User Guide for DataExplorer3D

Ingo Kirchner

September 30, 2024

Contents

..... Good luck and have fun (ik).

1 Elements of Climate Modelling and the Scope of DEX3D

In the world of climate and weather modelling a huge amount of raw model output is produced. The visualization of these outputs is a complex task and not easy, because each model produces it's own specific data. Few standards for the data formats exists, see GRIB and NetCDF, but also for the dictionary of the metadata, see CF conventions. Modern models, like ICON, store the data in NetCDF and follow the guidelines of common standards. This helps in understanding the meaning of all the different quantities in the model output. An other important issue is the data grid. The computational grid in ICON is a triangular grid. This unstructured grid can not visualize in the common world space coordinates with any plotting tool. Normally the model output passes through a postprocessing pipeline. At the end the data can be converted into pictures, collection of pictures or an animation. Addional steps are needed to get the model output quantities fully animated in a natural projection on the globe.

For the purpose of climate modelling the 3D-DataExplorer was designed, developed and implemented. The tool should fullfill the following requirements:

browser based ... independent from operation system, ready to use for data inspection without installation tasks

connect different data sources ... easy access to huge data storage, e.g. model output stored at HPC

usable for teaching ... no addional postprocessing step needed for visualization

natural representation of data ... view the data in our 3D-world

open for extentions ... written as OpenSource, modular using concepts of Python

well documented ... documentation is part of the application

Based on these terms the project the software technology was chosen

- visualize data with WebGL
- use only Python and JavaScript for coding
- combine potential of big servers for data processing with the flexibility of clients for visualization tasks in the 3D-studio, client-server-concept
- concentrate on NetCDF as input data format
- dynamize the client part with ajax

The whole system was developed mostly without using a framework. Frameworks can give huge pool of functionality, but it's an additional software layer between the raw software language, other standards and the application purpose. Therefore frameworks have also weak points. The supported features of frameworks is outside the control of the developer of an application. Many application requirements can not be realized or only be realized with constraints. The creativity during the development process is limited. Features are not realizable because of the limitations of such frameworks. That's why DEX3D is mostly raw coded in Python and Javascript. You will help to improve the application? Only knowledge of these standards is neccessary and you have to accept the coding guidelines of the DEX3D project. With your ideas it is possible to improve the usability of DEX3D.

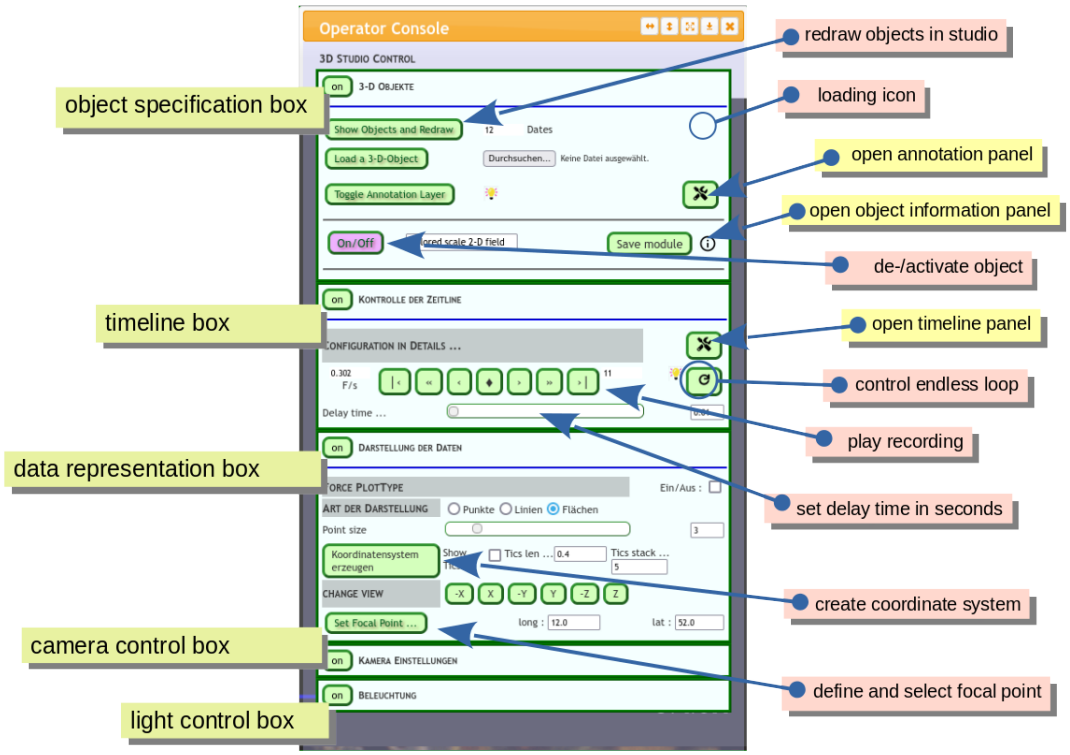

Figure 1: The most important functions in the 3D studio control panel

In the next chapter a short introduction of the functionality and the different components of DEX3D are described. In the third chapter the reader learns how the combination of all gui elements work together to organize the whole workflow for the data visualization in the WebGUI. In the last chapter the addons of the project will be proposed and few show cases illustrate the potential of DEX3D.

2 Objects and the 3D Studio

These chapter will guide the user to the most common tools and panels which are implemented in the WebGUI of the DEX3D. Normaly the entry point is the server url, e.g. https://dex3d.klimod.de. Read the start-up message or skip it next time with the button. The start-up message will gone after few seconds and the view into the 3D world is open. The WebGL canvas covers the complete browser window. Best will be, to use the whole screen for your browser window. The WebGL canvas is overlayed with the annotations. The WebGL canvas is your viewport into the 3D-studio. The annotation layer is prepared as a transparent 2D-picture. You can switch it off later on.

In the upper right corner the button box is visible and contains the doors to the three major gui-panels:

documentation ... all **documentation** is colleced here, except the developer guide.

3D modules ... with the **3D object designer** new objects can be created or existing objects can be modified or removed.

workbench ... the **3D studio control** holds all elements for the configuration of the

3D studio (see Fig.1):

- specify the visability or properties of all 3D objects **object specification box**
- configure the timeline **timeline box**
- modify the representation of the objects, add the axis of the coordinate system for orientation **data representation box**
- control the camera **camera control box**
- configure the lighting in the 3D studio **light control box**

Each of the master buttons will open an independet new panel on the WebGUI. As alternative option keyboard keys can be used to open the panels. Which key-codes are active descibed in the documentation.

The 3D studio is open but empty. The next step will be to setup a scene with 3D objects. In the 3D object designer two types of objects can be created. The simple objects are 3D body like a sphere or a cube. More complex objects use data, which will be provided by the server. As beginner you should play with the land-sea-mask or the topography object. Start with a low resolution and create a globe with the topography. Or you can start with simple objects and play with the mouse controls inside the 3D studio.

For the data projection one parameter controls the size of the globe, the **scalefactor of globeradius** is used for this. You can work with a unit of [meter]. That you set this to the real Earth radius (approx. 6370000 m). But the position of the camera has to be changed. The default position is at the coordinate (0,0,-100). All coordinates are relative and free of interpretation. The creation of new objects can take few seconds or longer, more complex longer load time. In the **studio control** there is a signal in the upper right corner of the **object specification box**. The box should be open to see the loading signal. You will see a new configuration box for your new 3D object in that box. Here you can switch the object on or off, reconfigure it or inspect all logmessages from the lifetime of the object, see the small info icon. Press **show objects and redraw** and all active objects will be visible in the 3D studio.

3 Explore Model Output

In a second try you open a NetCDF file and see, if the data can be visualized as scalar fields. Use the **colored scale 2-D field** object and create a new one. Now you have to connect to a NetCDF file, simple press the button **Dateiauswahl starten** and the **file panel** will be opened. Type in a folder-name with your data. The DEX3D server at the Institute of Meteorology in Berlin will data from your account. Start with your homefolder and press **show location**. Than you can navigate trough the subfolder or select a NetCDF file object. In the line after the navigation box your selection is placed and in few seconds you will see the inventory of the single file. Normaly the data from a model experiment (e.g. ICON) the data are spread over a group of files. Therfore you modify in the file selection entry the name of the file. Change parts with placeholders, like '*' or '?'. Which placeholders allowed you can read in the manpage of the bash the chapter **EXPANSION**. The pattern will be resolved using **pathname expansion**. Not all folders are accesible, you can try with **/daten/erafive/arch/data/dkrz-sync/ERA5/**. You can inspect the inventory processing process in the box **Informationen zum Datenobject**. Finaly you transfer the file configuration into the 3D object configuration. The file browser is closed

and you can continue with the configuration of your object. Select a quantity and specify optional more attributes. At the end you **load 3-D object** and wait until the loading is finished. For ERA5 example the loading takes a relative lobg time, approx.15 seconds for the Europe subsection .For an first test it is recommended to select only a subregion, e.g. Europe [-20°W,30°E,40°N,60°N] and use near surface temperatur (tas) monthly means. Redraw the object and place your camera to a position over Europe using the geographic coordinates from your reference point (e.g. the middle of your selected region). Simply **set focal point** in the **data representation box**. In addition you can change the **ocular** with the slicer in the **camera control box**. A value between 6 and 7 will expand the view best.

Now you can play with annotations. Use the tool icon in the **object specification box** and open the **annotation panel**. The following annotations are possible:

title row ... a short title line

frame counter ... show the position in the animation time line

vizualize timeline ... show a progress bar

visualize date/time ... show the real date of the visualized data frame

colorbars ... of each data object, the configuration is possible using the functionality of the **object specification box**

As minimum you should active the **progress bar** and **real date**. It will help to interpret which data frame is shown. We will make next an animation of our data. Start the **timeline panel** with the tool icon in the **timeline box**. In the **data timeline** container you see the start and stop date of your data. if the timeline contains more than one rate, you can try the recorder functions, see in the **timeline box**. The delay time should set to maximum and than press next frame. The next dataset is loaded. It's a little fast than the first step, because the mesh of our data not changed, only colors will be loaded from the server. If successful you can try loop and load all data frames of your dataset. Dependent on the number it will take a while. You will see the state in the progress bar. After the first loop you can set the delay time to minimum, set endless looping and play again. The picutes will be processed now from the memory. On a common client system the framerate can got approx. 15 frames pre second or more. The framerate will be monitored left from the recorder control. You can stop the animation with the middle button of the recorder control. If the framerate is too high you can make the animation slower by increasing the delay time. It's given in seconds.

4 Advanced Features

The DEX3D project provides special tools, which extend the functionality of the WebGUI, speed up the data pipeline and opens more data sources. The tools work outside the WebGUI. In addition in the project folder **.../scripts/** bash scripts are placed, which can be used as templates.

MakeIndex.py ... normaly the data inventory is produced on the fly, but for large data groups, each file has to be inspected. The indexer can store the inventory of each file in a same namen JSON file. It will be placed at the same location. The file browser will use such index files and the file selection will be faster in the file broswer of the WebGUI.

ProxyServer.py ... the master server will only provide data, which are accessible on the webserver. Often the files are located in the user space, e.g. on the personal computer. The ProxyServer can be started in user space and opens a data channel.

After a ICON experiment with the VAST-Environment the raw data placed in subfolder like **.../ANALYSIS/NNN/**. First you should run the indexer over all these output files. It makes no difference, if the data grid is lat-long or icosahedric, both can be used. The expeiment folder should be this *EXPDIR* and the DEX3D installdir is this *DEX3D=/daten/vast/work/vast/dex3d/wetomeda*. You run the following commands e.g. at *calc01.met.fu-berlin.de*. As result each NetCDF file have a JSON double. As example the scanner needs 5 minutes to produce indexfiles from 3000 data files of a 100 day experiment (India region 5 nests) with total of 9 TB of raw ICON data.

```
cd $EXPDIR
$DEX3D/tools/MakeIndex.py -i 'ANALYSIS/???/out*.nc'
```

For the visualization two computers are part of the game. The *client* is your personal/local one, which is running the webbrowser and ssh. The *server* is located at the Institute of Meteorology or on the HPC @dkrz, which holds your model data. On the *server* you will start the **ProxyServer** and the tunnel will connect your *client* with the ProxyServer port at the *server*.

```
$DEX3D/tools/ProxyServer.py
```

The ProxyServer will start with the next free port. Next you have to prepare on your client the ssh tunnel. You have to remember the proxyport *PROXY_PORT* and login with your username @IfM *IFM_USER*. For a Linux client the command looks like this ...

```
ssh -L 8000:localhost:PROXY_PORT IFM_USER@calc01.met.fu-berlin.de
```

If both is running, the ProxyServer and the ssh-tunnel, it's time to open the DEX3D WebGUI `https://dex3d.klimod.de` and in an additional browser window the ProxyServer `http://localhost:8000`. As first try you should check on the ProxyServer window the access to your experiment data. If successful you can use the proxy tunnel in the file browser of the DEX3D window and load your experiment data into the WebGUI. The host is all the time **localhost** but the portnumber depends on the way of connection to the proxy. Running with ssh-tunnel, as in the example, you have to use your local port number 8000.

The scenario for in-house computers, in-house means Institut for Meteorology, is much simpler. You start the ProxyServer on your in-house computer and use the ProxyServer port number in the DEX3D WebGUI. All data, which readable on your in-house computer can than be used in the WebGUI.

5 Install your own DEX3D

The DEX3D uses the http as communication protokoll. Normaly the entrypoint is provided by the central DEX3D server, see (https://dex3d.klimod.de). Using this central server, you have only access to data which are available on that server. Normaly the data are placed at a HPC host and your browser can not access to the HPC directly.

5.1 How can you combine the DEX3D with your data on HPC?

The solution is the **ProxyServer**, which is part of the DEX3D distribution. The ProxyServer is running in userspace and provides the full flexibility to access to your model outputs. First of all, you have to setup your own git clone on the HPC. You should start with an official release, later on you can also try the latest snapshot from the develoment line. The following command creates a new clone of the release branch in the folder *YOUR_DEX3D*.

```
git clone –branch release \
      https://gitlab.met.fu-berlin.de/climod/wetomeda.git YOUR_DEX3D
```

The export is open and you can use it as it is. If you are in the development-team of the project, you can help to improve the project. Now you can use the ProxyServer and play with it. First try can be ...

```
YOUR_DEX3D/tools/ProxyServer.py
```

... starts the ProxyServer on the next free port. Sometimes you have to install missing python packages, may be in a virtual environment. Now you can communicate with the ProxyPort. Simply try to load the page **http://localhost:PROXY_PORT/** into a browser, which runs on your host. If the host is only reachable with ssh from your personal client, than you have to build up a tunnel to your production host, e.g. the HPC and the running ProxyServer.

```
ssh -L LOCAL_PORT:localhost:PROXY_PORT ACCOUNT@HPC_HOST
```

In the command the parameters are

LOCAL_PORT ... the port on your client, which is used in the browser

PROXY_PORT ... the port from the ProxyServer, you see it in the terminal, where the ProxyServer was started

ACCOUNT ... your personal account at HPC

HPC_HOST ... the name of your login node on the HPC. It depends on the HPC, what you can access from the outside.

As next step, you can try to run the testcase. For this scenario your ProxyServer should be reached via your local port 10000. Without tunnel this is the ProxyServer port, but

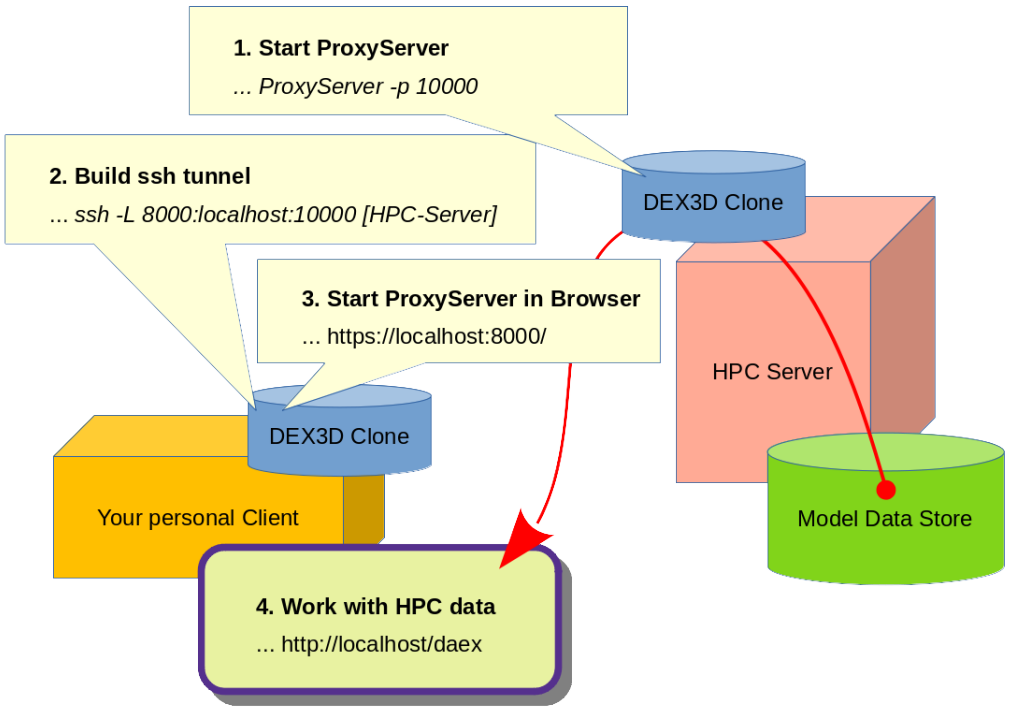

Figure 2: Connect your DEX3D with your HPC data

with tunnel it depends on the ssh-tunnel-command. You will need also a clone of the DEX3D project on your client machine, e.g install it on *YOUR_LOCAL_DEX3D*.

```
YOUR_LOCAL_DEX3D/testing/start-test.sh help info
```

Now you know the options of the test wrapper and can run one test, e.g. number 0 or number 2 are recommended. The tests can not run now, you have to install the driver. The test wrapper needs a binary of the geckdriver (https://github.com/mozilla/geckodriver/releases). You have to install it and modify the path-to-driver in the test wrapper.

5.2 Relevant Python Packages

Most of the project is coded in Python-3. For your personal installation of DEX3D you needs in the python module stack these additional packages:

selenium ... used for testing

xarray ... used in NetCDFData

A complete module list will be shown using ...

```
cd YOUR_LOCAL_DEX3D
make show_modules
```