

# A 3-D Framework for C2 Based on Web Technologies

**Hubert D. Callihan**

**John A. Balash**

NetSpace Corporation

9640 Downing Place

North Huntingdon, PA 15642

(724) 861-8228 Voice

(724) 861-0999 Fax

[admin@netspacecorp.com](mailto:admin@netspacecorp.com)

## Abstract

Web applications are proliferating in the commercial sector due in large part to developmental frameworks that promote rapid web site creation, simpler maintenance, and complete back-end database integration. While 3-D on the web has enjoyed a modest degree of success on the web, it is typically relegated to "dynamic picture" status where scenes are presented with the user in control of viewing angles and other camera-like actions. Few applications have shown the full capability of web 3-D as a basis for event handling whether user-induced actions or externally supplied by remote sensors. In other words, objects (including avatars) depicted in the scene act and react based on actual events, real time or otherwise. C2 offers a natural arena for collaboration in 3-D where interactions among users, remote events, and object control are very realistic. A web framework for developing such C2 applications will be presented along with some actual applications based on ideas filtered from our current work in network communications. Major issues related to the use and relevance of 3-D technologies for C2 such as *Remote Monitoring and Control, Collaboration, and 3-D Visual Frameworks* will be discussed.

## 1. Introduction

Recent advances in commercial Information Technology (IT) have produced a variety of *web-based technologies*, which hold considerable promise for domains of interest where dispensing of information is rapid, reliable, and timely, while giving the recipient the option of several degrees of interaction with the information in near real time. Naturally, the digital communications network is the key transport mechanism to deliver this information. In the commercial sector, it is the Internet with its standard TCP/IP protocols. In the military sector, it is similar but with added capability to incorporate encryption. Server and end-user devices are commercially available requiring little or no special customization features. Our interest has been in 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 covering a wide range of situations from command level to the war fighter.

Command and Control has been widely studied domain of interest that shares many of the same general features as current commercial systems supporting varied decision-making capabilities. These 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. These principles have been broadly discussed from a military perspective as well by Boyd [2], Orr [5], Lawson [4] and others. From this perspective, Lawson's model seemed to lend itself to visual representation and subsequent drill-down through the phases of his C2 process... [1] sense the environment, [2] process using the status of our own forces, [3] compare using a desired state, [4] decide, and [5] act using our own forces in the environment. We use Lawson's model to demonstrate the features of a 3-D framework to support C2.

## 2. Approach

Highlighted below are several major considerations that are important for web-centric C2 application development considered by Gardner, et al [3].

- "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."

Furthermore, we have used web technologies in concert with the development software to enable rapid creation of web pages and 3-D environments. In agreement with the work of Gardner, et al, we have realized several benefits to the end user: cross-platform interoperability, common browser-based user interfaces, operational scalability from a single user to the worldwide enterprise, and platform scalability from laptop PCs to high performance workstations.

## 3. A 3-D Framework for C2

Web-based development consists of *digital content*, which includes HTML-linked pages, images, audio, video, and 3-D objects having varying degrees of detail from simple color models to photo-realistic objects possessing their own behaviors and lifelike features. These elements can be easily combined to depict portions of the C2 environment. Pages can contain behavioral scripts to enable event handling, error capture, and animations. This custom behavioral code is included and does not expose itself except to present buttons and otherwise interact with the objects on the page. The 3-D approach provides a window into the world of interest with no restrictions on size and detail. Objects can come and go dynamically within the sphere of

interest. Their properties can be influenced by some external source such as sensor input, GPS location, database updates, etc.

Multi-site collaboration in a C2 decision support context can benefit from many of these attributes. 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.

We define a 3-D framework for these purposes as a development environment that supports rapid prototyping, unlimited extensibility, reusability of objects, a rich set of event-based behaviors, and simplified integration with existing data sources.



### 3.1 *Rapid Prototyping*





Rapid prototypes of a 3-D application are often not given consideration because of the time required to develop realistic models of objects, code to endow the objects with behaviors, and a user interface that permits direct interaction within the 3-D environment. These capabilities have always been present in 3-D code libraries such as OpenGL, but the time required to develop content is prohibitive when a rapid prototype is needed. We think of a rapid prototype as a demonstration system outfitted with many of the features of the eventual system but able to be developed in a matter of weeks or even days. Most importantly, the rapid prototype shows the target users what a system will look like and how it will be navigated. It often functions as a requirement-gathering tool for subdividing the problem space and showing possible user mechanisms to support interaction.

Our framework permits the prototype developer to incorporate existing libraries of objects and behaviors and assign attributes to common user interface widgets including 3-D controls as well as many multimedia controls such as audio, video, and animations.



Figure 1 Multi-Function Button Control

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 worldview, 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" would be 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 **Error! Unknown switch argument.** 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 within the 3-D frame. All of these button groups are part of a head-mounted display effect (HUD) 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, to complex representations of lifelike objects. They can be constructed using common CAD or Visual Modeling software and exported to the standard VRML 97 web format. Mapping textures to surfaces on objects is an easy way to gain apparent realism without extraordinary modeling detail. For example, in a network example, routers might be simple boxes or more detailed geometry complete with panels imported from 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 but are able to distinguish considerable detail as the distance between them is reduced. The greatest benefit this computer representation of the LOD phenomenon is increased performance.

### 3.2 *Event-Based Behaviors*

For most 3-D applications familiar to web enthusiasts, objects have striking appearances with textures, animations, and perhaps some degree of user interaction (rotate, pan, zoom). However, 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 link object, which may be as simple as a line between two routers, may turn red, or it may flash. If such a link showed an animation portraying the 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, some user interaction, or some outside influence, 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 behaviors. This is accomplished by using a modeling tool for creating behaviors on models.

### 3.3 *Unlimited Extensibility*

Early in our experience building these 3-D web applications, we found that they 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 be visible at all times. In fact, when much data is shown in one view, the user is often quite confused when trying to 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 having object templates, such as the button groups we described earlier with defined interfaces for an application, and objects that can be instantiated from a template, is that they are created once and used forever. Any changes to the object template will immediately trickle down to the applications that are using it. Moreover, they can be the basis for building other objects (inheritance) with the same or additional behaviors. Although the software regimen for developing these objects is akin to object-oriented development, it has neither been formalized in this 3-D context nor enforced by any of the web development tools, including this 3-D framework. Developers of frameworks such as this would benefit enormously from such formalization activity and tool sets that enforce a content development process.

Our notion of reusability for this discussion is aimed primarily at the significant reduction in time that results from using *prototype* objects over and over that have well-defined parameter interfaces to accept and produce parametric elements and event mechanisms. These are simply 3-D black box objects with a well-defined control interface. Instantiation of these objects consists of declaring an instance of the prototype, defining the interface parameters and connecting any event mechanisms in the interface that permit the object to send and receive events.

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, *eventIn* declares events receivable by the interface, and *eventOut* declares events that are generated internally by the object and receivable by objects external to the prototype. The *SFVec3f* indicates a field type consisting of a single ordered triple (3-D vector). *SFString* indicates a single-valued string of arbitrary length. *MFString* indicates multiple strings of arbitrary length. *SFInt32* indicates a 32-

bit integer. *SFBool* indicates a single-valued Boolean (true or false). Values to the right of field names are default values, which are used if the field is not declared when instantiated.

```

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 
  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 to tell group the status of group visibility
  eventOut MFString url_changed    # event reflecting current URL for the group
  eventOut SFBool  setVisibility   # toggles visibility of the group using 
  eventOut SFBool  playAudio       # event to toggle audio using 
  eventOut SFBool  setBlink        # event to toggle animation using 
  eventOut SFInt32 showMyIndex     # event to show index identifier
  eventOut SFBool  showSubMenu     # event to toggle submenu using 
  eventOut SFInt32 removeUrlIndex # event to unload this index item from memory
]

```

Figure 2 Button Prototype Interface. Details of the prototype are contained after the declaration and are not included here since instantiating this prototype does not require knowledge beyond the interface parameters.

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 instantiation of the *loaderButton* prototype object. Vectors are triples of numbers with or without decimals, and strings are 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, these multiple strings specify the order the items should be searched if the leftmost 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 buttons are hidden when a prototype is instantiated.

Although this may seem a bit technical at the outset, the value becomes evident when instantiating several of these button groups as shown in the screen shot in Figure 4. The buttons appear in gold above the 3-D controls in gray, which are part of the 3-D viewer plugin for Internet Explorer. The decoration at the top of the view is part of Explorer. The logo in the upper right is an instantiation of another prototype.

Clicking on the maps button loads a variety of maps and a submenu to toggle between them. These maps may be unloaded from memory by clicking the maps button again. The visibility button turns the maps on and off button still retains them in memory.

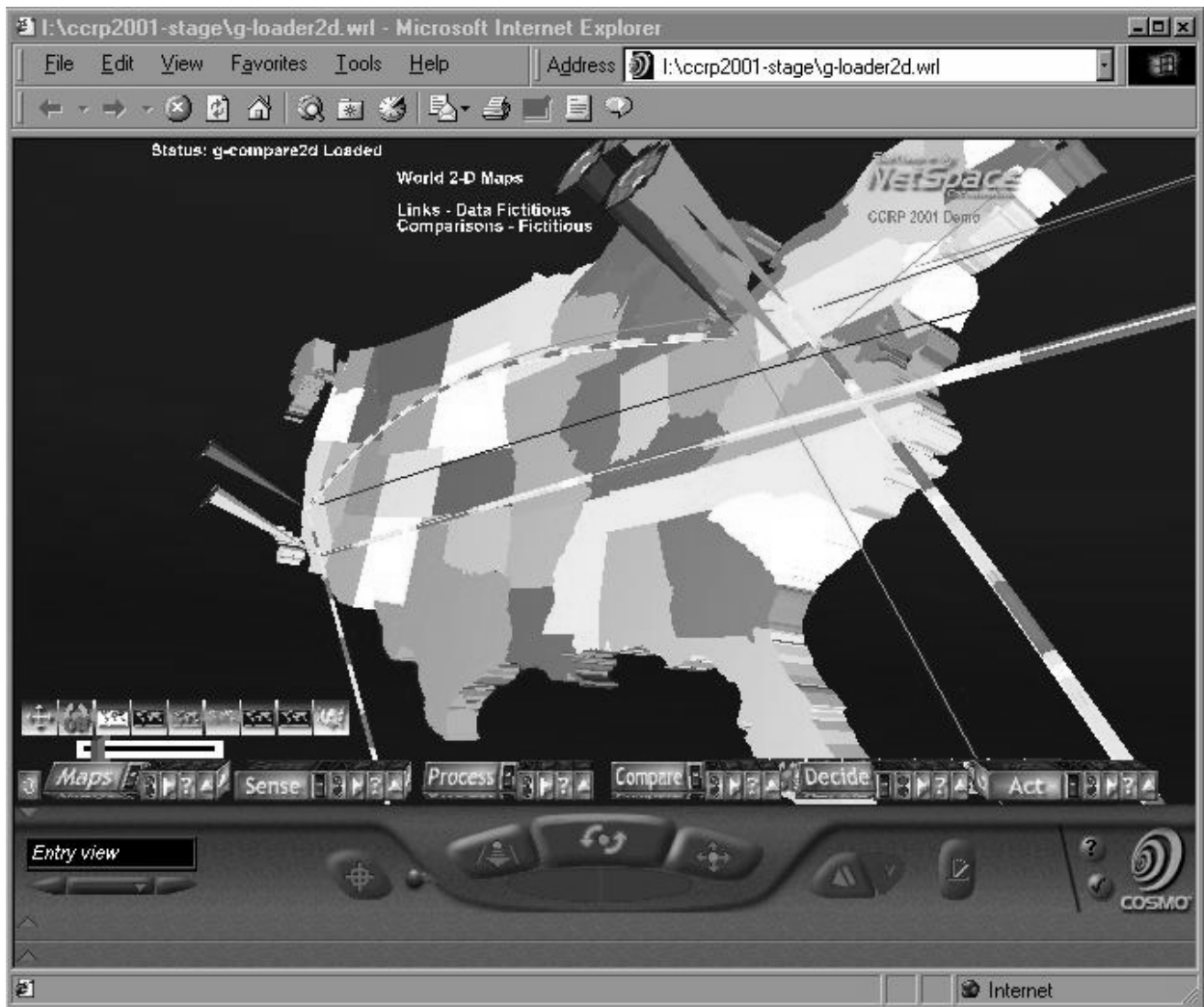


Figure 4 Framework Showing Buttons for C2 Processes

These buttons were instantiated using the declarations shown in Figure 5. Note that no audio or animations are defined for these buttons. The translation parameter locates the button group on the screen in the HUD context. For example, 0 -15 0, locates the group at x=0 (on screen left),

y=-15 (15 units 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. The `buttonSizeWHD` determines the span distance covered by the button group along x and y, where z represents the button depth along z.

```

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



### 3.5 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 by showing a 3-D representation and present it to a non-technical audience. The data were contained in text files and SQL databases. During the past, these vast tables of data resulted in extensive technical reports that required considerable time to read and assimilate on a weekly basis. We defined an ambitious goal to extract significant performance and availability information from these data 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.

## 4. A C2 Process Model

The work we present here reflects a transfer of these ideas to the C2 world. The remainder of this paper addresses the use of a 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 6, Lawson's Command and Control Model [1].

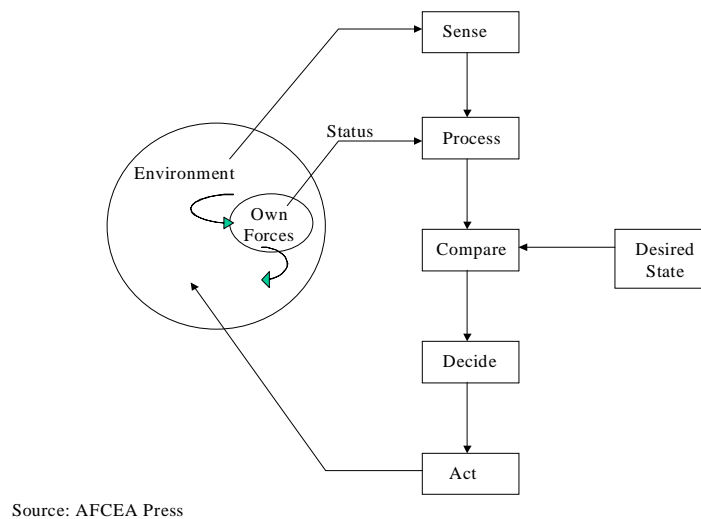


Figure 6 Lawson's Command and Control Model

We decided to represent this model in the 3-D framework and show how the five major C2 activities could be incorporated and made to respond to drill-down requests using objects 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 toggle on and off. Often the C2 environment requires that data be registered on a map to be meaningful. 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 very useful for representing space, ground, and undersea assets. With the event-handling capability of this 3-D environment, 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 colored pins representing location of some meaningful object have multiple attributes and can be used as the hyperlink to drill down to further detail. These could easily be military assets that could be shown with a realistic 3-D icon. In any event, these visual icons are cues to the user to probe further.

Experience has shown that the more detail and voluminous the 3-D environment becomes, the greater the utility of this framework to account for the objects and provide a perspective in a global sense. Because one can zoom, pan, and rotate within an infinite digital space, there is no limit to the extent one can include additional iconic 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 a battle space or a C2 space metaphor.

## 5. Software Requirements

Web browsers are freely downloadable and upgradable 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>

Sun's HotJava browser from <http://www.javasoft.com>

Complement for IE Explorer called NeoPlanet at <http://www.neoplanet.com>

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

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>

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. More information is available from this site. <http://www.xml.com>

Commercial product offerings for building specialized web applications are available from these sites.

3-D applications integrated into HTML web pages at <http://www.sgi.com>

## 6. Summary

The use of 3-D frameworks in a C2 environment is in its infancy. In this paper, we have discussed a candidate framework in the context of C2. In the near future, we expect bodies of digital objects and event scenarios to permit rapid population of these frameworks. This technology has the potential to have a unifying and simplifying effect on decision making within C2.

## 7. References

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

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

[3] Gardner, et al. "Exploitation of Web Technologies for C2," 5<sup>th</sup> ICCRTS, 1999.

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

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