

VisualIDs: Automatic Distinctive Icons for Desktop Interfaces

J.P. Lewis*
CGIT Lab
U. Southern California

Ruth Rosenholtz
Perceptual Science Group
Massachusetts Institute of Technology

Nickson Fong
ESC
Entertainment

Ulrich Neumann
CGIT Lab
U. Southern California

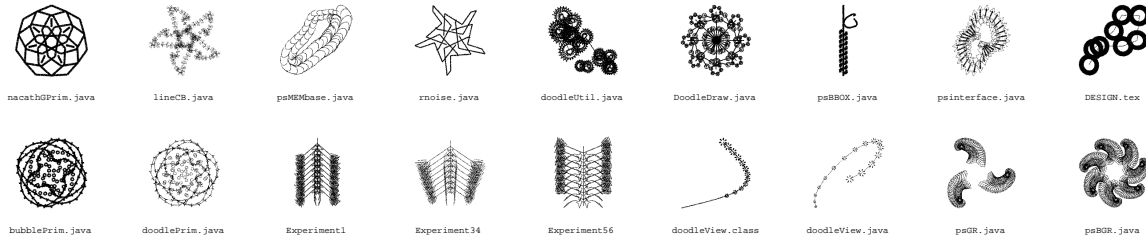


Figure 1: “Doodle” style file icons with name clustering.

Abstract

Although existing GUIs have a sense of space, they provide no sense of place. Numerous studies report that users misplace files and have trouble wayfinding in virtual worlds despite the fact that people have remarkable visual and spatial abilities. This issue is considered in the human-computer interface field and has been addressed with alternate display/navigation schemes. Our paper presents a fundamentally graphics based approach to this ‘lost in hyperspace’ problem. Specifically, we propose that spatial display of files is not sufficient to engage our visual skills; *scenery* (distinctive visual appearance) is needed as well. While scenery (in the form of custom icon assignments) is already possible in current operating systems, few if any users take the time to manually assign icons to all their files. As such, our proposal is to generate visually distinctive icons (“VisualIDs”) *automatically*, while allowing the user to replace the icon if desired. The paper discusses psychological and conceptual issues relating to icons, visual memory, and the necessary relation of scenery to data. A particular icon generation algorithm is described; subjects using these icons in simulated file search and recall tasks show significantly improved performance with little effort. Although the incorporation of scenery in a graphical user interface will introduce many new (and interesting) design problems that cannot be addressed in this paper, we show that automatically created scenery is both beneficial and feasible.

CR Categories: H.5.2 [Information Interfaces and Presentation]: User Interfaces—Graphical user interfaces

Keywords: Information retrieval, pattern synthesis, psychology

1 Introduction

Scenery is fundamental to our environment and is associated with a robust set of skills. If you visit a city that you last visited a decade

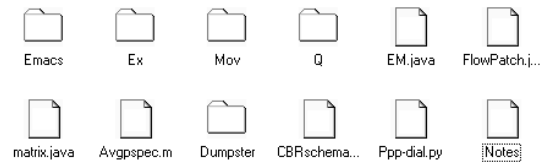


Figure 2: Caricature of indistinguishable icons in current GUIs.

ago, chances are that you still recognize parts of the city. Similarly, learning the appearance of a book is not something that requires hours of memorization and rehearsal, it takes only moments, and we often remember the appearance even when the title is forgotten.

A common problem across desktop metaphor computer interfaces, hypertext systems, and virtual world interfaces is that these systems do not engage our visual skills – multiple studies have found that users have trouble establishing a sense of place in these systems [Nielsen and Lyngbaek 1989; Darken and Sibert 1996]. Many people have trouble locating files in systems they visit regularly; one empirical study [Barreau and Nardi 1995] describes someone trying to find a file that they had created earlier the same day!

This is not surprising considering that computer GUIs provide us with little or no *scenery*: many or even most icons are indistinguishable (Fig. 2). The word “scenery” is used here in a general and perhaps abstract sense:¹ *scenery is the distinctive visual appearance of objects in a place* that allow us to recognize those objects and that place. Our premise is that fully engaging our visual brain for recognition, search, and navigation tasks requires *both*

1. a spatial data layout, and
2. the distinctive appearance (scenery) necessary to recognize objects and places.

Current interfaces provide only the former.

While some data types already have a natural visual representation (e.g. thumbnails for image data), many other types of data do not have an obvious distinguishable visual representation. And while current operating systems allow users to manually assign icons to

*zilla@computer.org

¹This definition is similar in spirit to that used in one dictionary [Word-net]. Other terms including “space”, “place”, and “appearance” have their standard meaning.

file data, few if any users allocate the time to assign icons to all their files. Automatically assigned scenery is therefore needed for most files. In this paper we explore a particular kind of automatic scenery, algorithmically generated abstract icons, or “VisualIDs”.

The remainder of this section reviews evidence that human visual-spatial skills should be exploited in computer related activities. Section 2 discusses design requirements for a scenery creation procedure. Section 3 describes an icon creation algorithm that satisfies many of the stated requirements. Section 4 shows results including experimental demonstrations that the generated icons easily accelerate simulated file search and recall tasks.

1.1 Psychology of Visual Search and Memory

Recognition of place, navigation, and formation of cognitive maps are among the fundamental activities for any creature, and so it is not surprising that humans have considerable skills in these areas. For example, subjects shown collections of hundreds or thousands of images are able to recognize the previously shown images with accuracies of 90% and more after only one brief viewing [Standing and Haber 1970]. Memory for briefly shown pictures is greater than that for words [Shepard 1967], and searching for a picture of a particular object among many is also faster than searching for the name of that object among other words [Paivio 1974] (this fact is central to our work). More generally the availability of rich visual cues in the environment is believed to help in the formation of memory and play a role in developing cognitive maps [Searleman and Herrmann 1994]; conversely, navigation in environments without distinctive visual appearance is difficult (parking garages with several identical levels are one common example).

Studies of office workers provide evidence that humans often *prefer* to use their visual skills in organizing their work. Although haphazard “stacks” of papers would seem to be an inefficient organizational strategy, stacks allow search based on the remembered location and appearance of documents, rather than relying on their more easily forgotten textual labels [Malone 1983; Lansdale 1988].

The relevant literature on visio-spatial skills in computer settings spans several decades and includes studies involving distinct skills including visual search, learning, memory, and navigation. This literature is too large to survey here; we will only mention several examples of relevant research. Visual data displays have been found to result in accelerated retrieval relative to list-based displays [Hightower et al. 1998; Robertson et al. 1998]. Interestingly, people often prefer to conceptualize data in visual terms even while using applications that do not display data in a visual-spatial form, using phrases such as “look for” and “wander around” [Maglio and Barrett 1997].

On the other hand, at least one study has found that the skills that people command in physical settings do not always transfer to computer filing systems [Jones and Dumais 1986], calling into question the validity of adopting spatial metaphors for data. From our point of view this study is consistent with the hypothesis that generic object appearance is not sufficient to engage our visual skills and that distinctive appearance is needed. Indeed, one study found the addition of generic landmarks (colored cubes) did not help performance in virtual navigation whereas more detailed (but out-of-context) landmarks (models of cups, clocks and other objects) did help [Ruddle et al. 1997].

In current operating systems most file types have distinct icons but all files of a type share the same icon. A study of such per-file-type icons [Byrne 1993] proposes and gives evidence for a two-stage search process, in which files of the desired type are first located visually, and then the particular file among these is found by filename.

It was experimentally found that easily discriminable icons allowed the fastest visual search (consistent with our premise), although the discussion incorrectly equates discriminability with simplicity – the two factors are certainly independent, for example, simple but indistinguishable icons are obviously possible, c.f. [Ruddle et al. 1997].

1.2 Prior and Related Work

The human-computer interface community has identified navigation in virtual worlds as one of the major problems of the field. This *lost in hyperspace* [Mukherjee 1999] problem has been addressed through a number of alternative interfaces that offer improved navigational mechanisms. Problems of impaired wayfinding and place blindness have been addressed most often by *focus+context* approaches, in which a warped view differentially enlarges the focus region while displaying much or all of the entire space at a smaller scale, thereby establishing the spatial context. These approaches include both distorted presentations such as fisheye views [Leung and Apperley 1994] and the use of simulated perspective to diminish distant objects [Mackinlay et al. 1991]. Sophisticated navigation does not entirely solve the problem however. If the data space is both large and relatively featureless, then showing the surrounding context may not entirely eliminate place blindness [Mukherjee 1999]. This problem is only exacerbated in fisheye and other distorted views wherein a changing focal point can result in large or even drastic changes to the display.

Several systems have had success with adding extraneous distinguishing elements such as region boundary geometry specifically for the purpose of reducing place blindness [Ingram and Benford 1995; Robertson et al. 1998]. Another interesting proposal is *appearance from history* [Hill et al. 1992], by which the appearance of objects will change slightly according to their access and edit histories. Although this provides useful information, the mechanisms that have been proposed (slight changes in color, histograms of the number of edits for each line in a file) are not intended to visually distinguish large numbers of objects.

A system designed explicitly to utilize spatial placement memory has been empirically shown to result in faster and more accurate search relative to a commercial program with a list-based interface [Robertson et al. 1998]. On the other hand it was found that document thumbnails were no more effective than generic icons in a recall test [Czerwinski et al. 1999] – thumbnails are mainly effective for image data. The role of simulated location and context has been explored in Infocockpit, a multiple-monitor and multimodal interface testbed [Tan et al. 2001]. Subjects memorized word pairs shown at various points on a three-monitor display, and in one condition a “context” was created by projecting a photograph (e.g. of a museum interior) as a panoramic backdrop in the test room. The panoramic context greatly aided later recall [Stefanucci and Proffitt 2002].

In the graphics community, Pad [Perlin and Fox 1993] and the earlier Spatial Data Management System [Donelson 1978] are particularly complementary to our purpose. Pad situates data on an infinite virtual landscape that invites full use of our spatial navigation skills, and uses graphics hardware to provide facile navigation. The success of purely visual navigation and search (as opposed to computer-aided search by name or content) will require a rich set of distinguishable landmarks, however: navigation over thousands of files is superfluous if most files look the same and particular files can only be located through a search-by-name dialog. Indeed, several systems have abandoned an initial large workspace in favor of multiple smaller desktops in order to give a better sense of place [Card et al. 1991]. The introduction of scenery can alleviate this problem and allow use of a single consistent workspace as intended.

From this review it is clear that many current and proposed interfaces target our strong visual skills. On the other hand, the experimental evidence shows that users often do poorly in virtual settings despite having these skills. Our contributions are to point out that solving the ‘lost-in-hyperspace’ problem is not only a matter of providing improved navigation mechanisms (distinctive appearance is required as well) and to demonstrate that effective “scenery” can be automatically generated. Although the proposal to invent visual identifiers raises new issues in areas including usability and aesthetics that are not addressed in the paper, the experimental results demonstrate that significant improvements to data search and recall are available even without fully exploring these issues.

2 Requirements for “Visual Identifiers”

Some questions need to be addressed before proceeding: what sort of visuals are suitable for making data distinguishable, how are they generated, and how are they associated with the data? The following are desirable characteristics of visual data identifiers:

Persistence. Objects should retain a fixed location and appearance until moved by the user.

Identifiers should be automatically assigned (with human override). As a general mnemonic principle, manual creation and assignment of icons to data should produce better recall (and indeed potentially better icons) than automatic construction and assignment. Manual creation of icons is not viable of course, both because of the time required and because most users are not artists. Manual assignment of pre-existing scenery is possible at present, but few if any users take the time to assign icons to all their data.

What type of scenery? Distinctive visual appearance may be assigned to the data itself or to the space around the data; the scenery may take have 2D or 3D appearance, etc. The approach of assigning appearance directly to the data is arguably more effective than placing scenery around the data – in the latter case, locations should be recognizable but individual files in those locations will still be indistinguishable. But by assigning appearance to the data, we get “places” for free, simply by recognizing the objects in those places.

Scenery should be perceptually diverse. Computer users tend to organize their work into directories or folders containing from dozens up to perhaps several hundred items each, while on the other hand existing data spaces contain many thousands or millions of items. From these numbers we surmise that a desirable scenery creation procedure should be able to synthesize very large numbers of distinct icons of which subsets of several hundreds of icons should be distinguishable at a glance and without study.

Appropriate Complexity. Visual identifiers should be tuned for the human visual system. Some psychological studies have shown an inverted-U curve of image preference as a function of complexity, i.e., a preference for intermediate complexity, with complexity being defined as the number of vertices in random polygons, etc. [Dorfman and McKenna 1966]. The noise image resulting from randomly chosen pixels is maximally complex from entropy and algorithmic complexity points of view, yet such images are not perceptually very distinct – human observers are unable to see the complexity in these images. Thus, it appears that the most distinguishable scenery will be of intermediate rather than the highest possible complexity.

Detail vs. Scale. Scenery should be as distinguishable as possible, not just under close examination but also in an overview in which many data items are visible and each is small. This indicates that distinctive detail should be generated across a range of scales, which in turn suggests a scale-recursive construction.

2.1 How Should an Icon Relate to its Data?

Although most authors (e.g. [Horton 1996]) adopt the view that the data icon should be based on meaningful characteristics of the data such as size, creation date, and content, we take the (perhaps surprising) contrary view that:

- scenery assignment can be fairly *arbitrary*

This is specifically because the role of scenery is not data visualization, rather it is to enable visual search and memory. [Ruddle et al. 1997] gives experimental support for the effectiveness of arbitrarily assigned landmarks in facilitating virtual navigation. The principle is easily seen in the real-world, however: although real scenery has its own internal logic, strong and consistent correlations between the “contents” of places or objects and their appearance are not always present. For example, we may select a good restaurant by reputation without knowing its appearance, but on a second visit we find it again having easily learned its appearance. Similarly, we cannot always guess the appearance of a book due to arrive in the mail from knowledge of its subject matter, but its somewhat arbitrary appearance is almost instantly learned and remembered once we see it. One proviso is that a book usually “looks like a book” despite its variable appearance. The issue of such class appearance will be touched on in section 5. The major point is that we do not need appearance to be consistently correlated with anything in order for us to recognize things – scenery is presented as a *fait accompli* and our visual brain is setup to rapidly learn this somewhat arbitrary mapping between appearance and content.

While we have concluded that the assignment of visual representations to data *can* be arbitrary rather than correlated to data characteristics, should it be? Quite possibly. For one, if the visual representation is based on data characteristics, and these characteristics change as the data is edited, the visuals will presumably change. This is contrary to the goal of providing a memorable identifier. Moreover, giving distinguishable appearance to data is a challenging problem in itself, and simultaneously requiring that the data be visualized only makes it harder.

2.2 Similar Identity versus Similar Content

Textual strings such as filenames generally serve as the ultimate identifier of digital objects. A closer consideration of the nature of filenames will help our understanding of visual identifiers. We believe that

- the filename represents the *meaning* of the file to the user

We further believe that this meaning often cannot be formally determined from the contents of the data. To clarify this statement, consider the three filenames `NIHfall`, `NIHfall_cover`, `NIHfold` containing respectively a report, a corresponding cover letter, and a backup copy of an earlier version of the report. Content analysis would find the two versions of the report to be more similar than the report and the cover letter, but in the user’s mind the report and its cover letter belong together as indicated by similar naming.

Since visual identifiers will play somewhat of an analogous role to filenames, we propose that

- Objects with similar (file)names should have similar visual identifiers

The points made in this section are summarized in the following table contrasting visualization with visual identifiers:

data visualization	visual identifiers
functional icons	memorable icons
distinguishability is a secondary design goal	distinguishability essential
icons reflect data content	reflecting content is secondary
appearance changes according to need	appearance is fairly permanent

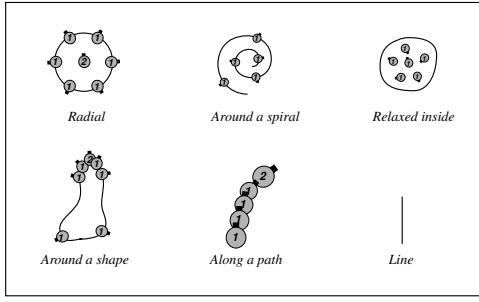


Figure 3: Productions in a simple shape grammar. The numbered circles indicate rewrite markers with the tab indicating orientation. Each figure should be imagined as the right side of a production with a single marker on the left hand side.

3 Synthesis Procedure

Various procedural modeling approaches (fractals, solid textures, L-systems, etc. [Ebert et al. 1998]) are candidates for generating VisualIDs. Among these possibilities, *shape grammars* [Gips 1975] have been applied to a particularly wide range of problems. We have adopted shape grammars as a pattern synthesis procedure suitable for exploring issues in data scenery generation, although other approaches might be equally suitable.

The procedure for synthesizing VisualIDs starts by identifying similar filenames using a clustering algorithm described in section 3.3. VisualID icons for unique filenames (those not part of a cluster) are generated using the shape grammar procedure, described next. The icon is arbitrarily assigned to the file by using a hash of the filename to seed the pseudorandom generator used in the shape grammar. For filenames that are members of a cluster, a hash of the first name added to the cluster provides the pseudorandom seed used to generate a prototype icon. The prototype is then “mutated” to obtain the VisualID for remaining filenames in the cluster. Mutation is described in section 3.4.

3.1 Shape Grammar

A shape grammar consists of a set of shapes (terminals), a set of markers (non-terminals), a set of productions that replace particular configurations of markers (and possibly shapes) with other configurations of shapes (and possibly markers), and a start configuration. Markers are non-drawing shapes that serve to orient and scale productions; a production is applicable when an affine transformation can be found that makes its left side identical to a configuration of markers somewhere in the image. The literature on shape grammars and related techniques is broad and includes many distinct formulations; see [Marriott et al. 1998] for a recent survey.

While most work on visual grammars has focused on modeling the properties of specific objects (a particular tiling, or a particular plant species in the case of L-systems), our goal is to produce a more diverse range of shapes. Classic shape grammars allow a marker to be rewritten with any applicable rule, and it is the selection of the particular rule that produces different shapes from the same grammar. Unfortunately as the number of rules increases, the approach of applying rules independently and at random can lead to incoherent collections of parts rather than diverse but coherent patterns (c.f. Fig. 2 in [Wonka et al. 2003]). To address this issue, we require all instances of a particular marker to be rewritten in the same way at each step. For example, in the *radial* production in Fig. 3 all of the ① markers will be replaced with the same derivation.

This restriction can be formalized by adding identifier generation and matching notations. An identifier-generator *i* returns a unique identifier each time it appears. The matching symbol *_* appearing on the left side of a production matches any identifier. With these

definitions our grammar can be notated as a tuple (N, T, M, P) , where *N* is the set of nonterminals, *T* is the set of terminals, *M* is a set of markers M_i that may be expanded into either non-terminals or terminals, *P* are the production rules, and M_1 is the start symbol. A simple grammar using the *radial* and *along-a-path* productions from Fig. 3 is:

M_1	\rightarrow	<i>radial</i> (M_1, M_1)
M_2	\rightarrow	<i>along-a-path</i> (M_1, M_1)
M_3	\rightarrow	<i>line</i>
M_4	\rightarrow	<i>null</i>

radial, along-a-path $\in M$, *line* $\in T$

A sample derivation from this grammar is:

1. $M_1 \rightarrow \text{radial}(M_2, M_3)$ (pick the *radial* production, marker 1 rewritten as M_2 , marker 2 rewritten as M_3)
2. $M_2 \rightarrow \text{radial}(M_4, M_5)$, $M_4 \rightarrow \text{line}$, $M_5 \rightarrow \text{null}$ (M_2 now expanded as a second *radial* production with its own marker 1 replaced by the *line* terminal and marker 2 replaced by *null*)
3. $M_3 \rightarrow \text{along-a-path}(M_6, M_7)$, $M_6 \rightarrow \text{line}$, $M_7 \rightarrow \text{line}$ (marker 2 of the top level *radial* expanded as *along-a-path* with its markers 1, 2 both replaced by the *line* terminal).

This derivation generates a shape grammar

M_1	\rightarrow	<i>radial</i> (M_2, M_3)
M_2	\rightarrow	<i>radial</i> (<i>line</i> , <i>null</i>)
M_3	\rightarrow	<i>along-a-path</i> (<i>line</i> , <i>line</i>)

i.e. “an *n*-gon with *n*-gons around the perimeter with lines coming off of them, and inside the main *n*-gon a curving path with lines coming off it.”

A derivation is complete when either every M_i has been rewritten to a terminal, or a user-specified maximum recursion level is reached. The complexity of the derivations can also be approximately controlled. Each generator calculates a complexity estimate as the sum of some intuitively assigned constant value for itself, plus the summed complexities of its children. Derivations more complex than a specified threshold are discarded and replaced by the derivation resulting from an incremented pseudorandom seed. During rendering, lines whose cumulative scaling is smaller than a threshold such as 1.5 pixels are discarded.

Arguably a grammatical formalism is not the simplest explanation of this construction. The construction can also be elegantly expressed using the higher-order function construct in functional languages such as *scheme* and *ML* (a higher-order function is a function that returns a function, as distinct from a function that returns a function pointer, as in C++). Fig. 4 lists the *scheme* pseudocode for the *radial* production shown in Fig. 3, omitting the center marker ② for simplicity (the code corresponding to a production will be termed a “generator” below). This code defines the vertex marker production as some randomly chosen derivation through the grammar (outer *lambda*, lines 2-6) and then applies this now determined derivation fragment (now represented by the *child* variable) at the vertex markers (inner *lambda*, lines 7-12).

Note that in calling *generate-child* the construction is not just randomly picking an applicable predefined production, rather, a particular production is in effect being invented by fixing free parameters of what might be considered a “parameterized production”. We call the parameters selected in the outer *lambda* “meta”

level parameters. The result of the meta-level evaluation is a derivation that may still have some random parameters to be evaluated (in our system these include random parameters that give a “sketch” feel to the individual drawing). This division of evaluation into a “meta” stage and a second stage involving a more fully specified grammar can also be seen in the formal grammar description above. Whereas grammars are generally defined in terms of fixed symbols, the notations M_i and $_$ can be viewed as inventing marker symbols and matching them to particular rewrites; the result of evaluating all M_i and $_$ is then a grammar with fixed symbols.

3.2 Implementation details

The simple grammar in Fig. 3 can by itself generate a wide range of patterns; implementation should be evident and is a matter of standard graphics programming. The around-a-shape shapes are made using Fourier filtering of noise, thus ensuring periodicity. The along-a-path generator creates a non-periodic random path, also by noise filtering. The figures in this paper use a grammar that extends the productions in Fig. 3 with additional parameters to add variety, and adds several additional generators. The scribble generator traces the path of a point mass with friction attracted by either random or patterned impulses. The figure generator creates an animal-like arrangement of line generators. symmetry calls its child generator with n -lateral radial symmetry.

While the set of generators and their various parameters were intuitively chosen, we do not believe our particular choice is crucial to the results. To give a better feel for the types of parameters that were used, the full set of parameters for the radial generator are described next (the complete shape grammar is described in the extended online version of this paper): The parameter n is the number of vertices in the n -gon. Boolean variables `hasVertex`, `hasMouth` control whether the markers at the vertices and in the center will be expanded (markers 1, 2 in Fig. 3). `VCscale`, `MCscale` are corresponding scale reduction factors for the respective child generators. `mouthOrg` is an offset of the center child down from the center of the n -gon (directions are relative to the local coordinate system of the generator). `hasEyes` places two additional children that might be interpreted as “eyes”. `eyeSep` is the horizontal separation between these, `eyeOrg` is the offset of the “eyes” up from the center, and `ECscale` is their scale reduction. `angleCut` causes vertex children to appear only within a range of angles from the top of the n -gon, and `angleCutoff` is the corresponding angular threshold in degrees.

The actual random parameter choices use several random generators each defined in terms of the standard uniform [0,1) pseudo-

```

1  (define gen-radial
2    (lambda (recursion-level)
3      (let ((child (generate-child
4                    (+ 1 recursion-level))))
5        (vc-scale (rndin 0.15 0.7))
6        (n (rndin-lowbias 3 12))))
7    (lambda ()
8      (dotimes (i n)
9        (save-ctm
10         (rotate (/ (* i 2PI) n))
11         (scale vc-scale)
12         (child))))))
13
14 (define generate-child
15   (lambda (recursion-level)
16     (let ((child (if (< recursion-level max-level)
17                     (rnd-pick-child recursion-level)
18                     (rnd-pick-terminal))))
19       (child))))

```

Figure 4: Scheme language shape-grammar pseudocode.

random generator `rnd()`. `rndprob(p)` returns true if `rnd` is greater than p . `rndin(a,b)` returns a random value in $[a,b]$; both integer and float versions of this routine are used. `rndinLowbias(a,b)` is similar but is biased toward smaller values by squaring the underlying pseudo-random value (i.e., $a + \text{rnd}()^2 * (b-a)$). The parameter choices for radial are:

<code>n</code>	<code>rndinLowbias(3,12)</code>
<code>hasVertex</code>	<code>rndprob(0.5)</code>
<code>VCscale</code>	<code>rndin(0.15,0.7)</code>
<code>hasMouth</code>	<code>rndprob(0.6)</code>
<code>MCscale</code>	<code>rndin(0.1,0.3)</code>
<code>mouthOrg</code>	<code>if hasEyes rndin(-0.2,-0.3)</code>
<code>hasEyes</code>	<code>rndprob(0.3)</code>
<code>ECscale</code>	<code>rndin(0.1,0.4)</code>
<code>eyeSep</code>	<code>rndin(0.2,0.3)</code>
<code>eyeOrg</code>	<code>rndin(0.2,0.3)</code>
<code>angleCut</code>	<code>rndprob(0.5)</code>
<code>angleCutoff</code>	<code>rndin(90,150)</code>

3.3 Name Clustering

Among the requirements listed in section two, it was suggested that objects with similar names should have similar visual identifiers. A first step toward this goal is to select a pairwise string distance function. Existing procedures such as the Levenshtein edit distance and the (normalized) longest common subsequence are candidates [Graham 1994]. We used a variant of normalized common subsequence that counts all common subsequences (of length ≥ 3), weighted by length. This choice was motivated by the consideration that the files `finalReport.doc` and `finalRevReport.doc` are probably related. The complete filename distance uses an ad-hoc weighing of this measure computed on the “name” part of the filename and a binary comparison of the file extension. Since the extension indicates the file type, however, it might better be used to directly control the icon appearance through grammar parameters or the choice of a grammar rather than participating in the string distance.

Given the string distance function, we adopted a simple one-level incremental name clustering procedure that easily maps onto the shape grammar construction. The following pseudocode describes the clustering algorithm:

```

procedure one-level-cluster(newname)
  find the closest match among existing filenames
  if this distance < threshold then
    add newname to the cluster containing closest match
  else
    create new cluster with newname as its only member
end

```

3.4 Mutation

To create similar VisualIDs for similar (clustered) filenames, the VisualID for the first filename in the cluster is “mutated” to create the remaining VisualIDs. Two examples of the results of mutation are shown in Fig. 5 (Figs. 1, 6 also contain several VisualIDs that are mutated due to filename clustering). In the construction described here “mutation” is relatively easy – the generated grammars can be walked through and one or more of the embedded parameters changed. The recursive construction provides an approximate

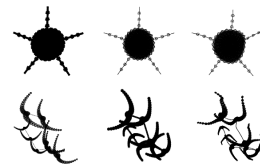


Figure 5: Examples of “mutation”.

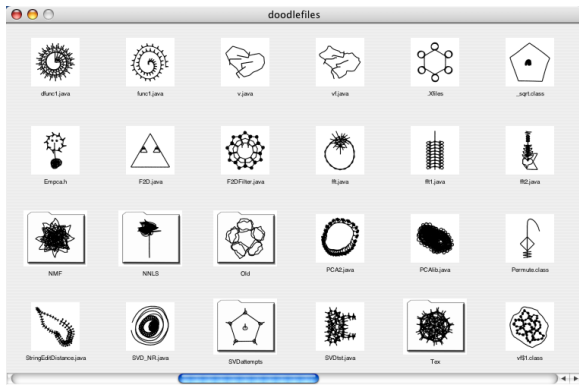


Figure 6: Screenshot of a simple file browser using VisualIDs.

ordering in terms of perceptual saliency, in that parameters at the first recursion level generally have larger effects on the image than those at deeper levels. This perceptual ordering is only approximate however, and a more principled approach such as Design Galleries [Marks et al. 1997] is desirable if suitable output vectors (in the terminology of [Marks et al. 1997]) could be determined. Another issue with this scheme is that the clustering threshold and the amount of variation generated by a “mutation” need to be mutually tuned. At present producing a workable scenery construction requires both artistry on the part of the programmer and tuning. In any case the user should be able to force generation of new icons if the existing ones are not satisfactory, and the user could have access to some of the parameters of the synthesis procedure.

4 Results

The design for the grammar in section 3 was inspired by doodles and reflect one of the authors’ preferences for irregular patterns, with lines sometimes rendered in an intentionally sketchy style. Figure 1 shows moderately complex VisualIDs (maximum recursion level set to 6) generated with this grammar; the diversity of possible shapes is evident. The compute times for these “doodles” range from insignificant to a minute or even more, as some figures contain many thousand short antialiased line segments. As such we envision VisualIDs being generated by a low priority background task, similar to the file indexing tasks in current operating systems.

Once VisualIDs are available they need to be integrated in the user interface. Fig. 6 is a screenshot of a simple file browser that incorporates VisualIDs. The browser is a Java application that, upon entering a directory (folder), computes icons for any files that do not already have them. The icons are computed in a separate thread and are stored as small individual images in a subdirectory. This simple file browser does not resolve issues involving the aesthetics of VisualIDs. In this paper we do not claim to explore aesthetics, rather our focus is simply showing that artificial scenery is feasible and potentially effective. Fig. 7 shows several concepts for icon styles, included here to suggest the range of possibilities.

4.1 Experimental Validation

The file browser prototype also does not in itself demonstrate that VisualIDs are effective, and subjective impressions of the browser are confounded by many factors such as its lack of hardware acceleration and the absence of favorite features of other browsers. On the other hand, there is already the large body of preexisting evidence reviewed in section 1 indicating that appearance is effectively used by humans in many contexts. To verify that these results transfer to artificially created icons as expected, we did several user studies. In these studies we focus on distinct appearance rather than on spatial memory and navigation – the latter are partially served by

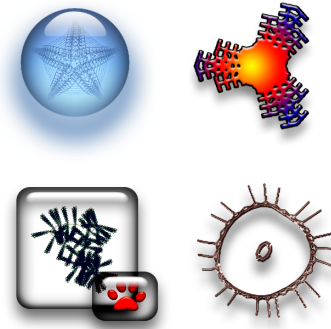


Figure 7: Examples rendering styles for VisualIDs. The design in the lower left illustrates including both a VisualID and a branding logo (a footprint).

existing desktop interfaces, but the experimental evidence shows that navigation without distinctive appearance is difficult [Searleman and Herrmann 1994; Ruddle et al. 1997].

Study 1. In the first study, users were asked to find files, specified by filename, in a “file browser” displaying files in a simulated folder, using either distinctive (VisualID) icons or generic (plain document) icons. A grid of 6x4 files was chosen as representative of a typical number of files in a single folder. In a within-subjects (paired samples) design, each user searched for 2 files, 3 times each, for each type of icon. The order of file targets for each subject was randomly chosen. This repeated search task tests a combination of visual search and short term icon memory: users quickly learn the association between filenames and VisualIDs and can thus use the VisualID to aid in search. The study was implemented as a Java applet that uploaded timing data to a server. The 25 users who completed the study required an average of 23.2 seconds to complete the four VisualID searches, versus 30.5 seconds for the generic-icon searches (paired t-test $p=0.029$), an improvement of approximately 30%. We would expect an even greater advantage if users were allowed to learn the icons over more than three trials.

Study 2. The first study did not test icon memory over more than a few minutes, and the file names were specified. Arguably in real life a user tries to find a file based on a memory of the contents of the file, and on vague memory of the file name and location; people are typically not given the file name and told to search for it. In the second study, we wanted the users to learn a more natural association between the icon and the “contents” of the file, and then search for a file specified by those contents. Users were first trained using a modified version of the game “Concentration.” Although a game might not seem the most obvious choice for our study, it has certain characteristics that make it a good substitute for the natural task of working with files and directories: there is no explicit need to memorize the association between the file “contents” and the name or icon, since the concentration task can be done entirely by memorizing the locations of the various cards. This is very similar to what would be true for VisualIDs – users would not explicitly memorize the associations. We want to know how users perform under these circumstances.

Users viewed a 5x4 array of cards, with either a VisualID or generic icon on each card, and a file name beneath (Fig. 8). Each user saw only one type of icon (VisualIDs or generic). When the user clicked on a pair of cards, this revealed the name of a country behind each card. The country thus plays the role of the “content” of the file. The user’s goal was to find all 10 matching pairs of countries. For additional training, the user was then asked 25 additional questions such as “Which country is north of the USA? (Click on the pair.)” The next day, users performed a second set of tasks. First, they were shown a series of icons, and asked to pick the associated country in a multiple choice question. The 21 users who completed both

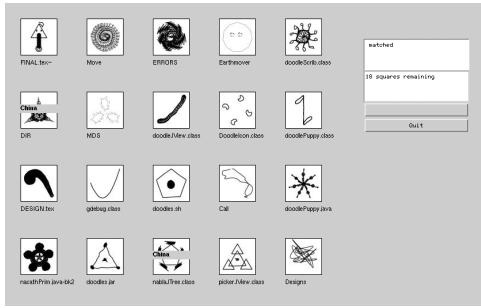


Figure 8: Screenshot of Study 2's concentration game (with-icon condition).

days were 37% correct with VisualIDs, and only 24% correct with generic icons, a significant difference ($p=.017$, heteroscedastic). In the second task, users searched for a card, as in the file search task of Study 1, except that the card was specified by its “content” (the associated country) rather than by name. In this task users with VisualIDs averaged 8 correct selections out of 20, versus 2.08 correct for the generic condition, $p=.007$.

We also asked several subjects to sketch and describe icons from memory two days later. They produced reasonable sketches (Fig. 9), and descriptions such as “a triangle with two eyes on a stick is Italy”. Approximately six weeks following this study we showed an accessible group of four of the former subjects a printed page containing a 4x4 grid of icons, nine of which they had seen six weeks previously. All were more than 80% correct in identifying the previously seen icons.

The results of the studies are summarized in the following table:

	generic icons	VisualIDs
study1: mean search time	30.5 secs.	23.2 secs.
study2: label-to-content (percent correct)	24%	37%
study2: content-to-label (correct/total)	2.08/20	8/20

Additional details on the studies and description of two earlier studies are found in the extended online version of this paper. Although many further studies are possible, the advantages of visual information display have already been well demonstrated in other contexts. The results here show that these advantages can be expected to transfer to distinguishable artificially constructed icons as well.

5 Discussion and Future Work

Coexistence, not Replacement. A major point is that

- VisualIDs do *not* replace existing mechanisms for manipulating files.

Although the potential utility of visual identifiers is clear there may be reasons why distinctive appearance is not for everyone. Just as some people have particularly orderly desks, some users may prefer the simplicity and uniformity of a scenery-less interface. A *view-by-appearance* mode analogous to the existing *view-by-date*, *view-by-type*, etc. modes could show scenery only when requested.

Scaling issues. Several scaling issues arise in visual search. First is the issue of whether pop-out search [Treisman and Gelade 1980] is possible. In pop-out search, a unique pattern sought among a relatively homogeneous array of distractors can be located in essentially constant time. While such efficient search is clearly desirable, it is inherently impossible with any sort of static pairing of icons to files, for a system of any reasonable scale. For efficient search the file identifiers must be sufficiently distinct along a simple feature dimension (e.g. color, orientation, size) that they are clearly distinguishable, yet homogeneous enough that pop-out search is possible.



Figure 9: User sketches of three icons from memory several days after study 2 (left), and the corresponding original icons (right).

For a small number of files (say, 4), this could be achieved by assigning saturated colors to the files, but for a more realistic number of files the criteria of distinguishability and homogeneity cannot be simultaneously satisfied, and a slower conjunctive or configuration search will be required. Whether searching for a word among words, or a pictorial file icon among other pictures, constant time pop-out search will not be possible with any reasonable number of files. The best one can do is to adopt a technique that provides a better search slope and offset than the alternatives. Evidence indicates that search for a picture among pictures is better than search for a word among words in this regard (e.g. [Paivio 1974]).

A second issue is whether we can really distinguish and remember icons for every possible file. A mid-90s study of 15 diverse computer users found that they had from 2,400 to 31,000 files [Barreau and Nardi 1995], and the economy of computer storage has only increased these numbers. People probably cannot remember or even easily distinguish VisualIDs for every file. In other words, VisualIDs do not “scale” in absolute terms (note that this is equally true of memory for filenames). Nonetheless it is easy to argue that such scaling is not necessary for VisualIDs to be useful. People often work on projects that involve only a subset of all their files, and they only need to remember the identifiers (whether visual or textual) for the files in the current project(s). VisualIDs are effective as long as icons for a significant fraction of the number of files accessed on a project can be easily learned and remembered.

Branding. Software vendors may object to having their branding logo (e.g. the “W” icon identifying a Word file) replaced with something more unique. It is possible to imagine schemes that address this objection, either by allowing simultaneous display of both the VisualID and an identifying logo as in Fig. 7, or through different viewing modes. Specific software packages could adopt characteristic classes of icons to represent their files, for example, icons for files created by a CAD package could be small drawings of imaginary machines created using a suitable shape grammar. The issue of branding versus unique identity is already present with image data, and has been addressed in one program by displaying the image thumbnail inside a “frame” that identifies the software package.

Future Direction: Visual Search Interfaces. The idea of *query-by-sketch* has been explored for image data [Jacobs et al. 1995], but visual identifiers might allow query-by-sketch to operate on other types of data. VisualIDs also addresses a fundamental problem with most forms of query-by-sketch: that many people are not very good at sketching. Since VisualIDs are algorithmically generated, users can potentially generate queries by guiding the scenery construction procedure e.g. with successive selection [Sims 1991]. Although the user would make some errors, a search that returned the N closest matches could be used in conjunction with other constraints (file type, date, ...) to produce a list of candidates. VisualIDs might also allow RSVP-style search for data, in which icons are rapidly displayed in succession and the user can stop and backup to examine items that resemble the one sought [Intraub 1981].

6 Conclusion

In two decades of discussion of graphical user interfaces the role of scenery has been remarkably overlooked. This paper has surveyed the psychological literature and presented additional empirical evidence that data scenery in the form of distinctive visual identifiers can allow us to more fully engage our visual brains at the computer interface. We have proposed that graphics techniques be used to au-

tomatically generate such visual identifiers, and have demonstrated one technique, a ‘meta’ shape grammar, that can generate a diverse range of patterns.

The research direction presented here addresses only one component of a complete user interface. Further exploration of pattern generation is needed and new design issues are introduced by this proposal. On the other hand, “VisualIDs” do not require a radical change to current desktop interfaces, and they complement existing research on spatial navigation schemes. Our experimental results are both strong and easily obtainable. While Nielsen notes that even minor interface inefficiencies become a major cost when multiplied across thousands of users [Nielsen 1993], the potential efficiency demonstrated here (e.g. a 30% increase in search speed after only a few minutes of use) is not minor.

Acknowledgements

We thank Suan Tan, Ben Cheung, Pat Hanrahan, Dan Piponi, Rob Burton, Lance Williams, Sean Jenkins, Lewis Siegel, Tae-Yong Kim, Zhenyao Mo, and the anonymous reviewers for their discussion and suggestions, and artists at ESC and ILM for participating in the studies. See the extended online version of this paper for additional figures and acknowledgements.

References

- BARREAU, D., AND NARDI, B. 1995. Finding and reminding: File organization from the desktop. *SIGCHI Bulletin* 27, 3, 39–43.
- BYRNE, M. 1993. Using icons to find documents: simplicity is critical. In *Proceedings of INTERCHI '93*. ACM, 446–453.
- CARD, S., ROBERTSON, G., AND MACKINLAY, J. 1991. The information visualizer, an information workspace. In *Proceedings CHI*, ACM, 181–288.
- CZERWINSKI, M. P., VAN DANTZICH, M., ROBERTSON, G., AND HOFFMAN, H. 1999. The contribution of thumbnail image, mouse-over text and spatial location memory to web page retrieval in 3d. In *Proc. INTERACT*, 163–170.
- DARKEN, R., AND SIBERT, J. 1996. Navigating large virtual spaces. *Int. J. Human-Computer Interaction* 8, 1, 49–71.
- DONELSON, W. 1978. Spatial management of information. In *Computer Graphics (Proc. ACM SIGGRAPH 78)*, 203–209.
- DORFMAN, D., AND MCKENNA, H. 1966. Pattern preference as a function of pattern uncertainty. *Canadian J. Psych.* 20, 143–153.
- EBERT, D., MUSGRAVE, F. K., PEACHEY, D., PERLIN, K., AND WORLEY, S. 1998. *Texturing & Modeling: A Procedural Approach*. Academic Press, Inc.
- GIPS, J. 1975. *Shape Grammars and their Uses*. Birkhauser.
- GRAHAM, S. 1994. *String Searching Algorithms*. World Scientific, Singapore.
- HIGHTOWER, R., RING, L., HELFMAN, J., BEDERSON, B., AND HOLLAN, J. 1998. Graphical multiscale web histories: A study of PadPrints. In *Proceedings Ninth ACM Conference on Hypertext and Hypermedia*, ACM, ACM, 58–65.
- HILL, W. C., HOLLAN, J. D., WROBLEWSKI, D., AND MCCANDLESS, T. 1992. Edit wear and read wear. In *Proc. CHI*, ACM, 3–9.
- HORTON, W. 1996. Designing icons and visual symbols. In *CHI Course Notes*. ACM.
- INGRAM, R., AND BENFORD, S. 1995. Legibility enhancement for information visualisation. In *Proc. IEEE Visualization*, IEEE, 209–216.
- INTRAUB, H. 1981. Rapid conceptual identification of sequentially presented pictures. *J. Experimental Psychology: Human Perception and Performance* 7, 3, 604–610.
- JACOBS, C. E., FINKELSTEIN, A., AND SALESIN, D. H. 1995. Fast multiresolution image querying. *Computer Graphics* 29, Annual Conference Series, 277–286.
- JONES, W., AND DUMAIS, S. 1986. The spatial metaphor for user interfaces: Experimental tests of reference by location versus name. *ACM Trans. Office Information Systems* 4, 1, 42–63.
- LANSDALE, M. 1988. The psychology of personal information management. *Applied Ergonomics* 19, 1, 55–66.
- LEUNG, Y. K., AND APPERLEY, M. 1994. A review and taxonomy of distortion-oriented presentation techniques. *ACM Trans. Computer-Human Interaction* 1, 2, 126–160.
- MACKINLAY, J., CARD, S., AND ROBERTSON, G. 1991. Perspective wall: Detail and context smoothly integrated. In *Proceedings of SIGCHI*, ACM, 173–179.
- MAGLIO, P., AND BARRETT, R. 1997. On the trail of information searchers. In *Proceedings of the Nineteenth Annual Conference of the Cognitive Science Society*. Mahwah, NJ: LEA.
- MALONE, T. 1983. How do people organise their desks? Implications for the design of office information systems. *ACM Trans. Office Info. Systems* 1, 1, 99–112.
- MARKS, J., ANDALMAN, B., BEARDSLEY, P. A., FREEMAN, W., GIBSON, S., HODGINS, J., KANG, T., MIRTICH, B., PFISTER, H., RUMMLER, W., RYALL, K., SEIMS, J., AND SHIEBER, S. 1997. Design galleries: a general approach to setting parameters for computer graphics and animation. *Proc. ACM SIGGRAPH 1997* 31, Annual Conference Series, 389–400.
- MARRIOTT, K., MEYER, B., AND WITTENBURG, K. 1998. A survey of visual language specification and recognition. In *Visual Language Theory*, B. M. K. Marriott, Ed. Springer, Berlin.
- MUKHERJEA, S. 1999. Information visualization for hypermedia systems. *ACM Computing Surveys* 31, 4es, 1–6.
- NIELSEN, J., AND LYNGBAEK, U. 1989. Two field studies of hypermedia usability. In *Hypertext: State of the Art*, Ablex, R. McAleese and C. Green, Eds., 64–72.
- NIELSEN, J. 1993. *Usability Engineering*. Academic, London.
- PAIVIO, A. 1974. Pictures and words in visual search. *Memory & Cognition* 2, 3, 515–521.
- PERLIN, K., AND FOX, D. 1993. Pad - an alternative approach to the computer interface. In *Proc. ACM SIGGRAPH 1993*, ACM, 57–64.
- ROBERTSON, G., CZERWINSKI, M., LARSON, K., ROBBINS, D., THIEL, D., AND VAN DANTZICH, M. 1998. Data Mountain: using spatial memory for document management. In *Proc. User Interface Software and Technology (UIST)*, ACM, 153–162.
- RUDDLE, R., PAYNE, S., AND JONES, D. 1997. Navigating buildings in desktop virtual environments: Experimental investigations using extended navigational experience. *J. Experimental Psychology - Applied* 3, 2, 143–159.
- SEARLEMAN, A., AND HERRMANN, D. 1994. *Memory from a Broader Perspective*. McGraw Hill, New York.
- SHEPARD, R. 1967. Recognition memory for words, sentences, and pictures. *J. Verbal Learning Verbal Behavior* 6, 156–163.
- SIMS, K. 1991. Artificial evolution for computer graphics. In *Computer Graphics (Proc. ACM SIGGRAPH 91)*, ACM, 319–328.
- STANDING, L., CONEZIO, J., AND HABER, R. 1970. Perception and memory for pictures: Single-trial learning of 2560 visual stimuli. *Psychonomic Sci.* 19, 73–74.
- STEFANUCCI, J. K., AND PROFFITT, D. R. 2002. Providing distinctive cues to augment human memory. Poster, 24th Annual meeting of the Cognitive Science Society, Fairfax, VA.
- TAN, D., STEFANUCCI, J., PROFFITT, D., AND PAUSCH, R. 2001. The Infocockpit: Providing location and place to aid human memory.
- TREISMAN, A. M., AND GELADE, G. 1980. A feature-integration theory of attention. *Cognitive Psych.* 12, 97–136.
- WONKA, P., WIMMER, M., SILLION, F., AND RIBARSKY, W. 2003. Instant architecture. *ACM Trans. Graph.* 22, 3, 669–677.
- WORDNET. Wordnet 1.7. www.cogsci.princeton.edu/~wn.

Shape Grammar

The complete parameters for the shape grammar are described here. In the parameter descriptions:

- `level` is the recursion level, starting at 1 and increasing with smaller scale detail.
- `Rnd11()` returns a uniform pseudo-random value in $[-1, 1]$.
- `rndinLowbias(a, b, p)` is a version of the low-bias generator where the power can be specified: $a + (b-a) * \text{pow}(\text{Rnd}(), p)$.
- `rational(lo, hi, den)` returns a rational number (cast to real) where the numerator is in `lo, hi` and the denominator is `den`.
- `rndprobLevel(p)`: The probability is reduced at each level using $p = p - p * \min(\text{level}, 5) / 6$
- `rndinLevel(a, b)`: the range is reduced by recursion level, i.e., $a + \text{Rnd}() * (b-a) / \text{level}$.

Around-a-spiral

<code>len</code>	<code>rndin(25, 60)</code>	number of segments in the spiral (if level=1)
<code>len</code>	<code>rndin(10, 30)</code>	number of segments in the spiral (if level=2)
<code>len</code>	<code>rndin(5, 10)</code>	number of segments in the spiral (if level=3)
<code>len</code>	5	number of segments in the spiral (if level > 3)
<code>aligned</code>	<code>rndprob(0.5)</code>	children are aligned on radial spokes
<code>ppr</code>	<code>rndinLowbias(3, 35)</code>	points per revolution (if aligned)
<code>ppr</code>	<code>rational(3, 35, rndin(3, 7))</code>	points per revolution (if not aligned)
<code>hasChild</code>	<code>rndin(0.5, 1)</code>	place a child generator at points along the spiral
<code>centerScale</code>	<code>rndin(0.05, 0.6)</code>	scale reduction for the child at center of spiral
<code>outerScale</code>	<code>rndin(0.25, 1)</code>	scale reduction for the child at outside of spiral

The spiral algorithm is

```
for( i=0; i < len; i++ ) {
    theta = 2*pi*float(i) / ppr;
    r = 0.15 * sqrt(theta);
    x = r * cos(theta);
    y = r * sin(theta);
}
```

The `ppr` is adjusted so that there are at least two revolutions in the spiral.

Around-a-Shape and Relaxed Inside

Both of these are implemented with the same code, with a boolean variable `hasOC` controlling whether a child is generated around the outline.

The outline is created with either a Fourier series or FFT filtering of noise, according to the variable `fourier`. Fourier series generation produces a more regular or coherent outline. Pseudocode for the Fourier series is

```
ncoef = 2*rndin(2, 5) + 1;
for( i = 0; i < ncoef; i++ )
    amp[i] = Rnd11() / float(i+1); // 1/f envelope
```

<code>n</code>	<code>rndinLevel(3, 6)</code>	number of objects (power)
<code>fourier</code>	<code>rndprob(0.5)</code>	use fourier series (else FFT)
<code>exponent</code>	<code>rndin(1.6, 2.2)</code>	spectral exponent for FFT filtering
<code>hasIC</code>	<code>rndprobLevel(0.6)</code>	a child is placed inside the outline
<code>hasOC</code>	<code>rndprobLevel(0.6)</code>	a child is placed around the outline
<code>OCscale</code>	<code>rndin(0.1, 0.6)</code>	scale reduction for the outline child

The number of interior children is 2^n , and each child is scaled down by $\text{rndin}(0.1, 2) / 2^n$.

Along a Path

The outline is created with either a Fourier series or FFT filtering of noise, according to the variable `fourier`. The code is similar to that for the `relaxed inside` generator, except that a subrange of the synthesized outlines between parameter 0,0.7 is used to produce an open, aperiodic curve, whereas `relaxed inside` uses the full $0,2\pi$ range of the outline.

The number of children placed along the curve is 2^n .

<code>hasChild</code>	<code>rndproblevel(0.8)</code>	place child along the path
<code>Cscale</code>	<code>rndin(0.1,0.6)</code>	child scale reduction
<code>n</code>	<code>rndinLevel(3,6)</code>	(power) number of children placed along the path
<code>fourier</code>	<code>rndprob(0.5)</code>	use Fourier series synthesis
<code>exponent</code>	<code>rndin(1.6,2.2)</code>	spectral exponent for FFT filtering

Scribble

The `scribble` generator traces a path attracted by random or patterned attractors and damped by “friction”. In the random case attractors are randomly distributed throughout the unit square centered on the origin. Patterned attractors `X[i]`, `Y[i]` are distributed in a “zig-zag” pattern according to the code:

```
float dx = 0.f;
float dy = 1.f;
float cx = 0.f;
float cy = -0.5f;
float opposingRatio = rndin(0.05f, 0.2f);
for( int i=0; i < ncvs; i++ ) {
    X[i] = cx;
    Y[i] = cy;
    if (i%2 == 0) {
        cx = cx + dx;
        cy = cy + dy;
    }
    else {
        cx = cx - dx;
        cy = cy - dy;
    }
    // move perpendicular
    cx = cx - opposingRatio * -dy;
    cy = cy - opposingRatio * dx;
} //for
```

The attractors are applied in sequence, with a new attractor taking effect when the line has approached a preset distance (0.03) from the current attractor. The attractor force is in the direction from the current point to the attractor. The scribble algorithm is best described in code:

```
float jitter = 0.1f;
float dx = X[1] - X[0]; // X,Y are the attractor locations
float dy = Y[1] - Y[0];
dx = dx + jitter * rndf11();
dy = dy + jitter * rndf11();
float len = (float)sqrt(dx*dx + dy*dy);
float distthresh = 0.03f;
float force = 0.01f;
float cx = X[0];
float cy = Y[0];
float vx = 0.f;
float vy = 0.f;
int ncvs = X.length;

moveto(cx, cy); // use moveto/lineto drawing

for( int icv=0; icv < ncvs; icv++ ) {
    int icv1 = (icv + 1) % ncvs;
    float spandx = X[icv1] - X[icv];
    float spandy = Y[icv1] - Y[icv];
    float dot = 1.f;
    boolean flag = true;
```

```

while(flag) {
  // integration
  force = (len > 0.2f) ? (force*1.1f) : (force*0.7f);
  if (force > 0.08f) force = 0.08f;
  if (force < 0.01f) force = 0.01f;
  vx = vx + force * dx;
  vy = vy + force * dy;
  vx = vx * _friction;
  vy = vy * _friction;
  cx = cx + vx;
  cy = cy + vy;
  lineto(cx, cy);

  // new direction
  dx = X[icv1] - cx;
  dy = Y[icv1] - cy;
  len = (float)sqrt(dx*dx + dy*dy);
  dx = dx + (float)abs(dx)*jitter*rndf11();
  dy = dy + (float)abs(dy)*jitter*rndf11();

  // end segment?
  dot = spandx*dx + spandy*dy;
  if (len < distthresh)
    // near attractor, break
    flag = false;
} //while(flag)
} //icv

```

zigzag	rndprob(0.5)	zig-zag patterned impulses (else random)
n	rndin(2,30)	if !zigzag and level=1
n	rndin(10,20)	if !zigzag and level>1
n	rndin(40,10)	if zigzag
friction	rndin(0.85,0.98)	if !zigzag
friction	rndin(0.8,0.9)	if zigzag

Symmetry

The symmetry generator draws nothing itself but applies its child in an n-lateral radial symmetry. If n=2 the child is mirrored.

n	rndinLevelLowbias(2,8)	n-lateral symmetry
offset	rndprob(0.3)	translate out from center
offsetTrans	rndin(0.1,0.5)	amount to translate out from center
Cscale	rndin(0.3,0.6)*(1-offsetTrans)	child scale reduction

Line generator

Although line serves as a terminal in the grammar we also use it as a non-terminal with these parameters:

len	rndin(0.5,1)	length of the line
n	rndinLowbias(3,10,1.5)	number of C3 children along the line
hasEC1	rndprob(0.75)	has child C1 at end 1
hasEC2	rndprob(0.75)	has child C2 at end 2
hasC3	rndprob(0.5)	has child 3 (along the line)
C1scale	rndin(0.15,0.6)	scale reduction for child 1
C2scale	rndin(0.15,0.6)	scale reduction for child 2
C3scale	rndinLowbias(0.05,0.7,1.5)	scale reduction for child 3
C3doublesided	rndprob(0.8)	child 3 is drawn mirrored on both sides
C3angle	rndin(-pi/4,pi/4)	angle of child3 relative to line normal
C3taper	rndprob(0.3)	reduce or increase scale of child3 by 1.5 along the line
C3increase	rndprob(0.3)	increase (rather than decrease) child3 scale along line

Figure generator

The figure generator produces an animal-like arrangement of generators. Most generators are forced to be `line`, while the “head” is allowed to be a `line` or `radial`.

headScale	<code>rndin(0.1,0.2)</code>	scale reduction for the head child
legScale	<code>rndin(0.3,0.6)</code>	scale reduction for the leg children
tailScale	<code>rndin(0.1,0.4)</code>	scale reduction for the “tail” child

FAQ

This paper has been read by a number of people and we have received lengthy comments about it, many of which refined our presentation. This section covers some of the objections that were raised.

Icons do not generally qualify as scenery. Our definition is not too different than the one in WordNet (which is evidently derived from actual usage): “the appearance of a place.” While some places (such as a beach) are difficult to divide into meaningful “objects”, others (such as a city, or most other man-made environments) are clearly composed of objects in the everyday sense. The scenery in a city is *the appearance of its non-moving objects* (buildings, etc.). If all the buildings in your town had a uniform size, shape, and color, you would have a lot more difficulty finding your way!

This “object-based” definition of scenery is in fact *necessary* for files, since files are discrete objects. The form of scenery you are probably thinking of (more like the beach example) would take the form of distinguishable appearance for the (folder) backdrop. This is also desirable and we advocate it in the paper as well, but it is not as essential. For example, if every folder had a distinctive picture as its backdrop when opened, but the file- (object-) based scenery was absent, navigating to other folders and finding files would still be difficult, since all sub-folders and files would have a generic appearance.

The experiment’s relationship to the larger issues of 3D navigation are not immediately apparent. We do not claim that this paper directly addresses navigation. But, navigation in featureless spaces is difficult and adding distinctive appearance is known to help navigation ([Searleman and Herrmann 1994; Ruddle et al. 1997] and to a lesser extent [Ingram and Benford 1995; Darken and Sibert 1996; Robertson et al. 1998]). Scenery as we define it *complements* spatial navigation schemes and is *required* to engage our visual brain in navigation.

People use natural languages to express information much more frequently than we use pictures. If people could not even remember the labels given in natural language, it’s hard to believe that they can remember graphical labels better than the textual labels. Both psychological and HCI evidence (Section 1,2) and our own studies 1,2 show the opposite.

There are not enough pixels on current screens to show all these icons. The icons used in the studies are 64² (at least one current OS has icons of this size) and are quite distinguishable. Further, the idea of zoomable interfaces (e.g. [Perlin and Fox 1993]) is well established and could be used if necessary to distinguish very similar icons such as those resulting from mutation.

You argue that the icon does not need to be related to the content. People normally remember the content but not the exact labels of the documents. I just don’t know how the content-irrelevant icons can help people recall. It is true that people remember the content of documents. Documents also have labels, and it’s true that people often do not remember the exact labels. The comparison here is between textual labels and textual+visual labels. All the evidence (prior and ours) supports the idea that humans have strong abilities to find and recall by appearance, and when given the choice, we try the visual approach first [Byrne 1993]. For example, I locate one of my books (having some desired content) by appearance, not by reading the title of each book in my collection.

How does the system treat the case in which a set of files is named similarly (paper1.tex, paper2.tex, ...) but one differently (final.tex) which, however, is similar in content but unfortunately not in the naming. The paper1.tex, paper2.tex, ..., final.tex issue was addressed in the **Similar Identity versus Similar Content** section on page 43. This is a good example of what was described there: although the content of all these files are similar (paper1, paper2, ... being successive versions of the paper), their *meaning* for the user is different. And this difference in meaning can be seen from the file name but not the file content! The user names the final version to final.tex to make it stand out from the others; it is the valuable final version, whereas the others are backup versions. In the mutation proposal its file icon will also stand out from the others, as desired.

What if the users change filenames often or even reorganize their file system. Then an icon that has been the visual hook for a file will be newly created and differ from the previous one. Would not the user be even more lost then? No. Users can certainly loose files by moving and renaming them, regardless of the current proposal. See the **Coexistence, not Replacement** section: the original file names are still there; the scenery is just one additional cue. As all the previous information is still there, no one should be “more lost”. The act of renaming a file has a non-trivial “cost” – the user must remember the new name. The evidence indicates that this remembering task is faster and more robust with appearance data than it is with only textual labels.

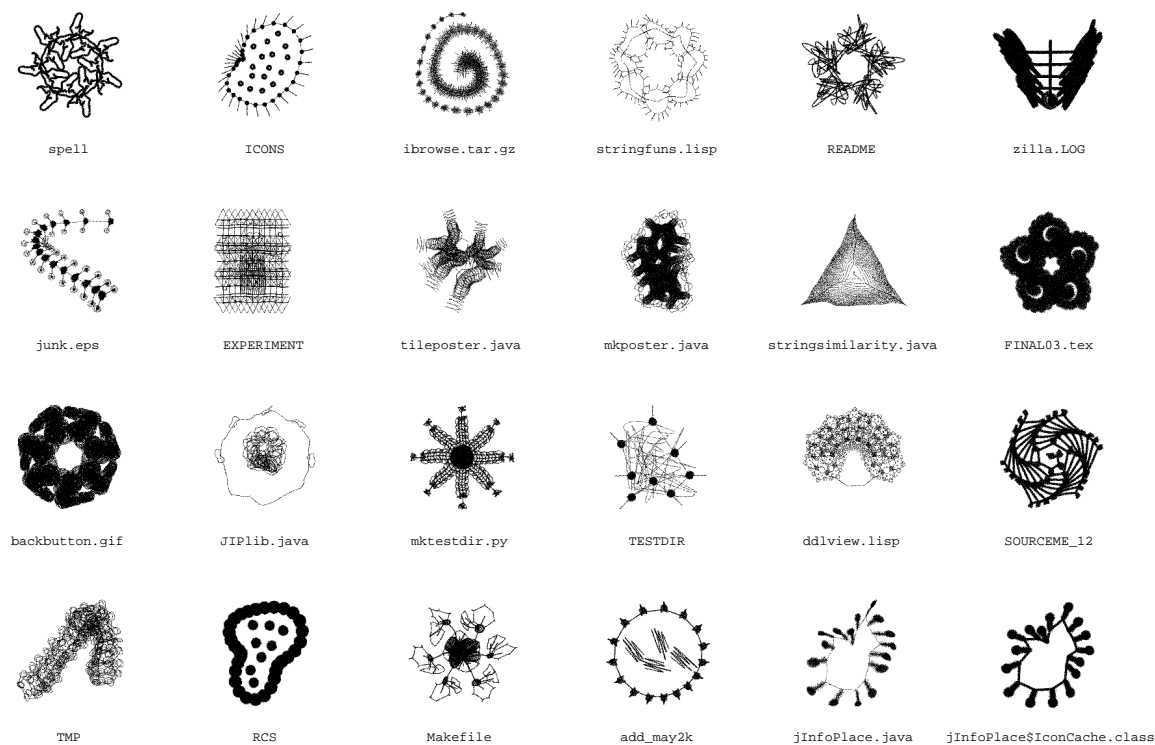
The other question suggested here is whether it is best to change the icon for a file when the name changes. By the argument above, neither possibility should really hurt performance, but perhaps one choice will improve performance more than the other. Although this question probably needs further study, we suspect that changing the icon along with the name is the most consistent choice. In the paper1.tex, paper2.tex, ..., final.tex example, we suppose that the user names or renames the final version final.tex to make it stand out from the others. If they are renaming the final version, and the icon does not also change, the file’s appearance will be misleading – it will look like one of the earlier versions.

If I rename a number of files and their icons all change, how can I ensure that the new icons bear the same relationship to each other that the old ones did? Probably this is not possible, but why is it necessary? There is nothing indicate that we would be particularly bothered by this scenario, on the contrary, it is a visual learning problem of the sort that we are quite skilled at. In the real world, my favorite bookstore and the adjacent coffee shop may well be replaced by a Starbucks and a 7-11, which do not bear the same relationship as the original two stores... but I’ll have a lot more difficulty remembering a mathematical fact, or even someone’s name, than the appearance of the new stores.

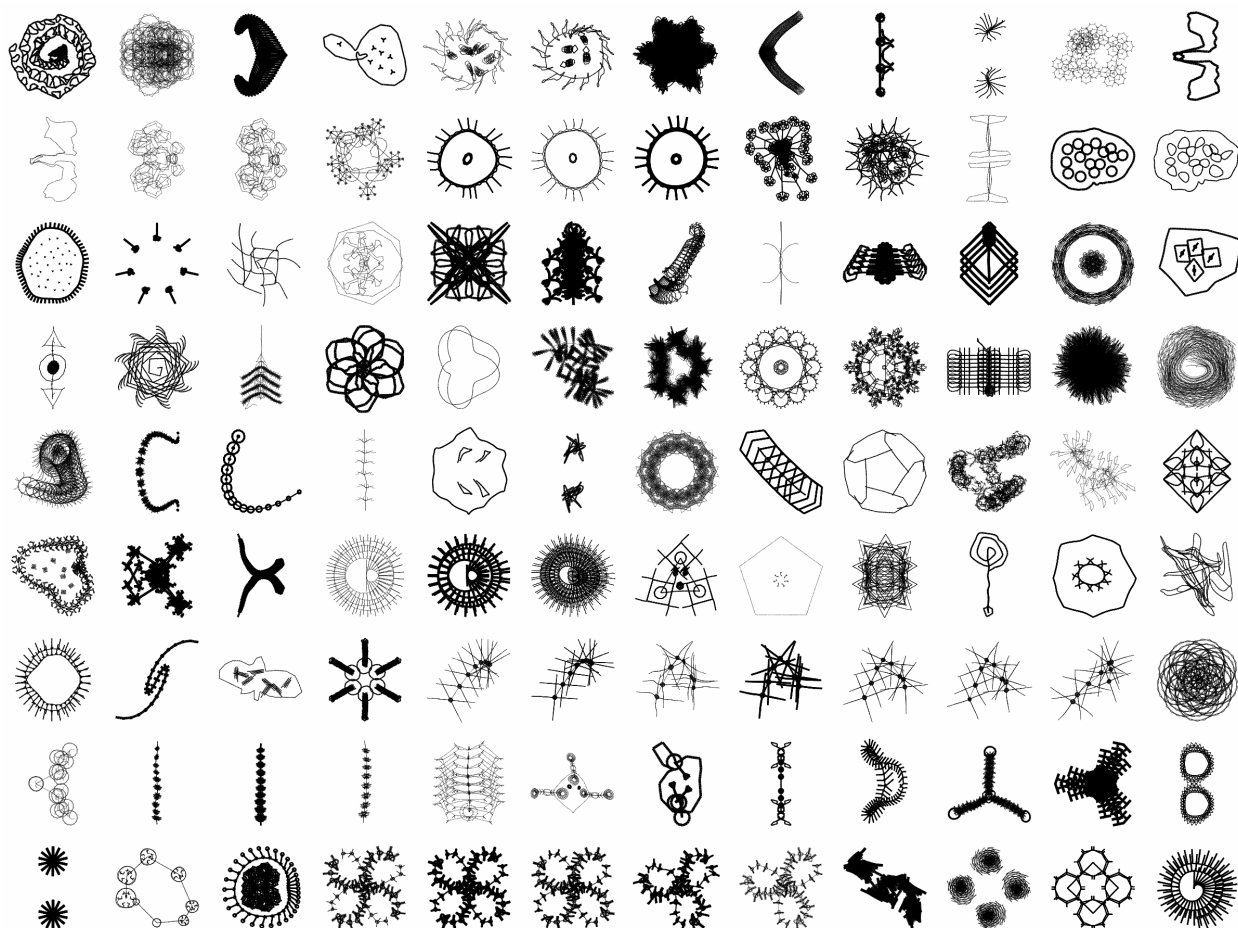
Photographs versus abstract icons. Small photographs have many of the same benefits as abstract scenery, and are likely to be more distinguishable than abstract icons. On the other hand there are reasons to prefer abstract icons: 1) the use of similar icons for similar filenames might be difficult or impossible with photographs; 2) algorithmic construction offers additional possibilities for branding and query; 3) abstract icons are relatively (though not entirely) free of the semantic and culturally specific issues of interpretation that can interfere with

icon success (A. MacEachren, *How Maps Work: Representation, Visualization, and Design*, Guilford Press, NY, 1995).

You did not consider [issue X]. Yes, we did not consider all issues that arise from this “scenery” idea. A single paper on most any topic cannot cover every related issue, and admittedly this paper has more unresolved (and interesting) issues than most.



More doodle-style file icons. The maximum recursion level is increased to generate more detailed icons. Clustered filenames appear together, but overall order is arbitrary due to traversing a hash table.



More doodles.

Additional details on the user studies:

To eliminate the time required for generating icons dynamically a set of pre-generated icons was used in the experiments. Filenames were taken from the project directory for this paper and icons were assigned to them arbitrarily (files ordered by the java File.list() call were matched with icons in order of generation).

There was a precursor to study 1.

In this version subjects (professional computer artists) were instructed to find particular files in a mock interface displaying 200 files in a scrollable window. There were two conditions, with- and without-doodle icons. Eight files were requested twice each, thus testing for spatial and icon memory. While the total search time for the 'scenic' condition was slightly shorter than under the generic icon condition (and was mildly significant) we decided to revise this study based on user feedback. The comments were that:

- 1) a single repetition was not enough
- 2) much of the elapsed time was spent on the mechanics of scrolling
- 3) in both conditions subjects reported that they made use of the spatial layout in finding files- for example they might recall that a particular file was somewhere to the upper left in the scrollable display.

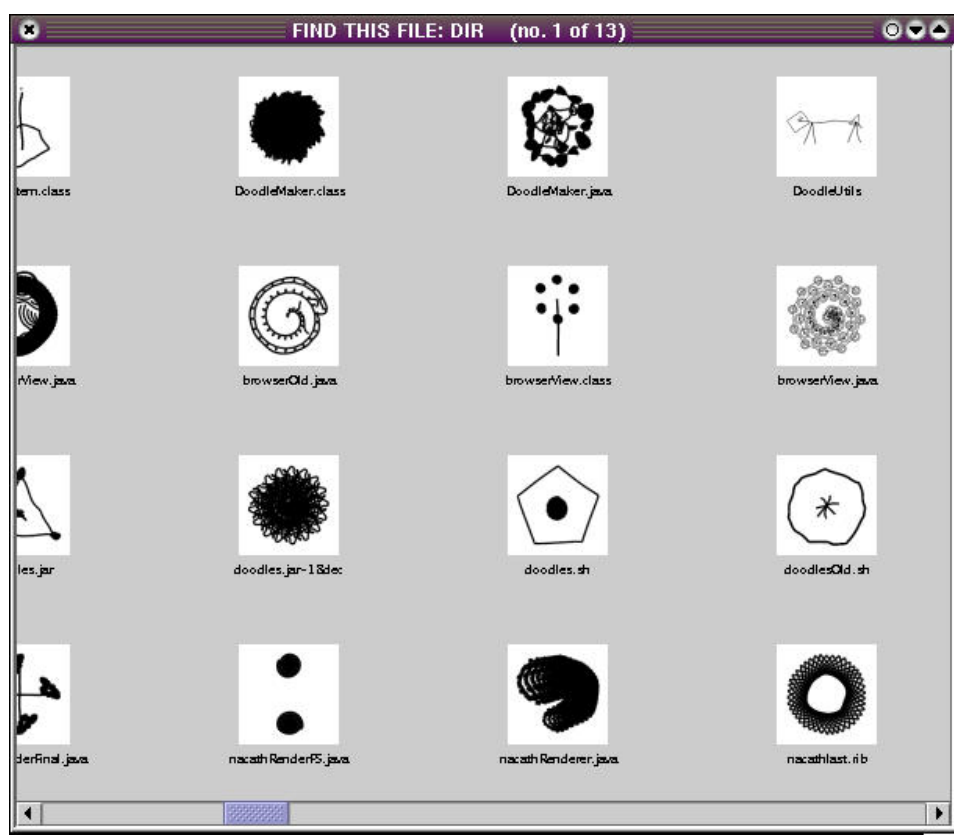
These comments suggest that the study design did not clearly show the potential effect of distinguishable icons. Simply running this study for a longer period (more than one repetition) would be a suitable fix, but we wanted to keep the total time to a 'polite' 5-10 minutes.

In study 1 the number of files was reduced to 24 -- this is probably a representative number for a single folder. This study was implemented as a java browser client that uploads results to a multithreaded server. Source for this and the second study is available at <http://www.idiom.com/~zilla/Work/VisualIDs/Expcode> (note that studies 1,2 are referred to as Experiment 4 and Experiments 5,6 in this code).

Participants were recruited by email. 56 people started the study but 26 completed it despite the relatively short length. The high number of incomplete results was probably due to a java/browser incompatibility that caused the applet to crash when accessing the icon list (but after opening the server connection). This bug was fixed before the second study. One result was discarded as the elapsed time was several hours (the user presumably was interrupted and returned to complete the study later). Screenshots of Studies 1 and 1-precursor are shown below.

The second study used the same browser/server approach developed for study 1. 41 people did part or all of day 1, 21 completed both days, of these 9 were in the with-icon condition. The participants were split among graduate students and professional computer artists. The large number of incomplete results in this study was probably due to it being a more difficult multi-day study.

In followup comments some users in the generic condition said 'how were we supposed to know [what to answer]', meaning, the idea

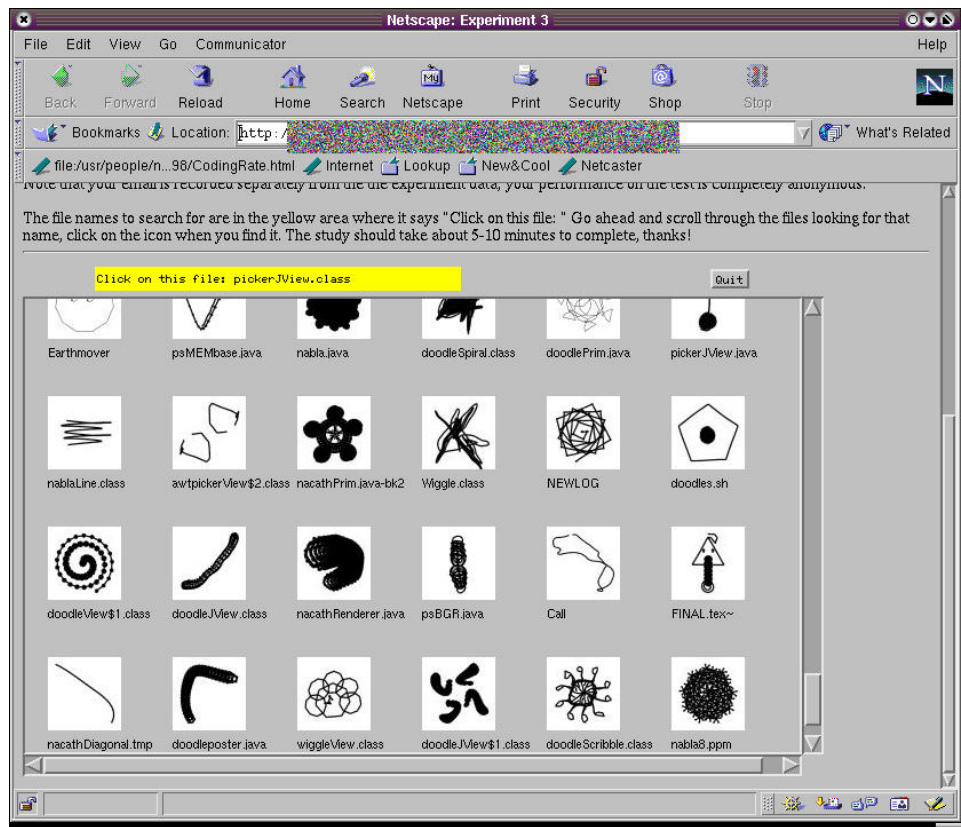


Screenshot of study 1 precursor.

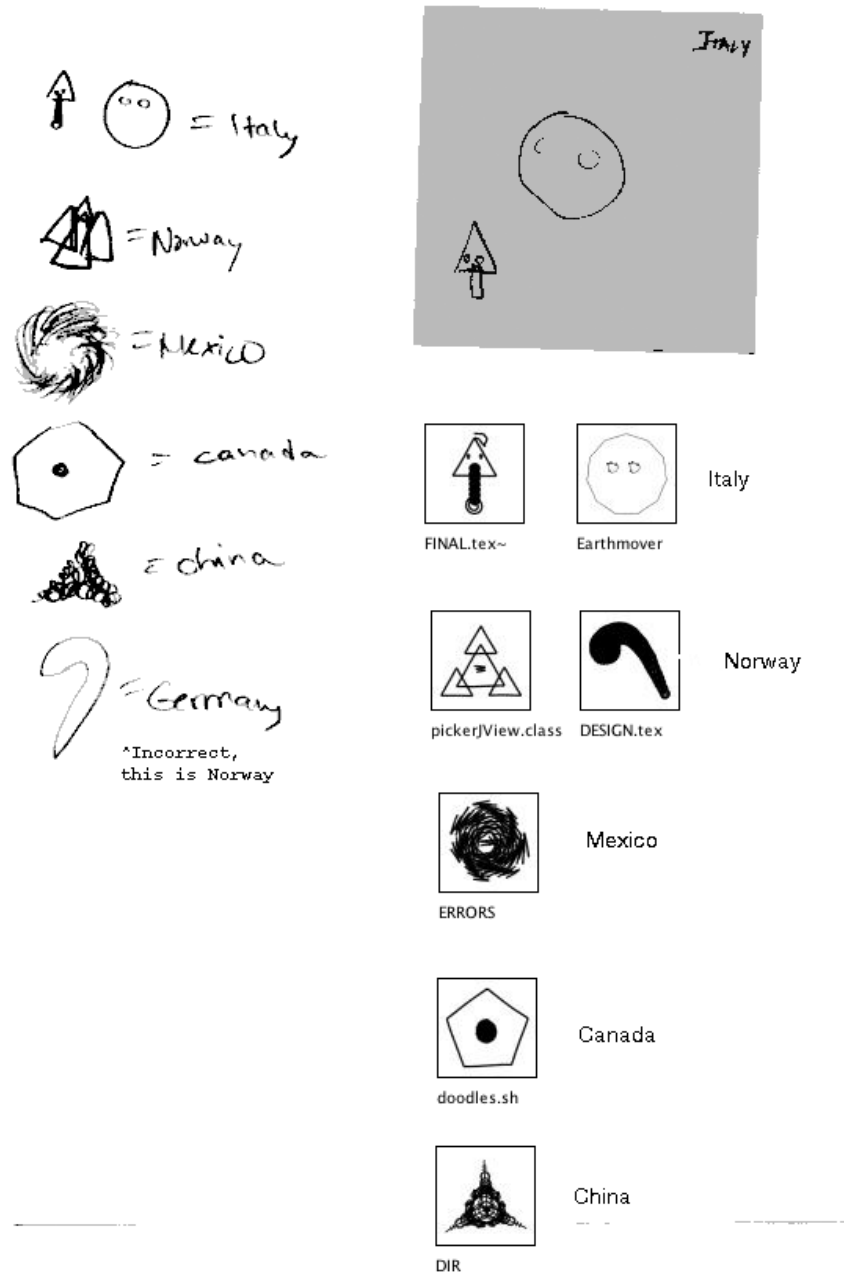
that it would be possible to learn a number of filename-content association with that amount of training did not even seem conceivable to them.

Roughly six weeks following study 2 we briefly quizzed an accessible sub-group of these people on their memory for the icons. Four people were given a printed sheet containing nine of the original icons and seven previously unseen icons; they were asked to indicate any icons they recognized. The results were remarkable: they got 4,2,2, and 1 choices incorrect, or more than 80% correct (this after only 10-20 minutes of exposure and more than a month of elapsed time).

In addition to the two studies reported in the paper, we have run a simple visual distinguishability test several times, in part to test the effectiveness of some additions to the icon grammar. People ran a program in which they viewed 100 icons for at their own pace but with instructions to spend 'a few seconds' on each icon. They then viewed a second set of 100 icons, half of which were new. The average results appear to be in the 70% range; one group of 5 people obtained an average of 73% correct for example (chance performance=50%). This level of recognition is below that obtained for real images but nonetheless demonstrates that artificially generated icons can be distinguished and recalled with little effort. This study is not reported in the paper because it is not testing any hypothesis and because the core issue (distinguishability) is tested in more realistic ways in the other studies.



Screenshot of study 1. The url was removed for anonymous submission to SIGGRAPH.



Examples of subjects' sketches of icons from memory several days after study 2, together with the corresponding original icons for comparison.