Towards a Pattern Language for Interactive Information Graphics

Christian Kohls, c.kohls@iwm-kmrc.de
Knowledge Media Research Center, Tuebingen, Germany

Tobias Windbrake, wb@fh-wedel.de
University of Applied Science Wedel, Wedel, Germany

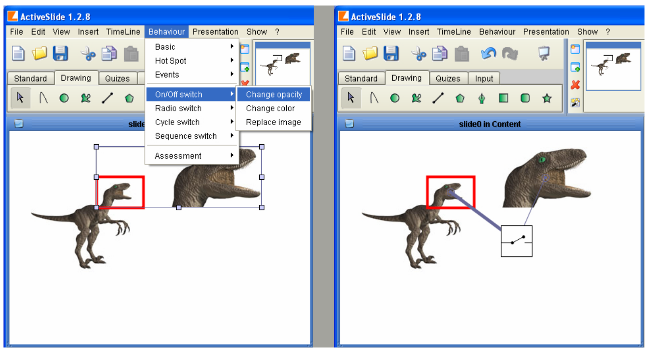
Abstract

Interactive graphics is a useful communication tool and new software tools make it easier than ever. However, few people know how to use interactive graphics properly and the new tools just allow them to build bad programs more quickly. This paper describes the start of a pattern language for using interactive graphics to communicate complex topics. It is based on our experience in education, but should also be helpful for modelling, news sites, and museums. The paper describes the structure of the pattern language, lists the 99 patterns that have been found so far, and gives brief descriptions of 16 patterns and detailed descriptions of 3.

1 Context for interactive information graphics

(Interactive) visualization helps people to analyze, understand and communicate models, concepts and data [SM00]. It is used for the exploration, monitoring, confirmation and presentation of information. There are many applications for such interactive graphics to present information, including news websites, edutainment kiosks in museums, self-paced learning courses and interactive whiteboards. According to the Multimedia Principle [MM99], there is evidence that the use of multimedia (that is, words and pictures) can improve learning. Dynamic and interactive pictures can be even more effective. Causal relationships can be directly and unequivocally perceived. Sequences of object or data movements can be captured. Processes of restructuring or rearrangements can be made explicit. Complex spatial actions can be shown. Cognitive load can be reduced by animations, as well, because the changes over time are directly shown.

We have mined (but not written all down) about 100 patterns by analyzing illustrations and multimedia applications, derived patterns from scientific research in information visualization, instructional design and cognitive theory, and have written down our own experiences in the design of interactive presentations. We became aware of the first patterns when we created demos with ActiveSlide, an authoring tool we developed for interactive presentations. In the first version of ActiveSlide, small scripts had to be written for each interactive feature. We ended up writing the same scripts again and again, just replacing element names and property values. Growing weary of this repetition, we soon isolated the reoccurring patterns of basic interaction. We integrated the patterns in the form of a visual programming language. It is now possible to assign patterns, such as SWITCH BETWEEN OBJECT STATES, SYNCHRONIZE OBJECT MOVEMENTS, DRAG RESTRICTION, TRANSPORT OBJECTS, PUSH OBJECTS, and CONTEXT SENSITIVE INFORMATION, directly to visual elements. This change saved a lot of scripting work and made the concepts easier to understand for end users.



Interaction patterns in presentation tools: The dinosaur and the enlarged dinosaur head are two images. A red box is put over the dinosaur and should trigger a property switch, i.e. change the visibility of the dinosaur head. The SWITCH BETWEEN OBJECT STATES pattern can be directly assigned to the elements.

However, in training sessions with teachers and pupils, we found out that many users understood the patterns of basic interaction, but did not know in which context to use them. For example, everybody understood what a SWITCH BETWEEN OBJECT STATES is and how it works. But using this pattern for DYNAMIC LABELS or to HIGHLIGHT INFORMATION was a more challenging task and required our assistance. Either the visualization techniques were not known or the users had difficulties in mapping them to the patterns included in the visual language. To capture our knowledge about visualization methods, we created a new class of patterns, such as ADJUST MOVEMENT SPEED, CHANGE LEVEL OF DETAIL, DETAILS ON DEMAND, MOVE OBJECTS TO COMPARE, and SELECTION-BY-THUMBNAILS. By linking these patterns to the basic interaction patterns, we made the mapping from visualization methods to interaction primitives explicit. The first step towards a pattern language had been made.

2 Patterns in human computer interaction

In the process of pattern mining, we were especially looking for patterns that were not available in ActiveSlide. One reason was to use the patterns as a specification for which new functions should be engineered into the tool. The other reason was to generalize the patterns, so that they can be of value whenever an interactive information graphic is designed – regardless of which (software) tools are in charge.

The resulting pattern language for interactive information graphics relates or overlaps with languages for human computer interaction [Ba97], [Bo01], [GLC01], [MJ98], [Tid05], the design of websites [WV03], [Sc05], [MLC05], [DLH04] and patterns for (e-)learning. The first patterns on education have been collected in the Pedagogical Patterns Project [PPP]. These kinds of patterns include, among others, the organization of courses or lessons (*Lay of the Land*, *Student Design Spring, Mini Project, Ask your Neighbour*) or assignments (*Questionnaire, Fill-In-The-Blank*). A similar approach was used in the German research project “Virtualization in the field of education”, which documents five years of research in instructional design as patterns [VW04]. The European research project E-LEN has created an E-Learning Design Pattern repository containing patterns in the categories “Learning resources and LMS”, “Lifelong learning”, “Collaborative learning”, and “Adaptive learning” [ND05].

3 Structure of the pattern language

The pattern language for interactive graphics consists of three levels:

1. Content-related visualization methods
2. Content-independent visualization methods that can support all types of content-related methods
3. Visualization and interaction primitives to serve visualization methods

The designer can choose from a set of content types. There are patterns to represent quantitative data, movements, processes, changes over time, structures, relations or the representation of objects. So, level one asks: “What does the graphic show?”

The second level provides visualization methods that are independent of the content, including strategies for comparing objects or data, presentation of small multiplies, context-sensitive information, reduction of complexity, focusing data, dynamic labeling, animation sequences and interaction sequences. Level two is about “How to show the graphic”.

To implement the methods, the designer will use primitive visual and interactive components, which are clustered according to their degree of interactivity. Patterns for static image components (e.g. use of shape and colors) are included here, because the rules apply for dynamic and interactive images, as well. Patterns for interaction are divided into patterns that navigate through the system (replace images, switch visual elements on/off) or actually change the system (change properties of visual elements, create new elements). On level three, the designer learns “Which primitive visual and interaction components to use”.

4 Example patterns

We have discovered 99 patterns so far, though most of them have not been described in detail. Even so, there is too much information for a short paper. Therefore, for each level, we will describe one pattern in detail, briefly describe another five or six patterns, and give a complete list of all the patterns in that level.

4.1 Detailed example of a content-related pattern

Let us assume the designer intends to create an interactive information graphic with the instructional intention to present information. He knows already that he wants to present information about some objects of the real world. So, he starts at level one where he finds the pattern OPTIMIZE OBJECT PERCEPTION.

Name: Optimize Object Perception

Problem overview:
In visual representations, you can both reduce or increase the amount of available information for objects and choose between many visualization alternatives, e.g. level of detail and level of abstraction.

Use when: Represent objects of the real world

Forces:

- The image should share a physical resemblance with the object it represents.
- Realistic images differ from abstract drawings in means of object recognition.
- People need sufficient time to scan and interpret visuals with many details.
- The performance to recognize an object depends on the viewpoint.
- Objects are 3D; screens are 2D.
- The interior of objects is not visible, but may be important to understand the internal structure of an object.
- Related text (e.g. labels) should be close to the object itself to avoid cognitive overload, but too many labels produce noise and may thereby increase the cognitive load.
- If the graphic accompanies text, then it should be congruent and relevant to the information in the text.

Solution summary:
Reduce the object to relevant details or allow variations of details, eliminate all distracting information, use an optimal viewpoint for the object and integrate explanatory text. Use DYNAMIC LABELING if there is a great deal of text you want to add.

Diagram: -- see examples --

Solution details:
Use pictures that clearly represent the object(s) of investigation and nothing else. Too much information can lead to decreased learning performance. This is especially the case if learners fail to extract the relevant information from complex graphics. Avoid decorative image elements in pictures that have representational purpose. The decoration may be disruptive. Overly decorative graphics may distract the learner's attention from the important information. Also, the learner could misinterpret the decoration as being part of the important information, which could even lead to misconceptions or, at the very least, to more cognitive work on the part of the learner to figure out what is relevant and what is not.

If only some of the details of the real object are of interest, use a schematic drawing rather than photographs. However, a good approach may be to first show a photograph (providing the context) and then transit the photo into a line drawing that overlays the photo, which finally fades out completely. This way you can provide the context of a realistic image and then point out the important information and which details are of interest. Instead of ANIMATION, you can use an interactive information graphic where the user can CHANGE THE LEVEL OF DETAIL for himself. You can also increase the number of details by ZOOMING into parts of a picture.

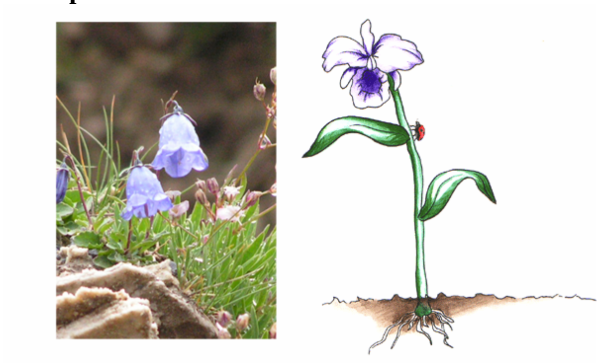
Most objects have canonical silhouettes that are easily recognizable, e.g. a cup of tea, a horse, a tree and so on. Whether you take a photograph or sketch a line drawing, try to show the object from a viewpoint where a canonical silhouette is perceived. If not all important aspects are shown from that viewpoint, you can use SMALL MULTIPLIES to show the object from different viewpoints at the same time. In an interactive information graphic, you can let the user CHANGE THE VIEWPOINT. Providing more than one viewpoint of the object can help to overcome the challenge of representing 3D objects in 2D space. However, there are many more techniques that PROVIDE DEPTH INFORMATION, for example, perspective clues, texture and size gradients, occlusion, depth of focus, cast shadows, shape-from-shading and structure from motion.

If you not only want to represent the object from the outside, but also from the inside, you may want to offer exploration techniques to VIEW INSIDE OBJECTS. In a CUTAWAY, you take some parts from the object away, thus revealing the interior. If you take away the car body, you can see what it looks like inside the car. An alternative would be a GHOSTING, where some object parts are set to semi-transparent rather then cutting them away completely. An EXPLOSION DRAWING also takes parts of the object away, but places them at a distance around the center; hence, you can learn about the interior and the structure of the object is made more explicit.

For complex objects, it is important to integrate explanatory text or short descriptions into the image. Printed word should be near corresponding graphics. Separating text from images can cause cognitive overload. However, adding too much text to the image can increase the complexity and make it more difficult to scan for information. Also, the resolution capabilities of computer screens are limited; thus, there may not be enough space to provide text for each detail of the object. You can provide text to one component at a time by using DYNAMIC LABELING.

Implementation: Trivial. All presentation and authoring environments allow the integration of image files. Most environments allow transition effects to blend over from one image to another, which can be used to fade in/out details, realism or labels. Changing the view or providing insights can be done by using multiple graphics on different slides.

Examples:



Left: a photograph showing flowers in a realistic environment
Right: The illustration shows a flower reduced to less detail. Also, from this viewpoint, the flower has a canonical silhouette.

Rationale:
Dwyer studied the learning effect of images that varied from highly detailed color photos to simple line drawings. Richly detailed visuals take time to scan and interpret. Learners may even skip graphics that appear to be complex [Dwy78]. The disruptive character of decorations in representational graphics is discussed in a research review by Levin, Anglin and Carney [LAC87]. Using silhouettes as a guideline to find a good viewpoint for objects is based on the theory that silhouette information might be used to extract the structure of objects [Mar82]. Information about representing 3D objects in 2D space can be found in [War04a]. The use of small multiplies for images that vary only in details is suggested in [Tuf90]. The integration of corresponding words in graphics is based on the contiguity principle by Clark and Mayer [CM03].
Related patterns:
The following patterns can also be of help: ANIMATION, CHANGE THE LEVEL OF DETAIL, CHANGE THE VIEWPOINT, CUTAWAY, DYNAMIC LABELING, EXPLOSION DRAWING, GHOSTING, PROVIDE DEPTH INFORMATION, SMALL MULTIPLIES, VIEW INSIDE OBJECTS, ZOOMING

4.2 Brief descriptions of content-related patterns

LEAVE TRAILS:
Use when: Show typical movement paths for an object.
Problem: For a moving object, you can only see where it is at that very moment; the former and succeeding positions of the object are volatile.
Solution: Draw a line along the path the object will move. Use arrows to indicate the movement direction. Use a different color or thickness for the line to indicate speed variations. If only some positions ahead/behind are of interest, showing multiple instances of the object with varying opacity is an option.
Example: Showing the movement of a ship on a map.

USE MULTIPLE CAMERAS:
Use when: Parts of objects are themselves in motion and there are dependencies between the moving parts.
Problem: If you see only one frontal view of the object, you cannot see all parts in motion at the same time. Hence, relations between parts in motion are difficult to discover.
Solution: Simultaneously use multiple cameras to show all relevant views of the object. Avoid a single camera that flies around the object, especially if many interrelated parts move at the same time.
Example: Show the motion of a steam engine.

EXPLOSION DRAWING:
Use when: You want to show single components of an object separately, but keep relative positions to show their structure.
Problem: Showing an object as a whole makes it hard to identify its single components. Also, some components may cover other components. But if you show each component alone, you may lose the context and not be able to see how the components fit together.
Solution: Let the object explode – split its components and move each component along the line that strokes through its own center and the center of the complete object. Let each object move the same distance; hence, the relative positions of the components are maintained.
Example: A photo camera is split up into its single components. All components are moved away from the center, so you can recognize each single component while still seeing the shape of the camera.

MULTIDIMENSIONAL TABLES:
Use when: You have multi-parameter data and want to categorize each entity in a table.
Problem: Standard tables provide only one or two dimensions.
Solution: If you take a two-dimensional table, you can split each cell into a table of its own, allowing up to 4 dimensions.
Example: A table showing the running results of a half-marathon. The x-axis splits running times, the y-axis splits age classes. In each cell on the left-hand side, there are names and exact results of female runners; on the right-hand side, the names and stats of male runners appear.

SHOW PROCESSING STEPS:
Use when: Representing a linear process from start to finish.
Problem: Only showing the start and finish state of a process does not show the process itself. Showing a complete process as an animation makes it difficult to access the relevant information for a certain state.
Solution: Provide a graphic for each relevant state of the process. Arrange the graphics in time or provide a navigation, which lets the user walk through the process steps. Keep each single graphic coherent with its successors in a way that the user can recognize the differences. You can use animation to transit from one process step to the next to clarify the change.
Example: Instruction manuals to assemble Ikea furniture or Lego toys.

4.3 Overview of content-related patterns

Show quantitative data:
Respect Data Types, Show Correlations, Show Histograms, Show Order,
Use Adequate Charts for Quantitative Data

Show structural data:
Adequate Use of Arrows, Layering and Separation, Multidimensional Tables, Optimize Tables, Use the Grammar of Maps, Use the Grammar of Node-Link-Diagrams, Show Compositions

Show object data:
Change the Viewpoint, Cutaway, Explosion Drawing, Ghosting, Optimize Object Representation, Provide Depth Information, View Inside Objects

Show processes:
Demonstration, Show Loops, Show Rhythm, Show Processing Steps, Use Flowcharts, Use State Diagrams

Show movements:
Leave Trails, Use Multiple Cameras

Quiz:
Arrangement, Drag&Drop Assignments, Fill-In-The-Blanks, Puzzle, Single/Multiple Choice Test

4.4 Detailed example of a content-independent pattern

The designer now has learned about how to represent objects. He finds DYNAMIC LABELING appropriate for his illustration and moves on to the related pattern of level 2.

Name: Dynamic Labeling

Problem overview:
Physical space limits the number of possible labels. Too many labels increase the complexity and make it harder to interpret the graphic.

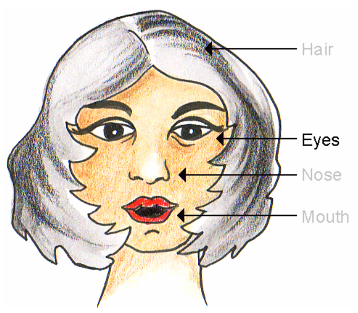
Use when:
- A lot of components of the graphic need to be labeled.
- Static labeling makes the graphic too complex or unreadable.
- Draw attention to only one or a limited number of components at a time.
- Provide explanatory verbal information in local contexts.
- Reduce the complexity of a graphic.
- You want to suggest an optimal sequence in which to read the graphic.

Forces:

- The user has to understand how to activate a dynamic label.
- The user may want to have more than one component to be labeled at the same time.
- To see all labels, the user has to activate each label manually.
- Labels can hide parts of the relevant information in the graphic.
- Readability of a label depends on its visual surroundings.

Solution summary: Dynamic labels appear on the screen only when needed.

Diagram:



Solution details:
The form of a dynamic label could be plain text that appears above the image area that is subject to be named or explained. You can also use a line or an arrow that links the image area of interest to a popup text that is placed somewhere else. By doing so, you avoid both the relevant area being covered by the text and the graphic interfering with the text. If text is placed beside (and not over) the relevant area, you can also put a box around the text to improve readability. A good combination of a textbox and a pointing arrow is a text bubble.

Of course, the dynamic label does not have to be text. You can also dynamically show sub-graphics within the main graphic. For example, the popup graphic could show a different view, further visual explanation or DETAILS ON DEMAND of the labeled area. Labels can show an enlarged picture of a selected area and are one approach for pre-defined ZOOMING.

Activating the labels should be made easy for the user. First of all, the user has to know that dynamic labels are available. Second, he needs instructions on where to find and activate them. If you have a single graphic using dynamic labels, you should write a hint that dynamic labeling is available. If you have many graphics with dynamic labeling, e.g. a web-based training course, you should give a brief instruction at the beginning. The instruction could include an animated DEMONSTRATION.

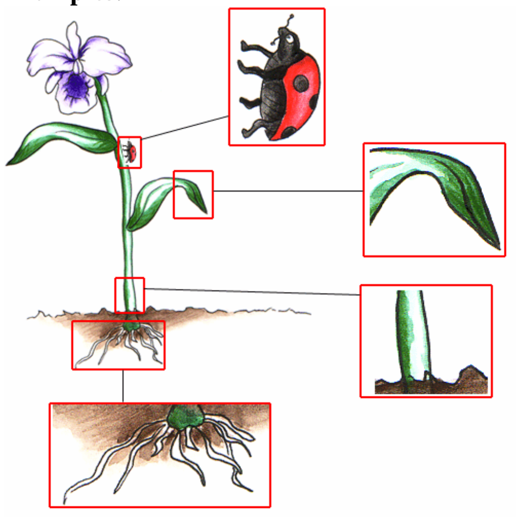
The simplest way to activate a dynamic label is a ROLL-OVER with the mouse. The user finds out very quickly where labels are by moving the mouse over areas he is interested in. He may not find all labels, but he finds all the labels he is interested in. The mouse pointer acts - as the name states – as a pointer. Thus, connection lines between the relevant area and the label are not required. The label could even appear outside the graphic. As soon as the mouse exits, the dynamic label disappears, which is useful to immediately reduce complexity again. However, if the label needs to be shown for a longer time frame, keeping the mouse pointer within the relevant area can be difficult. It also prevents the user from doing other tasks without losing the label. Showing more than one label is not possible using mouse roll-overs.

To keep a label permanently visible, you can use a SWITCH BETWEEN OBJECT STATES. If the user clicks on the relevant area, the label appears. To let it disappear, the user has to switch it off explicitly. This could be done by clicking on the label or on the relevant area. This way, more than one label can be displayed at the same time. Another option is to switch the label off as soon as another label is selected. This allows only one label at a time. Showing only one label can be useful in presentations to direct the audience's attention and HIGHLIGHT INFORMATION. If there are many dynamic labels - or each label is large in size - then multiple viewed labels would overlap. Automatically switching off one label as soon as another is switched on reduces the mouse clicks to move on from one label to another.

If activation of dynamic labels is done by clicking on HOT AREAS, the user needs to know where these clickable areas are. You can provide visual clues, such as drawn bounds or small circles, to indicate where label activation is possible. Many graphics, however, implicitly show areas, e.g. on a map, in which the bounds for different labeling are quite obvious. You can also use ROLL-OVER and a SWITCH BETWEEN OBJECT STATES in combination. Moving the mouse over an area can popup the label temporarily and clicking with the mouse will keep it permanently. The roll-over label could actually differ from the “permanent” label: it can act as a label preview (for large-sized labels) or as an indicator that a dynamic label is available (e.g. a “click here” popup or highlighting the bounds of the HOT AREA).

Implementation:
Most authoring environments allow you to dynamically show or hide components and you can implement the solution using a SWITCH BETWEEN OBJECT STATES. In contrast, presentation software usually does not provide features to switch on and off parts of a graphic. However, most presentation tools allow navigation and hyperlinking. If you run through a predefined sequence of dynamic labeling, you can just multiply the same graphic on a sequence of slides and use only one label per slide. The standard functions to step forward/back will guide the user from one label to the next. If each label should be accessible at any given time, you must provide hyperlinks on all slides showing the graphic. For the hyperlinks, use invisible objects (with transparent opacity) and place them at the areas where a mouse click should pop up the label. This method, of course, is only practicable if the number of labels is small.

Examples:



Dynamic labeling is used to show details of the flower. If the mouse pointer enters one of the small red boxes on the flower, then one of the larger boxes showing the details pops up. The enlarged views are linked to the areas they magnify by a line.

Rationale:
There are psychological reasons to use dynamic labeling. The working memory is limited and too much irrelevant information can cause cognitive overload. Dynamic labeling can provide all needed information just when needed. Also, you can place each label closer to the related area of the graphic, thereby following the contiguity principle [CM03].
Selectivity of attention is another reason to use dynamic labeling. Using only some instead of many labels gives the learner hints as to which areas he should pay attention. This is especially useful in presentations. The appearance of a new object in the visual field attracts attention itself [HY94].

Related patterns:
This pattern can be used by: DETAILS ON DEMAND, HIGHLIGHT INFORMATION, OBJECT REPRESENTATION, ZOOMING
This pattern can use: DEMONSTRATION, HOT AREAS, ROLL-OVER, SWITCH BETWEEN OBJECT STATES

4.5 Brief descriptions of content-independent patterns:

MOVE OBJECTS TO COMPARE:
Use when: You have small visual entities and observers are supposed to detect similarities, differences or relations.
Problem: Large spatial distances distract from making the comparison, because the viewer cannot focus on both entities at the same time.
Solution: Make each entity moveable. Provide a blank area on the screen as a container for two or three entities chosen by the user. If dragged into that area, the entities should be close enough to be focused on at the same time. To support a comparison, you can provide moveable rulers or structured backgrounds, e.g. grids.
Example: Small images of butterflies. Users want to compare the colors and patterns of the wings. By placing two butterflies side-by-side, it is easy to compare the differences or similarities.

POSITION-DEPENDENT PROPERTIES:
Use when: You want to visualize dependent and independent values.
Problem: Find a mapping between the spatial position of an element and its visual representation.
Solution: For the independent values - e.g. maps, coordinate systems, timelines - use a background graphic. For the dependent values, use small moveable graphics that overlay the background. Define HOT AREAS, in which the moveable graphics change their visual appearances according to the independent value(s) of that area. To define the different visual appearances, you can SNAPSHOT PROPERTIES.
Example: A map of the United Kingdom represents independent spatial data. An image showing a flag of the United Kingdom represents the dependent data. If the image is dragged over Scotland, it shows the Scottish flag. Dragged over England, it shows the English flag.

SEQUENCED GRAPHICS:
Use when: You want to show changes step-by-step; you want to add new components step-by step.
Problem: Putting relevant states of a system or object in one single graphic makes reading and interpreting hard. Representing relevant states in multiple graphics consumes a lot of space and requires the user to detect the actual changes by himself.
Solution: Use only one graphic that changes in a predefined sequence. You can replace the complete graphic, parts of it or change some of its visual properties. Provide at least navigation buttons to move forward or backwards to the next/previous step of the sequence. You can also provide one button for each step in the sequence, letting the user jump to each state at any time. To label the buttons, you can use numbers, names of the represented states or thumbnails.
Example: Constructing a complex finite automaton or an organization chart step by step.

ADJUST MOVEMENT SPEED:
Use when: Objects are moving very fast or slow; the movements are very complex.
Problem: The cognitive visual processing system of humans is limited in the number of frames it can process per second, in the number of entities it can consider at a time, and in the (position) differences it can detect.
Solution: Allow the user to control the animation speed so that he can slow down fast or complex movements and speed up slow motions according to his needs. Be aware that adjusting the animation speed costs cognitive resources, too. Therefore, offer optimized default speeds (do some runs with test users) that work for most users. To let the user find the best speed, offer a simple way to repeat parts of the animation, e.g. use a TIMELINE SLIDER.
Example: Slow motion of sport activities, e.g. which way the ball actually moved into the goal.

HIGHLIGHT CLASS MEMBERS:
Use when: You want to show which entities or visual areas are related.
Problem: Entities are perceived as part of a class by visual similarity. If entities of different classes look very similar, it is impossible to identify the classes visually (e.g. in a list of numbers, how do you visually distinguish between prime and non-prime numbers?). Also, if entities of the same class look very different or are separated by great distances, you will not relate them easily.
Solution: Whenever the user clicks on (or rolls over) an entity, visually highlight all entities of the same class. To highlight the entities, you can draw a shape around each entity. The shapes should popup strongly, so that the user can locate all of them. If the entities are connected to each other, you can also highlight the types of connections.
Example: A map of a city showing bus lines and stops. If the user clicks on any bus stop, then all other bus stop symbols of the same line are highlighted using a brighter color. Also, the street sections that connect the bus stops are overlaid with a semi-transparent stroke.

4.6 Overview of content-independent patterns

Comparison strategies:
Move Objects to Compare, Selection-By-Thumbnails, Selection-By-Timeline, Small Multiplies, Use Brushing, Use Multiple Zooms

Context-sensitive information:
Development Along Paths, Dynamic Labeling, Position-Dependent Properties, Provide Orientation Information, Timeline Slider, Visit Stations

Reduce complexity:
Adjust Movement Speed, Change Level of Detail, Details On Demand, Filter Data or Objects, Integrate Micro-/Macroviews, Sequenced Graphics, Zooming

Direct attention:
Highlight Areas, Highlight Changes, Highlight Class Members, Highlight Objects, Use Data Pointer, Use Speech Bubbles

4.7 Detailed example of an interaction primitives pattern

Now the designer has an idea what dynamic labeling can do for his illustration. To implement the labeling in his interactive graphic, he decides to use a SWITCH BETWEEN OBJECT STATES.

Name: Switch Between Object States

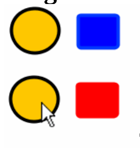
Problem overview:
Change parts of the graphic without replacing the complete graphic.

Use when:
- You want to hide or show parts of the graphic only temporarily
- You want to change visual properties, such as color or opacity, for some visual components to highlight them (or blur them out).
- You want to use the same area of the screen for changing graphical representations while the rest of the screen does not change.
- You want to visually represent On/Off states.

Forces:
- The complete information graphic must be aggregated from smaller graphical components in a way that each component can be altered independently.
- There must be a clear trigger that changes visual components.
- Two different visual states are required for each visual component that can change.
- The initial visual state must be defined.

Solution summary: Use one visual component as a trigger to change its own properties or those of other components.

Diagram:



The circle switches the color of the rectangle if the user clicks on it.

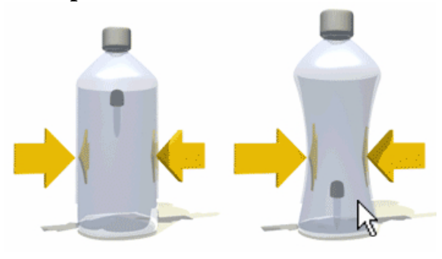
Solution details:
USE SMALL GRAPHICS to construct larger graphics. Each graphical component of the information graphic can then be manipulated independently from all other graphical components. If you have a graphic in one image file, you will need to use an image processing tool to split it into smaller components. For each changeable area of the information graphic, you need at least one separate image. Each of the graphical components can now be changed. Mouse clicks (and other events) on a graphical component can work as a trigger to switch between two states of another component. The most common operation is to show or to hide a graphic component, that is, to switch between the states *visible* and *invisible*. Because we always switch between two states, we can call these *On* and O*ff* states. The *On* state would be the visible graphic component; the *Off* state would be the hidden graphical one. Visibility is only one property you can change. Depending on the authoring environment you use, you can change opacity, size, color, orientation and other properties of the graphical component. For each property, you have to define two values, one representing the *On* state, the other representing the *Off* state. You can use different colors or sizes of a graphic component to HIGHLIGHT INFORMATION: The *Off* state is a small scale of the graphic component. If you switch the component to *On*, then the larger scale is used and the component stands out more.

Another interesting property to be changed is opacity. You can use a low opacity as the *Off* state. The graphical component can be perceived, but does not distract the user. The user is aware of the graphic component and can switch it *On* to full opacity the moment he wants to investigate it. The opacity can also work the other way around: If the *Off* state is full opacity, the graphic component hides everything that lays behind it. Switching the component *On* would actually reduce the opacity to semi-transparent so that the user can see both the graphic component and what is behind it. You can use this for GHOSTING.

Graphic components can also switch their own properties. For example, if you want a graphical component to display differently when you click on it, then the component switches itself from one state to another. It is also possible that you use one component as a trigger to switch the states of many other components in one single operation. You have to decide which mouse events actually trigger the switch and whether the resulting state is *On*, *Off*, or the reverse of the current state. A mouse click could switch another element always to *On* or reverse its current state. You can use mouse enter events to switch to the *On* state and mouse exit events to switch to the *Off* state; in this case, your switcher behaves as a ROLL-OVER component. Similarly, you can use mouse down and mouse up as *On* and *Off* switches. In this case, the state is only *On* as long as the mouse button is pressed on the trigger component. This is an interesting option to use switches for DYNAMIC LABELS.

Implementation:
Presentation tools usually do not support switching between states. Authoring environments usually provide a simple way to set properties. In Macromedia Flash, you can use sub-movies and define different states in each frame of a movie clip. In ActiveSlide, you can use the visual language to link two (or more) visual elements. The first linked element triggers the switch; all other elements are switching their states.

Examples:



Moving the mouse over the image replaces it with another image. The image component switches between visual states.

Rationale:
Buttons are standard components in user interface design. Switches are similar to buttons, but provide some additional features useful for interactive information graphics. First, switches always provide a very specific operation: changing properties to represent one of only two states. Second, switches know the current states of all components they can switch. A switch is more like a light switch that can turn on and turn off the lights in your room. The light stays on until you switch it off again.

Related patterns:
The pattern can be used by: CHANGE VIEWPOINT, COMPARE ITEMS, CUTAWAY, DETAILS ON DEMAND, DYNAMIC LABEL, GHOSTING, HIGHLIGHT INFORMATION, LAYERING AND SEPARATION, LEVEL OF DETAIL, REDUCE COMPLEXITY

4.8 Brief Descriptions of Interaction Primitives Pattern

DISTINGUISHABLE COLORS:
Use when: Use color for labeling; exploring multidimensional discrete data.
Problem: For rapid identification of similar and different colors, the colors must be well-distinguishable. The perception of colors depends on their environment: other nearby colors, hues, contrast with background, images that have been seen before.
Solution: Use color with maximum distances in color space. Do not use different saturations of the same color unless you want to represent different values for one class of entities. Here is a list of 12 colors recommended by Colin Ware for use in coding: Red, Green, Yellow, Blue, Black, White, Pink, Cyan, Gray, Orange, Brown, Purple.
Example: Showing on a map which political party has won in which electoral district.

RESPOND TO INTERSECTION EVENTS:
Use when: Collision detection; dynamic grouping of elements; object intersection has a meaning.
Problem: Intersecting elements should trigger an action.
Solution: Define a geometrical intersection event. The event includes visual elements that can intersect and trigger an event. Define an action list that is executed when the event is fired. Make some of the elements moveable. Whenever one of the elements has moved (e.g. the user drags it to some location), check if the conditions for the intersection event are satisfied. If so: execute the action. The action can be any sort of code. You can use SNAPSHOT PROPERTIES to immediately respond visually to an intersection event.
Example: In a quiz, the user is asked to put a substance in the corresponding pot. If intersected with the correct pot, an action is triggered which shows some text with positive user feedback.

SNAPSHOT PROPERTIES:
Use when: You want to apply preset property values to an element, e.g. set a new position.
Problem: Manually entering property values into a script or property field is time-consuming and a source for errors. Mapping alphanumerical values to visual appearance is difficult.
Solution: Manipulate visual elements “on stage” in a WYSISWYG editor and record property values for later use. The recorded snapshot can be applied unlimited times when the interactive graphic is in use. The recorded property values can be set directly to an element or an element can animate smoothly from its current property values towards the snapshot values. Snapshot properties can be used as an action for geometrical events, such as RESPOND TO INTERSECTION EVENTS.
Example: The user places a picture of a plane at the left top of the screen and makes a snapshot. He assigns the snapshot to a button. When using the graphic, end users can click the button and the plane moves to the recorded position.

TELEPORT OBJECTS:
Use when: Automatically teleport an object from A to B.
Problem: A moving object enters an area where it has to disappear because of the background’s visual semantic. An example is a moving car that is dragged over a tunnel entry. After a while, the element is supposed to reappear at a different place (e.g. the tunnel exit).

Solution: Use one visual element that is moveable. For this element, define two HOTSPOT AREAs, each defining both entry and exit point. If the visual element is dragged, check whether it intersects with one of the areas. In that case, let the visual element disappear and, after a preset delay, let it reappear in the opposite area.
Example: A person is dragged over a door of a house. The person disappears and reappears at a balcony.

TRANSPORT OBJECTS:
Use when: You want to represent the process of transportation; you want to dynamically group elements in a container.
Problem: Show explicitly which objects can transport other objects, at which times objects are loaded or unloaded, and at which time transportation starts and stops.
Solution: Define one visual element as a transport container and one or more elements as transportable objects. All elements must be independently moveable. However, if a transportable object is completely inside the transport container, it moves along with it. The transportable objects are temporarily synchronized with the transport container, but remain moveable. Thus, they can be dragged out of the container again.
Example: A blood cell transporting particles. The particles move along with the cell only if they are inside the cell.

SYNCHRONIZE OBJECT MOVEMENTS:
Use when: Movements of an object cause movements of other objects, too.
Problem: The movement relations between objects need to be defined.
Solution: Link two objects and observe the movements of each. Whenever one object moves, synchronize the other object by letting it perform the exact same movement. Alternatively, you can apply a transformation to the original movement vector, including rotation and speed. For example, a rotation of 180 degrees would let object A move to the right when object B moves to the left.
Example: Demonstration of a lifting block. The user pulls a person horizontally and the block moves vertically in synchronization.

4.9 Overview of interaction primitives patterns

Visual primitives:
Distinguishable Colors, Organize Working Space, Pop-out Information, Respect Cultural Differences, Use Gestalt Laws, Use Light Colors for Grids, Understandable Symbols

Primitive manipulation:
Change Properties, Create Objects, Direct Manipulation, Drag Restrictions, Group Objects, Hot Areas, Indirect Manipulation with GUI Elements, Push Objects, Remove Objects, Roll-Over, Switch Between Object States

Geometrical events:
Respond to Containment Events, Respond to Drop Events, Respond to Intersection Events, Respond to Movement Events, Respond to Pattern Identification Events

Navigation between images:
Bookmark Positions, Flip Through Pages, Show Current Position, Use Standard Navigation

Animation:
Keyframe Animation, Path Animation, Predefined Effects, Rule Based Animation, Snapshot Properties

Visual dynamics:
Bumper, Define Constraints, Define Movement Rules, Flow Fields, Physical Forces, Relative Changes, Restricted Areas, Synchronize Object Movements, Teleport Objects, Transport Objects

5. Outlook

For the pattern language, we are currently working on a website that gives straightforward access to all patterns and which mirrors the structure of the language, too. For the diagrams, we produce interactive explanations; for the example sections, we produce small Flash movies and try to collect web links of pages that use the patterns. Each pattern can be commented on and rated by online users. Rating scales include dimensions, such as readability, appropriateness and usefulness.

In the next school term, we will ask groups of teachers, students and pupils to design interactive information graphics for topics from their own domain. One group will get access to the patterns to perform the task. A control group without knowledge of the pattern language has to design graphics for the same topics. At the end of the term, we can analyze whether the knowledge of our design patterns led to a better output. Also, evaluating the solutions of our experimental groups can uncover new patterns, e.g. if many teachers or students come up with similar ideas to interactively visualize a particular topic.

We will advance the implementation of patterns in our authoring tool. Since many patterns have solutions that require step-by-step work procedures, we experiment with wizards that map to pattern solutions. Wizards are very simple to use while covering a lot of expert knowledge without letting the user recognize it. In contrast to written manuals, wizards not only capture all instructions, but produce a working output, as well. Each step allows various configurations, so the user can learn about different options for using a pattern. We are aware that the stringent flow of wizards may limit the creativity of users. The balance between flexibility and simplicity in creation in general is an area of conflict. Should designers implement the interaction patterns from scratch, e.g. in Java, allowing the most flexibility? Or, at the other end of the spectrum, are ready-to-use templates the better approach? Are wizards for content creation a fair compromise? To answer these questions, in the next few years, we will have to analyze the design outputs from the different methods of implementation.

References

[Al77] Alexander, C.; Ishikawa, S.; Silverstein, M.; Jakobson, M.; Fiksdahl-King, I.; Angel, S.: A Pattern Language; New York: Oxford University Press, 1977

[Al79] Alexander, C: The Timeless Way of Building. New York: Oxford University Press, 1979

[Ba97] Bayle, Elisabeth et al.: Putting It All Together: Towards a Pattern Language for Interaction Design, Summary Report of the CHI’97 Workshop

[Bor01] Borchers, Jan: A Pattern Approach to Interaction Design; John Wiley & Sons, 2001

[CM03] Clark, R.C.; Mayer, R.E.: e-Learning and the Science of Instruction, Pfeiffer, San Francisco., p67-81. 2003

[DLH04] Duynie, Douglas K. van; Landay, James A.; Hong, Jason I.: The Design of Sites, Addison-Wesley, 2004

[Dwy78] Dwyer, F. M.: Strategies for improving visual learning. State College, PA: Learning Services. 1978

[Gam95] Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Reading, Mass: Addison-Wesley, 1995

[GLC01] Granlund, Asa; Lafrenière, Daniel ; Carr, Daniel A.: A Pattern-Supported Approach to the User Interface Design Process, Proceedings of HCI International 2001, 9th International Conference on Human-Computer Interaction, 2001, New Orleans

[HY94] Hillstrom, A.P.; Yantis, S.: Visual attention and motion capture. Perception and Psychophysics 55(4), p.399-411. 1994

[LAC87] Levin, J., Anglin, G., & Carney, R.: On empirically validating functions of pictures in prose. In D. Willows & H. Houghton (Eds.), The psychology of illustration, volume 1: Instructional issues (pp. 51-85). New York: Springer-Verlag. 1978

[Mar82] Marr, D.: Vision. W.H. Freeman and Company, New York. 1982

[MD03] Meszaros, Gerard; Doble, Jim: A pattern language for pattern writing. http://hillside.net/patterns/writing/patterns.htm, (accessed: 10.18.05)

[MM99] Moreno, R.; Mayer, R.E.: Cognitive Principles of Multimedia Learning: The Role of Modality and Contiguity, Journal of Educational Psychology, 91, p. 358-368, 1999

[MJ98] Mahemoff, M.; Jonston, L.: Principles for usability-oriented pattern language, OZCHI ’98 Proceedings, Adelaide, Australia, S. 132-139

[MLC05] Malone, E.; Leacock, M.; Wheeler, C.: Implementing a Pattern Library in the Real World: Yahoo! Case Study. http://www.leacock.com/patterns/ (accessed 04.01.06)

[ND05] Niegemann, Helmut M.; Domagk, S: ELEN project Evaluation Report, Report of Work package 5. E-LEN project: a network of e-learning centres; http://www2.tisip.no/E-LEN/documents/ELEN-Deliverables/Evaluation_Report_E_LEN.pdf (accessed 03.29.06)

[Pav91] Paivio, A.: Dual coding theory: Retrospect and current status. Canadian Journal of Psychology, 45, 255-287, 1991

[PPP] The Pedagogical Patterns Project, http://www.pedagogicalpatterns.org/

[Rie90a] Rieber, L.P.: Computers, graphics, & learning. Englewood Cliffs, NJ: Prentice Hall, p. 15, 1990

[Rie90b] Rieber, L.P.: Computers, graphics, & learning. Englewood Cliffs, NJ: Prentice Hall, p. 29, 1990

[Rie90c] Rieber, L.P.: Computers, graphics, & learning. Englewood Cliffs, NJ: Prentice Hall, p. 129, 1990

[Sc05] Schmitt, Silke; Schreiner, Martin; Timmesfeld, Fel; Vucica, Martina; Wallach, Dieter: PatternCube.com: Ein webbasiertes Repository für User Interface Design Patterns. In: Hassenzahl, M.; Peissner, M. (Hrsg.): Usability Professionals 2005

[SM00] Schumann, H.; Müller, W.: Visualisierung. Grundlagen und allgemeine Methoden, Springer, Berlin, 2005

[Tid05] Tidwell, Jenifer: Designing Interfaces, O’Reilly, Sebastopol (2005)

[Tuf90] Tufte, E.R.: Envisioning Information. Graphics Press, Cheshire, CT. 1990

[Vl97] Vlissides, John: Patterns: The Top Ten Misconceptions, Object Magazine, 1997

[VW04] Vogel, R; Wippermann S.: Dokumentation didaktischen Wissens in der Hochschule Didaktische Design Patterns als eine Form des Best-Practice-Sharing im Bereich von IKT in der Hochschullehre, Wissenschaftsforschung Jahrbuch 2004, Berlin. 2005

[War04a] Ware, C: Information Visualization – Perception for Design. Morgan Kaufmann Publishers, San Francisco., p.259, 2004

[War04b] Ware, C: Information Visualization – Perception for Design. Morgan Kaufmann Publishers, San Francisco., p.303-307, 2004

[WV03] van Welie, M.; Veer, van der Gerrit, C.: Pattern Languages in Interaction Design: Structure and Organization, Interact 2003