Cover Page

Topics:
C2 Experimentation
Network-Centric Applications

Title:
An Experimental 3-D Framework to Support C2

Authors:

Dr. Hubert D. Callihan, POC
NetSpace Corporation
512 Ruby Street
Windber, PA 15963
United States of America
(814) 467-4537 Voice
(814) 467-4537 Fax
(814) 242-7282 Cell
callihan@netspacecorp.com

Mr. John A. Balash
NetSpace Corporation
11001 Belaire Drive
North Huntingdon, PA 15642
United States of America
(724) 446-7377 Voice
(724) 446-5559 Fax
 (412) 841-2980 Cell
balash@netspacecorp.com

7th International Command and Control Research Technology Symposium

Enabling Synchronized Operations

An Experimental 3-D Framework to Support C2

Hubert D. Callihan
John A. Balash

NetSpace Corporation
11001 Belaire Drive
North Huntingdon, Pennsylvania 15642
United States of America
(724) 446-7377 Voice
(724) 446-5559 Fax
callihan@netspacecorp.com

Abstract

Applications that are hosted in a web framework realize many advantages: familiarity to the user, captive environments, rapid development, mature toolsets, and reduced development time and cost. More recently, applications based on web 3-D technologies are showing considerable promise in C2. Because C2 applications tend to be very complex, casting them in a 3-D framework simplifies them considerably due to unlimited rapid drill down, visual realism, and the capacity to support C2 decision-making. We present an experimental scenario using a 3-D web-based framework to show the broad capabilities of interactive 3-D to support C2. This scenario addresses communication network infrastructure categories such as dynamic discovery and configuration, device connectivity, network performance, and device performance in a live demonstration. Although the domain of interest in this experiment targets network management in particular, we will demonstrate the ease with which this framework can be extended to many other domains of interest relevant to C2. The overall benefits realized will be shown as (1) improved understandability through a variety of visual contexts, (2) improved decision-making through manageable interfaces, (3) rapid application development using COTS tools integrated into the framework, and (4) potential for multiple collaborators to cooperate in a shared 3-D workspace.

1. Introduction

Previous papers have been presented in CCRP publications demonstrating the viability of a three-dimensional interface to C2 scenarios [Gardner, *et al. 1999,* Callihan and Balash, 2001]. These interfaces were based on recent advances in *web-based technologies*, which have been shown to hold considerable promise for domains of interest where a rapid, reliable, and timely push of information is needed. Our immediate interest has been in extending the work previously reported using 3-D web environments, integrated with the full suite of complementary web technologies, to create a compelling visual environment to support the C2 process. We present a new demonstration scenario as an experiment to show the efficacy of current work in connection with prior results.

Command and Control has been widely studied and shares many of the same general characteristics as commercial systems supporting varied decision-making capabilities. These commercial systems include: (1) identification of the current state of the business climate, (2) using what we know to compare it to some desirable profit direction, (3) decide what to do, and (4) take action to realize the decision. Militarily, these features were reported earlier [Boyd, 1984, Orr, 1983, Lawson, 1979, and others]. Lawson's Model was the basis for 3-D visualization and subsequent drill-down [Callihan and Balash, 2001] through the phases of his C2 process including (1) *sense*, (2) *process*, (3) *compare*, (4) *decide*, and (5) *act*. This same model is used in this paper to demonstrate the features of an extended 3-D experimental framework to support C2.

2. **Approach**

Several items are important for web-centric C2 application development as previously reported [Gardner, *et al*.].

- "Examine the use of web-centric technologies to rapidly prototype significant C2 applications. Many web components of varying degrees of complexity are available in the public domain or, in the case of 3-D models, from the user's own CAD, CAM or similar archives.
- Determine the utility of web technology in solving remote collaboration problems; real time data acquisition can be straightforwardly embedded.
- Produce an iterated prototype that can be extended quite easily rather than undergo complete redesign; Complex hyperlink systems can be developed in man-weeks or man-months.
- Make the software product easy for domain personnel to use; the "look-and-feel" approximates that of popular business or office system software. Where possible, test components in the field with military users.
- Produce a body of re-usable object components for new extensions and modifications; tools for development, modification and maintenance are widely available."

Our use of web technologies enables rapid creation of HTML pages with 3-D content and server-side components. As previously reported, the end user realizes four distinct benefits: (1) cross-platform interoperability, (2) common browser-based user interfaces, (3) operational scalability from a single user to the worldwide enterprise, and (4) platform scalability from laptop PCs to high performance workstations [Gardner, *et al*.].

3. **A Web-Based 3-D Framework for C2**

A web-based application consists largely of *digital content* that includes linked HTML pages with embedded 3-D content, images, audio, video, and 3-D objects. These elements can be readily combined to depict portions of the C2 environment. Pages can contain behavioral scripts that enable event handling, error capture, and animations. The addition of 3-D content provides a window into a 3-D world of interest with no restrictions on size and detail. Objects can appear disappear, and re-appear dynamically. Individual properties can be controlled by an external source such as sensor input, GPS location, database updates, collaborative user interactions, etc.

In particular, multi-site 3-D collaboration can advantage the C2 decision process by portraying the same scene but from different perspectives. In addition, applications using this type of 3-D framework tend toward simplified maintenance, co-existence with legacy applications, and even enhancement of these legacy applications. For example, nearly any database that contains geometric or other visually related information can benefit greatly from this 3-D approach.

We define an experimental *3-D framework* as a development environment that supports:

1. Rapid 3-D prototyping;
2. Unlimited extensibility;
3. Reusability of prototyped objects;
4. A rich set of event-based behaviors readily attachable to objects; and
5. Simplified integration with existing data sources.

3.1 *Rapid Prototyping in 3-D*

Rapid prototypes of 3-D applications are often not given much thought because of the time required to develop realistic models of objects, code to bestow the objects with behaviors, and a user interface that permits direct interaction within the 3-D environment. These capabilities have typically been present in 3-D code libraries such as OpenGL, but the time required to develop content is excessive when only a rapid prototype is needed. Characteristically, one can think of a *Rapid Prototype* as a demonstration system, outfitted with many of the features of the ultimate system, but able to be developed in a matter of weeks or even days. Most importantly, the rapid prototype shows the target users elements of the graphical user interface (GUI) with limited functionality. The GUI reveals how to navigate the system. It can be a requirement-gathering tool useful for subdividing the problem space, and it can show possible user mechanisms to support interaction.

Our experimental framework incorporates existing libraries of objects and behaviors. The framework supports the assignment of attributes to common user interface widgets including 3-D controls. Any multimedia controls such as audio, video, and animations can also be included.

Figure 1 Multi-Function Button Control. Shown is a typical button group that facilitates introduction of new 3-D objects into the scene, toggling them, animating them, endowing them with sound, providing context sensitive help, and allowing for a submenu. The panel can be moved about the scene to minimize obscuring of other objects.

For example, a common multi-function button control, Figure 1, can be situated in the environment to provide several capabilities. The [Label] control dynamically loads/unloads a collection of thematic 3-D elements, such as a collection of maps or globes, that are referenced as an http hyperlink, another 3-D world view, an HTML page, or any other mime-type association. A second button group might load a set of nodes in a network and place them on the map. A third button group might load the network connections between the nodes. Naturally,

"Label" in is a bitmap containing some text or icon associated with the corresponding button group, such as maps, networks, or links. The other buttons enable help , toggling the visibility of these items on and off , toggling any pre-defined audio, or sound effects, associated with the objects , toggling pre-defined animations , and loading a submenu for this button group. The button permits the button group to be dragged elsewhere on the screen within the 3-D frame. All of these button groups are part of the 3-D GUI and do not move with the 3-D items they control. Each group is an instantiation of the pre-defined button-group template object with parameters to be set by the user. Therefore, the definition of these user interface elements is fast and simple.

In short, these button groups control the handling of a collection of 3-D entities. They are defined by associating URL references in simple text form. Pre-defined audio, video, and animations are constructed at design time for each object to emphasize its features or attract the user's attention to its current state. They are activated by the buttons, but can also be activated by any event occurring within the scene, or from outside sensors, database changes, *etc*.

The objects that populate the scene can be virtually anything from simple geometry, text, to complex representations of lifelike objects. They can be constructed using common CAD or Visual Modeling software and exported to VRML 97 web format, which is standard. Mapping textures to surfaces of objects is an uncomplicated way to gain perceptible realism without extraordinary modeling detail. For example, in a control room example, computers might be simple boxes or more detailed geometry with panels showing photographs of the real thing. The 3-D environment benefits from level-of-detail in the same way humans can perceive shapes and colors from a distance. A human is able to distinguish considerable levels of detail (LOD) as the distance to the object is reduced. The greatest benefit this computer representation of the LOD phenomenon is improved performance when scenes are fairly complex, perhaps consisting of ½ million or more rendered polygons.

3.2   *Event-Based Behaviors*

For the majority of 3-D applications that may already be familiar to web enthusiasts, objects have striking appearances with textures, animations, and perhaps some degree of user interaction (rotate, pan, zoom). However, very few applications probe the capabilities that an event-driven framework containing these objects can offer. Revisiting our network example once more, if a link between two routers suddenly goes down, the virtual router in the scene flashes or in some way draws attention to its state. Similarly, any link object associated with this router, which may be as simple as a line or arc between two routers, may turn red, or it may flash. In contrast, if this link is animating to portray movement of data from one router to the other, then the animation would stop and perhaps flash red. It is fair to say that nearly any event, whether emanating from a source within the scene, from user interaction, or from some outside sensor, can be represented and portrayed realistically in this environment. In fact, the event mechanisms are one of the key ingredients of the 3-D framework we discuss.

Although CAD systems and most visual modelers do an exceptionally good job of defining detail on the objects themselves, they often do very little to aid the designer in establishing event

mechanisms that give the objects realistic functional and visual behaviors. Using a modeling tool for creating behaviors on models or creating the appropriate event mechanisms using a simple text editor accomplishes this goal.

3.3 *Unlimited Extensibility*

Early in our experience building such 3-D web applications, we found that the source files can grow considerably large if they are to be realistic. Thousands of items in the scene are common. Therefore, individual static scene construction often reaches its limits long before the details are all included. Based on this experience, we adopt a dynamic loading/unloading scenario for our 3-D scenes that require objects and their behaviors to be self-contained and have their own well-defined interfaces to send and receive events. This approach results in a considerably smaller memory footprint for typical application frameworks, decreased loading times, and faster overall performance on even mediocre PC platforms. Our framework exploits this dynamic capability and results in a very manageable scenario for even large applications, since most do not require all objects and data to be visible at one time. If a large amount of data is shown in one view, the user is often confused when trying to navigate and interpret it.

We claim the framework has infinite extensibility, since there are virtually no limits to the number of URL links and pages that can be loaded/unloaded separately. The user understands that he must manage this activity to achieve higher performance and views that are interpretable and useful.

3.4 *Reusability of Objects*

One of the real benefits of object templates, such as the button groups we described earlier, is that they can be created once and instanced forever. Changes to the object template immediately filters down to the applications using it. Moreover, they can be the basis for building other objects with the same or additional behaviors (inheritance). Although the software regimen for developing these objects is analogous to object-oriented development, it has neither been formalized in this 3-D context nor enforced by any of the current web development tools, including this 3-D framework. Developers of 3-D frameworks like this would benefit enormously from formalization of this 3-D content development process and creation of tools that enforce it.

Our notion of reusability is aimed primarily at significant reduction in time that results from instancing *prototype* template objects repeatedly. They are simply 3-D black box components with a well-defined control interface. Instantiation of an object consists of declaring an instance of the prototype, defining the interface parameters and using *ROUTE*s to wire events between objects thereby permitting the object to send and receive events across its interface.

For the button group presented earlier, this interface is defined as shown in Figure 2. The *field* parameters are treated as local hidden parameters in the interface that are defined once when the instance is declared. The *eventIn* parameter declares events receivable by the interface from a similarly typed *eventOut* parameter of another object. The *eventOut* declares an event that is

generated by the instanced object and sent to objects with a compatible *eventIn* interface parameter. This same *eventOut* can be fanned out to other *eventIn* events on receiving objects.

```
PROTO loaderButton [
   field SFVec3f  buttonSizeWHD 2 0.8 0.8      # width, height, and depth of the button group
   field SFVec3f  translation      0 0 0        # location of button group on the screen
   field MFString texture []                    # bitmap image for Label button, here: Label
   field MFString url []                        # URL for group of objects to be loaded
   field SFInt32  myIndex  -1            # index number to serve as identifier for this group
   field SFString statusLine  ""         # text message displayed when hovering over Label

   field MFString helpUrl  []            # URL for Help file when ? clicked
   field MFString audioUrl  []           # URL for loading audio (wav, au, mpeg, etc),
   eventIn SFBool    isVisible           # event input to tell group the status of group visibility
   eventOut MFString url_changed         # event output reflecting current URL for the group

   eventOut SFBool   setVisibility       # event output that toggles visibility using

   eventOut SFBool   playAudio           # event output to toggle audio using

   eventOut SFBool   setBlink            # event output to toggle animation using
   eventOut SFInt32  showMyIndex         # event output to show index identifier

   eventOut SFBool   showSubMenu         # event output to toggle submenu using
   eventOut SFInt32  removeUrlIndex  # event output to unload this index item from memory
]
```

Figure 2 Button Prototype Interface. Implementation details of the prototype are not shown since instantiating this prototype does not require knowledge beyond the interface parameters. A parameter can be a *field*, an *exposedField*, an *eventIn*, or an *eventOut*. The *SF* or *MF* prefix on a type determines whether it is *single-valued* or *multiple-valued*, respectively. The # precedes optional comments on the line that describe the purpose of the parameter. For further information on VRML, please see one of the VRML references listed in the bibliography [Marrin and Campbell, 1997, Hartman and Wernecke, 1997, Ames *et al.,* 1997].

The *SFVec3f* indicates a field type consisting of a single ordered triple (3-D vector). *SFString* is another data type that indicates a single-valued string of arbitrary length. The *SF* prefix indicates a single-valued field, whereas *MF* indicates a multiple-valued field. *MFString* indicates multiple strings of arbitrary length such as *field MFString Book ["Title", "Author", "Date"]. SFInt32* indicates a single 32-bit integer. *SFBool* indicates a single-valued Boolean (true or false value). Values to the right of field names are set as default values, which are used if the field is not declared when instantiated. Parameters declared using the *MF* prefix are treated as variable-length dynamic arrays using indexing similar to most programming languages, e.g. *Book[0], Book[1],* and *Book[2]* referring to *"Title", "Author"*, and *"Date",* respectively.

These parameters operate similarly to those in a standard windowing system, but have the advantage of being associated with 3-D objects in the framework space.

Instantiating such an object can be accomplished using a statement of the form shown in Figure 3. Here a button group named *Button1* is DEFined as an instance of the *loaderButton* prototype object. *Vec*tors are triples of numbers with or without decimals, and strings are UTF-8 char-acters in double quotes in the case of *SFString*. For *MFString*, either double quotes or brackets [] with multiple double quoted items inside if there is more than one string contained in the

MFString. In a URL declaration, these multiple strings specify the order the items should be searched if the left-most URL cannot be found.

```
DEF Button1 loaderButton {
    statusLine              "Click to Load/Unload Items"
    myIndex                 1
    buttonSizeWHD           2   0.8   0.8
    translation             0  -15.0   0.8
    url                     "g-maps.wrl"
    helpUrl                 "g-help-icons.html#controlicon"
    texture                 ["images/icons/g-maps.jpg", "/demo/images/icons/g-maps.jpg"]
}
```

Figure 3 Instantiation of a Button. The details of the button implementation are hidden when a prototype is instantiated. We define essential parameters determining the button's appearance and behavior when clicked or hovered. Once instanced, it is a simple matter to change the location parameters to see the effect on the screen. See Figure 2 for an explanation of their meanings. Note also that the default *field* values in Figure 2 are used if they are not specified in this instance.

Although this may seem a bit technical at the outset, its value becomes evident when instantiating several of these button groups as shown in the screen shot in Figure 4. The buttons appear in light blue above the 3-D controls in light blue that are part of the 3-D viewer plugin for Internet Explorer. This 3-D viewer shown is the newly released Pivoron Player plug-in from Nexternet, Inc. [See Section 5 for software details], which is an updated and enhanced revision of the very popular CosmoPlayer plug-in from Cosmo Software, Inc. The decoration at the top of the screen is part of Microsoft Internet Explorer (MSIE). The NetSpace logo in the upper right corner is an instantiation of yet another prototype.

Clicking on the maps button ![Maps] loads a variety of maps and a submenu to toggle between them. These maps may be unloaded from memory by clicking the maps button ![Maps] again. The visibility button ![icon] turns the maps on and off, while still retaining them in memory.

The button groups in the ![Maps] row in Figure 4 were instantiated using the declarations shown in Figure 5 below. No audio ![icon] or animations ![icon] are defined for the buttons in this example. The *translation* parameter locates the button group on the screen in a "heads up" context (HUD). For our example, *0 -15 0* locates the group at *x = 0* (on screen left), *y = -15* (15 down on *y*), and *z = 0* (at *0* depth into the screen). Each group is placed along increments of *5* along *x*, at the same *y* of *-15*, and the same depth of *z = 0*. Placing these groups at the top of the screen would mean changing the *y* value to *+15* for each group. Here, *y = 0* centers the icons horizontally at mid-screen. Keep in mind that changing *z* will move the icon further into the screen or closer to you. If too close, it may disappear since it is clipped from the viewing frustum of the screen. The *buttonSizeWHD* parameter determines the total width, height, and depth of the button group along *x* and *y*, where *z* represents the button depth along *z*.
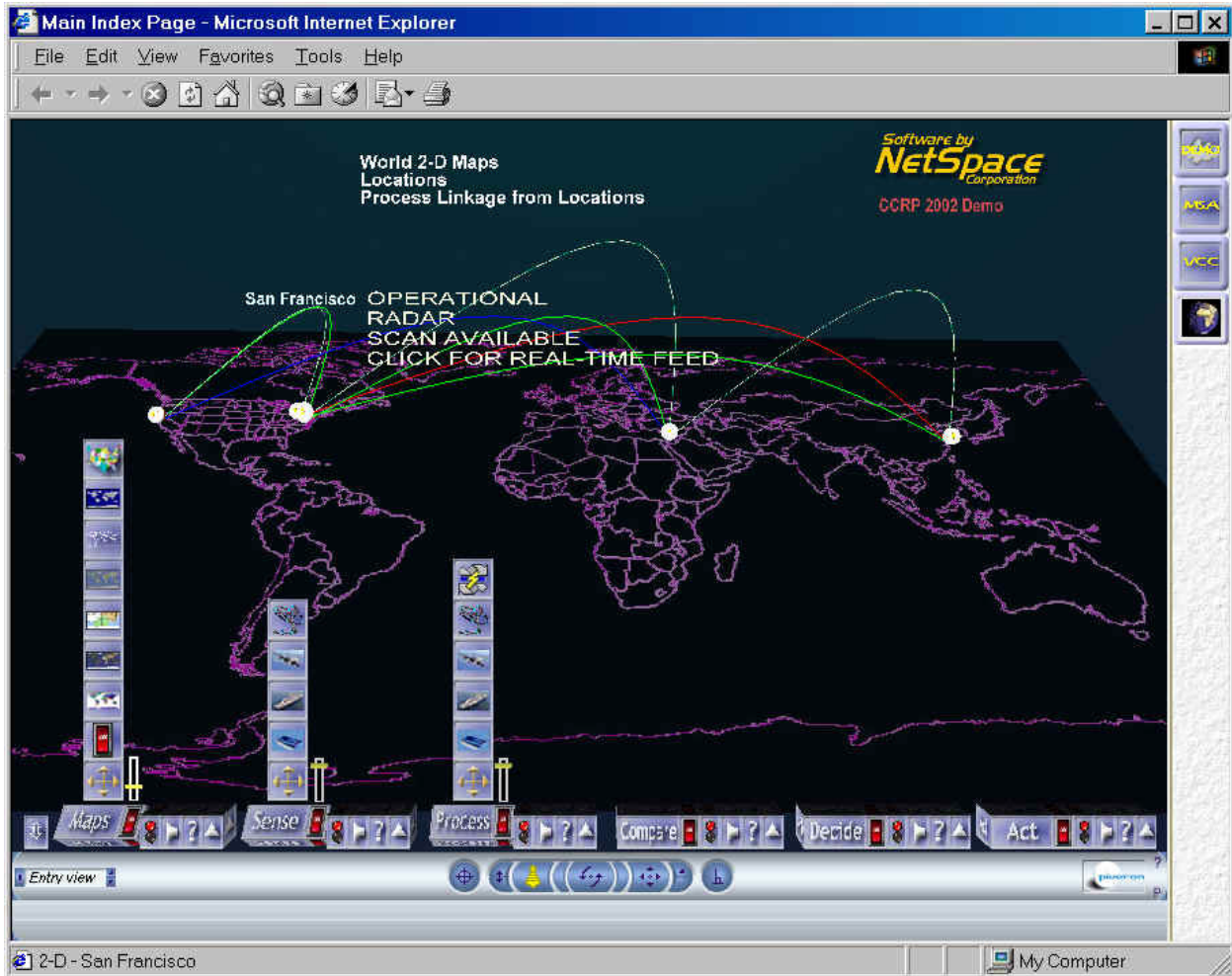
Figure 4 Framework Showing Buttons for C2 Processes. The panel of icons showing Maps , Sense , Process , Compare ,
Decide , and Act specify the choice for Lawson's C2 processes [Lawson, 1979]. Below this row of icons is the
panel of icons for controlling the scene in 3-D: **Go** , **Slide** , **Tilt** , **Rotate** , **Pan** ,
Zoom , **Undo Move** , and **Seek** .

3.5 *An Example… Simplified Legacy Integration*

In one of our development efforts, we were asked to synthesize collections of disparate legacy
network data contained in a plethora of files and databases. We decided to portray the merged
results by showing a 3-D representation. The justification lay in the fact that we had no idea how
complex the graphic rendering would become, and portraying it as a set of 2-D windows would
soon fall short of our objective… to show as much rendered data objects "through a portal
window into the 3-D space" as possible. Equally important, we needed to present the results to a
non-technical audience.

```
DEF Button1 loaderButton {       statusLine "Click to Load/Unload Map Items"
  myIndex      1
  buttonSizeWHD 2 0.8 0.8
  translation 0 -15 0
  url "g-maps.wrl"
  helpUrl "g-help-icons.html#controlicon"
  texture ["images/icons/g-maps.jpg", "/demo/images/icons/g-maps.jpg"]  }
DEF Button2 loaderButton {       statusLine "Click to Load/Unload Sensing Items"
  buttonSizeWHD 2 0.8 0.8
  myIndex      2
  translation 5 -15 0
  url "g-demonodes2d.wrl"
  helpUrl "g-help-icons.html#controlicon"
  texture ["images/icons/g-sense.jpg", "/demo/images/icons/g-sense.jpg"]  }
DEF Button3 loaderButton {       statusLine "Click to Load/Unload Processing Items"
  buttonSizeWHD 2 0.8 0.8
  myIndex      3
  translation 10 -15 0
  url "g-demolinks2d.wrl"
  helpUrl "g-help-icons.html#controlicon"
  texture ["images/icons/g-process.jpg", "/demo/images/icons/g-process.jpg"]  }
DEF Button4 loaderButton {       statusLine "Click to Load/Unload Comparison Items"
  buttonSizeWHD 2 0.8 0.8
  myIndex      4
  translation 15 -15 0
  url "g-compare2d.wrl"
  helpUrl "g-help-icons.html#controlicon"
  texture ["images/icons/g-compare.jpg", "/demo/images/icons/g-compare.jpg"]  }
DEF Button5 loaderButton {       statusLine "Click to Load/Unload Decision Items"
  buttonSizeWHD 2 0.8 0.8
  myIndex      5
  translation 20 -15 0
  url "g-decide2d.wrl"
  helpUrl "g-help-icons.html#controlicon"
  texture ["images/icons/g-decide.jpg", "/demo/images/icons/g-decide.jpg"]  }
DEF Button6 loaderButton {       statusLine "Click to Load/Unload Action Items"
  buttonSizeWHD 2 0.8 0.8
  myIndex      6
  translation 25 -15 0
  url "g-act2d.wrl"
  helpUrl "g-help-icons.html#controlicon"
  texture ["images/icons/g-act.jpg", "/demo/images/icons/g-act.jpg"]  }
```

Figure 5 Instantiations of Multiple Buttons. Here six separate button groups are instanced horizontally at the bottom of the screen ($y = -15$) in the *translation* parameter. The *x*-values for each are chosen by trial and error to center the icon group along the bottom. The translation *z*-value for each button group is set to the default value *0* (recommended).
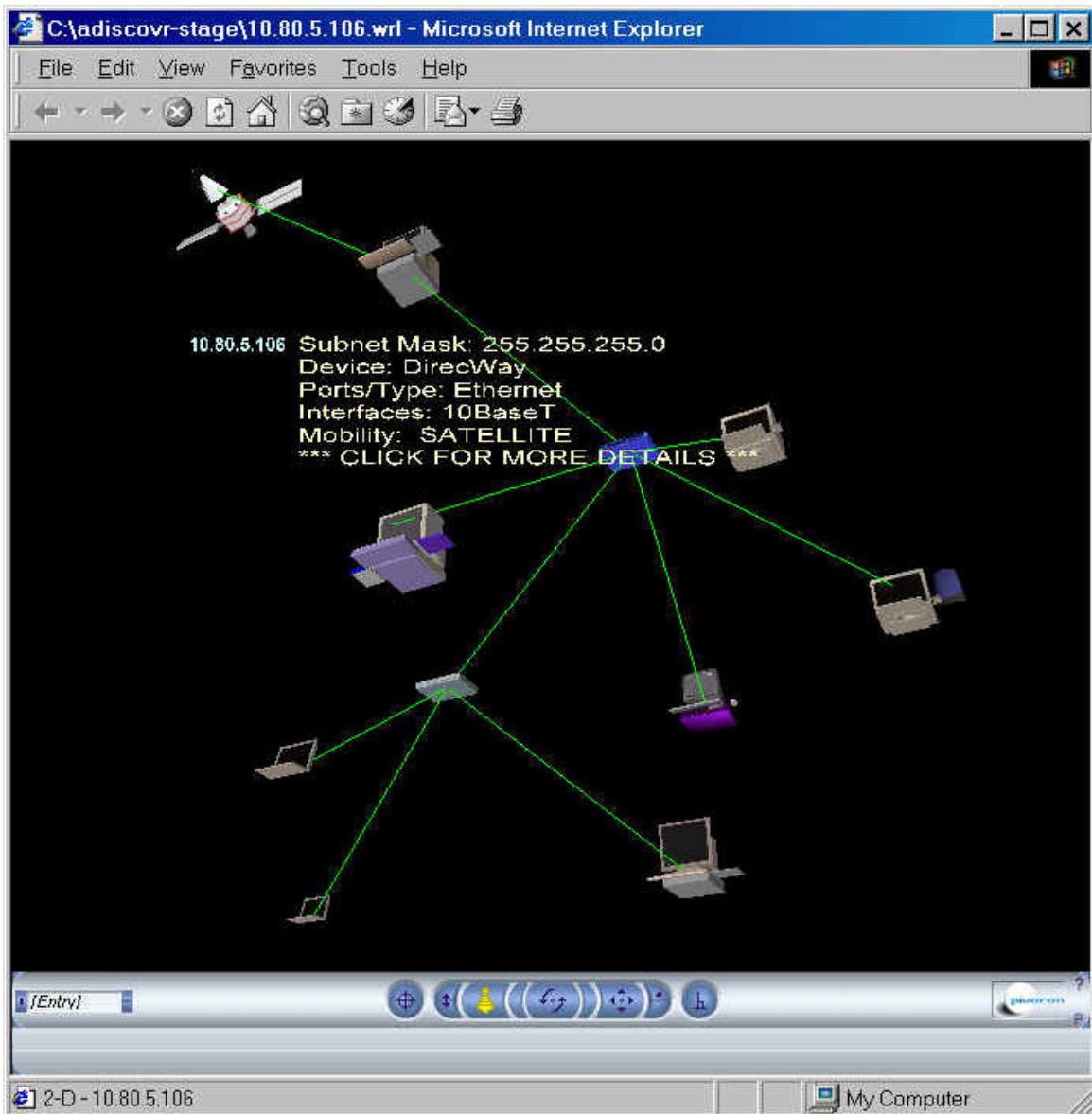
Figure 6. A 3-D representation of a network. Here we show a discovered network at a NetSpace office. The starting node (DirecWay satellite node) was used to seed the process. Other node connectivity was determined by instancing a *nodeWithLink* prototype showing the node's characteristics along with its connectivity to a parent node. Hubs show connectivity to a parent node (another hub in most cases) and all children nodes since they are data communication equipment (DCE). A node with no children is data terminal equipment (DTE). Hovering over any node or link shows its general characteristics. Clicking on a node or link shows another window with greater detail about that node or link. The location of objects in the descending tree used a simple algorithm that fanned out the children along a circle, one unit lower on *y* than itself.

The data were contained in text files and SQL databases. During the past several years, these vast tables of data were consolidated into extensive technical reports that required considerable time to read, parse, and assimilate on a weekly basis. We defined an ambitious goal to extract significant network performance and network availability information from these report sources and present it in very simplified form using a 3-D approach. The result was a comprehensible global view with drill-down capabilities to exploit more and more detail in the data required by

various levels from top-level management to technical personnel. Recently we have added capability to permit users to select arbitrary data sets and to dynamically discover network resources using pings and SNMP queries to build a 3-D representation of nodes and links on the fly. Although it can be time-consuming to discover nodes and links for large networks, it is impressive to see several hundred or even thousands of nodes and links rendered dynamically. A simplified version of this capability is shown in Figure 6 applied to a NetSpace office network.

Together, the collection of resources required to build a dynamic 3-D interface using current web technologies constitutes a viable *3-D framework* for development in a number of application domains where data complexity is an issue and rendering of the data lends itself to 3-D.

4. **Lawson's C2 Process Model**

The work we present here reflects a transfer of the *3-D framework* notions to the C2 world. The remainder of this paper addresses the use of our prototype 3-D framework to support Lawson's Model cited earlier as taken from Allard's *Command and Control and the Common Defense* as shown in Figure 7, *Lawson's Command and Control Model* [Allard, 1966]. We have extended the work presented earlier that suggested the use of Lawson's Model [Lawson, 1979].
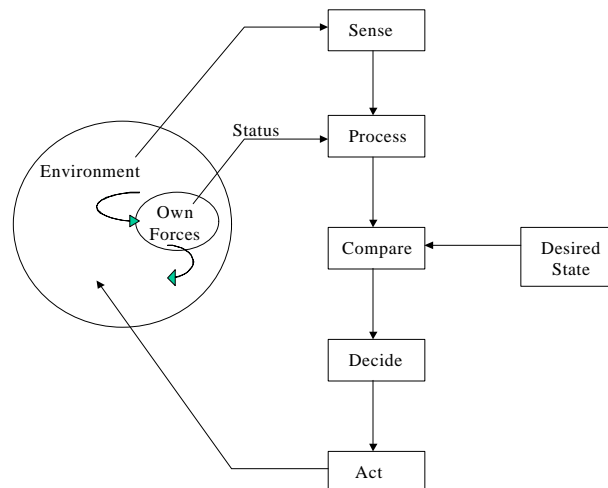
Source: AFCEA Press

Figure 7 Lawson's Command and Control Model [Lawson, 1979].

The 3-D framework is used to show how the five major C2 activities could be incorporated and made to respond to drill-down requests using objects that are common in C2.   Figure 4 shows how a high-level view of these activities could be presented. Naturally, the drill-down process expects to link to information that is presentable in a web browser context.

We have added a Maps capability to the displays where a variety of maps may be toggled. Data registered on a map is often meaningful to the user. Although not shown here, we have the option of displaying all data on a 3-D globe complete with registration at GPS coordinates. This is quite valuable for representing space, ground, and undersea assets. Using the event-handling capability of this 3-D framework, nearly any data can be used to drive the actions of the 3-D

objects to portray a realistic scene in near real time. These features provide a compelling way to use this framework to represent Lawson's five C2 processes. In Figure 4, the objects have multiple attributes and are used as hyperlinks to drill down to further detail. Hovering the cursor over a node or link will display general information about the object. These virtual objects could easily represent military assets shown with a realistic 3-D icon. In any event, these visual icons are cues to the user to probe for further details.

Experience has shown that the more detail and sizeable the information, the greater the utility this 3-D framework has to account for the objects and portray their characteristics and state. Because *zoom*, *pan*, and *rotate* within an infinite digital space are supported along with additional dynamic icons, one can include many additional objects representing a C2 process space. Unlike the desktop metaphor common in 2-D applications of the past, the 3-D window into the domain space gives rise to an information space appropriate as a C2 space metaphor.

5. Software Requirements for the 3-D Framework

Web browsers are freely downloadable and upgradeable from these sources.

> Microsoft's Internet Explorer for PCs and MACs at http://www.microsoft.com
> Netscape's Navigator/Communicator at http://home.netscape.com

3-D browser plug-ins and add-ons are freely available for Windows PCs from these sites.

> Nexternet's Pivoron Player VRML plug-in for IE Explorer 4.0+ at
> http://www.nexternet.com/download/
> Nexternet's Pivoron Player VRML plug-in for Netscape Navigator at
> http://www.nexternet.com/download/
> CAI's CosmoPlayer 2.1.1 VRML add-on for IE Explorer 4.0+ at
> http://www.cosmosoftware.com
> CAI's CosmoPlayer 2.1.1 VRML plug-in for Netscape Navigator at
> http://www.cosmosoftware.com
> VRML Tutorial (Excellent). http://www.virtualrealms.com.au/vrml/tute01/tutorial.htm
> VRMLPad 1.3 Context-Sensitive Editor, Parallelgraphics, Inc.
> http://www.parallelgraphics.com/products/vrmlpad/

Additional flexibility with dynamic content can be provided through the use of the Extensible Markup Language (XML) that significantly enhances HTML-like tagging for broad use. Our current work centers on providing a capability to directly import XML data into the 3-D framework. Coupled with the next generation of web 3-D known as X3D, XML promises to ease the development of data-driven 3-D frameworks even more. More information on XML and X3D is available from these sites: http://www.xml.com, and http://www.web3d.org.

6. Conclusion

The use of 3-D frameworks in a C2 environment is still in its infancy. In this paper, we have discussed a candidate 3-D framework in the context of C2. We have also given examples

extending previous work. We have built bodies of digital objects and event scenarios that facilitate rapid population and extension of these frameworks. This technology has the potential to deliver a unifying and simplifying effect on decision making within C2. We anticipate further work will continue to build out this C2 content toward a comprehensive and useable decision-support application.

7. **References**

[Allard, 1966] Allard, Kenneth. "Command, Control, and the Common Defense, 2nd Ed." *National Defense University, Institute for National Strategic Studies,* October 1966, pp154-163.

[Boyd, 1984] Boyd, John R. "Organic Design for Command and Control," briefing paper, March 1984, pp5, 32-35.

[Gardner *et al*., 1999] Gardner, S., Callihan, H., Balash, J. and Saverino, M. "Exploitation of Web Technologies for C2," *5th ICCRTS*, 1999.

[Lawson, 1979] Lawson, Joel S. "Naval Tactical C3 Architecture, 1985-1995." *Signal* 33:10 (Aug 1979), pp71-72.

[Orr, 1983] Orr, Maj. George E. *Combat Operations C3I: Fundamentals and Interactions* (Maxwell Air Force Base, AL: Air Univ. Press, 1983), pp 23-27.

[Callihan and Balash, 2001] Callihan, H. D. and Balash, J. A. "A 3-D Framework for C2 Based on Web Technologies," *2001 6th ICCRTS*, Annapolis MD, June 2001.

[Marrin and Campbell, 1997] Marrin, C. and Campbell, B. *Teach Yourself VRML in 21 Days*. Sams.net, Indianapolis, IN, 1997.

[Hartman and Wernecke, 1997] Hartman, J. and Wernecke, J. *The VRML 2.0 Handbook*. Addison-Wesley Developers Press, Reading, MA, July, 1997.

[Ames *et al*., 1977] Ames, A., Nadeau, D. and Moreland, J. *The VRML 2.0 Sourcebook*. Wiley, New York, 1997.