Hoag, L., & Bedrosian, J. L. (1992). Effects of speech output type, message length, and reaudi-torization on perceptions of the communicative competence of adult augmentative and alternative communication system users. *Journal of Speech and Hearing Research, 35*, 1363-1366.

Higginbotham, D. J., & Wilkins, D. P. (1999). Slipping through the timestream: Social issues of time and timing in augmented interactions. In D. Kovarsky, J. Duchan, & M. Maxwell, (Eds.), *Constructing (in)competence: Disabling evaluations in clinical and social interaction.* Mahwah, New Jersey: Lawrence Erlbaum Associates.

Light, J., Beer, D., Buchert, L., Casey, E., DiMarco, R., & Dolan, K. (1995). *The effect of grammatical completeness on the communicative competence of students who use AAC.* Paper presented at the ASHA annual Convention, Orlando, FL.

Todman, J., & Alm, N. (1997). Talk boards for social conversation. *Communication Matters, 11*, 13-15.

Vanderheyden, P. B. (1995a). *Organization of prestored text in augmentative and alternative communication systems: An interactive schema-based approach.* Technical Report #AAC9501, Applied Science and Engineering Laboratories, University of Delaware, Newark, DE.

Vanderheyden, P. B. (1995b). An augmentative communication interface based on conversational schemata. *Proceedings of the IJCAI '95 Workshop on Developing AI Applications for Disabled People,* (pp 203-212).

Division 12 is especially pleased to acknowledge the career accomplishments of **David R. Beukelman,** *recipient of ASHA's Honors, the Association's most prestigious award.*

Visual Design: Implications for Developing Dynamic Display Systems

Christine Scally
Clark County School District
Las Vegas, NV

Introduction

In 1999, I began work on a dynamic display concept that was eventually realized as Velocity, a configuration of communication pages for school age children available on the Enkidu Research Tablet Portable Impact. My initial goal was not to create a set of content pages, but to create a framework for holding content that was fast, expandable, and easy to learn and use. I wanted a better user interface.

In addition to the end product, Velocity's development yielded a universally applicable design process for visually coding information within a dynamic display user interface. The goal of the visual design process is to create a visually consistent and informative user interface that is easy to learn and use.

This paper describes the visual design process in detail, providing illustrations from the development of this system and instructions for applying the process to other devices. The process is a cookbook, of sorts, but it isn't about baking; it's about frosting. It isn't a recipe for building a configuration from scratch. It is a set of instructions for taking any existing dynamic display configuration and redesigning its appearance so that it is easier to learn and use. It is a process for creating a consistent, informative visual interface design. Before getting into the process, consider the current state of dynamic display visual design for comparison.

The State of the Art

Historically, only four aspects of dynamic display systems have been visually marked for the user in systematic ways. The message content of communication buttons, the specific command function of control buttons, and the specific destination of navigation buttons are all marked using the button's symbol and label. In addition, navigation buttons are usually visually distinct from other kinds of buttons. Depending on the device, this is accomplished by assigning a specific shape or color to navigation buttons or marking them with an arrow. While other visual design features are available in most devices, their application has been left up to the individual programmer, and no systematic means of applying these features has been offered. As a result, visual design features tend to be used in very unsystematic ways.

Consider the feature of button color, for example. In a single configuration, color might be used to identify syntactic categories within a Fitzgerald key; distinguish boys names from girls names, vowels from consonants, and buttons navigating to home pages from buttons navigating to school pages. It may be used to mark a control button on a page or to reflect the aesthetic preferences of the programmer or user. In such a system, the meaning of a button's color is hard to determine or predict. Pink might mean verb, girl, vowel, home page, backspace, or "Gee, that's pretty."

Designing the Visual Interface

As I designed my page configuration, I wanted to make certain that pink only ever meant one thing, and that the information pink conveyed was helpful to the user. At a more global level, I wanted the user to be able to predict the content and action of any page or button just by looking at it. The design process described below allowed me to achieve this goal. It involves three tasks:

1. Specifying the information about the configuration that the user needs to have,
2. Determining what visual elements are available for the dynamic display interface, and
3. Applying visual features to the information components in a systematic, rule-governed way.

Specifying Information

Pages and buttons are the standard physical components of any button grid dynamic display. These are the objects being marked. The goal of this task is to determine what specific information about the pages and buttons the user needs to predict the configuration's behavior.

The most important piece of information to specify is action. Pages and buttons can act in a number of different ways within a dynamic display system. Consider button action. With the exception of navigation, button actions are usually hidden. When selected, a button might speak, send information to a text editor, perform a control function, or carry out some combination of actions. In most configurations, the only way to know what a button will do is to select it. Visually marking button actions allows the user to predict what a button does just by looking at it.

Actions may not correspond directly to programming options within a specific device. The specification process is intended to define the behavior of the configuration from the user's point of view so that this behavior can be marked for the user. It in no way reflects how the action is achieved from the point of view of the programmer.

Consider page action. In Velocity, two page actions were specified: stationary pages and virtual pages. Stationary pages remain active until the user chooses to select a navigation button. Virtual pages disappear after a single hit, regardless of what button is selected. This is not a transparent subdivision of pages within any available device. In a Dynavox, for example, the two main page types are pages and pop-ups. Both pages and pop-ups are stationary pages by default. A pop-up can be turned into a virtual page by setting the auto-close feature of the pop-up to on. A page can be made virtual by linking every button on the page back to the previous page. In Enkidu Portable Impact devices, every page in a configuration can be both stationary and virtual. The action of a page is dependant on whether it is accessed via a navigation button or a visit button.

A second piece of information that can be specified is page type. Information about page type is used to specify information about the page and the navigation button that goes to it. The process of specifying page type should be undertaken with caution. In the Velocity there are a limited number of content-based page categories, and the navigation options provided on a page are determined by its category membership. Knowing the content category of a page allows the user to predict available content and informs him or her about the overall organization of the configuration. The categories are both learnable and useful. Depending on the configuration, there may be so many page types that there is no good way to code them and no good reason to learn them. Even if every page does not fit within a category structure, a few easily definable and important page types, such as pop-ups, the keyboard, or the main screen, can be specified.

In any dynamic display system, buttons can be divided into three primary action classes: navigation, communication, and control. Button actions are specified by determining their class and specific action. Navigation button actions can usually be categorized according to the type of page they go to. If page actions or page types are specified, the navigation buttons that link to those pages should also be specified. This will help the user predict where a navigation button goes. If pop-ups are used, it may be necessary to specify buttons that navigate to pop-ups versus buttons that navigate to pages. There may not be a one-to-one correspondence between navigation and page actions. If a page is sometimes accessed through a navigation button, and sometimes accessed through a visit button, code the action in the button, not in the page.

Many different options for communication buttons are offered in dynamic display, and the actions can be classified in different ways. A button can speak when selected or not, send content to the text editor or not. The message spoken when the button is selected may be the same as or different than the message sent to the text editor. When specifying the actions of communication buttons in a configuration, the key to predictable, user-friendly design is restraint. Only use options you need. Only provide the user with helpful information. In the concept configuration, only two actions are used. Buttons that contain complete utterances are "speak" buttons. They speak their message content when hit and do not send anything to the text editor. Buttons that contain a partial utterance (letter, word, or phrase) are "speak and send" buttons. They speak their message content when hit and send the message content to the text editor so that the entire message can be repeated after it is assembled.

Both types of button actions could also be classified based on whether or not the spoken content exactly matches the button label. I didn't specify this information because it isn't helpful. Marking this piece of information would just make the user interface busier and noisier than it needs to be.

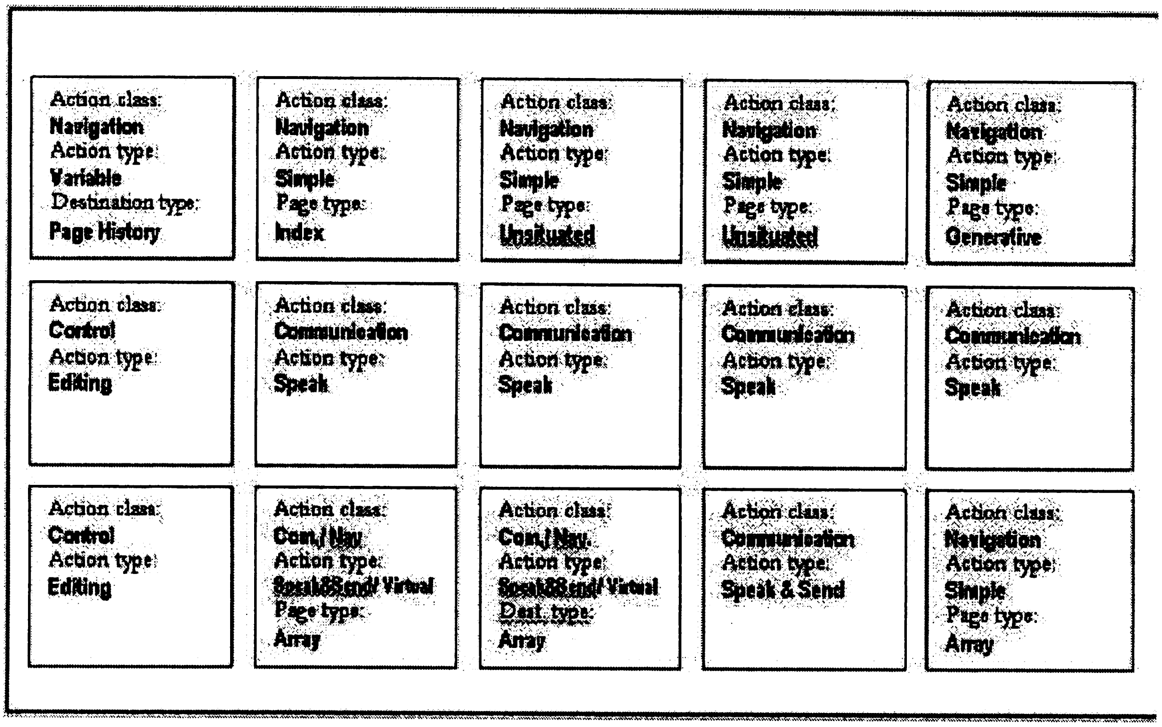
Control buttons can be specified based on the item that receives the action. Some buttons, such as speak, clear, and undo, control the text editor. Controls such as power, volume control, and scanning speed, which adjust the status or operation of the communication device at a global level, should be specified if they are included in the user interface. If the communication device is also used for computer access or environmental control, these types of control buttons should also be specified. See Figure 1 for an example of a communication page that includes all specified information.

Determining Design Options: Objects, Features, and Values.

There are five visual objects on a standard button grid: page background, buttons, borders, labels, and symbols. Each of these objects has multiple features that can be manipulated or assigned a specific unique value.

Consider a button as a visual object. It has at least six visual features that can be changed or assigned a value: shape, color, pattern, texture, size, and position. A button can be a square shape, a square with rounded or turned down corners, a rectangle, a circle, an arrow, or a file folder. A button can be red, yellow, blue, ...mauve. It may have a dotted, checked, or striped pattern; a raised, shadowed, or flat texture. It can be large, or small in size. Its position can be defined based on its location within the row-column structure.

Because I was working with pen and paper, and not with a specific device, I set out to consider any and all design options applicable to a button grid interface. I listed the objects available on a button grid, determined what features of each object could be manipulated, and considered the values available for each feature. The object, feature, value framework applies to any device, but the specific features and values available for each object may be limited. Know what you have to work with before you begin.



Action class: **Navigation** Action type: **Variable** Destination type: **Page History**	Action class: **Navigation** Action type: **Simple** Page type: **Index**	Action class: **Navigation** Action type: **Simple** Page type: **Unsituated**	Action class: **Navigation** Action type: **Simple** Page type: **Unsituated**	Action class: **Navigation** Action type: **Simple** Page type: **Generative**
Action class: **Control** Action type: **Editing**	Action class: **Communication** Action type: **Speak**	Action class: **Communication** Action type: **Speak**	Action class: **Communication** Action type: **Speak**	Action class: **Communication** Action type: **Speak**
Action class: **Control** Action type: **Editing**	Action class: **Com./ Nav.** Action type: **Speak&Send/ Virtual** Page type: **Array**	Action class: **Com./ Nav.** Action type: **Speak&Send/ Virtual** Dest. type: **Array**	Action class: **Communication** Action type: **Speak & Send**	Action class: **Navigation** Action type: **Simple** Page type: **Array**

Figure 1. Development schematic of a communication page including all specified information.

Applying Visual Elements

Two visual design goals constrain the application of visual elements within the configuration: The visual interface should be meaningful, and it should be easy to learn and use. Each specified information component needs to receive a unique visual mark. The meaning associated with each visual element should be learnable with minimal exposure to the configuration, and without direct instruction. The visual elements should be recognizable without effort.

Conclusion: Design Guidelines for Dynamic Display Devices

Creating a visually consistent, informative and *learnable* user interface requires careful planning and consideration of the graphic resources at your disposal. This article describes how it can be accomplished in a principled manner. The process works. After completing a 15-minute tutorial, my beta-testers were able to use its visual features to predict the action of buttons and navigate to specified pages.

The following guidelines will help ensure the meaningfulness and constrain the noise and complexity of the visual information presented on a single page and within the configuration as a whole.

Visual Design Guidelines

1. The symbol and label text are used to signify the specific

communication content, specific navigation destinations, and specific control actions of individual buttons. The remaining, more stylistic elements are used to signify action and category information.

2. Values must be easily distinguishable on a page and easy to recognize from page to page. If white and off white don't look very different, they can't both be used.

3. Any given feature-value can signify only one piece of information, regardless of what object it is applied to or what other features are applied to the object. "Blue" has to mean the same thing whether it is applied to a label, page, square button or round button.

4. Each information component should be coded only once. Control buttons can be marked by a triangular shape or a red border, but should not be marked by a triangle with a red border. The exception to this rule is button position. For buttons that occur frequently throughout a configuration, button position can and should be used as a secondary code for the button's action.

5. Every information component specified about a page or button needs to be conveyed. The design of a navigation button, for example, should indicate its action class (navigation), its specific action type (virtual) and its destination category (message array). [In the initial Velocity design, this button would be a raised (virtual) rose-colored (message-array) arrow (navigation).]

6. If a button performs two actions, both must be coded. [In Velocity, the button's first action determines it's primary features. If a button has a border, a second action takes place, and the border color indicates the specifics of the second action.]

7. The screen should not get too noisy. Page and button patterns should be used with caution. If it seems like too much information is on a page, it may be necessary to rethink and eliminate some of the information components being coded.

Figure 2 depicts a communication page with visual design applied.

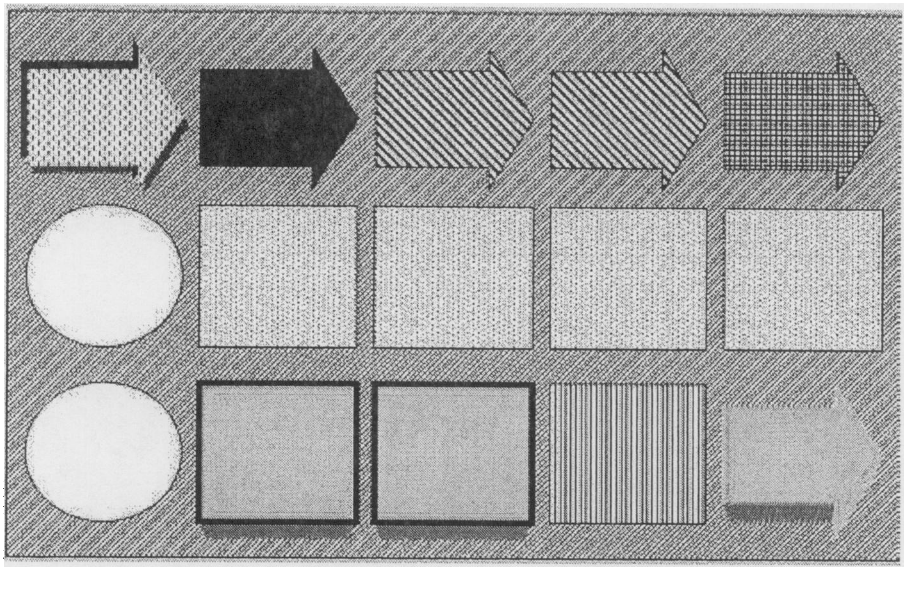


Figure 2. Development schematic of a communication page with visual design applied.

Conference Calendar

If you have a meeting, workshop, convention, or conference related to AAC that you'd like advertised, please submit information regarding event title, date(s) of the event, location, number of CEUs offered, and contact person information to: Ann Beck, Ph.D., 4720 Speech Pathology and Audiology, Illinois State University, Normal, IL 61790-4720, or e-mail her at arbeck@ilstu.edu.

Assistive Technology Industry Association's 2002 Conference
January 16-19, 2002
Orlando, FL
Phone: 877-687-2842
Web site: www.ATIA.org

Third AAC Leadership Conference
January 25-28, 2002
The Cloister on Sea Island, GA
Phone: 800-498-2071

Workshops on Communication Strategies for Children with Severe and Multiple Disabilities
January 31-February 1, 2002
Parent Seminar
February 2, 2002
Lafayette, LA
Contact: Alexandra Dorinson
Phone: 800-410-7069, ext. 102
E-Mail: dorinson@ohsu.edu
Web site: www.designtolearn.com

Professional Internship in AAC in Association With Camp Chatterbox
March 1, 2002 Application Deadline
August 18-25, 2002 Internship
5.0 Asha CEUs
Contact: Joan Bruno
Phone: 908-301-5451
Web site: augcomcamp@aol.com

8th Annual Pacific Rim Conference on Disabilities
March 4-5, 2002
Honolulu, HI
Contact: Valerie Shearer
Phone: 808-956-2673
Fax: 808-956-5713
Contact: Martha Guinan