# After tree view controls

A tree view control is a common graphical user interface widget that allows for the navigation of a hierarchy of information. This control allows a user to click into the nodes of the tree, commonly called branches, to view the farthest node or leaf of information. The information could be a folder structure with files inside or a networking domain with devices and device component IDs. While useful for exposing the hierarchies of data or the backend programmatic signature of the data, tree views should be employed with great scrutiny when concerns for swiftness of workflow are paramount to usability in a software context.

From their outset, tree views naturally introduce complexity to a user about the data they present. This obvious inheritance is often not necessary for a user to accomplish the task at hand and so the cognitive work of the user that is immediately escalated becomes an unnecessary source of friction that impedes expediency and builds a tone of intricacy into a software product that could otherwise benefit from a more fluidic, lightweight, and apparently simple interface.

Tree views demand that the user know *a priori* or learn the container structure and embedded elements, which can require significant drilling into node elements. This hunting and seeking of information, especially in the case of first time use, requires time as well as physical and mental effort to seek, click, seek, remember, and click (if luck permits) that unapologetically take away from any accomplishment of acting on the data that a user is hunting for. The question, 'Where is the thing?' heavily burdens the workflow of needing to do something with 'the thing' once it is located. While grouping elements within a container structure that may inherit a hierarchy is a distinct and foundational human tendency, within the context of software applications, there are other control presentation methods that can increase the speed that a desired outcome is attained and, through this unhindered workflow, the user gains a sense of satisfaction in their efforts that can lead to the sale or renewal of more products in a strict business sense and promote infectious positive energy in the broader societal sense.

A very clear usability limitation of tree views is that they often are restrained to vertically-biased rectangles, yet can contain nodes that expand two, three, or more levels deep. This presentation then requires the user to perform a kind of horizontal scrolling acrobatics, to expand a node, scroll, scan, and hopefully click to expose the details of the target node for further action. The fact that tree views are often not the location of action on a node, and if they were, are severely limiting in the screen real estate they permit, means that their implementation demands an engineer to exquisitely balance the concepts of action and navigation- not simple feats that challenge senior and principle level developers and designers alike. All the time that a user spends engaging with 'tree swinging' as they navigate a potentially complicated hierarchy is time spent not focused on achieving their ultimate goal; whatever that may be, it is most certainly not locating a node on a tree.

If we can collapse the distance between a user's current location and their destination then we have necessarily reduced the time of their actions and promoted their satisfaction with their efforts. Data structure and hierarchy can be crucial to understanding how to solve a given problem and learn workflow, but they can be exposed in areas perhaps more appropriate than a tree view. Conversely, a tree view can be immensely useful if the hierarchy it uncovers is shallow and if the tree itself is 'short' enough to allow for quick scanning for target nodes. As the tree's complexity grows, its usefulness decays rapidly, and much like physical trees under human gaze, it becomes an object of bewilderment and paralyzing bliss rather than an object under control. This statement applies with greater warning as the number of nodes encountered proliferates through an era of planetary size data structures.