Chunking

Part I of *Human Interface for the Aesthetically Impaired*

by Kevin O'Boyle

Introduction

This will be the first in a series of articles intended to introduce some of the basic scientific and engineering concepts that lie at the heart of good human interface design. While I will be introducing some technical terminology, I do not require the reader to be familiar with these terms; Indeed, I hope this article might serve as a launching point for the uninitiated into this rich and fascinating subject. My goal is simply to demonstrate that human/computer interface design is an engineering discipline (Human Factors Engineering, to be precise); And while it is true that, as with industrial design, a good sense of aesthetic and elegance may intuitively lead you to many of the truths of human factors, these same aesthetics can often lead you astray without an understanding of the principles that dictate good design. On the other hand, even without a good aesthetic sense, understanding the principles of interface design can teach you to produce and evaluate quality user interfaces.

Defining Human Factors (a.k.a. Technical Gobbledygook)

Human Factors Engineering attempts to meld the knowledge and techniques of a number of disciplines into a cohesive whole. Much of the fundamental Human Factors vocabulary (like "Chunking", "Motor Memory", "Iconic & Echoic Memory" & "The Stroop Effect") come from the cognitive and neuro sciences; The often referenced "Fitt's Law" was derived by studying human physiology and movement and is therefore more properly part of ergonomics. Many of the procedural analysis tools (Like "G.O.M.S" (Goals, Objects, Methods & Sequence) and "Semantic Decomposition") have their origin in methods engineering. Then, of course, there are significant contributions made by those in disciplines that seem far afield of this subject (Edward R. Tufte, a Professor of Political Science and Statistics, wrote perhaps "the" seminal book on data presentation: "The Visual Display of Quantitative Information").

In this series, I will try to touch on most of the significant ideas and principles mentioned above. I have opted to begin with "Chunking" since it seems to be both a far reaching and simple concept to put into use. Further, I believe many Human Interface (HI) principles, such as Fitt's Law, are implicit given a solid understanding of chunking.

So what is "Chunking"?

Well, it's not what you did in your dorm room toilet after your first frat party, nor is it the name of a Chinese food distributor. Chunking is how your brain deals with complexity. It turns out that we humans can't really handle much information at a given time; If you are a genius or an idiot, your working memory (the hip modern name for "short term memory") can retain, at most, only about 7±2 things at one time. "Wait a minute!" you're saying, "I deal with a lot more complicated things than that!!!" Well, you do and you don't: When we are confronted with more than 7±2 things to think about we try to group some of them into "chunks" and tune the others out until we are again dealing with at most 7±2 things. The price we pay for chunking is we can only operate consciously on one level of chunking at a time. In other words: If I look into the contents of a chunk then my mind loses track of the other chunks at the same level as the chunk I am examining.

Indeed, it may well be that the difference between the genius and the idiot may simply lie in their respective ability to break information down into meaningful chunks. This means that anything we, as interface designers, can do to facilitate meaningful chunking will improve the overall usability of the interface and make the user feel smarter.

Remember 7±2, this is the *magic chunking limit*, and make it rule #1 of Interface design:

> **Rule #1:** *Never make the user keep track of more than 7±2 things at a time.*

Keep in mind, also, that 7±2 is an upper limit to cognitive chunking; So what we are really saying is to, ideally, keep the number of chunks you provide to a user at 5 or fewer.

Moving from the General to the Precise

If our cognitive workspace is really this small, yet we manage to do all these amazingly complicated things, it must mean that we have a natural mechanism for gracefully shifting between chunking systems. And, of course, we do: We move from the general to the precise. We start with very broad mental concepts and, without even realizing we are doing it, we look into each of these concepts and start dividing them up into chunks. Then we look at each chunk and divide it up etcetera ad infinitum. We are left with an internal model that feels rich, complex and intricate.

$$ID = \log 2\left(\frac{A}{W}\right)$$

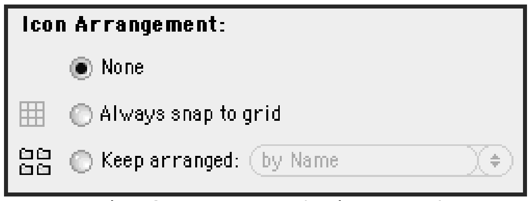
Figure 1: The general form (without the scalars) of the Fitt's law equation
ID = difficulty index, A = amplitude of motion, W = the target width

Fitt's Law (which I will discuss in detail in a later article) is really just a mathematical formula for describing how our motor system transitions from high amplitude/low precision to low amplitude/high precision movement.

So we can refer to this as rule #2 of interface design:

> **Rule #2:** *Facilitate the user moving gracefully from the general to the precise.*

In the context of presenting information to users, the human interface buzzword for this is "progressive disclosure" but let's be careful with exactly what we mean. Most implementations of progressive disclosure conceal detailed choices until a broad choice is made. A classic example would be a tab panel where the tab label indicates a general category and the contents of that panel represent the precise choices. However, it is often more meaningful simply to make the general stand out more boldly than the precise (*see* **Figure 2**). In human interface terms, this is to make the general "highly manifest" in comparison to the precise (the concept of being "manifest" and exactly what that means will also be covered in greater detail in another article).



Icon Arrangement:

● None

○ Always snap to grid

○ Keep arranged: by Name

Figure 2: An example of progressive disclosure without concealment.

In figure 2 we see two examples of Progressive Disclosure without concealment:

First, the heading "Icon Arrangement" is bold and stands out above the radio buttons that relate to icon arrangement. One could argue that there is really no progression taking place at all since "Icon Arrangement" is just a label but you'd be missing the fundamental point of this whole discussion: Chunking is a cognitive process. Because the interface designer organized these choices under a bold heading the user is facilitated in treating all the "Icon Arrangement" choices as one big chunk (to be ignored if icon arrangement does not interest them).

The second example is the "Keep arranged:" radio button which contains a disabled PopUp menu for selecting the type of arrangements the user may choose. Here, not only is the general choice more manifest than the precise choice, the precise choice is inaccessible (while still being visible) until the general choice has been selected. It could be argued that it would be more efficient to keep the PopUp enabled even when "Keep arranged:" is not selected and simply interpret the selection of an arrangement as implicitly selecting "Keep arranged:" thus saving the user a needless step. But there is a false economy in this thinking: There are a number of advantages to the design used. The transition of the PopUp menu from disabled to enabled makes the control highly manifest -- it comes alive! -- communicating to the user that more choices can be made; This serves as a notification to the user to validate their choice of arrangement. While in its disabled state it is significantly less manifest than the rest of the interface allowing the user to focus on the choices relevant to them. By communicating information more efficiently and by changing the emphasis of the display to reflect the users choices, this design requires less mental overhead to interpret and will generally test faster and generate fewer errors than the "more efficient" version described above.

Chunky Bars

Menu bars, of course, are also examples of progressive disclosure: the menu title being the general form and the menu item being the precise form. As with Tabs, this is an enforced form of chunking, the application designer (not the user) determines what will appear under a menu or tab label. Since these precise commands are hidden until the tab or menu is selected, it is incumbent on the designer to make sure that these labels convey a broad category that the user would cognitively associate with the tools contained therein. Menu Layout and, as part of that, choosing the appropriate menu titles, is one of the most important yet most often ignored aspects of Interface design. And small wonder, trying to do it well can feel much like trying to be a mind reader; After all, what seems like a perfectly logical grouping to me may not seem particularly intuitive to you. To really do justice to menu layout I'd have to devote an article just to that subject but, beyond the obvious suggestion of paying close attention to the conventions in use for menu layout, there are a few tips that can be derived from chunking:

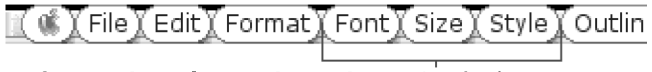


Figure 3: Typeface options clustered together - This facilitates cognitive chunking which, in turn, allows the user to treat this region as one big target.

1) Create Cognitive Chunks - If you are writing a word processor, for example, and need several menus that manipulate typeface then keep these menus together. This facilitates the user in creating a mental chunk that can then treat the typeface options as one big target (*see* **Figure 3**).

2) Progress from the general to the precise - Menu bars have a distinct western language bias, and we westerners scan text from the left to the right. Thus a menu bar should list commands that effect the whole environment to the left of those that effect the document to the left of those that effect the specific mode of operation. This creates stability on the left side of the menu bar which reinforces learning. (Note: there is actually another left to right progression that needs to be considered in Good menu design, from high priority to low priority. I'll get into this dimension of menu design in my article on making things manifest -- which will cover, in detail, menu bar and button bar design).

3) Remember the *magic chunking limit* of 7±2 - If your application has 5 or fewer menus then you are in good shape -- as long as the menu labels are logical and informative. If you have 6-9 menus then the yellow "chunking" warning light should be on, very dimly at 6 but growing brighter as you approach 9; Therefore, you might want to see what can be done to reduce the number of menus but you may still be OK. If you have more than 9 menus, however, count on users being confused by your menu bar.

I have to caution that there is much more to well designed menus than just the titles in the menu bar; Reducing the number of menus by cluttering the remaining menus with lots of commands is almost definitely jumping from the frying pan into the fire. On the Macintosh (and OS X) the menu bar is one of the easiest targets to hit but menu options grow increasingly difficult to target the further down the menu they are; Dramatically more difficult if the menu option is in a sub menu (I'll cover this and more in detail when I discuss Fitt's law). Therefore, if you can't help but to present a lot of commands to the user, it is always better to compromise by adding more menus than by having fewer menus that are long and cluttered.

Chunks on the Menu

The menus themselves need to facilitate cognitive chunking (probably more so than the menu bar); That means clustering commands into groups using separators (for Windows and the Mac OS) or spacers (for OS X) again using the same magic number (7±2). What's wonderful about knowing the magic chunking limit is it reduces the ambiguity of statements like this:

> "*How many dividers to use* [within a menu] *is partially an aesthetic decision. Remember that the Macintosh interface relies on aesthetic integrity as a means of good communication.*"
>
> -- (*chapter 4, Page 62)* Macintosh Human Interface Guidelines

As I said in the beginning of the article many of the truths of human factors engineering can be intuitively discovered by someone with a good sense of aesthetics. It is clear that, in at least this part of the Macintosh Guidelines, the author was driven by their aesthetic sense and did not know the cognitive science behind it. Thus, they got it right and were able to provide an example of a properly grouped menu (**Figure 4**) but they failed to articulate effectively exactly what determines an effective use of dividers.

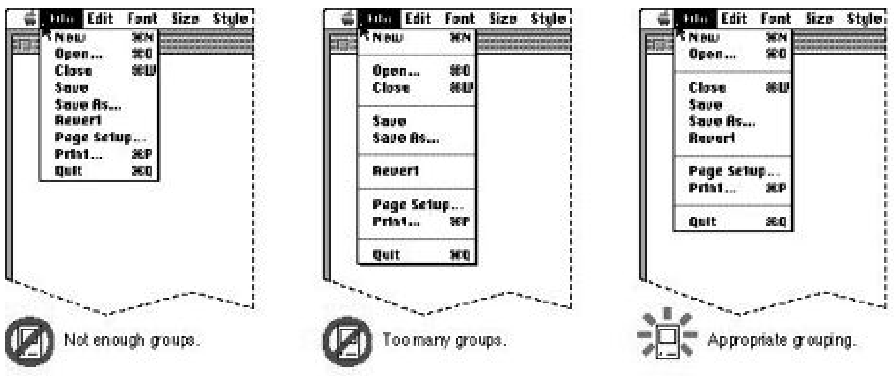


Figure 4: Creating Cognitive Chunks within Menus - Excerpted from the *Macintosh Human Interface Guidelines*, this example demonstrates an intuitive yet excellent grasp of cognitive chunking. (Click on the figure for a more detailed view.)

In the example of appropriate grouping in **Figure 4**, notice how Apple first chose to break the menu into cognitively related chunks then sorted the chunks by priority: First there is a chunk containing commands to create a new window (of the highest priority since the application does nothing without an open window), next come commands to manipulate an existing window (to preserve and reset state), after that are commands for sending the contents of a window to a printer and, lastly (safely alone at the end of the list), the command to quit the application; This yields 4 chunks with no more than 4 items in each chunk.

You may have noticed that there are no groups of 7 or that the second bad example provides 6 cognitive groups of 2 items. Heck, you may argue, there are only 9 items in the list to begin with and isn't that within the upper limits of 7±2? This last argument would come from the same people who don't pay attention to their fuel and oil warning lights until they glow continuously. Remember: 7±2 represents the upper limit of cognitive chunking not the optimal range of chunking. Having 4 to 5 chunks, all other things being equal, is optimal.

In the first bad example, the number of items is at the extreme upper limit of people's ability to chunk information (you're making the users brain work too hard). In the second bad example (which is still better than the first), you are into the caution zone of having 6 cognitive groups (still well within the tolerances of most people but you are making their brains work a bit). Just as significantly, however, you have a maximum of a paltry 2 items in each grouping; Clearly you are underutilizing these chunks. By having 4 groups, with no more than 4 items at both the chunk and the item level, you are imposing cognitive chunks that are very easy to assimilate. There is also another advantage in limiting the number of chunks in a menu: the further down the menu an item is, the harder it is to target; thus, the more dividers you have the further you push down the items on the bottom of the list and decrease the users ability to access them.

Don't blow your chunks!

Human computer interaction is a consensual experience: The user is entrusting the software (and therefore its designer) with the duty of guiding them through whatever task the user has chosen to accomplish. By organizing the information presented to the user using cues to facilitate visual association and chunking a software designer can passively guide users through their tasks. Jean-Louis Gassée, while speaking about programming for the Macintosh, once poetically described good interface design as "*standing underwater ready to place your hands where ever the user may choose to step in order to give them a walking on water experience.*"

By focusing on cognitive chunking (reducing clutter so that the mind can be gracefully guided from 5 or so big concepts to 5 or so more detailed concepts to 5 or so even more detailed concepts and so on) you give the user a sense of being in complete control over the sea of information before them and the freedom to explore far afield without ever feeling lost or overwhelmed. It is far better to have 5 choices that are immediately meaningful than to have instant access to 125 choices without knowing which one you want. Sometimes, it is best to conceal the information until a user selects a chunk, sometimes it is best simply to de-emphasize a chunks contents, but it is, ultimately, the interface designer who chooses how information should be grouped and presented.

It is more important that information is grouped into *manageable* chunks (5 or fewer items) than it is that that information is grouped into *meaningful* chunks. Those of us who remember green bar computer paper will know how helpful it was that the background on the paper alternated from green to white with every 4 lines of output. There was nothing meaningful in this grouping mechanism but it was helpful.

We see examples of mechanisms to help us chunk information every day:

> US Phone Numbers: 1(800)555-1212
> US Social Security Numbers: 199-99-9999
> Even the US Postal codes: Zip: 33033
> Zip+4: 33033-1221

Some of these mechanisms (like the format of the social security number) are completely arbitrary and simply serve to break apart a long number. Some mechanisms (Like a phone number) assign special meaning to each cluster of digits (an area code, an exchange, etc.). Notice, in each example, how the information was parsed into clusters of 5 or fewer digits using dashes and parentheses. Our mind uses these simple visual cues to break the information into it's constituent chunks. If we represented numbers as continuous strings (18005551212, 199999999 or 330331221) it would force our mind to search around for a meaningful way to organize this information and in the process we would easily become lost and confused. These formats provide a visual progression that guide us into chunking the information in an efficient way and the formats themselves become patterns that we can use to quickly glean the meaning behind the string of numbers. It may be too abstract to be useful but it is perfectly valid to define a computer as an automatic chunking machine that attempts to to form manageable and meaningful chunks out of a continuous and changing binary stream.

There are more ways to facilitate the chunking of visual information than I can ever hope to itemize but they all render down to 3 basic techniques:

1. **Visual Separation -** Using white space to separate information into groups.
2. **Visual Differentiation -** Changing the characteristics of specific chunks of information to visually differentiate them from the other chunks.
3. **Visual Progression -** Relying on visual and cognitive cues to guide the order with which users internalize information.

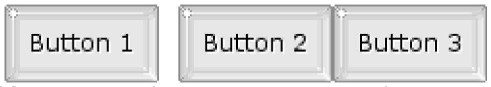


Figure 5: Using white space - 3 buttons are separated into 2 cognitive chunks by changing the spacing between the buttons. This example demonstrates visual separation, differentiation, and progression.

Visual Separation

Had I distributed the buttons in **figure 5** evenly, either by putting a 10 pixel space between each or by cozying them all up against each other, they would have become one cognitive group (with 3 members of roughly equal weight). There is a threshold of spatial separation where cognitive chunking starts breaking down but as long as the objects are aligned on one axis and the spacing between them is apparently equal, the mind will tend to associate them as a single cognitive chunk. Significant variations in alignment, size or spacing all tend to provide powerful cues for cognitive chunking; Notice how powerfully the physical contact between [Button 2] and [Button 3] suggest a single cognitive chunk while [Button 1] which is the same size and type face seems different and more significant.

Visual Differentiation

In **figure 5**, [Button 1] appears significantly more manifest than either of the other buttons. Because it stands by itself at the same cognitive level as the cluster of [Button 2] and [Button 3] and because it is to the left of this cluster (where westerners have been conditioned to start reading). [Button 2] and [Button 3], likewise have been de-emphasized as independent cognitive entities by bringing them into physical contact. This is *differentiation by isolation*; It is easier to find Waldo in a snowy field all by himself than it is in a crowd of colorful characters. White space is your friend; It helps the users mind isolate, identify and prioritize chunks. Back in the days of mainframes, dumb terminals and transaction processing it was important to crowd as much information on to a screen as possible to minimize the transaction load from slow computers over slow networks. With the World Wide Web there can be similar concerns but for most computer interface design there is no need to crowd information on the screen. By isolating and selectively managing the emphasis of chunks of information a software designer can dramatically improve the speed and accuracy of a user.

Visual emphasis can be achieved a number of ways: By isolation (as illustrated above), by increasing visual contrast, by increasing size, or by adding color. Be warned about using color, however: First, a significant percentage of the population is color blind; Second, color is such a powerful mechanism for cognitive emphasis that it should be used sparingly lest you overwhelm the user with too many things demanding attention. As a guiding principle: it is best to design in black and white then add color only to accent and improve visual emphasis.

The only mechanism more powerful than color, for emphasis, is motion -- whose effect is so dramatic it should seldom be used except when it is explicitly important to get and hold the users attention. I will delve more deeply into color and motion when I discuss cognitive overhead but suffice it to say, here, that motion is distracting and can dramatically impact a users ability to focus.

Visual Progression

Most directional forms of visual progression are culturally biased. This doesn’t imply that visual progression is not meaningful rather that the quality of the cues for visual progression can (and should) be tuned for their target audience. In **figure 5**, [Button 1] is made more manifest by its position to the left of the other two buttons because most western languages read from left to right. If in doubt, regarding your target audience, assume a western cultural bias (i.e. information progresses across then down from the top left to the bottom right), not only does this address the cognitive assumptions of the vast majority of current computer users but, because mathematical expression has been globally standardized, it is the most widely used second standard of progression in cultures that adopt different progressions as their primary. The human mind is very good at adapting to nonstandard progressions if there are reasonable visual cues to indicate a starting point and scanning direction but don’t take this as an endorsement for nonstandard progressions; You get a lot of things for free by adhering to the expectations of your users.

However, your user’s expectation may not be quite what you expect: The first item is the most manifest in any progression but there is a trick to progressions that may not seem obvious: It is the last, not the second, item that is the next most manifest. This effect is enhanced by framing information in a window or group box. As the users eye is guided by the rectangle of the box their mind will focus on the top left and bottom right corners (if these corners frame things that can easily be treated as discrete chunks).

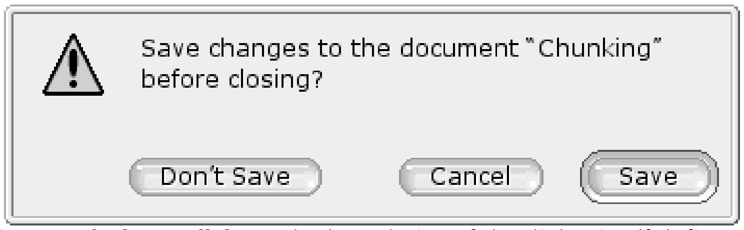


Figure 6a: A typical save dialog - The boundaries of the dialog itself define a chunk. beginning with the Alert Icon (top, left) and ending with the [Save] button (bottom, right). Thus before the mind has parsed out the contents of the dialog the Icon and the [Save] button are already made manifest.

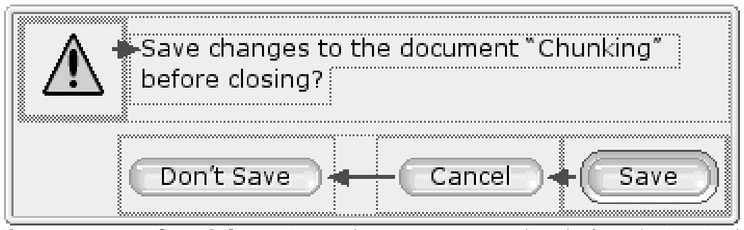


Figure 6b: Anatomy of a Dialog - Once the user enters the dialog their minds break apart it's contents into separate cognitive chunks that direct the users attention through the dialog.

In **figure 6a & b**; The Icon becomes a visual anchor leading the eye to the text to the right of it and together they form one big cognitive chunk: A proclamation to the user. Likewise, the [Save] button becomes a visual anchor for a procession of buttons away from the bottom right corner and this row of buttons becomes one big cognitive chunk: the users choice of responses. So powerful is the positional emphasis of this design that users will frequently hit the [Save] button unconsciously and the [Don't Save] button becomes so transparent that a significant percentage of users tested will not notice it at all.

This example was specifically chosen to illustrate the power and importance of position. Notice that the button arrangement here is exactly the same as the button arrangement provided as an example in **figure 5** with exactly the opposite effect. The positional emphasis of the [Save] button, in the context of the dialog window, is so dramatic that it is made manifest before the brain starts to parse out the contents of the dialog. When the Dialog is reduced to cognitive chunks the cognitive status of the [Save] button refocuses the emphasis of the entire button chunk; lending greater emphasis to the [Cancel] button by proximity and de-emphasizing the [Don't Save] button through distance.

Conclusion

When I decided to write this series of articles I spent some time debating what concept I should start with and I found that even when I considered concepts other than chunking, I was forced to introduce some basic knowledge of chunking before I could explain these concepts; Chunking is that fundamental. As I progress through this series I will inevitably be referring back to and elaborating on Chunking. There is much I didn't cover here.

You may have been struck by the lack of novelty in the interface examples provided; This was deliberate for 2 reasons: First, was to illustrate how transparent the mechanisms of chunking are; to make the reader aware, on a conscious level, of what they might have already known subconsciously. Second, I hoped, that by using common examples of good chunking, the reader would realize that these examples seem so obvious now simply because somebody spent the effort to get it right and was so successful that we no longer see anything but the elegant solution -- no other approaches present themselves to us. This is probably the best measure of elegant design: A design that becomes so transparent to the user that they dedicate no part of their brain to thinking about the interface at all.

It is a worthy exercise to think about chunking as you go through your life: How is my mind breaking apart the wide array of stimulus that I contend with moment by moment? It is a question that will lead to fascinating insights. The way in which our perceptual systems interact with our brains, construct our sense of reality and the whole dynamic of learning, thinking and experiencing is a frontier of science that has a magical, almost metaphysical, quality to it. It's exploration is a path that starts with concepts as mundane as those introduced in this article but which leads quickly to ideas that tug at the very threads of reality. While I do not intend this series to probe that deeply into the mysteries of cognitive science and consciousness, I hope that some of you will be inspired to take that journey.

I sort of lied at the beginning of the article when I said that understanding the concepts of human factors engineering could teach someone to design and evaluate good user interfaces without an aesthetic sense. The truth is that the nuances of human factors engineering are too involved and, in some cases, too ill defined to be practical when brought to bear algorithmically on a design problem. The human mind is an amazing thing and there is no substitute for an aesthetic sense. However, understanding the concepts of Human Factors engineering will lead to the development of an aesthetic sense and can refine and focus an already developed one. Chunking is probably the most fundamental and fruitful of these concepts to understand.

Stay tuned for Part II of this article, "Channeling," coming soon.

[an error occurred while processing this directive]