# „The Dimension of Clouds and Bones“: real-time volume rendering of large scale meteorological data

Projektdokumentation

Marco Winter, Jan Storm
14.7.2019

# Inhalt

# 1. Einleitung

Dieses Dokument dient als Handbuch und Software-Dokumentation zur Nutzung und Weiterentwicklung der im Rahmen des Projekts „The Dimension of Clouds and Bones“ im Sommersemester 2019 von Jan Storm und Marco Winter erstellten Anwendung.

Zweck der entwickelten Anwendung ist die volumetrische Echtzeit-Darstellung von meteorologischen Daten der europäischen Copernicus-Mission. Mithilfe von Transferfunktionen, deren Anpassungen in Echtzeit dargestellt werden können, sowie zwei verschiedenen Darstellungsformen („Karte“/“Globus“) wird dem Nutzer die genaue Beobachtung und Untersuchung von meteorologischen Phänomenen und Gegebenheiten ermöglicht.

Im Rahmen dieser Dokumentation wird zunächst Schritt für Schritt die Nutzung der Anwendung erläutert. Anschließend werden die Projektarchitektur sowie die einzelnen Komponenten und Scripts sowie deren Zusammenhänge vorgestellt. Zuletzt erfolgt ein Ausblick auf mögliche Erweiterungen der Anwendung sowie eine Auflistung aller genutzten Referenzen und Quellen.

# 2. Nutzung der Anwendung

## 2.1 Datenimport und -umwandlung

Aufgrund der Beschaffenheit der Datenquelle und ihrer technischen Restriktionen (Wartezeiten etc.) ist es nicht möglich, einen Live-Zugriff auf die gewünschten meteorologischen Daten zu erhalten. Stattdessen wird entweder das Webportal oder eine von den Verantwortlichen auf Seiten der Copernicus-Mission zur Verfügung gestellte Schnittstelle[1] für die Auswahl und den Download der gewünschten Daten genutzt. Hierfür wurde dem Projektteam ein Python-Script zur Verfügung gestellt, in welchem die Höhenlevel, Parameter und der Zeitraum nach Belieben eingestellt werden können. In beiden Fällen ist eine Registrierung unter https://cds.climate.copernicus.eu/user/register vonnöten.

Im unter Abb. 1 gezeigten Beispiel werden die Werte des Geopotentials in diversen Höhenlagen bzw. Luftdruck-Leveln in hPa (1000 entspricht hierbei der Erdoberfläche) für alle Tage im Februar 2018 jeweils um 0:00 Uhr, 6:00 Uhr, 12:00 Uhr und 18:00 Uhr angefragt und heruntergeladen.

```python
import cdsapi

c = cdsapi.Client()

c.retrieve(
    'reanalysis-era5-pressure-levels',
    {
        'product_type':'reanalysis',
        'format':'netcdf',
        'variable':'geopotential',
        'pressure_level':[
            '1','2','3',
            '5','7','10',
            '20','30','50',
            '70','100','125',
            '150','175','200',
            '225','250','300',
            '350','400','450',
            '500','550','600',
            '650','700','750',
            '775','800','825',
            '850','875','900',
            '925','950','975',
            '1000'
        ],
        'year':'2018',
        'month':'02',
        'day':[
            '01','02','03',
            '04','05','06',
            '07','08','09',
            '10','11','12',
            '13','14','15',
            '16','17','18',
            '19','20','21',
            '22','23','24',
            '25','26','27',
            '28'
        ],
        'time':[
            '00:00','06:00','12:00',
            '18:00'
        ]
    },
    'download.nc')
```

Abb. 1: Beispiel Python-Script zum Download eines Beispiel-Datensatzes

[1] https://cds.climate.copernicus.eu/api-how-to

Die heruntergeladenen Daten liegen im offenen Format NetCDF vor und müssen vor der Verwendung durch die Anwendung in ein nutzbares Format konvertiert werden. Hierfür wurde zu Beginn des Projekts ein Matlab-Script zur Verfügung gestellt, welches einen NetCDF-Datensatz in eine Reihe von TIF-Bildern umwandelt, welche zur Darstellung genutzt werden können.

Dieses Script wurde im Rahmen des Projekts erweitert und ist in seiner aktuellen Fassung in der Lage, den NetCDF-Datensatz hinsichtlich der gewählten Attribute (Temperatur, Geopotential etc.) aufzuteilen. Hierfür werden die entsprechenden Metadaten in einer XML-Datei festgehalten und entsprechende Unterverzeichnisse für alle Attribute erstellt. Zwischen diesen kann in der Anwendung nach dem initialen Ladevorgang problemlos gewechselt werden.

Das Ergebnis des Scripts umfasst wie bereits erwähnt eine oder mehrere Serien von TIF-Bildern, welche in verschiedenen Unterverzeichnissen nach folgendem Muster gespeichert werden:

Data\Attributname\DD-MM-YYYY_HH-MM-SS\Attributname_Level(hPa).TIF

So ergibt sich bspw. für die Temperatur am 01.01.2019 um 12:00 Uhr in den drei niedrigsten Höhenlagen (1000, 975, 950 hPa) folgende Dateipfade:

data\temperature\01-01-2019_12-00-00\temperature_950.TIF
data\temperature\01-01-2019_12-00-00\temperature_975.TIF
data\temperature\01-01-2019_12-00-00\temperature_1000.TIF

## 2.2 User Interface

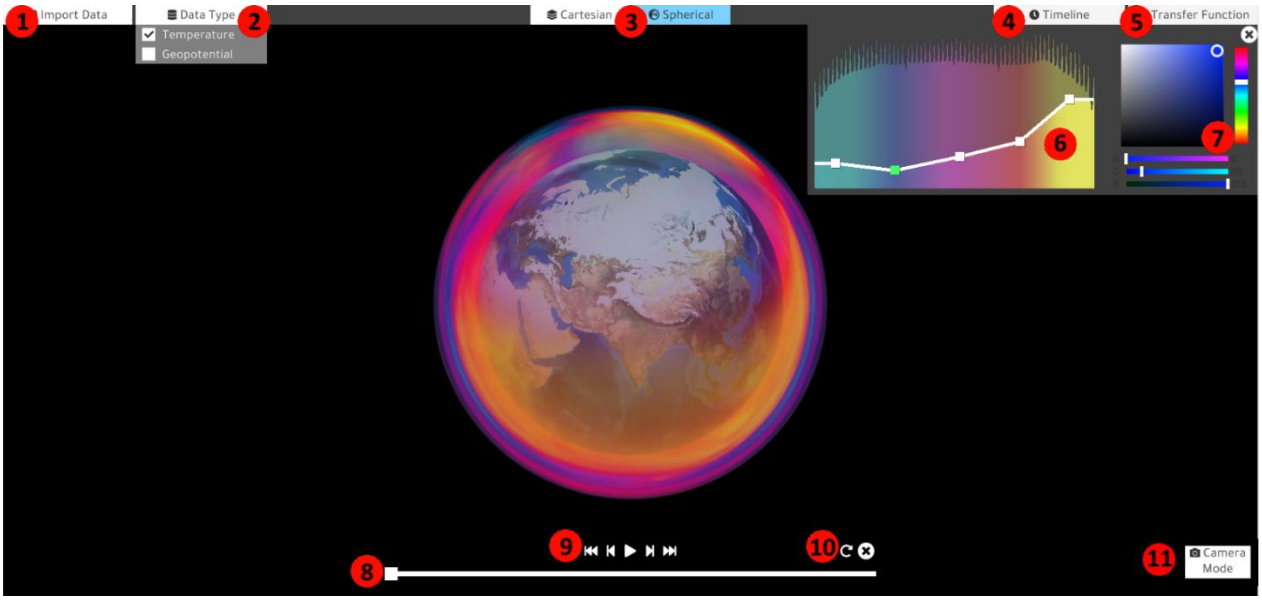


Abb. 2: UI-Elemente

Die Bedienung der Anwendung erfolgt über zehn Elemente bzw. Elementgruppen (siehe Tab. 1). Die Aktivierung bzw. Deaktivierung der Elemente erfolgt, sofern nicht anders beschrieben, über einen Klick der linken Maustaste.

Die Kameraführung erfolgt je nach Kameramodus auf zwei unterschiedliche Weisen. Bei gebundener Kameraführung kann das dargestellte Volumen (Karte/Globus) durch das Ziehen mit der rechten Maustaste rotiert werden. Darüber hinaus dient das Mausrad zum Zoomen der Ansicht.

Bei freier Kameraführung kann die Ansicht wie folgt gesteuert werden:

W, A, S, D / Pfeiltasten: vor/zurück/links/rechts

Q/E: hoch/runter

Shift: schnelle Bewegung

Rechte Maustaste: Blinkwinkel ändern

Mausrad: Zoom in/out

| | | |
|---|---|---|
| 1 | Import Data | Öffnet einen Dateiauswahl-Dialog. Um einen Datensatz zu öffnen, welcher mit dem unter 2.1 erläuterten Matlab-Script umgewandelt wurde, muss lediglich die data.xml – Metadatei im Basisverzeichnis des jeweiligen Datensatzes ausgewählt und geöffnet werden. |
| 2 | Data Type | Aktiviert ein Dropdown-Menü mit den im Datensatz verfügbaren Attributen. Mit einem Mausklick auf das jeweilige Kontrollkästchen kann zwischen den Attributen gewechselt werden. |
| 3 | Rendering Mode | Ermöglicht den Wechsel zwischen der kartesischen und sphärischen Darstellung (Kartenansicht und virtueller Globus). |
| 4 | Toggle Timeline | Aktiviert/Deaktiviert das Timeline-UI (8-10) |
| 5 | Toggle Transfer Function | Aktiviert/Deaktiviert die UI-Elemente zur Anpassung der Transferfunktion (6-7) |
| 6 | Histogramm/ Kontrollpunkte | Ermöglicht die Anpassung der Transferfunktion (Farbe/Transparenz) über Kontrollpunkte und zeigt das Histogramm der angezeigten Serie. Bestehende Kontrollpunkte können mit gehaltener linker Maustaste verschoben werden.<br>Mit der rechten Maustaste können Kontrollpunkte angelegt oder gelöscht werden. |
| 7 | Color Picker | Ermöglicht Anpassungen eines Kontrollpunktes |
| 8 | Timeline Slider | Zeigt die relative Position der aktuell gezeigten Serie und ermöglicht das schnelle Scrollen durch alle verfügbaren Serien. |
| 9 | Timeline Controls | Ermöglicht die Kontrolle über das Abspielen des Datensatzes durch folgende Funktionen:<br>Zum Anfang / eine Serie zurück /Abspielen / eine Serie vor / Zum Ende |
| 10 | Toggle Repeat | Aktiviert/Deaktiviert die automatische Wiederholung |
| 11 | Camera Mode | Wechsel zwischen freier und gebundener Kameraführung |

Tabelle 1: UI-Elemente und ihre Funktionen

## 2.3 Advanced: Tweaks und Properties

Die im Folgenden erläuterten Tweaks und Eigenschaften sind lediglich für die weitere Entwicklung der Anwendung gedacht und nur innerhalb des Unity-Editors zu ändern.

Shader Properties:

Die unter Abb. 3 gezeigten Eigenschaften der beiden verwendeten Shader für die kartesische und die sphärische Darstellung ermöglichen in erster Linie die Veränderung diverser Grenzwerte. Diese können vornehmlich zu Testzwecken eingesetzt werden, sollten allerdings für den Live-Betrieb nicht verändert werden. Ausgenommen hiervon sind die Eigenschaften zur Intensitäts-Normalisierung und der Anzahl der Schritte durch das Volumen. Diese beiden Eigenschaften haben einen massiven Einfluss auf die Performanz und das visuelle Ergebnis und können für individuelle Builds an bestehende Hardware-Restriktionen oder Wünsche hinsichtlich der Optik angepasst werden.

Im Editor sind die Properties über das Volume_Renderer Asset zu erreichen.

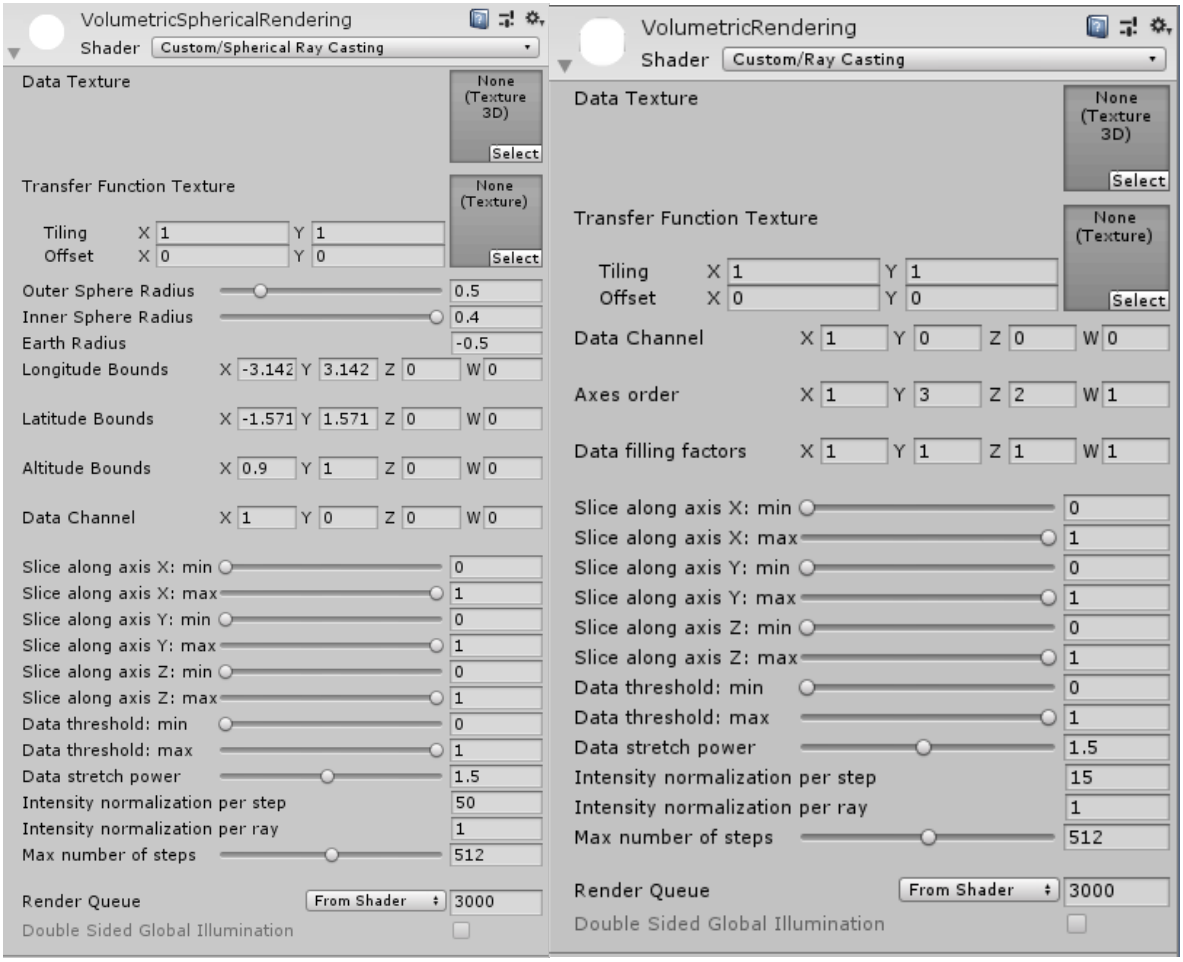

Abb. 3: Shader-properties

Über die Shader-Tweaks hinaus lassen sich die Kamera-Geschwindigkeit und die FPS der Timeline-Wiedergabe über die entsprechenden Assets anpassen.

# 3. Projektaufbau / -Bestandteile

## 3.1 Szene

Als Kernelement der Anwendung beinhaltet die Szene in Unity sämtliche Assets sowie deren Verknüpfungen mit Shadern, Scripts, Materials etc. Diese Assets sind die Kamera, der Volume Renderer, das User Interface und sämtliche Umgebungsobjekte wie bspw. die Groundplane zur Kartenansicht und der virtuelle Globus.

Eine Übersicht zu den einzelnen Assets und ihrer Hierarchie der Szene zeigt Abbildung 4.

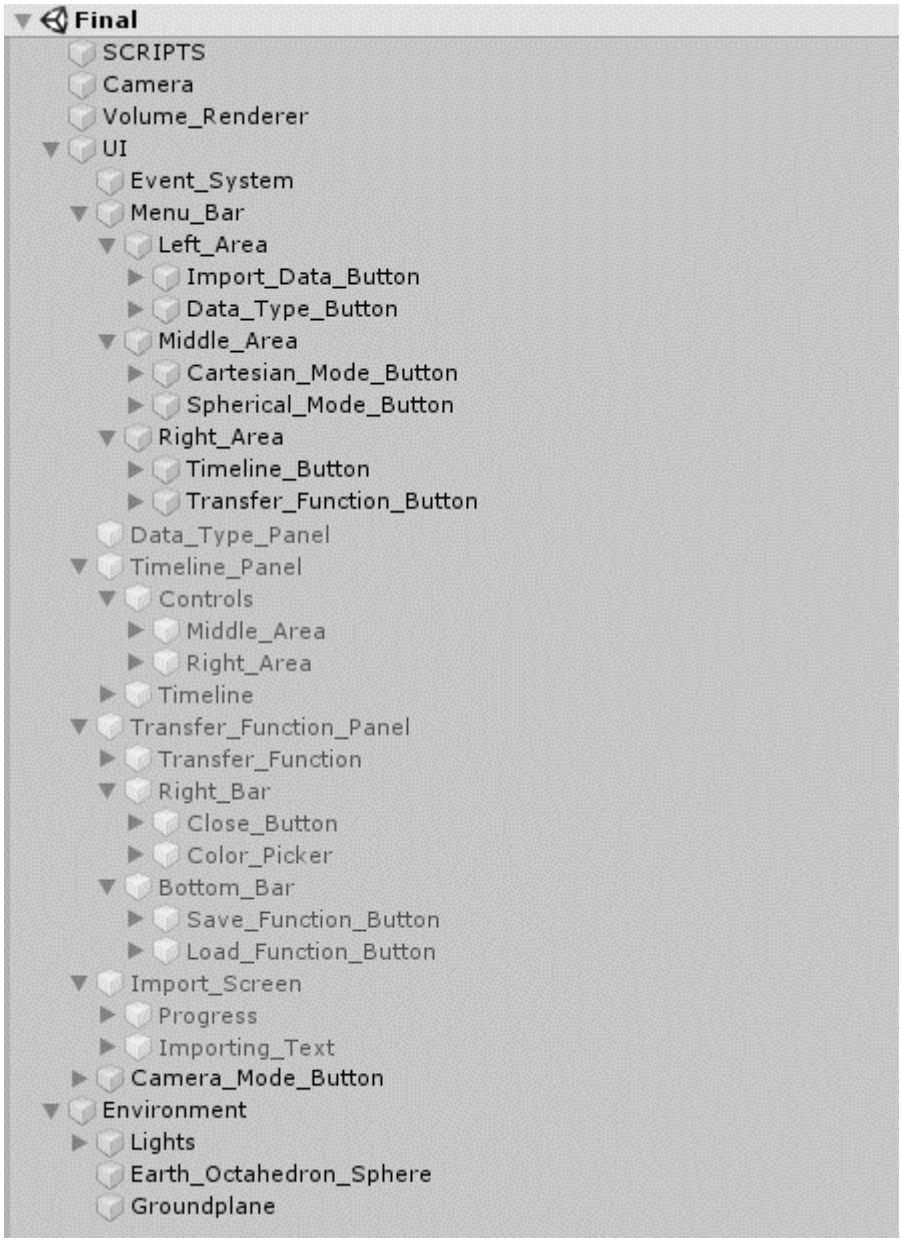


Abb. 4: Scene Assets und Hierachie

## 3.2 Plugins

Die im Rahmen der Anwendung benötigten Plugins belaufen sich nebst dem mittlerweile zum Standard gewordenen TextMeshPro und der nativen Unity-UI-Extensions auf das Plugin StandaloneFileBrowser zum Aufrufen eines Filebrowser-Dialogs beim Importieren eines Datensatzes sowie auf das Plugin LibTiff.Net. Letzteres ermöglicht das Laden und Einbinden von TIF-Bildern zur Laufzeit und ist somit unabdingbar für die Funktionalität der Anwendung.

## 3.3 Shader

Für die Funktion der Anwendung werden drei Shader genutzt. Der HistogramShader dient für die Erzeugung und Darstellung der 2D-Textur für das Transferfunktion-Histogramm, der RayCaster beinhaltet das Volumenrendering für den kartesischen Darstellungsmodus und der SphericalRayCaster dient der sphärischen Darstellung eines Datensatzes.

## 3.4 Scripts

Die für die Anwendung entwickelten Scripts sind in den Kategorien Datenimport, Transferfunktion, User Interface sowie Kamerasteuerung und Rendering aufgeteilt. Eine Auflistung der Scripts nach Kategorie und Funktionalität erfolgt unter Tabelle 2:

| Kategorie | Name | Funktion |
|---|---|---|
| Datenimport | DataAsset | Definition DataAsset-Klasse |
| Datenimport | DataAssetBuilder | Erzeugt und füllt DataAsset-Objekte |
| Datenimport | DataLoaderFromTIFFs | Lädt TIF-Serien |
| Datenimport | DataManager | Kern des Ladeprozesses, verbindet und nutzt die übrigen Scripts zum Datenimport |
| Datenimport | IDataAssetBuilder | Interface für den DataAssetBuilder |
| Datenimport | IDataLoader | Interface für DataLoaderFromTIFFs |
| Datenimport | IMetaData | Interface für Metadaten |
| Datenimport | IMetaDataReader | Interface für Metadaten-Reader |
| Datenimport | MetaDataReader | Liest XML-Metadaten |
| Transferfunktion | Transferfunction | Definition Transferfunction-Klasse |
| Transferfunktion | Controlpoints | Definition Controlpoint-Klasse |
| User Interface | DataTypeUI | Panel zur Auswahl des dargestellten Attributs |
| User Interface | TimelineUI | Panel zur Darstellung der Timeline-Elemente |
| User Interface | DataImportUI | Import-Button und FileBrowser-Dialog |
| User Interface | ModeUI | Wechsel des Rendering-Modus |
| User Interface | TogglePanel | Aktiviert/Deaktiviert die verschiedenen UI-Panels |
| User Interface | Transferfunction-ControlPointUI | Anpassung, Erzeugung und Löschung der Kontrollpunkte im Transferfunktion-Panel |
| User Interface | TransferfunctionUI | Darstellung Transferfunktion-Panel |
| Kamera | CameraMode | Wechsel zwischen freier und gebundener Kamera |
| Kamera | CameraFree | Freie Kamerasteuerung |
| Kamera | CameraOrbit | Gebundene Orbit-Kamerasteuerung |
| Rendering | VolumeRenderer | Verbindet den Volume Renderer mit weiteren Scripts (Rendering-Mode, TF, Dataset) |
| Rendering | VolumeRendererMode | Liste der verfügbaren Rendering-Modi |

Tabelle 2: Scripts nach Kategorie und Funktion

## 3.5 Sprites / Materials / Fonts

Die Verzeichnisse Sprites, Materials und Fonts beinhalten dem Namen entsprechend Texturen und Schriftarten, welche im Projekt genutzt werden. Hierzu zählen die Materialien für das Histogramm und die von den Raycastern durchlaufenen Volumen sowie das Material für die kartesische Groundplane und den virtuellen Globus, welches auf einer 2D-Darstellung der Erdoberfläche basiert. Darüber hinaus findet FontAwesome für die Einbindung von passenden Icons in die jeweiligen UI-Elemente Verwendung.

## 3.6 Octahedron-Sphere

Für die korrekte Darstellung der Erdkugel im sphärischen Rendering-Modus konnte keine Unity-Sphere genutzt werden, da beim Mapping der 2D-Textur auf ein solches Mesh Verzerrungen und Seams an den Polen und entlang einzelner Längengrade auftreten.

Anstelle eines neuen Mapping-Algorithmus wird für die Darstellung des virtuellen Globus ein auf einem Octahedron basierendes Kugel-Mesh erzeugt. Dieses verfügt einerseits über eine bessere Symmetrie entlang der Pole und anderseits durch insgesamt sechs Subdivisions über eine wesentlich höhere Anzahl Polygone. Beide Aspekte ermöglichen eine wesentlich bessere und realistischere Darstellung des virtuellen Globus im Vergleich zur Unity-Sphere.

## 3.7 UI Prefab-Assets

Hier finden sich die Prefabs der Objekte, die per Script für das UI erst zur Laufzeit dynamisch erzeugt werden. Dazu gehört zum einen ein Kontrollpunkt für das Interface der Transferfunktion. Zum anderen ist auch eine Checkbox enthalten, die für jedes Datenattribut erzeugt wird und so als Auswahl zur Verfügung steht.

# 4. Architektur

## 4.1 Datenimport

Die Architektur des Datenimports unterteilt vor allem drei Szenarien, die entsprechend von unterschiedlichen Klassen implementiert werden:

1. Import der Metadaten
2. Laden der einzelnen TIFF-Bilder und Umwandlung in Byte Arrays
3. Umwandlung der Byte Arrays in lineares Color Array zum Erzeugen der 3D Textur

Der DataManager kontrolliert den kompletten Lade- und Umwandlungsvorgang. Dieser wird entsprechend über das Interface angesteuert. Ebenfalls ist der DataManager auch gleichzeitig die Speicherverwaltung aller aktuell geladenen Assets, die nach ihrem Attribut entsprechend abgelegt sind.

Der eigentliche Import beginnt mit dem Laden der Metadaten im XML Format. Dabei wird einfach auf die in C# standardmäßigen Funktionen zum Deserialisieren einer XML-Datei zurückgegriffen. Die geladenen Metadaten enthalten unter anderem Informationen über die Bittiefe, Höhe und Breite aller TIFF-Bilder und auch die Anzahl der unterschiedlichen Höhenlevel und damit die Anzahl der TIFF-Bilder selbst. Ebenso enthalten sind ist eine Liste aller Attribute angegeben über ihren Namen.

Da nun die für den Import der Daten wichtigen Informationen vorhanden sind, kann mit dem Import der TIFF-Bilder begonnen werden. Um vor allem Speicherplatz zu sparen, allokieren die Klassen im Voraus die Byte oder Color Buffer, die sie später benötigen. Die unterschiedlichen TIFF-Bilder in einem gegebenen Zeitpunkt werden nacheinander mit Hilfe der LibTiff.Net Bibliothek geladen und in ein zweidimensionales Byte Array konvertiert. Dieses wird dann direkt weitergegeben, um aus diesem Array ein sogenanntes DataAsset zu erstellen. Diese beinhaltet vor allem die 3D Textur, die zunächst durch die Umwandlung des Byte Arrays in ein Color Array erzeugt wird. Ebenso wird zugleich auch die für die Daten passende 2D Histogramm-Textur erzeugt.

Dieser gesamte Prozess wiederholt sich für jedes Attribut, welches über die Metadaten angegeben wurde und für jeden Zeitpunkt darin. Jeder Zeitpunkt wird also einzeln verarbeitet. Dabei wird davon ausgegangen, dass die TIFF-Bilder in der im Kapitel 2.1 angegebenen Ordnerstruktur abgelegt sind.

Um eventuelle zukünftige Änderungen am Datenimport zu vereinfachen, werden von allen relevanten Klassen Interfaces implementiert, die die grundlegenden Funktionen definieren. In der aktuellen Fassung stellen diese auch schon Funktionen zum Laden von 16-Bit Daten bzw. TIFFs zur Verfügung. Deren Implementierung in den jeweiligen Klassen steht jedoch noch aus.

## 4.2 Schnittstellen

Die Logik der Anwendung wird hauptsächlich über das Interface gesteuert. Dazu zählt zum einen das Starten eines Imports aber auch etwa das Wechseln eines Attributs. Die dazu gehörige Datenhaltung übernimmt der DataManager. Dieser teilt anderen Skripten bzw. Systemen über Events entsprechende Änderungen mit. Das kann zum einen ein fertiger Import sein oder die Tatsache, dass das zum Darstellen ausgewählte Asset geändert wurde. Diese Änderung des Assets wird vor allem durch das Interface der Timeline verursacht, die entsprechend über alle verfügbaren Assets läuft und zwischen ihnen wechselt.

Das VolumeRenderer Skript dient für andere vor allem als Schnittstelle zur Änderung für jegliche spezifischen Änderungen an den Material Eigenschaften des Shaders. Dazu zählt das Umschalten zwischen dem kartesischen und sphärischen Modus, sowie das Setzen des aktuellen DataAssets bzw. der darin enthaltenen 3D Textur und das Setzen der Textur der Transferunktion.

Das Skript für das Interface der Transferfunktion knüpft an die Events des DataManagers ebenfalls an, um über die Änderung eines Assets informiert zu werden. Dadurch kann das Histogramm ordnungsgemäß aktualisiert werden. Ebenso wird erwartungsgemäß hier das Generieren der Transferunktion übernommen. Die dafür notwendige Textur wird bei jeglicher Änderung neu erzeugt. Als Änderung zählt das Anpassen der Farbe, die Positionsänderung eines Kontrollpunktes und das Löschen oder Erzeugen eines Kontrollpunktes. Für die Generierung der Textur wird vollständig auf bereits implementierte Logik aus den Open Source Projekten zurückgegriffen.

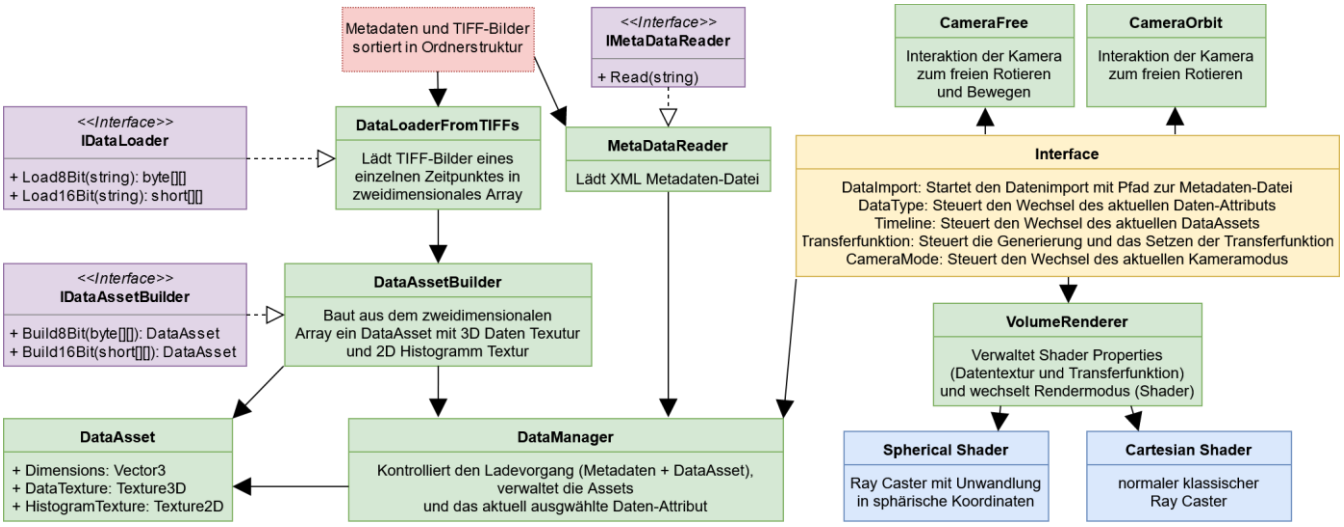

Abb. 5: Architektur-Diagramm

# 5. Future Plans

## 5.1 Automatisierung des Datenimports

In der aktuellen Fassung der Anwendung erfolgt der Download und die Umwandlung der Daten aus der Datenbank der Copernicus-Mission über externe Scripts. Dieser Schritt kann in die Anwendung integriert werden, indem die Ausführung des Python- und Matlab-Scripts mit dem Import-Button verknüpft und ein entsprechendes UI-Element zur Auswahl der gewünschten Attribute, Zeiträume etc. hinzufügt.

## 5.2 Variable TIF-Größe (>=16Bit)

Im Rahmen der Weiteren Entwicklung muss die Anwendung um die Möglichkeit, TIF-Serien von größerer Bit-Tiefe (>= 16 Bit) zu importieren, erweitert werden. Hierbei könnte es zu Problemen hinsichtlich der Performanz bekommen, sodass im Rahmen des Ladevorgangs möglicherweise eine passende Datenstruktur angelegt werden muss.

## 5.3 Speichern/Laden von Daten und Transferfunktionen

Um bei wiederholtem Arbeiten insbesondere mit großen Datensätzen lange Ladezeiten und den Konfigurationsaufwand hinsichtlich der Transferfunktion zu vermeiden, sollte eine Speicherfunktion für Kontrollpunkte sowie die verwendeten 3D-Texturen entwickelt werden. Da 3D-Texturen nicht als solche gespeichert werden können, wäre womöglich eine bitweise Darstellung aller Voxel der Textur eine passende Lösung zur Reduktion der Ladezeit.

## 5.4 Region of interest (ROI)-Darstellung mithilfe von Bounding-Volumes

Für die Darstellung von meteorologischen Phänomenen wie dem Polarwirbel-Split, El Niño oder Wirbelstürmen besonderen Ausmaßes könnte eine ROI-Darstellung von großem Nutzen sein. Nach Liang et al. (2014) können hierfür spezielle Bounding-Volumes genutzt werden. Diese können erzeugt werden, indem vier Basispunkte auf der Erdoberfläche ausgewählt werden. Diese werden mithilfe sphärischer Interpolation fester Auflösung miteinander verbunden und die Zwischenschritte in Form von Punkten gespeichert. Hierbei entsteht ein gekrümmtes Rechteck, dessen Kanten in mehrere Segmente unterteilt sind.

Anschließend werden entweder horizontal oder vertikal die einander gegenüberliegenden Punkte zweier Kanten ebenfalls via sphärischer Interpolation verbunden und die Koordinaten der Zwischenschritte gespeichert. Das Ergebnis der zweiten Interpolations-Runde ist ein gekrümmtes, auf der Kugeloberfläche aufliegendes Punktgitter, welches die untere Schicht des Bounding-Volumes bildet. Für die Koordinaten der oberen Schicht müssen sämtliche Punkte des Gitters anhand der Normalen der Kugel um eine festgelegte Länge nach außen verschoben und separat gespeichert werden.

Aus den beiden Punktgittern lassen sich nun die Dreiecke und Normalen des Bounding-Volumes und somit das entsprechende Mesh in Form eines gekrümmten Quaders erzeugen. Nach Erzeugung des Bounding-Volumes erfolgt die Darstellung wie von Liang et al. (2014) beschrieben mithilfe zweier Texturbuffer, in welchen die Ein- und Austrittspunkte der Strahlen durch jedes Bildschirmpixel gespeichert werden. Die Strecken zwischen den korrespondieren Werten beider Buffer werden gesampled und die Density-Werte aus dem Volumendatensatz mithilfe einer Transferfunktion in Farbwerte konvertiert.

# 6. Referenzen / Quellen

LIANG, Jianming, et al. Visualizing 3D atmospheric data with spherical volume texture on virtual globes. *Computers & geosciences*, 2014, 68. Jg., S. 81-91.

LAVIK, Matias. UnityVolumeRendering, GitHub (2019), unter: https://github.com/mlavik1/UnityVolumeRendering

FERRAND, Gilles. Unity-RayTracing, GitHub (2017), unter: https://github.com/gillesferrand/Unity-RayTracing

FLICK, Jasper. Creating an Octahedron Sphere in Unity, binPress (2014), unter: https://www.binpress.com/creating-octahedron-sphere-unity/

Plugins

https://bitbucket.org/UnityUIExtensions/unity-ui-extensions/src/master/

https://github.com/BitMiracle/libtiff.net

https://github.com/gkngkc/UnityStandaloneFileBrowser