TITLE: “EFFECTIVE SCREEN DESIGN”

by Tim Bryce
Managing Director
M. Bryce & Associates (MBA)
P.O. Box 1637
Palm Harbor, FL 34682-1637
United States
Tel: 727/786-4567
E-Mail: timb001@phmainstreet.com
WWW: http://www.phmainstreet.com/mba/
Since 1971: *"Software for the finest computer - the Mind"*

"Successful screen design is based on how well the developer knows both the user and the data."
- Bryce's Law

Some time ago I was working with a hospital in the Midwest who was trying to automate some patient admission forms. Hospital forms are notoriously complicated and voluminous (thanks to the lawyers), and this hospital was no different. This made it difficult for the hospital to gather the necessary data about a patient, their physician, and their insurance carrier. As such, they wanted to automate the forms thereby simplifying the collection of data. Unfortunately, the resulting screen designs were essentially no different than the forms. They were very busy and complicated with little editing checks. Frankly, they were no better than the forms they were trying to replace and, because of this, use of the screens were spotty at best.

Designing a computer screen is essentially no different than designing a paper form. But since most of today's developers have little experience in forms design perhaps it is time to review some of the basic elements of good design. First, because a screen or form represents how a human being will interface with a system, we must consider the man/machine interface; its ergonomics. This means we must first understand the intended user, including his/her intelligence level and senses. Someone with a greater proficiency in using a computer will have less difficulty in using complicated screens than someone less conversant in computer technology. As to senses, there is little point in devising an elaborate color scheme if the user may be colorblind. Again, know thy intended user. For more information on ergonomics, see:

No. 65 - *"What Ever Happened to Ergonomics?"* - March 6, 2006
http://www.phmainstreet.com/mba/ss060306.pdf

The objective, therefore, in good screen design (and forms design) is to make something that is easy to use (intuitive; requiring little interpretation and confusion) and effective for collecting data and displaying information. Although the following discussion can be applied to screens as used in some character based operating systems, it is primarily concerned with Graphical User Interfaces (GUI) as used in today's popular operating systems.

The GUI was originally introduced with Xerox's Star computer in the early 1980's. Following this, several companies emulated the Star, including Apple, Microsoft, IBM, and Sun. The GUI was extremely popular as it offered an ease of use never before thought possible. The only problem was that it lacked standards, whereby one GUI implemented program did not behave in the same manner as another GUI program. Fortunately, standards started to appear in the late 1980's with IBM's CUA standards (Common User Access) which provided a detailed list of design standards for developing a GUI based program. (NOTE: CUA was an important part of IBM's System Application Architecture standards - SAA). The benefit of CUA standardization was that users familiar with one GUI program could quickly be trained in how to use another GUI program, since they essentially behaved the same. Today, there are now different interpretations of the CUA standards as implemented by different computer vendors (Gee, what a surprise! ;-) Nonetheless, designing a GUI screen in accordance with accepted standards is preferred over developing of a screen without such standards.

DESIGN CONSIDERATIONS

Today there are some pretty slick tools to quickly build screens. Regardless of their capabilities, a developer should be cognizant of three basic design considerations: Layout, Data Entry, and Support:

A. Layout

The objective here is to make the screen "clean" and consistent. Too much detail makes the screen cluttered and abrasive to the end-user. When designing your screen, consider eye movement, eye strain and, where appropriate, add magnification. Here are some tips for consideration:

Alignment - there should be some simple symmetry to the screen. Disjointed alignment of fields, text, and images tends to alienate users. There should be a comfortable amount of spacing not only around the edge of

(continued on page 2)

(continued from page 1)

the screen, but between sections of the screen. Because GUI windows can be resized (either maximum or to a height and width devised by the user), consider how the screen will look in either form. Borders are useful for defining sections on the screen, but be careful they do not become overbearing and distracting.

Zoning - this refers to the establishment of sections within the screen. This is useful if different types of users are going to be accessing the same screen, or if different sections serve distinctly separate purposes (thereby not confusing one with another). Borders and colors can be useful for distinguishing sections. In a GUI window, notebook tabs can be useful.

Flow - there should be an obvious flow to the screen that will naturally catch the user's eye and prompt him/her in the proper direction. Understand this, Western countries generally observe things from left-to-right and top-down; Eastern countries observe things top-down and from left-to-right; and Middle Eastern countries observe things from right-to-left and top-down. Also understand that the tab order of the keyboard provides direction for the user. As such, the tab order on a screen should go in a logical order and not jump around meaninglessly.

Type Fonts - use common fonts familiar to users. Fancy fonts may be impressive, but will they be supported on all of the computers where the screen will be accessed from? Commonly accepted fonts include Arial, Courier, Sans Serif, and Times Roman. Devise a standard font point size; 10 is generally agreed to be readable by the average person, but then again, will your end-user be an average person? Also, devise a standard scheme for upper-case and lower-case lettering and type styles (e.g., bold, italic); such subtleties will naturally attract the eye.

Colors can be helpful for highlighting sections, accenting required field entries, or for general appearance. Although colors can be helpful, they can also be distracting if they become overbearing. Be sensitive to color contrasts so the user can adequately read the screen. Also be cognizant of end-users who are might be color-blind.

Headings - screen headings should be placed in a standard position for easy identification by the user. A formal name and, where appropriate, a screen number should be clearly visible to the user.

Keyboard/mouse relationship - if in the event a computer mouse either breaks down or is simply not available, the user should still be able to execute the screen using simple keyboard commands. CUA standards are particularly useful in this regard.

B. Data Entry

The proper entry of data is just as important as the physical layout of the screen. Regrettably, many designers take a superficial approach to data collection and, consequently, a lot of time is spent later on cleaning up data in the data base. Considerable time can be saved with a little effort here in screen design. Your objective, therefore, is to produce a screen that will collect "clean" data (as opposed to "dirty" data that will have to be corrected later on).

Before embarking on screen design, the developer should be intimate with the data specifications. This can be obtained either from a good data dictionary/repository, or from the physical data base design. Basically, the developer is looking for the data element's:

• **Length** - the maximum number of characters which may be assigned to a data element.

• **Class** - the type of characters to be expressed; e.g, alphabetic, numeric, alphanumeric, signed numeric, etc.

• **Justification** - the alignment of data within a field when the number of characters is less than the length of the receiving field, e.g., left, right, around the decimal point.

• **Fill Character** - the character to be used to complete a field when the data item to be placed in the field is shorter than the maximum length, e.g., blank, zero, X, etc.

• **Void Character** - the character to be used when a data item's value is unknown or nonexistent, e.g., blank, zero, X, etc.

• **Unit of Measure** - the representation of numeric data, e.g., area, volume, weight, length, time, energy rate, money, etc.

• **Precision** - for numeric data, the number of significant digits in a number.

• **Scale** - for numeric data, the placement of the decimal point.

• **Validation Rules** - the specific values which the data

(continued on page 3)

(continued from page 2)

element may assume, including default values. For example, Yes/No, specific codes or numbers to be used, editing rules, etc. This includes such things as the expression of dates:

20051211
December 11, 2005
12/11/2005
2005/12/11
11-DEC-05

• **Generated data** - quite often it is necessary to show computations based on primary values being inputted by the user. As such, it is necessary to know the data dependencies and the formulas for calculating the generated values.

• **Program Label** - although this will not be visible to the user inputting the data, the developer must understand how the data element is referenced in the data base.

NOW IS NOT THE TIME TO GUESS WHAT THE DATA DEFINITION IS; NOW IS THE TIME TO BE AS PRECISE AS POSSIBLE. Armed with this knowledge, the developer then determines the most suitable mechanisms for collecting the data; for GUI windows, this primarily includes such things as field entries, radio buttons, check boxes, selection lists, and text boxes. The objective here is to force the user to make correct entries as easily as possible. Some considerations:

• Mandate certain field entries be completed before allowing processing to continue. This can be done by: forcing the focus of the window to the field(s) requiring entry; attaching a "hot" color to required field entries (red) and; pop-up messages to prompt the user of problem entries.

• Automatically enter default values into field entries; this saves time for the user (as well as forcing proper entries). One good example of this is to have the user enter a Zip Code first, which should then automatically populate City and State entries.

• Check characters entered and automatically adjust accordingly. For example, automatically upshift or downshift characters - this is particularly useful when entering State Postal Codes (upshift), and entering e-mail addresses (downshift). Also, reject certain character entries and check formats.

• Make active use of selection lists, thereby forcing the user to select a choice from a prescribed list as opposed to typing an entry.

• Encrypt certain sensitive entries, such as credit card numbers and passwords.

• If your application is to allow Asian characters (e.g., Chinese, Japanese, or Korean), provide the ability to allow for the Double Byte Character Set (DBCS). For info, see:
http://publib.boulder.ibm.com/iseries/v5r2/ic2924/index.htm?info/dm/rbal3mst187.htm

• Accommodate the expression of local units of measure, such as dates, times, money, etc. This "personalizes" the screen for the user.

• Depending on the situation, provide or negate the use of the computer's clipboard for field entries.

• Where applicable, provide for data entry using voice/speech-type dictation.

Finally, format the collected data to suit the targeted physical data base.

By making data entry "foolproof" you will be saving a lot of time and effort for the end-user, the DBA, and yourself.

C. Support

To minimize user confusion, be sure to include sufficient Help text and messaging facilities into the screen. Too often I have seen screens with little support in this regards. Again, CUA standards should be observed whenever possible.

Help Text - should be provided for:

A. The screen overall - explaining its overall purpose, who should be using it, and how the data will be processed (its behavior). The Playscript language technique for writing procedures is particularly useful in this regards (see "References" below for details).

B. The various sections of the screen sections (if multiple sections).

C. Field entries - showing the name of the field entry, input specifications, along with some sample and suggested entries. If a generated value is displayed, explain how it is computed (from other field entries).

"Help" push buttons on the screen are helpful, but everything should be related to the F1 Help key, particularly field entries. Further, all screens should feature a Help

(continued on page 4)

(continued from page 3)

action-bar-choice which includes an Index of subjects, and "About" (identifying the name and version of the software in use).

Messages

Messages basically come in three forms: Informational (requiring no action), Warning (that a potential problem might exist), and Error (prohibiting processing). All messages should be clearly written and easy for the user to understand. For warning and error messages, do not simply report a problem to the user, but also advise him on what he should do about it. In other words, point him in the right direction and don't leave him hanging.

CONCLUSION

Good screen design requires a developer in tune with his intended audience and who can create a simple and effective solution that is easy for the user to execute, yet promotes the collection of "clean" data. The developer must strike a careful balance between what is graphically elegant and what is practical for the user to use.

One element of design that is alluded to in this discussion is the development of universal systems whereby screens can be translated into foreign languages. There are some simple tricks for doing this. Be sure to read:

No. 03 - *"Creating Universal Systems"* - Dec 20, 2004
http://www.phmainstreet.com/mba/ss041220.pdf

Above all else, the developer should observe all pertinent design standards when creating screens. As mentioned earlier, users will be more likely to accept and implement new programs if their design is similar to programs they are already familiar with. The need for standardization cannot be stressed enough. To this end, some companies even go so far to devise a library of standard screen templates for developers to use. This does two things; it helps enforce design standards, and; it expedites the development of the screen. But in the end, successful screen design is based on how well the developer knows both the user and the data.

REFERENCES

For vendor CUA (Common User Access) Standards, see:

IBM
http://www-306.ibm.com/ibm/easy/eou_ext.nsf/publish/558

Microsoft
http://msdn.microsoft.com/library/default.asp?URL=/library/books/winguide/fore.htm

Apple
http://developer.apple.com/documentation/index.html

Sun
http://docs.sun.com/app/docs/doc/802-6490

For a description of the "Playscript" procedure language, see:

No. 38 - *"The Language of Systems"* - Aug 22, 2005
http://www.phmainstreet.com/mba/ss050822.pdf

END

About the Author

Tim Bryce is the Managing Director of M. Bryce & Associates (MBA) of Palm Harbor, Florida and has 30 years of experience in the field of Information Resource Management (IRM). He is available for training and consulting on an international basis.