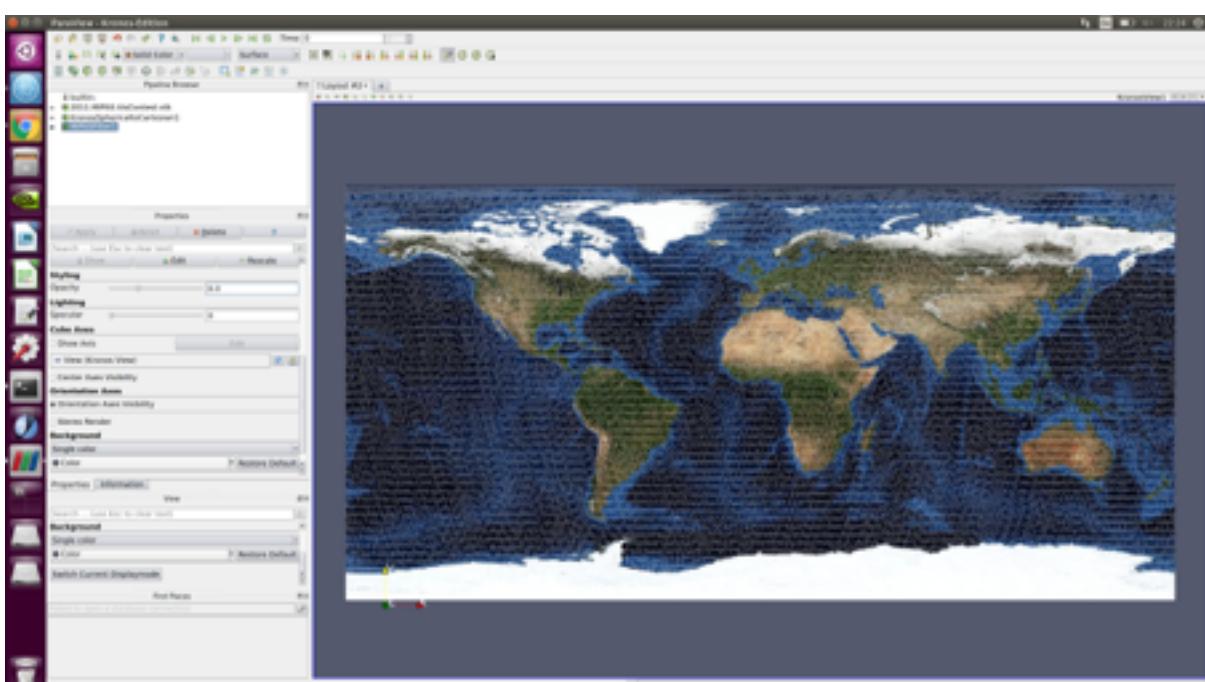


Darstellung der MIPAS Daten auf dem Globus





Darstellung der MIPAS Daten mit Point Sprites auf dem Globus



Darstellung der MIPAS Daten auf der 2,5D Karte

## Darstellung der AIRS Datensätze

Zum Darstellen der AIRS-Datensätze wird der Filter [AIRSFilter](#) verwendet. Diese beiden Filter erlauben es über die Zeit, den Asche Index und die SO<sub>2</sub>-Aerosole zu filtern. Da die Datensätze von ihrer orbitalen Umlaufbahn Lücken aufweisen, ist es nötig den [Delaunay 2D Filter](#) zu nutzen, um ein Mesh zu erstellen. Der Shader in Paraview interpoliert dieses dann. Es wird empfohlen den [Delaunay 2D Filter](#) zuerst im 2D Modus an zu wenden und erst anschließend

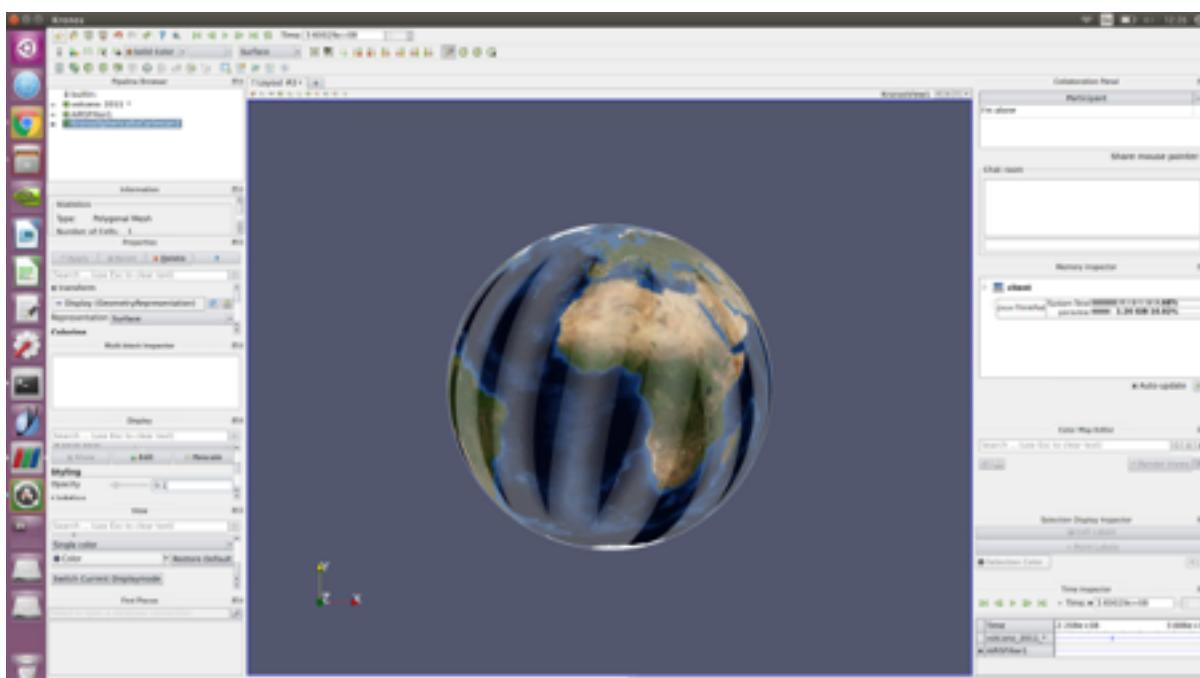


diesen auf den Globus zu mappen. Bevor man beim **AIRSFilter** auf "Apply" drückt, sollte man rechts neben den Minimal- und Maximumwerten auf den rotierenden Pfeil drücken. So werden alle für diesen Datensatz relevanten Minimal- und Maximalwerte automatisch eingestellt. Es muss dabei beachtet werden dass das Minimum der SO<sub>2</sub> Datensätze nicht kleiner als 0.5 und das Asche Minimum größer als 0.0 eingestellt werden soll. Negative Datenpunkte enthalten keine relevanten Messungen und können ignoriert werden.

Über verschiedene Slider hat der Nutzer die Möglichkeit die einzelnen Attribute der Datensätze zu verändern. Somit kann beispielsweise ein bestimmter Tag ausgewählt werden und an diesem Tag nach besonders relevanten Attributen gefiltert werden. An dem Tag eines Vulkanausbruchs ist die Aschekonzentration besonders interessant, während dieser Wert mit zunehmenden Abstand zu dem Ereignis immer irrelevant wird.

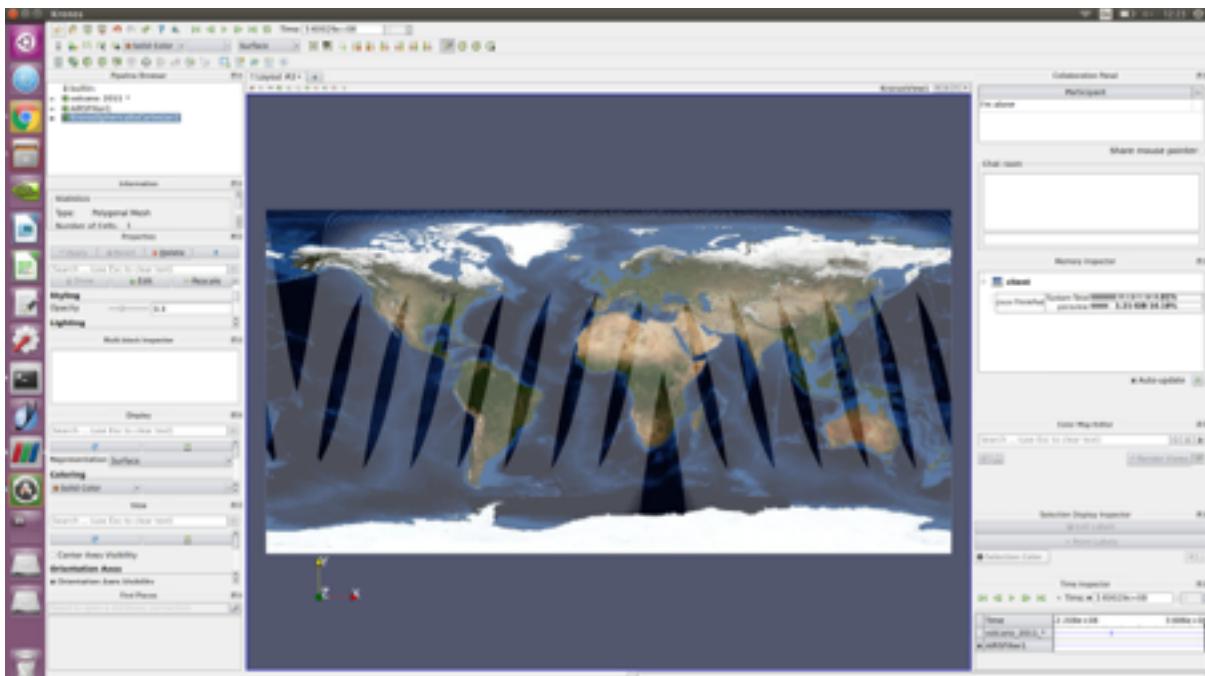
Da in den Daten die Zeit in Julian Sekunden gespeichert ist, muss beachtet werden, dass das Startdatum, der AIRS\_am-Daten ungefähr der 30.05.2011 um 00:00 UTC ist. Die AIRS\_pm-Daten beginnen am 30.05.2011 um 12:00 UTC . Da eine Julian Sekunde einer Sekunde entspricht, enthält ein Tag genau 86400 Sekunden. Um einen gewünschten Tag n auswählen zu können, muss man daher n mal 86400 zu dem Startwert dazu addieren.

Folgende Bilder zeigen die AIRS-Daten auf dem Globus und der Karte, als auch die Interpolation auf dem Globus.

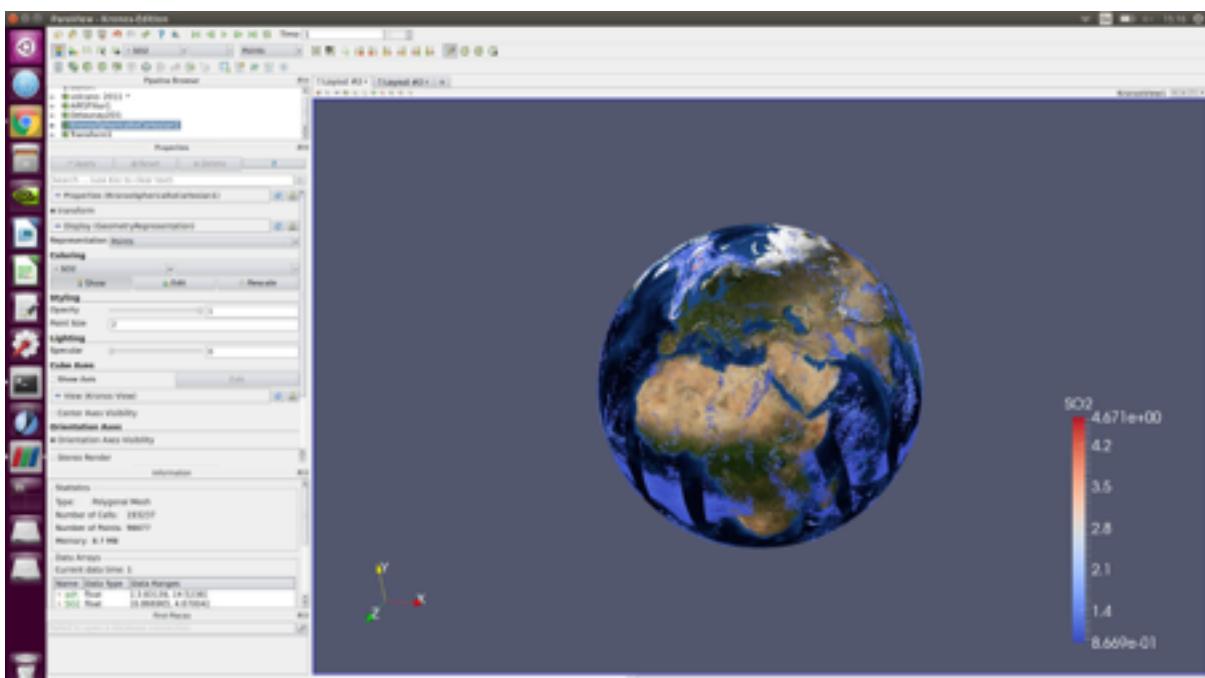


Darstellung der AIRS Daten auf dem Globus





Darstellung der AIRS Daten auf dem Globus



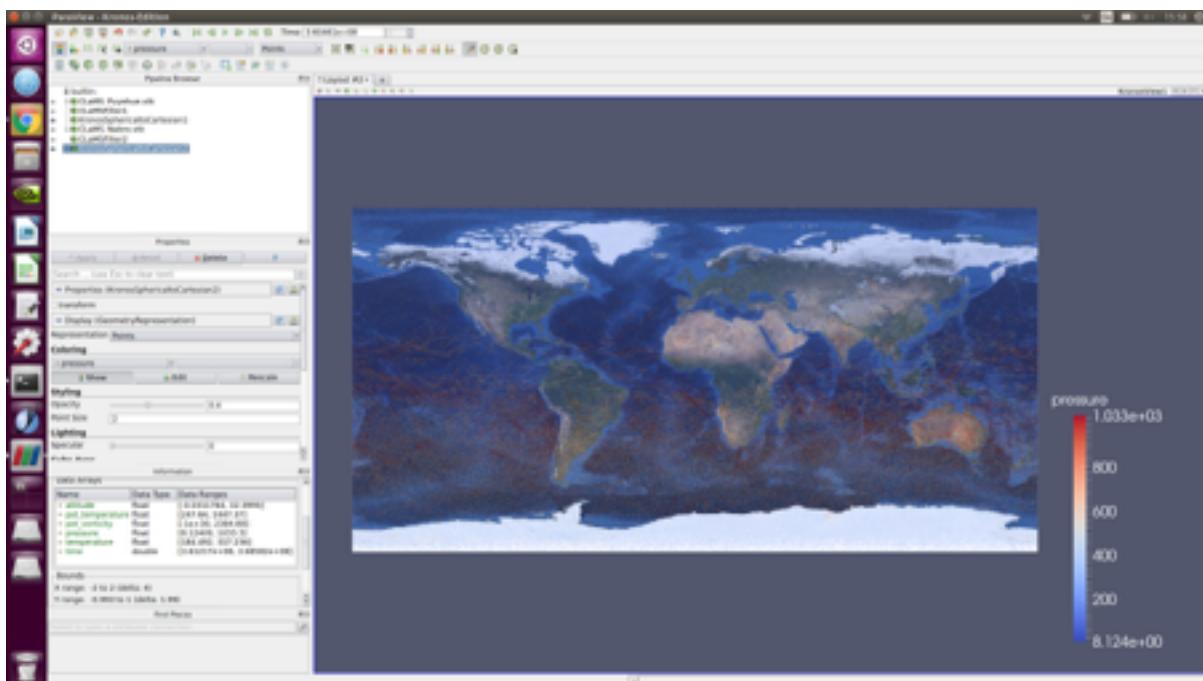
Darstellung der interpolierten AIRS Daten



## Darstellung der CLaMS Datensätze

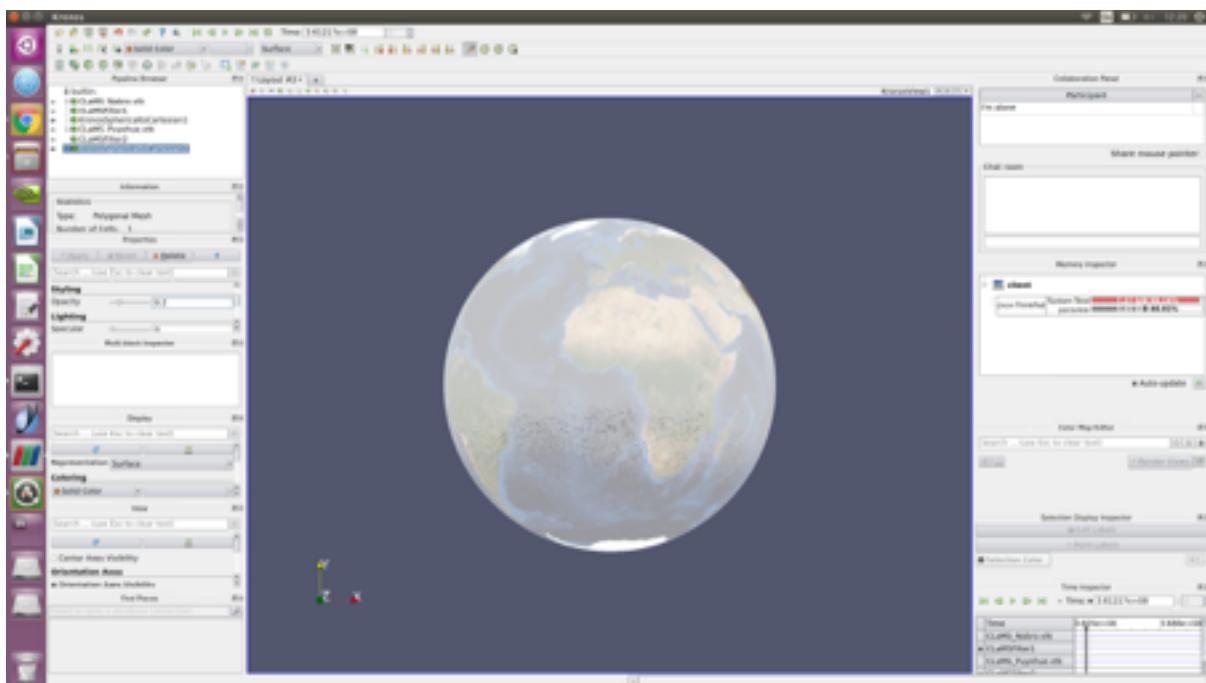
Zum Darstellen der CLaMS-Datensätze wird der Filter **CLaMSFilter** verwendet. Mit diesem Filter kann man die Attribute "Zeit", "Druck", "Temperatur", "Wirbelstärke", "potentielle Temperatur" und die "Höhe" darstellen. Bevor man beim **CLaMSFilter** auf "Apply" drückt sollte man rechts neben den Minimal- und Maximalwerten auf den rotierenden Pfeil drücken. So werden alle für diesen Datensatz relevanten Minimal- und Maximalwerte automatisch eingestellt. Will man die Daten anpassen, so sind für jedes Attribut Slider und eine Texteingabe verfügbar. Es ist über die Filter möglich sich ein Attribut genauer anzusehen und die anderen dabei auszublenden. Vorsicht ist geboten, da viele Attribute voneinander abhängig sind. Gewisse Zeitabschnitte besitzen einfach keine Aufnahmen. Es sollte klar sein, dass es zu diesen Zeiten auch nicht möglich ist, alle Datensätze anzeigen zu können.

Im Folgenden werden die Filter auf dem Globus, als auch auf der Karte gezeigt.



Darstellung der CLaMS Daten



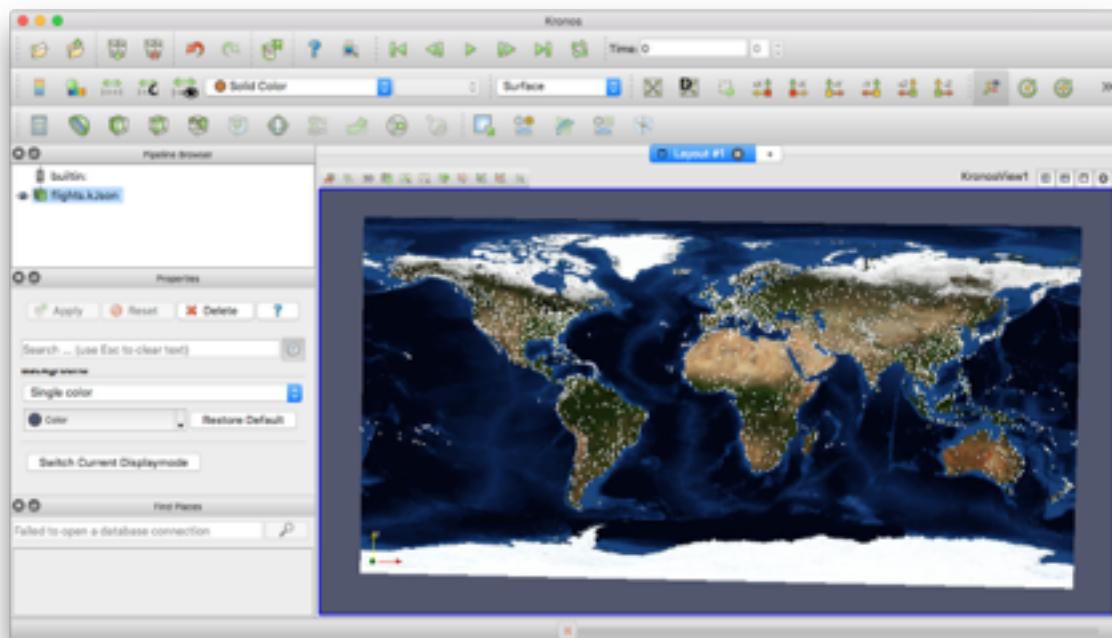


Darstellung der CLaMS Daten von Puyehue und Nabro auf dem Globus



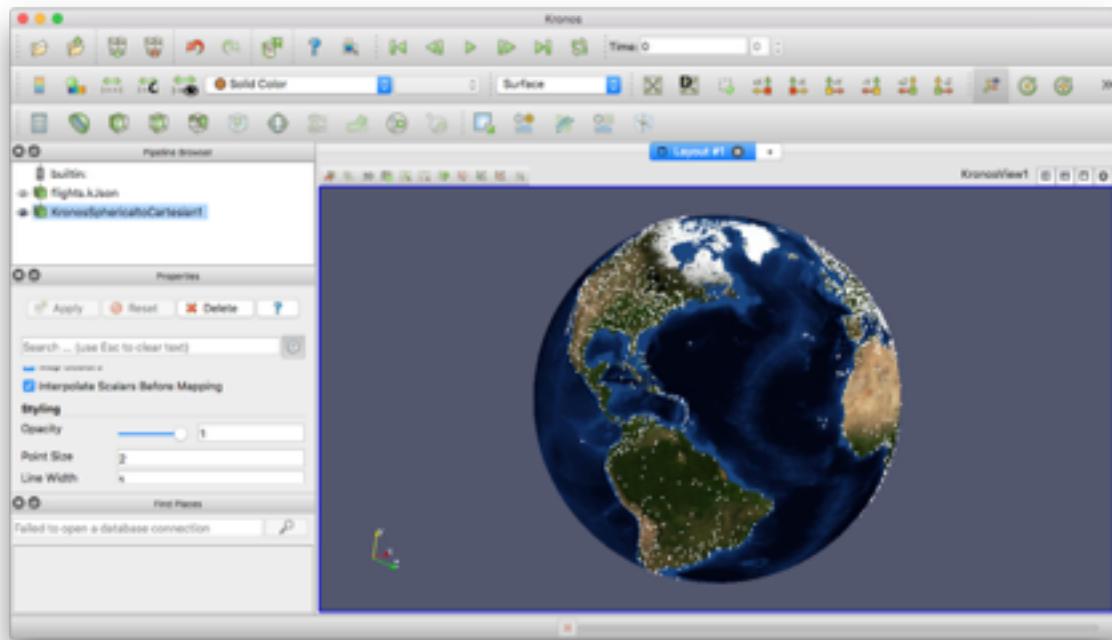
# Visualisierung von Flugdaten

Zunächst müssen Flugdaten geladen werden. Ein Beispieldatensatz ist unter dem Namen "flights.kJson" in "STUPRO/scripts/data-conversion" zu finden. In ParaView kann man unter "File/Open" diese Datei laden. Anschließend auf "Apply" drücken und die Daten werden als Punktdaten dargestellt. Hierbei werden die Startflughäfen der Flüge visualisiert.



Die Flugdaten wurden geladen auf einer 2,5D Karte. Der Startflughafen wird jeweils durch einen Punkt visualisiert.





Darstellung der Flugdaten auf dem Globus

Anschließend unter "Filter/Kronos" den Filter "FlightFilter" auswählen. Nun kann man die Flugdaten in den Properties nach verschiedenen Kriterien filtern:

## Filtern nach Airlines

In dem Feld "Airline" die Airlines, deren Flüge man anzeigen will, mit Kommata separiert eingeben. Hierbei kann man auswählen zwischen "Matching" und "Containing". Bei dem Modus "Matching" werden nur die Flüge der Airlines angezeigt, die exakt dem eingegebenen Suchtext entsprechen. Bei "Containing" werden auch Airlines angezeigt, bei welchen nur ein Teil des Namens mit dem Suchtext überein stimmt. Nach der Eingabe der Airlines auf "Apply" drücken und die Filterung wird angewandt.

## Filtern nach Flughäfen

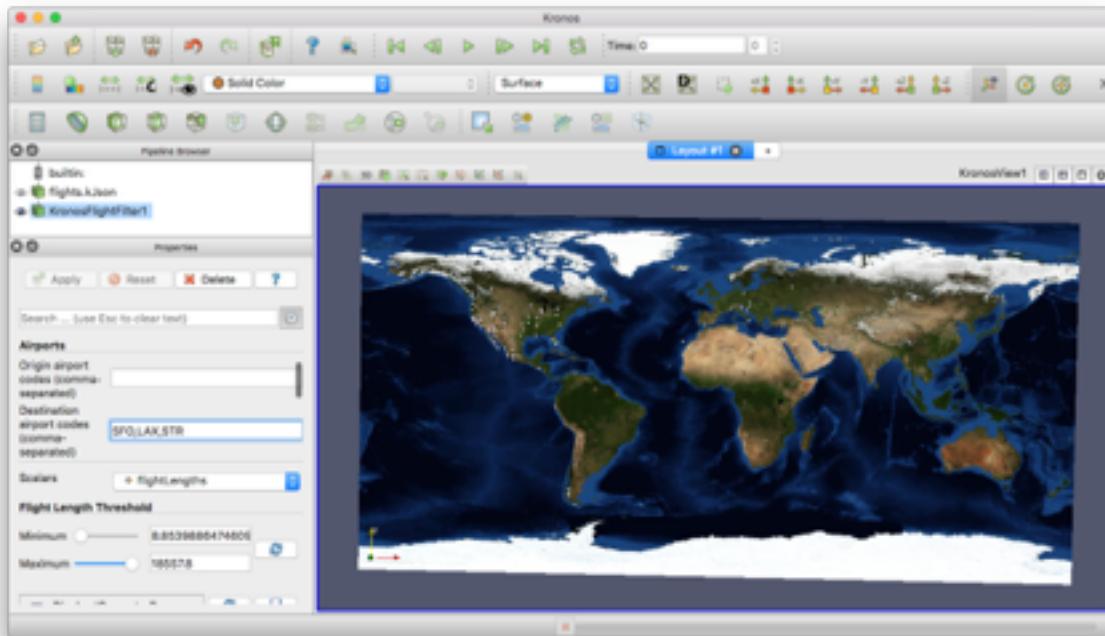
In dem Abschnitt "Airports" können sowohl die Flughafenkürzel der Startflughäfen, als auch der Zielflughäfen als Filteroption benutzt werden. Mehrere Suchanfragen müssen wieder mit Kommata getrennt werden. Da ein "Containing"-Modus aufgrund der Kürze der Flughafenkürzel wenig Sinn macht, werden die Flüge nur bei exakter Übereinstimmung mit dem Kürzel angezeigt. Nach der Eingabe der Kürzel auf "Apply" drücken und die Filterung wird angewandt.

## Filtern nach Fluglänge

Man kann in der Kategorie "Flight length" nach der Fluglänge filtern, indem man die Slider entsprechend verschiebt und danach auf Apply drückt. Angezeigt werden nun nur noch die



Flüge, deren Länge in dem ausgewählten Bereich liegt.



So sieht der Kronos Flight Filter aus. Der FlightFilter filtert hierbei nach den drei angegebenen Zielflughäfen.

## Visualisierung der Flugrouten durch Geodäten

Kronos kann auch Geodäten von Flügen anzeigen. Hierzu muss man zunächst die Flüge filtern wie im Abschnitt "Visualisierung der Flugdaten" erklärt. Von den verbleibenden Flügen können anschließend Geodäten generieren werden. Dazu unter **Filter/Kronos** den Filter "**Kronos Generate Flight Geodesics**" auswählen. Anschließend können folgende Parameter eingestellt werden:

### Detailgrad (Level of detail)

Je größer der Detailgrad, desto schöner wirken die Geodäten. Im Gegenzug werden mehr Zwischenpunkte pro Flug berechnet, sodass die Berechnung länger dauert.

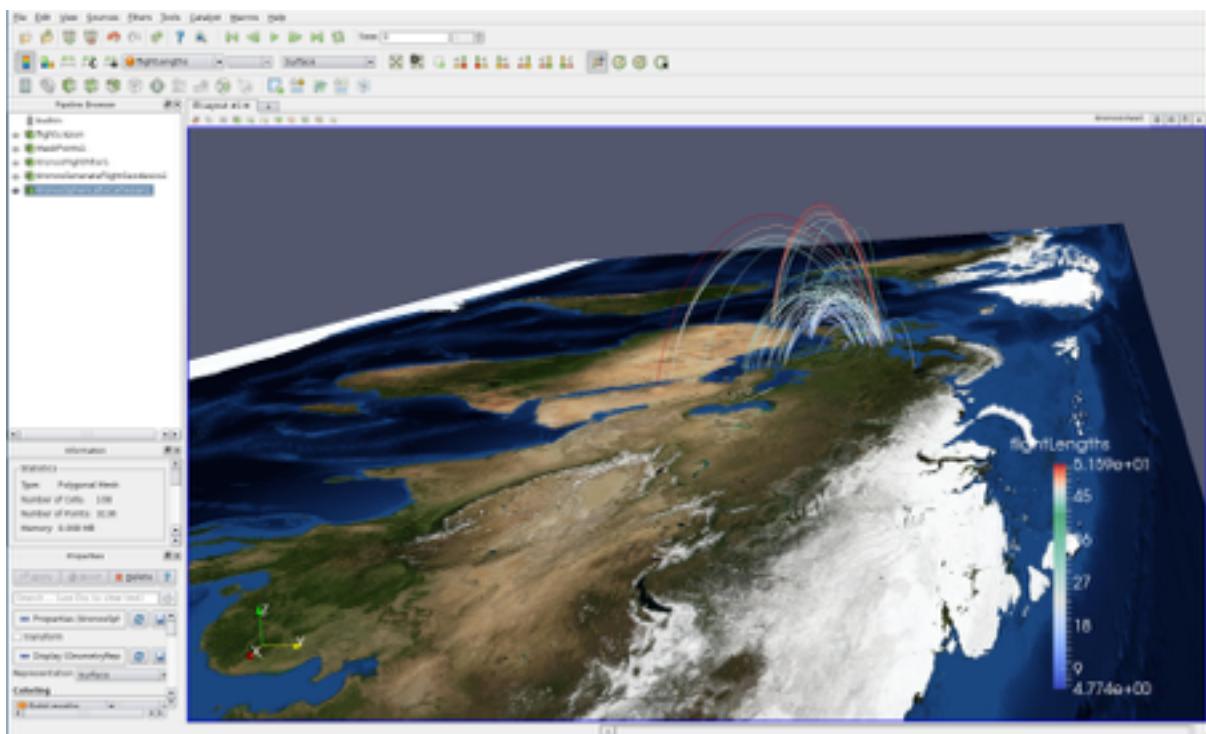
### Bogenhöhe (Arc Size)

Die Geodäten müssen nicht zwangsläufig auf der theoretischen Kugeloberfläche kleben. Wird die Bogenhöhe erhöht, so heben sich die Geodäten aus der Kugeloberfläche. Kurze Flüge erreichen hierbei eine kleinere Höhe, sodass die Übersicht gewährleistet ist.

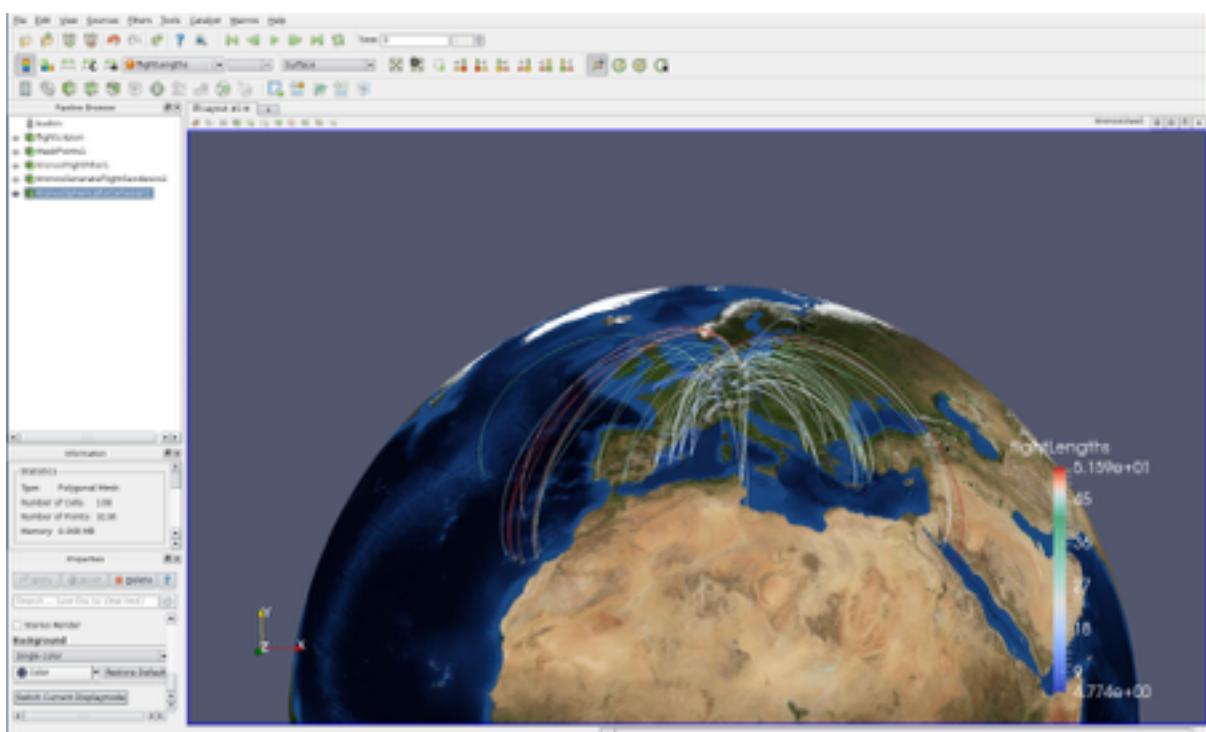
### Berechnungstiefe (Limit calculation depth)

Wenn die Berechnungstiefe limitiert wird, ist sichergestellt, dass sich das Programm bei der Ausführung nicht in einer Endlosschleife verrennt. Dies kann jedoch auf Kosten des Detailgrades gehen. Wird diese Limitierung getriggert, so blendet Kronos einen Hinweis ein. Weiteres siehe unter "Technische Details".





Einige Geodäten der Flug-Daten auf der 2,5D Karte (nicht transformiert).



Einige Geodäten der Flug-Daten



## Mögliche Fehlerquellen

Der *Kronos Generate Flight Geodesics Filter* kann keine echten Geodäten generieren, falls die Eingabedaten bereits transformiert wurden. In diesem Fall werden zur einer großen Wahrscheinlichkeit inkorrekte Ergebnisse produziert. Auch Programmabstürze können in diesem Fall nicht ausgeschlossen werden.

## Technische Details

Die Geodäten werden baumartig verfeinert. Dazu wird zunächst eine gerade Linie (im Dreidimensionalen) von Start- zu Endpunkt gezogen. Entspricht diese noch nicht dem gewünschten Detailgrad, so wird ein Zwischenpunkt in der Mitte berechnet. Der Detailgrad ist exponentiell skaliert, eine Erhöhung des Level of Details um 0,1 entspricht einer Verdopplung des Detailgrads (also eine Verdoppelung der Punkte).

Um die Bogenhöhe zu vergrößern, wird für den Kreisabschnitt der Geodäten nicht der Erdmittelpunkt als Mittelpunkt genutzt. Vielmehr wird stattdessen ein Kreismittelpunkt mithilfe des Start- und Endpunktes sowie der Bogenhöhe (exponentiell skaliert) berechnet, der oberhalb des Erdmittelpunkts liegt. Dadurch geht der entstehende Kreisabschnitt über die Erdoberfläche hinaus.

Falls die Berechnungstiefe vom Nutzer limitiert wird, bricht die Rekursion bei Tiefe 10 ab. Sind dem Nutzer die Ergebnisse zu ungenau, kann er diese Limitierung entfernen. Dabei muss allerdings der Absturz von Kronos durch eine potentielle Endlosrekursion in Kauf genommen werden.



# Visualisierung von Klima- und Wetterdaten zur Analyse der Flugsicherheit und Aerosoltransport

## Darstellung von Temperatur

### Visualisierung der Temperaturdaten als Heatmap

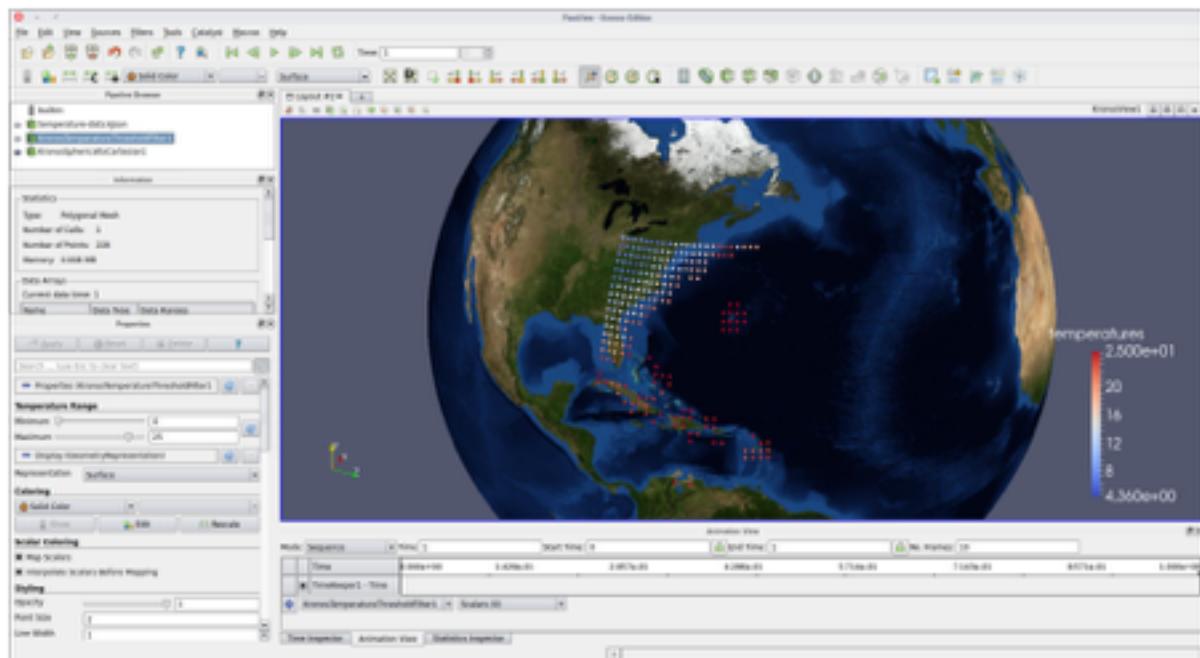
Eine wichtige Visualisierungsmöglichkeit der Temperaturdaten ist die Darstellung als Heatmap. Sind die Daten bereits in ParaView importiert, kann ein Delaunay-2D-Filter auf dem Datensatz angewandt werden, um ein Mesh zu erhalten. ParaView interpoliert dann automatisch die Skalare im Shader, um eine Heatmap zu erzeugen.

### Filterung nach Minimal- und Maximaltemperatur

Zunächst müssen Temperaturdaten mit dem Kronos-Reader geladen werden. Es genügt hierzu, in ParaView eine "kJson"-Datei zu öffnen, die Temperaturdaten enthält. Ein Beispieldatensatz ist unter dem Namen "temperatures.kJson" in "STUPRO/scripts/data-conversion" zu finden. Wählt man im Properties-Menü unter Coloring statt "Solid Color" das Skalar-Array "temperatures" aus, so werden die Temperaturen farblich visualisiert.

Sollen diese Daten nun auf ein gewisses Temperaturintervall beschränkt werden, so kann auf die Eingabedaten der "Kronos Temperature Threshold Filter" angewandt werden. Über die beiden Slider im Bedienfeld "Temperature Range" können Minimal- und Maximaltemperatur ausgewählt werden, ein Klick auf "Apply" bestätigt dies. Punkte außerhalb der festgelegten Grenzen werden entfernt.

Bitte beachten Sie, dass die möglichen Minimal- und Maximalwerte von den Minimal- und Maximaltemperaturen der Daten des aktuellen Zeitschritts abhängen. Haben Sie diesen geändert, so müssen die Slider über einen Klick auf den Knopf mit dem blauen Pfeil aktualisiert werden.



Temperaturdaten mit dem TemperatureThresholdFilter



## Zeitliche Aggregation von Temperaturdaten

Als weitere Form der Datenanalyse können die Werte von Datenpunkten über alle vorliegenden Zeitschritte aggregiert werden, indem nach dem Laden des Datensatzes der "Kronos Temporal Aggregation Filter" angewandt wird.

Speziell bei Temperaturdaten wird hierbei der Durchschnittswert aller vorhandenen Datenpunkte über die gesamten temporalen Ausmaße des Datensatzes berechnet. Die Ausgabe des Filters enthält alle individuellen Punkte aus allen Zeitschritten in den Ausgangsdaten sowie ein Feld mit dem Name "Average Temperatures", in dem sich jeweils das arithmetische Mittel der Temperaturen eines jeden Datenpunktes befindet.

Weitere Informationen befinden sich im Hauptkapitel "Zeitliche Aggregation" dieses Handbuchs.

## Zeitliche Interpolation von Temperaturdaten

Temperaturdaten können mit dem "Kronos Temporal Interpolation Filter" zeitlich interpoliert werden. Dabei werden die Temperaturwerte linear interpoliert und zeitliche Datenlücken aufgefüllt.

Weitere Informationen befinden sich im Hauptkapitel "Zeitliche Interpolation" dieses Handbuchs.

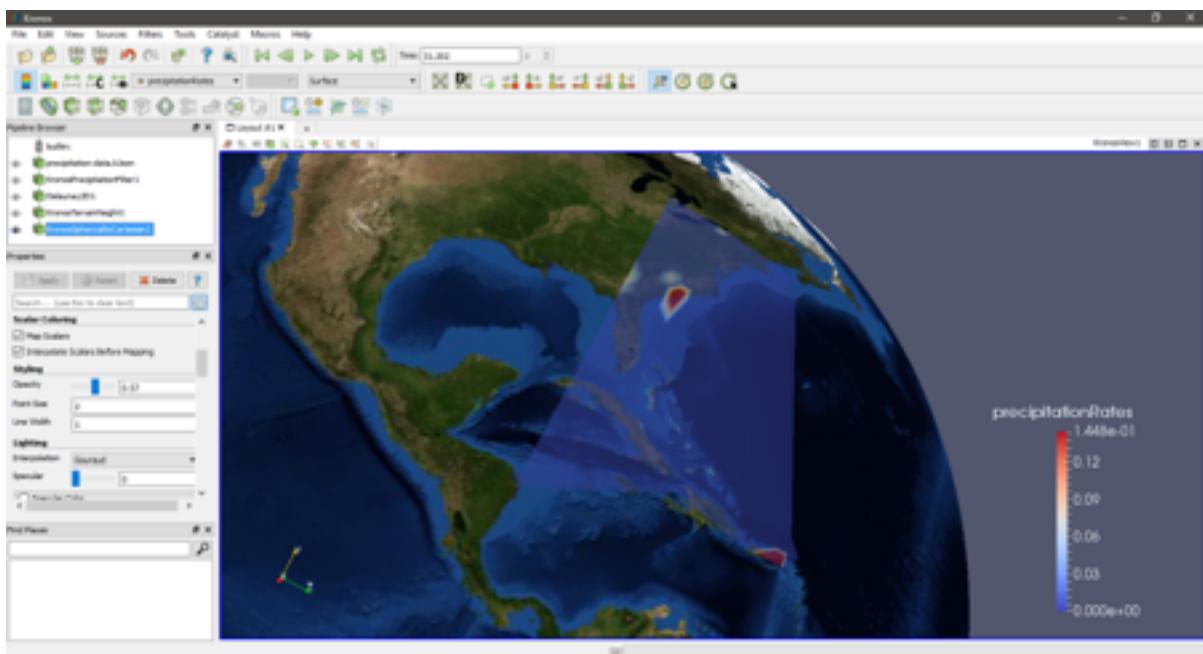
## Darstellung von Niederschlag

### Visualisierung der Niederschlagsdaten als Heatmap

Eine Möglichkeit den Niederschlag zu visualisieren ist es, die Daten als Heatmap darzustellen. Wurden die Niederschlagsdaten durch den Reader in Kronos geladen, kann man bei den Properties unter "Coloring" auswählen, ob man die Niederschlagsraten oder die Niederschlagstypen visualisiert haben will. Im Falle der Raten werden die Skalare entsprechend ihrer Ausprägung eingefärbt. Im Falle der Niederschlagstypen handelt es sich um diskrete Werte. Hier steht jede Farbe für einen bestimmten Niederschlagstyp, welche da wären: Regen, Schnee, Schneeregen und Hagel.

Um aus den Datenpunkten eine Heatmap zu generieren, muss der Delaunay-2D-Filter auf den Daten geladen und bestätigt werden. Dieser Filter erzeugt ein Dreiecks-Mesh aus den Punkten, sodass ParaView die Skalare interpolieren kann.





Heatmap der Niederschlagsraten.

## Filterung nach Niederschlagstypen

In den Daten kann auch nach den einzelnen Niederschlagstypen gefiltert werden. Hierzu muss der "Kronos Precipitation Filter" aus dem Filtermenü geladen und bestätigt werden.

Anschließend erhält man in den Properties Checkboxen für jeden Niederschlagstyp. Diese können beliebig an- und abgewählt werden. Nach einem Klick auf "Apply" werden nur noch die angewählten Daten angezeigt.

## Zeitliche Aggregation von Niederschlagsdaten

Als weitere Form der Datenanalyse können die Werte von Datenpunkten über alle Zeitschritte aggregiert werden, indem nach dem Laden des Datensatzes der "Kronos Temporal Aggregation Filter" angewandt wird.

Speziell bei Niederschlagsdaten wird hierbei die insgesamt gefallene Niederschlagsmenge ausgehend von den in den Daten gegebene Niederschlagsrate für jeden Punkt berechnet. Für diese Akkumulation wird also gewissermaßen die Funktion der Niederschlagsraten integriert, wobei die im Ausgangsdatensatz vorhandene Länge eines jeden Zeitschritts durch die Pipeline propagiert und verwendet wird. Die Ausgabe des Filters enthält alle individuellen Punkte aus allen Zeitschritten in den Ausgangsdaten sowie ein Feld mit dem Namen "Accumulated Precipitation Amounts", in dem sich jeweils die gesamte gefallene Niederschlagsmenge eines jeden Datenpunkts befindet.

Weitere Informationen befinden sich im Hauptkapitel "Zeitliche Aggregation" dieses Handbuchs.

## Zeitliche Interpolation von Niederschlagsdaten

Niederschlagsdaten können mit dem "Kronos Temporal Interpolation Filter" interpoliert werden. Dabei werden die Niederschlagsraten linear interpoliert und zeitliche Datenlücken aufgefüllt. Bei der Interpolation der Niederschlagstypen zwischen zwei Punkten wird jeweils der Niederschlagstyp des zeitlich näheren Punktes verwendet.



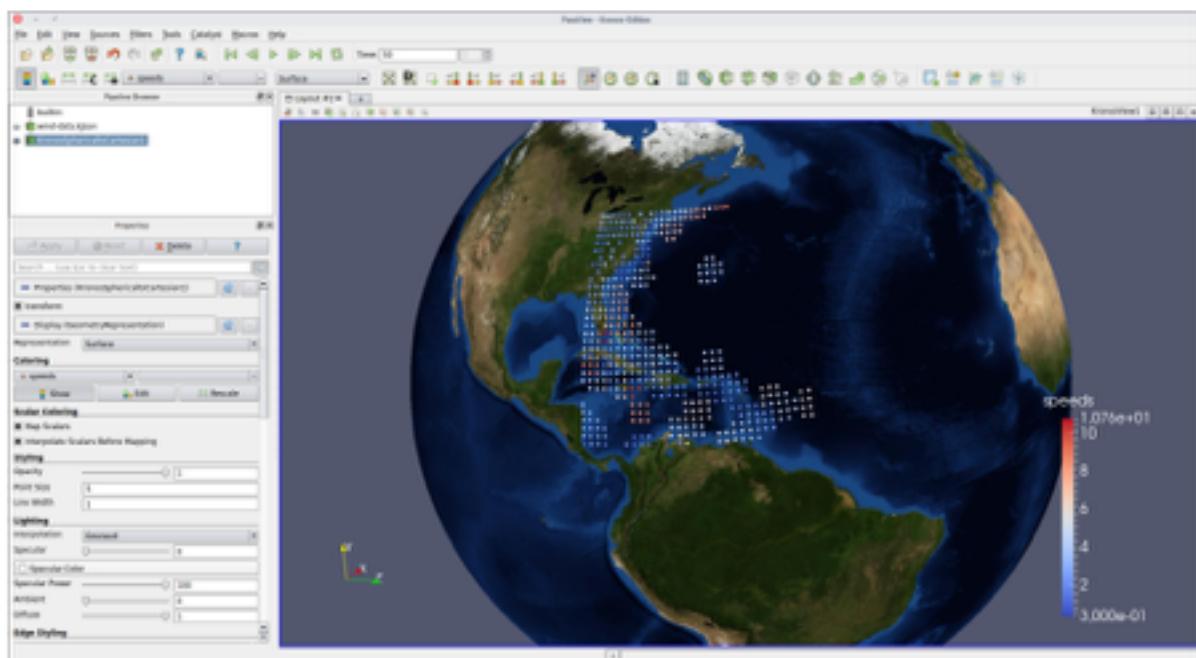
Weitere Informationen befinden sich im Hauptkapitel "Zeitliche Interpolation" dieses Handbuchs.

## Darstellung von Wind

### Visualisierung der Windgeschwindigkeiten als Heatmap

Von Kronos geladene Winddaten enthalten Punkte und deren Windgeschwindigkeit und -richtung in Grad, wobei 0 Grad in Richtung des geographischen Nordpols zeigt und die Gradzahl im Uhrzeigersinn fortschreitet. Für die später beschriebene Flow-Visualisierung enthält jeder Punkt zusätzlich noch einen Windvektor für jeden Zeitschritt. Ist man aber nur an den Windgeschwindigkeiten interessiert, so können diese als Heatmap dargestellt werden. Dazu muss unter "Coloring" das Array "speeds" statt der Option "Solid Color" ausgewählt werden.

Selbstverständlich ist auch hier wieder eine räumliche Interpolation nach der Delaunay-Triangulation möglich.



Visualisierung der Windgeschwindigkeiten.

## Filterung nach Windgeschwindigkeit

Winddaten können auf einen gewissen Geschwindigkeitsbereich beschränkt werden. Hierzu müssen zunächst Winddaten geladen und darauf der "Kronos Wind Speed Threshold Filter" angewandt werden. Dieser stellt nun Bedienfelder zur Einstellung von Minimal- und Maximalwindgeschwindigkeit bereit. Die Slider können über einen Klick auf den blauen Pfeil an die Geschwindigkeitswerte im aktuellen Zeitschritt angepasst werden.

Nach einem Klick auf "Apply" werden nur diejenigen Datenpunkte weitergegeben, deren Windgeschwindigkeit im definierten Bereich liegt.



## Zeitliche Aggregation von Winddaten

Als weitere Form der Datenanalyse können die Werte von Datenpunkten aggregiert werden, indem nach Laden des Datensatzes der "Kronos Temporal Aggregation Filter" angewandt wird.

Speziell bei Winddaten werden hierbei die Durchschnittsgeschwindigkeit und -richtung aller vorhandenen Datenpunkte über die gesamten temporalen Ausmaße des Datensatzes berechnet. Die Ausgabe des Filters enthält alle individuellen Punkte aus allen Zeitschritten in den Ausgangsdaten sowie Felder mit den Namen "Average Wind Directions" und "Average Wind Speeds", in denen sich jeweils das arithmetische Mittel der Windrichtungen bzw. -geschwindigkeiten eines jeden Datenpunktes befindet.

Zudem werden aus den Mittelwerten noch für jeden Punkt ein mittlerer Windvektor berechnet.

Weitere Informationen befinden sich im Hauptkapitel "Zeitliche Aggregation" dieses Handbuchs.

## Zeitliche Interpolation von Winddaten

Winddaten können mit dem "Kronos Temporal Interpolation Filter" interpoliert werden. Dabei werden die Windrichtungen und -geschwindigkeiten linear interpoliert und zeitliche Datenlücken aufgefüllt.

Speziell bei Winddaten werden auch die Windvektoren eines jeden Punktes linear interpoliert und deren Längen korrigiert.

Weitere Informationen befinden sich im Hauptkapitel "Zeitliche Interpolation" dieses Handbuchs.

## Flow-Visualisierung von Winddaten

Um die Flow-Visualisierung in ParaView generell zu ermöglichen muss zunächst das passende ParaView-Plugin geladen werden, das mit einer Standard-Installation von ParaView mitgeliefert wird. Im Folgenden wird das Vorgehen für "Line Integral Convolution" erklärt.

Über den Eintrag "Manage Plugins..." im Menü "Tools" gelangt man zum Plugin Manager. Ist das nötige Plugin nicht bereits hinzugefügt, so lädt man über "Load New..." ein neues Plugin. Nun muss unter Linux eine Plugin-Datei mit dem Name "libSurfaceLIC" ausgewählt werden, die sich für gewöhnlich im "lib"-Ordner des Verzeichnisses befindet, in das ParaView kompiliert wurde, unter Windows muss im "bin"-Ordner desselben Verzeichnisses die "SurfaceLIC.dll" ausgewählt werden. Ist das Plugin bereits vorhanden, muss es eventuell noch geladen werden.

Nach Bestätigung der Auswahl wird das Plugin als geladen angezeigt. Der Plugin Manager kann wieder geschlossen werden.

Um nun Windströmungen zu visualisieren, müssen natürlich zunächst Winddaten geladen werden. Hierbei ist zu beachten, dass diese Daten nur Windrichtung und -geschwindigkeit in gewissen Punkten zu gewissen Zeiten beinhalten. Die Flow-Visualisierung benötigt jedoch Beschleunigungsvektoren für jeden Punkt. Dass diese Vektoren zunächst noch nicht enthalten sind, ermöglicht die einfache Manipulation der Winddaten, ohne dass auf eine Korrektur der Vektorlängen geachtet werden muss. Beispielsweise können die Daten mit Kronos-eigenen Filtern beliebig zeitlich aggregiert oder interpoliert werden.

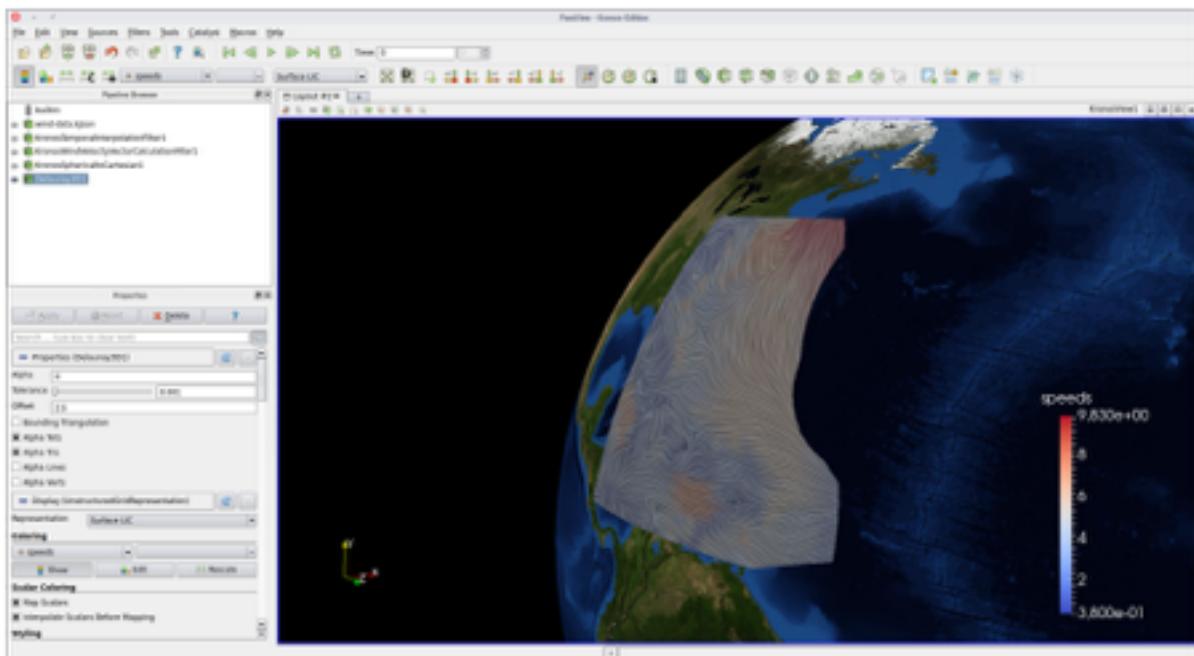
Soll ein solcher Datensatz schließlich als Strömungsvisualisierung dargestellt werden, müssen zunächst die zuvor genannten Windvektoren berechnet werden. Hierzu genügt es, den "Kronos Wind Velocity Vector Calculation Filter" auf die Daten anzuwenden. Zuletzt müssen diese



Vektoren für die Strömungsvisualisierung noch räumlich interpoliert werden, was ein Polygon-Mesh voraussetzt. Dieses ist mit einem Delaunay-Filter zu erreichen.

Außerdem ist zu bedenken, dass die Transformation der Daten auf den Globus die Vektoren ebenfalls korrekt mittransformiert. Um korrekte Vektoren zu erhalten, sollten diese also zunächst mit dem "Kronos Wind Velocity Vector Calculation Filter" generiert und danach mit dem "Kronos Spherical To Cartesian Filter" auf den Globus abgebildet werden.

Nun muss noch die Representation auf "Surface LIC" umgeschaltet werden und die Strömungsvisualisierung wird angezeigt. Zusätzlich ist beispielsweise die Einfärbung der Visualisierung nach Windgeschwindigkeiten möglich.

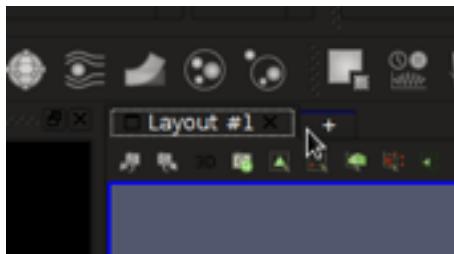


Analog ist die Visualisierung mit sogenannten Streaming Particles möglich.



# Erstellung einer Terrainvisualisierung der Erde

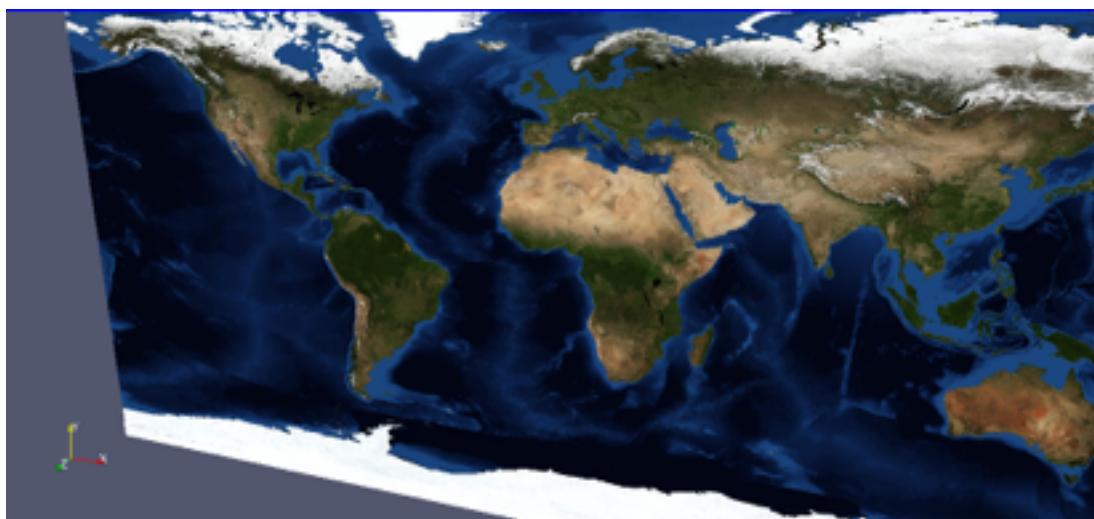
Mithilfe der in Kronos enthaltenen Kronos View wird ein Globus im Hauptfenster dargestellt. Um die Kronos View zu öffnen, muss zunächst eine neue View angelegt werden, was mithilfe der Tab-Leiste in ParaView möglich ist. In der daraufhin erscheinenden Viewauswahlliste kann dann die Kronos View ausgewählt werden.



## Globus und Flachkarte

Um die Globusvisualisierung zwischen einer Kugel und einer mit Plattkartenprojektion dargestellten 2,5D-Karte zu wechseln, wird von Kronos die Schaltfläche "Switch Current Displaymode" bereitgestellt, die über die View-Werkzeugeleiste erreichbar ist.





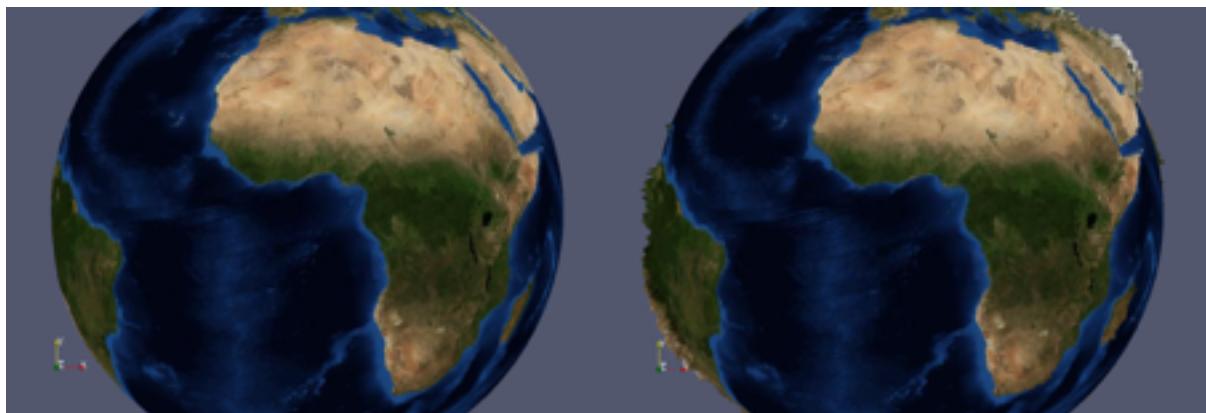
## Manuelle Einstellung der Überhöhung

Um den Überhöhungsfaktor des Globus, der Flachkarte sowie aller mit dem Kronos Terrain Height-Filter transformierten Daten einzustellen, muss die Datei `configuration.json` im res-Unterverzeichnis angepasst werden.

Die Option "heightFactor" im Abschnitt "globe" steuert diesen Überhöhungsfaktor. Durch das Anpassen des Zahlenwerts an der entsprechenden Stelle in der Datei, direkt nach dem Doppelpunkt, gefolgt von einem Neustart von Kronos, kann die Überhöhung je nach Belieben des Nutzers angepasst werden. Wichtig ist, dass der angegebene Wert immer einen Punkt als Dezimalseparator enthält, da es sonst beim Laden der Konfigurationsdatei zu Fehlern kommen kann.



```
{  
  "globe": {  
    "radius": 1.0,  
    "flatMapSize": {  
      "width": 4.0,  
      "height": 2.0  
    },  
    "earthRadius": 6367444.7,  
    "internalPlaneSize": 1.0,  
    "heightFactor": 10.0,  
    "transitionEffectSpeed": 0.125,  
    "tilePoolSize": 64,  
    "timerDelay": 17,  
    "terrainHeightFilter": {  
      "heightmapZoomLevel": 1  
    }  
},
```



Der Unterschied zwischen 10-facher (Standard) und 100-facher Überhöhung.



# Ortssuche

Kronos stellt ein Widget zur Ortssuche bereit, mit der die von Kronos bereitgestellte Städte-Datenbank durchsucht werden kann. Wird die Datenbank zum ersten Mal verwendet, muss diese zunächst konfiguriert werden. Zur Einrichtung der Datenbank sei an die Installationsanleitung verwiesen.

Beim ersten Start wird das Ortssuche-Widget ("Find Places") bereits angezeigt. Ansonsten kann es über das Menü über View->Find Places wieder geöffnet werden. In der Suchleiste des Widgets kann die gesuchte Stadt eingegeben werden. Ein Drücken des Such-Knopfs oder der Enter-Taste durchsucht die Datenbank nach der Eingabe. Wurde eine Stadt mit diesem oder einem ähnlichen Namen gefunden, bewegt sich die Kamera automatisch auf das erste Suchergebnis. Alle der Suche entsprechenden Ergebnisse werden auch unter der Suchleiste, nach absteigender Relevanz sortiert, aufgelistet. Durch einen Klick auf eines der Ergebnisse in der Liste bewegt sich die Kamera auf die entsprechende Stadt. Der Norden ist dann immer oben in der Ansicht. Ist die Liste leer, so wurden keine Städte gefunden, welche der Eingabe entsprechen.

## Mögliche Fehlerquellen

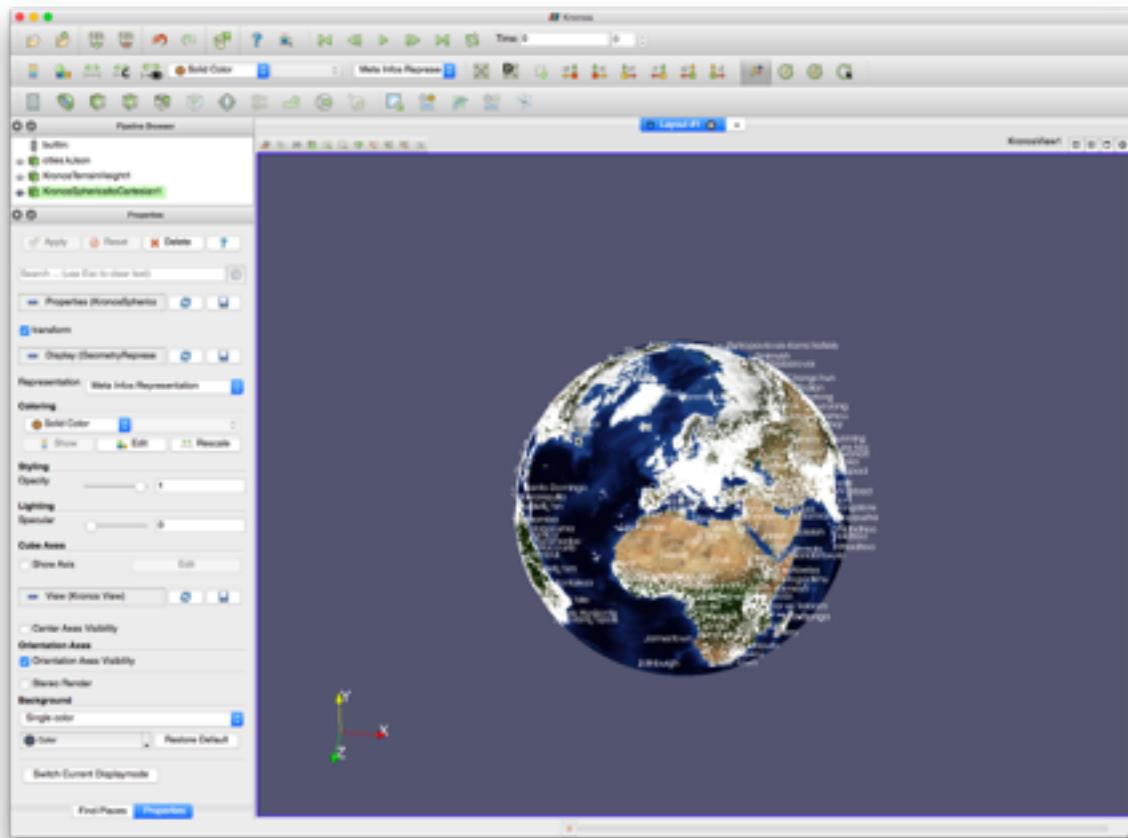
Wird das Ortssuche-Widget ausgegraut dargestellt, deutet dies darauf hin, dass es einen Fehler bei der Verbindung mit der PostgreSQL Datenbank gab. Dann sollte der Benutzer überprüfen, ob die Datenbank korrekt eingerichtet wurde und die PostgreSQL-Instanz auf dem aktuellen Rechner unter localhost, Port 5432 erreichbar ist.

Ist das Widget zwar nicht ausgegraut, aber es werden selbst bei Eingaben wie "Berlin" keine Ergebnisse gefunden, dann wurde die PostgreSQL-Datenbank vermutlich nicht vollständig eingerichtet. In diesem Fall sollte der Nutzer sicherstellen, dass alle in der Installationsanleitung aufgeführten Schritte korrekt ausgeführt wurden, um den SQL-Dump in die Datenbank zu laden.



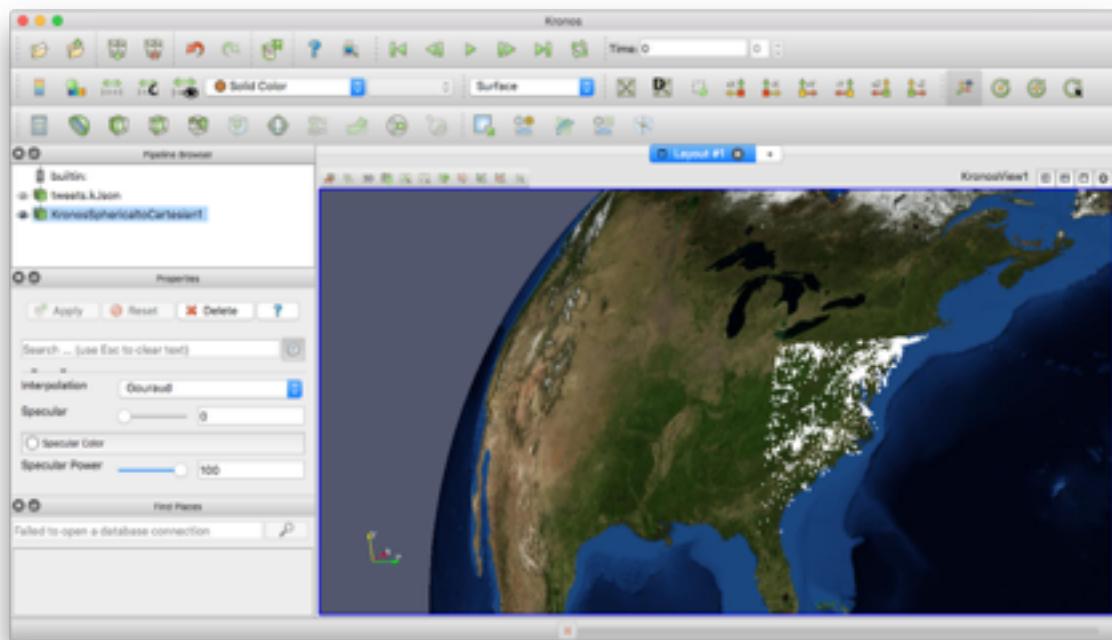
# Ortsdarstellung

Um Ortsnamen auf der Karte darzustellen, müssen die in der Kronos Software beigelegten Städte Daten geladen werden (cities.kJson). Anschließend muss man bei der Repräsentation auf "Meta Info Representation" wechseln.



# Einbinden geolokalisierter Twitter Daten

Zunächst müssen Twitterdaten geladen werden. Ein Beispieldatensatz ist unter dem Namen "tweets.kJson" in "STUPRO/scripts/data-conversion" zu finden. In ParaView kann man unter "File/Open" diese Datei laden. Anschließend auf "Apply" drücken und die Daten werden dargestellt. Dabei werden die Tweets standardmäßig als Punkte gemäß ihrer Ortsinformationen dargestellt.



Twitter Datensatz an der amerikanischen Ostküste

Anschließend unter "Filter/Kronos" den Filter "TwitterFilter" auswählen. Nun kann man die Twitterdaten in den Properties nach verschiedenen Kriterien filtern:

## Filtern nach Autor

In dem Eingabefeld "Autoren" die Autoren, deren Tweets man darstellen will, mit Komma trennt eingeben. Achtung: In dem Beispieldatensatz bestehen die Namen der Autoren aus Zahlen. Hierbei kann man auswählen zwischen "Matching" und "Containing". Bei dem Modus "Matching" werden nur die Autoren angezeigt, die exakt dem eingegebenen Suchtext entsprechen. Bei "Containing" werden auch Autoren angezeigt, bei welchen nur ein Teil des Namens mit dem Suchtext überein stimmt. Nach der Eingabe der Airlines auf "Apply" drücken und die Filterung wird angewandt.

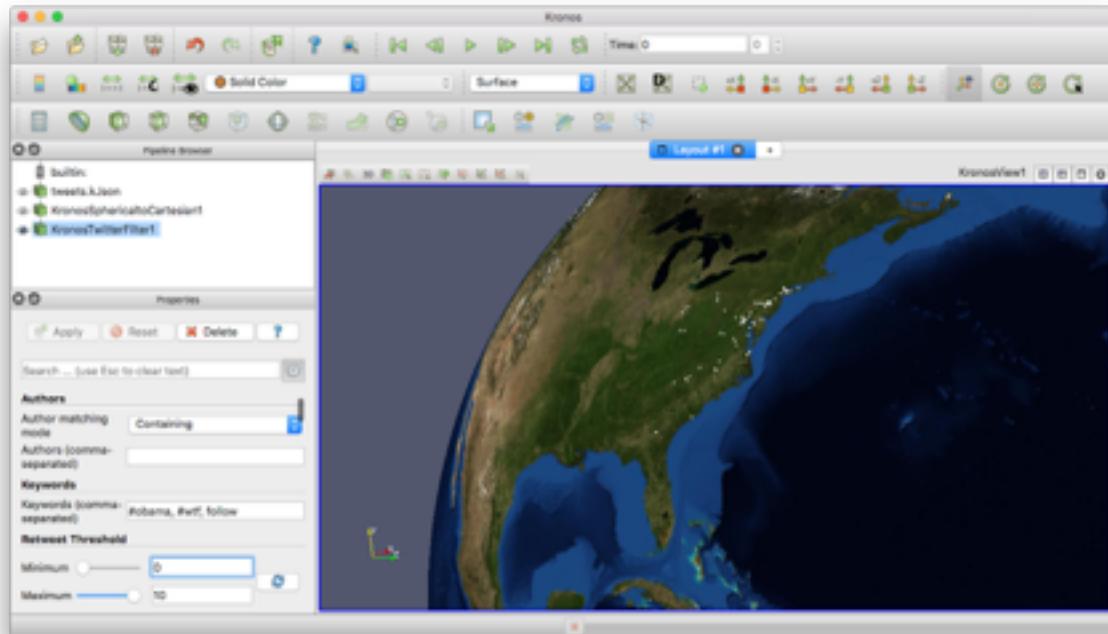
## Filtern nach Schlüsselwörtern und Hashtags

In dem Abschnitt "Keywords" kann man nach Schlüsselwörtern in den Tweets suchen. Dabei bietet sich beispielsweise eine Suche nach Hashtags an. Mehrere Suchbegriffe müssen wieder mit Komma getrennt werden. Ein Drücken des "Apply"-Knopfes wendet den Filter an.



## Filtern nach Anzahl der Retweets (Kriterium für Relevanz eines Tweets)

Hierbei können die angezeigten Tweets nach ihrer Anzahl der Retweets gefiltert werden. Dazu einfach mit den beiden Slidern den gewünschten Bereich auswählen und auf "Apply" drücken.



Twitter-Daten werden nach bestimmten Hashtags und Schlüsselwörtern gefiltert.

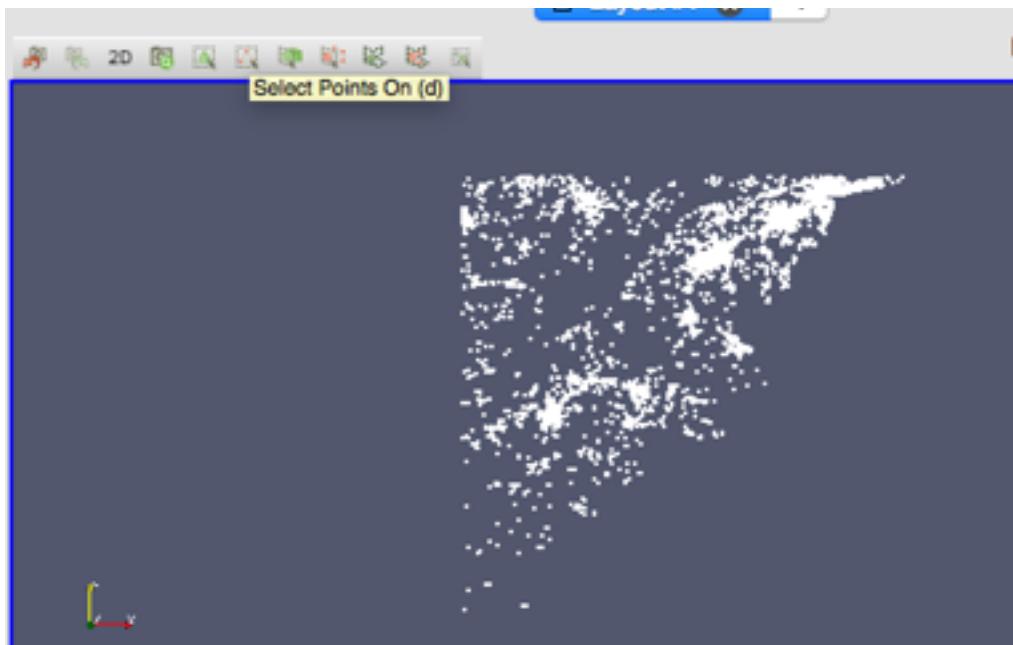
## Visualisierung als Heatmap

Diese Twitter-Daten können noch weiter untersucht werden, insbesondere an welchen Orten besonders viel getwittert wird. Dazu sind weitere Informationen in "Datenübergreifende Filteroptionen" > "Heatmap Density Filter" verfügbar.

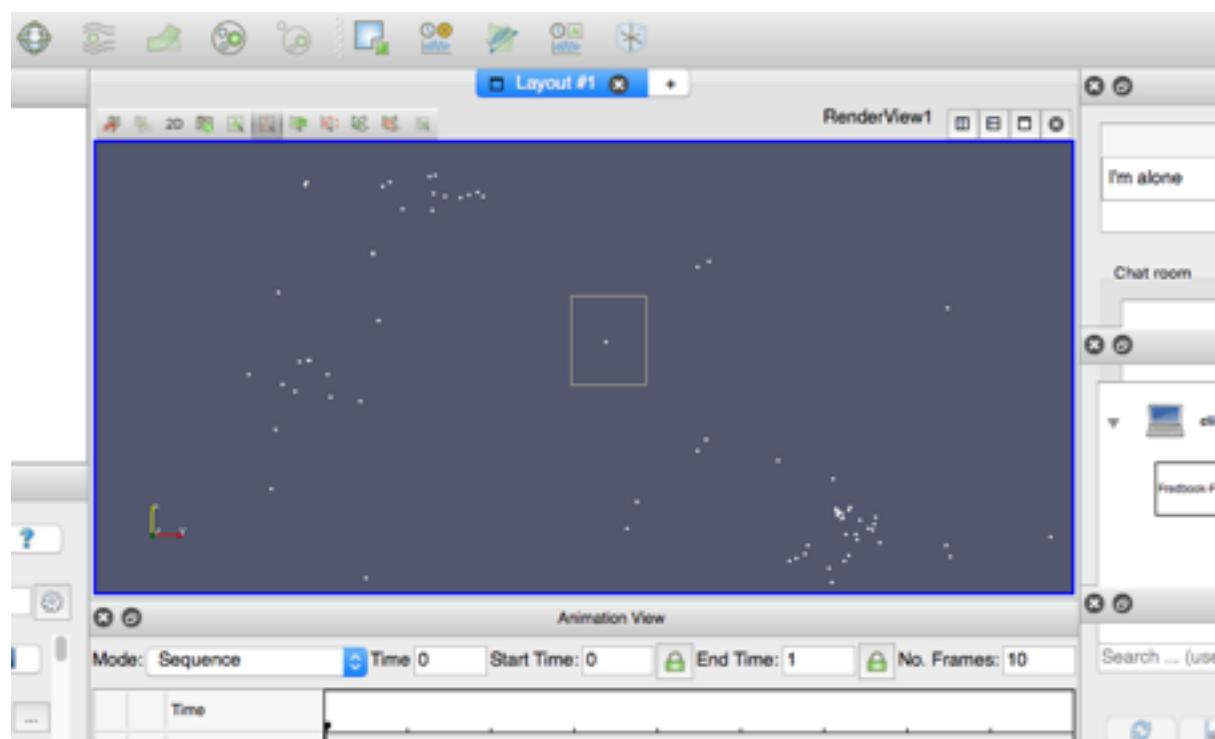
## Anzeigen der Twitternachrichten / Detailansicht

Kronos kann jedoch nicht nur große Datenmengen visualisieren, sondern unterstützt auch die Analyse einzelne Datenpunkte. Um zu einem Tweet die Details anzuzeigen, muss man diesen über das Kronos-Auswahlwerkzeug auswählen. Dieses befindet sich links oben im Visualisierungsfenster ("Select Points"):





Nun können bequem einer oder auch mehrere Tweets ausgewählt werden.



Über dem Visualisierungsfenster auf das "+" drücken und "SpreadSheet View" auswählen. Nun werden alle Datenpunkte als Tabelle angezeigt. Es sollen aber nur die ausgewählten Datenpunkte angezeigt werden, um diese genauer untersuchen zu können. Dazu muss man auf "Show only selected elements" klicken.



Showing tweets.kjson Attribute: Point Data Precision: 6

	Point ID	Points	authors	contents	numberOfRetweets	priorities	time	Show only selected elements.
0	0	-76.5174...	33725780	Reality just s...	10	10	0	
1	1	-77.907...	308414348	Ø£Ù ØÙØ...	10	10	0	
2	2	-76.4581...	415109652	Sunday night...	10	10	0	
3	3	-79.0554...	81273019	Our puppy h...	10	10	0	
4	4	-76.0441...	434623334	My tummy is...	10	10	0	
5	5	-81.1426...	377727799	@Niquelle_K...	10	10	0	
6	6	-74.4151...	80968221	I love this mi...	10	10	0	
7	7	-75.1756...	632547759	Go Cowboys !!!	10	10	0	

Nun werden nur die ausgewählten Tweets mit ihren Detailinformationen angezeigt:

Showing tweets.kjson Attribute: Point Data Precision: 6

	Point ID	Points	authors	contents	numberOfRetweets	priorities	timestamps
0	8368	-77.6196...	298729440	Chilling with ...	10	10	0

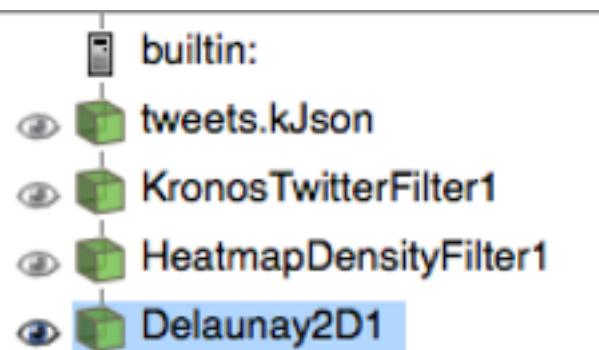


**Profitipp:** Dieses Verfahren lässt sich natürlich auch mit anderen Datenquellen durchführen, um Detailinformationen zu den Datenpunkten zu erhalten.

## Anwendungsbeispiel: Heatmap zu einem bestimmten Hashtag generieren

Eine wichtige Funktion bei der Visualisierung von Tweets ist das Erstellen von Heatmaps zu einem bestimmten Hashtag, um die Popularität von Ereignissen in bestimmten geographischen Regionen vergleichen und untersuchen zu können.

Dazu werden der Kronos-TwitterFilter und der Kronos-HeatmapDensityFilter hintereinander an die Daten geschaltet:



Dank der modularen Bauweise von Kronos kann nun zurück auf den KronosTwitterFilter gewechselt und dessen Suchparameter verändert werden.



Sobald nun auf "Apply" gedrückt wird, ändert sich ebenfalls die generierte Heatmap passend zu den gewählten Suchparametern.



# Zeitliche Interpolation

Viele häufig benötigte temporale Daten und auch diejenigen Datentypen, die speziell von Kronos unterstützt werden (wie Wetterdaten oder Tweets), sind diskreter Natur. Die mit den Punkten assoziierten skalaren Werte ändern sich nur zu jedem ganzzahligen Zeitschritt und bleiben zwischen Zeitschritten konstant.

Für viele Anwendungsfälle ist es jedoch hilfreich, wenn die Werte auch zwischen Zeitschritten sinnvoll fluktuieren, was mit linearer Interpolation erreicht werden kann.

In VTK und damit auch in ParaView existiert bereits ein Filter "Temporal Interpolation". Dieser weist jedoch zwei große Mängel auf. Der Filter kann nur mit Datensätzen umgehen, deren Datenpunkte in jedem Zeitschritt genau an der gleichen Position vorhanden sind. Dies ist aber speziell bei den von Kronos gelesenen Daten nicht vorausgesetzt. Stattdessen können Punkte in manchen Zeitschritten auch fehlen. Außerdem führt der VTK Filter seine Interpolation mit jeder Anfrage eines Zeitschritts erneut durch, was sich deutlich auf die Performance auswirkt.

Um diese Probleme zu lösen, enthält Kronos den "Kronos Temporal Interpolation Filter". Dieser geht einen Schritt weiter und kann auch die oben beschriebenen Datenlücken mittels linearer Interpolation auffüllen. Dabei werden diese vervollständigten Daten zwischengespeichert. Erfolgt nun eine Anfrage an einen beliebigen Zeitschritt, muss nur noch eine einfache lineare Interpolation zwischen den angrenzenden ganzzahligen Zeitschritten erfolgen. Dies beeinträchtigt die Leistung nicht merklich und liefert interpolierte Ergebnisse.

Der Filter unterstützt die zeitliche Interpolation von Temperaturwerten, Niederschlagsraten und -typen und Bewölkungsgraden. Ebenso können Windrichtungen und -geschwindigkeiten zeitlich interpoliert werden, die Windvektoren der Datenpunkte für die Flow-Visualisierung werden hierbei ebenfalls interpoliert und deren Länge automatisch korrigiert. Zuletzt können auch vom "Heatmap Density Filter" generierte Heatmaps zeitlich interpoliert werden, welche im dazugehörigen Kapitel beschrieben wurden.

Die interpolierten Daten des Filters können wie gewohnt in Kronos weiter verwendet werden.



# Räumliche Interpolation

Kronos kann Skalare von Datensätzen räumlich interpolieren. Das bedeutet, dass räumlich zwischen vorhandenen Datenpunkten weitere Datenpunkte eingefügt werden, deren skalaren Werte aus den vorhandenen Datenpunkten linear interpoliert werden.

Dies ist nützlich, um vorhandene Datenlücken zu schließen, um den Gesamteindruck der Visualisierung vollkommener wirken zu lassen.

In Kronos wird zunächst eine Delaunay-Triangulation auf unseren Datensatz angewandt. Diesen finden Sie unter "Filters" > "Alphabetical" > "Delaunay". ParaView interpoliert dann automatisch im Shading die skalen Werte linear im entstehenden Dreiecks-Mesh.

Mit dem "Color Map Editor" kann die Transferfunktion bearbeitet werden, die die Skalare auf eine Farbe mappt.



# Zeitliche Aggregation

Zur Unterstützung der Datenanalyse kann Kronos temporale Daten aggregieren. Hierzu wird der "Kronos Temporal Aggregation Filter" auf den gewünschten Datensatz angewandt. Unterstützt werden Temperatur-, Wind-, Bewölkungs- und Niederschlagsdaten. Die genaue Funktionsweise für die spezifischen Datentypen befindet sich jeweils im den Datentyp zugehörigen Kapitel dieses Handbuchs.

Bei diesem Filter ist zu beachten, dass er Datenpunkte über Zeitschritte hinweg nicht anhand ihrer ID, sondern der Koordinaten identifiziert. Dies ermöglicht es, auch die Werte von Datensätzen zu aggregieren, bei denen Datenpunkte nicht zwingend in jedem Zeitschritt vorhanden sind. Entsprechend enthält die Ausgabe des Aggregationsfilters alle Datenpunkte, die über alle Zeitschritte hinweg im Ausgangsdatensatz vorkommen.

Bei der Ermittlung von Durchschnittswerten werden also nur alle vorhandenen Werte eines jeden Punkts berücksichtigt, bei der Akkumulation von Daten werden die benutzten Raten von Punkten, deren "Integral" berechnet wird (z.B. Niederschlagsraten) so lange als konstant angenommen, bis neue Werte verfügbar sind.

Dieses Verhalten kann angepasst werden, indem zwischen Datensatz und Aggregationsfilter der "Kronos Temporal Interpolation Filter" geschaltet wird. Dadurch werden die Werte aller fehlenden Datenpunkte linear interpoliert.



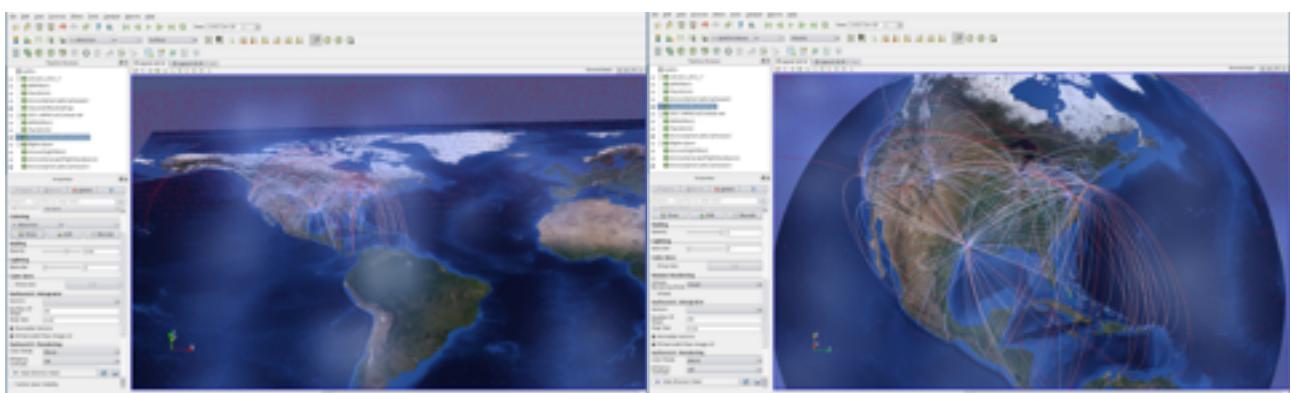
# Transformationsfilter

Mithilfe des Transformationsfilters können Daten im GPS-Format (also Kugelkoordinaten, die in Paraview zunächst kartesisch dargestellt werden) in das kartesische Koordinatensystem überführt werden. Visuell äußert sich der Unterschied darin, dass aus einer Flachkarte (also 2D bzw. 2,5D) ein Globus wird (also 3D). Analog können alle Daten transformiert werden.

Um Daten zu transformieren wählt man unter Filter/Kronos den Filter "Kronos Spherical to Cartesian" aus.

Damit bei Änderung des Display-Modus in der Kronos-View der Transformationsfilter nicht ständig entfernt bzw. neu hinzugefügt werden muss, kann im Filter angegeben werden, ob die Daten transformiert werden sollen. Werden die Daten nicht transformiert, so werden sie auf die Größe der Flachkarte skaliert.

Der Transformationsfilter sollte IMMER angewendet werden. Dabei muss er stets am Ende der Pipeline sein, um mögliche Fehler zu umgehen.



Im Vergleich: Die selben Daten, links auf der 2,5D Karte und rechts mit dem Transformationsfilter.

## Häufige Fehlerquellen

Viele Filter können mit den transformierten Daten nicht korrekt umgehen. Es empfiehlt sich daher, den Transformationsfilter in der Visualisierungspipeline stets an das Ende zu setzen.



# Datenübergreifende Filteroptionen

## Data Point Density Reduction Filter

Dieser Filter ermöglicht die Verminderung der Dichte von Daten um einen vom Nutzer einstellbaren Wert. Da Punkte mit den exakt gleichen Koordinaten ungewollte Folgen haben kann, bietet es sich an vor diesen Filter einen CleanToGrid Filter von Paraview anzuwenden, dieser entfernt identische Punkte.

Die Datenwerte der wegfällenden Punkte werden zusammengefasst und der Durchschnitt dieser berechnet, falls möglich. Nach Bedarf kann zur Filterung auch der kMeans-Algorithmus verwendet werden, dieser ist zwar in der Regel langsamer als der Default-Algorithmus, bietet sich aber bei ungleichmäßig verteilten Punkten beziehungsweise Punktclustern an.

## Heatmap Density Filter

Der Heatmap Density Filter erstellt zu einem gegebenen Datensatz eine Heatmap, die die Datendichte der verfügbaren Daten visualisiert.

Dieser wurde vor allem für den Einsatz mit Twitterdaten konzipiert, kann jedoch aufgrund der Modularität in Kronos mit jedem Datensatz verwendet werden.

Der Filter zeigt in Regionen mit hoher Datendichte einen hohen Wert, in Regionen mit niedriger Datendichte einen niedrigen Datenwert. Somit kann durch ein Zusammenspiel von TwitterFilter und Heatmap Density Filter beispielsweise die Popularität gewisser Hashtags in einer Heatmap visualisiert werden.

Es bietet sich an, außerdem einen Delaunay 2D Filter über den Heatmap Density Filter zu legen, um eine räumliche Interpolation im Shading-Schritt zu erhalten.

Empfohlene Vorgehensweise beim Anwenden des Filters:

1. Den zu filternden Datensatz im Pipeline Browser auswählen. Dies können auch bereits gefilterte Daten sein. Die Daten müssen im vtkPolyData-Format vorliegen.
2. Im Menü "Filters" > "Kronos" > "Heatmap Density Filter" auswählen
3. Auf "Apply" drücken.
4. Unter "Coloring" anstatt "Solid Color" nun "density" auswählen. Dies ist das Property, welches die Datendichte repräsentiert
5. Optional kann im "Color Map Editor" nun noch die Transferfunktion angepasst werden. Dafür in "View > Color Map Editor" auswählen.  
Es ist auch möglich eine logarithmische Transferfunktion anzuwenden, dazu muss im "Color Map Editor" der Haken "Use log scale when mapping data to color" ausgewählt sein.
6. Wir empfehlen nun noch einen Delaunay 2D Filter über die Heatmap zu legen, um die Visualisierung zu vervollständigen. Dieser befindet sich unter "Filters" > "Alphabetical" > "Delaunay".



**Profi-Tipp:** Nach kurzer Zeit werden Sie die Namen der wichtigsten Filter bereits auswendig kennen. Um für Power-User das Auffinden von Filtern in Kronos so schnell wie möglich zu gestalten, gibt es die sogenannte Kronos-Filter-Spotlight Funktion: Mit der Tastenkombination Strg + Leertaste wird ein Suchfenster eingeblendet, wo Sie sofort mit der Eingabe beginnen können. Ist der passende Filter gefunden, kann dies direkt mit der Entertaste bestätigt werden. Und schon ist der Filter geladen. Jetzt müssen Sie nur noch auf "Apply" drücken.



## Terrain Height Filter

Der Terrain Height Filter muss auf Daten angewandt werden, die zusammen mit der Karte oder dem Globus dargestellt werden sollen. Dieser wendet die Überhöhung auch auf die Daten an, damit diese nicht in Bergen "verschwinden", sondern stets auf der Erdoberfläche visualisiert werden. Der Terrain Height filter sollte als letzter Filter vor dem Spherical To Cartesian Filter angewendet werden



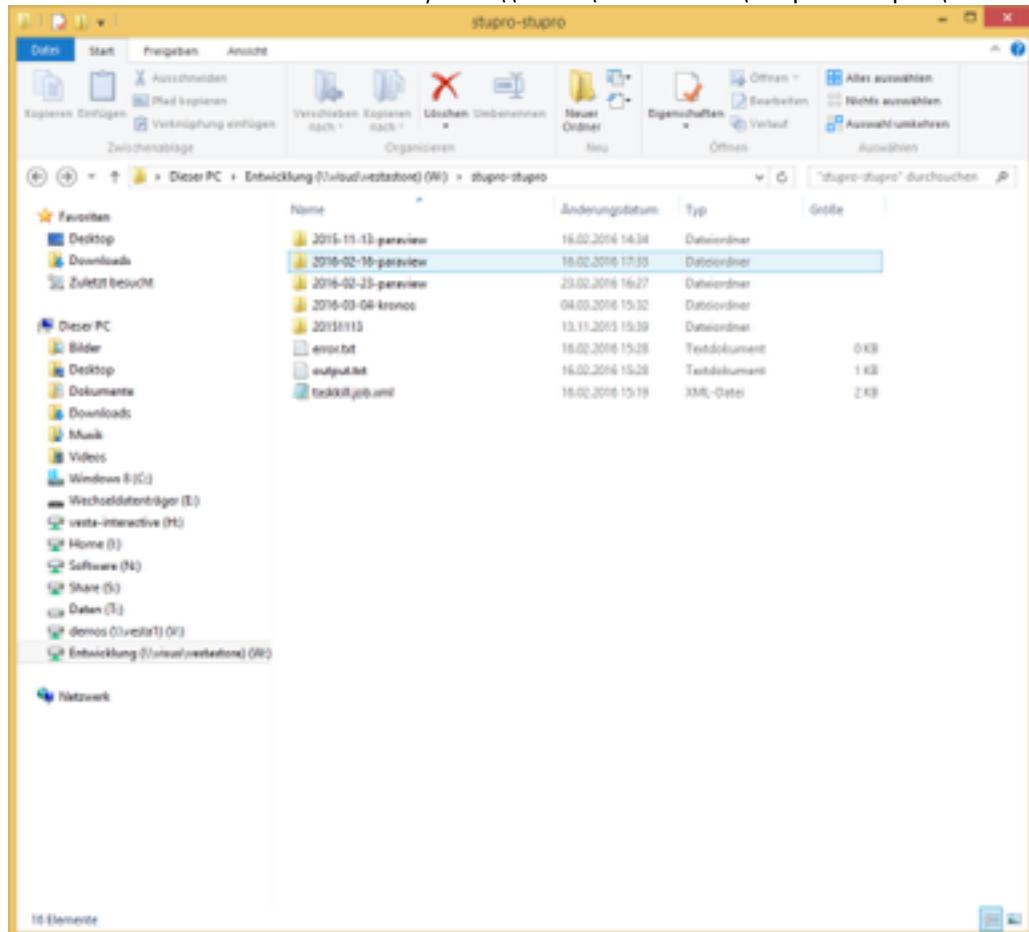
# Anbindung an die Powerwall

Um Kronos an der Powerwall nutzen zu können, müssen vorher folgende Schritte ausgeführt werden:

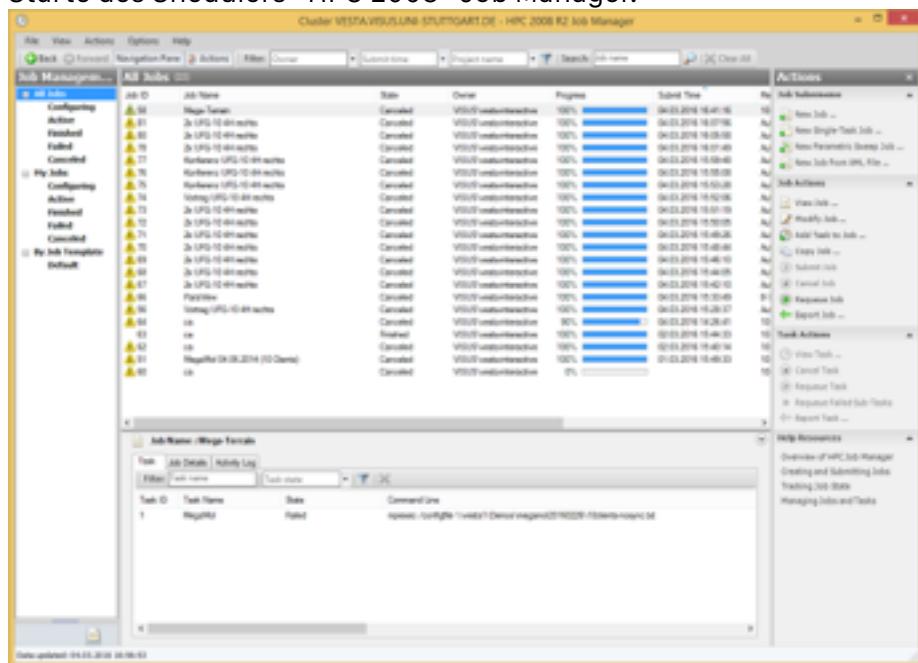
1. Alle Computer, welche Teil der Powerwall sind, müssen sich auf dem Desktop befinden. Und die Beamer für das Rechte Auge sollten eingeschaltet sein, die Beamer für das Linke Auge aus.



2. Kopieren der Kronos Software sowie den Server von der CD in ein von allen Knoten erreichbares Netzwerklaufwerk, z.B. \\visus\vestastore\stupro-stupro\Kronos



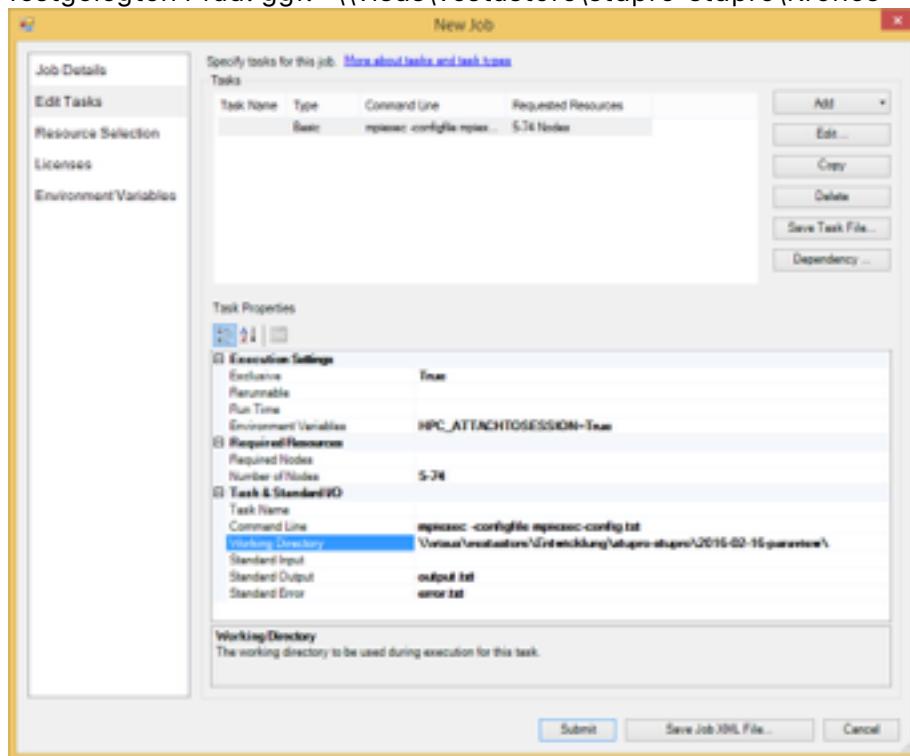
3. Starte des Shudlers "HPC 2008" Job Manager.



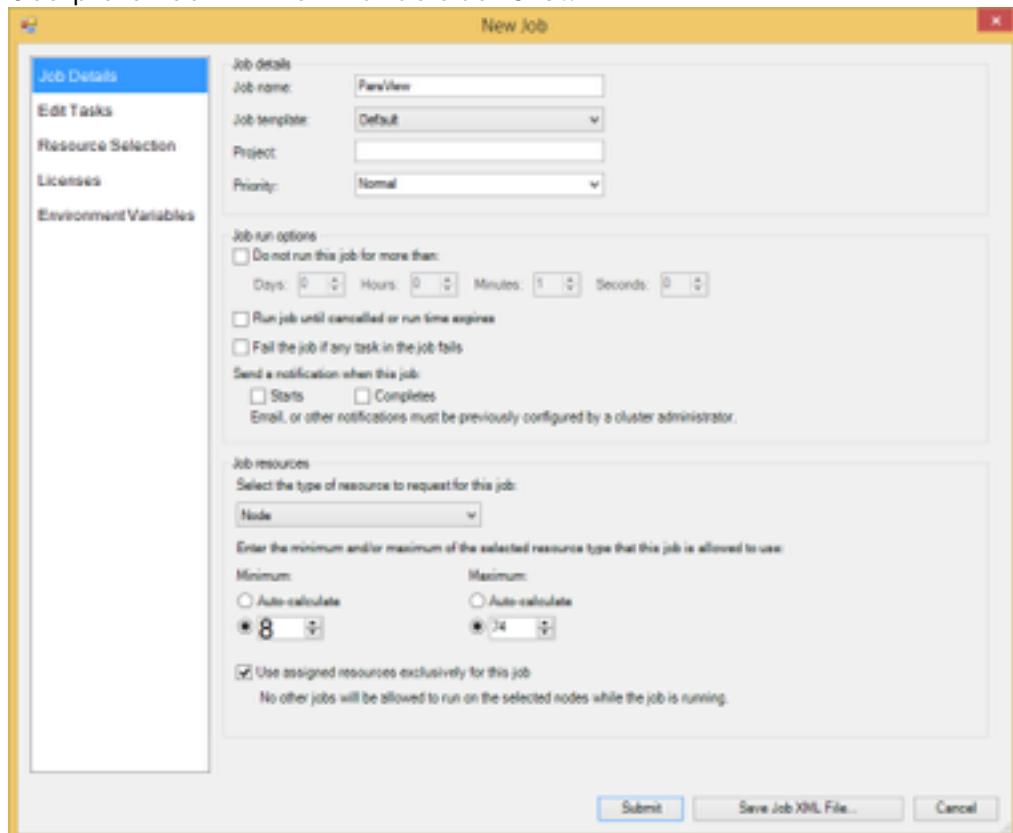
Klicken auf "New Job form XML File..." option, anschliesend kronos.job.xml in \\visus\vestastore\stupro-stupro\2016-03-04-kronos



4. Anpassen des Pfades zur Kronos Software. Ändern des Working Directoys auf in Punkt 2 festgelegten Pfad: ggf. "\\\visus\\vestastore\\stupro-stupro\\Kronos"



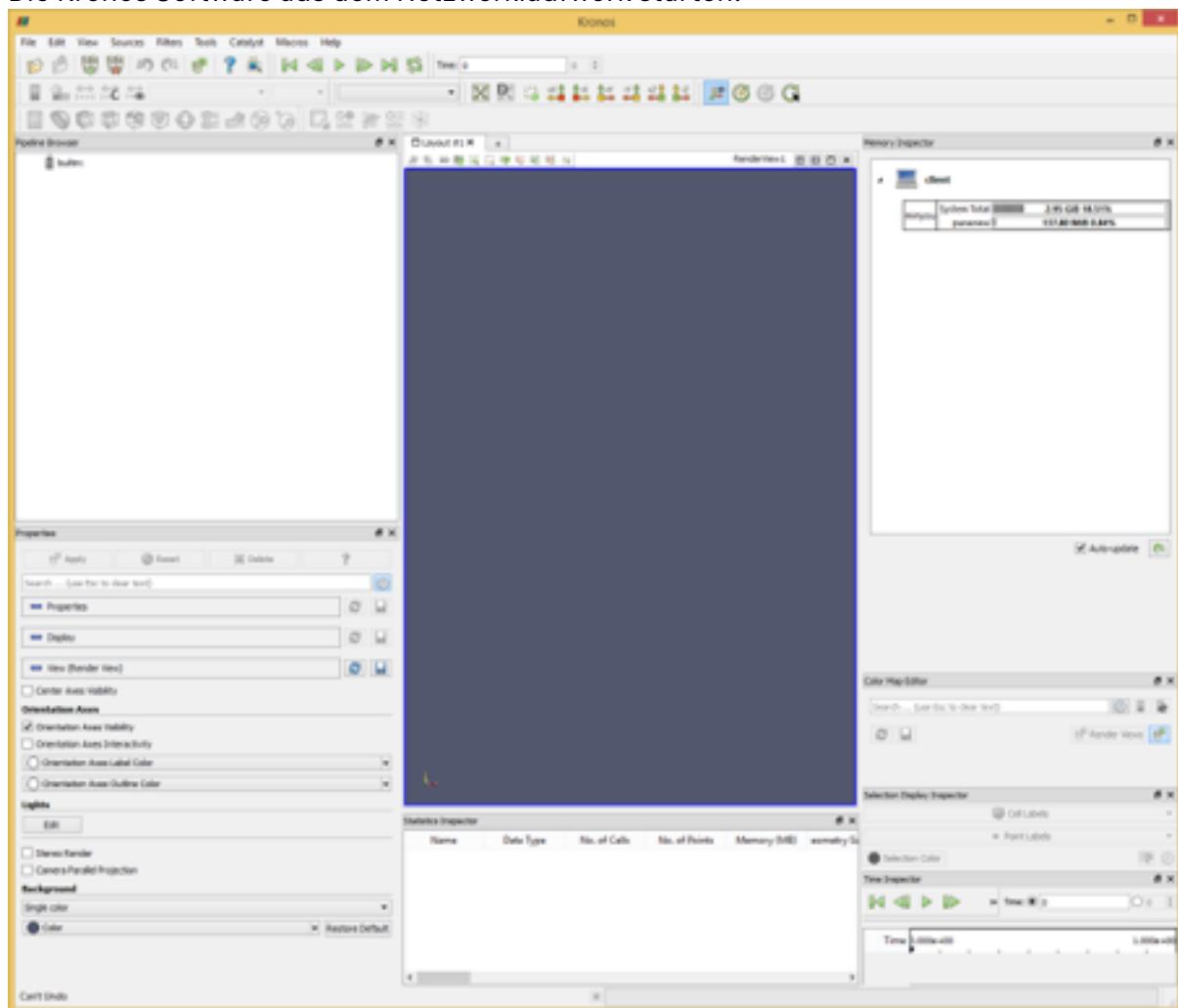
5. Überprüfen ob minimum Variable auf 8 ist.



Scheduler starten, durch Klick auf Submit.

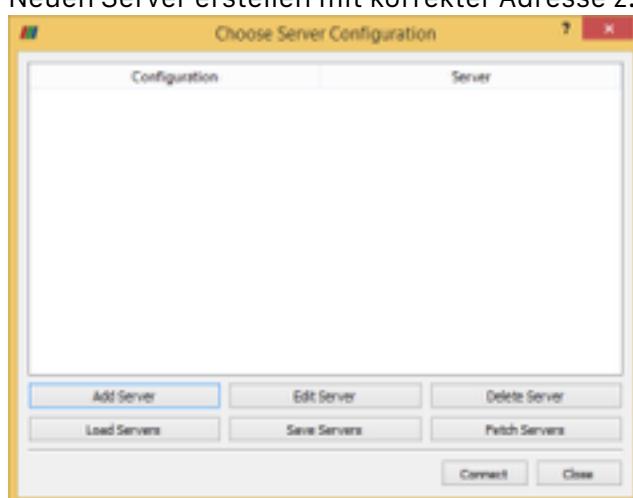


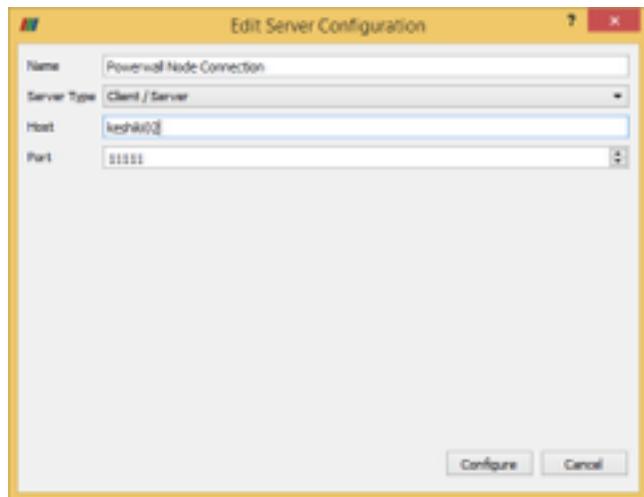
6. Die Kronos Software aus dem Netzwerklauwerk starten.



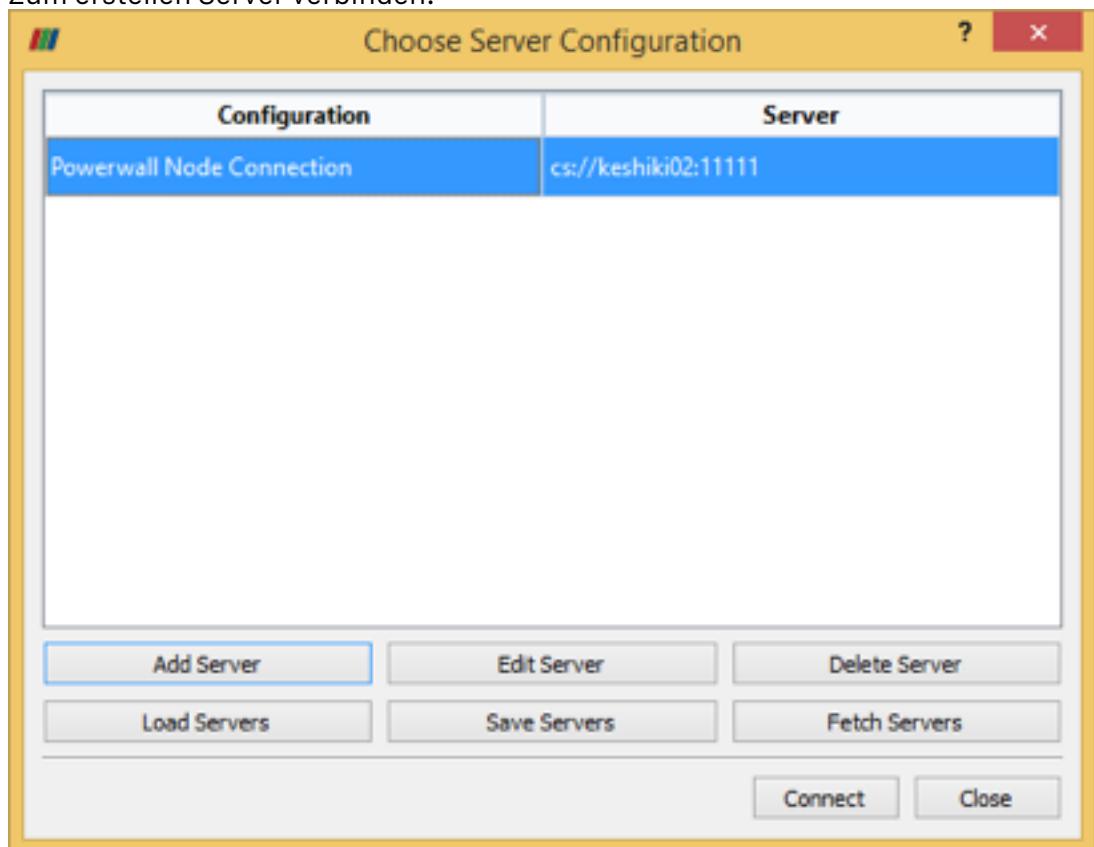
"Connect to Server Button" drücken

7. Neuen Server erstellen mit korrekter Adresse z.B keshiki02....





8. Zum erstellen Server verbinden.



# Einrichten von dynamischen Zoomstufen

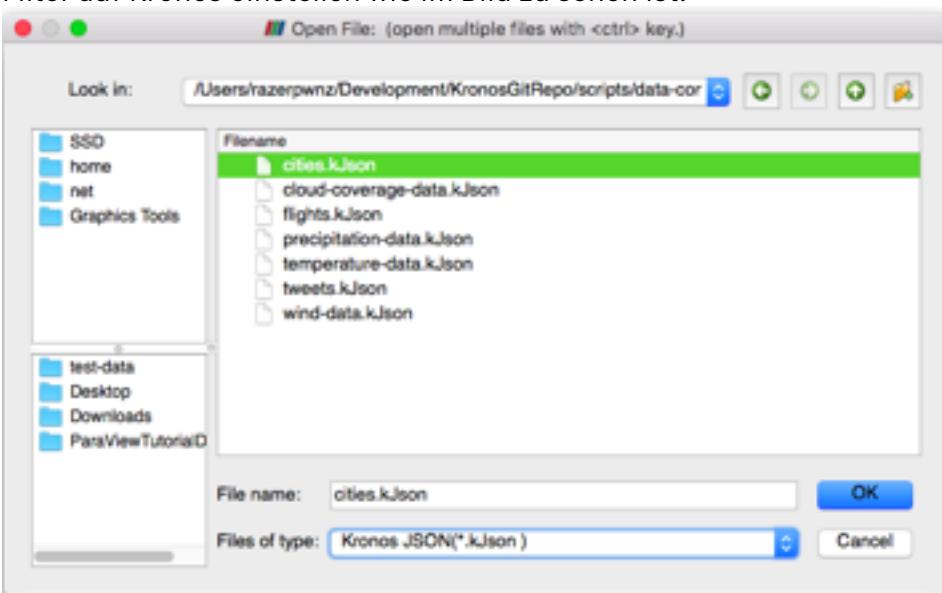
Die Kronos Software bietet die Möglichkeit Daten, welche vom Kronos-Reader eingelesen wurden (.kJson), dynamisch an die Entfernung zum Globus anzupassen. Praktisch bedeutet dies: Je näher sich die Kamera am Globus befindet, desto detaillierter sind die Daten. Um diese Funktion nutzen zu können muss ein Property-Link zwischen Kamera und Reader erstellt werden.

Im Folgenden wird die Erstellung eines Property-Links genauer beschrieben:

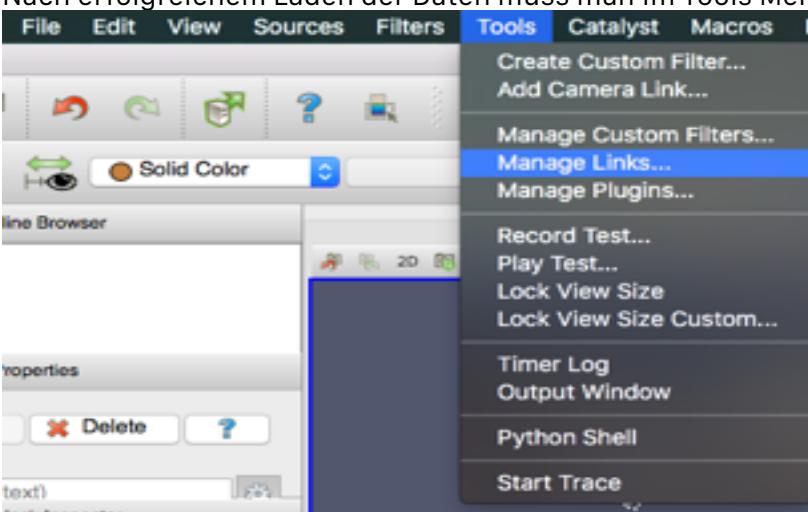
1. Laden von Daten durch Klick auf den "Open"-Button



2. Diese Funktion wird nur von ".kJson"-Daten unterstützt, deshalb sollte man den Filetype Filter auf Kronos einstellen wie im Bild zu sehen ist.



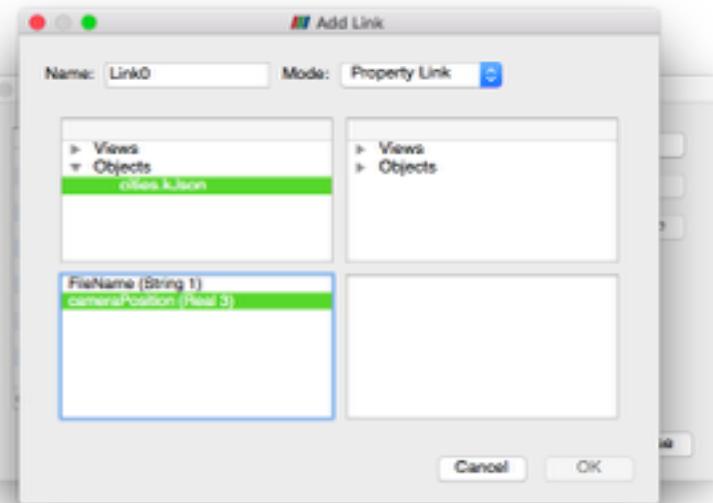
3. Nach erfolgreichem Laden der Daten muss man im Tools Menu "Manage Links" auswählen.



4. Im Link Manager muss "Add" gewählt werden, um einen Link zu erzeugen.
5. Im Mode "dropdown-Menu" muss "Property Link" ausgewählt werden.  
Add Link Menü
6. In der Objects Liste werden alle Daten, die momentan geladen sind, aufgeführt. Hier wählen wir die im Punkt 1 geladene kJson-Datei, wobei es egal ist, ob man das linke oder rechte Menü nutzt.  
Add Link Menü



7. Die Auswahl wird mit dem OK-Button bestätigt.



8. Im Link Manager sollte nun der Property Link aufgeführt sein.

9. Nun sind die dynamischen Zoomlevel aktiv und der Link Manager kann mit Close geschlossen werden.
  10. Falls man den Link wieder deaktivieren möchte, wählt man nach Klick auf den Link den "Remove"-Button.



# Datenformate und Datenkonvertierungen

Im Umfang von Kronos sind verschiedene Beispieldatensätze enthalten, um die Funktionsweise der einzelnen erstellten Module zu demonstrieren. Der Sinn der von Kronos ist es jedoch auch eigene Datensätze verwenden zu können. Um die eigenen Daten in Kronos laden zu können, müssen diese jedoch erst in das Kronos eigene .kJson-Format gebracht werden. Hierzu sind im Lieferumfang unter "/scripts/data-conversion/" diverse Python-Skripts enthalten, die für die Konvertierung der eigenen Daten verwendet werden können und auch für die Erstellung der Testdaten Anwendung fanden.

## Das Kronos JSON-Format

Das .kJson-Format kann Daten enthalten, die entweder nur von einer Ortsinformation oder von Orts- und Zeitinformationen abhängen. Bei beiden werden alle Knoten geographisch angeordnet, letztere Art von Knoten trägt zusätzlich einen Timestamp. Die Knoten erhalten je nach Datentyp eine Metrik, um sie nach Wichtigkeit zu bewerten, was später im Level-of-Detail-System verwendet wird. Das kann beispielsweise die Größe einer Stadt oder die Länge eines Fluges sein. Aus der Metrik soll eine ganzzahlige Priorität zwischen 0 und 10 ermittelt werden, wobei 0 die höchste Priorität darstellt. Dies wird in der Datei dadurch repräsentiert, dass ein Knoten der Wichtigkeit n demjenigen Knoten der Wichtigkeit n - 1 untergeordnet wird, der ihm geographisch am nächsten ist.

### Nicht temporale Daten

Nicht-zeitabhängige Daten, wie etwa Städte, bestehen aus einem einfachen Baum in JSON-Notation, der beispielsweise folgendermaßen aussehen kann:



```
{
  "meta": {
    "dataType": "cities",
    "temporal": false
  },
  "root": {
    "children": [
      {
        "name": "Dubai",
        "longitude": 55.304718,
        "latitude": 25.258171,
        "children": []
      },
      {
        "name": "Los Angeles",
        "longitude": -118.242775,
        "latitude": 34.052223,
        "children": [
          {
            "name": "San Francisco",
            "longitude": -122.418335,
            "latitude": 37.775,
            "children": [
              {
                "name": "Burlingame",
                "longitude": -122.365,
                "latitude": 37.584167,
                "children": []
              }
            ]
          }
        ]
      }
    ]
  }
}
```

## Temporale Daten

Zeitabhängige Daten enthalten zusätzlich einen UNIX-Timestamp, werden aber dennoch primär geographisch angeordnet. Zudem enthält jede Datei das Attribut `timeResolution`, das die Länge eines Zeitschritts in Sekunden angibt.



```
{
  "meta": {
    "dataType": "tweets",
    "temporal": true,
    "timeResolution": 60
  },
  "root": {
    "children": [
      {
        "author": "BarackObama",
        "content": "Progress #president",
        "timestamp": 1439297110,
        "longitude": 55.304718,
        "latitude": 25.258171,
        "children": []
      },
      {
        "author": "elonmusk",
        "content": "Is this working?",
        "timestamp": 1439280065,
        "longitude": -118.242775,
        "latitude": 34.052223,
        "children": [
          {
            "author": "nytimes",
            "content": "This volcano looks like it is erupting #lava
#dangerous",
            "timestamp": 1439288745,
            "longitude": -122.418335,
            "latitude": 37.775,
            "children": []
          }
        ]
      }
    ]
  }
}
```

Lässt man das children-Attribut für die Baumstruktur aus, so enthalten die Knoten jeder Art von Datentyp einen Payload, der die eigentlichen Informationen beschreibt.

Zudem hat jeder dieser Datentypen einen Meta-Header. Payload und Header der einzelnen Datensätze werden in den folgenden Abschnitt spezifiziert.

## Flugdaten

Die verwendeten Flugdaten stammen von <http://openflights.org/data.html> und bestehen aus mehreren .csv-Dateien. Es können bei Bedarf auch Daten anderer Seiten verwendet werden. Damit der Kronos Json-Reader diese Daten verarbeiten kann, muss aus diesen Daten zunächst eine Flightroutes.csv-Datei erstellt werden, die folgende Spalten enthält:

id, srcLatitude, srcLongitude, destLatitude, destLongitude, airline, srcAirport, destAirport  
Seite 51 von 57



- id: durchlaufende Nummerierung der Flüge
- scrLatitude: Breitengrad des Startflughafens
- srcLongitude: Längengrad des Startflughafens
- destLatitude: Breitengrad des Zielflughafens
- destLongitude: Längengrad des Zielflughafens
- airline: Name der Airline
- srcAirport: Airportkürzel des Startflughafens
- destAirport: Airportkürzel des Zielflughafens

Ist eine solche Datei erstellt, kann durch ausführen des unter "/scripts/data-conversion/FlightData/flights.py" liegendem Skripts eine flights.json erzeugt werden, welche noch in .kJson umbenannt werden muss und fortan vom Kronos Json-Reader eingelesen werden kann. Zu beachten ist hierbei, dass sich die .csv-Datei im gleichen Ordner wie das Skript befinden muss.

## kJSON Format für Flüge (nicht temporal)

Header:

```
"meta": {
  "dataType": "flights",
  "temporal": false
}
```

Payload:

```
"startPosition": {
  "airportCode": "DXB",
  "longitude": 55.304718,
  "latitude": 25.258171
},
"endPosition": {
  "airportCode": "LAX",
  "longitude": -118.242775,
  "latitude": 34.052223
},
"airline": "Lufthansa"
```

## Städtedaten

Die verwendeten Städtedaten stammen von <http://www.geonames.org/export/web-services.html> und bestehen aus einer .csv-Datei, welche mit einem Java-Skript in eine xml-Datei konvertiert wurde, welche in einer 10-stufigen Prioritätskette geschachtelt ist. Aus dieser Schachtelung wird in Kronos später das dynamische LOD erzeugt. In den Testdaten ist dies nach der Städtegröße (Einwohnerzahl) und der Nähe zur nächst größeren Stadt geschehen. Diese xml-Datei kann bei Bedarf durch eine eigene, mit anderen Metriken ersetzt werden.

Ist eine solche Datei erstellt, kann durch Ausführen des unter "/scripts/data-conversion/CityData/cities.py" liegendem Skripts eine cities.json erzeugt werden, welche noch in .kJson umbenannt werden muss und fortan vom Kronos Json-Reader eingelesen werden kann.

## kJSON Format für Städte (nicht temporal)

Header:



```
"meta": {  
    "dataType": "cities",  
    "temporal": false  
}
```

Payload:

```
"name": "Dubai",  
"longitude": 55.304718,  
"latitude": 25.258171
```

## Wetterdaten

Die verwendeten Wetterdaten (Niederschlag, Temperatur, Wind und als Bonus Bewölkung), stammen aus der Dark Sky Forecast API. Von dieser Quelle lassen sich mittels des unter "/scripts/data-conversion/WeatherData/weather-data-collector.py" liegendem Skripts eigene Datensätze herunterladen. Hierzu muss zunächst die im gleichen Ordner liegende forecastio.config-Datei um einen eigenen API-Key ergänzt werden (den Key einfach reinschreiben). Zusätzlich müssen in der weather-data-collector.json-Datei im Meta-Header, die gewünschten Daten ersetzt werden, wie zum Beispiel Start- und Endzeitpunkt und die Koordinaten des oberen linken und des unteren linken Eckpunkts, aus welchen die Wetterdaten heruntergeladen werden.

Das Skript schreibt die gesammelten Daten aller Wetterquellen in die gleiche json-Datei.

Damit der Kronos Json-Reader die Daten in vtkPolyData konvertieren und laden kann, müssen die Daten noch auf jeweils eine Datei für jede Wetterquelle aufgesplittet werden. Dies geschieht mit dem weather-converter.py Skript. Nach dem Ausführen des Skripts erhält man jeweils eine Datei pro Wetterquelle, welche noch in .kJson umbenannt werden müssen und fortan vom Kronos Json-Reader eingelesen werden können.

### kJSON Format für Niederschlag (temporal)

Header:

```
"meta": {  
    "dataType": "precipitation",  
    "temporal": true,  
    "timeResolution": 60  
}
```

Payload:

```
"precipitationRate": 24, // Millimetres per hour  
"precipitationType": "snow", // Could be 'rain', 'snow', 'sleet',  
'hail' or 'NoData'  
"timestamp": 1439288745,  
"longitude": -122.418335,  
"latitude": 37.775
```

### kJSON Format für Temperatur (temporal)

Header:



```
"meta": {  
    "dataType": "temperature",  
    "temporal": true,  
    "timeResolution": 60  
}
```

Payload:

```
"temperature": 21, // Degrees Celsius  
"timestamp": 1439288745,  
"longitude": -122.418335,  
"latitude": 37.775
```

## kJSON Format für Wind (temporal)

Header:

```
"meta": {  
    "dataType": "wind",  
    "temporal": true,  
    "timeResolution": 60  
}
```

Payload:

```
"speed": 0.4, // Metres per second  
"direction": 120, // Degrees, with 0 being true North, progressing  
clock-wise  
"timestamp": 1439288745,  
"longitude": -122.418335,  
"latitude": 37.775
```

## kJSON Format für Bewölkung (temporal)

Header:

```
"meta": {  
    "dataType": "cloudCover",  
    "temporal": true,  
    "timeResolution": 60  
}
```

Payload:

```
"cloudCover": 0.4, // Cloud coverage (value in interval [0, 1])  
"timestamp": 1439288745,  
"longitude": -122.418335,  
"latitude": 37.775
```



## Twitterdaten

Ein Testdatensatz von Twitterdaten wurde uns in Form einer tabellierten Textdatei zur Verfügung gestellt. Durch ein Skript werden die Daten in unser spezifiziertes kJson Format konvertiert.

Selbstverständlich können auch eigene Datensätze eingebunden werden. Dazu müssen die Daten nur in das von uns vorgegebene Format konvertieren werden.

Falls die Daten ebenfalls in Form einer tabellierten Textdatei vorhanden sind, kann das von uns angefertigtes Python-Script genutzt werden, um die kJson Datei zu erhalten.

Das Skript befindet sich in "/scripts/data-conversion/WeatherData/tweets\_tabbed.py".

### kJSON Format für Tweets (temporal)

Header:

```
"meta": {  
    "dataType": "tweets",  
    "temporal": true,  
    "timeResolution": 60  
}
```

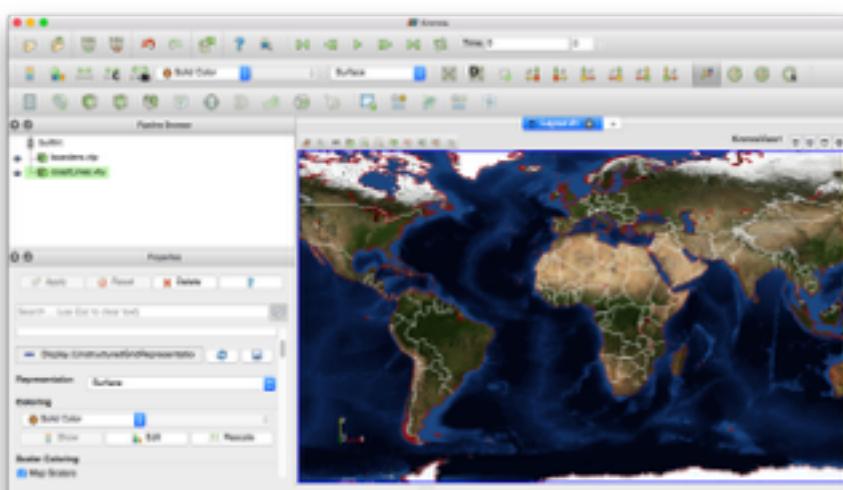
Payload:

```
"author": "nytimes",  
"content": "This volcano looks like it is erupting #lava #dangerous",  
"timestamp": 1439288745,  
"longitude": -122.418335,  
"latitude": 37.775
```

## Küstenlinien und Ländergrenzen

Der Kronos Software liegen Datensätze zur Darstellung von Küstenlinien und Ländergrenzen bei.

Die Daten befinden sich auf der Installation CD der Software und können über die  "Open" Schaltfläche geladen werden. Mit Hilfe des Transformations Filters können die Daten für den Globus transformiert werden. Für mehr Details siehe Kapitel über Transformationsfilter.



## Erweiterung durch neue Datenquellen

Neben den bereits bereitgestellten Datenquellen, können vergleichsweise einfach auch eigene angelegt werden. Dafür sind Anpassungen im Quellcode nötig, allerdings nur an wenigen Stellen, welche an dieser Stelle kurz beschrieben werden. Alle zu ändernden Klassen können in folgendem Pfad gefunden werden: "src/Reader/DataReader"

Basis ist wie bei allen anderen Datenquellen eine .kJson-Datei wie sie analog zum Kapitel "Das Kronos JSON-Format" erstellt werden muss. Dies kann auf verschiedenen Wegen geschehen und wird deswegen an dieser Stelle nicht weiter beschrieben.

Zunächst muss eine neue Model-Klasse für die neue Datenquelle angelegt werden, welche entweder von TemporalDataPoint oder von NonTemporalDataPoint erbt. Diese Klassen erben wiederum von der Klasse DataPoint. Wie eine solche Klasse umgesetzt werden kann, lässt sich anhand einer der bereits vorhandenen Klassen sehen (zum Beispiel in CityDatapoint oder in TweetDataPoint).

Anschließend muss der Data-Klasse mitgeteilt werden, dass es eine neue Datenquelle gibt, indem man sie in die QMap einfügt. Hier kann sich ebenfalls an den bereits vorhandenen Datenquellen orientieren. Ähnliches muss auch in der Klasse JsonReaderFactory geschehen.

Natürlich muss Kronos die Daten aus der kJson-Datei einlesen und in Punktdaten konvertieren, was im JsonReader geschieht. In der Methode indexDataPoints() wird für jeden Datenpunkt der eingelesenen Datei ein Datenpunkt der vorhin erstellten Modelklasse erzeugt, indem man die Switch-Case-Anweisung um einen neuen Case erweitert.

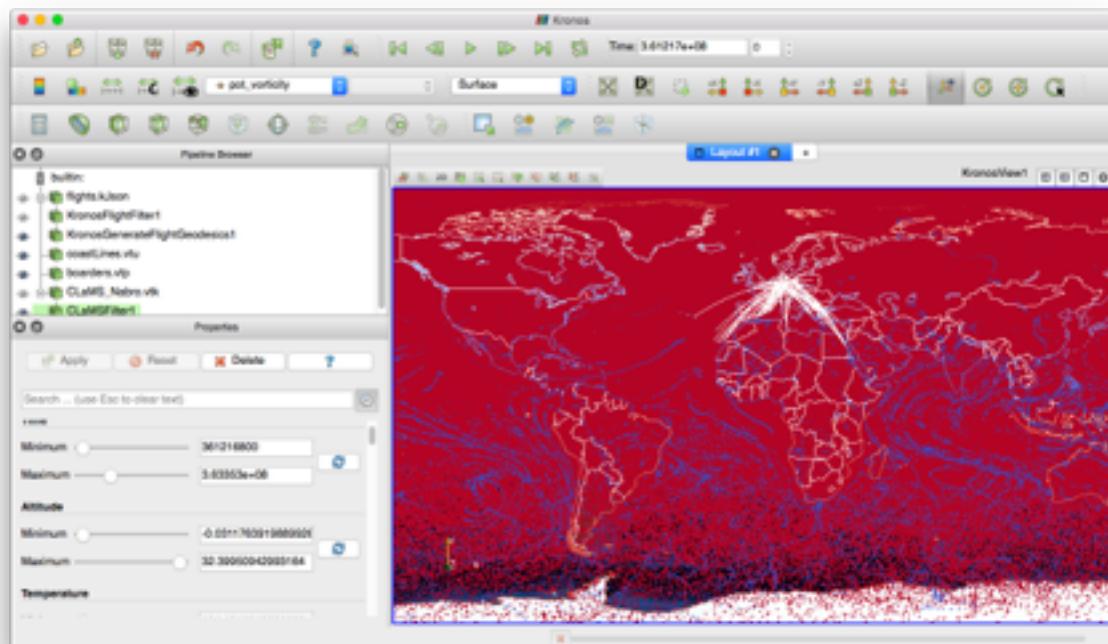
Schlussendlich werden in der Klasse PolyDataSetHelper aus den Punktdaten, welche der Json-Reader eingelesen hat, ein vtkPolyData erzeugt, welche Kronos schließlich visualisieren kann. Dies geschieht in der Methode createPolyDataSet(). Darin ist ein Switch Case-Konstrukt enthalten, mit je einem Case pro Datenquelle. An dieser Stelle muss analog zu den anderen, ein neuer Case für die neue Datenquelle erzeugt werden. Dabei werden die Punktdaten für jede Datenart, die der Json-Reader zuvor erzeugt hat, in ein eigenes Array geschrieben, welche dann am Ende dem PolyData zugefügt werden.

Von jetzt an kann die neue Datenquelle von Kronos eingelesen werden und mit Hilfe von Filtern visualisiert werden.

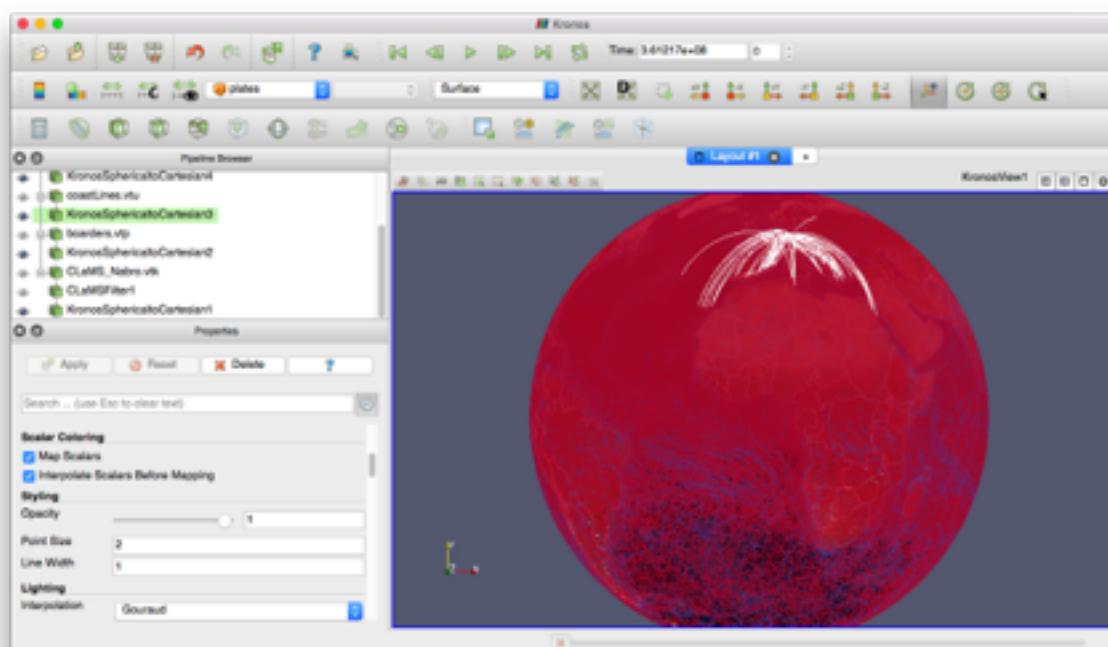


# Visualisierung von mehreren Datensätzen

Die Kronos Software unterstützt das Visualisieren von mehreren Daten gleichzeitig dies kann wie folgt aussehen:



Clams Daten (Nabro) mit Geodäten von Flugdaten mit Zielflughafen Berlin sowie Länder und Küstenlinien



Clams Daten (Nabro) mit Geodäten von Flugdaten mit Zielflughafen Berlin sowie Länder und Küstenlinien

