A Decade of Prototype Displays

Daniel M. Sunday and Christopher J. Duhon

We describe an evolution of display prototypes that APL developed for Aegis and the Cooperative Engagement Capability. The work spanned more than a decade from the late 1980s to 2000. Many of the ideas and software produced were incorporated into the Navy's most significant combat system displays (Aegis Display System, Advanced Combat Direction System, and Ship Self-Defense System) as well as numerous engineering displays. This article recounts the work and describes some of the display features along with significant aspects of the display architecture.

INTRODUCTION

From the late 1980s through 2000, APL developed prototype combat system displays. These prototypes have now transitioned to industry and are being used as the foundation for the most significant Navy production combat display systems, including Lockheed Martin's Aegis Display System (ADS), Raytheon's Advanced Combat Direction System (ACDS) command station, Raytheon's Ship Self-Defense System (SSDS) display, and the Naval Air Systems Command's E-2C Advanced Control Indicator Set (ACIS) display. In all cases, APL provided a generic display engine along with a software toolkit of display components that was collectively referred to as the Common Display Kernel (CDK). Vendors of platform-specific display systems then tailored and enhanced CDK to meet their mission requirements. In addition to these production displays, CDK was used in numerous engineering and simulation displays, such as the Battle Force Tactical Trainer and APL's Cooperative Engagement Capability (CEC) engineering displays.

In this article we first review the evolution of the APL prototype displays from the mid-1980s, then describe the functionality provided in the final builds of CDK, and give a short overview of its architecture.

BACKGROUND

APL's display prototype work was initiated as the Command Support At-Sea Experiment (CS@SE), which evolved through four phases. In 1987, the kickoff Phase I was led by Jennings Willey as a follow-on to his seminal paper[1] on advanced graphics techniques. Phase I primarily investigated the use of color and area fill for information discrimination in Navy combat displays. The Phase I prototypes underwent at-sea testing aboard the Aegis cruisers USS *Ticonderoga* (CG 47) and USS *Yorktown* (CG 48).

In 1988–1989, Phase II, led by Dave Buscher, extended the experiment to include numerous new features (e.g., enhanced geographic displays) and combined real-time sensor and non-real-time over-the-horizon tracks in the same display.[2] The Phase II prototypes were evaluated aboard the Aegis cruiser USS *Leyte Gulf* (CG 55) and the carrier USS *America* (CV 66). The Phase II display also had two important spin-offs: the first auto-identification (auto-ID) displays being developed by Roger Sumey and a first CEC engineering display developed by John Peden and Dan Sunday.

In 1989–1991, Phase III, led by Conrad Grant, further investigated the fusion of real-time sensor and non-real-time over-the-horizon track information with a prototype network track server that implemented a multi-hypothesis correlation algorithm.[3] The Phase III prototypes were evaluated aboard USS *Leyte Gulf* and the carrier USS *Roosevelt* (CVN 71), and were used on those ships during Desert Storm in the Persian Gulf.

In 1991–1993, the final Phase IV, led by Dan Sunday, further evolved CS@SE into a network-centric system with multiple operator stations and multiple servers for track fusion, Defense Mapping Agency and National Imagery and Mapping Agency (NIMA) geographic databases, and a network archive-playback server (NAPS) of mission-recorded scenarios. New ideas for the Human–Machine Interface (HMI) Graphical User Interface (GUI) were explored using X Windows and the Motif GUI toolkit, and CEC track data became another real-time input. The Phase IV prototypes were evaluated aboard the Aegis cruisers USS *Leyte Gulf* and USS *Antietam* (CG 54).

Following Phase IV of the CS@SE, a short period of redirection occurred for APL's display prototype work: a CEC-specific engineering display project split off as a separate entity. Briefly there was a Phase V led by Conrad Grant, but it quickly changed into a Baseline 1 Backfit (B1B) project for Aegis cruisers. Within a few years, B1B was renamed the Combat Display Control System (CDCS) project that transitioned to the Baseline 5 Aegis cruisers USS *Anzio* (CG 68) and USS *Cape St. George* (CG 71) participating in CEC at-sea testing. At this point, the common software being used by the CDCS and the CEC displays was packaged as CDK, which became an independent project. CDK adapted some architectural elements from the Battle Group Anti-Air Warfare Coordination Program's Force Threat Evaluation and Weapons Assignment displays, a lineage it shares with the Area Air Defense Coordinator (AADC), to be described in the next issue of the *Digest*. Lead personnel for this multiplicity of projects over the following years included Conrad Grant, Dan Sunday, Eric Conn, Don Davis, Chris Duhon, Paul McMullin, and Dave Nesbitt.

In the mid-1990s, there was another spin-off from the CEC display to develop a production display system for the E-2C Hawkeye 2000 upgrade (which included CEC). At this time, Duhon was the lead for the CEC displays aboard all surface ships, Sunday led the E-2C airborne displays, McMullin led the CDCS effort, and Davis was the CDK lead. Two of these prototype systems, CDCS and CEC, were aboard the Aegis cruisers USS *Cape St. George* and USS *Anzio* simultaneously. Figure 1 shows the Combat Information Center (CIC) of *Cape St. George* with these displays as well as other ADSs.

Throughout the 1990s, the CEC and CDCS displays provided CEC engineering and operational support at many major Navy exercises: Crown Mountain (St. Thomas), Atlantic Fleet Weapons Training Facility (Puerto Rico), Mountain Top (Hawaii), and dozens of others near Norfolk, Virginia, culminating in the CEC Operational Evaluation in 2001. Many new display ideas were introduced to support CEC integration, including color-coding of cooperating units (CUs) and their sensors, and using these colors to make

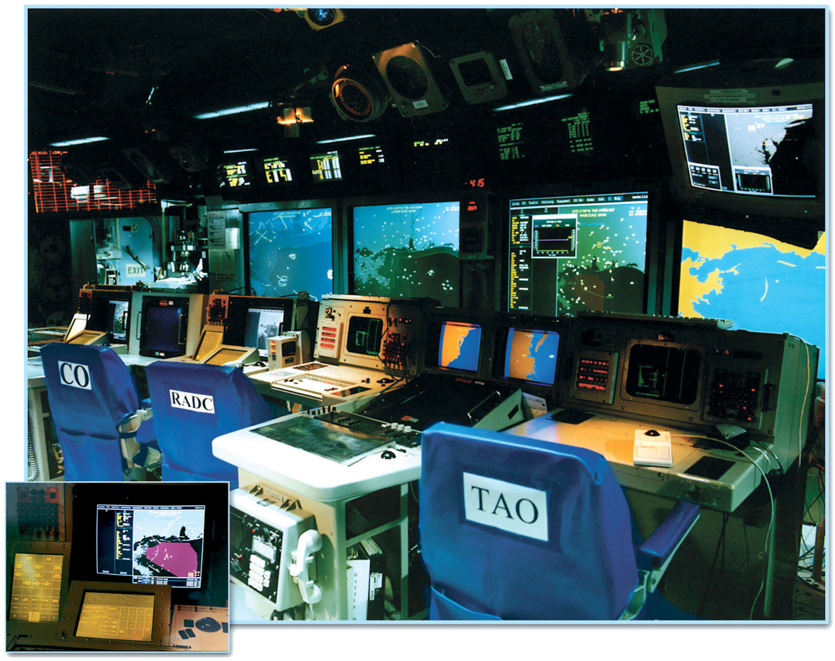


Figure 1. The Combat Information Center aboard USS *Cape St. George*. Two APL prototype displays are shown alongside the Baseline 5 ADS. A CEC display can be seen on both a 21-in. monitor and on large-screen display #3 (second from the right). The CDCS occupies large-screen displays #1 and #2 as well as the Commanding Officer's console (inset). Other displays are ADSs.

associations in the displays. In particular, engagements and their providers were color-coded by CU, as were colored-dot history trails of CU sensor events behind each track. In addition, new tactical decision aids (TDAs), such as the Radar Masking tool and the CU Position Planner, were developed for the CEC displays. These displays were aboard USS *Eisenhower* (CVN 69) during its 1994 Mediterranean deployment, which also included the CDCS aboard the two Aegis cruisers.

At APL, CDK software development continued to provide across-the-board support up to 1999. The Navy then directed the Laboratory to transition CDK to industry for development of the production combat systems: to Lockheed Martin for Aegis Baseline 6 (Fig. 2), and to Raytheon for SSDS Mk 2. After that, the continuing display work at APL focused solely on the CEC and E-2C displays, both of which are also being transitioned to industry.

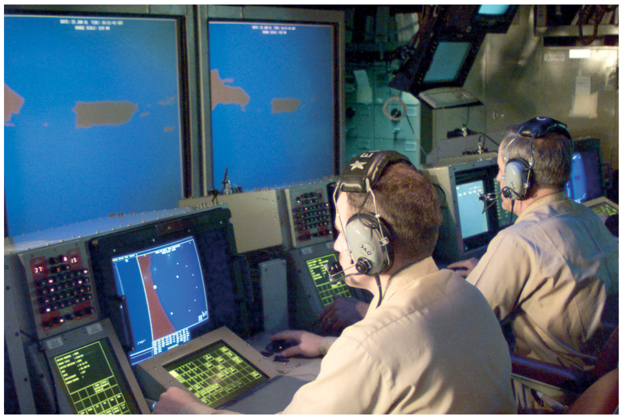


Figure 2. The Combat Information Center aboard USS *Vicksburg* (CG 69), an Aegis Baseline 6 Phase I cruiser. Shown is the ADS, which is directly based on APL's CDK Version 3.0. All ADS consoles and large-screen displays use CDK software and also incorporate many CEC display functions.

Although active development of these prototypes ended at APL, the results of this work continue in both the Navy combat systems they helped define and numerous engineering display systems within and outside the Laboratory. Versions of the CEC, CDCS, and CDK software were distributed to the following Navy commands and organizations: Naval Research and Development, San Diego; Naval Surface Warfare Center (NSWC), Dahlgren Division; NSWC, Port Hueneme Division (PHD), East Coast Operations; NSWC, Crane; Naval Undersea Warfare Center (NUWC), New London; Digital Systems Research; Hughes (Fullerton and San Diego; Hughes is now part of Raytheon); INRI; ITT/Gilfillan; Lockheed Martin (Egan, Moorestown, Orlando, and Syracuse); Motorola; Northrop Grumman; ORINCON; and Raytheon (St. Petersburg and San Diego).

A timeline summarizing the evolution of the APL display prototypes is shown in (Fig. 3).

DISPLAY FUNCTIONALITY

From the beginning, the CS@SE displays were feature-rich from the operator's point of view. Following Phase II of the CS@SE, Buscher and Sunday[2] described many features that had proven valuable in at-sea exercises such as the use of color and area fill, online archiving and playback, fusion of multiple-source track data, online detailed NIMA geographic databases, online airway data, auxiliary plots, etc. These features were not in the combat system displays of that era, and some features, such as online airway databases which can change monthly, were controversial. Nevertheless, the experiments continued to explore more issues such as track fusion algorithms, color fill for track symbols, color-coding of sensors and platforms for CEC, colored-dot history trails for CEC air tracks, the use of window systems (X/Motif) to construct easy-to-use capability-rich HMIs, etc. A sample CEC display screen is shown in Fig. 4.

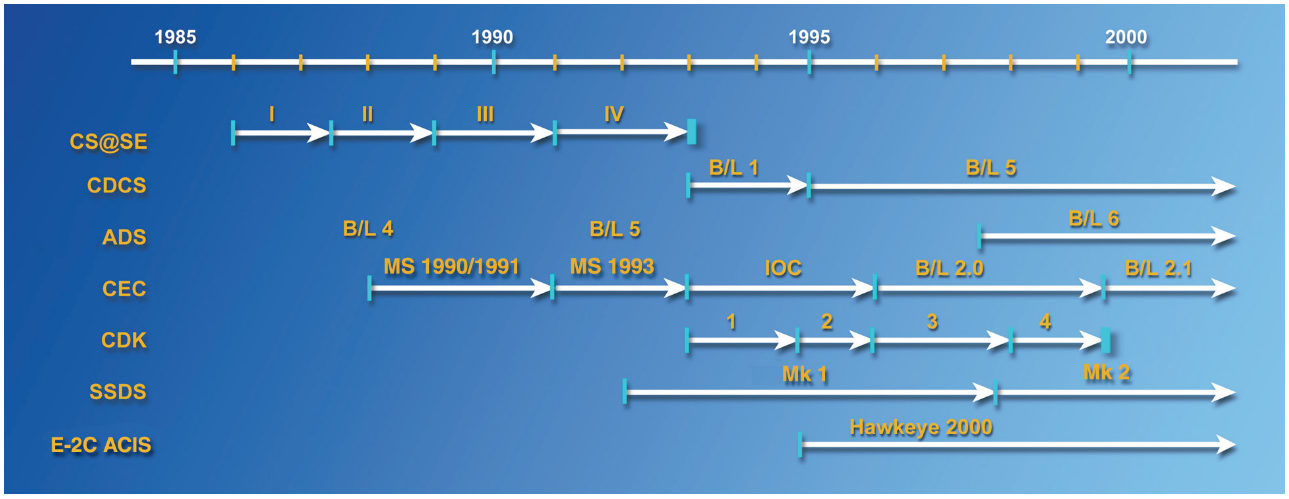
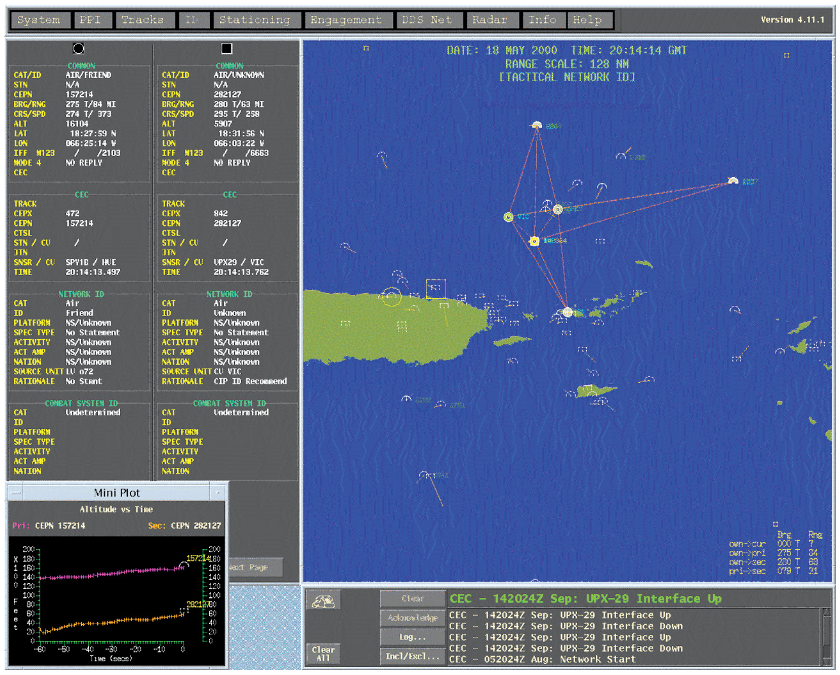


Figure 3. Timeline of APL's prototype display development (B/L = baseline, MS = milestone, IOC = initial operational capability).



```
COMMON                              COMMON
CAT/ID      AIR/FRIEND              CAT/ID      AIR/UNKNOWN
STN         N/A                     STN         N/A
CEPN        157214                  CEPN        282127
BRG/RNG     275 T/84 MI             BRG/RNG     280 T/63 MI
CRS/SPD     274 T/ 373              CRS/SPD     295 T/ 258
ALT         16104                   ALT         5907
LAT          18:27:59 N             LAT          18:31:56 N
LON         066:25:14 W             LON         066:03:22 W
IFF  M123     /    /2103            IFF  M123     /    /6663
MODE 4      NO REPLY                MODE 4      NO REPLY
CEC                                 CEC

CEC                                 CEC
TRACK                               TRACK
CEPX        472                     CEPX        842
CEPN        157214                  CEPN        282127
CTSL                                CTSL
STN / CU      /                     STN / CU      /
JTN                                 JTN
SNSR / CU   SPY1B / HUE             SNSR / CU   UPX29 / VIC
TIME        20:14:13.497            TIME        20:14:13.762

NETWORK ID                          NETWORK ID
CAT         Air                     CAT         Air
ID          Friend                  ID          Unknown
PLATFORM    NS/Unknown              PLATFORM    NS/Unknown
SPEC TYPE   No Statement            SPEC TYPE   No Statement
ACTIVITY    NS/Unknown              ACTIVITY    NS/Unknown
ACT AMP     NS/Unknown              ACT AMP     NS/Unknown
NATION      NS/Unknown              NATION      NS/Unknown
SOURCE UNIT LU o72                  SOURCE UNIT CU VIC
RATIONALE   No Stmnt                RATIONALE   CIP ID Recommend

COMBAT SYSTEM ID                    COMBAT SYSTEM ID
CAT         Undetermined            CAT         Undetermined
ID                                  ID
PLATFORM                            PLATFORM
SPEC TYPE                           SPEC TYPE
ACTIVITY                            ACTIVITY
ACT AMP                             ACT AMP
NATION                              NATION

Pri: CEPN 157214     Sec: CEPN 282127

           Brg     Rng
own->cur   000 T   7
own->pri   275 T   84
own->sec   280 T   63
pri->sec   078 T   21

CEC - 142024Z Sep: UPX-29 Interface Up
CEC - 142024Z Sep: UPX-29 Interface Up
CEC - 142024Z Sep: UPX-29 Interface Down
CEC - 142024Z Sep: UPX-29 Interface Up
CEC - 142024Z Sep: UPX-29 Interface Down
CEC - 052024Z Aug: Network Start
```

Figure 4. Typical CEC engineering display running on a Sun workstation. In addition to CEC engineering and development, these displays are used for test and evaluation support, prototype functionality development, and tactical situational awareness aboard all CEC-equipped ships.

In the final stages of their evolution, these displays focused on a central tactical situation (TACSIT) Plan Position Indicator (PPI) two-dimensional latitude–longitude geographic display with overlaid tactical graphics (e.g., regions, pairing lines) and Navy Tactical Data System (NTDS) track symbols. The E-2C display even had two PPIs. The displays also had character read-out (CRO) panels that would show information for the hooked objects in the PPI which were selected using PC-familiar point-and-click operations with a mouse or trackball. Other features included a pulldown menu bar that stretched across the top of the screen, a pop-up menu for the PPI, a variable action button panel at the bottom, and a miniature hooked-track altitude-plot panel. These essential ingredients were then embellished for mission-specific displays with a taskbar at the screen bottom, pop-up alerts, and numerous operator control panels (usually elicited from the pulldown menus). All this made for a very powerful and flexible, yet very simple and clear, display system that operators rapidly learned to appreciate.

The success of these displays was in part due to the similarity of their X/Motif windows environment to the PCs that many people were already comfortable with. However, these systems were not deployed on PCs, but rather on UNIX workstations with the power and robustness to support the near-real-time reliable processing required by a combat system. Early experiments were performed using DEC, Sun, and Silicon Graphics workstations. However, the Navy displays eventually were deployed under HP-UX running on the Navy's new Q-70 display hardware. Recently, the CEC displays have been successfully ported to PCs running Linux. All code for these displays was written in C and C++ and is highly portable. Using UNIX-based software rather than commercial turnkey PC operating systems gave the display developers the essential control over computer resources that was needed for the performance demands of these systems.

Navy combat displays have tight performance requirements. They must accept high-throughput input data for the extremely large number of objects being tracked, update and refresh the displays immediately, and respond instantaneously to operator actions (e.g., hooking to get a data readout, range changing, responding to an alert). Therefore, much of APL's display development work involved streamlining the performance of all system elements. The end result was a well-constructed, high-performance TACSIT graphics-engine architecture with a clean Application Programmer Interface (API).[4] This display engine was delivered in the final CDK 4.4 software release in December 2000.

DISPLAY ARCHITECTURE

To demonstrate the rich feature set and near-real-time capabilities required by future combat system displays, the CS@SE prototypes—and later the CDK software suite—used, and in some cases invented, advanced architectural elements not normally seen in modern combat systems. A number of these features are described next.

UNIX Shared Memory

CDK-based displays, such as the ADS and ACIS, maintain and update large amounts of data, including information about all tracks in the combat system, operator preferences, and system status. These data must be available at any moment to multiple display components. For example, track data must be simultaneously available for rendering in the PPI, display in the CRO, filtering to determine output characteristics such as color and brightness, and positional updates from the combat system. These types of data are shared between applications through UNIX shared memory. Shared-memory segments can be created by any UNIX application and, once created, can be read from and written to by any other applications. Thus they provide a means for many applications to share a single memory block. CS@SE and CDK made extensive use of shared memory; for example, a typical CEC display uses well over 30 Mbytes of shared memory.

While merely creating and attaching to shared-memory segments might be a common software practice, CDK introduced an advanced object-oriented version known as the "named buffer." Named buffers eliminated many features of shared-memory segments that were unwieldy and error-prone while at the same time introducing the concept of machine-independent shared memory. With a named buffer, an application attaches to shared memory not just by size and key, but also by segment name and computer name. Thus while a named buffer might reside on one computer, it could be transparently accessed by any application on any other computer on the network.

Three significant named buffer shared-memory segments used in all CDK-based displays are the track file, a display filtering segment, and the graphics entity data (GED)[5] cache. The track file holds information about all combat system tracks, including position, identification, course, range, and bearing. The filter buffer contains user-entered settings that control the appearance of all graphical items on the PPI. The GED buffer contains instructions in a GED language for drawing objects that are to appear in the PPI. Data generator applications convert tactical data (e.g., tracks, engagements, radar ranges) into geometric data (e.g., track symbols, lines, polygons, and sectors) using the GED language.

Triple Buffering

One feature of APL's prototype displays is raster-based drawing as opposed to vector drawing. In raster drawing, the PPI is updated in a single step versus the independent erasing and redrawing of individual graphical items as their positions change. In this process, the current positions and characteristics of each item to be displayed in the PPI are periodically retrieved from the various databases and then drawn. To prevent flickering, this drawing step must be performed as quickly as possible. A common technique that reduces flicker is double buffering, in which new graphics are first drawn into a hidden background window and then quickly copied to, or swapped with, the visible foreground window. Many entertainment systems use this method to achieve smooth animation.

Initially, the CS@SE used a double buffering scheme to reduce flicker and enhance drawing performance. However, some PPI elements, most notably the filled land areas, could not be redrawn fast enough for a smooth presentation to the operator. Those areas were restricted to being drawn on only certain screen planes, which would not be erased and redrawn unless necessary. This greatly enhanced performance. However, because land areas and tactical graphics were segregated into separate screen planes, the number of available colors was significantly reduced. CDK solved this problem by taking the double buffering technique one step further and introducing triple buffering to the combat system display. With triple buffering, the PPI was no longer limited to a single background buffer for fixed maps. Instead, an arbitrary number of auxiliary buffers was introduced, each of which held either fixed or transient graphics. However, whether fixed or transient, during each PPI update the auxiliary buffers were first copied to an interim hidden buffer, and that hidden buffer was copied to the visible foreground buffer. This not only increased the number of available colors, it also allowed for new PPI features, such as software-animated radar sweeps. Figure 5 illustrates this concept.

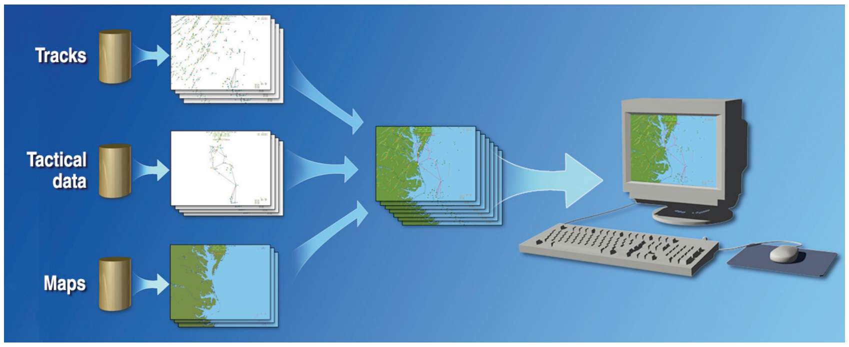


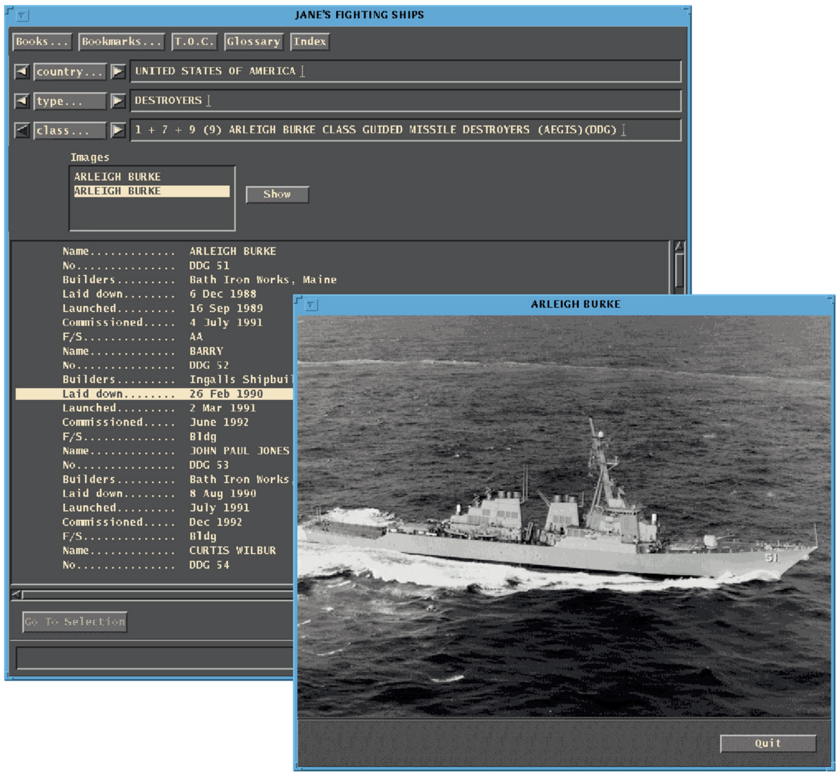
Figure 5. Triple buffering allowed CDK displays to expand the number of available colors, which enabled enhanced visualization of tactical data.

Prototype Intelligence Databases

The use of online intelligence databases was explored beginning with Phase IV and continued through CDCS development. Experimental GUIs were created to look into HMI features that would enhance the warfighter's situational awareness yet not saturate the warfighter with information overload. Special-purpose browsers were created for database access and search as well as for image retrieval. The prototype database chosen for initial implementation was the Jane's series of texts such as *Jane's Fighting Ships* and *Jane's Underwater Warfare Systems*. Database extraction software was written to prepare the texts for rapid online retrieval, and a CD-ROM jukebox was installed for image access. An example of the database GUI used during the CDCS experiments is shown in Fig. 6.

Automatic Prehooking

Hooking is the process of selecting a track or other graphic object on the PPI and designating it as the



```
country...   UNITED STATES OF AMERICA
type...      DESTROYERS
class...     1 + 7 + 9 (9) ARLEIGH BURKE CLASS GUIDED MISSILE DESTROYERS (AEGIS)(DDG)

Images
ARLEIGH BURKE
ARLEIGH BURKE

Name............  ARLEIGH BURKE
No..............  DDG 51
Builders........  Bath Iron Works, Maine
Laid down.......  6 Dec 1988
Launched........  16 Sep 1989
Commissioned....  4 July 1991
F/S.............  AA
Name............  BARRY
No..............  DDG 52
Builders........  Ingalls Shipbuil
Laid down.......  26 Feb 1990
Launched........  2 Mar 1991
Commissioned....  June 1992
F/S.............  Bldg
Name............  JOHN PAUL JONES
No..............  DDG 53
Builders........  Bath Iron Works,
Laid down.......  8 Aug 1990
Launched........  July 1991
Commissioned....  Dec 1992
F/S.............  Bldg
Name............  CURTIS WILBUR
No..............  DDG 54
```

Figure 6. The Jane's database browser. The various Jane's books were selected for prototype online intelligence database applications. The browser, especially designed for CS@SE, provided a unique hierarchical browsing mechanism, bookmarks, simplified hyperlinks, keyword searching, and indexing. Text was stored on display hard drives, but images remained on CD-ROMs because disk space was a scarce commodity in the early 1990s.

current object of interest. The operator typically accomplishes this by rolling the trackball pointer around the PPI until the pointer is near the desired object and then clicking a trackball button. The software scans the database of visible items and determines, within a certain threshold, which object is nearest to the pointer. The nearest object is then placed under close control, that is, it is "hooked." Most display systems allow two separate objects to be in close control at the same time. If the operator needs information about some other object, one of the two hooks must be dropped (released from close control) so that the third object can be hooked. Any functions that were being carried out on the previously hooked object will be interrupted or lost.

Phase IV of the CS@SE included one of the first implementations of the concept of advanced hooking or prehooking.[6] Prehooking gives the operator additional information about objects in the PPI without disturbing the two currently hooked items. To prehook a graphic object (e.g., a track), the operator merely moves the pointer near the track. Graphical feedback is provided to indicate which track is nearest the pointer, and brief ephemeral track information, similar to CRO data, is drawn directly on the PPI. As the pointer is moved around the PPI, the prehook information is continuously updated in real time. The prehook also indicates which object would be hooked if the operator clicked the trackball button. This helps the operator select specific objects in a congested PPI. The prehook algorithms perform extremely high-speed scans of displayed graphics to determine, in real time for every pointer motion event, which object is nearest the pointer.

Intermachine Data

Computer programmers familiar with the C programming language often deal with data of different scopes. The scope of a C variable indicates where its value can be accessed and which parts of an application, or set of applications, can access it. The following four scopes are commonly found in many C applications[7]:

- Block scope: Variables of this scope can be accessed only from within the segment of code in which they are declared. It is the most restrictive scope.
- File scope: This scope limits a variable's access to the file in which it is declared. The keyword "`static`" typically denotes this type of variable.
- Global scope: Global variables, denoted by the keyword "`extern`," have a scope that extends their access throughout an application. This includes library modules and other compilation units.
- Machine scope: This scope extends beyond the C language itself into the operating system and is often accomplished via the shared-memory mechanism. It allows two or more applications to share data structures and common values.

A fifth scope of data access extends beyond the application and beyond the machine, and binds applications that may be running on separate computer systems by providing a common, machine-independent memory access mechanism. CDK implemented a form of intermachine data access that used the named buffer paradigm to give C++ application developers the ability to write software that stored, retrieved, and modified data, independent of the machines on which their applications ran. For example, the CDCS used the fifth scope to allow operators to address and manipulate remote display screens within the Combat Information Center. CDCS operators sitting at one console could activate display features on other consoles running on other computers. Using the fifth scope paradigm, only one set of software code was required to control display features on both the local and remote computers.

Fault Tolerance

It is the nature of combat systems that they should be available all the time and suffer no faults. It is also the nature of software systems that, because of their complexity, it is usually not possible to test all components under all possible input combinations. The former condition strives for no faults, while the latter admits that faults will occur. One goal of APL's prototype systems was to create a robust environment that could allow for faults, identify them, and recover.

An architecture for dealing with software faults that occurred in the form of an abnormal termination of one or more applications was devised. It consisted of a daemon program called the siva daemon (*sivad*) process, which itself initiated all other display programs. As a parent of those programs, *sivad*, via the UNIX "`fork`" mechanisms, could monitor and detect abnormal terminations. When such an abnormality was detected, the parent *sivad* process would immediately reinitiate the faulty application. This mechanism has allowed many of APL's prototype systems to remain operational for weeks at a time while participating in at-sea trials.

Client/Server Applications

While a client/server programming paradigm is commonplace today (anyone who uses a Web browser is running the client end of a client/server pair), this was not the case a decade ago. CS@SE, CDCS, and CDK incorporated this paradigm in several instances, including a centralized geographic database server (geoserver), a TDA server for the CEC Position Planner and Radar Masking tools, and NAPS. Each of these tools allowed for the storage and retrieval of large amounts of data on a remote server computer that was allocated specifically for those tasks. Client applications ran on the individual consoles and sent data request messages.

The client/server paradigm was incorporated into the display architecture for several reasons. In the case of geoserver and NAPS, disk space limitations on individual consoles were tight. Storing the geographic databases and the archive playback data on a remote machine freed up console disk space for more pressing needs. In the TDA case, the server typically performed operations that were computationally intensive. Running the server on a remote computer prevented those calculations from interfering with other console functions. These and other architectural elements are illustrated in Fig. 7.

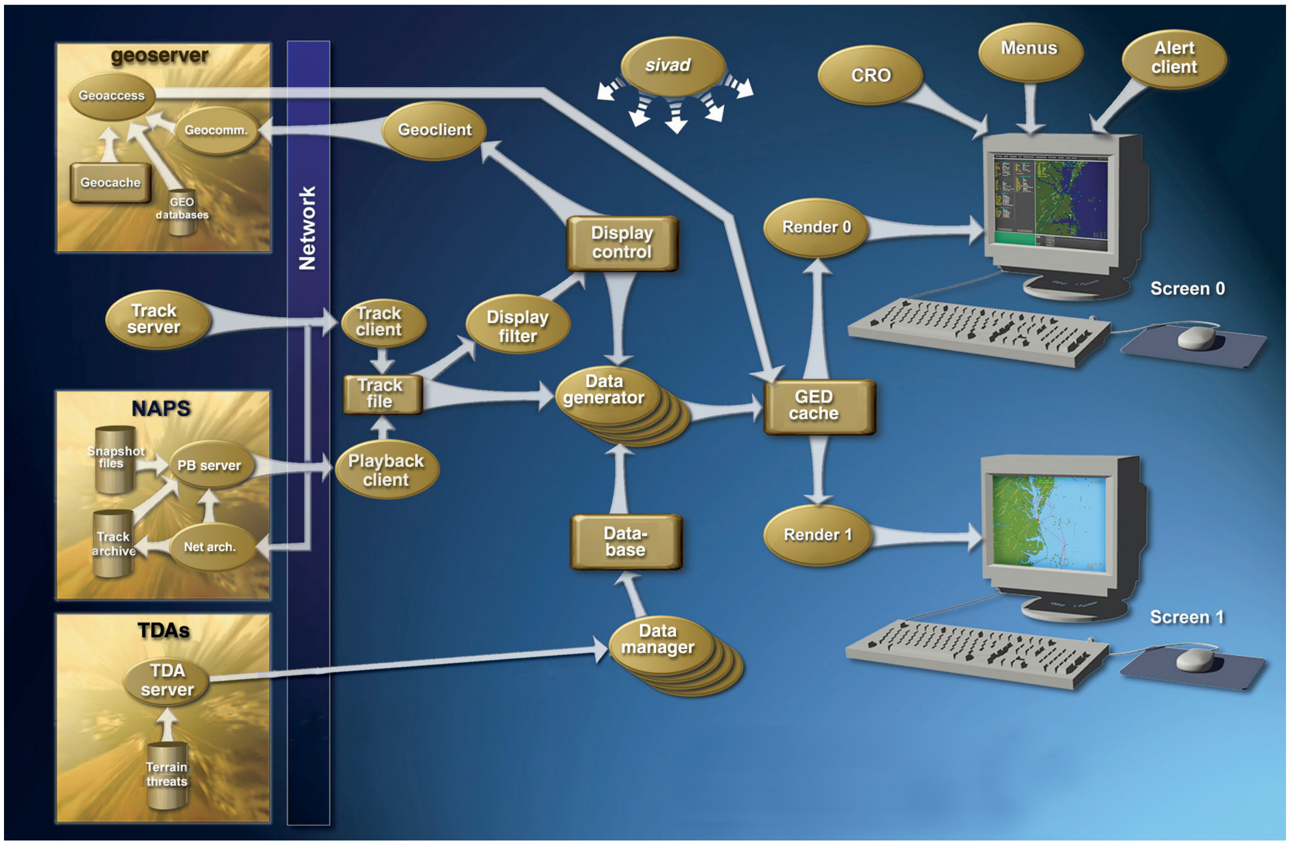


Figure 7. An early CDK top-level data flow diagram.

CONCLUSION

In the late 1990s, the Navy officially transitioned the CDK project from APL to industry. After that, the CDK team dispersed and no more new code was developed. However, the Laboratory collected bug fixes from the numerous CDK users, and a final stable release (CDK 4.4)[4] was delivered in December 2000. A CD-ROM with this software is available from APL through the authors.

REFERENCES

[1]Willey, F. J., and Nesbitt, D. W., "Advanced Graphics for Command Displays," *Nav. Eng. J.* **98**, 130–137 (May 1986).

[2]Buscher, D. J., and Sunday, D. M., "The Command Support At-Sea Experiment," *Nav. Eng. J.* **102**(3), 25–36 (May 1990).

[3]Grant, C. J., "Aegis AAW Correlator/Tracker (AACT) Experiment," *Nav. Eng. J.* **102**(3), 37–42 (May 1990).

[4]Davis, D. E., Nesbitt, D. W., and Mallder, V. A., *Common Display Kernel (CDK) V4.4 Application Programmer Interface (API)*, ADS-98-078, JHU/APL, Laurel, MD (Aug 2000).

[5]Nesbitt, D. W., *Graphics Entity Data Format Specification*, F3D-3-1695, JHU/APL, Laurel, MD (17 Mar 1995).

[6]Osga, G., *Combat Information Center Human–Computer Interface Design Studies*, Naval Command, Control, and Ocean Surveillance Center, San Diego, CA (Jun 1995).

[7]Stroustrup, B., *The C++ Programming Language*, AT&T Bell Telephone Laboratories (1991).

ACKNOWLEDGMENTS: The work described in this article was sponsored by the Navy's Aegis (PMS-400), CEC (PMS-465), and SSDS (PMS-461) projects. At APL, numerous managers, software developers, and hardware specialists supported this effort. In addition to the software leads mentioned in the Background section, key management support came from Jerry Bath, Warren Citrin, Gary Gafke, Jerry Krill, Bob Lundy, Jim Palmer, Dennis Serpico, Wink Wilkinson, and Bill Zinger. Also, numerous software developers supported these projects, so many that they cannot be listed here. The efforts of all these people ultimately made these projects a success.

THE AUTHORS

DANIEL M. SUNDAY is a Principal Professional Staff mathematician in APL's Air Defense Systems Department. He obtained a B.Sc. in mathematics and physics from the University of Toronto in 1967 and a Ph.D. in mathematics from the University of Minnesota in 1971. From 1975 to 1978 he was an NRSA Postdoctoral Fellow in bioengineering at the University of California, Berkeley. Dr. Sunday joined APL in 1980 and has worked on projects that include the TRIMIS medical system, the HIOS network for the Army, CS@SE, and CEC. Since 1995 he has been the lead software engineer for the E-2C "Hawkeye 2000" ACIS production displays now being deployed in Navy squadrons. Dr. Sunday has notable publications in computer science; his areas of interest include computer algorithms, computer graphics, computational geometry, and software development engineering. He teaches courses in computer graphics and computational geometry at the JHU Whiting School of Engineering. His e-mail address is daniel.sunday@jhuapl.edu.

CHRISTOPHER J. DUHON is a member of APL's Senior Professional Staff and Supervisor of the Advanced Display Technology and Development Section in ADSD's Computer Systems Development Group. He received a B.S. degree in computer science and in mathematics from the University of Southwestern Louisiana in 1984 and an M.S. degree in computer science from Texas A&M University in 1990. Mr. Duhon joined APL in 1991 and has been involved in the design and development of advanced command and control software systems for the surface Navy, including the CS@SE, CDK, and CEC. He is currently lead engineer for the CEC display system, working on prototype user interfaces and graphical visualization. His e-mail address is chris.duhon@jhuapl.edu.