LiquidGlobe

Curran Kelleher 11/19/2010

abstract

LiquidGlobe is an interactive visualization tool comprised of two coordinated views: 1.) a 3D globe with a choropleth map on it 2.) a timeline with a time series line chart. The tool is characterized by two main properties of both the globe and timeline: 1.) Smooth, continuous pan and zoom based on a “mass in a viscous fluid” simulation (for camera positioning and motion) 2.) automatic drill down and drill up (including automated data fetching and data source resolution) based on threshold zoom levels. The data driving the colors of the choropleth map and the lines in the timeline will be derived from the Semantic Web and Universal Data Cube, enabling seamless automatic transition between data providers during interaction.

- **The Globe**
  - Visualization
    - A 3D sphere covered by a choropleth map
    - Hierarchical multiscale geometries on the sphere
    - Geometries linked with the Semantic Web
    - Geometry names used as labels
    - Geometry properties (measures and indicators) as color
  - Interaction
    - Smooth, animated, continuous pan+zoom
      - Based on “mass in viscous fluid” simulation
    - Automated drill-down based on zoom
      - i.e. Automatically break states into counties at a threshold zoom
    - Brushed selection and probing between the timeline and globe views

- **The Timeline**
  - Visualization
    - Standard time series line plot visualization
      - Visualization of records visualized on the globe
      - Uses spline interpolation of data points
      - Coordinated with globe view
        - Timeslice in timeline drives input to globe choropleth map
          - Accounts for spline interpolation
            - Enables very smooth color animation
  - Interaction
    - Smooth, animated, continuous timeline navigation (pan + zoom)
      - Based on “mass in viscous fluid” simulation
    - Automated drill-down based on zoom
      - i.e. Automatically break years into months at a threshold zoom
    - Brushed selection and probing between the timeline and globe views

- **The Data**
  - The geometries
    - Quadstream geometry
      - Multiscale geometry representation
      - Stored in a database
      - Browsable with LiquidGlobe
        - Using prefetching based on anticipated motion

                - Geometries linked with identifiers in the Semantic Web
            - Time frame
                - Serves as metadata for the temporal relevance of geometries
                    - i. e. “Massachusetts has existed since February 6, 1788”
                - Includes
                    - Beginning of existence/relevance
                    - End of existence/relevance
                - Both (beginning and end) could be either:
                    - Point in time
                    - Interval in time
                    - Point at “infinity”
                        - “Since the beginning of time”
                        - “Until the present”
        - The measures and indicators
            - Retreived from the Universal Data Cube
                - i. e. “Population” from the US Census
                - i. e. “Population” from EuroStat
                - i. e. All data available in Wikipedia fact boxes
                - i. e. “Temperature” on a multi-resolution grid from weather sources
                - i. e. “Wind direction” on a multi-resolution grid
                    - Could be visualized as a vector field
                - All of the above accessed and visualized simultaneously
                - Other data sets could be imported
                    - aggregated Flickr geotag data
                    - aggregated OpenStreetMap (OSM) data
                        - Measures aggregated from Map Features
            - Loaded dynamically (using SPARQL/UDC-query) as navigation occurs
                - Using prefetching based on anticipated pan+zoom motion

- **Collaboration Aspects**
    - The tool can run in a web browser
    - View configurations can be stored published on the web
    - Derivative works can be created and published
    - View configurations can be embedded into web pages
        - Either globe only, timeline only, or both together
        - View configurations remain interactive when embedded
            - Enables audience to further explore the presented data/findings

11/19
Architecture sketch and candidate technologies
Goal: keep technology involved as simple as possible

Question: How will multiscale polygons be stored and queried?
Answer: 2 options:

- stored using a database, queried via a RESTful web service
    - unnecessarily complex, would require specialized server software
- stored using the file system, queried via direct HTTP GET
    - the simplest way to go, would allow usage of standard file servers

HTML5 tests:

- Got canvas full window size from here

12/17/2010

- Decided to try pursuing Java as a base technology rather than HTML5
    - Does not preclude a future HTML5 client
- Tried getting Processing to work as a Java library using OpenGL. Window resize did not work. Did not launch consistently (sometimes just hung). Decided against it, decided instead to go with raw JOGL from the ground up.
- Made JOGL test
    - F11 to make fullscreen.
    - ESC to exit fullscreen or exit the program.
    - Pass -Dsun.awt.noerasebackground=true to the VM for no flickering on resize.
    - The aspect ratio is always maintained.
    - Drew a circle

12/18/2010

- Realized a fundamental requirement is to work on multiple devices: large multitouch (most likely on Windows), small multitouch (on Android), and normal desktop PC (Windows, Linux, Mac). Therefore an additional layer of abstraction is needed between the application code and the environment.
- Recreated a version of the Processing API to account for
    - Potential simulataneous multitouch and mouse interactions
        - changed “mousePressed” to “pointPressed” etc.
            - the “id” argument -
    - Running with different display implementations
        - Created a Java2D based implementation
        - Created an OpenGL (via JOGL) based implementation

12/22/2010

- Idea: Joystick
    - (x rotate, y rotate) = pan
    - z twist = zoom
    - thumb (x,y) = select element
- Idea: use tile-style “zoom levels” for labels
    - At each zoom level, compute a label layout
- Idea: Each geometry has an associated min and max scale at which its label is visible.
    - UI contains a “Scale window” in which labels are displayed
        - Tied to zoom

1/02/2011

-