Symmetry for successful interactive systems

Harold Thimbleby
UCLIC, UCL Interaction Centre
26 Bedford Way
LONDON, WC1H 0AP
+44 (0)20 7679 5357
h.thimbleby@ucl.ac.uk

ABSTRACT
HCI has some rich and suggestive ideas, like affordance and direct manipulation. Abstract (not just geometrical) symmetry is a powerful explanation of why these concepts work, and it can be generalised to guide new design for more effective user interfaces. Symmetry makes user interfaces easier to learn, easier to use, and easier to program — and hence more reliable. Symmetry raises in very clear ways many design trade-offs. In particular, symmetry can be abused when it used to design only superficially symmetric systems, which may look good but are deceptive.

Keywords
Affordance, ambiguity, direct manipulation, illusion, objects, symmetry, user interface design, virtual reality.

INTRODUCTION
It seems self-evident that interactive systems of all sorts could be easier to use. Whether we are using consumer products such as photocopiers or mobile phones, or walk up and use systems, or bespoke products such as aircraft flight management systems…, there are always sources of user problem. Many problems seem, in hindsight, to be avoidable. The problems that, further, seem correctable are the ones we most regret and are perhaps the most motivating parts of the field of HCI.

Apart from the simplest of systems (such as light switches), interactive systems are far more complex than their user interfaces can directly reveal. Users' models of systems have to be simpler than the systems themselves. An inevitable consequence is that user interfaces are ambiguous. Users find out how to use ambiguous systems by interacting with them: the interaction provides the missing information. If this interaction is to provide useful knowledge, the system and the interface must be structured in such a way that the user gains insight. Conversely, humans are enormously complex (and situated in the world), and computers cannot model them without ambiguity. If interaction is to provide useful knowledge, the user and the interface must be structured in such a way that the computer gains insight.

Unfortunately interaction almost always changes the state of systems in unintended ways, and perhaps in ways that cause the user further problems. Users become reluctant to experiment to solve their problems, because experience shows that experiment may only make things worse.

In the physical world in which we live, things are very different. We learn that objects have natural ways to be used, and that rules of interaction once learnt are reliable. Interacting with objects (except in extreme ways) never breaks the objects or unintentionally changes them beyond usefulness.[1]

THE ROLE OF SYMMETRY IN DESIGN
Formal ideas of symmetry can be used directly in user interface design. Humans are very well adapted to detecting and exploiting particular sorts of symmetry (generally, ones relevant to the physical world): when these are used in interactive systems, users are enormously empowered. Many user interfaces that are "easier to use" map the fundamentally complex state space of the computer system onto simpler spaces that behave in simpler ways by embodying familiar physical symmetries.

The core role of symmetry can be expressed as design recommendations:

- When designing a new interactive system, find out what the symmetries of the user task and activity are.
- How can these symmetries be represented in the user interface?
- How can the user interface support the user interacting with objects without changing them, so that they can eliminate ambiguities and illusions?
- and so on, as this paper will elaborate.

Symmetry is also aesthetic, and it is tempting to make user interfaces look symmetric — regardless of their underlying structure. As a design recommendation, therefore, it is important to separate task symmetries from aesthetic presentation decisions. For example, it might look nice and symmetrical to have a row of identical looking buttons, but if the underlying system does not have the same symmetries, the visual symmetry is superficial and misleading. Users will succumb to capture errors [12] —

[1] Such operations typically generate a group. Although group theory is implicit in much of this paper, it has been written to be intelligible without recourse to abstract algebra, but see [10,21,22,23].

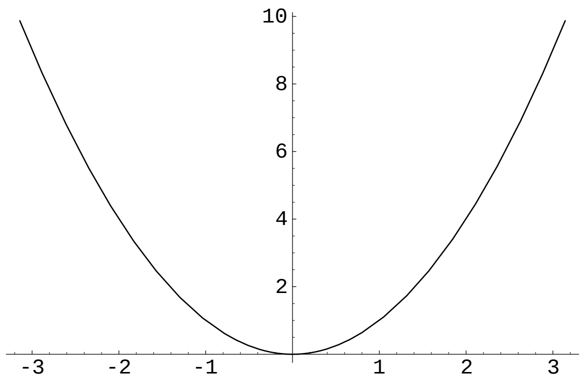

Figure 1. Squaring, a simple symmetric function about *x* = 0.

and in safety critical systems, users (and designers) would be well-advised to put different labels or markers or other symmetry-breaking features over the switches to reduce confusion. (See the section below, on ambiguity.)

BACKGROUND CONCEPTS

Affordance was introduced by James Gibson [7] to describe invariant properties objects have that encourage particular use. His ideas were part of a larger ecological psychology programme, but were picked up by Don Norman in his *Psychology of Everyday Things* [13] and applied to the design of physical artefacts, such as switches. In turn Bill Gaver [5] extended the ideas to user interfaces, and raised issues such as the affordances of buttons. In this form, the concept of affordance entered HCI lore, though it raises certain issues: a picture of a button on a screen can't successfully be pressed (unless it is a touch screen), so in what sense does it have an affordance to be pressed? What is important is that an observer readily perceives the affordances, and therefore believes that certain actions can be undertaken with the object. In the real world, typical affordances are represented by concepts such as 'graspable' and 'throwable.' In most user interfaces, objects look like they have affordances, and the user has to rely on many learned conventions. The affordances of an object are invariant over a wide range of observation conditions, and this is what interested Gibson. Even when objects are at strange angles, partly occluded, moving, and so forth, humans are very good at recognising their affordances.

Norman later revisited affordance and made some finer distinctions that had been glossed by practitioners [14]. Affordance is a controversial and sophisticated concept in psychology; here we are interested in it as an abstract concept. Whether Gibson had the right *psychological* explanation is irrelevant here; we need concepts that work for *design*.

Symmetry is a well-known concept [23], and often applied to visual pictures and patterns. An object or picture having mirror symmetry, for instance, means it looks the same if it is reflected in a mirror. Most generally, symmetry is an algebraic concept: objects have symmetries if they can be transformed and yet preserve certain properties. A simple example is a symmetric function f, where $f(x) = f(-x)$. Here the transformation of reflecting the coordinate system (x goes to $-x$) leaves the property f unchanged. If we draw the graph of such a function (e.g., the square function) it *looks* symmetrical: see Figure 1; however, in general symmetries need not have simple visual forms. In particular, in this paper we are interested in symmetries that apply to interaction, the transformation is therefore something the user does through interaction over time, and the preserved property or properties is something the user wants out of the interaction.

We will use the term *natural computation* to refer to whatever the brain and body (perceptual, motor, cognitive functions) do most effectively. Humans are clearly very flexible and powerful computers, but some things we do effortlessly — particularly in problems involving perception, visual recognition [4], spatial location — and affordance. For example, our perceptual system readily identifies pictures of buttons in large screens of other information, particularly if the pictures have depth features such as shading.

Natural computation is technically the term used for computation "in the world" [1] — such as the computation soap bubbles do in finding shapes of minimum energy. This use of the term is consistent with ours, although we are obviously more interested in how humans interact with computers than how soap bubbles might. So, in this paper, by natural computation we mean computation as is achieved by humans, and we are not committed to whether the computation occurs in brains, retinas, or anatomy or otherwise.

Natural computation does not make all things equally easy. Human natural computation is very specialised, and this must be borne in mind if design is to be effective. Friedhoff and Peercy [4] give some persuasive examples both of the power of natural computation and the ease with which apparently trivial changes in representation can defeat it. Reeves and Nass [16] gives some examples of how natural computation applies also to certain social situations; see also [6]. Byrne [3] gives an excellent account of cognitive models applied to conventional HCI.

SYMMETRY, AFFORDANCE AND INTERACTION

Elsewhere I pointed out [21,22] that symmetry captures a key concept in affordance. Specifically, humans recognise symmetries in objects (in the general algebraic sense, not just simple mirror and rotational symmetries), and particular symmetries define affordances.

A simple example will illustrate these ideas: a knob that has an affordance for turning has rotational symmetry. Indeed, if the knob had no rotational symmetry it would be impossible to turn (though it might be attached to an axle which had hidden rotational symmetry that enabled it to turn). Humans may learn that pointer knobs, which have no externally obvious rotational symmetry, can indeed be rotated. When such a knob is rotated, it does not break — it is still the same knob. Its 'knobness' property is unchanged through rotation.

Symmetry, then, formalises at least part of affordance. Affordance works because humans learn to (and have evolved to) associate symmetries with perceptions. Since symmetry is a very general concept, we need natural computation to filter the symmetries that are salient (i.e., ones which a user might detect easily, preferably 'automatically') from ones that are technical symmetries that are present but which would take a user too much, or distracting, effort to detect. If a user has to get into formal reasoning to understand a design, then they are not using natural computation directly — they are using it to create a virtual machine to do (typically conscious) reasoning. Inefficiency cannot lead to an affordance. Such inefficiency is also the source of the user pursuing conscious goals — of course, that is an essential part of interaction [11] — but it can only happen reliably when the user interface works the right way, at the level we are focussing on in this paper.

Symmetry in design

Symmetry allows affordance to be used constructively in design. An ordinary pencil has various affordances, and if we allow that people learn, it has affordances for gripping and writing. Furthermore, a pencil has various symmetries. Since pencils are cylinders (typically circular or hexagonal) they have two major symmetries:

- Rotational symmetry about their long axis. This symmetry allows a pencil to be grasped at any angle, and it will work just as well. The flexibility endowed by the symmetry makes a pencil easier to use than, say, a fountain pen.
- Mirror symmetry end-to-end. This is a partial symmetry for most pencils, since only one end is sharp. This near symmetry makes a pencil harder to use: a user might mistakenly pick it up with the wrong orientation. This near symmetry is motivated more by aesthetics than the needs of writing.
- A child's pencil may further break symmetry, for instance having a strangely shaped writing end. The intention is to train the child to hold pencils properly. Here, the goal is not to have an easy-to-use writing instrument, but to train the user. As with any design principle, use of symmetry has to be traded-off against other priorities.

From this small example, the relations of symmetry, usability and affordance are clear. Exact symmetries make objects easier to use, whereas partial symmetries tend to make objects misleading and harder to use. A poignant example of this was a death caused by a partially symmetric X-ray [2].

Consider, next, a clock. Time has cyclic symmetries. Today at 12.30 is the same as tomorrow at 12.30 (on a clock, maybe not on a calendar). The only physical way to represent general cyclic symmetries is in rotational symmetry: hence analogue clock faces. Reference [21] develops this example to alarm clocks and to digital alarm clocks, both interactive forms of clock.

The problem with these initial examples is that they are simple; indeed they are hindsight. This paper now turns to providing more substantial examples, interleaved with developing the rationalisations.

Direct manipulation

Prior to the invention of direct manipulation (DM), general purpose user interfaces were textual and command based. Command languages have various usability problems, particularly in how they handle user errors. DM removes problems of textual command based interfaces and provides something far more natural and easy to use.

Direct manipulation user interfaces have been very successful, and this is obviously because objects are represented on the screen so that they seem to have affordances that suit their manipulation directly by moving the pointer (typically, a mouse).

DM is part of a larger complex that has many other features [8,9,17], such as requiring rapid response to interaction. Symmetry is relevant in several ways:

- When objects are dragged around the screen, they are still "the same" objects. The standard DM operations — selecting, dragging — are transformations that preserve properties: in other words, symmetries.
- The 2D symmetries that a user can experience on the screen (and with the mouse) are familiar, indeed natural, to users because these symmetries apply in the real world.
- Some essential features of DM (such as reversible and incremental actions) relate to the formal ideas of symmetry (i.e., group actions).

In short DM works because the complicated and generally arbitrary state space of the program is projected into a space with natural symmetries, namely a 2D Euclidean space. Another important feature is that DM factors the projection into objects. Each object is projected in to the 2D space independently. This orthogonality is crucial to DM's ability to scale up gracefully (useful for file system interfaces): user tasks involving multiple objects only become linearly more complex at the user interface, whereas in conventional user interfaces there would be a combinatorial explosion that the user had to deal with.

The human perceptual system is naturally attuned to recognising the visual representation of objects. In DM, it takes essentially no effort to perceive objects and to transform them. This is very different from, say, text-based interfaces, where a user has to consciously ‘work out’ what is what.

Conceptually, DM is hard to implement, but fortunately it lends itself to object oriented programming. Programmers can deal with representations of first class objects explicitly, and the programming paradigm hides how to project those objects onto the user interface. DM systems tend to be more reliable than other approaches because of this hidden support — bad programmers cannot easily get the crucial user interface code wrong!

AMBIGUITY

Systems may be unusable because they are slow, because they do not support the design requirements, because they are designed in a way that seems to encourage errors… and for very many other reasons. A large class of fixable problems occurs where the user does or starts some action other than the optimal action at that point in an interaction. A special case of an action is of course where the user does nothing, perhaps believing that this is the best that can be done, when in fact the system has various relevant options that the user has not tried.

In this particular sense of unusability, an unusable system is one that is ambiguous. Ambiguity allows the user to do various things, but somehow or sometimes the user does the ‘wrong thing’ (which can include doing nothing).

Ambiguity has been heavily studied. From a user-centric perspective, it is important to distinguish between slips and mistakes [15]: different sorts of error arising through either wrong actions or wrong intentions. From a system-centric perspective, two quite separate ideas have been conflated as *mode*. In one view, a mode is an interpretation of the user’s actions: if the system is in the wrong mode, the correct actions will have undesirable effects. A simple example is the modes that windows create: a user may ‘type into’ the wrong window thus with unintended effects [19] — whether this error occurs because of a ‘slip of the mouse’ or some deeper intentional error is irrelevant to the classification. The other meaning of mode is a context that forbids alternative action: hence the phrase, “don’t mode me in” [18] — dialog boxes that *have* to be attended to are examples of modes in this sense. Modes in the former sense are examples of ambiguity; modes in the latter sense are where the user wants control over ambiguity!

Optical illusions

It is very hard to be abstract in HCI, since any illustrative example is so interesting in its own right that the abstract ideas easily get lost in the detailed possibilities that relate to the concrete example. To take a step back, then, we will briefly consider another source of ambiguity: optical illusions.

The familiar 3D world does not offer many clear examples of ambiguity;[2] but we are very familiar with 2D representations of 3D, which do. When 3D objects are projected onto 2D, information is thrown away, and inevitably some images can be interpreted in more ways. Our 3D “perceptual instruction set” makes errors when only provided with 2D information. Many optical illusions follow.

Illusions (of any sort) are perceived discrepancies from the truth, and — by definition — very rarely do we notice illusions except when they are pointed out to us. A wide range of two dimensional optical illusions have been devised, and we have become quite familiar with many — so much so that we often identify illusions because we have seen them before, not because we fall again for the perceptual confusion.

Briefly, optical illusions are perceptions about figures that can be changed by viewing the figure differently. In particular, using a straight edge or a measuring ruler can often provide persuasive evidence that the appearance of the figure is deceptive. Since we know straight edges are always straight and rulers always of the same length, if they give unexpected measurements in different parts of a figure then our perceptions must be at variance.

In some illusions, the ambiguity is essentially artificial (e.g., the Muller-Lyer illusion; see Figure 2). The image is objectively unambiguous, as it were, but it appears different to what it ‘really’ is, as revealed by interaction with more reliable instruments such as rulers. Other illusions are deliberately ambiguous (e.g., the duck-rabbit) — in 3D reality, one could easily tell the difference by going up to the supposed duck-rabbit and offering it a lettuce. (The psychological processes relevant in each case are very different, but that is beyond the scope of this paper.)

We circumvent the impasse of illusion by interacting with the illusion. On future encounters with the illusion, we no longer unwittingly experience the ambiguity. We say the lines *are* the same length, despite appearances, or the figure is “a duck-rabbit,” or whatever. Perhaps there are too many perceptual effects called illusion to generalise too easily, but the key lesson is:

Ambiguity is resolved by interaction

Gibson stressed this: if a user is forced to sit passively, almost any image will be ambiguous; if the user is free to move and interact, changes in sensations will be tied to the movement. In the Muller-Lyer case, interaction is a trivial matter of using a ruler. There is an important caveat, which applies to HCI too: ambiguity can be resolved by learning as well as or instead of by doing. There may be other ways

[2] Unless you are trying to explain 4D or more complex objects — er, like computer programs.

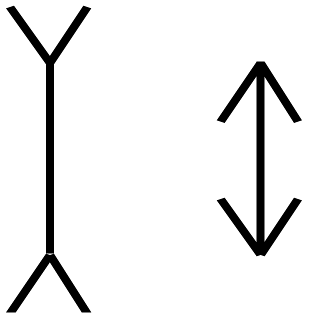

Figure 2. The Muller-Lyer illusion. Notice how one vertical line looks longer than the other; measurement will reveal they are the same length.

of establishing what is going on than by direct interaction, such as imagining what is happening, reading a good user manual, and so on.

To summarise: very many illusions can be explained by arguing the human visual system is designed for three dimensions. Two dimensional figures create illusions when they trigger ambiguous recognition of fragments of three dimensional images. In the 3D world, we can walk up to an object and check it: our embodied visual system is not fooled. In the 2D illusion, we have to decide whether the image is a projection from 3D or an abstract 2D image. Crucially, as we change our point of view, we get different perspectives, but neither in 2 or 3D does the object itself change. Hence, to disambiguate visual images, we rely on symmetry.

ABSTRACT SPACE

We live and interact with objects in a 3D Euclidean space.[3] Evolution has endowed us with impressive perceptual, cognitive and other skills to cope with and manipulate 3D objects and events. Optical illusions give us one example of how presenting a 3D world in a lower dimensional form creates ambiguity (either directly or because perceptual systems evolved in 3D are confused), which can only be resolved — if at all — by some sort of interaction, such as employing a ruler or moving in 3D. In either case, we rely on the object not changing as we change our point of view: we rely on symmetry.

Computer systems operate in enormously complex discrete spaces, and we see them through user interfaces of much lower dimensionality. The 2D screen of most computer systems has far fewer dimensions than the dimensions of the space it represents. Inevitably "optical illusions" follow: images on the screen have multiple meanings and users may misinterpret them. (cf. Modes arise when the same appearance can mean several things.)

[3] We are in space-time too with information spaces, such as sound and other modalities, but this is not a paper on physics.

As with optical illusions, to reduce ambiguity, we need to interact. Unfortunately, most times when we interact with computers the interaction changes something else. Far from resolving ambiguity, we suffer from side-effects of our interaction.

Ambiguity can only be resolved by interaction that preserves the invariants that were represented ambiguously.

One of the key reasons why computer systems are hard to use is that they do not provide natural ways of interaction that allow users to resolve ambiguities without side effects. Whereas a typical perceptual illusion can be resolved by walking around it, or getting out a ruler, interacting in even quite trivial ways with a computer makes the original confusing state of affairs unreachable before we even know what it was.

Direct manipulation revisited

In DM, the complex multidimensional space of a running computer program is projected onto two 2-dimensional spaces: the screen and the mouse surface. Once the user has become accustomed to the translation of the mouse space into the screen space — which is peculiar particularly at its boundaries when the mouse runs into things and cannot move further — there is an almost natural 2D space to interact in.

Direct manipulation is very successful for implementing office and bureaucratic tasks, because these tasks are familiar to users in 2D spaces — namely as bits of paper. Folders or directories are more problematic because they do not have traditional 2D representations: the representations of folders are typically ambiguous.

(*i*) *We expect illusions*. Unless the program is running a 2D (or less) model, it cannot be projected into two dimensions without ambiguity. (All sorts of visual conventions are created to help reduce ambiguities; scrollbars being an example — we cannot project the entire text of a document onto a small screen and still be legible, so scrollbars are used to at least indicate where in the document the projection is coming from.)

(*ii*) *We expect ambiguity to be resolved by interaction*. The user may be unsure whether an icon represents a file or a folder. A disambiguating interaction is to first try dragging something onto the icon. If the icon highlights, it is a folder (and if released the other object would relocate within it), otherwise it is a file. There are typically other disambiguating interactions, such as invoking a 'get info' command on it. Conversely, just looking at the screen without interaction is not going to disambiguate the icon; interaction is essential.

(*iii*) *We expect the user to detect 2D invariants easily*, because the user has evolved in the natural 2D/3D world. For example, when objects are moved or rotated in 2D space they are 'the same thing' (an invariant), and this is easily recognised. DM exploits such invariants, both directly and subtly. Directly, when an icon is moved the perceptual system does some tricky visual computing and recognises it is the same object as it is moved around the screen space. More subtly, when the user tries to move a complex object the computer may have been programmed to move only an outline, since this is more efficient. Now there are two objects corresponding to what the user has selected, but the user is not confused because this invariant (single things stay single things) is so rarely broken in the real world, it is easy to ignore.

(*iv*) *We expect some application and tasks to map cleanly onto 2D*. Thus drawing and painting programs were some of the early and most persuasive applications of DM. Office applications are typically based on pieces of paper: paper is a natural 2D medium, and once scrolling is sorted out, it maps cleanly to the 2D screen. Desktops, too, are 2D.

Overlapping in 2D is sometimes called 2.5D, and this could be discussed further— for instance, elaborating the way natural computation can detect occlusion and overlapping easily. The key point is that 2.5D is still a lower dimensionality that most applications require, and the discussion about DM (which was restricted to 2D) applies.

Unresolvable ambiguity requires redesign

Suppose we are using a video recorder to record a favourite programme, and we are watching it on the TV. My video recorder does not say what channel it is recording, nor does the TV. One way to check whether the TV is 'watching' the channel the VCR is recording is to stop the VCR recording and try changing the channel: if nothing seems to happen on the TV, then the TV is tuned to another channel. There is an ambiguity, but the interaction to disambiguate the state has the unfortunate side-effect of guaranteeing the VCR would not be recording what we want!

You might think that we can change the TV channel to confirm what the VCR is recording. However, without the remote control, we can only increment or decrement the channel — it cannot be set to a particular channel directly (in this case, we'd like to be certain it is set to watch the VCR channel). If you try changing the channel, you just get another programme — which is what happens regardless of whether or what the VCR is recording. In this case, we can interact, and the interaction has no unintended side-effect on recording, but it does not tell us anything that helps disambiguate the problem.

The solution, of course, is to redesign the VCR user interface. It should display the channel number it is recording. Or the TV display the channel number (perhaps on demand); or the TV should have buttons that select specific channels…there are many solutions. Another problem, again clearly expressed as a failure of symmetry in the design, is that the user interface on the TV is different from the user interface on the remote control. The laws of interaction on the remote should apply on the TV too: if they did, the TV would not just have had incremental channel change buttons, and the problem would go away.

The value of symmetry is that it has provided a concise, analytic description of a problem (here necessarily hindsight, but with some sense that it might have anticipated the problem during the design process), with hints of solutions.

- the user interface is ambiguous;
- it does not have symmetries for the user to resolve the ambiguities conservatively (without side effects).

How a designer chooses to proceed is another matter; that they *do* proceed because they *know* there is a problem to fix is what is important for improving the user interface.

Digital alarm clocks

Setting an alarm clock requires checking (or believing) that the clock is correctly set. In many alarm clocks, it is very difficult to be certain.

There is no simple experiment that can be undertaken to check how to set an alarm, or to check what time (e.g., AM or PM) the alarm has been set to. One way to check might be to set the alarm (hopefully), then change the time of the clock to a few minutes before the alarm time. Unfortunately, many alarm clocks reset themselves after ringing, so this process may have to be done more than once so that the user can learn the correct procedure — to use it one final time that is not checked (until the alarm rings or fails to ring the next morning).

The main problem is that most digital alarm clocks project the time and alarm state spaces, that is, two times, onto a single time display. Only four digits are used, when eight are needed to be unambiguous.

If enough digits cannot be used (for cost or space reasons) then the solution is to make it very clear to users which number is being displayed.

Small screen browsing

DM (or graphical user interfaces more generally) was a key reason for the popular uptake of the world wide web. Window-based browsers made interaction with the internet more natural. For example, multiple connections — which would be very confusing using text based interfaces — map nicely to separate windows. Even though windows may overlap, the human visual system is adept at recognising the 'top' and therefore active window. (We are very familiar with the style of interaction, and it does not need spelling out further here.)

The success of browsers encouraged moving the interface to the small screen, and the design of a new protocol WAP. However, WAP was a usability failure, particularly in comparison to the phenomenal success of SMS and conventional browsing. One major reason for the failure was that the small screen eliminated natural visual concepts like overlapping windows. Natural computation was no longer used to support complexity; simultaneously the user had additional burdens, such as increased demands on scrolling.

In short, WAP tried to retain the sophistication of desktop browsing but through a smaller screen, eliminating windows and DM, while making the user interface even less able to exploit natural computation. (Most WAP devices also eliminated the mouse and its 2D plane.)

The design solution would either have been to make the business and social use models *much* more attractive to compensate; to reduce the ambiguity in the user interface; or to increase symmetries, and hence increase ways in which the user can safely interact to disambiguate. Unfortunately, the network model of WAP ensured that all interaction was at direct cost to the user, a further disincentive to use.

Microsoft Word

This paper was written using Microsoft Word. The paper has two figures, and as the text of the paper was edited in Word, the figures jumped around unpredictably, moving horizontally and vertically, even partially off the page on some occasions. As the user interacted with Word, the transformations to the text did not leave properties of the figures (their position or relative position) unchanged. A symmetry that would have made Word easier to use has not been implemented.

Word is an example of a so-called WYSIWYG editor: what is on the screen is what is printed [19]. Actually, no. What should have been a simple symmetry (transform the space from screen to paper, leave layout unchanged) fails in numerous ways. Also the 2D Euclidean symmetries of paper are not present in the text editor: cursor keys do not move in a 2D space, as one might have expected [19].

When text fields are edited in Word (e.g., in forms), the details of interaction are very different than when editing other types of text. For example, it is not possible to select continuous fragments of text using the mouse, as one can do elsewhere. The obvious symmetry, namely, “rules of editing are the same in all contexts,” has not been implemented. Curiously, Word must be a larger program because of the *ad hoc* ways in which it has been implemented: more symmetries in its user interface would have made the program smaller and easier to manage.

Unfortunately the user manual for Word does not describe these differences in text editing in various contexts (body text, fields, tables, equations, footnotes, bulleted lists, dialogue boxes, and so on, as well as the various side-effects of autoformatting affecting editing rules — all of which are subtly different). The documentation gives a misleading impression of Word’s symmetries. As we saw with the pencil example, this appearance of symmetry inevitably encourages errors and makes Word harder to use than it need have been.

Virtual reality

Virtual reality (VR) is, more or less, a 3D version of DM. The extra dimension has considerable advantages: primarily VR represents the very space we live in, and therefore there is no unusual ambiguity in VR. VR exploits natural computation directly; it is therefore ‘immersive’ and this in itself distracts users from noticing problems with interaction. After all, reality does not have problems; 3D *is* reality!

SYMMETRY FOR COMPLEX SYSTEMS

When complex systems can be projected onto familiar structures (e.g., with Euclidean symmetries), all well and good. Such user interfaces, particularly using DM and VR, will seem natural. But it is not always possible to do this; in fact it is often misleading to use the success of DM or VR to justify merely simulating existing physical objects (such as on-screen calculators mimicking handheld calculators [20]; or mobile phones mimicking desktop browsers) rather than to trying to do better.

If users are constructing new models and are less able, therefore, to exploit natural computation, then these models should be small and powerful. The smaller and more powerful, the less effort the user will have to expend in constructing the model.

Symmetry helps here too. If a user interface has a complex state space, the user’s model (if adequate) will also be complex. But suppose the state space has abstract symmetries — not necessarily ones with any physical realisation: then interaction with the system will leave some properties of the system unchanged. Now these properties do not need to be learnt everywhere, and the user can make savings.

Here is a simple (non-interactive) example: if a user wanted to learn a picture, if the picture had mirror symmetry, they would only need to learn half of it plus the fact of its symmetry. In general, then, symmetry compresses the description of systems. Thus the trivial symmetric function f of Figure 1 can be tabulated knowing only $f(x)$ for $x \geq 0$ and that $f(x) = f(-x)$. Explicit details of $f(x)$ for all x are not needed. One symmetry in this case halves the knowledge load to learn the relevant facts.

Symmetry not only makes systems easier to use (in resolving ambiguities, and most effectively when the symmetries are natural) but it also makes systems easier to model — easier to learn, easier to understand, easier to remember — because the complete, correct models are smaller. Symmetry also makes systems easier to program and debug, so it helps the user indirectly by making systems more reliable.

THE COMPUTER IS A USER TOO

The projection from the computer’s internal state space to the user interface should be done to enforce certain natural symmetries. But this is only part of the story. The user has a complex state space in their head (which of course has access to the rest of the world through the senses, body and memory). Somehow the interaction in the interface has to "make sense" in the user's model. Or, perhaps, if we were to be less conventional we could invert all the discussion to this point and wonder how the computer interacts with the human: the human's complex state space has to be projected onto the user interface so that the computer can interact with and understand the human. Computers suffer from illusions and mistaken beliefs about users, just as users do about computers. The solutions are the same in both cases. Computers need to interact with humans to disambiguate their illusions — indeed, there is a pleasing symmetry!

Because symmetry is an abstract mathematical concept, it transcends the usual computer-based approach *versus* human-based approach dichotomy. Any effective symmetry in the user interface is just as much in the user's head as in the computer's program. If in the interface, it should also be in the user manual, and — as we have emphasised, it should also be in the user's original task.

CONCLUSIONS

It took artists millennia to understand how to draw representational 3D images in 2D. The discovery of perspective took a surprisingly long time; it was a tremendous achievement for the renaissance artist Brunelleschi, but now is taught in primary schools. For centuries nobody had thought that pictures could be what today we call realistic. What today we think of as 'flat' would have been what everybody thought pictures *were*. Similarly, today's computer systems have all the naïvity that early paintings have: they don't represent reality well (even when they are trying to). In our terms: they have all sorts of broken invariants that have not been properly projected onto the user interface to fit with the reality they implement. Perhaps there is some future style of interaction that will be so natural that today's interfaces will seem unnecessarily quirky.

Useful computer programs are enormously complex, and indeed far more complex than 3D scenes. Programs have to be represented in much lower dimensional spaces to be usable, and this creates massive problems for user interface designers — and we do not even yet have concepts like "perspective" to help.

Necessarily, user interfaces create ambiguities or illusions. When users misinterpret illusions, things go awry. To disambiguate, users must be able to interact, and yet the interaction must not disturb the user's goals. This requirement can only be achieved when there are appropriate symmetries (by definition). If these symmetries are also natural ones (e.g., ones users are familiar with from the physical world), then user interfaces become very much easier to use.

Exploiting affordance, direct manipulation, virtual reality, and reducing ambiguity are quite general and informal design heuristics. Symmetry is a deeper concept and one that can be rigorously used, and has wider application. However, symmetry can also be abused in design: things may be made symmetric because it is aesthetic to do so — but the semantics of the underlying system may not have the symmetries suggested by appearances. To be successful, symmetry must be used as a deeper and consistent algebraic concept: it is not just about the user interface and its appearance, but applies from the internal workings of the program through to the workings of the user's model and beyond.

Finally, although further work is needed, symmetry can be formalised, and it can be formalised abstractly — which makes it a very versatile approach. Different parts of interactive systems will have different sorts of symmetry, and the formalisation enables designers to consider explicit trade-offs rigorously.

ACKNOWLEDGMENTS

Harold Thimbleby is a Royal Society-Wolfson Research Merit Award Holder, and gratefully acknowledges their support in this work. The author is grateful for excellent feedback and ideas from Ann Blandford, George Buchanan, Michael Harrison, Matt Jones and Prue Thimbleby.

REFERENCES

1. Ballard, D. H., *An Introduction to Natural Computation*, MIT Press, 1997.
2. de Bruxelles, S., "Kidney Death Surgeon 'Reversed X-ray'," *The Times*, June 14, p13, 2002.
3. Byrne, M. D., "ACT-R/PM and Menu Selection: Applying Cognitive Architecture to HCI," *International Journal of Human-Computer Stu*dies, **55**, pp41–84, 2001.
4. Friedhoff, R. M. & Peercy, M. S., *Visual Computing*, Scientific American Library, 2000.
5. Gaver, W. W., "Technology Affordances," *ACM CHI'91 Conference*, pp79–84, 1991.
6. Gaver, W. W., "Affordances for Interaction: The Social is Material for Design," *Ecological Psychology*, **8**(2), pp111–129, 1996.
7. Gibson, J. J., The Ecological Approach to Visual Perception, Houghton Mifflin, 1979.
8. Hutchins, E. L., Hollan, J. D. & Norman, D. A., "Direct Manipulation Interfaces," in Norman, D. A. & Draper, S. W., *User Centered System Design*, Lawrence Erlbaum Associates, 1986.
9. Jacob, R. J. K., "A Specification Language for Direct-Manipulation User Interfaces," *ACM Transactions on Graphics*, **5**(4), pp283–317, 1986.

10. Johnson, D. L., *Symmetries*, Springer, 2001.

11. Nardie, B. A. editor, *Context and Consciousness*, MIT Press, 1997.

12. Norman, D. A., "Design Rules Based on Analysis of Human Error," *Communications of the ACM*, **26**(4), pp254–258, 1983.

13. Norman, D. A., *The Psychology of Everyday Things*, Basic Books, 1988.

14. Norman, D. A., "Affordance, Conventions, and Design," *ACM Interactions*, **6**(3), pp38–43, 1999.

15. Reason, J., *Human Error*, Cambridge University Press, 1990.

16. Reeves, C. & Nass, B., *The Media Equation*, Cambridge Univesity Press, 1996.

17. Shneiderman, B., "Direct Manipulation: A Step Beyond Programming Languages," *IEEE Computer*, **16**(8), pp57–69, 1983.

18. Tesler, L. "The Smalltalk Environment," *BYTE*, **6**(8), pp90–147, 1981.

19. Thimbleby, H., *User Interface Design*, Addison-Wesley, 1990.

20. Thimbleby, H., "Calculators are Needlessly Bad," *International Journal of Human-Computer Studies*, **52**(6), pp1031–1069, 2000.

21. Thimbleby, H., "Affordance and Symmetry," in Johnson, C. editor, *Interactive Systems: Design, Specification, and Verification*, 8th. International Workshop, DSV-IS 2001. Lecture Notes in Computer Science, **2220**, pp199–217, Springer Verlag, 2001.

22. Thimbleby, H., "Reflections on Symmetry," *Proceedings of Advanced Visual Interfaces*, AVI2002, pp28–33, 2002.

23. Weyl, H., *Symmetry*, Princeton University Press, 1952.