The Burke Institute

# Knowledge Web
# UI Analysis
## Version 0.3

## Revision History

| Date | Version | Description | Author |
|---|---|---|---|
| January 1, 2002 | 0.2 | First Draft | James A. Zaun |
| January 2, 2002 | 0.3 | Corrections and updates. | James A. Zaun |

# Table of Contents

## 1. Scope

This is a preliminary user interface analysis of the James Burke's interface spec, (see [Burke] reference below). The document analyzes Burke's vision from both a usability and an implementation prospective and suggests some changes to improve the user experience. I only made it part way into Burke's vision with much remaining to analyze.

## 2. References

[Burke] Burke, James, *James Burke's Knowledge Web: Outline description and interface*, December 2nd, 2002.

[Welch] Welch, Terry, "A Technique for High-Performance Data Compression", *IEEE Computer*, June 1984.

[Kumar] Kumar, A. and Fowler, R., *A spring modeling algorithm to position nodes of an undirected graph in three dimensions*, Technical report, Department of Computer Science, University of Texas, 1994.

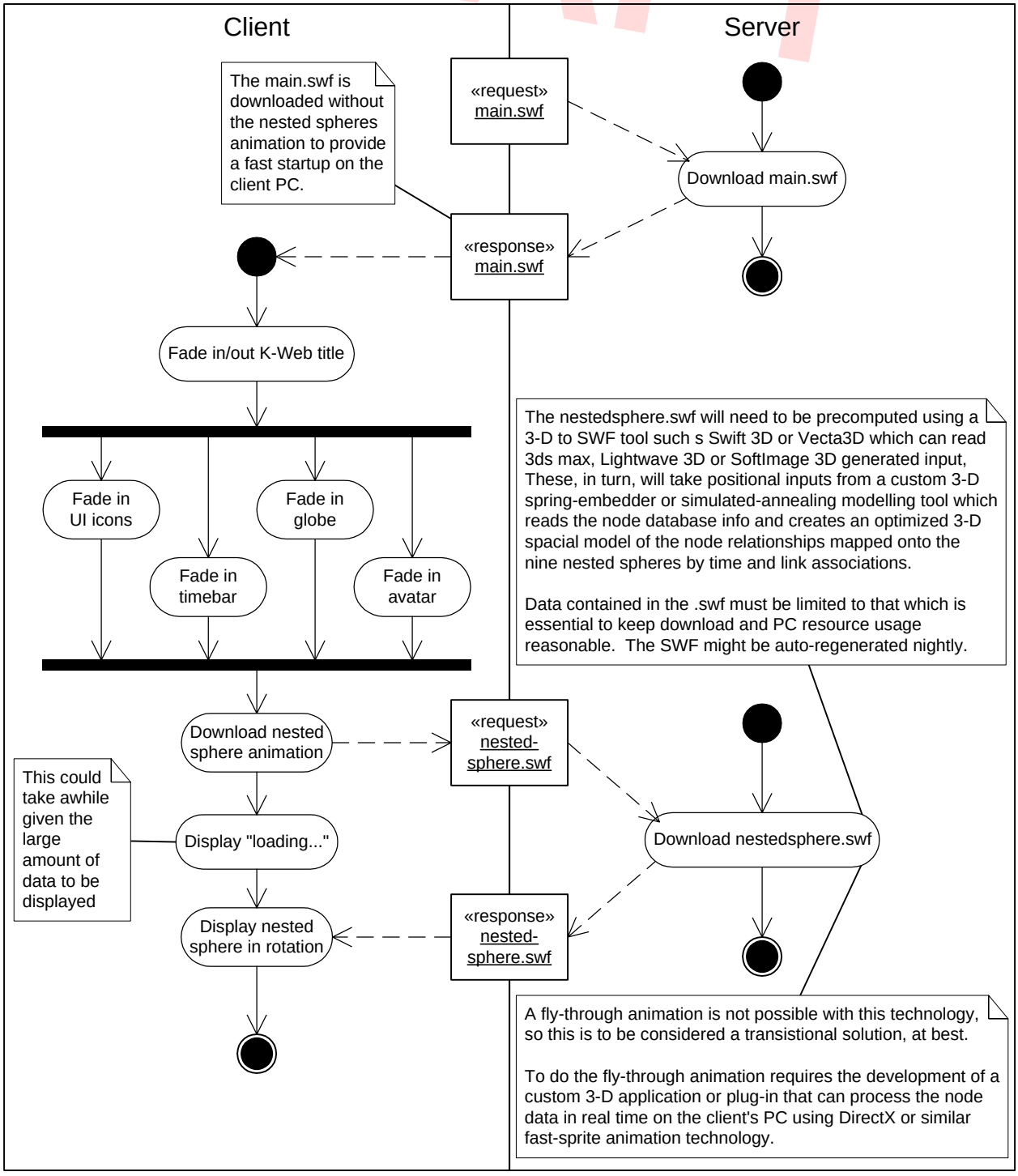
## 3. Glossary

**Term**: Definition...
continues on next line.

## 4. Startup screen

4.1 **Activity Diagram:** Client and server actions to display startup screen.

4.1.1 Justification: [Burke] Last part of INTRODUCTION and first part of NESTED SPHERE CONSTRUCT.
4.1.2 Priority: Essential. However, the fly-through requirement will have to be postponed.
4.1.3 Risks: Auto-generating the nested sphere SWF . Fly-through custom app development.
4.1.4 Preconditions: Client log-in needed to maintain persistent personalized journeys.



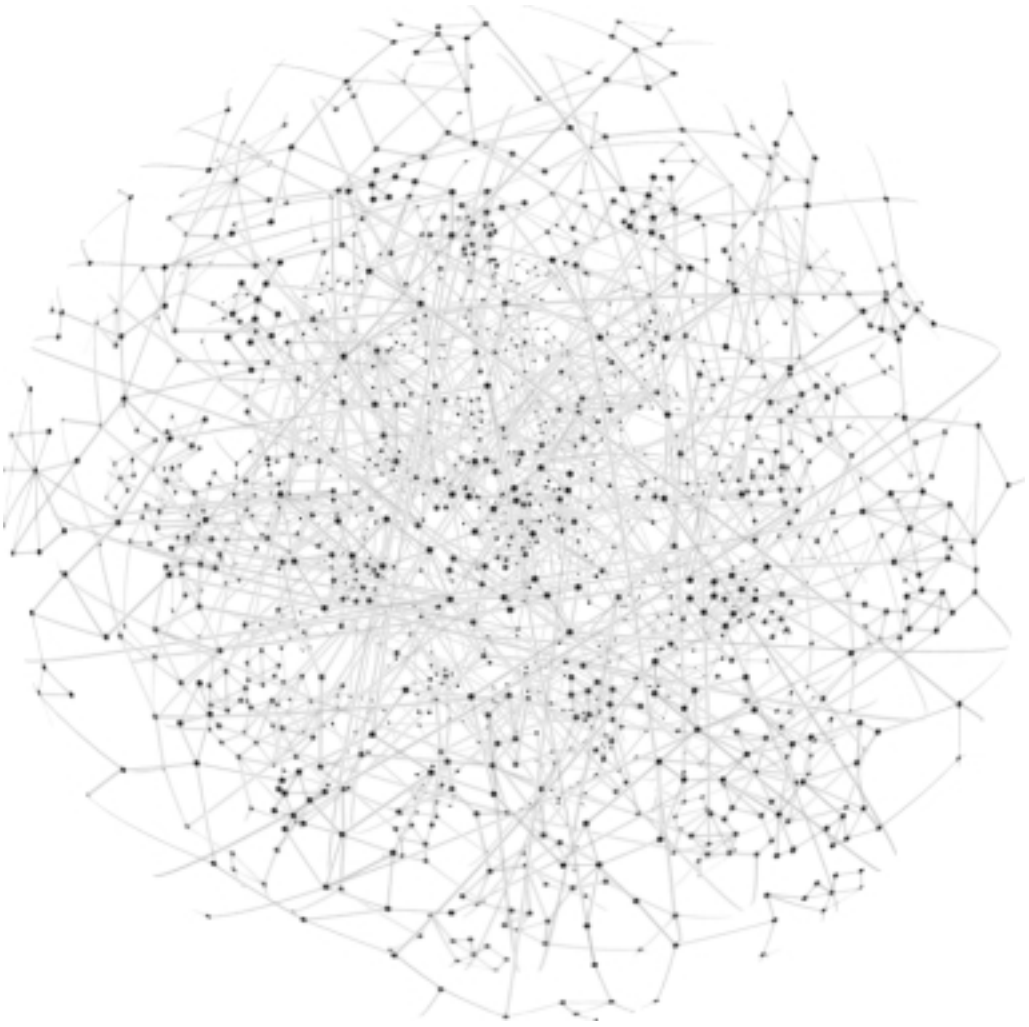

  

## 5. Nested sphere appearence

5.1 **Node renderings:** Nested sphere rendering issues.

5.1.1 Reference: [Burke] NESTED SPHERE CONSTRUCT section.

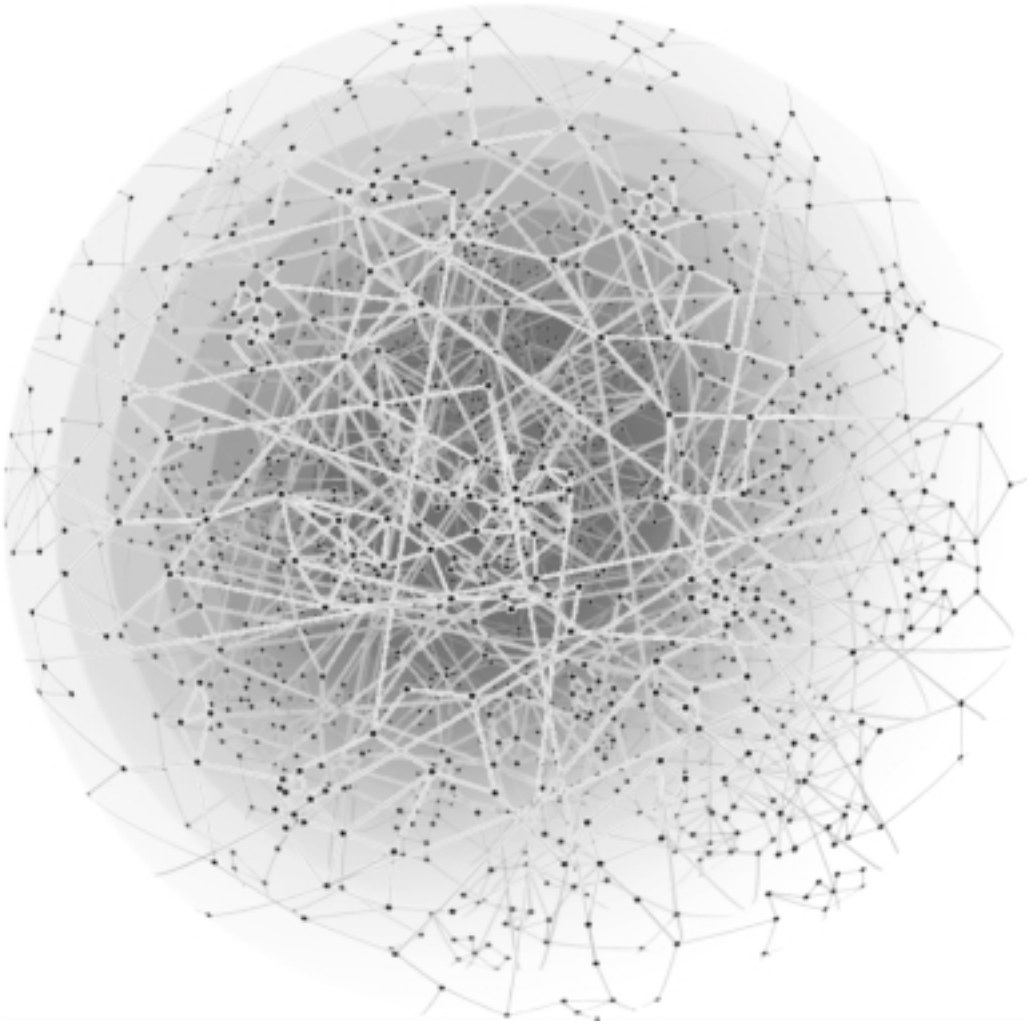
5.1.2 Rendering all nine nested spheres with the filaments between them will create a graph that is so dense that humans will find it difficult to comprehend. The nine layers of nested onion skins isn't immediately obvious either. Maybe the use of different colors would help differentiate the layers but how would we deal with the links that cross between layers? (And, there are considerations with color blind users too.) Here is a mockup of such a view out to nine layers; (image colors are inverted for less demanding output when printed).



Actually, this rendering is a bit incorrect as I could not figure out how to turn off hidden line removal from the 3-D rendering program used to generate this. So, what you see is half the data on the front hemispheres of each layer only.

5.1 **Node renderings:** Nested sphere rendering issues *(continued).*

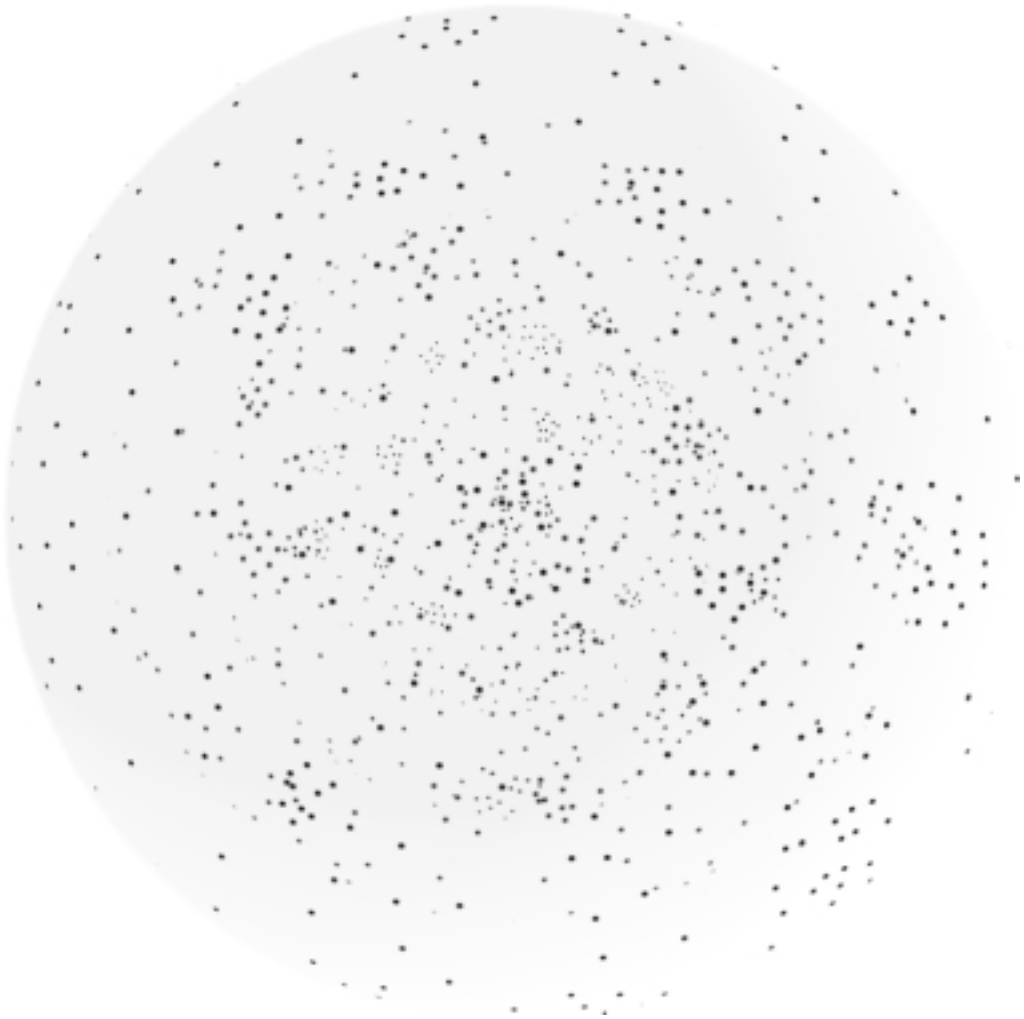
5.1.3 Adding translucent onionskin layers masks the last four layers as the node brightness is reduced by 50-90% but it does give a better sense of how the onionskins are nested within each other. Here is a mockup of such a view out to nine layers; (image colors are inverted for less demanding printed output).



The onionskin spheres cause another problem. There are places where the shading of the onionskins matches that of the links. To get around this, I added a halo around the links and nodes so they are still visible. Maybe there are better ways to deal with this.

5.1 **Node renderings:** Nested sphere rendering issues *(continued).*

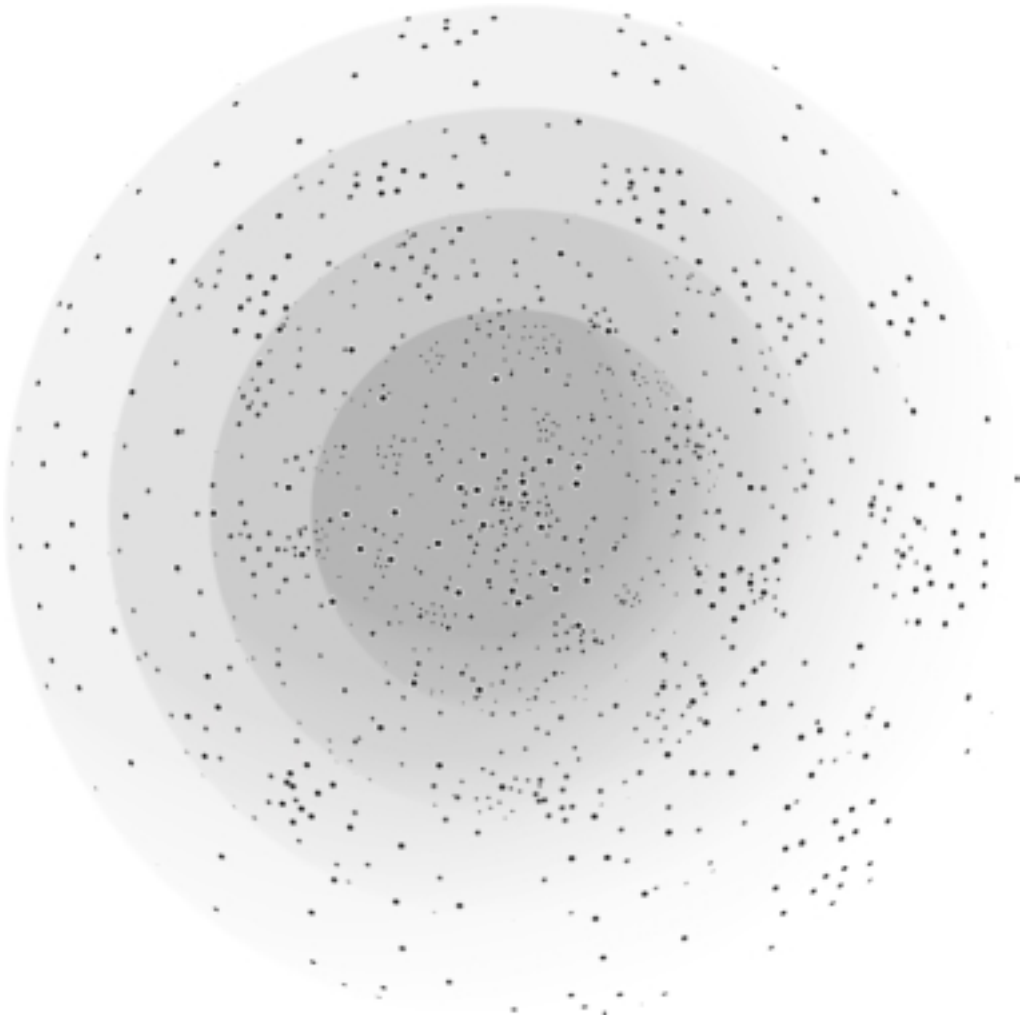
5.1.4 Perhaps it would be better not to show the links between nodes or all the layers for two reasons: (1) the output looks more like a star field which is more pleasing to look at, (2) it would require less download time and less computational capacity on the client PC to render. I would wait until the user mouses over a node to show the links that radiate from the moused node to adjacent nodes and so on. Here is a rendering out to 4 layers without the links; (image colors are inverted for less demanding printed output).



5.1 **Node renderings:** Nested sphere rendering issues *(continued).*

5.1.5 Here is the same "star field" with the translucent onionskin renderings added. Because the number of visible layers is reduced to 4 (in this instance – might be a user preference), the visual distance between the layers can be increased. The idea is that if one where to fly into one or more outer layers then one of more inner layers would then become visible maintaining the maximum number of visible layers at 4; (See diagram below, image colors are inverted for less demanding printed output).

5.1.6 If there are links from the moused node to nodes in invisible layers, only those links and nodes needed for selection in the invisible layers are shown; (the layer, as a whole, still remains invisible). Once a node in an invisible layer is selected, the whole layer becomes visible, while previous layers outside the new selected range (except for selected nodes) become invisible to maintain the no more than 4 rule.



5.1 **Node renderings:** Nested sphere rendering issues *(continued)*.

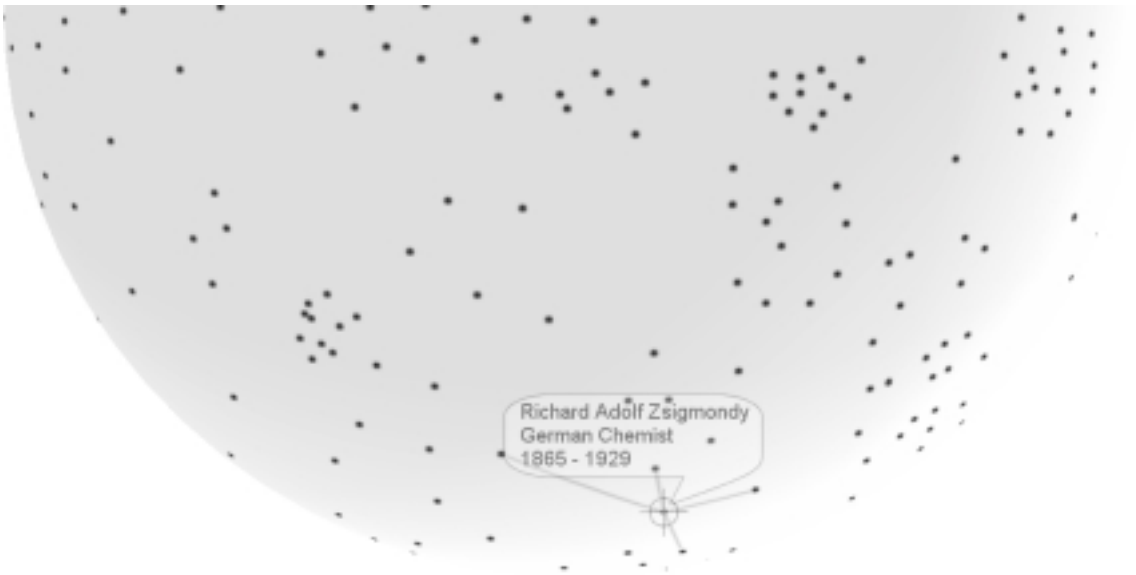
5.1.7 Reference: [Burke] last part of NESTED SPHERE CONSTRUCT.

5.1.8 Identifying nodes: The [Burke] spec says that the nodes transform from spots to names as one "flys" closer. I don't think this will work very well as one would have to get so close to a spot to read its name that all prospective is lost. Perhaps a better way to handle this is as follows: as a target cursor moves and passes over a spot (node), a translucent callout (tooltip) along with the node's connecting filaments to adjacent nodes pops into view. When the cursor leaves the node, the callout fades away almost immediately but the filaments fade away much more slowly so the user has time to trace the connections to adjacent nodes when moving the cursor. If the cursor touches a filament before it decays to 50% brightness the corresponding adjacent node and it's filaments are automatically highlighted as if the cursor passed over the adjacent node. (Perhaps touching a filament also displays a callout showing the node relationships.) One advantage of using just-in-time callouts is that more information than just the name can be displayed without having to zoom in on a densely populated display. [See diagram below.]

5.1.9 Selecting nodes: The [Burke] spec says clicking on a spot or location name jumps the user to the corresponding external online resource (website). This seems overly disruptive to me. I would make that a double click. I think a single click should hold the state of the current node such that the filaments don't fade and the selected spot slowly pulses (like a pulsar; color might be used as a secondary indicator but accessibility issues override). As long as the user selects nodes that are connected all previously selected nodes with associated filaments remain active. If the user clicks on a non-adjacent node, all previously selected nodes return to their normal state causing their filaments to fade; (we need a multiple undo to restore previous selections). If the user clicks again on a previously selected adjacent node that node only returns to its normal state without affecting other nodes (even if that means braking a chain in two; the next time a node is selected any chains no longer included in the selected chain revert to their normal state).

5.1.10 Right clicking on a node: [Windows/UNIX only] displays a context menu with additional options such as setting the begin and end points of computed journeys; (what is the equivalent approach on Macs? – hold the button until a context menu appears?)

Here is a rendering of one layer where links and node identifications are shown in a just-in-time fashion; (image colors are inverted for less demanding printed output).



5.2 **2-D projections of nodes mapped to 3-D spheres:** If we ignore certain issues with prospective, projecting 3-D nodes into 2-D is straightforward and possibly efficient enough to render in SWF (Shockwave-Flash).

5.2.1 Computing node positions. Early last year I created a true 3-D rendering of 100+ nodes centered around Shelley using *3-D spring-embedder modeling* [Kumar] to VRML2. The same software, with several modifications, could be used to generate the node positions for the entire KWeb with the addition of time position information for the nested spheres which was not attempted in the earlier rendering. The modifications are not trivial and it would take several hours of running time to optimize the node position data, but once the node positions are computed they would not have to computed again until new nodes are added or subtracted. The previous rendering was generated directly in rectangular (Cartesian) 3-space whereas I now believe it would be better to generate node positions in spherical coordinates because that system fits the KWeb more naturally especially with regard to rotation. Time maps directly to $r$ and, $\theta$ and $\phi$ map to locations on a sphere (i.e. latitude and longitude).

5.2.2 Node data. The amount of node data created for the nested spheres needs to be minimized to keep the download time to a minimum. The only node data needed to render the nested spheres (without links) are:

- $i$ node id in 16-bit unsigned integer (that limits the KWeb to 65,536 nodes).
- $r$ radius in 16-bit unsigned integer which could map to 1/12 of a $d_{yr}$ year increment in AD and 1/6 of a $d_{yr}$ year increment in BC covering 2,730 years AD and 5,461 years BC with a zero offset of 32,768. We'll ignore the fact that there is no 0 year between BC and AD. $d_{yr}$ is computed from the midpoint of a person's lifetime or event range. $d_{yr} = (r–32768)/6$ when $r < 32768$; and, $d_{yr} = (r–32768)/12$ when $r > 32768$.
- $\phi$ longitude in 16-bit unsigned integer in increments of 0.000095874 radians.
- $\theta$ latitude in 16-bit 2s-complement signed integer in increments of 0.000095874 radians.

That is 8 bytes of information per node, so with say 2,400 nodes the total amount of data downloaded to generate the nested spheres is only 20K bytes. If we were to add filament information that would take up another 80K bytes which could be loaded in the background while the nested spheres are being rendered. Other information such as the node name, dates, descriptions and links could be download on an "as-needed" basis from the server as the user's mouse pointer passes over the nodes. (Issues with server delays and the need to maintain a continuous TCP connection with the server may call for preloading that information as well, at an estimated additional cost of 300K bytes if compressed with Lempel-Ziv-Welch compression [Welch].)

5.2.3 Projecting nodes onto a 2-D screen. Because the node data is in spherical coordinates each node position must be converted back into rectangular coordinates before being rendered. In addition, we must consider our viewpoint when looking at the nested spheres which can also be expressed in spherical coordinates. Here are the transformations from spherical to rectangular which must be computed for every node after a viewpoint movement:

$$x = s_v\, r \sin\theta \cos\phi–\phi_v$$
$$z = s_v\, r \sin\theta \sin\phi–\phi_v$$
$$y = s_v\, r \cos\theta$$

Rotating our view around the equator of the nested spheres is equivalent to changing our viewpoint $\phi_v$ and only $x$ and $z$ needs to be recomputed for each node. Zooming in or out of the center of the spheres is equivalent to scaling with multiplier $s_v$. If we limited our movement with respect to the spheres to simple rotation and scaling combined with simple pan movements over the converted rectangular coordinates, it might be possible to "fly," in a limited sense, around the nested spheres using Flash.

5.2.4 Node sizing and clipping regions. To get a 3-D "feeling" without using true prospective we can scale the size and brightness of the nodes according to the computed $z$ using the inverse square rule. Nodes with a negative z that exceeds some threshold can be hidden from view to give the appearance that the node is behind us. The initial value of $s_v$ would to set such that the outmost nested sphere would just fill the screen or window canvas.

# 6. Timebar

6.1 **UI issues:** The timebar is used to narrow the range of visible nodes for easier selection.

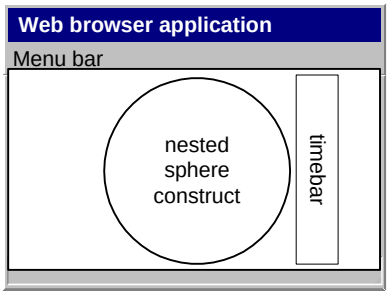
6.1.1 Justification: [Burke] Entry via timebar/nested sphere construct in SCREEN-EDGE ICONS/ENTRY MODES.

6.1.2 Priority: Important.

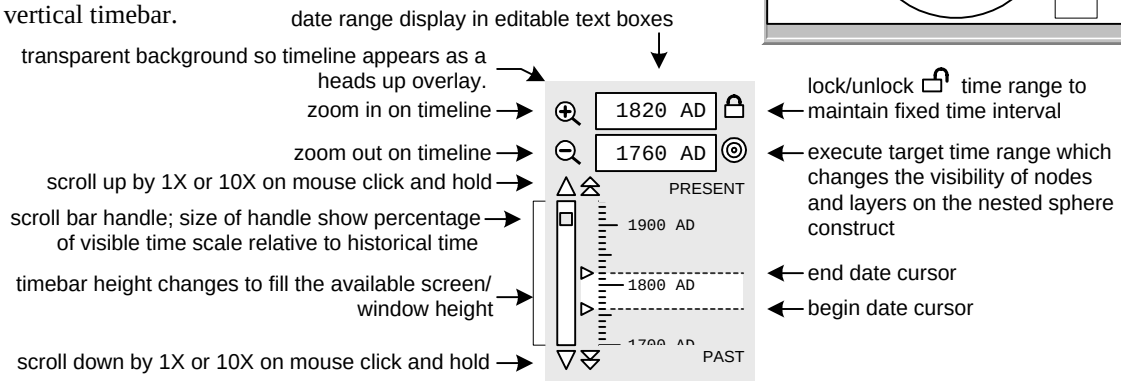
6.1.3 Risks:

6.1.4 Preconditions: Nested sphere construct implemented.

6.1.5 UI issues: Due to the aspect ratio of the typical computer screen (with more width than height) and the need to maximize the available screen real estate for the nested sphere construct, it seems best to place the timebar left or right of the spheres rather than above or below. That, implies that the time bar should be vertical rather than horizontal, like this, ⟶




6.1.6 Timebar construction. The [Burke] spec hints at but does not fully develop the notion of time periods. A single scroll bar is not sufficient to capture time periods as two cursors are needed to set the time range. One may also what to lock time intervals such that moving one cursor also moves the other by the same amount, thus maintaining a fixed time window. Here is a possible rendering of a vertical timebar.




6.1.7 Timebar behavior: Initially the time range is set to all historical time (e.g. 2050 AD to 5,000 BC) and the date cursors are set to the end limits the timebar. In addition, the size of the scroll bar handle changes according to the percentage of the timebar coverage over historical time (but if the coverage is below 3% the handle shrinks no further so there is enough of a handle to drag with a mouse). Initially the handle would cover the entire scroll range as all of history is visible. When the "zoom in" icon is clicked the time range is narrowed from centuries to decades to years centered on the midline of the timeline. In the mockup above the zoom is set to decades. One or both date cursors may disappear from view during zoom ins as the cursors may move outside the visible timeline range but their positions will be marked by lines on the scroll bar so they can be brought back into range; (this is not visible in the mockup above as both cursor marks are covered by the scroll bar handle).

6.1.8 Event indicators: Perhaps there could be an "event detail" feature that expands the timebar to the right with events in history listed next to its date position on the timeline. These events would be clickable. A single click causes the corresponding node on the nested sphere construct to pulsate and highlight its filaments to adjacent nodes; (same behavior as single-clicking on the node directly). A double-click causes the external resource to be displayed. The implementation of this could be messy as additional information will have to be added to the system state to determine which events are significant enough to be promoted to the timeline and at what zoom factor. Or, we could use Google's approach which promotes nodes with the highest number of links referring to it as being most significant.