Lab 3

Goal

- **Convert a tif file of Rwanda's elevation from 16 bit unsigned integers to**
  - A monochrome jpeg image
  - A 3D graphic object for viewing in Unity

Output JPEG Image

- **The python pillow library will read a tif file**
  - Input the tif file (it is in np.uint16 format)
  - Find the minimum value and record it (this it the minimum elevation in Rwanda in meters)
  - Next, process the image to map *all* the maximum values to 0
    - ✓ Use numpy.where() to do this efficiently in one line of code
    - ✓ The maximum values are all points *outside* Rwanda!
  - Next, find the *new* maximum value after the above remapping: this is the true maximum elevation in Rwanda
    - ✓ Report all these values in your lab report
  - Finally, linearly rescale the elevation data so that 0 values become pixel intensity 0 and the maximum elevation value becomes pixel intensity 255; output a monochrome jpeg image
    - ✓ Remember that you must convert to np.uint8 after this rescaling before you can output the image

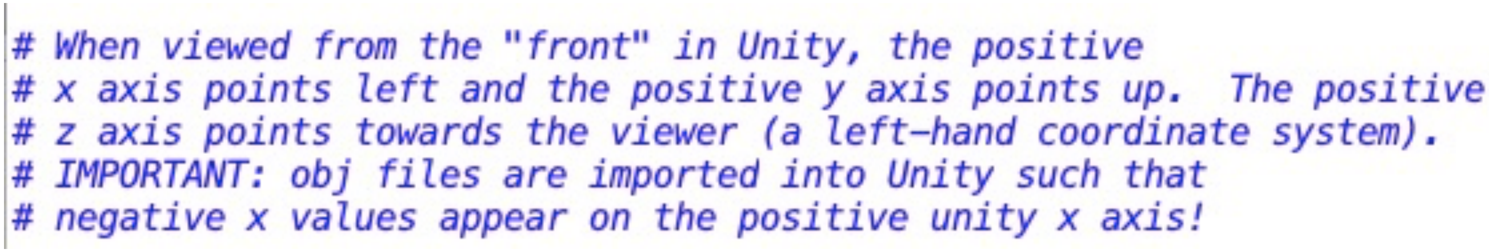
Rwanda Topology Object for Unity

- **Build a 3D terrain object and import it as a game object into a Unity project**
  - The tif elevation image is 7342 cols by 6460 rows, and the samples are spaced every 30m
  - Sub-sample by a factor of 10 in each direction to get a picture of manageable size
    - ✓ Do this by replacing each block of size 10x10 with the maximum value in the block
  - Color triangles below Rwanda's minimum elevation green. Color triangles above a threshold, *T*, that you choose, blue. Color remaining triangles red. Choose *T* to give a pleasing result and *include your choice of T in your report*
  - Position a directional light and camera in your simulation in a manner that achieves a nice effect w.r.t shadows
  - Include in your report, as appropriate, screen captures taken from inside the Oculus headset of your terrain object

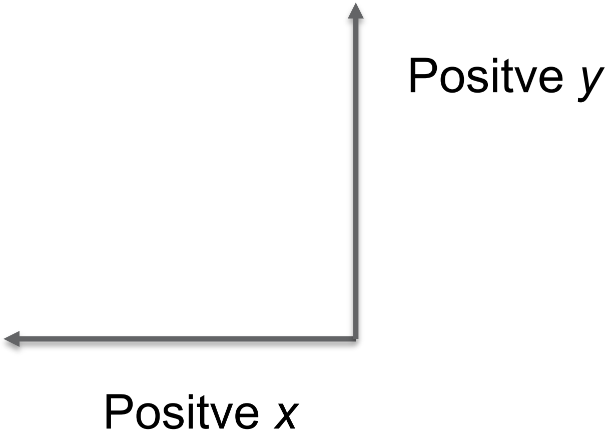
More Details

- **Your object will be in the .obj format as discussed in class and later in this document**

- **Axes in Unity**



```
# When viewed from the "front" in Unity, the positive
# x axis points left and the positive y axis points up.  The positive
# z axis points towards the viewer (a left-hand coordinate system).
# IMPORTANT: obj files are imported into Unity such that
# negative x values appear on the positive unity x axis!
```

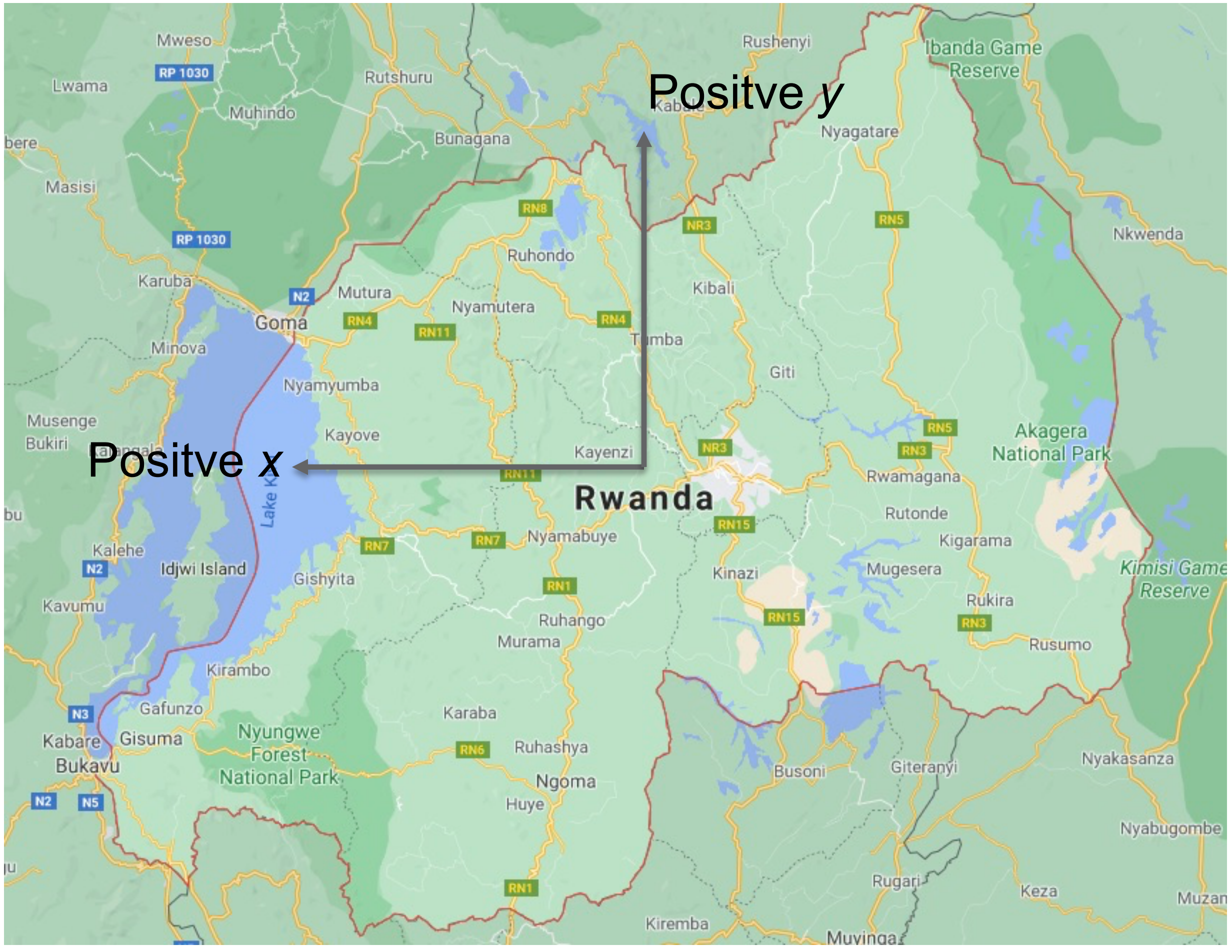


You are in positve z space when looking at this diagram. This is a left-hand coordinate system!

More Details

Carnegie Mellon Africa

- **Your object should be oriented like this w.r.t Unity axes**



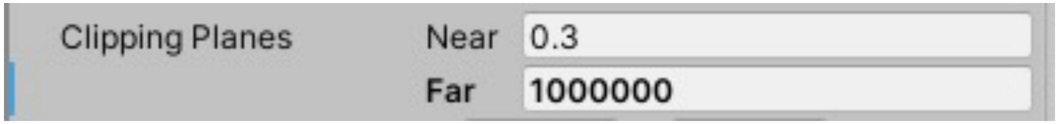
This means values to the west of the y-axis should be negative when you build your .obj object! Values to the north should be positive

More Details

- **Import your .obj object into Unity by dragging it into the project area**

- **You will need an OVRCameraRig and a directional light**

- **You will need to adjust the far clipping plane value to something big**
  - Experiment with this value and see what effect it has
  - Explain in your report why it needed to be changed to something large



Clipping Planes Near 0.3

Far 1000000

- **Include in your report the virtual height above the terrain at which your camera is positioned**

- **Use at least two positions for the directional light and discuss how the lights position affects what you see**

More Details

- **Some references on OBJ format**
  - http://paulbourke.net/dataformats/obj/
  - https://www.fileformat.info/format/wavefrontobj/egff.htm
  - https://all3dp.com/1/obj-file-format-3d-printing-cad/
- **You can find some free OBJ models here**
  - https://free3d.com/3d-models/obj