

We're ramping up our visualisation capability...



Visualisation services and tools for ACCESS models

Owen Kaluza owen.kaluza@anu.edu.au or @OwKal on ACCESS Hive

- In house visualisation specialist in the Model Evaluation and Diagnostics (MED) team.
- Complemented by domain specific and general visualisation expertise throughout the ACCESS community and NCI Vizlab
- MED software projects improving visualisation tools for ACCESS-NRI models

Initial goals

- Collect a library of tools and techniques - visualisation recipes
- Produce some showcase visualisation examples

Soon...

- Provide specialist advanced visualisation services (3D modelling/4D animations, cinematic visualisation)
- Improve visualisation tools and develop new ones

Advanced visualisation?

Illustrate aspects of the data that are difficult to translate to 2D and use our brain's capability to explore/analyse in 3D environments while increasing accessibility and impact to the wider public

- Interactivity (3D, adjusting parameters)
- Time varying : Animations (4D)
- Complex 3D models, fly throughs, higher dimensional data
- Advanced lighting, and rendering, volume rendering, ray tracing
- VR and other specialist vis hardware
- Cinematic Scientific Visualisation

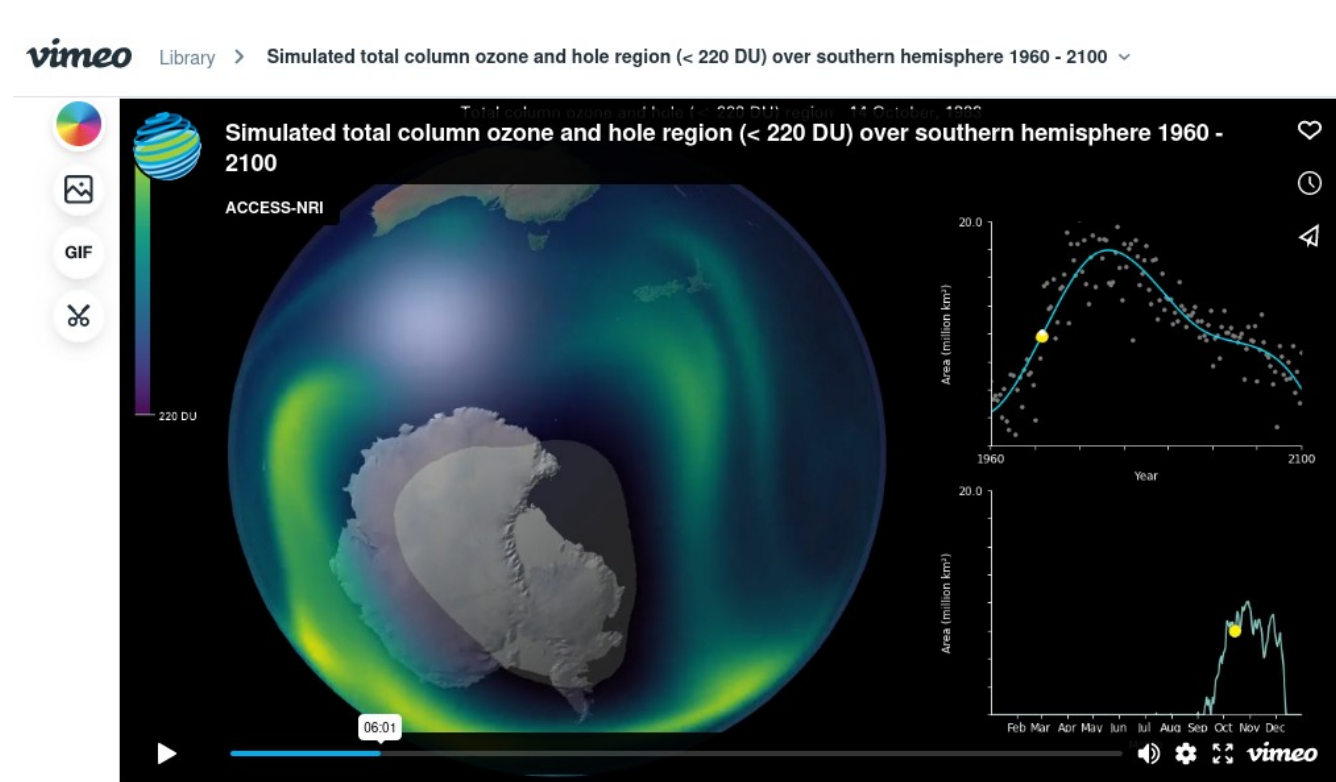
Cinematic scientific visualisation?

Taking visualisation to the next level
Media quality visual output, using filmmaking techniques and tools (cinematography, lighting, and composition) and visual effects software such as Blender and Houdini

Watch this space...

Check out our Vimeo channel for visualisation projects as we release them...

<https://vimeo.com/accessnri>



Simulated total column ozone and hole region (< 220 DU) over southern hemisphere 1960 - 2100

Total column ozone was simulated using the ACCESS-CM2-Chem[1] model as part of the Chemistry Climate Model Initiative 2022 (intercomparison project[2]). Data from this model run is available via the Centre for Environmental Data Analysis (CEDA) repository[3].

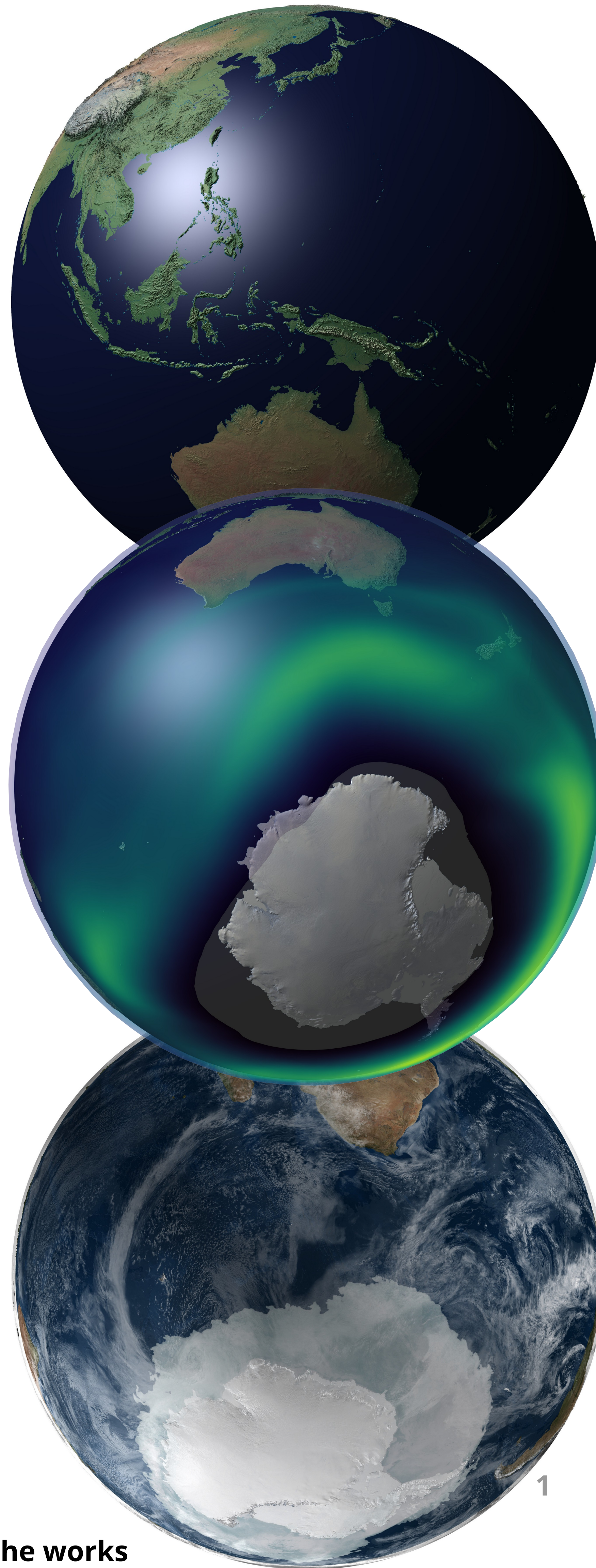
- <https://www.publish.csiro.au/ES202015>
- <https://blogs.reading.ac.uk/earthcom/2022/>
- <https://data.ceda.ac.uk/badc/cm/cm/data/post-cmp/cmcm/2022/CSIROACCESS-CM2>

Projects in the works

- *Ozone layer recovery* 3D vis of Ozone simulation 1960 - 2100
- *Sea ice* Recreating animations of sea ice in 4D
- *Aus400* Tackling a high res model with advanced visualisation techniques
- *Circulation of the Southern Ocean* Exploring ocean currents

Image credits

1. NASA/Goddard Space Flight Center Scientific Visualization Studio. The Blue Marble data is courtesy of Reto Stockli (NASA/GSFC)
 2. NASA SVS Perpetual Ocean <https://svs.gsfc.nasa.gov/3827>
- All others (c) ACCESS-NRI, Owen Kaluza 2023



Earth Model

Data sources:

Colour texture

- Blue marble: https://www.hackml.net/map/download/world_shaded_43k.jpg
- Shaded relief (public domain) <https://www.shadedrelief.com/natural3/pages/textures.html>

DEM topography:

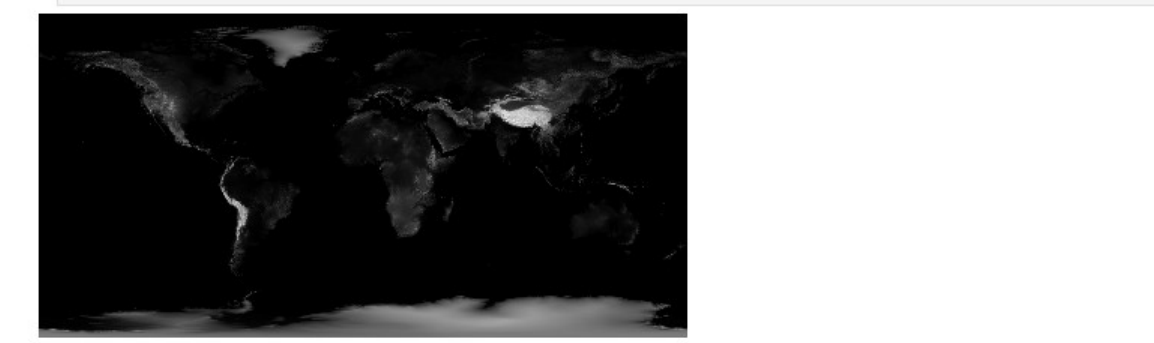
- <https://visibearth.nasa.gov/images/7904/topography>
- https://images.gis.nasa.gov/images/imagerecord/7300/7304/geco_08_rev_elev_2160x10800.png

```
30 [1] import accessvis, lavavu
import numpy as np
import math, os
import matplotlib.pyplot as plt
import matplotlib.image as img
from PIL import Image
Image.MAX_IMAGE_PIXELS = 1061683200

30 [2] Earth's radius * 0.7138 we'll use 2000's of km units (km)
radius = 6371 * 1e-3 #Modulus 10 km
#20e = 1e-6 #Conversion factor from metres

30 [3] height = data = np.array(Image.open('geco_08_rev_elev_2160x10800.png'))
print(height.shape, height.max(), height.min(), height.dtype)
(10800, 21600) 0 255 uint8

30 [4] #renders a jpeg downscaled view
height = height.reshape(height.shape[0], height.shape[1], 1)
image = lavavu.Image(data=height[100, :100])
image.display()
```



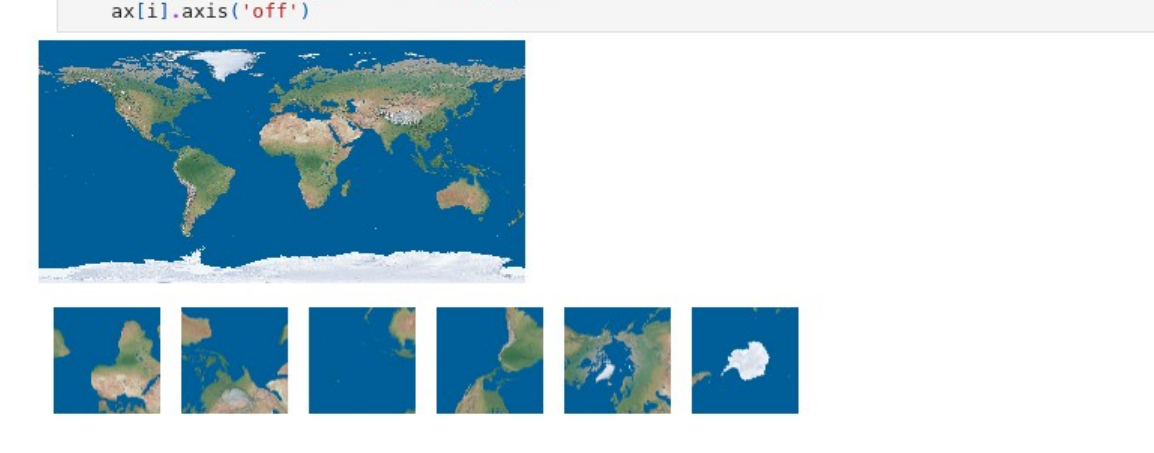
```
30 [5] #Normalise then exaggerate height
height = height / 255.0 / 10.
print(height.shape, height.max(), height.min())
(10800, 21600, 1) 0.0 0.1

30 [6] #Split the topography equiangular image into cube map files
GRIDRES = 2048 #Resolution of colour map / sea ice grid
heights = accessvis.split_text(height, GRIDRES)

30 [7] #Split the colour equiangular image into cube map files
fn = '3_no_ice_clouds_38k.jpg' #shaded relief
col = np.array(Image.open(fn))
#renders a downscaled view
image = lavavu.Image(data=col[100, :100])
image.display()
```

```
#Report individual textures
if not os.path.exists('t' + '.png'):
    #Resolution of the colour texture - defines the colour detail
    TEXRES = 4096
    textures = accessvis.split_text(col, TEXRES)
    #Write colour texture files
    for f in ['R', 'G', 'B', 'L', 'U', 'D']:
        fn = f + '.png'
        if not os.path.exists(fn):
            tex = lavavu.Image(data=textures[f])
            tex.save(fn)

#display images
matplotlib.rcParams['fig.figsize'] = (20, 20)
for i, f in enumerate(['R', 'G', 'B', 'L', 'U', 'D']):
    ax = plt.subplot(2, 3, i+1)
    ax.imshow(Image.open(f + '.png'))
    ax.axis('off')
```



Render 3D globe

```
30 [8] lv = lavavu.viewer(border=False, axis=False, resolution=(1920,1080), background='white')
#lv.generate_color_cube_grid
ii = np.linspace(1, 1, GRIDRES, dtype=float32)
jj = np.linspace(1, 1, GRIDRES, dtype=float32)
zz = np.linspace(0.0, 0.0, 1, dtype=float32) #3rd dim
for f in ['R', 'G', 'B', 'L', 'U', 'D']:
    #generate cube face grid
    if f == 'R':
        vertices = np.dstack(ii, jj, zz + 1.0)
    elif f == 'B':
        vertices = np.dstack(ii, jj, zz - 1.0)
    elif f == 'G':
        vertices = np.dstack(ii + 1.0, jj, zz)
    elif f == 'L':
        vertices = np.dstack(ii - 1.0, jj, zz)
    elif f == 'U':
        vertices = np.dstack(ii, zz + 1.0, jj)
    elif f == 'D':
        vertices = np.dstack(ii, zz - 1.0, jj)
    #normalize the vectors to form spherical patch (normalized cube)
    v = vertices / vertices.sum(axis=1, keepdims=True)
    norms = np.sqrt(v.dot(v), axis=1, keepdims=True)
    vnorms = v / norms
    verts = vnorms * radius #offset the heights and apply scaling
    q = lv.quads(vertices, verticesverts, textures + '.png',
                lightmap=False, flipnorm=[1, 1, 1, 1, 1, 1], wherea='facing',
                renderers='simpletriangles', opaque=True)

30 [9] lv['diffuse'] = 0.7
lv['ambient'] = 0.25
#Adjust specular highlights
lv['shininess'] = 0.06
lv['refractive'] = 0.25
lv['brightness'] = 0
lv['texturemap'] = 1
lv['light'] = [1, 1, 0.9, 1] #R,G,B colour, setting final component disables two-sided lighting
lv['contrast'] = 1.0
dist = 151800 #251.85 million km earth -> sun, in our units
c = [1, 25, 1, 25, 1]
g = math.sqrt(0.5*(1+math.cos(math.radians(c[0]*math.pi/180) + c[1]*math.pi/180 + c[2]*math.pi/180))
lv['lightspos'] = [p * c[0], c[1], c[2]] #Can set 4th component to 1 to enable fixed light pos
lv.display(400,400)

30 [10] lv.translation(0, 0, 0, -17)
lv.rotation(0, 0, 135, 0, 0)
lv.display(400,400)

# Create Stratosphere mesh to plot ozone
30 [11] #radius in units of 2000 km (diameter of earth: 6778 km)
r = radius * radius = 1.038 #radius = small offset for stratosphere
lv.addStep(0) #Add a timeslot or things don't work on load
lv['sphere'] = lv.sphere('radius', radius, segments=6, colours='grey', vertices=(0,0,0))
lv['rotate'] = [0, 90, 0] #This rotates the sphere to align with out (0,360) longitude texture
lv['texture'] = 'top.png' #show an initial texture or textures will not be generated
lv.render()
#renders sphere vertices, textures etc
lv.bake()
lv['sphere'] = False #must disable this for the ozone plot
lv['rotate'] = [0, 0, 0]
lv['display'] = 400,400)

# Converting LINES 0 TRIS 1 PTS 0
Converting LINES 0 TRIS 0 PTS 0
```



Create Stratosphere mesh to plot ozone

```
30 [11] #radius in units of 2000 km (diameter of earth: 6778 km)
r = radius * radius = 1.038 #radius = small offset for stratosphere
lv.addStep(0) #Add a timeslot or things don't work on load
lv['sphere'] = lv.sphere('radius', radius, segments=6, colours='grey', vertices=(0,0,0))
lv['rotate'] = [0, 90, 0] #This rotates the sphere to align with out (0,360) longitude texture
lv['texture'] = 'top.png' #show an initial texture or textures will not be generated
lv.render()
#renders sphere vertices, textures etc
lv.bake()
lv['sphere'] = False #must disable this for the ozone plot
lv['rotate'] = [0, 0, 0]
lv['display'] = 400,400)

# Converting LINES 0 TRIS 1 PTS 0
Converting LINES 0 TRIS 0 PTS 0
```

