Earthquakes Visualization Project

Alari Varmann [*], Master's Student in Computational Science
Supervisor: Anders Hast

Introduction

The data provided covers the whole Italy between the summer of 2012 to the autumn of 2014.

I am submitting the code in which the data isn't read to VTK format so the animation could be slower, but it seems actually quite fast. I did my very best, but I couldn't get the VTK conversion working normally (see below). Maybe you could help me analyze why my VTK file didn't work (I uploaded this as well).

Visualization Approach Taken

This is the 3D visualisation displaying the earthquakes in Italy and in heighbouring regions.

- Sphereglyphs were chosen to represent the earthquake magnitude. The glyph size is dependent and corresponds to the Richter magnitude of an earthquake. Since the earthquake magnitude is a scalar attribute and a sphere is obviously symmetric, thus isotropic with respect to direction - given a center, it is parametrized only by its radius, thus it fits ideally to encode for the magnitudes. The Tufte Visualization lie plays a small effect with change in magnitude corresponding to nonlinear change in percept since sphere volume is radius cubed, but the relative influence is small if the scalefactor is not too high like in this case.
- The static space dimension is 3D - latitude, longitude, depth. The user can interact with the visualization interactively – zoom and rotate. Through interactor, user can add the 4th dimension – time – and make the animation dynamic.

Python VTK Coding Approach

The Python code consists of the following methods

- `main`
- `parse_csv` - Takes the events csv file as an input and returns a list of lists containing row values: [timestamp, longitude, depth, magnitude, source].
- `distance` – Returns distance in kilometres based on latitude and longitude of 2 coordinates.

[*] e-mail:alariv8@gmail.com

- `generate_3D_map_data` – Uses the data to create VTK objects for rendering – location, magnitude and time (seconds). Location is of vtkPoints class, magnitude and time are vtkFloatArrays. Minimum filtered strength is set here to be 2.8 to reduce the input data dimension somewhat.
- `convert_to_vtk` — I had this method in my static file and I could convert this into VTK format. `However, this did not work`. I literally spent my whole Friday evening trying to get it to work, but it didn't. It seems that for some reason, the Active scalars were never set to "Strength" – and I don't know why. Maybe there was some other problem.

```
unstructuredgrid_data=vtk.vtkUnstructuredGrid()

# Read arguments
unstructuredgrid_data.SetPoints(points)
unstructuredgrid_data.GetPointData().AddArray(strength)
unstructuredgrid_data.GetPointData().AddArray(time_)
unstructuredgrid_data.GetPointData().SetActiveScalars("strength")

filename = "earthquake_vtk_r.vtk"
output_at_unstructured=vtk.vtkUnstructuredGridWriter()
output_at_unstructured.SetInput(unstructuredgrid_data)
output_at_unstructured.SetFileName(filename)
output_at_unstructured.Write()



f render_3D_points(min_filtered_strength, max_filtered_strength):
  # if I define a docstring for a class A, then if I run help(A) , then I get the docstring output

  unstructured_reader = vtkUnstructuredGridReader()
  unstructured_reader.SetFileName("earthquake_vtk_r.vtk")
  unstructured_reader.Update()


  threshold_filter = vtk.vtkThresholdPoints() # creates the filter for magnitude filtering
  threshold_filter.SetInput(unstructured_reader.GetOutput())


  #threshold_filter.SetInputArrayToProcess(0, 0, 0, "strength")
  threshold_filter.ThresholdBetween(min_filtered_strength,max_filtered_strength)
  threshold_filter.Update()
```

Figure 1: *12.*

It was very strange that I didn't get it to work because my friend did. I suspect there were some problems with AddArray command – it would be nice to get it solved since I spent so much time on it....

- `render_3D_points` – Takes location, magnitude, time and data as input from the file written by `vtkUnstructuredGridWriter` using `vtkUnstructuredReader`.
  `PNGReader`, `vtkTexture` were used to get the image and texture map it onto the plane, defined by 3 points (planesource and 2 additional points). Filtering is done by earthquake strength. In general, the usual VTK pipeline is followed: data → reader → mapper → actor → renderer → renderwindow → interactor.
- `KeyboardInterface` class – gives the option to start the earthquake animation by pressing "0" key. Filtering is done

by time, producing a dynamic animation

- script part of the code - sets the camera, creates glyphs and some other minor things.

I designed my colorbar to have as gradual lightness change as possible and don't get mixed up with the map colors.

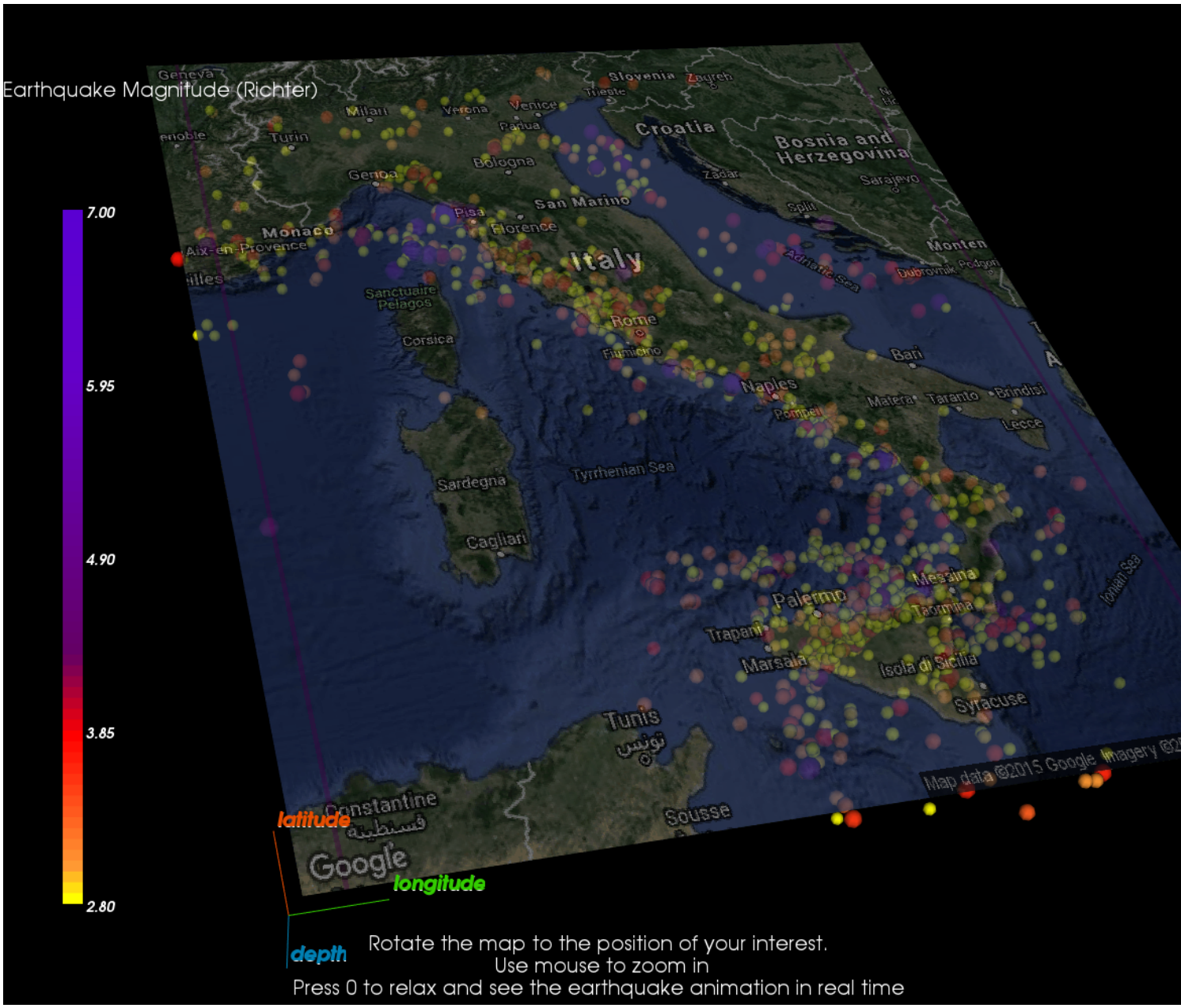

Figure 2: *10.*

ColorMap

It is known that the key terms for the use of color as a label are: distinctness, uniqueness, contrast with background and prevent color blindness. Lightness contrast is the most important aspect for readability of text and fine detail and visual acuity depends on lightness/brightness of colors.
The scalarbar colormap should represent constant lightness contrast and perceptual linearity as we move across it, but it doesn't suite for absolute quantitative assessment.