#	TITLE	NUMBER OF PATTERNS
S1	A Classification of Design Patterns to Support Mobile Groupware Systems	3
S2	A Two-Phase Method of User Interface Adaptation for People with Special Needs	3
S3	Automated Usability Testing for Mobile Applications	4
S4	Design patterns for touchscreen-based mobile devices: users above all!	1
S5	Design Patterns for User Interface for Mobile Applications	14
S6	From Requirement to Design Patterns for Ubiquitous Computing Applications	2
S7	HCI Design Patterns for PDA Running Space Structured Applications	4
S8	Interaction patterns for Windows 8 tablet applications	5
S9	Method for mobile user interface design patterns creation for iOS platform	15
S10	Musical interaction patterns: communicating computer music knowledge in a multidisciplinary project	4
S11	Patterns for Interactive Line Charts on Mobile Devices	4
S12	Patterns of trust in ubiquitous environments	1
S13	RUCID: Rapid Usable Consistent Interaction Design Patterns-Based Mobile Phone UI Design Library, Process and Tool	1
S14	Spatial data and mobile applications - general solutions for interface design	2
S15	Speech Augmented Multitouch Interaction Patterns	6
S16	Test Patterns for Android Mobile Applications	2
S17	Towards User Interface Patterns for ERP Applications on Smartphones	1
S18	UI Design Pattern-driven Rapid Prototyping for Agile Development of Mobile Applications	108
S19	Usability-Improving Mobile Application Development Patterns	1
S20	User Interface Patterns for Multimodal Interaction	12
S21	Web design patterns for mobile devices	21
B1	Designing Mobile Interfaces- Steven Hoober, Eric Berkman	76
B2	The Essential Guide to Mobile Design Patterns: A Deeper Look At the Hottest Apps Today	46

S1 - ARTICLE: A CLASSIFICATION OF DESIGN PATTERNS TO SUPPORT MOBILE GROUPWARE SYSTEMS

PATTERN: #1

4.4.1 Pattern 1. Communication dimension
Pattern name: Participation request.
Problem: What to do for a group member who request to speak in a synchronous communication process through mobile groupware interface?
Usability principle: Match between the system and real world.
Context: Interaction is one of most important factors to consider in the communication process. Group members can intervene in a conversation by raising their hand when interested in speaking. Such interventions encourage participation among members of a group, a defined characteristic for a group to be effective. Campos [5] defines this type of intervention as a bodily gesture kinesic gesture, interaction regulator– used to regulate interventions in a conversation.
Solution: The groupware user interface should provide group members a symbolic object that allows them to generate this kind of interventions. The user interface should represent the users' language, verbal (oral and written) and non-verbal (symbols and signs, which are familiar to users).
Example: Figure 5 shows the (raise-hand) button on an interface, which allows the group to claim the floor during a conversation.

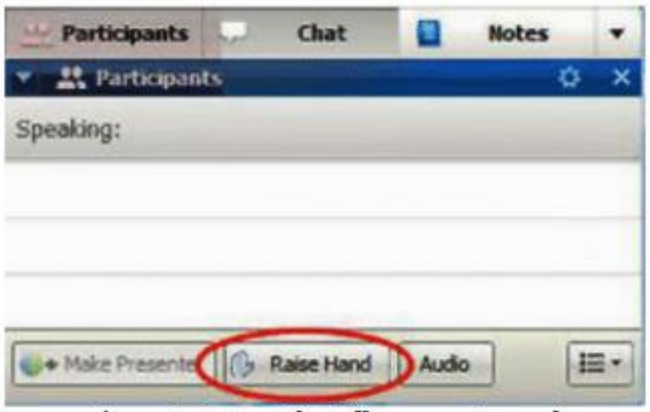


Figure 5. Button that allows one to speak, courtesy [6].

Gesture: Touch, tap.
Related patterns: Activity request, mediate the participation.

PATTERN: #2

4.4.2 Pattern 2. Coordination dimension
Pattern name: Activity request.
Problem: How to make a team member requests an activity through the interface?
Usability principle: Match between the system and real world.
Context: In the context of a meeting, activities and tasks are an approach of group work, and that work is focused on achieving the group's goal. The activities and

tasks can be requested by group members in different ways, e.g. several participants share a list of tasks in which they have to work on, and each member takes an activity or task in which he/she will work on.
Solution: The interface should allow the group members to request and select an activity within the assigned group tasks.
Example: Figure 6 shows two examples of design patterns that can be used to design the interface.

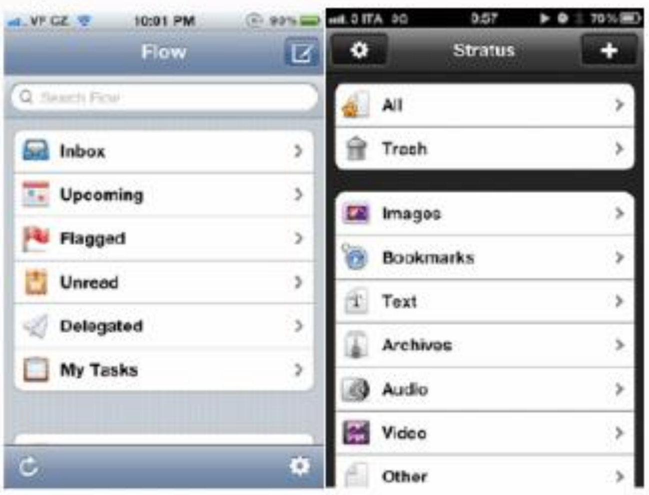


Figure 6. List menus: Valspar Paint and Kayak, courtesy [18].

Gesture: Vertical swipe, horizontal swipe, touch, tap.
Related patterns: Participation request, activity select.

PATTERN: #3

4.4.3 Pattern 3. Collaboration dimension
Pattern name: Complementary collaboration tools.
Problem: A group member cannot carry out his/her activities, collaborate or communicate with other group members because the interface does not allow these actions.
Usability principle: Match between the system and real world.
Context: Complementary tools are all materials and technological equipment that allows group members to collaborate. These tools serve to reinforce verbal communication –stimulation of the left hemisphere of the brain– and nonverbal –stimulation of the right hemisphere of the brain–, therefore both hemispheres are stimulated. This pattern deals with interactions, collaborations and support among group members.
Solution: The interface should provide group members the right tools to allow them to participate and collaborate synchronously or asynchronously, that is, it should provide the needed resources to achieve the goals of the group.
Example: Figure 7 shows two examples of design patterns that can be used to design the interface.

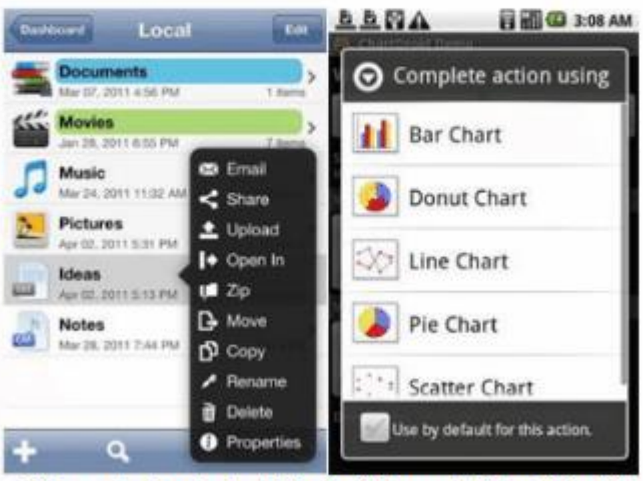


Figure 7. Contextual Tools: Files and ChardDroid, courtesy [18].

Gesture: Touch, tap, vertical swipe, horizontal swipe. Related patterns: Participation offer, activity delegation, encouraging feedback.

S2 - ARTICLE: A TWO-PHASE METHOD OF USER INTERFACE ADAPTATION FOR PEOPLE WITH SPECIAL NEEDS

PATTERN: #4
Pattern name Interface element audio output Problem User can’t recognize Interface Element to read text on it Context Vision Impairment Range > Medium and User’s Device has speakers Solution Replace Interface Element with Audio Output Interface Element
PATTERN: #5
Pattern name Interface element audio input Problem User can’t use Interface Element to input data into it Context (Vision Impairment Range > Medium or Motorics Impairment Range > Low) and User’s Device has microphone Solution Replace Interface Element with Audio Input Interface Element
PATTERN: #6
Pattern name Audio output volume increasing Problem User can’t hear Feedback from application Context Time of Waiting >30 s and User’s Device has speakers Solution Repeat Last Command with Current Volume + 20%

S3 - ARTICLE: AUTOMATED USABILITY TESTING FOR MOBILE APPLICATIONS

PATTERN: #7
Name Fitts's Law Problem Specification User misses UI-Element (e.g., button) several times Solution Make UI-Element bigger and/or move to center Weighting Major usability problem (3) Reference (Fitts, 1992) & (Henze and Boll, 2011)
PATTERN: #8
Name Silent Misentry Problem Specification User repeatedly touches UI elements without functionality (e.g., imageview) Solution Analyze pressed UI-element and figure out which functionality the user intended; e.g., imageview: image-zoom; add functionality or make clear function is missing Weighting Cosmetic problem only (1) Reference -
PATTERN: #9
Name Navigational Burden Problem Specification User switches back and forth between two views multiple times (e.g., master-/detailview) Solution User is looking for some information which is presented in detailview; needs the way over the masterview to open a new detailview Weighting Minor usability problem (2) Reference (Ahmad et al., 2006)
PATTERN: #10
Name Accidental Touch

Problem Specification
User touches the screen accidentally and and activates view change; he immediately revokes input
Solution
Check accidentally touched UIelement; move/resize/remove it
Weighting
Cosmetic problem only (1)
Reference
(Matero and Colley, 2012)

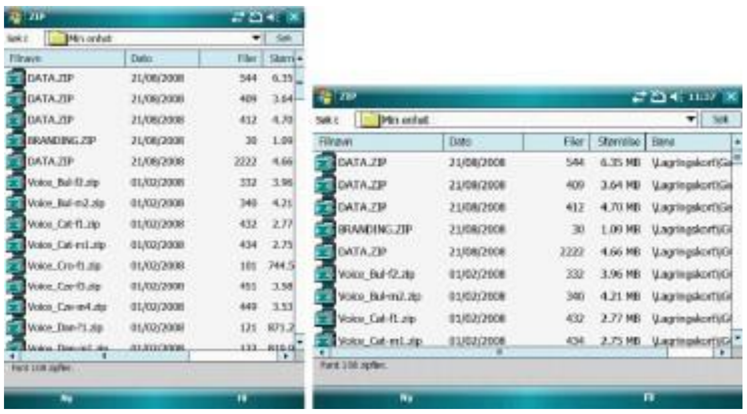
S4 - ARTICLE: DESIGN PATTERNS FOR TOUCHSCREEN-BASED MOBILE DEVICES: USERS ABOVE ALL!

PATTERN: #11
3.1 The Patterns TMDP1.1 The thumb rule For: Smartphones/phablets. Use When: Designing the interface. Placing main elements/options on the screen How: Place main elements within a range of a semicircle with a 2,7 inches' radius from the right-middle side of the screen. Why: The average length of a human thumb is 2,7 inches. Considering that statistically, most of users hold the phone with their right hand and use their right thumb to interact with the device, main elements should be placed within user's reach. Figure 1 shows difference between an Apple iPhone 3GS and a Samsung Galaxy Note II device. Writing with one hand in the Samsung device is almost impossible. Associated Heuristic(s): TMD12.   **Figure 1. The thumb rule: (left) Apple Iphone 3GS. (right) Samsung Galaxy Note II.** TMDP1.2 The thumb rule #2. TMDP2 Performance and feedback TMDP3 Explicit user control TMDP4 Recognizable icons TMDP5 Clean form fields TMDP6 Shape of buttons

S5 - ARTICLE: DESIGN PATTERNS FOR USER INTERFACE FOR MOBILE APPLICATIONS

PATTERN: #12

3.1.2. Design pattern: Change the screen orientation
3.1.2.1. Use when
If this is an option on the platform, it will by default reduce the need for horizontal (and increase the need for vertical) scrolling, as illustrated in Fig. 1. An important choice if this solution is used is whether the user should be given the opportunity to switch between landscape and portrait, or if only landscape should be available. The first choice imposes a number of new problems. The latter choice is easier to realize, but offers less flexibility for the user, and may reduce which devices/versions of the operating system that may be used to run the application.



Filnavn	Dato	Filer	Størr
DATA.ZIP	21/08/2008	544	6.35
DATA.ZIP	21/08/2008	409	3.64
DATA.ZIP	21/08/2008	412	4.70
BRANDING.ZIP	21/08/2008	30	1.09
DATA.ZIP	21/08/2008	2222	4.66
Voice_Bul-f2.zip	01/02/2008	332	3.96
Voice_Bul-m2.zip	01/02/2008	340	4.21
Voice_Cat-f1.zip	01/02/2008	432	2.77
Voice_Cat-m1.zip	01/02/2008	434	2.75
Voice_Cro-f1.zip	01/02/2008	101	744.5
Voice_Cze-f3.zip	01/02/2008	455	3.58
Voice_Cze-m4.zip	01/02/2008	449	3.53
Voice_Dan-f1.zip	01/02/2008	121	871.2

Filnavn	Dato	Filer	Størrelse	Bane
DATA.ZIP	21/08/2008	544	6.35 MB	\Lagringskort\Ga
DATA.ZIP	21/08/2008	409	3.64 MB	\Lagringskort\Ga
DATA.ZIP	21/08/2008	412	4.70 MB	\Lagringskort\Ga
BRANDING.ZIP	21/08/2008	30	1.09 MB	\Lagringskort\G
DATA.ZIP	21/08/2008	2222	4.66 MB	\Lagringskort\G
Voice_Bul-f2.zip	01/02/2008	332	3.96 MB	\Lagringskort\G
Voice_Bul-m2.zip	01/02/2008	340	4.21 MB	\Lagringskort\G
Voice_Cat-f1.zip	01/02/2008	432	2.77 MB	\Lagringskort\G
Voice_Cat-m1.zip	01/02/2008	434	2.75 MB	\Lagringskort\G

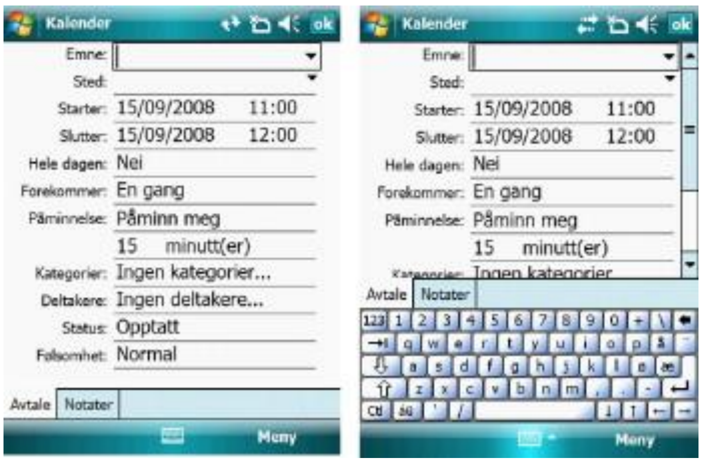
Fig. 1. Changing screen orientation to reduce horizontal scrolling.

3.1.2.2. How
Provide a version of the UI in landscape format, alone or in addition to a version in portrait format.

3.1.2.3. Why
Horizontal scrolling should be avoided in all UIs, but is probably worse on mobile devices than on larger displays, partly because the amount of context information is larger when the screen is larger. Also, on a larger screen, it is usually possible to make the window larger to decrease the need for horizontal scrolling. And even worse, because the screen is smaller, the need for horizontal scrolling occurs more often.

PATTERN: #13

3.2.2. Design pattern: Add or adjust scroll bars
An obvious and simple solution to this problem is to add or adjust scroll bars when the keyboard appears, as illustrated in Fig. 2. The other solutions presented below are solutions where the need for adding scroll bars are removed or reduced.


Kalender

Emne:
Sted:
Starter: 15/09/2008 11:00
Slutter: 15/09/2008 12:00
Hele dagen: Nei
Forekommer: En gang
Påminnelse: Påminn meg
15 minutt(er)
Kategorier: Ingen kategorier...
Deltakere: Ingen deltakere...
Status: Opptatt
Følsomhet: Normal

Kalender

Emne:
Sted:
Starter: 15/09/2008 11:00
Slutter: 15/09/2008 12:00
Hele dagen: Nei
Forekommer: En gang
Påminnelse: Påminn meg
15 minutt(er)
Kategorier: Ingen kategorier

Fig. 2. Adding scroll bar when keyboard is shown.

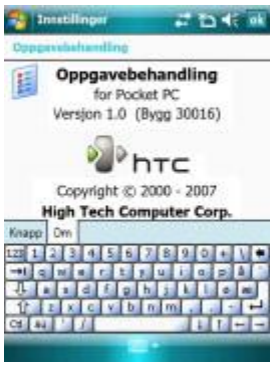
3.2.2.1. Use when
This solutions should be used when a simple and inexpensive solution is sought, or when none of the other patterns are useful.

3.2.2.2. How
Provide two sizes of the view for the dialog.

3.2.2.3. Why
The solution is simple and inexpensive, yet easy to understand.

PATTERN: #14

3.2.3. Design pattern: Let the keyboard cover part of the UI
How “bad” this solution depends on what is placed on the part of the screen that will be covered by the keyboard, as illustrated in Fig. 3.


Innstillinger

Oppgavebehandling

Oppgavebehandling
for Pocket PC
Versjon 1.0 (Bygg 30016)

Copyright © 2000 - 2007
High Tech Computer Corp.

Fig. 3. Keyboard covers part of UI.

3.2.3.1. Use when
If this part is occupied by output fields, the solution may work fine as long as the keyboard is removed when not needed. If this part of the screen contains important input fields or tab folders the solution is useless.

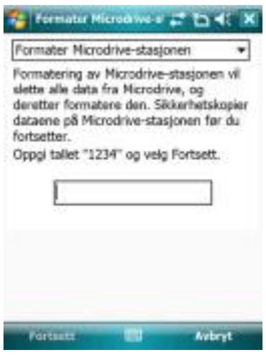
3.2.3.2. How
This solution is in essence “doing nothing”.

3.2.3.3. Why
The solution is simple and inexpensive, though not always very user friendly.

PATTERN: #15

3.2.4. Design pattern: Only use the part of the screen that will not be covered by the keyboard
In practice, what this solution does is to reduce the size of the part of the screen that may be exploited.

3.2.4.1. Use when
This solution may be OK for dialog boxes as illustrated in Fig. 4, but is seldom practical for normal windows.


Formater Microdrive-stasjonen

Formatering av Microdrive-stasjonen vil slette alle data fra Microdrive, og deretter formatere den. Sikkerhetskopier dataene på Microdrive-stasjonen før du fortsetter.
Oppgi tallet "1234" og velg Fortsett.

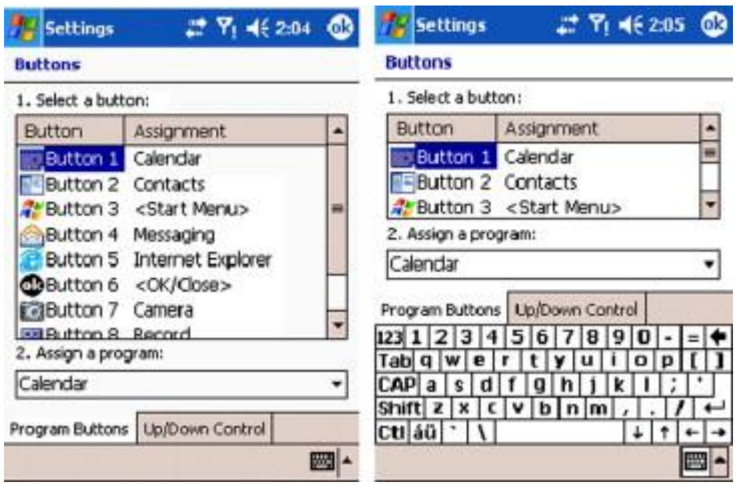
Fig. 4. Dialog box which leaves room for keyboard.

3.2.4.2. How
Restrict the amount of information in the dialog.

3.2.4.3. Why
The solution is simple and inexpensive.

PATTERN: #16

3.2.5. Design pattern: Use one large UI control as a buffer
By this we mean that when the keyboard is added, one of the controls is reduced vertically to be just as much smaller as the size of the keyboard, as illustrated with the list box control in Fig. 5.


Settings 2:04
Buttons
1. Select a button:

Button	Assignment
Button 1	Calendar
Button 2	Contacts
Button 3	<Start Menu>
Button 4	Messaging
Button 5	Internet Explorer
Button 6	<OK/Close>
Button 7	Camera
Button 8	Record

2. Assign a program: Calendar

Fig. 5. List box control that shrinks when keyboard is added.

3.2.5.1. Use when
The solution is relevant when the UI contains one or more controls that may be used as a buffer.

3.2.5.2. How
General controls that may be used for this are primarily list boxes and multi line text boxes.

3.2.5.3. Why
The solution is simple and inexpensive, yet it usually does not confuse the user.

PATTERN: #17

3.2.6. Design pattern: Keyboard as part of layout
Instead of using a built-in software keyboard that the application have to adjust to, it is also possible to have an application specific keyboard that is designed to be part of the layout, as illustrated in Fig. 6.

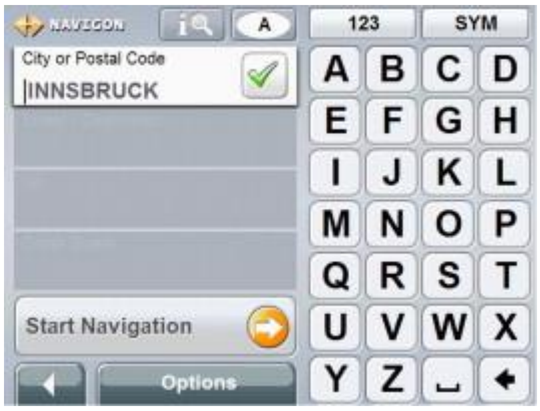


Fig. 6. Keyboard as part of layout of an application.

3.2.6.1. Use when
The solution is most appropriate in mass market products where the extra costs for

designing application specific keyboards will pay off, or when such solutions are supported by the OS (like on the iPhone platform).

3.2.6.2. How
An application specific keyboard must be developed.

3.2.6.3. Why
The solution may provide both very efficient, and user – as well as finger–friendly UIs.

PATTERN: #18

3.3.2. Design pattern: Auto complete
This is a mechanism that tries to guess what the user is about to write and suggests this by filling in the suggested text ahead of the writing of the user, as illustrated in Fig. 7.

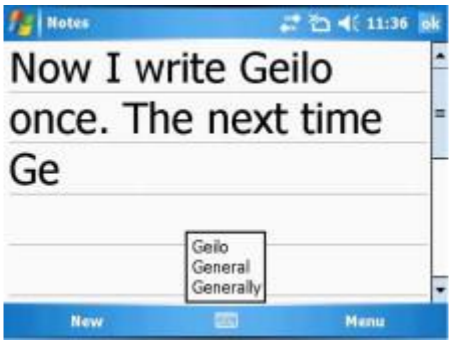

Now I write Geilo once. The next time Ge

Fig. 7. Auto-complete in a notes application.

3.3.2.1. Use when
The solution is relevant when there are some patterns in what the user writes that are repeated over time.

3.3.2.2. How
On the Windows Mobile platform an adaptive auto complete mechanism is included in all the generic input mechanisms. Specialized applications specific auto complete mechanisms in certain field are usually more efficient. This is common when writing an URL in most web browsers and when writing names in an email client. Common for such solutions is that they use the history of values used earlier to suggest the new ones.

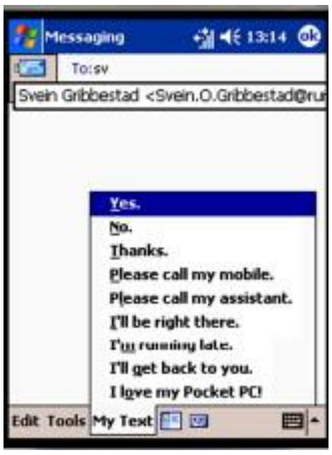
3.3.2.3. Why
The solution reduces the amount of repetitive typing.

PATTERN: #19

3.3.3. Design pattern: Predefined values
By this we mean having a list of all (or the most common) texts to enter in a field.

3.3.3.1. Use when
The solution is relevant when there is a small set of words or phrases that are used more often than others.

3.3.3.2. How
The list of values may be accessed from a menu or from a combo box, as illustrated in Fig. 8. The values in the list may also be dynamic based on user behaviour.



Svein Gribbestad <Svein.O.Gribbestad@ru

Yes.
No.
Thanks.
Please call my mobile.
Please call my assistant.
I'll be right there.
I'm running late.
I'll get back to you.
I love my Pocket PC!

Fig. 8. Using predefined answer alternatives in a messaging application.

3.3.3.3. Why
The solution reduces the need for typing commonly used words and phrases.

PATTERN: #20

3.3.4. Design pattern: Alternative input mechanisms
By this we mean using UI controls that are operated directly on the screen as an alternative to keyboard, as illustrated in Fig. 9.

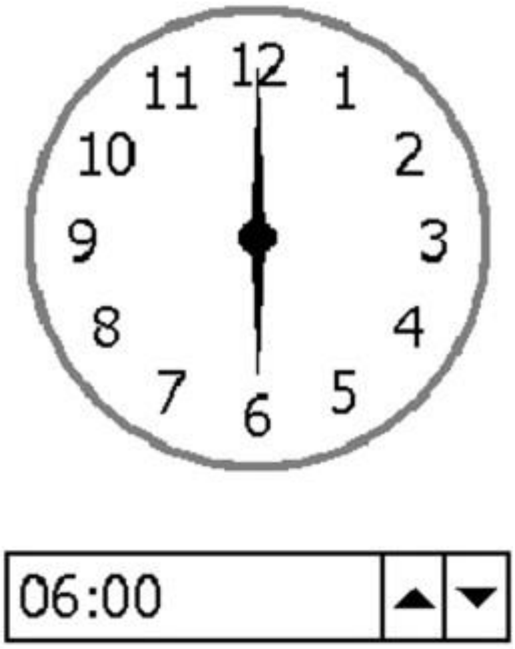


Fig. 9. Using clock and spinners for adjusting the time.

3.3.4.1. Use when
Most of the relevant mechanisms require that there is some sort of restrictions on the domain of the attributes that should be entered through the mechanism.

3.3.4.2. How
In addition to radio buttons, combo and check boxes, spinners, sliders, and menus are the most common controls for this.

3.3.4.3. Why
Direct manipulation is usually more efficient and easier to perform than typing.

PATTERN: #21

3.3.5. Design pattern: Specialized input mechanisms
By this we mean using (a combination of) existing controls in a new way to implement a creative solution.

3.3.5.1. Use when
The solution is appropriate in most situations, specially when other patterns are not relevant.

3.3.5.2. How
An example of this approach is the mechanism used in an application for service technicians, where the user may write common fault description in a natural language like syntax by choosing from a set of drop down list with commonly used nouns, verbs and preposition expressions.

3.3.5.3. Why
Having a restricted number of values in each dropdown list still facilitates entering a

very large number of possible sentences in a simple way.

PATTERN: #22

3.4.2. Design pattern: Finger friendly menu choices
There are design patterns for finger friendly interaction for a number of interaction mechanisms, like finger friendly lists, menus, buttons, keyboards, tab folders etc. Here we include finger friendly menu choices as an example of such design patterns.

3.4.2.1. Use when
The solution is appropriate when the user wants or is required to operate an application using finger interaction.

3.4.2.2. How
An alternative to standard menus or buttons that are always visible is to provide menu choices in a small popup panel at the bottom of the screen, as illustrated in Fig. 10, showing how this is done in an iPhone application. Similar solutions are applied on HTCs TouchFlo 3D user interface on Windows Mobile.

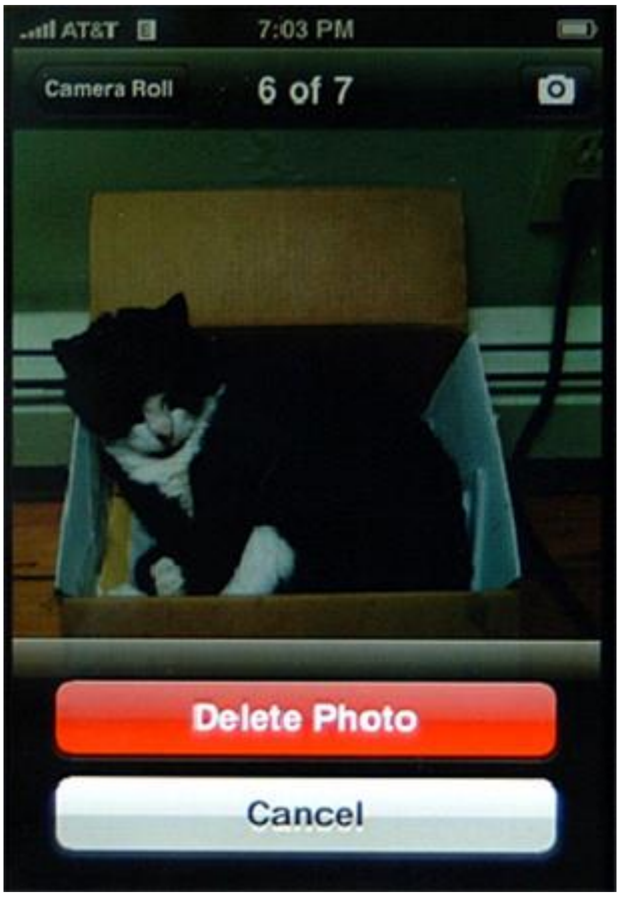


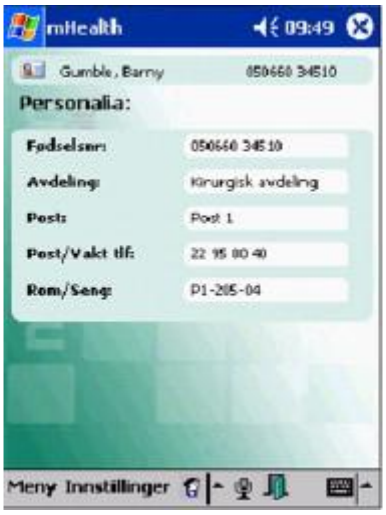
Fig. 10. Finger friendly menu in photo application on iPhone.

3.4.2.3. Why
Menu items, both used as part of pull-up menus and context menus are difficult to operate using fingers. The solution also uses less screen space than buttons that

are always visible.

PATTERN: #23

3.5.2. Design pattern: Brand the standard
By this we mean adding branding elements to the platform standard instead for building the elements that make up a brand from scratch, as illustrated in Fig. 11.



Gumble, Barny 050660 34510

Personalia:

Fodselsnr:	050660 34510
Avdeling:	Kirurgisk avdeling
Post:	Post 1
Post/Vakt tlf:	22 95 00 40
Rom/Seng:	P1-205-04

Fig. 11. Adding branding elements within a standard UI.

3.5.2.1. Use when
The solution is appropriate when both branding and compliance to standards are needed.

3.5.2.2. How
This should be done using subtle means, like changing background colours or adding a pattern or an abstract image as part of controls and/or backgrounds. Also, using a specific font may be a good branding mechanism. The main problem with this solution is that the branding may be difficult to recognize. On the other hand, implementing it does not need to be too costly.

3.5.2.3. Why
The solution combines having controls that are close to platform standards with branding.

PATTERN: #24

3.5.3. Design pattern: Branding the controls
By this we mean generalizing the principle to also cover branding that is further from the standards, as illustrated in Fig. 12.

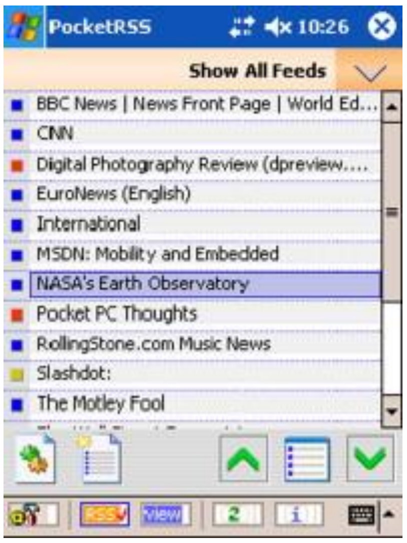

PocketRSS
10:26
Show All Feeds
BBC News | News Front Page | World Ed...
CNN
Digital Photography Review (dpreview....
EuroNews (English)
International
MSDN: Mobility and Embedded
NASA's Earth Observatory
Pocket PC Thoughts
RollingStone.com Music News
Slashdot:
The Motley Fool

Fig. 12. Compact RSS client with branded controls.

3.5.3.1. Use when

The solution is appropriate when branding is more important than compliance to standards.

3.5.3.2. How

Specialized controls need to be developed.

3.5.3.3. Why

This will usually take up less screen space than adding additional purely visual elements (like icons and advanced borders) as the main branding means. Doing more "radical" branding of UI controls may be quite expensive to implement. Using purely visual elements as branding means is less expensive to implement.

PATTERN: #25

3.6.2. Design pattern: Inform the user about what is happening

Informing the user about what is happing (in addition to indicating progress).

3.6.2.1. Use when

The solution is appropriate when this level of information is required or easily obtained.

3.6.2.2. How

This may be done as a scrolling text that the user can browse back in, just a small list showing the latest events, or as single text changing as events happen, as illustrated in Fig. 13. Independent of how the information is shown, it should be presented in a way that is comprehensible by the user – i.e. related to user concepts and user tasks.

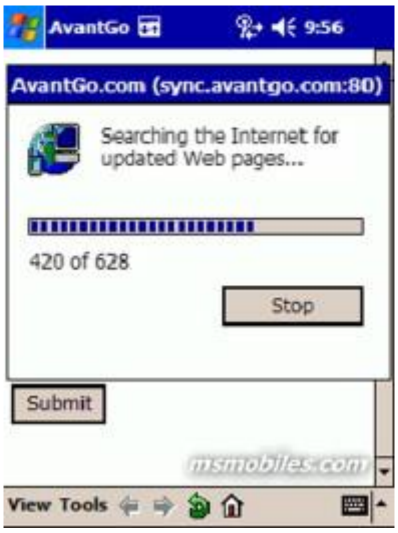

AvantGo
9:56
AvantGo.com (sync.avantgo.com:80)
Searching the Internet for updated Web pages...
420 of 628
Stop
Submit
msmobiles.com
View Tools

Fig. 13. Informing the user what is happening.

3.6.2.3. Why

Providing information about level of progress as well as what is happening will make the user more patient.

S6 - ARTICLE: FROM REQUIREMENT TO DESIGN PATTERNS FOR UBIQUITOUS COMPUTING APPLICATIONS

PATTERN: #26

Pattern name
Control of Autonomous Adaptation

Problem
Autonomous adaptations can result in usability problems. The goal of the pattern is to prevent the feeling of loss of control. Users may sense a loss of control if the behavior of an application is not comprehensible or if the behavior disturbs the current interaction with the application. The pattern helps to create understandable autonomous adaption and prevents the feeling of loss of control.

Forces and Context
Informational self determination: To support the user's self determination in case of autonomous adaptation, the ultimate decision-making authority has to remain with the user - otherwise the system can adapt to unintended and irreversible states.
Transparency: The autonomous adaptation is a black-box concept to the user. If the user does not receive information about next adaptation steps nor the possibility to govern automatically executed actions, he will experience loss of control and a missing overview on the different states and steps.

Solution
The user should be enabled to keep control of autonomous adaptations. This prevents the feeling of loss of control. Two cases have to be distinguished:
1) The user is currently interacting with the application. In this case, the application should notify the user about the upcoming adaptation and enable the user to determine if the application should adapt. The user should have a choice to accept, decline or delay the adaptation.
2) The user is currently not interacting with the application. This means that the adaptation can be performed. However, the application needs to provide the user an option to revert the adaptation. Adaptations with substantial effects on the system should be recorded in a history. Such a change may be the switching off of a surveillance system or of a ringtone. The adaptation design needs to be tailored to the application domain, development platform, and target user group. The cooperation with a usability engineer and/or a trust engineer is recommended.

Consequences
The pattern is influenced by and influences the user interface design of the application. The adaptation notifications need to be integrated into the user interface design.

PATTERN: #27

Pattern name
Emergency Button

Problem
This pattern should be applied if the application collects and uses personal data. It enables the user to halt collection and use of his personal data in a simple manner at any time.

Forces and Context
Informational self determination: The appliance of this pattern supports the user's right to informational self determination by disabling any use or gathering of personal data by the application. It enables the user to maintain control of his/her own data.
Trust: By providing a mechanism to the user to disable the collection and use of personal data, the user's acceptance and trust into the application can increase. This holds especially true in that situations where the user wants to be invisible to the application.

Solution
The implementation and the user interface design of an emergency button depend on the application domain and development platform. The button should be easily accessible at all times. It is important to give feedback to the user after activating the button. After pressing the button, the system stops immediately collecting and using personal data. Herein, all data from which other personal data can be inferred is included. If pressing the button impairs application functionalities, the application highlights these functions to provide visual feedback.

Consequences
When pressing the button, all functionalities, which require personal data, need to be deactivated to prevent errors at runtime. The Emergency Button Pattern can be combined with the Enable/Disable Functions Pattern which addresses similar concerns.

S7 - ARTICLE: HCI DESIGN PATTERNS FOR PDA RUNNING SPACE STRUCTURED APPLICATIONS

PATTERN: #28

4.1 Orientation Patterns
This category introduces HCI patterns to help users to get oriented into a physical space. These patterns improve virtual/physical synchronization of space in order to locate users within the space. They cope with the issues described in point 1 of SSA characteristics. Patterns belonging to this category are the following: 1. You are here (aka Address): A user tries to identify any space somehow. Usually, public spaces are identified by names; so they should be supplied to the user. This pattern is widely used on Web.
2. Multi-Layer Map: Sometimes users need to know their physical position within a space. Physical spaces are structured as a hierarchy and user position can be determined by user space position on each level.
3. Signs: This pattern helps users to get oriented when they spend a long a time into a space and get lost in there. So, a sign is used to synchronize virtual and physical space 4. Door at back: This pattern helps users to get oriented when a space transition occurs. A space transition happens when a users moves virtually and physically from one space to another; for instance form a room to another. As an example of this category we present Door at Back.

Door at Back
1. Synopsis: Spaces are graphically represented by maps. Large buildings have several rooms. As all rooms of a building do not fit on screen at once, each one is represented by a different screen, producing space transitions when a user moves from one space to another.
2. Context: Users pass through different rooms while visiting buildings. When users move from one space to another, a transition on screen occurs.
3. Forces: This interface transition leads to user disorientation between physical and virtual space.
4. Solution: Virtual space orientation is usually represented by a map. This map should be automatically oriented according to the door used by the user is at the bottom of screen. The door should be clearly marked, as an arrow, pointing the same direction the user; as seen on Fig. 1.
5. Consequences: User gets oriented on space by recalling at first sight the room he had when he entered first time.
6. Schematic Description:

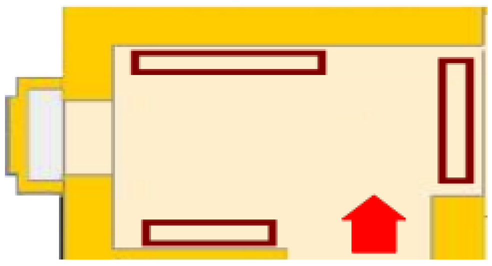


Fig. 1. Sample of "Door at back" pattern

Fig. 1. Sample of “Door at back” pattern
7. Related Patterns: It can be used jointly with Address. Map orientation may be combined with layout changes depending on map shape (Layout patterns).
Based on W3C Common Sense Suggestions for Developing Multimodal User Interfaces principles [18] this pattern focuses on:
• Satisfying Real-world constraints taking into account physical suggestions and environmental suggestions (physical space orientation is treated in this pattern).
• Communicating clearly, concisely, and consistently with users (an arrow represents user entrance direction).
Making users comfortable by easing user’s short term memory (the arrow help users to get “back to the basics” - the moment he / she entered the room -).

We can relate this pattern with Tidwell’s HCI patterns [13] [14]. So, the following sublanguages may apply to this pattern definition: Go Back One step and Go Back to a Safe Place (arrow may be used to go to a safe place to orientate user); Bookmark
(entrance is automatically as a safe place) and; Remembered State (at the time user entered into the room).
From Van Welie et al. [16] point of view, we can relate this pattern with feedback (user gets oriented based on a previous known position) and visibility (User guidance) problem. So, it improves Learnability and Memorability.

PATTERN: #29

4.2 Layout Patterns
Layout patterns were introduced to organize SSA. Screen resolution on mobile devices are restricted, information to be displayed is increased due to virtual / physical space relationship and objects extra information. Point 2 of SSA characteristics can be designed using the following patterns in this category:
• Landscape: This pattern proposes to use PDA in Landscape direction.
• Vertical-Horizontal Layout: Modify the application layout according to the information to be displayed.
• Layout Transition: It shows layout change transition.
As an example of this category we present Vertical-Horizontal Layout.
Vertical-Horizontal Layout
1. Synopsis: Information to be displayed on a portable devices screen should be optimized because screen space.
2. Context: Usually, there are two types of information to be displayed: main information (information that fulfil screen objective) and secondary (additional information to perform other operations).
3. Forces: Main information shape and size vary. For instance, maps, photos and videos may be displayed in portrait or landscape.
4. Solution: To optimize screen visualization for main information, screen layout is changed to fit main information the best way as possible. Secondary information is displayed “around” main information to have it available.
5. Consequences: Primary data information is optimized to fit screen and secondary information is displayed on available space.
6. Schematic Description:

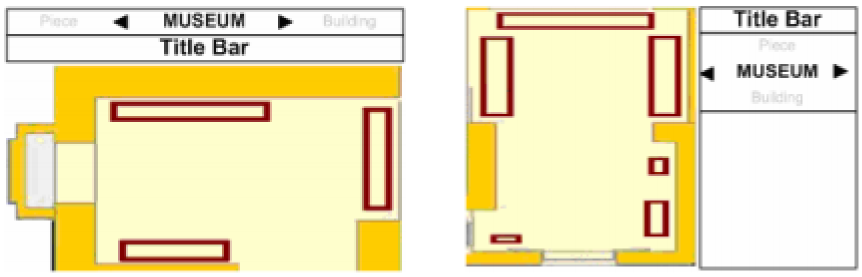


Fig. 2. Sample of "Vertical-Horizontal Layout" pattern

Fig. 2. Sample of "Vertical-Horizontal Layout" pattern
7. Related Patterns: This pattern is close related to Landscape and Layout Transition.
This pattern satisfies the following principles of W3C Common Sense Suggestions for Developing Multimodal User Interfaces [18]:
• Communicate clearly, concisely, and consistently with users by switching presentation modes when information is not easily presented in the current mode. Screen layout adapts interface to main information. It keeps interface as simple as possible, changing control layout instead of controls themselves.
• Make users comfortable by reducing learning gap of a new user interface.
Relationship to Tidwell's HCI patterns [13] [14] sublanguages are: Disabled Irrelevant things (although secondary items are not disabled, they are not treated in the same level of relevance as main information) and Good Defaults (information default layout changes according to main information to be displayed).
If we analyze this pattern from [16] perspective, the problems it affords are related to Conceptual Model and Natural Mapping (user knows exactly how to perform operations, if the user had previous experience with the interface - before layout transformation -). We try to cope with Learnability and Memoability usability issues.

PATTERN: #30

4.3 Guide Patterns
Design patterns on this category are used to model routes and paths that users may follow to guide users through any physical space based on user preferences. So, point
3 and slightly 1 of SSA characteristics are boarded here. Patterns belonging to this category are the following:
• Free Will Navigation: This pattern provides a method to access spaces at any level through the application using cursor keys only.
• Routes: Routes pattern provides routes to focus a visit on user preferences.
As an example of this category we present Free Will Navigation.
Free Will Navigation (aka Up-Down and Left-Right or No Guide)
1. Synopsis: Virtual space navigation is performed by cursor keys only.
2. Context: Usually, people using SSAs do not have both hands free (carry baggage).
So, people should be able to hold and operate a device with one hand only.
3. Forces: As one of the most important things to be performed by this kind of applications is space navigation, it should be easily performed by one hand and be

learned quickly.
4. Solution: To cope with this navigation problem we propose to control navigation by cursor buttons using: Left – Right keys to navigate across space levels (interlevel). Right button goes one level into selected space (if a piece is selected on a showcase, when right button is pressed, it goes into selected piece). While Left arrow cursor button goes one level up (if a showcase is being shown, when left button is pressed, it goes to the room enclosing this showcase). And Up – down buttons are used to navigate across same level spaces (intra-level). It selects a subspace into the same space. Up and down buttons changes selection to labelled items. Labeling actions representing cursors on screen provides action feedback to user. See fig 3.
5. Consequences: User is aware of navigation destination using cursor keys. If proposed control is accepted as a standard on SSAs, learning gap will be minimized. A disadvantage of using labels is the fact that they may obscure map.
6. Schematic Description

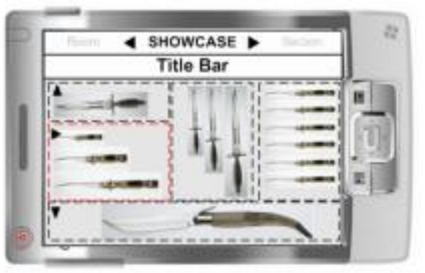


Fig. 3. Sample of "Free Will Navigation" pattern

Fig. 3. Sample of "Free Will Navigation" pattern
7. Related Patterns: Main relationship is established with Landscape Layout pattern because portable devices, as PDAs, can be used with one hand only, if they are in landscape position. It is also related to Right-Left handed users.
On W3C Common Sense Suggestions for Developing Multimodal User Interfaces principles [18] this pattern satisfies the following principles:
• Satisfy real-world constraints by assigning cursor key to most common operation on this kind of application. It also applies physical suggestions by using one hand only instead of both hands.
• Communicate clearly, concisely, and consistently with users by using the same keys through navigation system regarding of space level (keeping interface simple).
According to Tidwell's sublanguages [13] [14] this pattern is related to: Optional Detail On Demand (user access information according to space level); Short description (information about navigation is displayed on screen) and Convenient environment Actions (people usually goes one level up and down only).
Finally, according to Van Welie's [16] perspective, the problems it affords are related to Visibility (user guidance, navigation can be used to guide users across building); Affordance (it uses the space metaphor); Feedback (operations are labelled). And usability issues we try to cope with are Learnability and Memoability.
Note: We propose this pattern as a standard way of navigating across SSAs.

PATTERN: #31

4.4 Accessibility Patterns:Right – Left Handed users
Accessibility category is used to group patterns that can be applied to improve application access to disabled people. Patterns related to Point 4 of SSA characteristics are grouped here.
• Space Audio Perception: A voice tells the user which space has selected
• Right – Left Handed users: It adapts a SSA application designed using the Landscape pattern to be used by right or left handed people.
• Zoom: It provides controls to change font size when users are reading documents.
As an example of this category we present Right-Left handed users.
Right-Left Handed Users
1. Synopsis: This pattern adapts the system to be used by most skilled hand of the user.
2. Context: Usually people do not have the same skills on both hands. So, if an application that should be used with one hand only, it is logical that the hand used to perform operations be the skilled one.
3. Forces: Right – Left handed users
4. Solution: Solution lays on two issues: mirroring screen horizontally and Change cursor control behaviour (Up - Down) (Left - Right).
5. Schematic Description:

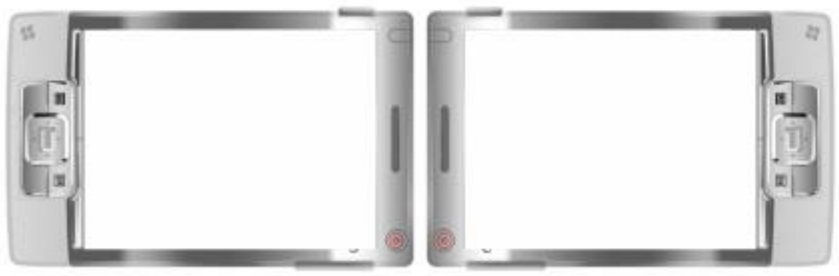


Fig. 4. Sample of "Left-Right handed users" pattern

Fig. 4. Sample of "Left-Right handed users" pattern
6. Related Patterns: On W3C Common Sense Suggestions for Developing Multimodal User Interfaces principles [18] this pattern satisfies the following principles:
• Satisfy real-world constraints by using the easiest mode available on the device to perform each task.
• Communicate clearly, concisely, and consistently with users by making command consistent and organizational suggestions keeps interface simple.
Relating this pattern with Tidwell's [13] [14] we found it is related to Convenient environment actions (actions are adjusted to user's perspective).This pattern improves
flexibility providing explicit control. It also improves learnability and memorability.

S8 - ARTICLE: INTERACTION PATTERNS FOR WINDOWS 8 TABLET APPLICATIONS

PATTERN: #32

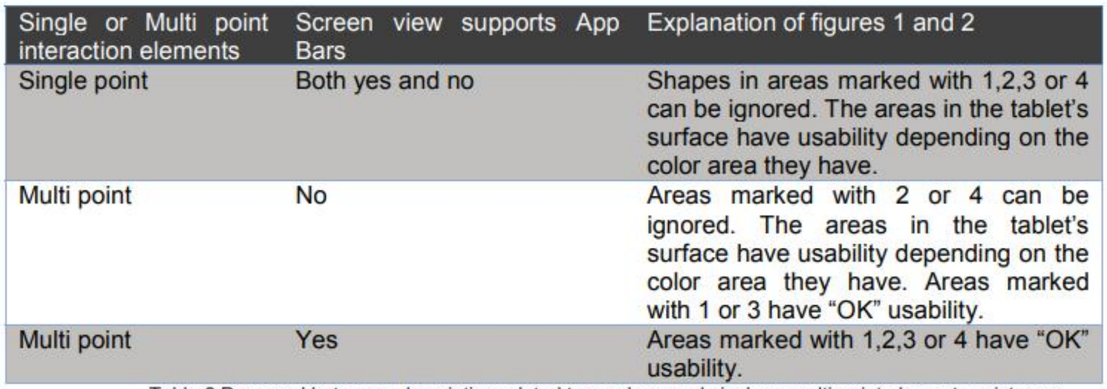
4. PATTERN: HOT AREAS FOR INTERACTION ELEMENTS
4.1 Context
When the user interface designer designs Windows 8 applications that are to be interacted with touch gestures, she must carefully consider where to place the interaction elements (that touch gestures can be applied on). For the sake of brevity, we'll call the areas referred to by "where" as the "hot areas". These user interface elements can be split in two major categories; elements that can be pressed or held (e.g. TAP FOR PRIMARY ACTION), thereafter called "single point interaction elements" (i.e. the interaction involves only one screen point) or that can be dragged or swiped (DRAG, SWIPE), thereafter called "multipoint interaction elements" (i.e. the interaction involves many screen points, such as movement of the finger). Furthermore, the interaction designer has the option to support either landscape or portrait orientation or, of course, both. However, most (if not all) Windows 8 tablet devices' aspect ratio is 16:9 (whereas, for instance iPad's is 4:3) at the time of writing, so they offer much better experience in landscape mode. Consequently, we encountered few apps in the Windows 8 Store that support portrait view.

4.2 Problem statement
When designing a Windows 8 app, the user interface designer has to carefully select the hot areas for single point interaction elements (e.g. buttons, checkboxes) and multi point interaction elements (such as swiping pages in order to read a digital magazine) in the supported app orientations (landscape or portrait,
or both). The problem that arises is where these elements should be placed in order for the app to have the best usability possible.

4.3 Forces
User ergonomics forces
x The users can hold tablets either in landscape or portrait mode
x Interaction elements can either be single point or multi point
x The users usually hold the tablets along the side [24]
x The screen should not obscured during the touch gesture. If this is unavoidable, the obscuration during the course of the gesture should be minimal
x The user feels comfortable when she does not have to move her fingers a lot, in order to perform a gesture
x The placement of the interaction areas should be aligned to the hands' position (i.e. in the way the user is holding the tablet device)
x The interaction elements should be well suited to how a user usually holds a tablet. This, owing to the fact that the user should feel comfortable in moving her fingers from where she's holding the tablet to the interaction area, where these elements are located
Windows 8 standards/requirements regarding edges swipe forces
x The interaction designer should not place interaction areas that can be dragged or swiped close to the edges of the screen, because swipes on all four edges can activate Windows 8 system commands and a user could easily activate them by

mistake
x Whether an app has App Bars is up to the interaction designer
x App Bars can be activated with gestures on top and bottom of the screen
x Interaction areas that can be dragged or swiped should not be placed extremely close to the screen edges, because the user may mistakenly activate “App Bar” (swiping from top or bottom, if the view supports it), navigate to the previously opened app (swiping on the left edge of the screen) or open the “Charms Bar” (swiping on the right edge of the screen)
Tablet orientation implications forces
x Most Windows 8 apps support only landscape display mode. If, however, the application supports
portrait display mode, the interaction areas should move accordingly. In this way, they should
locate themselves towards the user fingers’ position

4.4 Solution
The solution is based on the fact that elements closer to the user’s fingers are easier to interact with. We are proposing three solutions, in relation to whether the application view in question supports App Bars or whether we are discussing single or multi point interaction elements. All three solutions can be illustrated in a special figure, modified from the one found in Windows Interaction Guidelines [25].



Single or Multi point interaction elements	Screen view supports App Bars	Explanation of figures 1 and 2
Single point	Both yes and no	Shapes in areas marked with 1,2,3 or 4 can be ignored. The areas in the tablet’s surface have usability depending on the color area they have.
Multi point	No	Areas marked with 2 or 4 can be ignored. The areas in the tablet’s surface have usability depending on the color area they have. Areas marked with 1 or 3 have “OK” usability.
Multi point	Yes	Areas marked with 1,2,3 or 4 have “OK” usability.

Table 2 Proposed hot areas description related to app bars and single or multi-point elements existence.

Of course, if the view has both single and multi-point interaction elements (which is the most common scenario), the interaction designer can resort to a combination of the proposed solutions.
a) The “OK” area. As the name implies, it is an “OK” hot area to place interaction elements (i.e. not a bad one, but not the best). However, elements are usually far from where the user’s fingers are located and large parts of the screen could be obscured in subsequent attempts to interact with them.
b) The “Better” area. It’s not that bad of a design decision to place interaction elements here. The elements are not that close to where the user is holding the device and some parts of the screen could be obscured
while the user is trying to perform touch gestures.
c) The “Best” area. This is the best hot area for interaction elements. They are pretty close to the user’s fingers, and obscuration upon interaction is extremely limited, if any at all.

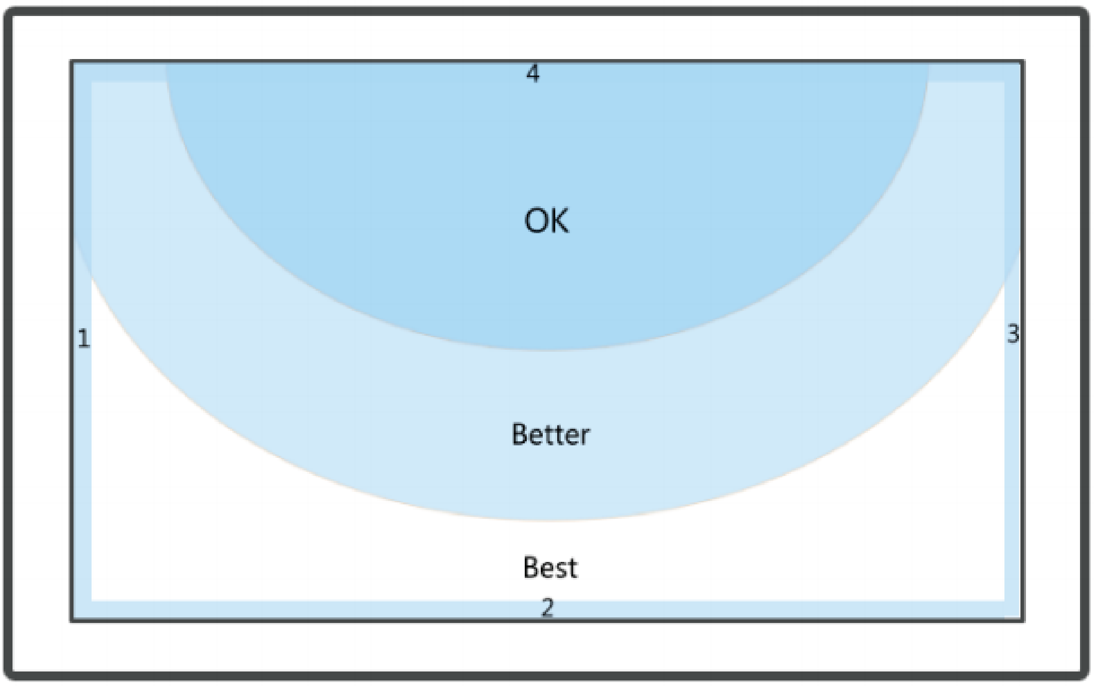


Figure 1 Interaction areas for tablet interfaces in landscape mode

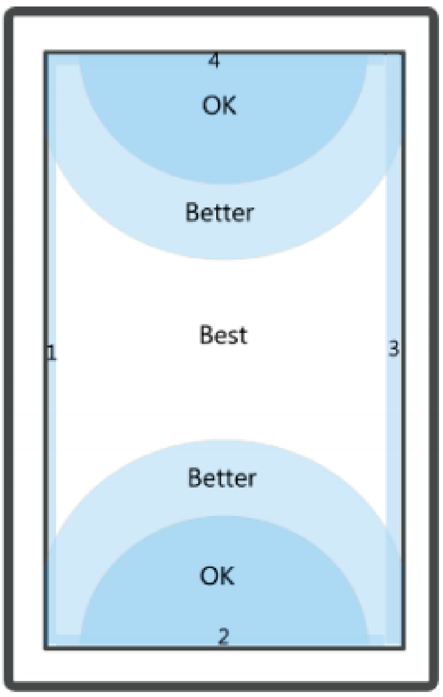


Figure 2 Interaction areas for tablet interfaces in portrait mode

4.5 Rationale of the solution

There is an interesting analysis of how tablets are usually held (using both hands) in Microsoft’s user experience guidelines for Windows 8 [24]. Microsoft recommends placing interaction elements according to a special way, found in Windows 8 Interaction Guidelines [25]. There, one can find the recommendation for app views in landscape orientation and portrait orientation. In both cases, the screen is split into different areas separated by gray color variants, categorized as “Best”, “Better” and “OK”, according to the usability they offer (described in the pattern’s solution section), if an interaction element is placed upon them. However, as already described in the solution, we decided to go one step further than this and take into account whether the app view supports app bars and whether it has single or multi-point interaction elements. Therefore, the solution we propose must be aligned to the above way users are usually holding their Windows 8 tablet devices while taking into consideration Windows 8’s edge swipe gestures. By viewing the above way the user is holding her tablet, we can easily notice that

1. The areas that are more accessible to the user’s fingers (either thumb or index) are close to where she has them, while holding the tablet
2. Parts of the screen can be obscured if the user moves the fingers to an area that is far away from the point she is holding the tablet Windows 8 operating system guidelines propose the exact scheme that we presented about the placement of interaction elements/areas. This is aligned to the before mentioned ways users are holding their tablets and can also work for other tablet operating systems. Generally, this scheme does a pretty good job in describing the ideal placements for single point interaction elements in an app. However, as previously mentioned, in Windows 8 the user can swipe from the edges of the screen to reveal system or app commands, launch a previously opened app or open the “Charms Bar”. So, if the user interaction designer places a multi-point interaction close to the edges of the screen, there is a high risk of the user mistakenly activating the respective system command, instead of performing those gestures on the required element on the screen. Consequently, regarding the placement of multi point interaction elements in Windows 8 we propose a differentiation of this figure, depending on whether the specific app view supports App Bars. This differentiation can be viewed in Figures 1 and 2, where the areas marked with 1, 2, 3 or 4 (which act as an extension to the “OK” area) can be seen in all edges of the screen. Those areas offer “OK” usability for interaction elements placement. The areas marked with 2 or 4 can be ignored, depending on whether the specific app view supports App Bars. If the app view in question does not support App Bars, then the areas marked with 2 or 4 can be ignored and the area below them offers placement usability depending on the color area it originally belonged. If there are App Bars, then areas marked with 1, 2, 3 or 4 offer “OK” usability for interaction elements placement. Consequently, multi point interaction elements should be placed somewhat far away from the edges of the screen that do support swipes (left and right always do whereas top and bottom may or may not), to enhance accuracy of user’s gestures on them. Of course, there can be combinations of the three solutions if the app view has both single point and multi point interaction elements.

Moreover, we propose that the size of the “OK” area close to the edges should be large enough in order to minimize the risk of the user performing the wrong gesture, if an interaction element is placed near it. Microsoft recommends that the minimum touch size for accuracy should be 9x9mm [24], so it is our estimation that this area’s width or height should be at least equal to it. In this way, it would be easier for the user’s fingers to correctly perform touch gestures, either on the multi-point interaction elements or to the edges of the screen.

4.6 Known Uses

First Puzzles: Animal Kingdom

In this application, the draggable puzzle pieces are scattered towards the bottom of the screen and are to be dragged all over the screen in order to complete the puzzle. Moreover, a single point interaction element ("go back" button) exists on the top left of the screen. We observe that multi point interaction elements are placed towards the bottom edge of the screen, where is the best interaction area. Furthermore, this app has no App Bars, so the extra "OK" gray bars cover only the left and right side of the screen.

Letter & Spelling Fun! for Preschoolers and Early Readers

In this application, interaction elements that can be tapped are placed on the "Best" part of the screen and have some gap between the screen edges.

Tots PlaySchool

In this application, interaction elements that can be dragged are placed on the "Best" part of the screen.

First Words with Phonics

In this application, letters are initially scattered towards "Better" area of the screen and are to be dragged to the bottom of it, in order to complete the required word. The target area for this is on the "Best" part of the screen. This app could be improved by placing the target area on the "Better" or "OK" part of the screen and the letters on the "Best" part. Furthermore, the dolphin image on the left that can be dragged is mostly on the "Better" and "Best" areas. Finally, two single point interaction elements, a "select animal" button and a "next animal" button are located on the top cornes, thus among the "Best" and "Better" area.

Kindergarten8

In this application, there is a color/paint mini-app. In this mini-app, the color selection control is located on the bottom of the screen whereas the tool selection control is located on the right. Plus, there is a "go back" button at the top left of the screen. All these controls are single point interaction elements and are located on "Best" and "Better" parts of the screen, allowing for easy and accurate input by the user.

Animal Sounds

This application is one of the few that we encountered and supports portrait orientation. In this app, the main interaction elements (the three animals) are mostly located on the Best and Better screen areas and they are big enough, allowing for easy interactions. The interaction designer could further optimize the app by moving the "previous" and "play sound" buttons lower, towards the Better area.

PATTERN: #33

5. PATTERN: INTERACTION BETWEEN SNAPPED AND FILLED VIEW

5.1 Context

A Windows 8 app that its user interaction designer wishes to implement interaction functionality between two apps that are running at the same time.

5.2 Problem statement

A user interaction designer wants to design a snapped app in a way that it is able to interact with a filled app. Interaction could be of any sort, including (but not limited to) copying and pasting data and opening a filled app from a snapped one.

5.3 Forces

x The interaction will take place either to or from a snapped app

x Interaction is limited to what Windows 8 operating system currently offers

(copy/paste and hyperlink tapping)
x At the time of writing, Windows 8 Software Development Kit allows only the launch of the default browser on filled view, when the calling app is on snapped view and the launch is implemented via a hyperlink-like control
x At the time of writing, the only possible way to transfer data between an app running in snapped view and an app running in filled view is by using the classic mechanism of copy/paste
x The user may want to copy some data from the one app and paste it into the other
x The user can only copy and paste textual data between snapped and filled apps

5.4 Solution
In the current version of the Windows 8 Software Development Kit, two options are available for interaction between a snapped and a filled app. The first one is to copy text from the snapped app to the filled app (or vice versa) and the second is to launch the default browser from a snapped app. These options are thoroughly explored in the following sections.
Copying and pasting from a snapped to a filled app (and vice versa)
The user interaction designer will have to implement copy/paste friendly controls on respective apps. For instance, a TextBox (control accepting text input) is a user interface control that accepts paste, whereas a TextBlock (control presenting text) is a control that can have part of or its entire content copied via tap and hold gesture [TAP AND HOLD]. In this way, user can select and copy part of the text presented and paste it in the other app.
Launching the default browser from a snapped app
The user interaction designer can use hyperlink-like controls in the snapped app to allow the launch of the default browser by tapping on this control. These hyperlink controls have to contain a link that points at a web page (URL). Upon user tapping [TAP], the default browser will launch in filled view, browsing the specific web page pointed at by the hyperlink control's content.

5.5 Rationale of the solution
As described, this solution is twofold related to the kind of the desired interaction.
a) In regards to the copy/paste interaction, the only necessary implementation is the use of
copy/paste-friendly user interface controls (such as textboxes for text input and blocks of text
(TextBlock controls) for text presentation). User can also copy text from a web page, displayed on
a web browser.
b) In regards to the app launch interaction, the only allowed app that can be launched is the default
browser. Hence, the user interaction designer has to use a hyperlink-like control that points to an
internet address (URL). Upon tapping, the default browser will automatically launch.

PATTERN: #34

6. PATTERN: SNAPPED VIEW TRANSFORMED FROM HORIZONTAL TO VERTICAL

6.1 Context
Windows 8 app (or app view) that contains independent or grouped pieces and they are displayed in a horizontally panning view (horizontal list). This may include (but not be limited to) the “master” section of master-detail apps. Examples are the articles list section of a news reading app, the customers’ list section of a customer relationship app, games list page of an app that contains mini-games etc.

6.2 Problem statement
User interaction designer needs to design the snapped view of the app (or of one of the app’s views) in order for the app to be as functional as possible.

6.3 Forces
x App (or app view) contains pieces (either independent or grouped) laid out on a horizontal list that can be dragged either left or right
x Snapped view must allow scrolling
x Snapped view should display as many items as possible
x In snapped view, height is greater than width
x Horizontal scrolling is unacceptable when the height of the app is greater than its width x Snapped app must remain active

6.4 Solution
Shift data lists from horizontal to vertical orientation.

6.5 Rationale of the solution
Many Windows 8 apps are designed in order to “slide” from left to right. This is evident in the operating system’s Start Screen, on the preinstalled apps that come with the operating system and last but not least, in the default project templates in the Visual Studio 2012 development environment (the prominent development tool for Windows 8 apps). Horizontal orientation and scrolling is certainly not functional in the case an app goes to snapped view because of the limited width this state supports. However, the height of the snapped view is exactly the height of the device’s screen (768 pixels minimum). Consequently, for “slide left to right” kind of apps, the user interface designer could alter the layout so as it appears in a vertical list with minimal information. As an example, the Kindergarten8 app displays a vertical list with rich information when running on full-screen and gets its layout transformed on snapped view. Plus, the user interface designer has opted to minimize the presented information, displaying only images respective of the mini-game they represent. Windows 8 Software Development Kit has user interface controls for data display that allow for orientation conversion from horizontal to vertical.

6.6 Known uses
Kindergarten8 app uses a horizontal list to display image shortcuts for the various mini games it contains. When the application goes into snapped view, the list becomes vertical and the images become much smaller, thus allowing for more selections.
Moreover, CNN app displays a horizontal list with data (each list item can contain another list) on full screen. When the app goes on snapped view, every item in every list is grouped in a single vertical list.

PATTERN: #35
7. PATTERN: UNWILLING TO BECOME SNAPPED 7.1 Context Windows 8 apps that look good only when displayed in full screen mode (or at least in filled mode), such as games. These apps usually have an absolute necessity of screen space, in order to be displayed correctly and offer a great experience. This may include (but not be limited to) classic games apps, apps that require as much screen estate as possible in order to operate correctly (such as drawing apps) etc. 7.2 Problem statement User interaction designer has to design the snapped mode for apps that look good and work efficiently only on full screen. 7.3 Forces x These apps do not offer the anticipated experience if redesigned for snapped view (i.e. scaled down graphics, smaller items so difficult to use touch gestures) x These apps require lots of screen space (in a specified width and height manner) in order to function correctly x Since these apps cannot work in snapped mode, their state should be maintained and the app should be paused x User must be informed that the app cannot work properly in snapped view 7.4 Solution Allow snapped view and show a “played only in full screen” like message. The app’s state should be paused. The app could display relevant messages along with the “go to full screen” one, such as current game level, global/local scoreboards, achievements, any notifications etc. Most games should stick to this solution. 7.5 Rationale of the solution Some Windows 8 apps (especially games) have been designed from the ground up in order to take advantage of entire screen estate by spreading into its width. These apps could have their usability degrade severely if, for example, they got scaled down in order to fit into the snapped view. For these cases, we highly recommend that the app, when into snapped view, shows a message that it should move to full screen (or filled view) in order to continue its execution. This message should obscure the entire app screen. Moreover, app could provide some useful data (such as a user’s current score, a list of high scores etc.). 7.6 Known uses First Puzzles: Animal Kingdom app would have its usability degraded if it scaled down its graphics in order to work in snapped view. Thus, the user interface designer has opted for a full size image that informs the user that the game is not playable in snapped view and she should transform it into full screen view. Additionally, Cut the Rope app would be unusable in a “height much bigger than width” layout (such as snapped view) because of its gameplay engine. Therefore, when it is transformed into snapped view, it informs the user that she needs to switch the app into full screen (or filled view, of course) in order to continue playing. The app is paused during the time it is on snapped view.

PATTERN: #36

8. PATTERN: SHRINKING AN APP WHEN SNAPPED

8.1 Context

Apps running in Windows 8 that can afford to have their content and interactive elements scaled down and/or rearranged and/or hidden when in snapped view. Examples are the “details” section of “master details” apps, the article details section of a news reading app, the customer view/edit section of a customer relationship management app, board games etc.

8.2 Problem statement

User interaction designer needs to design the snapped view for apps or app pages that can have their content scaled, layout arranged and/or interaction elements hidden when in snapped view in order to fit as many elements as possible plus minimize the impact on user experience.

8.3 Forces

x Snapped app or app view must maintain state and remain active
x App has some elements that could have their size decreased while on snapped view
x There are interaction elements on the app that could be hidden while on snapped view
x User interaction designer can add additional interaction elements on the snapped app to enhance usability

8.4 Solution

Scale and rearrange content while possibly hiding not absolutely necessary user interface elements. This is suitable for views that can afford to have their content scaled, change position or collapsed (hidden), provided the usability does not degrade dramatically.

8.5 Rationale of the solution

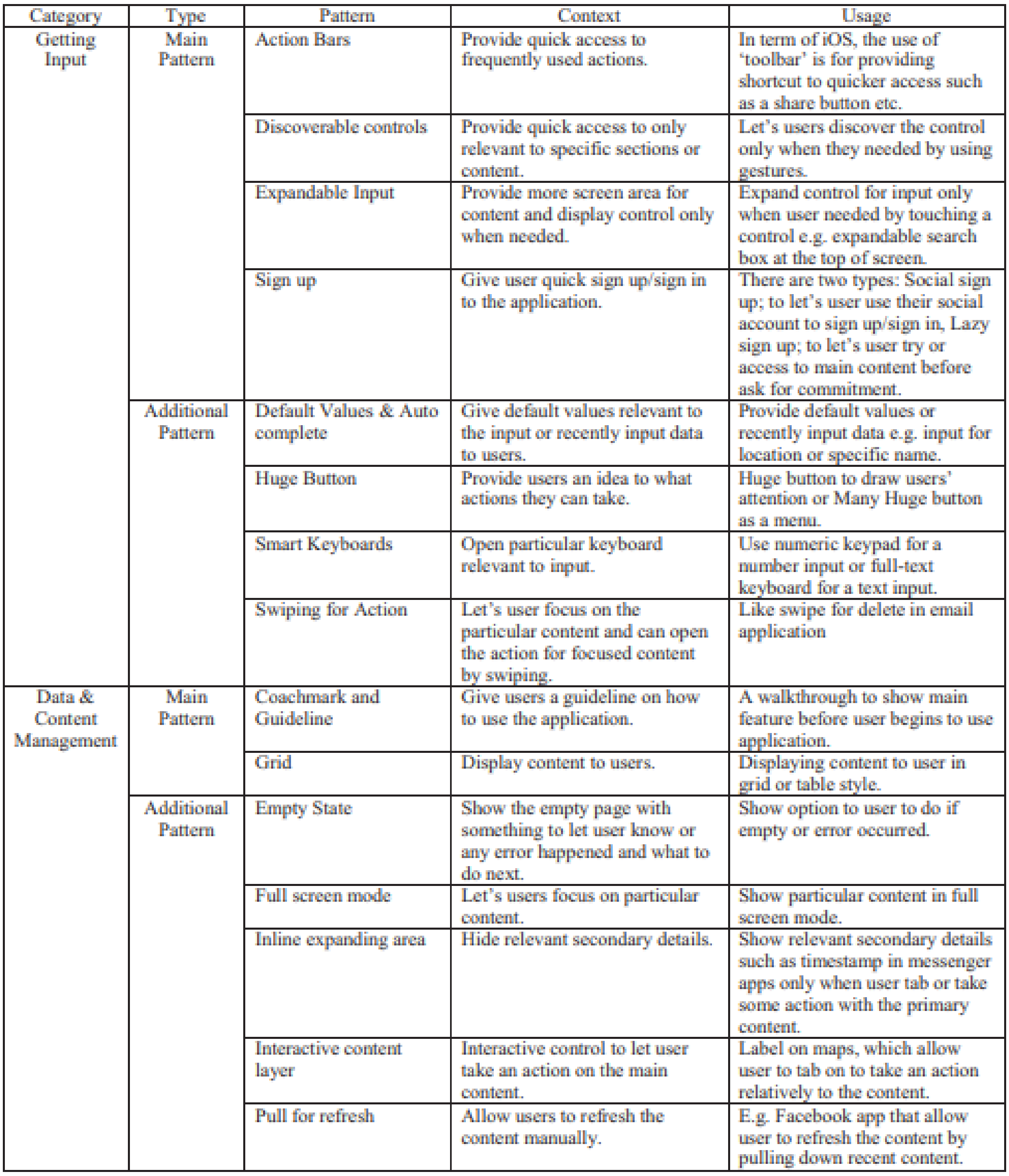
This pattern attempts to transfer the whole experience to the snapped view, by resizing (scale to fit) user interface elements and transforming the layout by moving interaction areas in the app. Plus, it may hide content or interactive elements otherwise visible in the full screen version of the app. This solution might be suitable for apps such as board games, where the board will be scaled down in a way that the game is still playable. Plus, graphics should be scaled down no more than necessary, in order to be absolutely visible and (if possible) usable. There should be some user acceptance testing in this view, though. If the result is less than satisfying, then user interaction designer should revert to the other patterns related to snapped view.

8.6 Known uses

“News” app initially has the two game boards displayed on horizontal orientation. When it is transformed into snapped view, the game boards are scaled down and displayed in vertical orientation, which does not degrade the app’s usability.
When a user uses the MetroTube app in full screen and watches a video, she is presented with the video being watched, comments and lists with related videos and

this video’s YouTube channel. If the user snaps the app, the video becomes smaller, some user interface controls get a bit smaller whereas the lists with related videos and channel videos are completely hidden.

S9 - ARTICLE: METHOD FOR MOBILE USER INTERFACE DESIGN PATTERNS CREATION FOR IOS PLATFORM



Category	Type	Pattern	Context	Usage
Getting Input	Main Pattern	Action Bars	Provide quick access to frequently used actions.	In term of iOS, the use of 'toolbar' is for providing shortcut to quicker access such as a share button etc.
		Discoverable controls	Provide quick access to only relevant to specific sections or content.	Let's users discover the control only when they needed by using gestures.
		Expandable Input	Provide more screen area for content and display control only when needed.	Expand control for input only when user needed by touching a control e.g. expandable search box at the top of screen.
		Sign up	Give user quick sign up/sign in to the application.	There are two types: Social sign up; to let's user use their social account to sign up/sign in, Lazy sign up; to let's user try or access to main content before ask for commitment.
	Additional Pattern	Default Values & Auto complete	Give default values relevant to the input or recently input data to users.	Provide default values or recently input data e.g. input for location or specific name.
		Huge Button	Provide users an idea to what actions they can take.	Huge button to draw users' attention or Many Huge button as a menu.
		Smart Keyboards	Open particular keyboard relevant to input.	Use numeric keypad for a number input or full-text keyboard for a text input.
		Swiping for Action	Let's user focus on the particular content and can open the action for focused content by swiping.	Like swipe for delete in email application
Data & Content Management	Main Pattern	Coachmark and Guideline	Give users a guideline on how to use the application.	A walkthrough to show main feature before user begins to use application.
		Grid	Display content to users.	Displaying content to user in grid or table style.
	Additional Pattern	Empty State	Show the empty page with something to let user know or any error happened and what to do next.	Show option to user to do if empty or error occurred.
		Full screen mode	Let's users focus on particular content.	Show particular content in full screen mode.
		Inline expanding area	Hide relevant secondary details.	Show relevant secondary details such as timestamp in messenger apps only when user tab or take some action with the primary content.
		Interactive content layer	Interactive control to let user take an action on the main content.	Label on maps, which allow user to tab on to take an action relatively to the content.
		Pull for refresh	Allow users to refresh the content manually.	E.g. Facebook app that allow user to refresh the content by pulling down recent content.

PATTERN: #37 À #51 - https://doi.org/10.1109/JCSSE.2015.7219787

S10 - ARTICLE: MUSICAL INTERACTION PATTERNS: COMMUNICATING COMPUTER MUSIC KNOWLEDGE IN A MULTIDISCIPLINARY PROJECT

PATTERN: #52

3.1 Pattern: Natural Interaction / Natural Behavior

Solution: Imitate real-world, natural interaction.

Description: This pattern corresponds to musical interaction which imitates real interaction with a sound-producing object, or with an acoustic musical instrument. Thus, all musical gestures that we might regard as “natural” may be explored herein: striking, scrubbing, shaking, plucking, bowing, blowing, etc. One advantage of designing interaction as a reproduction of natural musical gesture is that it will generally include a passive haptic (tactile) feedback, similar to the one we have when interacting with real sound-producing objects. This “primary” feedback (linked to the secondary feedback of hearing the resulting sound) [10] may be important for a “fine-tuned” control of the musical interaction – that “intimate” control suggested by Wessel and Wright [21], which allows the performer to achieve a sonic result that is closer to the intended, and that also facilitates the development of performance technique. For example, a rhythm performance activity may be implemented using the touchscreen of a PDA, where sounds are triggered when it is gently striked with the stylus, like on a real drum. Or, one may implement a shaker-like instrument by using accelerometer sensors of some mobile device, and musically interacting with this instrument by shaking the device. But exploring “naturality” in musical interaction design refers not only to designing user input as natural musical gestures, but also to simulating, through user interface (UI) output, any natural behavior which is expected from real-life objects when they produce sound (i.e., behavior that is linked to sound producing phenomena). This can be implemented either through representations on the graphical interface (GUI), or through an adequate mapping, applied to the physical UI, between possible gestures and their naturally expected sonic results. In our “Drum!” prototype, the user “strikes” the PDA screen and hears a percussion sound, what would be naturally expected. In our “Bouncing Balls” prototype, little “balls” are constantly moving horizontally on the device’s screen, making sound every time they “bounce” on “obstacles” (a barrier or the sides of the screen). Notice that this natural behavior has one drawback: it will generally limit musical interaction to the “one-gesture-to-one-acoustic-result” rule of nature (except for some very particular cases).

Motivation for use: To make musical interaction more “intuitive”, that is, to take advantage of what Jef Raskin [14] prefers to call the user’s “familiarity” with the interaction. This is justified by the hypothesis that, by designing interaction in a form which “resembles or is identical to something the user has already learned” [14], its learning curve is reduced, what is a usability attribute (learnability).

PATTERN: #53

3.2 Pattern: Event Sequencing
Solution: Allow the user to access the timeline of the musical piece, and to “schedule”

musical events in this timeline, making it possible for him/her to arrange a whole set of events at once.

Description: In this pattern, users interact with music by editing sequences of musical events. This can be applied to any interpretation of these – individual notes, whole samples, modification parameters, in short, any kind of “musical material”. Now, it is important to state that, although our interaction patterns aim primarily at musical control, this does not imply a necessary coupling with performance activities. Neither is this pattern, of event sequencing, useful solely for composition. They are all higher level abstractions which may be applied creatively to any type of musical activity, and should be much more useful if regarded this way. In this sense, it may even be preferable to classify them not under “musical control”, but as “music manipulation patterns”. Actually, event sequencing is a good example of this flexibility, since it can be observed both in CODES (asynchronous, compositional tool; see Figure 1) [9] and, for instance, in Yamaha’s Tenori-On portable instrument (real-time performance) [11], where the sequences execution is looped, but they can be edited (and so played) in real-time (see Figure 2). This last, synchronous use was also added later to our Drum! prototype (described in the next section), the first prototype in which we combined patterns.

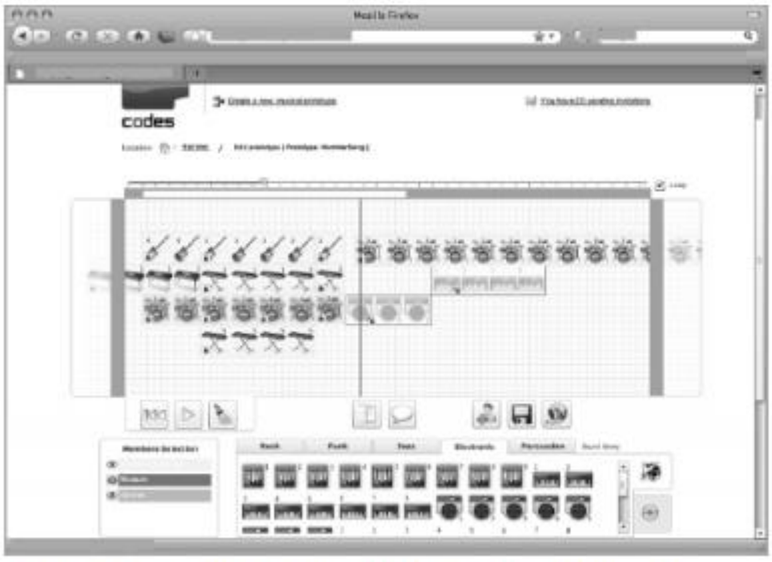


Figure 1. Asynchronous Event Sequencing in CODES, a music composition tool [9].

Figure 1. Asynchronous Event Sequencing in CODES, a music composition tool [9].
From designing Drum! and Bouncing Balls we conclude that, by combining interaction patterns, it is possible to create richer interaction.

Motivation for use: Usually, to extend interaction possibilities – increase interaction flexibility – by explicitly allowing, and facilitating, epistemic actions as a complement to pragmatic actions on the system [17, 8].

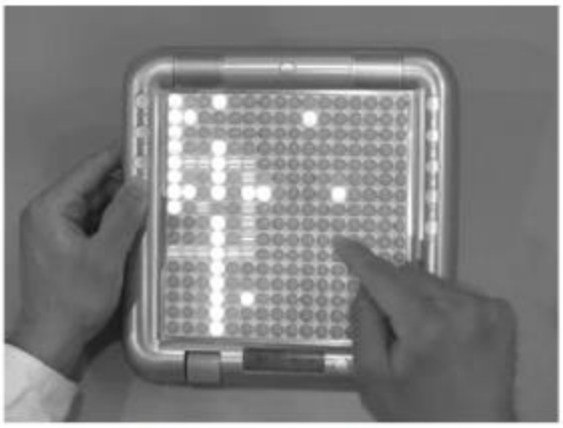
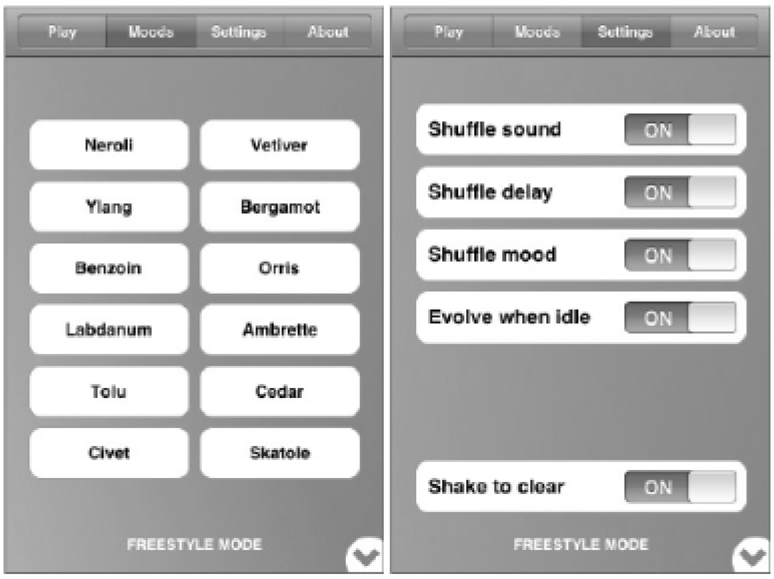


Figure 2. Event Sequencing in Tenori-On, during a looped real-time performance [11].

PATTERN: #54

3.3 Pattern: Process Control
Solution: Free the user from event-by-event music manipulation, by allowing him/her to control a process which, in turn, generates the actual musical events or musical material.

Description: This interaction pattern corresponds to the control of parameters from a generative musical algorithm. It solves that important problem in mobile music, which is the repurposing of non-specific devices: how can we "play" a cell phone, with its very limited keyboard, not ergonomically suited to be played like a piano keyboard? The Process Control solution suggests a mapping from the (limited) interaction features of mobile devices, not to musical events, but to a small set of musical process parameters. This way, we free the user from manipulating music event-after-event, him/her needing only to start the process – which generates a continuous stream of musical events, usually through generative grammars or algorithms – and then to manipulate its parameters. One possible analogy is with the conductor of an orchestra: he doesn't play the actual notes, but he controls the orchestra. For the mapping, we find it useful to follow suggestions given by Wessel and Wright [21] when describing their metaphor of a "space of musical processes". Put simple, the idea is that mapping parameters into a key matrix (a keyboard) or a touch-sensitive surface does not need to follow much previous planning: an "intuitive" arrangement of controls in the "parametric space", done by a musician or computer music expert, is enough to yield a satisfactory mapping. Although it is possible to apply the "parametric navigation" metaphor from these authors, as we did in our Arpeggiator prototype, we believe that it is also possible to use other metaphors they suggest, for the control of interactive musical processes: drag & drop, scrubbing, dipping, and catch & throw [21]. Another useful heuristic for designs using this pattern is that of allowing the user him/herself to configure which process parameters does he/she wants to manipulate. An example of applying the parametric control of a musical process is the Bloom application for iPhones [12]. This software was developed in collaboration with musician Brian Eno, and allows one to introduce events, through the touch screen, into a generative process. Then, the user may alter the "path" of the process, changing parameters while the music is playing (see Figure 3).



Shuffle sound ON
Shuffle delay ON
Shuffle mood ON
Evolve when idle ON
Shake to clear ON

Figure 3. Parameter configuration for a generative musical process in Bloom, an iPhone application [12].

Motivation for use: To avoid the paradigm of event-by-event music manipulation, allowing for more complex musical results through simpler interaction with a process, which in turn deals automatically with the details of generating the definitive musical material. This pattern implements HCI principles like "simplicity" and "process automation". Since it simplifies interaction, it is also a sound answer to design restrictions imposed by the limitation in interaction features, which is typical of standard mobile devices.

PATTERN: #55

3.4 Pattern: Sound Mixing
Solution: Music manipulation through real-time control of the parallel execution of longer musical structures (musical material) – i.e. by mixing musical material.

Description: This pattern consists in selecting and triggering multiple sounds, so that they may play simultaneously. If a sound is triggered while another is still playing, they are mixed and play together, hence the name of the pattern. Here, music is made as a layered composition of sounds, but by real-time triggering of events, so we may see sound mixing as the real-time version of event sequencing. The musical events in this case are sounds or musical structures, and may be of any duration. If they are long (one may even be an entire music sample, triggered just once, or a small but looped sample), we are again avoiding, with this pattern, the traditional note-by-note paradigm of musical control, which is very difficult to implement in conventional mobile devices. But remember: this can be applied not only to music performance. Our "mixDroid" prototype, for example, is a compositional tool where the user records quick, small performances, and combines those into a complete composition. Sound triggering may be also not necessarily instantaneous. One way to instantiate this pattern is by emulating a real sound mixer (see Figure 4). Sounds will be already playing, but all muted initially. The user will then combine these sounds by manipulating their intensities, maybe gradually. In this form, interaction by sound mixing can be noticed as the method of choice in modern popular electronic music.

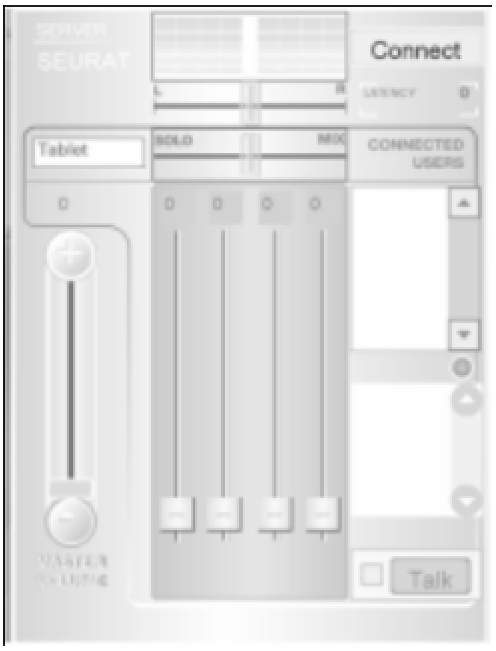


Figure 4. GUI from Tanaka's system for PDAs [18], based on volume-controlled mixing of network transmitted music streams.

Motivation for use: As in Process Control, to avoid the paradigm of event-by-event music manipulation, that is very difficult to implement in conventional mobile devices. Each musical gesture from the user will result in a longer, more complex acoustic result, and the user will be focused in combining these “layers” of sounding musical material.

S11 - ARTICLE: PATTERNS FOR INTERACTIVE LINE CHARTS ON MOBILE DEVICES?

PATTERN: #56

2.2 DETAILS SLIDER
Context: Line charts are displayed on a mobile device and the user needs a method of interaction with individual data points to view additional details on demand.

Problem: Line charts have multiple data points and displaying all these points on a mobile screen makes the screen appear cluttered. Also, the conventional way of demanding data on visualizations has been through hover or right click, which is not possible on mobile screens due to the limited interaction options of a touch screen.

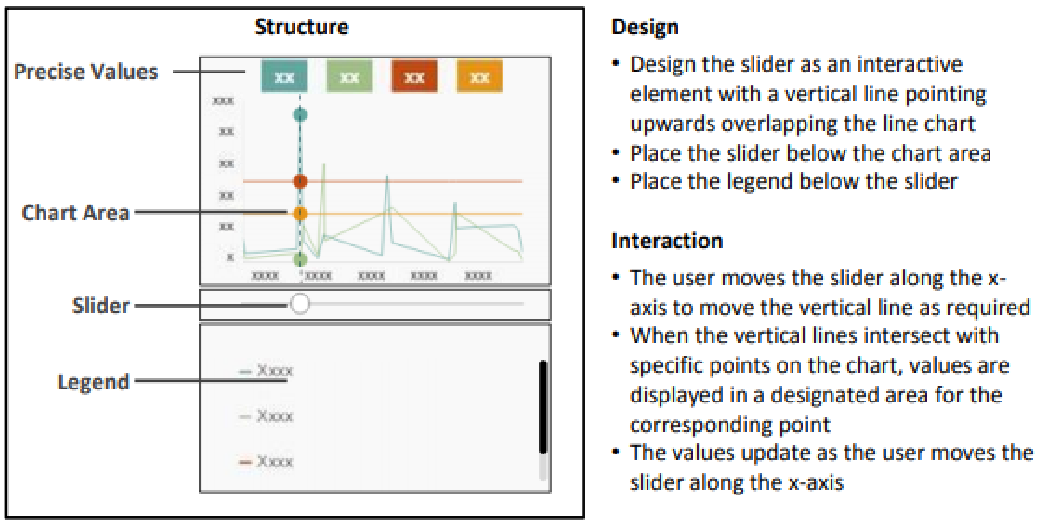
Forces: This problem is tough to solve due to the following reasons:
-Smaller Screen Sizes: As mobile devices have much smaller screens, high amounts of data cannot be displayed to the users. Additionally there’s always a need to keep the number of elements as low as possible to keep things clutter free.
-Lack of equivalents to conventional desktop interactions: Mobiles offer a variety of interactive options like tap, double tap, long press, gestures etc. While these have gained traction, there are still no well-defined guidelines for scenarios like data on demand. Desktop users would hover on a point in the chart, however on mobiles it is a challenge to design the right interaction for this task.
-Variation in user’s finger size: Unlike at the desktop where the pointer has a fixed size, users use their fingers to point to elements on screen. Different users may have different finger sizes. This is an additional aspect to consider when designing for mobile devices.
-Possibility of finger obstructing the view: Similarly, unlike an onscreen pointer, the user's finger could obstruct the view. Additionally as users would be a mix of right and left handed people, it is uncertain what area of the screen would be obstructed.
-Variation in hand orientation: As users can be right or left handed, they would be able to access different regions of the screen easily. Interactive elements would hence have to be placed such that both right and left handed users can access it with ease.

Solution: Add a slider below the line chart with a vertical line attached such that it points upwards over the chart. Let users move the slider and select specific chart points using the vertical line. Display precise data about the selected points in a dedicated area.

Additionally, the positioning of the slider is such that sufficient space can be allotted for the various possible finger sizes of different users thus solving the challenge of variable finger sizes. Also, as the slider is placed below the chart, both right as well as left handed users would be obstructing the area below the chart, leaving the chart itself completely visible to the user. Thus the DETAILS SLIDER helps make selection as well as consumption of data in the chart easy and straightforward. Figure-1 illustrates the pattern concept in greater detail.

Design

• Design the slider as an interactive element with a vertical line pointing upwards overlapping the line chart
• Place the slider below the chart area
• Place the legend below the slider
Interaction
• The user moves the slider along the xaxis to move the vertical line as required
• When the vertical lines intersect with specific points on the chart, values are displayed in a designated area for the corresponding point
• The values update as the user moves the slider along the x-axis



Design

- Design the slider as an interactive element with a vertical line pointing upwards overlapping the line chart
- Place the slider below the chart area
- Place the legend below the slider

Interaction

- The user moves the slider along the x-axis to move the vertical line as required
- When the vertical lines intersect with specific points on the chart, values are displayed in a designated area for the corresponding point
- The values update as the user moves the slider along the x-axis

Figure 1: Details Slider Pattern - Solution Outline

Consequences: This pattern helps make designing and consumption of data easier and intuitive when using charts on mobile devices. Key benefits (+) and liabilities (-) of this pattern are as follows:
+ Information is neatly displayed in spite of smaller screen sizes
+ The user is provided with a useful alternative to hover functionality
+ Space is allotted for variations in user finger sizes
+ Slider design does not obstruct the view of the chart
+ Both right as well as left handed users are able to use the pattern with ease
– The solution adds one more element to the interface which can add clutter in some cases
– The solution can get complex when there are large number of lines on the multi-line chart. However, this can be solved by applying the LEGEND FILTER pattern as described in section 2.3 of this paper.
– User has to put additional effort to point to precise areas if data points occur frequently on a line.
This experience however can be improved by zooming in using the SELECTION BRUSH pattern as described in section 2.4 of this paper. Further fine tuning of the view can be done using the pinch and spread method [10].

Known Uses: This pattern has effectively been applied to the CPU Utilization Chart depicted in Figure-2. The use case of the CPU utilization chart was to display CPU usage data across a timespan of six years. The frequency of data was random, that is, some data points appeared very close to each other (two dates within the same month) while other data points appeared far from each other (two dates in two different years).
Users were asked to point to specific data values using the slider. It was observed that the use of the slider eased these tasks and allowed users to find specific values with lesser time and effort.

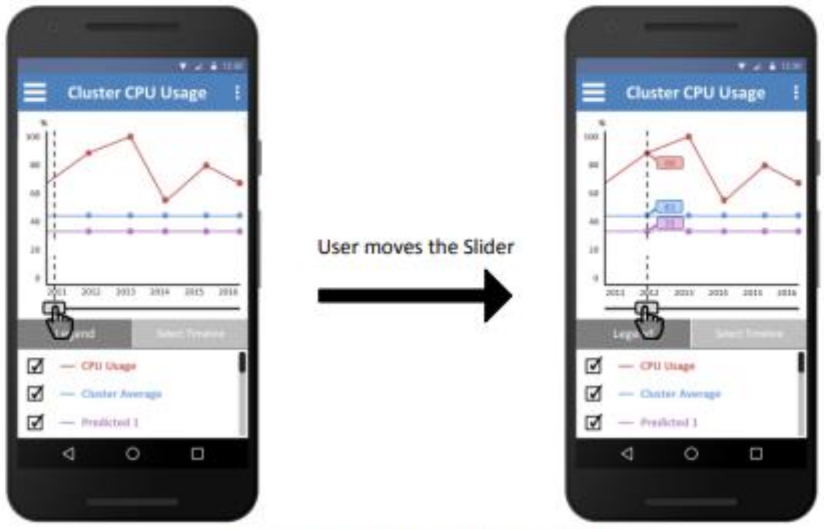


Figure 2: Details Slider Pattern - Illustration

BURDEN

The pattern is also applied in multiple javascript charting libraries like ZingChart[11], Highcharts[12], FusionCharts[15] and amCharts[16].

PATTERN: #57

2.3 LEGEND FILTER

Context: Multiple lines are being displayed in a line chart on a mobile device. In order to reduce overwhelm the user needs a method of interaction to hide or display individual lines on the chart.

Problem: In situations where there are multiple lines on a mobile visualization, the interface might become too cluttered to gain any insights out of it. In such cases, the user would want to manually filter display of a few less important lines to focus on more relevant ones for comparison. However, adding more filtering elements to the small mobile screen can add to the clutter and make it difficult to interact and consume data from the chart.

Forces: This problem is tough to solve due to the following reasons:

• Small Screen Sizes: Adding an additional filter element can be challenging due to the lack of space on the screen. When working with small mobile screens it becomes necessary to keep the number of elements on the screen as low as possible. The interface could get quite cluttered especially when the filter has a large number of options. While a filter itself is an additional element, the number of options on the element (number of lines) would be an additional aspect that could add to the clutter of the interface. This is because in some cases the number of lines could be quite high.

• Limited Interactive Features: Unlike desktops, mobiles have a limited set of interactive options. While there are options like swipe and pinch, this requires the use of additional elements to signify the same. These additional elements can add to the clutter on the screen. This also adds more steps for the user, when performing the filtering task.

Solution: Create an interactive legend with checkboxes or an equivalent interaction that allows users to enable or disable the legend elements. Control visibility of a line in the line chart according to the state of the corresponding legend element.

As the legend would be displaying the same data as a filter element, a single element can be used to reduce repetition of data as shown in Figure-3. Though there is an additional function of filtering that needs to be added, the same is incorporated into an existing element (legend) so as to avoid the need to add an additional element. The design of the legend should also be scalable as there is no limit to how many data points would need to be represented.

This can be done by making the list scrollable and then sorting the list in a logical order for example alphabetical, numerical etc. As most users would only want to enable a few options, users should be allowed to deselect all at a click in cases of high number of options. Users would have to tap on one of the lines being shown on the legend, which in turn will toggle the display of the related line in the multi-line chart.

This method will hence help optimize space utilization and reduce the possibility of clutter on the interface.

Figure-3 outlines the solution of LEGEND FILTER pattern: user interface structure and interaction.

Design

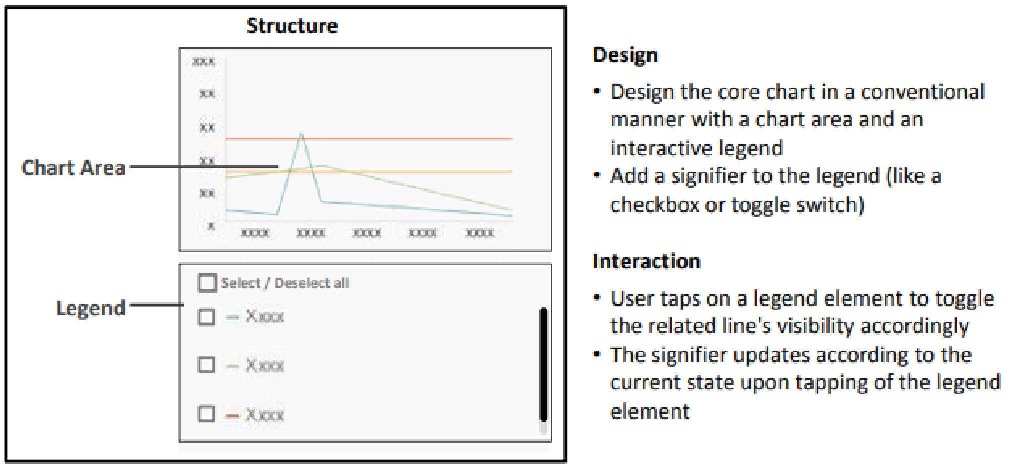
• Design the core chart in a conventional manner with a chart area and an interactive legend

• Add a signifier to the legend (like a checkbox or toggle switch)

Interaction

• User taps on a legend element to toggle the related line's visibility accordingly

• The signifier updates according to the current state upon tapping of the legend element



Design

- Design the core chart in a conventional manner with a chart area and an interactive legend
- Add a signifier to the legend (like a checkbox or toggle switch)

Interaction

- User taps on a legend element to toggle the related line's visibility accordingly
- The signifier updates according to the current state upon tapping of the legend element

Figure 3: Legend Filter Pattern – Solution Outline

Consequences: Key benefits (+) and liabilities (-) of this pattern are as follows:
+ Removes the need to add an additional element as the existing legend element is used optimizing space utilization
+ Eliminates the need to perform additional steps to filter data, as an element that is displayed upfront is reused
– A large number of options lends itself to confusion, particularly if the layout and arrangement of options is not well managed. Hence, sorting and layout needs to be given extra attention keeping in mind user's context to avoid confusion.

Known Uses: In the CPU utilization chart as depicted in Figure-4, along with CPU usage, the system also displays CPU usage predictions and average CPU usage data. Users may need to compare a combination of two or three trend lines (For example, CPU and average, CPU and prediction 1, prediction 1 and prediction 2 and so on). Check boxes provided next to the category names in the LEGEND FILTER can be used to toggle the visibility of respective lines represented in the legend. Thus, it was possible to enable and disable the display of lines allowing users to reveal only the relevant values for comparison.
It was observed, that the existence of an upfront filter option on the legend itself, was very intuitive to follow. Users were able to enable and disable the display of lines with lesser effort and time as it required lesser number of steps to complete.

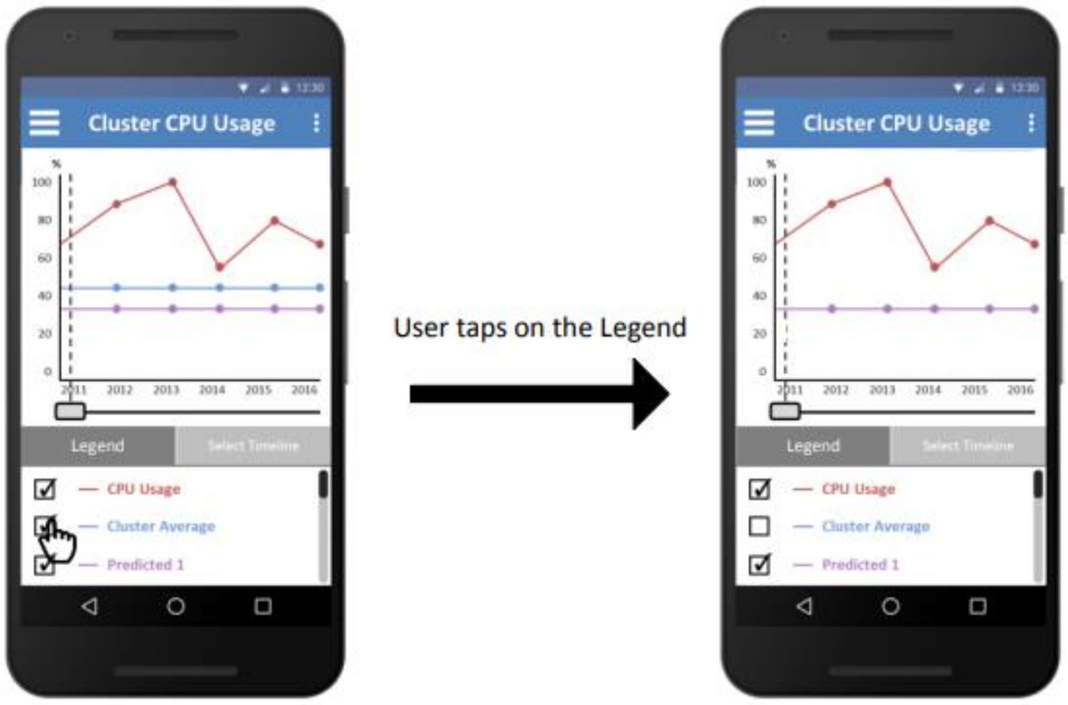


Figure 4: Legend Filter Pattern - Illustration

This pattern is also applied as a feature in various javascript libraries like ZingChart [11], Highcharts[13], FusionCharts[15], plotly [18] and amCharts[17].

PATTERN: #58

2.4 SELECTION BRUSH

Context: A line chart containing data for a large timeframe is displayed on a mobile device. The user wants to investigate a certain section in the timeframe without losing perspective of the entire chart.

Problem: While investigating a certain section in the chart the user may not want to lose perspective of the overall big picture. Enabling this can involve displaying more than one version of the same data simultaneously on the same screen. This however can be challenging as there would be a need to display more elements and interactions on a small screen making the overall visualization cluttered and inconvenient to use.

Forces: This problem is tough to solve due to the following reasons:

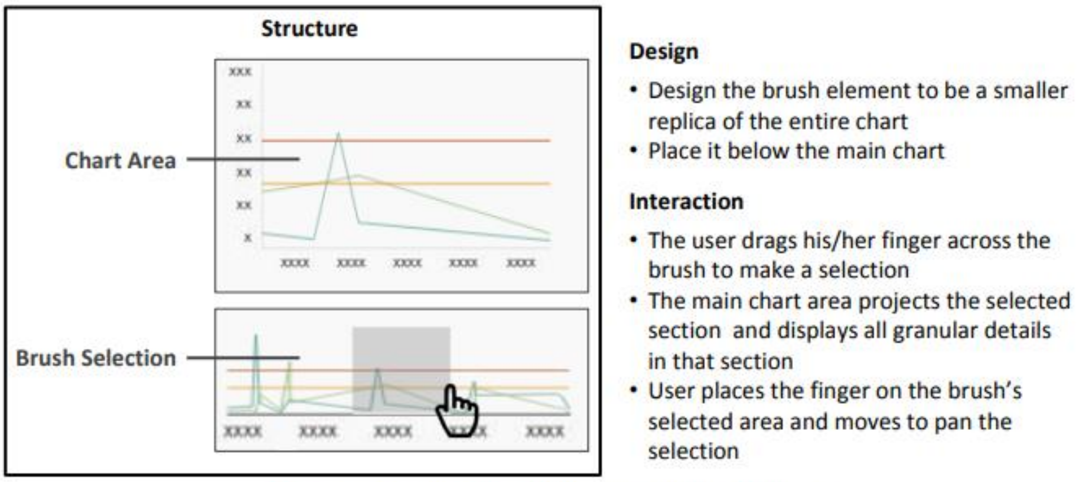
• Small Screen Size: As mobile screens are much smaller than those of a desktop, it becomes challenging to add more elements and details on the same view. The interface can easily get cluttered and confusing for users.

• Need to balance details and overview: Getting granular details and at the same time not losing the high level view are contradictory in nature. This contradictory nature makes even the simplest of charts difficult to design when attempting to meet this need.

• Limited Interactive Features: As mobiles have limited interactive features, interactions can be challenging. Incorporating interactions like swipes, pinch and so on would also need additional elements to signify the same and would inevitably lead to a lot of movement of the interface.

Solution: Add a miniature representation of entire chart below the chart. Allow users to select and pan specific areas by dragging over the miniature representation. Update the main chart to display the selected area in greater detail. As the brush is a miniature version, it does not need a lot of place. The visibility of the entire chart in a miniature form gives the user much more context about the section being inspected. User can understand trends within as well as out of the selection, determine causes for any deviation in chart trends and also understand the current location of the section within the entire chart itself.

Design
• Design the brush element to be a smaller replica of the entire chart
• Place it below the main chart
Interaction
• The user drags his/her finger across the brush to make a selection
• The main chart area projects the selected section and displays all granular details in that section
• User places the finger on the brush's selected area and moves to pan the selection



Design
• Design the brush element to be a smaller replica of the entire chart
• Place it below the main chart

Interaction
• The user drags his/her finger across the brush to make a selection
• The main chart area projects the selected section and displays all granular details in that section
• User places the finger on the brush's selected area and moves to pan the selection

Figure 5: Selection Brush Pattern – Solution Outline

Consequences: Key benefits (+) and liabilities (-) of this pattern are as follows:
+ Combines two functionalities, that is, displaying the entire chart and allowing the user to select relevant sections in the same element, thus reducing clutter and optimizing space utilization
+ As the brush is displayed upfront, users can directly interact with it, and also use it to compare the detailed version with the big picture.
– Making very precise section selections can still be challenging, especially when the range of the data is huge and, in comparison, the selection is smaller. The process can however be made a little easier using the pinch and zoom method to refine the selection further.

Known Uses: As shown in Figure-6, the chart displays CPU usage data for six years with additional measures such as predictions and average data. In this chart, users may need to investigate specific sections across the timespan in detail. For example: a particular month in a specific year, an entire year, a custom period etc. Users can drag across the brush to select specific areas. The main chart area would update to display details specific to the selected timeframe. It was observed that users navigated different sections of the timeframe easily. They were also able to understand specific trends better with the context provided by the miniature version of the entire chart.

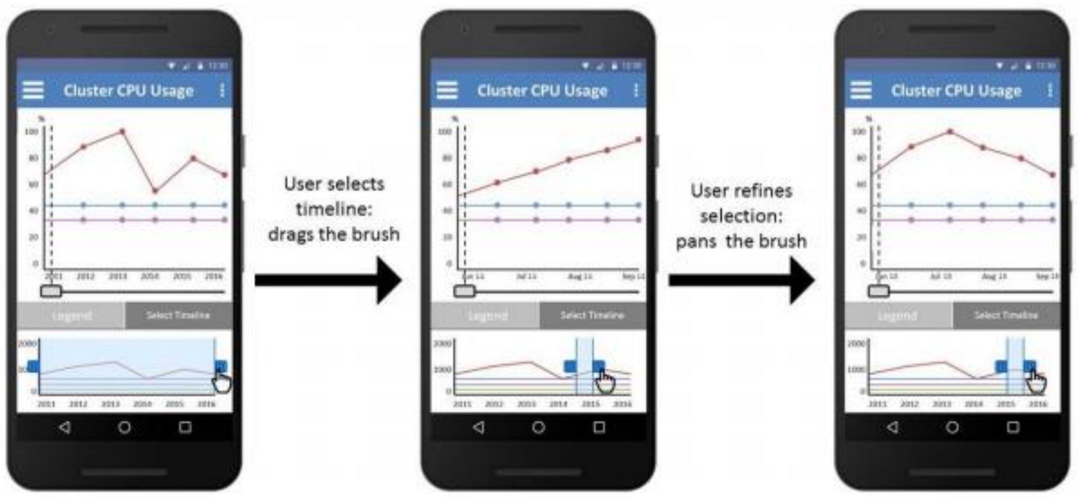

Cluster CPU Usage
User selects timeline: drags the brush
Cluster CPU Usage
User refines selection: pans the brush
Cluster CPU Usage
Legend
Select Timeline

Figure 6: Selection Brush Pattern – Illustration

Various JavaScript libraries have also applied this pattern as a feature. Some of these libraries include ReactD3 [23], ZingChart [11], Highcharts[14] and amCharts[16].

PATTERN: #59

2.5 RESPONSIVE DATA GRID

Context: On mobile screens, charts are used for visualization and a user wants to see detailed data of the chart upfront.

Problem: In cases where the user wants to quickly compare multiple data points or there are minor differences in values of a line chart, users may want to see all data upfront. However, displaying all data on the visualization can lead to an almost useless visualization as it would be tough to consume the data in such form. In such scenarios, users generally expect a table along with the visualization so the data can be navigated and consumed as the user deems fit. However, displaying a conventional table along with the visualization would be challenging due to the limitations of mobile phones.

Forces: This problem is tough to solve due to the following reasons:

• Small Screen Size: As mobile devices have much smaller screens, high amounts of data cannot be displayed to the users. Granular data would appear extremely close to or even overlapping each other when displayed on a small screen. It would be tough to derive any insights in such cases.

• Large amount of Data: Multi-line chart data can get quite complex as it would be data about multiple entities (lines) across a variable timeline (often even years).

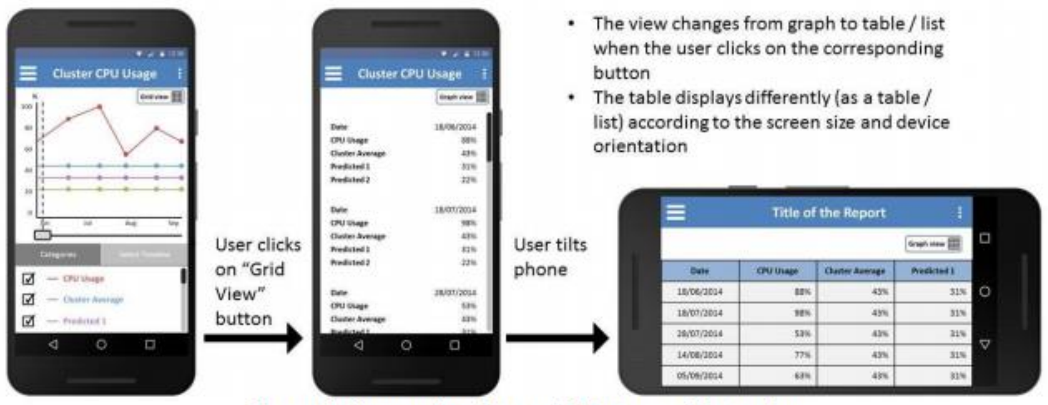
Hence, visualization would be equally complex, if all granular data was displayed at one time on a small screen.
• Legibility: As a large amount would have to be displayed on a small screen in a single visualization, they would appear extremely close to each other. This would make the data illegible.
• Orientation: Users might hold the device in either landscape or portrait mode. This brings in variation as the length and breadth of the available viewport interchange the values. In such cases, the entire solution would need to be flexible enough to accommodate this variation.
• High Variation in data: As the data itself is very unpredictable, it would be tough to budget dedicated space for each data point. The amount of data points on a line, number of lines, variation in the trends of each line and values of each data points can differ vastly from chart to chart and hence would need a flexible solution.

Solution: Display a toggle button along with the chart that allows users to toggle between a chart view and a RESPONSIVE DATA GRID view. For the RESPONSIVE DATA GRID view, display the same data in the form of a table in landscape mode and a list in portrait mode.

The table would also adjust its formatting depending on what the screen size and orientation of the mobile phone is. This would keep the data accessible in a comparatively simpler and easier form. The toggle button would also make it easier for the user to quickly navigate between the two versions of data display, thus making it easier to view and understand the data as well as the visualization. Figure-7 illustrates the responsive data grid pattern in detail.
Design
• Design the button such that it toggles between "grid view" and "graph view" depending on the current view
• Display data in the form of a table or data list depending on screen size and device orientation when in table view the data must be displayed
Interaction
• The view toggles between grid and graph when the user clicks on the corresponding button

Consequences: Key benefits (+) and liabilities (-) of this pattern are as follows:
+ A smart table responding to size and orientation would display information in a clean and orderly format irrespective of screen size
+ It would be easier for users to read and interact with the data
− User would have to scroll a lot horizontally for huge amounts of data, and may lose perspective
− Not suitable for multi-dimensional data

Known Uses: Figure-8 depicts the use case where CPU usage data is displayed for six years. This data is served in the form of a chart and also in the form of a smart table, on demand.


Cluster CPU Usage
Grid view
CPU Usage
Cluster Average
Predicted 1
User clicks on "Grid View" button
Cluster CPU Usage
Graph view
User tilts phone
• The view changes from graph to table / list when the user clicks on the corresponding button
• The table displays differently (as a table / list) according to the screen size and device orientation
Title of the Report
Graph view

Date	CPU Usage	Cluster Average	Predicted 1
18/06/2014	88%	43%	31%
18/07/2014	98%	43%	31%
28/07/2014	53%	43%	31%
14/08/2014	77%	43%	31%
05/09/2014	63%	43%	31%

Figure 8: Responsive data grid Pattern – Illustration

In some cases, the user would want to view precise data for CPU usage, average as well as predictions for a large timeframe like a few months or even years. On clicking the "grid view" button, data regarding CPU usage, cluster average, predictions for any selected timeframe appears as a table or list depending on the device’s screen size and orientation. The data would display as a table in landscape mode and a list in portrait mode. It was observed that users found it easy and intuitive to read contents from the table/list alternative.

This pattern is also been applied as a feature in various JavaScript libraries like slacktable [19], Basic Table [20], Responsive Tables JS [21] and Responsive [22].

S12 - ARTICLE: PATTERNS OF TRUST IN UBIQUITOUS ENVIRONMENTS

PATTERN: #60
Pattern name: Trust and Authentification ** Pattern Categories: Expectations, Moral principles Illustration: A user wants to authentificate himself to enter a room. In order to do so he must identify himself to the system. This already assumes that a trust between the user and the system is established. Problem (When used): During the authentification process, it is important that the trustor (human or machine) trusts the trustee (machine). Context: Web-based environment: Imagine a mobile or desktop web-application where users have to authenticate themselves before they can use the systems. Ubiquitious environment: Imagine a ubiquitous application where users are authenticated automatically. Forces: -Security issue: Authenticator can be faked or forgotten. -Privacy issue: The system should not provide personal information without the accordance of the user. Solution (How): Exchanging Signals: Giving a reason that the trustor might trust the trustee. Give early feedback about the system state, i.e. whether the authenticator is available and works. To improve understandability and credibility, give the user a simple and straightforward advice how to use the authenticator. Examples: (1) A user shall enter account data but needs to trust in the webservice. An indication might be to use a secure webpage so that the user can check the issuer of the security certificate. (2) A user shall be authenticated automatically by voice which allows her to enter a room. She needs to trust that the device only allows her in the room but not informs a third party that she entered the room (privacy protection). Trusting Action: Establishing a trust relationship: The trustor risks that his trust action will not be fulfilled because the trustee might defect his request or action. With respect to understandability, follow typical authentication processes known from similar systems. To improve reliability and understandability, ensure that similar processes are designed in the similar way. Give feedback by reacting to the user action. Design for error to improve fault- tolerance. To improve security, use more than one and an alternative authenticator. Example: Instead of using one authenticator which can be forgotten or faked, the application can use several authenticators. The application can check IP address, user name, user password, and might ask a security question. Fulfillment: A trust relationship is established, and the trustee fulfills the trustor's request. The system should work properly and correct. Give feedback that the request was fulfilled. If not, give direct and simple advice to improve understandability and

credibility.

Related Patterns:
Trust and Authentication is the parent pattern of Trust and Identification and Trust and Authorization. Patterns in external pattern languages are for example Checking for Correctness, Retrieving Forgotten Passwords [Bass:2001] and User Profiles [Folmer:2003].

Known Uses (Examples):
Online Banking, Mobile Banking, Digital Signatures

S13 - ARTICLE: RUCID: RAPID USABLE CONSISTENT INTERACTION DESIGN PATTERNS-BASED MOBILE PHONE UI DESIGN LIBRARY, PROCESS AND TOOL

PATTERN: #61

Name:
Soft key Window - Sample Widget pattern

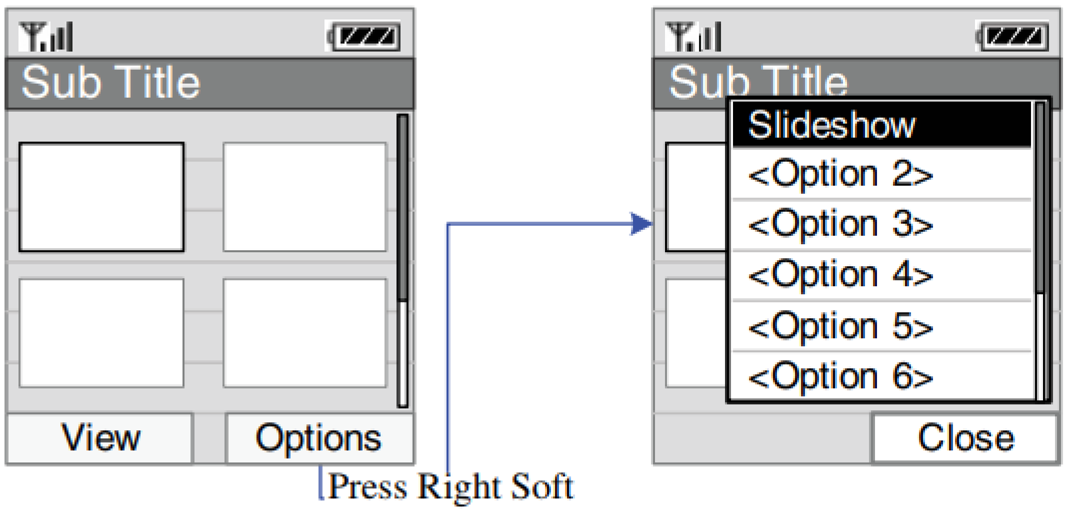
Problem
User needs to access additional functions that can be performed on a screen.

Context
For a given Screen, a User has more number of actions possible than the maximum number of Soft keys.

Solution
One of the options that can be accessed through a softkey can provide gateway to multiple options. The User can move to these options and select a desired one.

Rationale
A limited number of Softkeys can be displayed at any one time. A dedicated key cannot be assigned to each option because:
1) The options and their number keep changing depending on the screen.
2) The surface area of mobile is small and limited.
Using a single key to access a variable sized list allows for any number of items to be accommodated.

Examples



Related Patterns
Softkey, List, Scroll

S14 - ARTICLE: SPATIAL DATA AND MOBILE APPLICATIONS - GENERAL SOLUTIONS FOR INTERFACE DESIGN

PATTERN: #62

Name:
Infinitive Area + Context. This pattern extends the Infinitive Area.

Problem:
A complex and/or interactive visual information that should be presented as a single image. Consider the need to convey a lot of specific spatial information - either of homogeneous or heterogeneous type but interconnected in some way – that are located in offscreen space.

Solution:
The solution is to display coloured visual metaphors, which provide information clues about different sectors of the off-screen space. Use different colour intensities or dimensions to give qualitative and quantitative information about objects located in the corresponding off-screen sector.

Variations:
Multimodal Infinitive Area + Context Infinitive Area

Interaction Details:
When users change the spatial focus the information represented by the visual 192 metaphors are automatically updated.

Presentation Details:
The visual metaphors position must be significant. It gives intuitive information about directions. In a general way they can be inscribed along the borders of the view of the application that in many cases corresponds to the whole screen of the device.

Antipatterns:
Use caution with the size of the metaphors. Graphic objects which are too small may be misunderstood by users. Be careful with the number of the graphic objects. A large number may reduce the usability of the ordinary user interface. Use Itten's theory to choose the right colour combination of the contiguous graphic objects. A wrong combination may make users confused.

Examples and figures:
Consider a high resolution image where the subjects are human faces. Supposing that the software automatically recognizes the faces as points of interest, the frame can guide users towards them. (fig 8)
Consider a spread text like the web page of a newspaper wherein users look for a particular word through the appropriate search tool. A frame can guide users to find the word in the text, the colour intensities could give information about the number of the words in the given direction or the distance from the current view. (fig 9)
Consider the case when people use a map application to find restaurants and bus stops around them in a range of 3 km. Restaurants and bus stops are two different categories of POIs. Two nested frames – each one related to a specific category of POIs – provide users with information about the directions to take. (fig 10)

In each case the colour intensities of a given portion of the adopted visual metaphor can give information about different qualitative elements such as the distance from the current focus, the number of elements located outside the screen or a combination of them.

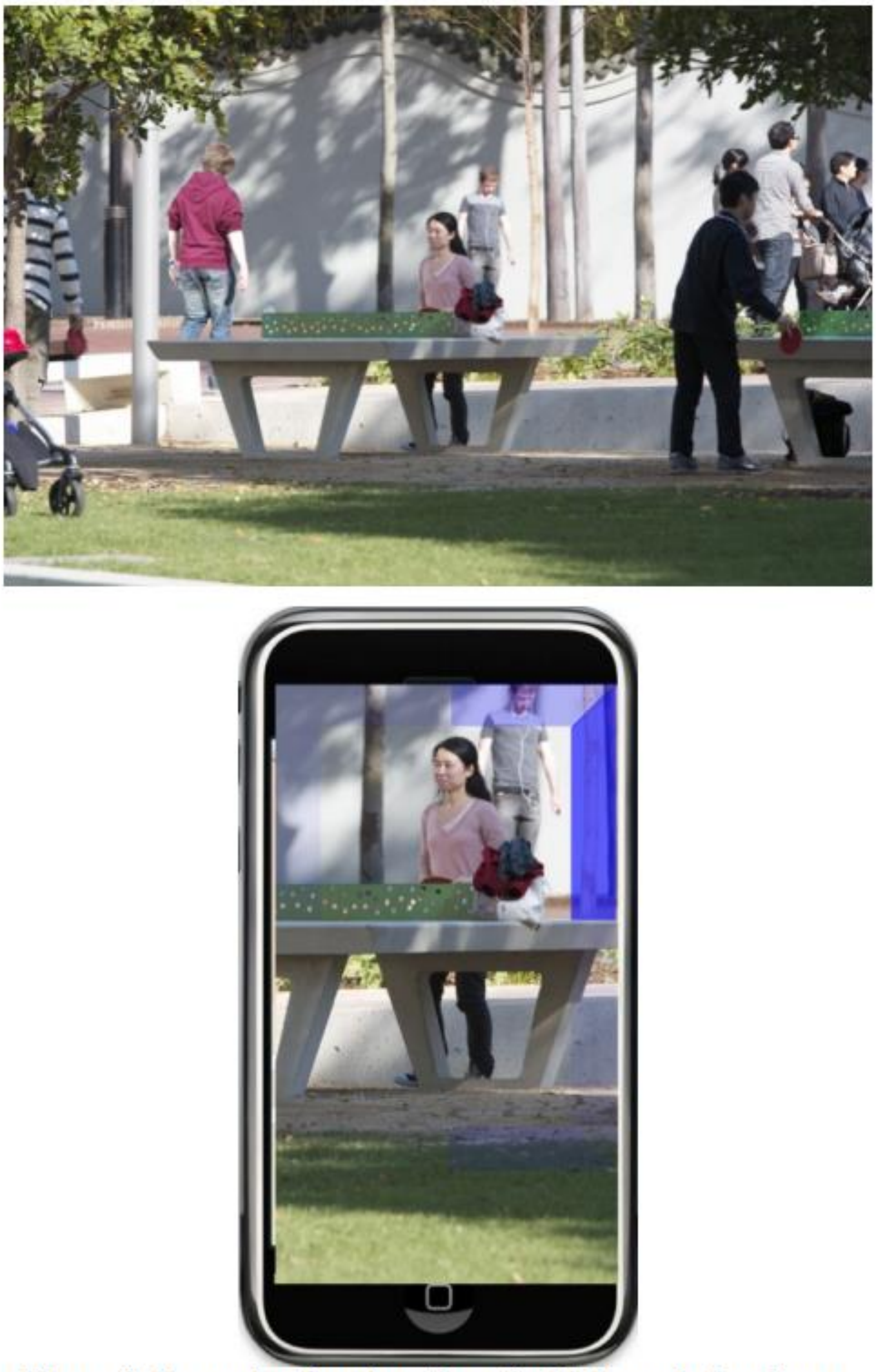
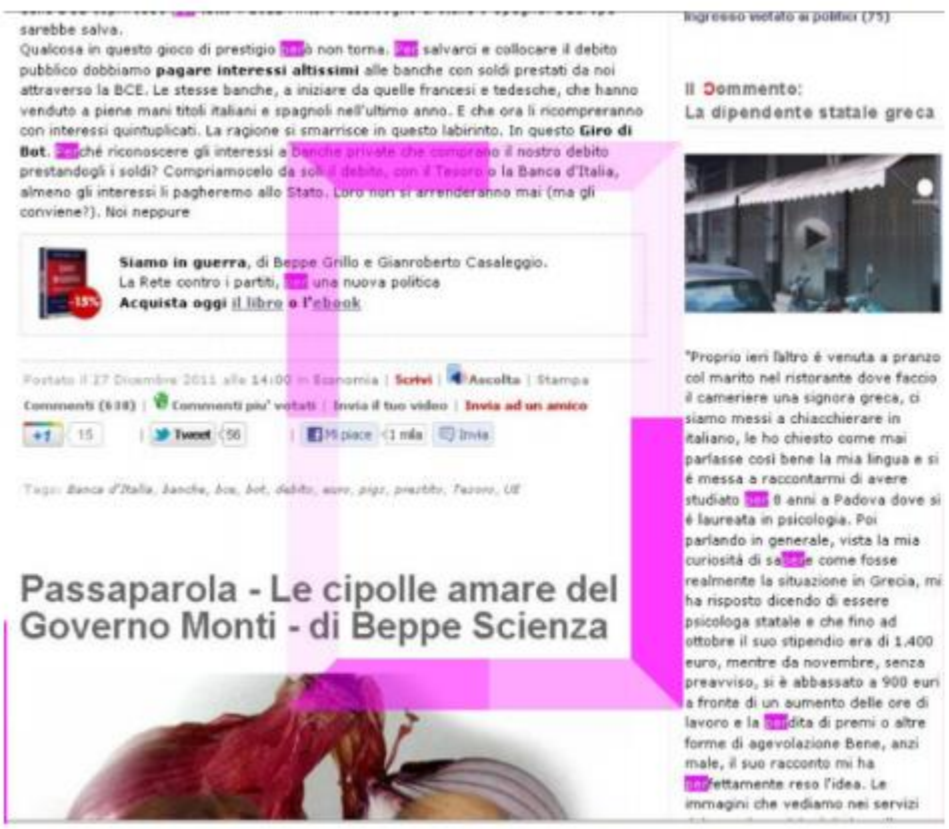


Figure 8. Example of exploration of a high resolution image



sarebbe salva.
Qualcosa in questo gioco di prestigio però non torna. Per salvarci e collocare il debito pubblico dobbiamo **pagare interessi altissimi** alle banche con soldi prestati da noi attraverso la BCE. Le stesse banche, a iniziare da quelle francesi e tedesche, che hanno venduto a piene mani titoli italiani e spagnoli nell'ultimo anno. E che ora li ricompreranno con interessi quintuplicati. La ragione si smarrisce in questo labirinto. In questo **Giro di Bot**. Perché riconoscere gli interessi a banche private che comprano il nostro debito prestandogli i soldi? Compriamocelo da soli il debito, con il Tesoro o la Banca d'Italia, almeno gli interessi li pagheremo allo Stato. Loro non si arrenderanno mai (ma gli conviene?). Noi neppure

Siamo in guerra, di Beppe Grillo e Gianroberto Casaleggio.
La Rete contro i partiti, per una nuova politica
Acquista oggi il libro **o l'**ebook

Postato il 27 Dicembre 2011 alle 14:00 in Economia | Scrivi | Ascolta | Stampa
Commenti (638) | Commenti piu' votati | Invia il tuo video | Invia ad un amico

Tags: *Banca d'Italia, banche, bce, bot, debito, euro, pigs, prestito, Tesoro, UE*

Passaparola - Le cipolle amare del Governo Monti - di Beppe Scienza

Ingresso vietato ai politici (75)

Il Commento:
La dipendente statale greca

"Proprio ieri l'altro é venuta a pranzo col marito nel ristorante dove faccio il cameriere una signora greca, ci siamo messi a chiacchierare in italiano, le ho chiesto come mai parlasse cosi bene la mia lingua e si é messa a raccontarmi di avere studiato per 8 anni a Padova dove si é laureata in psicologia. Poi parlando in generale, vista la mia curiosità di sapere come fosse realmente la situazione in Grecia, mi ha risposto dicendo di essere psicologa statale e che fino ad ottobre il suo stipendio era di 1.400 euro, mentre da novembre, senza preavviso, si è abbassato a 900 euri a fronte di un aumento delle ore di lavoro e la perdita di premi o altre forme di agevolazione Bene, anzi male, il suo racconto mi ha perfettamente reso l'idea. Le immagini che vediamo nei servizi

Figure 9. Example of research of words in a web site

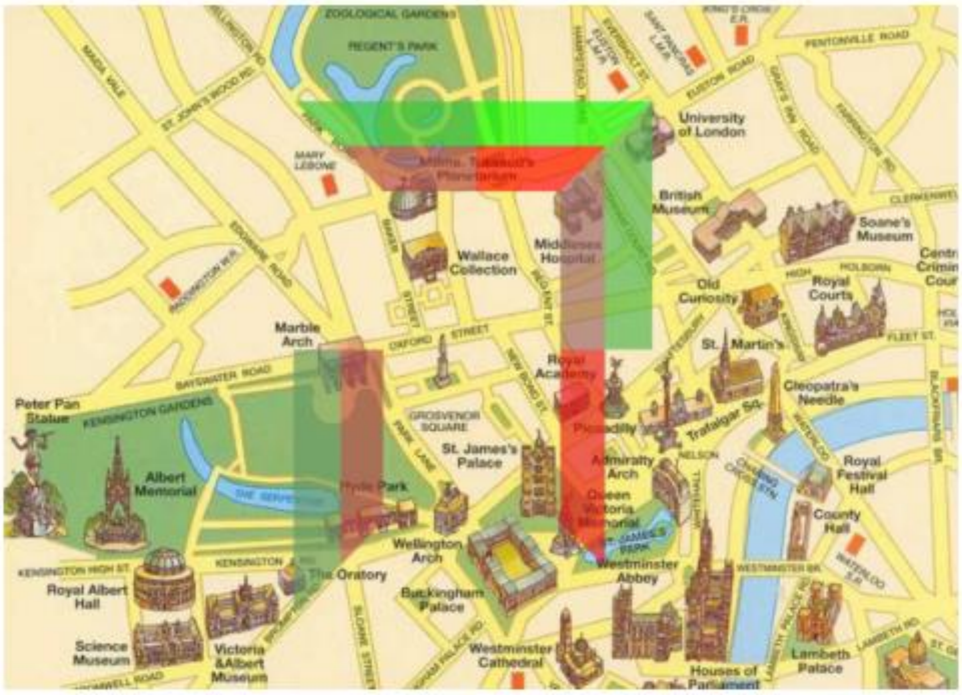


Figure 10. Example of exploration of POIs on a map

PATTERN: #63
Name: Multimodal Infinitive Area + Context. This pattern extends the Infinitive Area + Context. Problem: Consider the need to support users through different senses (to deal with diverse users capabilities or with extraordinary environmental situations). Solution: The visual metaphors are associated with auditory or vibro-tactile feedback. Use different sound sources for each metaphor or modulate tactile vibration. Variations: Infinitive Area Infinitive Area + Context Interaction Details: When users change the spatial focus the information perceived changes accordingly. Users interact with the multimodal interface either performing gestures on the screen, or triggering the movement sensors off or through the GPS module. Presentation Details Different sound sources must be clearly discernible as well as vibro-tactile modulation. Antipatterns: Use caution with the usage of vibration, it may annoy users. Examples and figures: Consider the case when a visually impaired user needs to orient himself while walking. He can use a mobile map application assisting his impaired sight with sounds and vibrations triggered off by his touch on the screen or his movements.

S15 - ARTICLE: SPEECH AUGMENTED MULTITOUCH INTERACTION PATTERNS

PATTERN: #64

Name:
AUDITORY MODE SWITCH

Intent:
Use voice commands to provide mode information to determine the exact operation semantics of an user interaction.

Context:
A number of operations on common operating systems require mode information to determine the exact operation semantics. This mode switch is usually temporary in nature and only maintained for the duration of the usersystem-interaction. On desktop computers it is commonly achieved with key presses. For example, when a user drags a file object, the CTRL-key is used to determine whether the file should be moved or copied.

Problem:
Touch-operated devices often do not come with physical keyboards that could be used to provide (temporary) mode information to ongoing operations (cf. VOICE AS TEXT INPUT; fallback strategies like explicit mode switches or mode selection based on timing suffer lower efficient. Thus, the problem remains, how the user can be enabled to (temporary) switch the mode of an ongoing operation, e.g. drag&drop, with touch based systems.

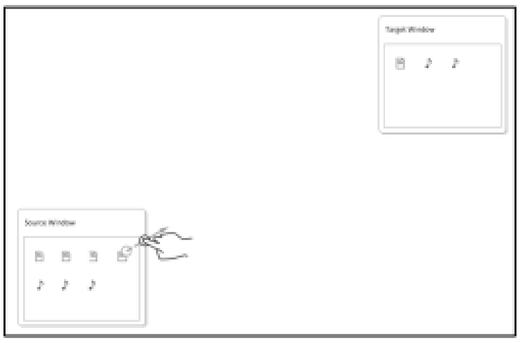
Forces:
—Some operation, e.g. drag&drop, may require additional input to give the operation a meaning.
—A second mouse button or keyboard does not exist for touchscreens.
—A popup-menu requires an additional operation and is thus less efficient.

Solution:
The solution starts after the user started to drag an object. The start of the dragging operation is used to start the recognition process and expect a spoken command from the user to give the drag operation a meaning. To implement this strategy, consider the following:
(1) The user performs a multitouch operation, e.g. drags an object.
(2) Expect a spoken command during the drag operation to select the mode of operation, e.g.copy.
(3) Visualize the selected mode.
(4) In case of permanent recognition errors, fall back to show a popup-menu to select the mode.

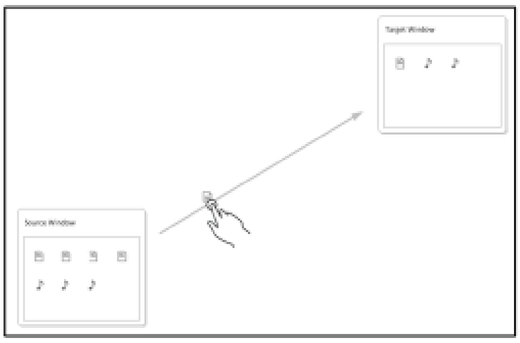
Consequences:
-Provides an efficient way to switch the mode
-Mode can be flexibly changed during the drag operation
-Can be conflicting for multiple users

Known Uses:
The Eee PC [Asus] is a lightweight PC that can be operated by touch and speech. Supported by the Windows 7 operating system it is possible to drag an icon with the finger and utter the command Show context menu to emulate the right mouse click. As a consequence, a popup menu appears from which the user can select an item by simply speaking the label.

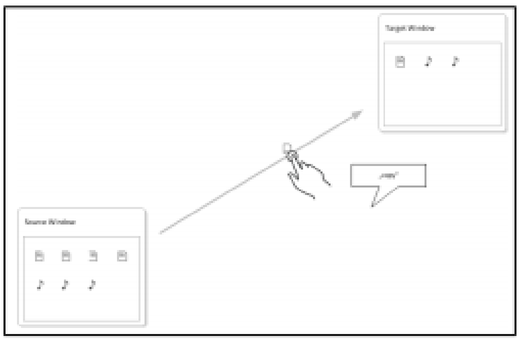
Related Patterns:
Can be combined with VOICE-BASED DISTAL ACCESS to move the target object into the user's reach



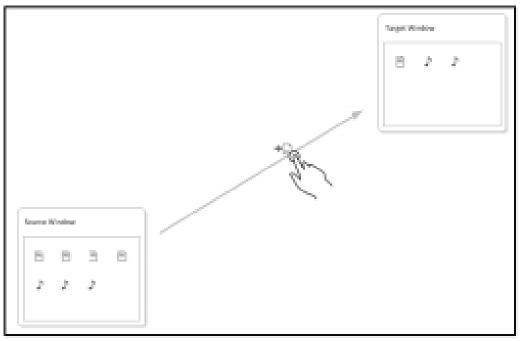
(a) user starts dragging object



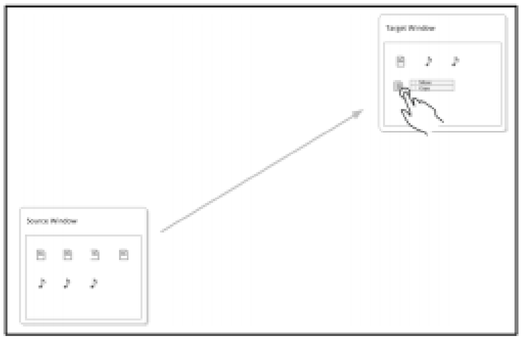
(b) user moves object towards destination



(c) user utters command for mode switch



(d) mode switch is visualized



(e) in case of permanent recognition errors, a menu is displayed for mode selection

Fig. 3: Sketch of the AUDITORY MODE SWITCH process

PATTERN: #65

Name:
VOICE-BASED DISTAL ACCESS

Intent:
Enables the user to efficiently manipulate and link objects beyond his/her immediate reach. Context As noted in section 1.2, large displays are very efficient for navigation in large datasets. However, when touch input is used to allow easier collaborative situation exploration or collaborative decision making, some of the objects displayed on the screen may be out of reach for a user who desires to manipulate them or link them to other objects (because users have to stand close to the display to use touch-input).

Problem:
Touch interaction is a natural and efficient means for manipulating objects as long as they are within immediate reach. For objects beyond immediate reach, the user either has to change his/her physical location or revert to zoom-and-pan navigation. As the latter were found to be inferior to physical navigation [Ball et al. 2007], the question arises how the manipulation of objects beyond immediate reach can be enabled without requiring the user to move larger distances, but also without reverting to a zoom-and-pan interface?

Forces:
—Efficient touch interaction requires that objects are within the user's immediate reach.
—Physical movement (of users) towards an object can be time consuming.
—The physical dimensions of the display may exceed the size of the user.
—Resorting to voice-only interfaces may decrease efficiency, i.e. the exact description of distal objects can be time-consuming; relevant meta-data for the description may not be visually present.

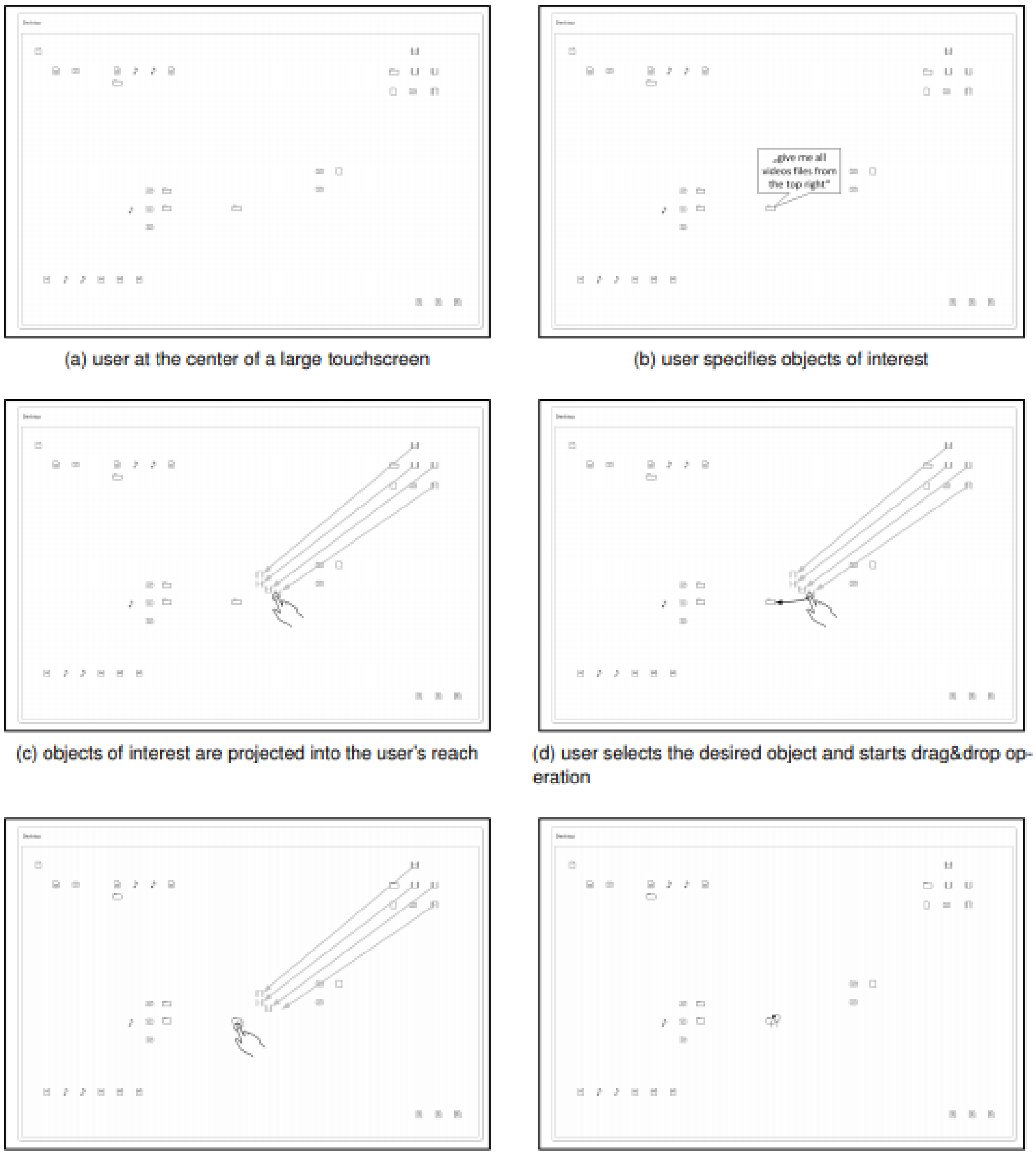
Solution:
If an object is out of reach, the user can provide a description of the targeted object using natural language: to avoid the potentially time-consuming comprehensive description of the targeted object, the user can use an incomplete description which will subsequently be moved into his immediate reach. Using touch- input the desired object can then be selected without the need for a comprehensive description via speech. To implement this strategy, consider the following:
(1) The user provides a (potentially incomplete) description of the desired object by using attributes from its meta-data (e.g. "last modified", "object/file type"), or its location on the screen (e.g. "located in sector x" when a screen grid is used).
(2) Mark those objects that match the current description and project them into the users immediate reach (maintain the original objects' positions for other users).
(3) The user selects the desired object, and manipulates or links it to other objects via touch-interaction.
(4) Once the user confirms he/she has completed his operation, remove the projected objects.

Consequences:
-Objects out of reach can be manipulated.

-Selection based on meta-data can gain new insights into data.
-Using different types of meta-data, users can progress slowly from novice to expert with object selection (e.g. using location-based specifications at the beginning, but modification-dates or even combinations with more training).
-May be cumbersome to describe the location exact enough.

Known Uses:
While targeting rather standard desktop PCs instead of devices with very large displays and geared more towards mouse- instead of touch-input, the Windows mousegrid [Microsoft] can serve as an example for this pattern. The mousegrid is invoked by the spoken command mousegrid and partitions the screen into nine sectors which can be addressed by their numbers. Using click Xth sector, it is possible to navigate nearer to the desired target. For systems with large displays and touch control this could be made significantly easier by moving the desired sector into the range of the user and proceeding further via touch control.

Related Patterns:
VOICE-BASED DISTAL ACCESS is a concrete implementation of the (rather broad) EXTENDING REACHABILITY pattern [Remy et al. 2010].



(a) user at the center of a large touchscreen

(b) user specifies objects of interest

(c) objects of interest are projected into the user's reach

(d) user selects the desired object and starts drag&drop operation

(e) user completes drag&drop operation

(f) projection is dismissed after operation completion

Fig. 4: Sketch of the VOICE-BASED DISTAL ACCESS process

PATTERN: #66

Name:
VOICE AS TEXT INPUT

Intent:
Facilitate text input in situations where standard keyboard input is not feasible or not efficient.

Context:
Many touch-operated devices do not feature a physical keyboard. When text input

is required, fallback solutions, e.g. soft keyboards, have to be employed.

Problem:
While touch input is quite efficient for selecting and directly manipulating objects, touch devices often exhibits less than optimal performance for text input. Especially for large vertical touchscreens or multi-user settings on interactive tabletops, the use of physical keyboards is often not feasible, or, at least, inconvenient. Soft keyboards, which have commonly been used as a replacement, lack tactile feedback and are tedious to use in vertical orientations. Speech input, on the other hand, need trigger points to distinguish between inter-personal communication, commands and actual text input.

Forces:
—Physical keyboards are not available e.g. for touch devices.
—Physical keyboards or soft keyboards require additional space.
—Vertical touch displays may not offer a convenient storage for a physical keyboard (near to the display).
—A typist may not be available or too expensive.
—Speech recognition is error prone and may require user input for word disambiguation.
—Speech is asymmetric, i.e. humans can speak faster than they type and listen slower than they read.
—Text input via soft keyboards is usually less efficient compared to physical keyboards.
—Vertical touch displays make text input via soft keyboards uncomfortable.
—Soft keyboards cover display space, thus making it unavailable to information presentation.
—The available space should be used to display information.
—Current operating systems do not support concurrent text-input of multiple users by multiple keyboards.

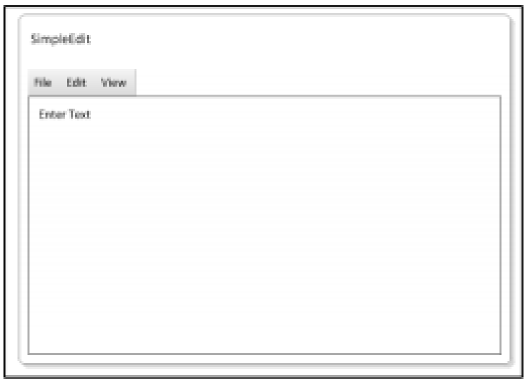
Solution:
For objects which allow text input (e.g. for labels or large portions of text as part of a document), the user can select the respective input region. Upon activation of the text input region, the speech processor can be activated and the desired text supplied via voice-input. Given separate voice input channels (e.g. using individual headsets) and user tracking facilities, this strategy can also be applied for multiple users concurrently. To implement this strategy, consider the following:
(1) Watch for events that indicate an input region accepting text has become active
(a) If there is only a single voice input channel, activate it.
(b) If multiple users are present and the user's identity is known, activate the respective voice channel.
(2) Start voice recognition and translate voice into text until the user finishes (i.e. the text input region becomes inactive again).
(3) Transfer recognized text into active input region.

Consequences:
-Provides an alternative way to enter text when physical keyboards are not accessible or cumbersome to use

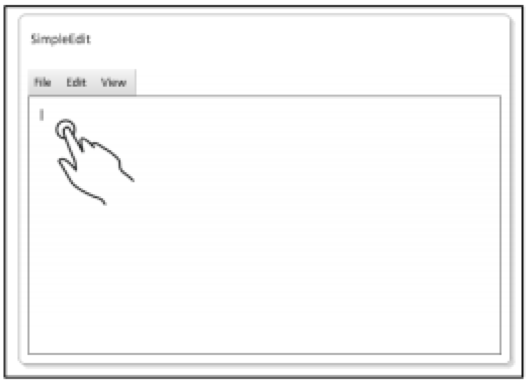
-Speech is more efficient than typing for text input [Cohen and Oviatt 1995]
-Given separate voice-channels, multiple users can input text simultaneously, increasing overall productivity
-Requires an error correction strategy

Known Uses:
This pattern has been implemented in mind mapping scenarios, where the user can point somewhere and utter a word to enter the text. One of these implementations is WordPlay [Hunter and Maes 2008].

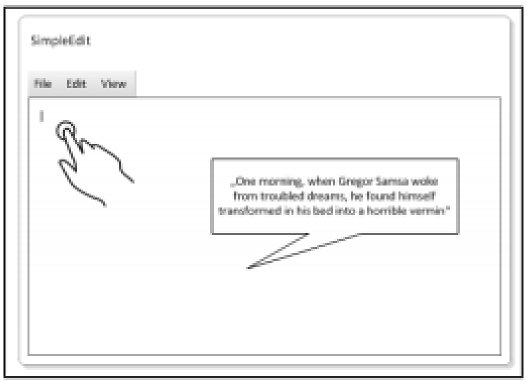
Related Patterns:
Can be combined with VOICE AS PRIVATE OUTPUT CHANNEL when multiple users are involved an private information is also required for reference. Errors can be handled via TOUCH-BASED ERROR CORRECTION. PEN INPUT DEVICE, PHYSICAL KEYBOARD or ON-SCREEN KEYBOARD as described in [Remy et al. 2010] offer alternative ways to enter text. Expected to be more suitable with TILTED TABLE than with a LARGE COLLABORATION TABLE [Remy et al. 2010]. Offers separate input-channels for different users as requested in USER IDENTIFICATION [Remy et al. 2010].



(a) user wants to input text at touchscreen

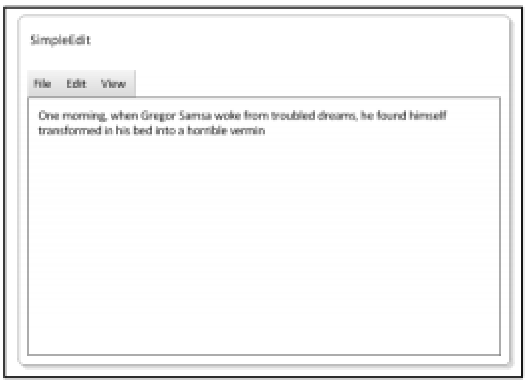


(b) user activates input region by touching and holding the text area



„One morning, when Gregor Samsa woke from troubled dreams, he found himself transformed in his bed into a horrible vermin"

(c) user utters the text for the new document



One morning, when Gregor Samsa woke from troubled dreams, he found himself transformed in his bed into a horrible vermin

(d) on release, the respective recognized text is transferred to the text area

Fig. 5: Sketch of the VOICE AS TEXT INPUT process

PATTERN: #67

Name:
VOICE AS PRIVATE OUTPUT CHANNEL

Intent:
Provide a separate channel for the output of private information on shared displays.

Context:
Large screens are suitable means to display shared visual content, which can be consumed by all participants and enable collaborative work and simultaneous interactions of multiple users. A common strategy to provide feedback to the users is the additional use of auditory feedback, e.g. to indicate errors.

Problem:
When multiple users engage in an interactive session at a shared display, information is public by default. While this is usually desired for shared visual content, this is not necessarily wanted for shared audio content. The spatial character of audio does not allow for a separation of the delivered information as it is possible in graphical interfaces through the means of a PRIVATE SPACE [Remy et al. 2010]. It has also been shown that simultaneous audio feedback can confuse the user [Everitt et al. 2004] How to provide individual users with separate channels on a shared display?

Forces:
—Information that is relevant for an individual (only) should not be exhibited via the shared display.
—Some information is interesting for all.
—Semantically separated tasks should be spatially separated [Tse et al. 2004]. —Individual feedback to specific operations may be desired.
—In collaborative settings, the users are not always aware about the interaction of their peers [Hancock et al. 2005]
—Audio output to other participant’s actions distract other collaborators.
—Audio-based feedback can be useful in the absence of haptic feedback.
—Headsets block the users ear and hinder her from collaborating with others.

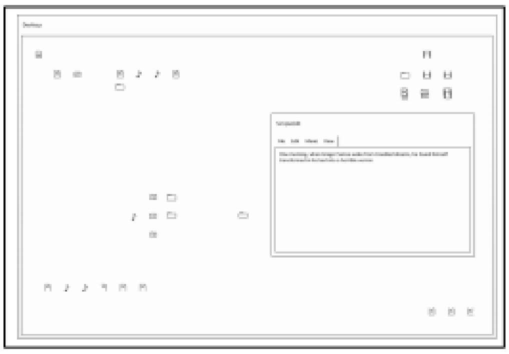
Solution:
At a shared display, information or feedback for specific actions directed at a specific user can be supported by using voice output via individual audio channels for the respective user. To implement this strategy, consider the following:
(1) Supply users with individual audio channels, e.g. via mono-aural bluetooth headsets
(2) The USER IDENTIFICATION pattern can be employed to identify individual users’ interactions.
(3) Use the individual audio channel to redirect feedback while interacting with the objects.

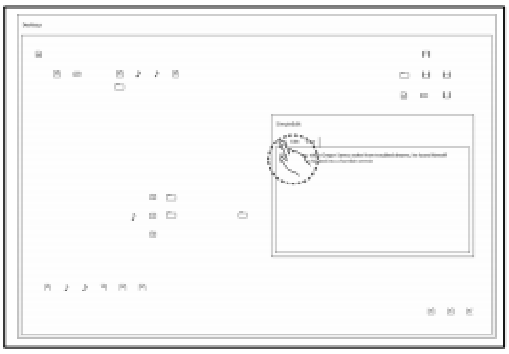
Consequences:
-Provides a way to present individual information or feedback for specific actions to

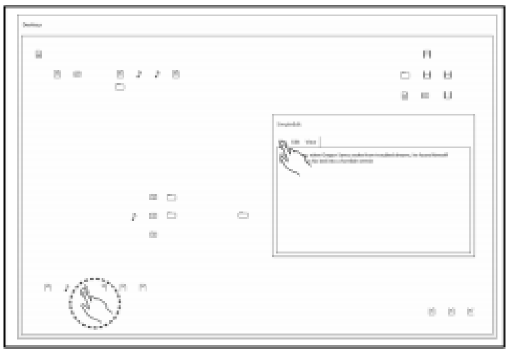
individual users at a shared display.
-Can also be used to deliver confidential information (a technique called attribute gates can be used to quickly set the corresponding access rights [Sulaiman and Olivier 2008]). However, this would require in depth thoughts solutions to determine the privacy relevance of information.
-Users are still able to collaborate with others.
-Audio information is transient, thus information needs be remembered by the respective user, putting additional cognitive load onto this user.
-Requires headset for each user.

Known Uses:
The work of Morris et al. [Morris et al. 2004] show that individual audio channels can be used to the benefits of collaborative tasks on a shared touch display. Even when the information exhibited via the individual audio channels is not strictly private, these channels allow an individual participant to listen to information (music in the case of the study) without disturbing others.

Related Patterns:
Uses USER IDENTIFICATION [Remy et al. 2010] to identify the user. Can be combined with VOICE AS TEXT INPUT CHANNEL to reuse the headset for text input. Offers an alternative solution for some privacy as it is described for graphical media in PRIVATE SPACE [Remy et al. 2010] for audio.


(a) user A at the right of the touch screen


(b) user A works with an application


(c) user B at the left side starts to work with another application

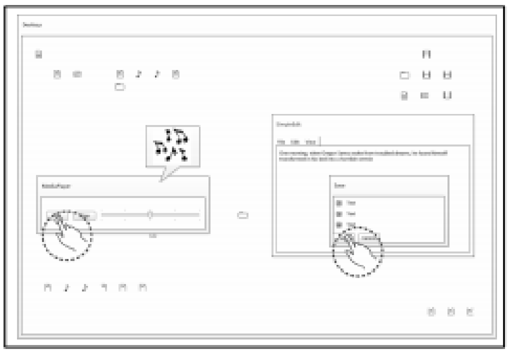

(d) the output of user B's application is delivered via a private output channel, so that user B is not influenced while working with her application

Fig. 6: Sketch of the VOICE AS PRIVATE OUTPUT CHANNEL process

PATTERN: #68

Name:
TOUCH-BASED ERROR CORRECTION

Intent:
Provide an easy way to select the desired word, when speech recognition fails.

Context:
The use of speech enables the user with efficient means to enter their data or to issue commands which is exploited by all of the patterns in this language. This statement is true as long as no recognition errors occur.

Problem:
Speech recognition is error prone and may lead to new errors when the user tries to correct the error. Additionally, voice based error correction turned out to be very time consuming. A single misrecognized word may lead to the situation that the system is not able to determine the user goals. Here, an efficient way of correcting the misrecognized parts of the input is needed. How to disambiguate unclear speech recognition results?

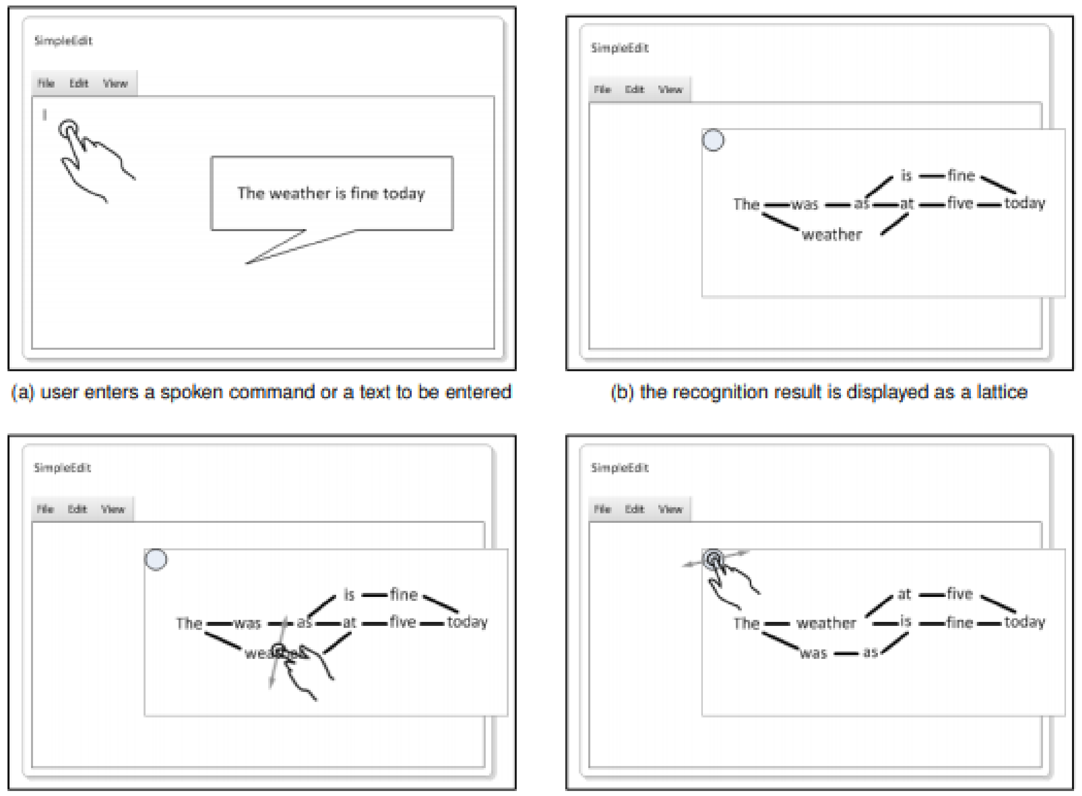
Forces:
—Efficient speech based interaction requires a high recognition accuracy.
—Voice based error correction is time consuming and may lead to an error spiral if new errors occur during the correction phase [Schnelle-Walka 2010].
—Users need an efficient means to correct spoken input errors.
—Multimodal error correction is faster than unimodal correction by respeaking [Suhm et al. 2001].
—Displaying different recognition hypotheses requires additional screen space.
—The available space should be used to display actual work artifacts.

Solution:
The solution deals with a visualized representation of the different recognition hypotheses as a word graph, also know as word lattice. The user can use touch to rearrange it so that it matches the desired utterance. To implement this strategy, consider the following:
(1) After the user uttered a phrase, the system visualizes the possible recognition hypotheses as a word lattice if there is low confidence in the recognition result.
(2) Add touch based interaction to allow the user to adjust the displayed recognition hypotheses as wanted (e.g. by using multitouch marking menus [Lepinski et al. 2010]).
(3) End the correction phase either e.g. after the user dragged the lattice to the target input location.

Consequences:
-Easy and intuitive way to correct the spoken input.
-An input can be accepted although it had a low confidence score.
-Efficient means of correction recognition errors.
-Can be time consuming when error rate is high.
-Cannot be applied when speech recognition fails completely .
-Cannot be applied when speech recognition did not detect desired word as a valid hit.
-Limited suitability for mobile settings due to the limitations of the available screen space.

Known Uses:
Huggins et al. describe a system that visualizes the word lattice after the recognition process [Huggins-Daines and Rudnicky 2008]. The user can select the utterance using touch gestures that have been simulated by a mouse.

Related Patterns:
Can be combined with VOICE AS TEXT INPUT CHANNEL as an error correction strategy.



(a) user enters a spoken command or a text to be entered

(b) the recognition result is displayed as a lattice

(c) user corrects the recognition result by dragging the possible words

(d) once the user is satisfied, the correction phase is terminated by dragging the corrected result to the target input location

Fig. 7: Sketch of the TOUCH-BASED ERROR CORRECTION process

PATTERN: #69

Name:
SELECT BY TOUCH, OPERATE BY VOICE

Intent:
Combine the strength of touch (easy selection) and voice (quick shortcuts to commands) to allow for an efficient
interaction.

Context:
Objects displayed on the screen can be manipulated by touch-based interaction at your finger tips. Usually, there are multiple actions that the user may perform from which she has to select the wanted action.

Problem:
Command activations in multitouch scenarios are usually implemented by gestures [Dietz and Leigh 2001; Wu and Balakrishnan 2003] or menus [Brandl et al. 2008; Wu and Balakrishnan 2003]. However, gestures are not always natural and must be

learned and memorized [Wobbrock et al. 2009] which make them suitable only for only a very limited set of of commands. Menus do not have this limitation but consume additional space on the display and ”. . . are potentially awkward to use with finger touches”[Hesselmann et al. 2009]. How to efficiently select and manipulate objects on the screen with a low learning curve and without the need for additional space on the display?

Forces:
—Exact definition of relevant object via voice can be cumbersome.
—Selecting an object by touching it is a natural and efficient way to identify an object.
—Manipulating objects by gestures is not necessarily intuitive.
—Voice commands provide convenient shortcuts to operations.
—The system needs an indicator to distinguish between conversation among the collaborators (or to others passing by) and spoken commands to interact with the system.
—Displaying the available actions that can be performed with an object requires additional screen space, thus putting additional cognitive load onto the users to remember the information.
—The available screen space should be used to display the artifacts.

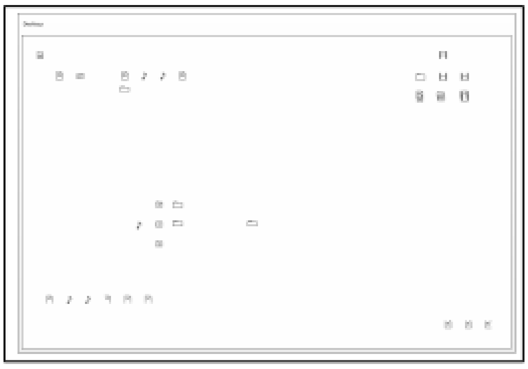
Solution:
The solution uses the information given by the selection of an object by a touch interaction to operate with it using voice commands. It exploits a common practice in everyday life to reference objects by gestures or touches without explicit naming them. From a technical perspective, this also helps to determine when a user may issue commands. To implement this pattern, consider a three-step approach:
(1) Provide handlers for selecting an object on the screen via touch
(2) Once a selection of an object has been recognized limit the expected commands that the user may utter to the valid actions that may be performed with the selected object [Weinberg and Harsham 2010].
(3) Expect a spoken command from the user and distinguish the following cases
(a) If the user utters a valid command, e.g. (”Delete”), execute the corresponding action.
(b) If the user does not say anything within a predefined time-span or the user started a gesture or uses another action (e.g. releases the object), it is likely that the user does not want to speak an command to execute an action. In this case, deactivate the speech recognizer.
(c) If an invalid command is detected, either if the user used a wrong command or speaks to another person, use RAPID REPROMPT [Schnelle-Walka 2010] to give the user another try and restart by expecting a spoken command.

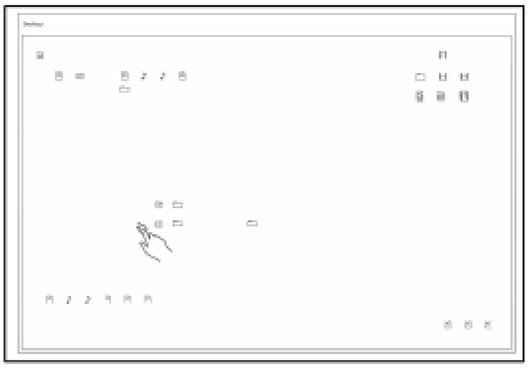
Consequences:
-Provides a way to perform voice operations on an object without the need to explicitly reference via name.
-Can also be applied to implement HIGH PRECISION INPUT [Remy et al. 2010] which is limited to graphical interaction.
-Using touch selection prior to voice command is proposed to make voice recognition easier and more reliable.
-Voice commands are not visible on the screen and have to be learned.

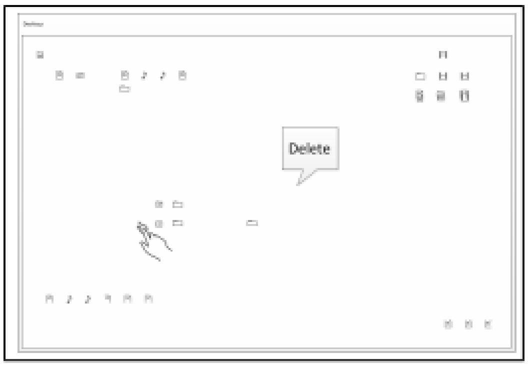
Known Uses:
The SELECT BY TOUCH, OPERATE BY VOICE-pattern has been adopted by a number of researchers for efficient and natural interaction: While initial work by Bolt [Bolt 1980], as well as newer works in the area of emergency management and geospatial data [Rauschert et al. 2002], rely on the combination of freehand gestures and voice input, others, such as the work of Zhang and Takatsuka [Zhang and Takatsuka 2007], quite literally make use of SELECT BY TOUCH, OPERATE BY VOICE, using it for a collaborative task on a large interactive tabletop. AT&T developed an application for mobile phones where users could touch a point on a map that was displayed on a smart phone and utter a spoken command to query for restaurants nearby [Johnston 2009].

Related Patterns:
Can be combined with VOICE AS TEXT INPUT to label previously unnamed objects. Can be a successor of VOICE-BASED DISTAL ACCESS if the object can not be reached.

Also Known As:
In the context of groupware and collaborative work, this pattern is also known as DEICTIC REFERENCE, see e.g. [Pinelle et al. 2003].


(a) user at the center of the touch screen


(b) user selects an object by touch



(c) user utters a command to trigger the action to perform with the selected object

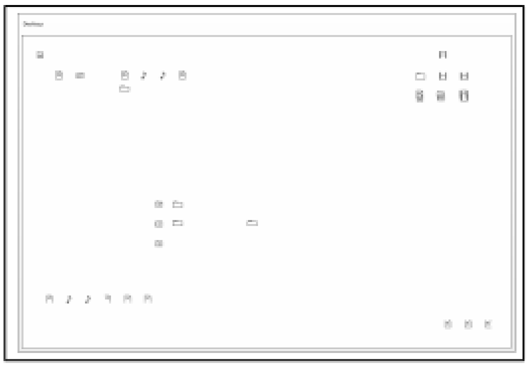

(d) the action is performed

Fig. 8: Sketch of the SELECT BY TOUCH, OPERATE BY VOICE process

S16 - ARTICLE: TEST PATTERNS FOR ANDROID MOBILE APPLICATIONS

PATTERN: #70
3.2.1 Side Drawer UI Test Pattern. Context: The Android OS provides several forms of navigation through its different screens and hierarchy. One of these is the Side Drawer (or Navigation Drawer) UI Pattern, i.e., a transient menu that opens when the user swipes the screen from the left edge to the centre of the screen or clicks on the application icon on the left of the application's Action Bar. Figure 1 depicts an example of this UI pattern, presenting an example of an application before and after opening the side drawer. According to Android's guidelines [Android 2015c], when this menu is open it should occupy the full height of the screen. Problem: A good Android developer should follow the Android guidelines in order to provide the best application possible to the end user. One of those guidelines refers to the position and size of the side drawer menu. Forces: —It may not be easy to identify when a side drawer is available to be open; —It is not trivial to identify the side drawer element; Solution The UI Test Pattern for this type of behaviour is only applied when the side drawer exists and consists in opening the side drawer and checking if it takes up all the screen height 1) Test Goal: "Side Drawer position on screen" 2) Set of Variables V: {} 3) Sequence of Actions A: [open side drawer] 4) Set of Checks C: "side drawer takes up all the screen's height" 5) Set of Preconditions P: "side drawer available" Consequences —Assurance that the application follows the guidelines for the Side Drawer design pattern —In case of a false negative, it may provide the user a false sense of correctness to the user Application Candidates —Side Drawer [Neil 2014; Android 2015b] (e.g. Vueling4, Booking5, Google Calendar6, Google Keep7)

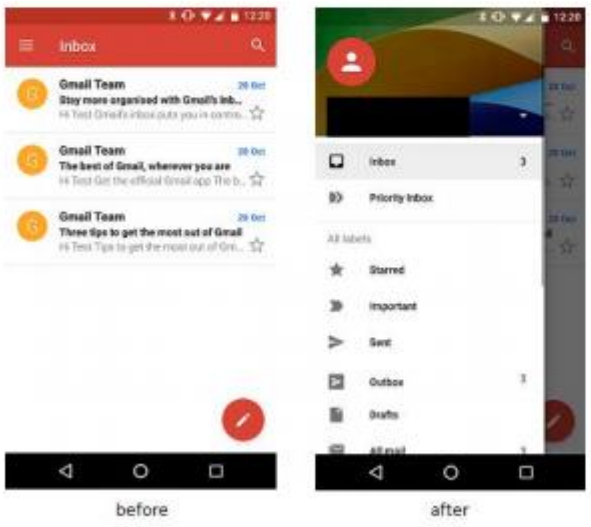


Fig. 1. Example of a *Side Drawer* UI Pattern (before and after being opened)

PATTERN: #71

3.2.2 Orientation UI Test Pattern.

Context:
Android devices have two possible orientations: portrait and landscape. When rotating the device, the screen of the application also rotates and its layout is updated. However, according to AndroidâA˘ Zs Guidelines for testing ´[Android 2015a] there are two main aspects the developers should test: custom UI code can handle the changes and no user input data should be lost.

Problem:
When the orientation of an Android mobile device changes, the application should adapt to this change without loosing any information previously introduced by the user.

Forces:
—The option of changing the orientation upon the deviceâA˘ Zs rotation must be enabled on the device ´
—It may not be trivial to compare different states of the screen because the items' position may change and they may not be visible without scrolling the screen

Solution:
In order to check the behaviour of the screen rotation, it is necessary to check if the screen elements and the previously input inserted data were not lost after rotating the screen.
1) Test Goal: "Data unchanged when screen rotates"
2) Set of Variables V: {}
3) Sequence of Actions A: [rotate screen]
4) Set of Checks C: "user entered data was not lost"

5) Set of Preconditions P: "orientation change possible and data inserted"
and
1) Test Goal: "UI elements available when screen rotates"
2) Set of Variables V: {}
3) Sequence of Actions A: [rotate screen]
4) Set of Checks C: "elements available before rotation are available after rotation"
5) Set of Preconditions P: "orientation change possible and a screen change detected"

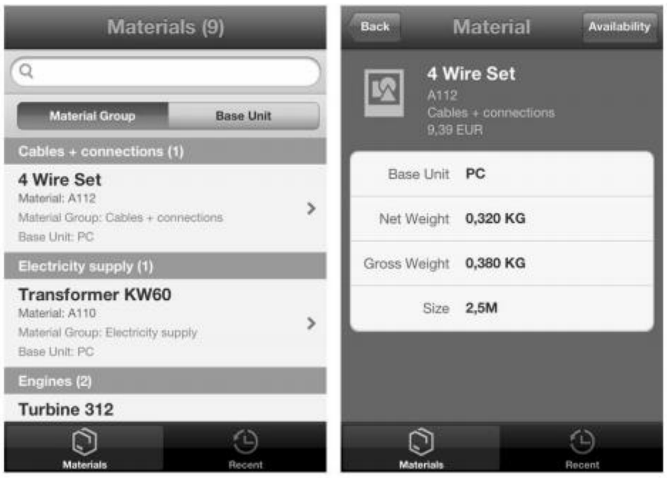
Consequences:
—Assurance that all the screens handle the change in orientation correctly
—In case of a false negative, it may provide the user a false sense of correctness to the user

Application Candidates:
—Writing an email (e.g., Gmail)
—Filling a search field (e.g., Youtube)

S17 - ARTICLE: TOWARDS USER INTERFACE PATTERNS FOR ERP APPLICATIONS ON SMARTPHONES

PATTERN: #72

4.4 Example Pattern: Nested Forms
Name: Nested Form Pattern
What: A nested navigation between a summarized listing of a business object and its attributes or a navigation between related business objects.
Use when: This pattern should be primarily used to establish a navigation link between a selection screen (listing, search, filter) and the screen to display the attributes of a selected business object. It should be also used to navigate between related business objects.
Why: Due to limited screen size of smartphones it is generally not possible to display the attributes of more than one business object. In order to solve this problem the Nested Form Pattern should be used.
How: To implement this pattern a navigation link between the summarized business object and its detail view must be implemented. The summarized business object should be presented in a selectable cell. It should have an indicator in form of an arrowhead to indicate the cell is selectable. There should be a navigation link to the detail view when selecting the cell. There should be also a button on the top of the details screen that gives the user the possibility to navigate backwards.
Examples: This example illustrates the nested forms pattern using the SAP materials availability application for the iPhone as example. The left screens illustrated a selection screen with different materials. By selecting one material its attributes are shown. It is possible to leave the attributes screen by selecting the back button.


Materials (9)
Material Group | Base Unit
Cables + connections (1)
4 Wire Set
Material: A112
Material Group: Cables + connections
Base Unit: PC
Electricity supply (1)
Transformer KW60
Material: A110
Material Group: Electricity supply
Base Unit: PC
Engines (2)
Turbine 312
Materials | Recent

Back | Material | Availability
4 Wire Set
A112
Cables + connections
9,39 EUR

Base Unit	PC
Net Weight	0,320 KG
Gross Weight	0,380 KG
Size	2,5M

Materials | Recent

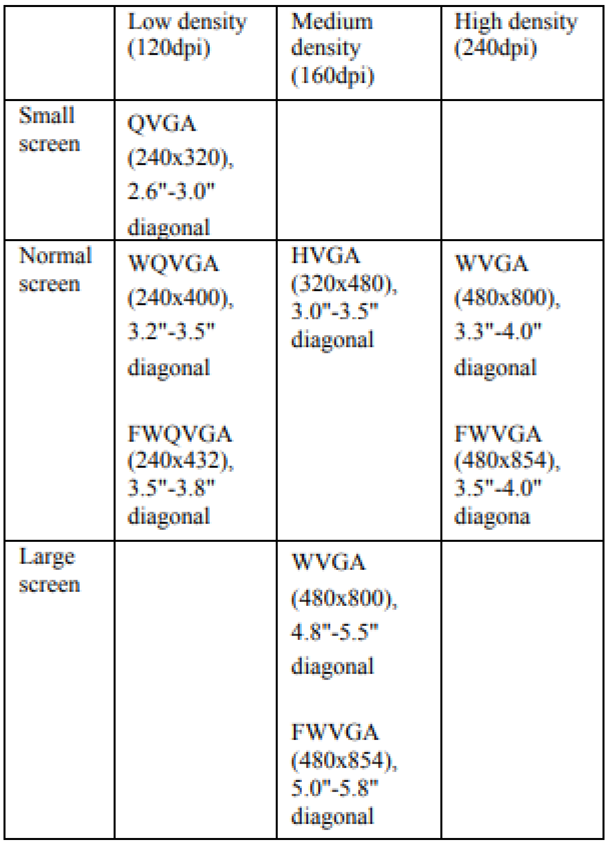
Fig. 4. Nested Forms Patterns illustrated within the SAP Materials Availability iPhone Application

S19 - ARTICLE: USABILITY-IMPROVING MOBILE APPLICATION DEVELOPMENT PATTERNS

PATTERN: #73
2.1 Client-side Multi-Screen Support ** Interaction Category/ies: User interface presentation, device independence Context: A mobile application that runs on a mobile device with local data storage and logic with no or infrequent access to a server. There are multiple mobile devices and, due to short innovation cycles, new mobile devices are launched every day. Scenario: A user wants to access the system using his own specific type of device, and he expects the application just downloaded to run on his device. It runs, but the content is not rendered properly. Another user downloads the mobile application but it does not start at all: the mobile platform on his mobile device filtered out the application because it could not display it correctly – his device's screen size is smaller than expected by the application developers. Problem: Mobile platforms run on different devices with different screen sizes and resolutions. In order to support all user interfaces, and thus usability, application developers have to optimize usage of their applications to be displayed on multiple screens. How can developers design their applications for correct display on many devices efficiently? Forces: • The heterogeneity of devices and the short innovation cycles in general make designing for all devices a time-consuming task. • If variants for each device are provided, the application needs too much hard disk storage. • Mobile platforms manage most of adaptation of an application to the current screen, but for a precise control it is necessary to create screen-specific resources. • Some scaling mechanisms for resources as images can be CPU-expensive. • Height and length of a UI-element (e.g., a button) are usually defined through pixel. A screen's density is defined as pixel per inch, so that a UI-element is displayed bigger on a screen with low density than on a screen with high density. Usage Data Recorder Unit Test Application Mobile Application Test Suite Client-side Multi-Screen Support Model-View Controller End User Test Good User Interface Presentation Testing Applications Evaluating Designs Main Goals: General Pattern: Test and Evaluate Detailed Patterns: Dialog Flow Manager Content Adaptation • Application developers have to carefully consider the possibilities of design, content and functionality within the small screen area. Screen layout could seem cluttered at a smaller device and offer to much unused space on a bigger device. Solution: Design for standard screen sizes and ratio and use a small number of predefined media classes that are scaled to the need of the specific device.

Scaling mechanisms provided by the platform can display applications properly on most devices, especially when the screen is the same size or larger. But it may need minor adjustments before they display on smaller screens, because of the reduced screen area there may be tradeoffs in design, content, and function. The application should therefore be designed for a standard screen of intermediate size and medium density. For example, Table 2 shows the range of typical screen sizes for mobile devices covered by HVGA (320 x 480) as standard screen.

Table 2. Range of screens supported by Android [DA2010]



	Low density (120dpi)	Medium density (160dpi)	High density (240dpi)
Small screen	QVGA (240x320), 2.6"-3.0" diagonal		
Normal screen	WQVGA (240x400), 3.2"-3.5" diagonal FWQVGA (240x432), 3.5"-3.8" diagonal	HVGA (320x480), 3.0"-3.5" diagonal	WVGA (480x800), 3.3"-4.0" diagonal FWVGA (480x854), 3.5"-4.0" diagona
Large screen		WVGA (480x800), 4.8"-5.5" diagonal FWVGA (480x854), 5.0"-5.8" diagonal	

The layout defined should be flexible, using relative units and positions, instead of absolute pixels. Also, layout containers should not demand fixed positions, and they should position their contents flexible. For the exact control of the application's design, specific resources should be prepared and provided. Using media variants for classes of devices (high, medium, small density classes) also provides a compromise between memory and screen optimization. Instead of using a default directory, directories named according to the density provide specific resources for classes of devices.

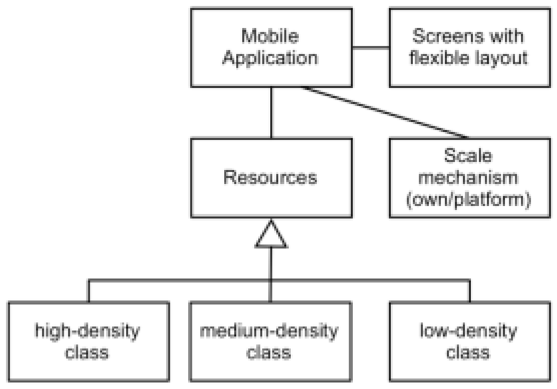


Figure 2. Components of the solution

Consequences:
[+] Programming effort is minimized when few variants are used.
[+] Regarding disc storage, providing only few variants for classes of devices needs less space than providing variants for all devices.
[+] If groups of screen-specific resources are designed, media is only scaled a little, which is nearly invisible to the user.
[+] Flexible layout improves results because then positions can be determined directly for the device.
[+/-] Some scaling mechanisms for resources as images can be CPU-expensive: If the mobile application creates a resource internally, e.g., a bitmap, and draws something on it, it creates bitmaps at the moment when it is drawn ("draw time"). Scaling at draw time is more CPU-expensive, but it uses less memory.
[+/-] Using the relative layout for a standard screen and scaling, the application will be layed out nicely for different devices. But if the screen sizes vary a lot, it makes sense to think about different versions of the application with different functionalities available.
[-] Design efforts are higher than providing just one variant, because several versions of media (images, video) have to be designed.
[-] Programming effort is higher than providing just one variant, but usability is improved. Usability consequences:
[+] Utility: The application will be displayed correctly when screen sizes and density are considered during the design.
[+] User experience: the application must be displayed correctly for a good user experience. As users expect this, there is no additional benefit for them. A correct presentation is a basic usability expectation that must be met. If it is absent, only then users will notice, and they will not like it.

Example:
The Android platform supports the developer in two ways: it selects the variants automatically based on name conventions, and it provides a scale mechanism. Developers could also make use of their own scale mechanism that adapts resources to match the display's density.

Related Patterns:
The parent pattern is MODEL VIEW CONTROLLER which also refers to alternative solutions for client/server systems. Alternatives are the DIALOG FLOW MANAGER

[6] and SERVER-SIDE CONTENT ADAPTATION [7] for server-side multi-channel access and the INTERMEDIATE CONTENT ADAPTATION [7] if a proxy is used. The application has to be tested, even during development. The patterns of the MOBILE APPLICATION TEST SUITE show how.

Known uses:
Android applications best practice [8]

S20 - ARTICLE: USER INTERFACE PATTERNS FOR MULTIMODAL INTERACTION

PATTERN: #74
4.1 Voice-Based Interaction Shortcut Context. The user has to select items from a large set. Consider selecting an action from a menu or selecting a list item from a drop-down chooser. In this case, this pattern is frequently used as an enhancement of Contextual Menu [10] or Dropdown Chooser [12]. Either the number of choices is quite large or screen size is scarse such that the items cannot be displayed all at once. The interaction device is supporting speech input. Problem. Which interaction style allows the user to quickly select the desired item without having to perform tedious navigation and scrolling actions? Forces – Selection via pointing requires almost no learning effort. But if the selection options are numerous and cannot be presented simultaneously on screen they have to be arranged into scrolling lists or hierarchical menu structures. Menu navigation and list scrolling may slow the user down. – Shortcut keys are a valid alternative for menu/command selection. However, it is difficult to assign shortcut characters that are easy to remember to a wide range of commands or menu options. Thus, some shortcut keys seem arbitrary and require a certain learning effort. – Typing meaningful words is easier to learn than shortcut or function keys. The users even might not need to type the whole word, because the list can be pre-filtered on the first few characters. However, typing is not always appropriate: Not all users are skilled typers. Typing with mobile devices is very awkward and slow and thus inappropriate for accelerating interaction. Solution. Design the system to support speech input to speed up interaction. The user should be able to select the desired action or data object by naming it. Especially frequent users to whom the commands, options, parameters and item names are well known can profit from speech input. When there are more than one appropriate alternative wording for the desired action, the designer should check which synonyms should be included into the speech recognition grammar. User tests, including tests with simulated speech functionality might be useful to elicit familiar wordings for some system functionality. Even when the designer has elicited familiar wordings, it is not guaranteed that especially first time and occasional users are able to anticipate them, too. For this, menus and lists should not disappear from the system. Voice-based Interaction Shortcut should instead coexist with these and be one of Multiple Ways of Input. The wordings used in the visual menus and drop-down lists should be identical to the respective speech commands such that the user's learning efforts can be reduced to a minimum.

After the user has uttered a speech command the system should update the visual display in the same way as it would do after menu or list selection. In addition, it might be appropriate to provide spoken system feedback, as it cannot be taken for sure that the user is looking at the screen when using speech. In this case the system adapts its output strategy according to the user behaviour (cf. the pattern Context Adaptation).
Speech recognisers return recognition results based on statistical calculations. Sometimes the actual user utterance does not match the best estimate by the recogniser but one among the five or ten best hypotheses. The pattern Mutimodal N-best Selection handles these cases by displaying the n-best list. The user can select from this (smaller) list via pointing.
With problematic (very large) sets of options, speech recognition is likely to fail. Using the pattern Spelling-based Hypothesis Reduction might help: When the user inputs a few letters, the list can be filtered and reduced. It might still be too large for being displayed as a whole in a selection list, but the speech recognition vocabulary could be now small enough for reliable speech recognition.
To avoid that background noise or private speech is interpreted as input, a push-to-talk button might be needed. This way, speech input is activated only while the user is holding down the button (cf. Tidwell's pattern Spring-loaded Mode [12]) or during a certain time window after (cf. Tidwell's pattern Oneoff Mode). When Voice-based Interaction Shortcut is used in combination with Context Menu [10], the right mouse button or the context menu key of the keyboard can serve as push-to-talk button.

Consequences
– Screen clutter and the need of menu navigation can be minimised by enabling speech input.
– There is no more need for the user to remember arbitrary action-keymappings or to obey to strict menu hierarchies.
– Typing can be reduced to a minimum.
– If the selection set is large, speech recognition performance may deteriorate, especially when there are similar sounding words. Even worse: some wordings might be ambiguous within the application context. If this cannot be avoided, the application has to provide clarification dialogs. In the worst case, all speed advantages might be lost.
– The users might actually not learn the exact wording of the items they probably would like to select in the future. That's why menus or dropdown selectors, which provide the visual context, should remain available as an alternative interaction strategy.

Rationale.
Users prefer speech input to input descriptive data, or to select objects among large or invisible sets [132, 133]. This counts especially when the list is large and the users know exactly the wording of the item they want to select.

Known Uses.
NoteBook is a multimodal notebook implemented on NeXT. The user can edit textual notes, and browse the created notes. Content editing is only supported via typing. Browsing, deleting and creating notes can be done via button clicks or voice input alternatively [19]. VoiceLauncher from Treoware enables speech input for Treos,

Centro, and Tungsten/T3 devices [134]. Microsoft Voice Command can be used to enable speech input for Windows Mobile based smartphones (such as HP's iPAQ 514 [135]) Using this software extension, the user can bring up the calendar or contact details in one interaction step [136].

Related Patterns.
This pattern can be used as an alternative or complement to van Welie's Continuous Filter, Context Menu [10], Tidwell's Autocompletion, Dropdown Chooser, Hub and Spoke etc. [12]. In the same way as Dropdown Choosers are used in Forms [11] Voice-based Interaction Shortcuts are used to implement Speech-enabled Forms.
To adapt the system feedback to the user behaviour, the pattern Context Adaptation might be used. In addition, Voice-based Interaction Shortcut should not be used as the only but as one of Multiple Ways of Input.
For error handling and avoidance this pattern can be combined with Multimodal N-best Selection and Spelling-based Hypothesis Reduction. To control recogniser activation either Tidwell's [12] pattern Spring-loaded Mode or One-off Mode can be used.

PATTERN: #75

4.2 Speech-Enabled Form

Context.
The user has to input structured data which can be mapped to some kind of form consisting of a set of atomic fields. Devices such as PDAs do not provide a keyboard for comfortable string input. In other situations the device may support keyboard input but the user has only one hand available for interacting with the system.

Problem.
How to simplify string input in form filling applications, especially in mobile scenarios and with small devices, when there is no comfortable keyboard?

Forces
– Selecting areas in 2D-space is accomplished very comfortably with a pointing device but string input via pointing (with on-screen keyboards) is very awkward.
– Values for some form items (academic degree, nationality etc.) are restricted and can be input by using drop-down choosers (comboboxes). But this may lead to screen clutter and additional navigation and scrolling.
– Speech recognition is very comfortable for selecting invisible items but the input of unconstrained text suffers from recognition errors. Speech recognition works more reliably with smaller input vocabularies.

Solution.
Let the user select the desired form field via pointing and dictate the content subsequently. The speech input vocabulary can be simplified. Only the vocabulary of the respective form field plus some generic commands need to be activated at the time and recognition errors can be avoided.
Whereever possible, determine acceptable values for each form field to reduce the input vocabulary to a context-specific subset. As alternative to speech input, support value selection via Dropdown Choosers and Autocompletion.

To avoid that the speech recogniser interprets background noise etc. as input, the speech recogniser should be activated only when the user is performing input. One possibility is, to activate the speech recogniser only as long as the user is holding down the pen or mouse button over the desired form field (cf. the pattern Spring-loaded Mode [12]). Another approach is to activate the recogniser during a certain time window after the user has selected the form field (cf. the pattern One-off Mode [12].

Consequences
– The user can combine pointing input via pen, touch-screen or mouse (for selecting an input field) and speech input to fill in this form field. Text input via on-screen keyboards can be avoided.
– Navigation and scrolling in drop-down lists can be avoided.
– Constraining the speech recognition vocabulary to context-specific data for the selected input field helps to avoid speech recognition errors.
– Speech recognition errors might occur anyway. In case of poor recognition performance, all speed advantages might be lost due to the need of error handling.

Rationale.
Users prefer speech input to input descriptive data, or to select objects among large or invisible sets [137, 132, 133]. According to Oviatt et al. [138], a structured (form-like) presentation of the interface reduces speech utterance length and the amount of information input per utterance. Thus, the variability of user utterances is reduced, which helps to make speech recognition more robust.

Known Uses.
Mobile Systems such as Microsoft's MiPad [120, 121, 122] and IBM's Personal Speech Assistant [123] are good examples.
In MiPad the user can create e-mail messages via Tap And Talk [121]. When the user selects the receiver field the speech recognition vocabulary is constrained to address book entries. If the user selects the subject or message field an unconstrained vocabulary is activated so that the user can input unconstrained text.
The multimodal facilities offered by X+V (XHTML and VoiceXML) and supported by the Opera Browser are heavily focused on this Speech-enabled Form paradigm [125, 126, 139, 140].

Related Patterns.
This pattern is a multimodal extension of Form [11] and the speech-based Form Filling [141, 142]. It is implemented using the pattern Voice-based Interaction Shortcut in the same way as Forms are implemented using patterns such as Dropdown Chooser and Autocompletion.
Tidwell's [12] patterns Spring-loaded Mode and One-off Mode can be used to control recogniser activation. For error handling, consider to use Multimodal N-Best-Selection and Spelling-based Hypothesis Reduction.

PATTERN: #76

4.3 Speech-Enabled Palette

Context.

Direct manipulation with graphic applications allows the user to edit visually presented objects directly. In order to manipulate these objects the user has to select sometimes special tools. In this context, the patterns Canvas plus Palette [12], Contextual Menu [10] and Mode Cursor [10] are used. This means that the user has to move the mouse out of the manipulation area (the canvas) in order to select the desired tool from the palette and then reenter the manipulation area in order to proceed the manipulation action. This might be very annoying, especially in drawing applications. Problem. How to enable the user to select tools from the palette without having to deplace the mouse between canvas and palette or the hand between mouse and keyboard?

Forces

– Both graphic manipulation tasks and selecting tools from a palette are accomplished very comfortably via pointing. But performing both subtasks alternately several times as is needed in design applications is very annoying and time-consuming.

– Using context menus which are opened on right clicks may reduce but not avoid totally lengthy cursor movements. Additionally the context menu (unless transparent) obscures the main manipulation area.

– Using the keyboard instead of the mouse for selecting commands may solve this problem in some cases. However, there might arise a new one: The user has to change his right hand between mouse and keyboard which is time consuming as well.

– Another solution would be to assign graphic manipulation tasks to the right hand, which controls the mouse, and action/menu selection tasks to the left hand, which remains on the keyboard and presses shortcut keys to select the necessary tools.2 But the user would have to remember awkward key mappings and possibly to look down to the keyboard to find the desired key.

Solution.

Allow the user to select tools using speech input. The palette itself should remain visible and allow the user to explore the interface. Each tool on the palette should have a meaningful name which is made obvious to the user e.g. via tooltip hints. These tool names should be used for the respective speech commands to reduce the learning effort for the user.

To control recogniser activation, Tidwell's [12] patterns Spring-loaded Mode or One-off Mode might be needed: Speech recognition should be activated only after the user has pressed or while the user holds down e.g. the right mouse button. This way, the system is prevented from interpreting background noise as speech input.

Consequences

– The user can select the desired tool via voice input without the need to replace the mouse cursor between tool palette and manipulation area.

– The screen and especially the main manipulation area is not obscured by popup windows or menus.

– The right hand can stay on the mouse and need not be replaced between keyboard and mouse.

– There is no need to remember awkward key mappings or to look down to the keyboard.

– As users are able to use the motor and vocal channels simultaneously, combining spoken commands and pointing speeds up interaction significantly.
– Speech recognition errors might occur. In case of poor recognition performance, some speed advantages might be lost due to the need of error handling.

Rationale.
Studies conducted by Ren et al. [143] have revealed that the combination of pointing devices such as pen or mouse with speech input is fruitful in both CAD systems and map-based interfaces. This way, interaction performance can be increased.
Positive results were also found in experiments with S-tgif [114]. This graphic design application offers the user keyboard and voice shortcuts for system commands as alternatives to pointing gestures.

Known Uses.
Graphic applications such as VoicePaint [113], S-tgif [114] and Speak'n'Sketch [115] are examples of systems which allow the user to select a tool from the palette via speech without removing the mouse cursor from the graphics manipulation area.

Related Patterns.
This pattern makes use of Canvas plus Palette [12], Mode Cursor [10] and Voice-based Interaction Shortcut.
To control recogniser activation, Tiwell's [12] pattern Spring-loaded Mode or One-off Mode can be used.
This pattern is an alternative to van Welie's Helping Hands [144].

PATTERN: #77

4.4 Gesture-Enhanced Speech Command

Context.
Some applications require the input of composed commands consisting of several parameters. Consider copying one object to another location which consists of inputting the command, the object to be selected and the destination. Consider setting up an email message: Input the command, input receivers of the message. Consider searching a location in a map-based application: Input the command, input area constraints (square C 5), input keywords (italian restaurants).

Problem.
How to enable the user to quickly input composed commands consisting of several parameters of different data types (spatial, conceptual, numerical data)?

Forces
– Complex commands can be input efficiently via typed or spoken command languages. But consindering some parameters such as file names, directory locations, positions on a city map, users rather remember where these are than how these are named internally. This might lead to typing errors. When speech interaction is used, users might not know how to pronounce these (partly cryptical) filenames or they would pronounce them in a way that cannot be handled by the speech recogniser.
– Inputting spatial parameters or selecting objects displayed on the screen is most

easily done via pointing. But inputting actions or textual parameters would lead to one or more additional interaction steps (button clicks, navigation in menus, scrolling through drop-down lists) or screen clutter.
– Early systems combined pointing gestures with typed natural language input. This way, the user was able to select objects directly via pointing and input commands and queries with the keyboard. However, the user has to move his hands repeatedly between keyboard and pointing devices which slows down interaction.

Solution.
Let the user interact via spoken language and provide pointing gestures to specify locations or interactive objects. Consider folllowing cases:
– The user selects a file, says “copy this file there” and selects the target location.
– The user draws a rectangle onto a map and says “are there any supermarkets?”
– The user clicks the button create mail and says “to Margret Smith”.
At a first glance, this seems to be simply an application of the pattern Voice based Interaction Shortcut. But in the case of Gesture-enhanced Speech Command the single interaction steps are seen in the context of a multimodal command language: Appropriate gestures and speech commands – which themselves might already consist of several parts – are combined into composed commands.
This requires the application of multimodal recognition technology and grammar formats for defining multimodal command languages such as MM-DCG [149] and prototyoing frameworks [150, 151, 152] as well as data collection and training frameworks for multimodal recognisers [153].
First time users need a way to explore the interface. That’s why Gesture enhanced Speech Command should not be a replacement for alternative interaction styles such as direct manipulation but rather be intergrated into them. 3 cf. Shoptalk [145, 146] and XTRA [147, 148]. 132 A. Ratzka

Consequences
– Commands and textual data can be input as text.
– Parameters such as file or directory names, locations on a map etc. can be input directly via pointing. There is no need to invent and remember cryptic names.
– Typing and recognition errors can be reduced as pointing replaces the input of the information parts that are hard to formulate as text.
– The user is able to utter simultaneously spoken language queries and pointing gestures: This way, inputting spatial parameters and selecting objects can be done using pointing devices. At the same time, the input of textual data can be done via spoken language. The user need not change his hands between different input devices.
– Screen clutter, i.e. the need of drop-down menus and popup dialogs which would obscure the potentially scarse screen space can be minimised when combining pointing with spoken natural language input. There is no need of additional buttons, dropdown menus, popup dialogs.
– Even if user input can be accelerated this way, recognition errors might compromise this advantage. Clarification and error handling dialogs have to be designed with care. This is especially true when “natural” speech input is supported. The system should find a stable way of error handling without discouraging the user to make use of efficient interaction styles in future.

Rationale.
Users prefer speech input to input descriptive data. Pointing devices are preferred for inputting spacial or sketch-based data [132, 133].
Combining direct manipulation with written natural language was shown to be plausible for some tasks [146, 154, 155]. A comparison of direct manipulation and the multimodal pen- and speech-based interface of QuickSet revealed multimodal interaction to be faster [63].

Known Uses.
This is one of the patterns found in the first multimodal systems. Bolt [156] describes a voice- and gesture-based interface which combines pointing with natural language. The title of Bolt's article outlines this pictorially: "Put that there".
HOME [157] provides a multimodal interface for home appliances, which takes into account special needs of elderly and disabled users and attempts to provide more natural communication consisting of speech and free gestures. The user can interact by using the touch screen, free gestures and spoken language. These input modalities can be combined:
User: Switch the light on.
System: Which light?
User: This lamp? <points to the lamp>
System: <switches the lamp on>
The map-based system GEORAL allows the user to input spoken multi-token natural language utterances while pointing to the relevant position on the map [158]. The user can select a place or an area (by pointing to a single point or encircling a zone) while asking questions such as Are there any beaches in this locality?, Where are the campsites? or Show me the castles in this zone.
Further examples are CUBRICON [145], QuickSet [107], MVIEWS [159], MATCH [110, 109], SmartKom [160, 161], ARCHIVUS [162], COMPASS2008 [124], RASA [163] and DAVE G [164].

Related Patterns.
This pattern is together with Location-sensitive Gesture a specialisation of Composed Command [11].
There are similarities to Voice-based Interaction Shortcut but Gesture enhanced Speech Command explicitly supports the semantic integration of gestures and speech. Furthermore Gesture-enhanced Speech Command combines speech commands and gestures that might themselves be already composed of several parts.

PATTERN: #78

4.5 Location-Sensitive Gesture

Context.
Some devices support pointing actions being performed with so-called direct pointing devices such as graphic tablets and pens. There is no keyboard available or the user has only one hand free for interaction and this hand controls the pointing device. Speech input is not supported or not appropriate due to context factors.

Problem.

How to enable the user to input easily and quickly standard commands (such as deleting, moving etc.) consisting of selecting items and performing actions on them?

Forces
– Commands can be input efficiently textually or via speech. Command parameters such as the selected files can be mapped onto text, too. But when neither keyboard nor speech input is available annoying on-screen keyboards have to be used.
– Selecting areas or objects displayed on the screen can be done easily via pointing. Inputting commands via pointing is possible, too. But this would lead to at least one additional interaction step (clicking on buttons, navigating through menus, scrolling etc.) and to cluttered screens.

Solution.
Let the user interact with the system as he would do with paper: draw meaningful symbols / pen gestures onto the object of interest – encircle items or cross them out, draw arrows and the like. A pattern recogniser transforms the gesture into a command. Onset and offset pen positions can be interpreted as positional parameters, i.e. they are used to relate the painted command gesture to the respective interaction object.
Pen gestures have to be thoroughly planned. One aspect is that different gestures should be designed in a way that they are not too similar and too likely to be confused by the system. It might be necessary to restrict the amount of available gestures to a few ones, to allow reliable recognition. Furthermore, toolkit-based design support might be needed [cf. 165].

Consequences
– There is no more need to find an awkward textual representation for items displayed on the screen. The user does not have to remember strange names, to type them in, or to try to pronounce them. He has simply to recognise the desired object displayed on the screen.
– There is no need to require keyboard or speech input.
– Screen clutter, scrolling and menu navigation can be avoided.
– The user can input a complex action in one simple step: Drawing a meaningful gesture onto an object displayed on the screen leads to opening, copying, deleting and the like. In usual graphic user interfaces, the user would have to select the object first and then select an action. This would require more steps.
– The user has to learn and remember a series of gestures. But some users might prefer commands instead. The system should provide a command interface as an alternative.
– Gesture recognition is not always fully reliable because of the high variability of human gestures. Pen gestures can be misinterpreted: The system can miss user input or recognise some, where there is none.
– Slips of pen can be misinterpreted as gestures and lead to unwanted actions.
– Although gestures should be designed to be meaningful, first time users will require some time of training before mastering the interfaces with all its advantages.
– When the system supports a wider range of (distinct) gestures, the user is forced to remain within a very narrow range of variations for each one of the standardised gestures. This increases the learning effort as well as the cognitive load during usage.

Rationale.
Gestures are easy to learn and additionally provide a means of terse and powerful interaction, because both position and movement patterns can be exploited to convey information [166].

Known Uses.
QuickSet [107, 63] allows the user to place military units via drawing military icons directly onto a map. The form of the icon designates the type of military base and the drawing position corresponds to the desired target location. Furthermore QuickSet supports specific editing gestures such as for crossing out (deleting) objects on the map.
Further examples can be found in TAPAGE [118] and the systems described by Fiore et al. [167] and Ou et al. [168].

Related Patterns.
This pattern is, besides Gesture-enhanced Speech Command, a specialisation of Composed Command [11].

Variant.
A variant of this pattern uses handwriting instead of gestures, as with handwriting you can simultaneously input spatial and linguistic information, too. However, to input spacial data more precisely, it is better to combine a separate pointing gesture with subsequent (and not parallel) handwriting. Such a combined input is supported by map-based systems such as PAVE [169] and MATCH [110, 109].

PATTERN: #79

5.1 Redundant Output

Context.
Communication channels might be unpredictably distorted or blocked due to bad lighting conditions, background noise, technical (network) problems or disabilities such as speech, motor or perception disorders.
Public systems that are supposed to be accessible for everybody should use this pattern, especially during the first interaction steps when it is not clear which interaction channels are appropriate for the current user.
The user’s attention may not be focused on the interaction device in situations when urgent information has to be conveyed by the system.

Problem.
How to assure information output when communication channels are distorted in an unforseeable way or the user is currently not paying attention to system output?

Forces
– The system can be configured or adapted to output information using modalities that are less affected by channel disorders. However, in some cases several interaction channels are distorted to some degree.
Examples are:
• Visually impaired or illiterate people who want to interact in loud environments.

• Deaf people that want to interact in bad lighting conditions or while moving around.
– Potential channel distortions might be circumvented by selecting alternative interaction channels. However, if the potentially distorted channel were otherwise the best candidate, abandoning this channel cannot be justified.
– The system can use those modalities that are most appropriate in the current environmental context. However, when the user's attention does not fit to the situation he might miss important notifications. When the system provides for instance only visual output due to a high background noise level but the user's visual attention is focused elsewhere, the system fails to notify the user.

Solution.
Combine several output channels in order to make use of redundancy. Information should be output both visually and acoustically and possibly even in a haptic way (e.g. using vibration) to raise the probability that it is perceived and can be understood by the user.

Consequences
– The use of several channels raises the probability that the user is able to perceive the information conveyed to him by the system. Visually impaired people in loud environments or deaf people in bad lighting conditions can process more data when output is presented redundantly to them.
– You don't need to abandon an output channel totally, only because it might be distorted. Visually impaired people might have problems reading a text and recognise each letter. After hearing the spoken variant and knowing what the text is about, the visual representation can be used as memory hook. The same might be true for dark environments or mobile scenarios, when it is difficult to fix visual attention to the text.
– It is more likely to attract the user's attention when information is output via several channels, e.g. both audio and sight, than when only one output modality is used.

Rationale.
Important and urgent information is more likely to be perceived by the user when several channels are used. Reaction times can be reduced by about 30% when redundant (visual, acoustic, haptic) signals are used [67]. Independent distortions of different channels rarely affect the same aspects of the content. Multi-channel feedback of written and spoken text has proven to be effective for elderly [40] and visually impaired users [41, 42, 43, 44].
Plosives ([p], [t], [k], [b], [d], [g]) sound similar and are likely to be confused when sound quality is low. At the same time these phones have distinctive lip shapes such as open lips (in the case of [g] and [k]) vs. initially closed lips (in the case of [b] and [p]). Lip shapes may differ for some similar sounding vowels, too.
In the context of language understanding, it has been proven that people understand spoken language better when they can look at the lips of their interlocutor while listening [68, 69, 70, 71, 72, 170, 15], especially in loud environments.

Known Uses.
Systems that display a talking head such as PPP [73], NUMACK [74], COMIC [75] and SmartKom ["Smartakus" 76] can exploit the advantages of audiovisual language understanding [13]. Mobile phones combine visual (blinking), auditory (ringing) and

haptic (vibrating) signals in order to notify the user about phone calls or incoming short text messages. Visual and auditory cues are combined to remind the user to plug in his laptop or mobile phone so that the batteries are charged.

Related Patterns.
Tidwell's Important Message [11] is a concretisation of this pattern. The focus of Tidwell's Important Message is on handling cases where the user's attention might not be directed to the interactive device. The pattern Redundant Output is more generic and addresses additionally comprehension problems caused by context factors such as bad light or background noise.

PATTERN: #80

5.2 Redundant Input

Context.
Communication channels might be unpredictably distorted due to bad lighting conditions, background noise, technical (network) problems or disabilities such as speech or motor disorders.

Problem.
How to assure input when communication channels are distorted in an unforseeable way?

Forces
– The system can be configured or adapted to recognise and interpret the modality that is less affected by channel disorders, but in some cases all available interaction channels are distorted to some degree.
• In loud environments users with motor disabilities or illiterate people have problems to interact with the system.
• In dark environments or in hands-free scenarios (e.g. while carrying a bag, wandering around, driving a car) people with speech disorders have problems to input data.
• In extreme conditions – e.g. while carrying out an exhausting primary task, both speech input and pen gestures are problematic.
– Interaction can be alleviated if modalities for implicit data input (gaze input, free gestures) or authentication (voice recognition, face recognition) are used. However, these interaction channels are error prone so that they cannot be applied directly.

Solution.
Combine several interaction channels in order to make use of redundancy. Input coming from several channels (visual: e.g. lip movements, auditory: e.g. the speech signal) should be interpreted in combination in order to reduce liability to errors.

Consequences
– The use of several channels raises the probability that the system is able to recognise and interpret the information input by the user in the desired way.
– Even if several channels are distorted the distortion rarely affects exactly the same pieces of information. Fusion mechanisms allow for reconstructing of at least some part of the input information.

– “Imperfect”, error prone interaction channels can be combined to mutually disambiguate recognition errors.

Rationale.
Independent distortions of different channels rarely affect the same aspects of the content. That’s why for instance audio-visual speech recognition, which combines acoustic and visual (lip movement analysis) signals, leads to better recognition performance than unimodal speech recognition in loud environments [cf. 13, p. 24 f.]:
Plosives ([p], [t], [k], [b], [d], [g]) sound similar and are likely to be confused when sound quality is low. At the same time these phones have distinctive lip shapes such as open lips (in the case of [g] and [k]) vs. initially closed lips (in the case of [b] and [p]). Lip shapes may differ for some similar sounding vowels, too. Channel distortions rarely affect both the recognition of a specific phoneme in the acoustic signal and of the corresponding viseme in the visual signal in the same way. Fusion algorithms allow to combine sound pieces of information from several channels such that some distorted parts can be reconstructed [13].
Studies conducted by Oviatt [171] revealed that an appropriate recogniser architecture that combines gesture and speech recognition can reduce recognition errors. This was shown for non-native speakers, in loud environments [172, 173, 174] and for cases where the users were carrying out an exhausting task in parallel [175].

Known Uses.
This pattern is manifested in very different application areas including data input (audio-visual speech recognition), scene analysis [176], person identification [177, 65, 66, 178, 179], affective computing and emotion recognition [180, 83, 100, 102, 101] and the like. Following modality combinations are used:
– virtual reality and speech [181],
– gaze direction and speech [182, 183, 184, 185],
– lip-reading in loud environments [64], e.g. to filter out simultaneous speakers [186],
– speech and gesture [187, 188],
– voice, ink and touchtone [189],
– biometrics, voice and face to identify persons [65, 66].

Related Patterns.
Spelling-based Hypothesis Reduction as well as Multimodal N-best Selection are refinements of this pattern.

PATTERN: #81

5.3 Multimodal N-best Selection

Context.
This pattern can be used in multimodal systems that offer speech input of unconstrained text or speech-based selection of items from very large sets such as timetable or navigation systems.

Problem.
Speech recognition is based on statistical processes. The recognition of input

phrases results in a set of several recognition hypotheses. Usually the recogniser returns the best match as “the result”. It is frequently the case that the original user input does not match this “best guess” but is included in the list of n best hypotheses returned by the recogniser.

Forces
– When a speech input attempt fails, the user can be prompted to repeat or to switch to another interaction modality. But it is inefficient to throw away input data which has failed the goal just by a hair.
– Playing back the n best recognition hypotheses, prompting for (spoken) selection and reducing the speech recognition vocabulary to this reduced list of n items can correct the error in just one further interaction step. However, items contained in the n-best list are likely to have some acoustic similarity so that they might be mixed up repeatedly by the recogniser.
– Playing back just the item on top of the n best recognition hypotheses and prompting for accepting or rejecting may resolve this problem in a few steps, but if the desired item is only the fifth (sixth, seventh ...) best recognition hypothesis five (six, seven ...) error correction steps are needed.

Solution.
Provide the user a means of selecting the correct result from a set of recognition hypotheses via pointing or key presses.
The number of alternatives should be restricted to a manageable size so that the system remains easy to control. In order to satisfy cases where the desired item cannot be found in the n-best list there has to be a way of explicitly leaving the list selection dialog and to start over the input attempt.
Disabled users or users in restricted environments might want to use speech input as well for this. In order to circumvent recognition problems that arise because the vocabulary contains similar sounding words, the list should be presented in a standardised way and contain line numbers in addition to the item wordings. The user should then be encouraged to speak the line number plus the item name so that the system can distinguish better between alternative hypotheses.

Consequences
– Imperfect recognition results are not thrown away but reused in subsequent interaction steps.
– Instead of re-speaking the misrecognised item, the user can point to the item displayed in the list. By changing the input strategy, recurring recognition problems are avoided.
– Frequently, instead of endless error correction loops, this pattern helps to correct recognition errors in just one additional interaction step.

Rationale.
Suhm et al. [190] point out that re-speaking the same word or phrase after recognition failure is not the most promising form of error recovery in interactive systems, although this might seem to the user to be the most obvious strategy. Changing the input modality to list selection seems to be more promising [191, 192].

Known Uses.

Directory assistance, timetable information systems, speech based driver assistance systems, office applications such as MedSpeak [193] and Human-Centric Word Processor [194, 195] support n-best selection.

Related Patterns.
This pattern can be used in conjunction with Speech Enabled Form, Drop-down Chooser [12] and Autocompletion [12] to alleviate error handling in speech-enhanced input forms. This pattern and Spelling-based Hypothesis Reduction are refinements of Redundant Input. Multimodal N-best Selection may display the n best recognition hypotheses in a Drop-down Chooser [12].

Variant.
The solution described in this pattern is not restricted to multimodal interaction in a narrower sense. Speech-only systems provide similar speech based approaches of selecting the desired item from a list of hypotheses.
If n-best selection via speech is supported, it is important to offer input modifications to avoid repeated errors. The user should be given the possibility to select the desired option via speaking the line number or re-speaking the item plus some distinctive features such as the first letter(s).
In these cases, the user has to be prompted in a way that reveals alternative selection strategies apart from simple re-speaking, e.g.: “Did you mean one Jonathan Smith, two John Griffith, three Joseph Reddish or new input”.
While the list is being read out the user should have the possibility to interrupt the playback. In full-duplex systems with cleanly separated channels for audio input and output, barge-in can be used. That means that system playback is stopped when the user starts speaking. In half-duplex systems, the user should be able to interrupt system playback using a push-to-talk button. In the cases of both barge-in and push-to-talk, the time information when speech output was interrupted can be used as a further information source to identify the selected item [196, 197].

PATTERN: #82

5.4 Spelling-Based Hypothesis Reduction

Context.
Examples where this pattern can be used are systems that offer speech input of unconstrained text or speech-based selection of items from very large sets such as timetable or navigation systems. Errors are particularly likely in cases, when the user has to select from lists with similar sounding words or words with inconsistent pronunciation such as foreign names.

Problem.
Large recognition vocabularies entail error-prone speech recognition, especially when many similar sounding words have to be distinguished or a wider range of pronunciation variants has to be supported.

Forces
– Speech input can be used for selecting items from a list that cannot be displayed completely on a small screen, but if the list is too large for speech recognition or includes several similar sounding or problematic words, speech recognition is likely

to fail. Problem areas are names in directory assistance systems as well as album and song titles in entertainment systems.
– In the case of recognition failures, the user can switch to text input (typing), but in some applications such as driver assistance systems only unhandy (if any) string input facilities are available.
– Typing the first letters can reduce the size of the selection list so that pointing is possible again, but in some applications such as navigation systems, a lot of characters need to be input (using an impractical input device) before the list can reduced to a size feasible for display and list selection.
– Some speech recognisers, especially those for small devices, have only restricted resources such that only small vocabularies e.g. for number recognition or for recognising letters of the alphabet are supported. But operating applications this way provides only little added value in contrast to using a small keypad, especially since some letters sound similar and are likely to be confused by the recogniser.

Solution.
Offer the user to type the first few letters of the word or list item before speaking it. Use this substring to filter the list of selectable alternatives, i.e. to reduce the size of the active speech recognition vocabulary. With this smaller vocabulary, speech recognition is more reliable.
Alternatively, record speech input attempts and keep this recording available for a disambiguation step. When recognition fails, the user should be encouraged to input the first (few) character(s) e.g. via typing. Now the list of alternatives is filtered and the size of the active vocabulary is reduced. The speech recogniser can then be re-run with the previous recording and the smaller vocabulary.

Consequences
– By inputting quite few letters, the set of alternatives can be reduced to a size that
– although still unsuited for pointing-based list selection
– allows robust speech recognition.
– The user needs to input only a few letters. This is important in cases where typing is inconvenient.
– There is no need to navigate and scroll through lists.
– Recognition of names, song titles and other problematic words becomes more robust.
– This technique of reducing speech recognition vocabularies simplifies speech recognition on platforms with limited resources.

Rationale.
With an increasing amount of words or phrases activated, speech recognition accuracy decreases. By reducing the active speech recognition vocabulary the recognition performance can be improved significantly.

Known Uses.
Marx and Schmandt [127] describe the messaging system Chatter which allows the user to input contact names via voice spelling, touch-tone spelling and speech-based naming in a combined fashion. The prototype of the multimodal driver assistance system CarMMI [58] allows the user to input the first letters using a rotary knob mounted on the centre console, so that the speech recognition vocabulary can

be reduced. Similar examples can be found in Suhm et al. [190] and Tan et al. [183].

Related Patterns.
This pattern can be used in conjunction with Speech enabled Form, Drop-down Chooser [12] and Autocompletion [12] to alleviate error handling in speech-enabled input forms.
This pattern and Multimodal N-best Selection are refinements of Redundant Input. Spelling-based Hypothesis Reduction uses (or is) some kind of variant of Continuous Filter [10]. Instead of filtering the items of a selection list, the recognition vocabulary is reduced according to the letters input by the user.

Variant
Some systems allow the user to dictate (i.e. speak) characters (voice spelling) to reduce the active recognition vocabulary [127]. The recognised spellings are expanded by using a so-called confusability matrix to avoid that misrecognized characters reduce the list of input alternatives too much.
In addition, phonetic alphabets can be used to reduce recognition failures. But this is only feasible if the target user group is expected to be proficient in that phonetic alphabet.
When a phonetic alphabet is supported the user is not proficient in, this might result in spontaneous wordings such as Motel instead of Mike or October instead of Oscar. This way, recognition errors might even increase.

PATTERN: #83

6.1 Multiple Ways of Input

Context.
Context factors are not always predictable. This holds especially for mobile interaction in changing environments or public information kiosks that should be usable for quite different user groups.

Problem.
How can input modalities be adapted to the context of use without burdening the user with additional configuration tasks?

Forces
– The user should be able to interact with the system using preferred and task appropriate interaction styles. However, disabilities or changing environmental conditions such as lighting or background noise may affect the usability and robustness of task-optimised interaction modalities.
– If background noise is low, speech input and output provide a valid interaction style. However, in public environments, bystanders might feel annoyed by persons conversing with an interactive system.
– Environmental factors can be controlled by using special installations:
• Specially mounted lamps help to overcome bad lighting conditions.
• Enclosed areas (e.g. phone booths, kiosks etc.) reduce background noise and help to assure privacy. But these measures are not viable in every case such as mobile interaction.
– Users might differ according to preferences, language, reading, typing skills etc.

The system can provide alternative interaction styles and adapt to the user. However, the system is not able to predict precisely which input modalities are most likely to be selected by the user in the current situation.

Solution.
Enable the user to trigger each system function by using one of several alternative interaction modalities, be it speech, typing or pointing. The alternative modalities should be active and ready to use all the time without further configuration needs. The system has to be designed in a way, that the user can choose alternative modalities wherever sensible, even for single interaction steps of a more complex task. Labels, help messages, prompts, console and speech commands should share a uniform wording in order to minimise learning efforts and confusion for the user, when he wants to sidestep to another modalitiy.

Consequences
– By providing several alternative interaction styles, the system can be used by physically disabled or illiterate people.
– As the user can choose among several modalities which are each differently affected by environmental factors, the system can be used in varying environmental conditions.
• In loud or public environments the users can simply sidestep to pointing or text input.
• When background noise is low users can opt for speech interaction.
– Expert users can estimate whether speech input or pointing gestures are possible or desirable at the current moment.
– It is up to the users to select their preferred interaction style. They do not need to wait for the system to automatically adapt. Instead, they gain self confidence since they are controlling the system which leads to higher user satisfaction.
– First time users might not know which interaction styles are available at all. The system must provide effective help and prompting strategies that reveal alternative interaction modalities.
– The more flexible a system is, the more planning, testing and reviewing is needed during design since the number of error sources increases rapidly with system complexity.

Rationale.
According to user characteristics, preferences, environment and situation, different interaction modalities are preferable [133, 6, 7]. Users can judge better than the system, which interaction modality and style is appropriate, e.g. whether background noise or bad lighting impedes the use of speech or graphics, whether surrounding people might feel annoyed because of spoken interaction or whether private information is being conveyed.
Ibrahim and Johansson [198] have shown for their multimodal TV guide, that users prefer, when they can choose to use direct manipulation and speech either unaccompanied or combined in order to adapt to the current context of use. Users learn to estimate and select the most context-appropriate modality. After recognition errors, users tend to switch the interaction modality [133, 199, 200, 171].

Known Uses.

SmartKom [161] allows the user to interact either via pen or speech. Mobile systems such as Microsoft's MiPad [120, 121, 122] and IBM's Personal Speech Assistant [123] are good examples for systems that allow users to flexibly select input modalities.
The same holds for driver assistance systems such as the ones described by Neuss [58] and Pieraccini et al. [201]. Further examples of this pattern can be found in the interactive TV guide by Ibrahim and Johansson [198], and MICASSEM [202].

Related Patterns.
This pattern is on the one hand an alternative to Global Channel Configuration and Context Adaptation. On the other hand these patterns can be combined.
Whereas Global Channel Configuration requires an additional interaction step to select the input / output profile of the system, Multiple Ways of Input keeps all input modalities activated so that they can be used without additional configuration steps. In contrast to Global Channel Configuration, this pattern is restricted to the adaptation of user input.
That's why this pattern should be used together with Context Adaptation. This way, system output can be adapted according to the user's input behaviour. In contrast to system-driven Context Adaptation, Multiple Ways of Input offers, similarly to Global Channel Configuration, user-driven system adaptation.

PATTERN: #84

6.2 Context Adaptation

Context.
Examples for this pattern can be found in interactive devices that support different alternative and complementary interaction channels such as audio output and input, typing and graphic manipulation.

Problem.
How can interaction (input and output) be adapted to the current situation, environment and user without the user having to perform additional interaction steps?

Forces
– Redundant output via several channels can assure information perception by users. However, superfluous spoken output might disrupt or slow down the user's secondary task or annoy third persons.
– Alternatively, one could let the user configure system input and output according to the current context of use. But, as long as sufficient information is available to the system, additional configuration steps should be avoided.
– Letting the user configure the system himself seems not to be a problem at first. But the user might not be able to remember the current configuration state of the system and to reconfigure the system at each situation change.

Solution.
The system should analyse as much assured context information as available to setup system configuration autonomously. One information source that can be used is the interaction history:

– If the user interacts via speech, it is not clear where the user looks. In this case speech output should complement display updates.
– If the user does not want to annoy surrounding people, he will avoid speech interaction.
– If the user is currently typing or using a pointing device
– might be in order not to annoy surrounding people
– he is usually looking at the display such that speech output is superfluous.
Other aspects of system state can be exploited for adaptation, too: A driver assistance system should disable touch screen input while the user is driving – the driver should keep his hands on the steering wheel. Smartphones should disable touch screen input during a telephone call – the ear of the user should not lead to unwanted actions.
In addition to these situational aspects, the system can adapt system output according to the user expertise level. For first time users, more verbose speech output prompts are necessary. When the user already has proven to use a system function successfully several times, shorter prompts are sufficient [203, 197].
Systems that can be used by several users need a user identification mechanism (e.g. a log-in facility) to keep a specific user model available for future interaction sessions with this user.
The system should not be prematurely adapted based on statistical assumptions with low confidence scores, but only after enough context information has been analysed. In order to keep adaptation predictable for the user, it should take place in a transparent way that can be undone easily.

Consequences
– Information output can be restricted according to the context of use.
– The user does not need to reconfigure the system repeatedly where this can be done by the system itself.
– The user need not remember configuration states and reconfigure the system at each situation change.
– Automatic adaptation can fail or be unappropriate. That's why the user should always have the possibility to carry over control and perform interaction configuration himself.

Rationale.
So-called plastic user interfaces adapt automatically to the context of use. For this, they analyse comprehensive context information [47, 50, 48, 49] and require an appropriate runtime architecture [54, 4].

Known Uses.
SmartKom [161] allows the user to interact either via pen or speech. System feedback is adapted in a way that fits to the user's attention: The TV-guide subsystem of SmartKom presents the TV-program usually as a listing. But when the user query has been done by voice input spoken feedback is played back. When the user is watching TV, the system presents the program list verbally, too, because it supposes that the visual channel is occupied. Driver assistance system such as the ones decribed by Neuss [58] and Pieraccini et al. [201] offer the user to interact using speech or manual input devices. System output is adapted accordingly. Smartphones disable touch screen input during telephone calls.

Related Patterns.
This pattern is an alternative to Global Channel Configuration and Multiple Ways of Input but may be used complementarily to those ones. In contrast to those user-driven adaptation strategies, Context Adaptation implements a system-initiated adaptation strategy. This pattern can be used as complement to Multiple Ways of Input in order to adapt the system output according to the user input. In addition to Context Adaptation, Global Channel Configuration should be supported to give the user control over the system.

PATTERN: #85

6.3 Global Channel Configuration

Context.
Interactive devices that offer several alternative and complementary interaction channels such as audio input and output, typing and graphic manipulation have to be adapted to the context of use.

Problem.
How can interaction (input and output) be adapted to the current context of use, while giving the user control over the system without burdening him with too much configuration tasks?

Forces
– One can keep several input channels active and leave it up to the user to select the interaction modality that is most appropriate for the current context. However, the system might try to interpret input from unused distorted channels. This might lead to situations where the system misinterprets background noise as input.
– Redundant output via several channels can assure information perception by users. However, in public environments, bystanders might feel annoyed by persons conversing with an interactive system. In the same way, it is not desirable, that private data be read out loudly by the system.
– The system may analyse interaction behaviour, lighting conditions, background noise, movement and position changes and adapt to the user and context of use. However, the system does not find out as fast as the user does which interaction modalities are most appropriate in the current context of use or for the current user.
– Even if automatic adaptation works quite well, many users will prefer being able to control the system.

Solution.
Provide several interaction profiles with input and output channel configurations tailored to each context of use. Enable the user to select the interaction profile of the system (speech input, audio output, verbosity level of speech output etc. or, for example, the notification profile of a mobile phone such as ringing, vibrating, mute) with only one additional interaction step.
For this, display for instance always on top buttons or widgets with self explaining icons or provide physical push-to-talk and mute buttons. Systems that are used by several users require an identification (log-in) mechanism to keep the profile lastly selected by a specific user available for future interaction sessions.

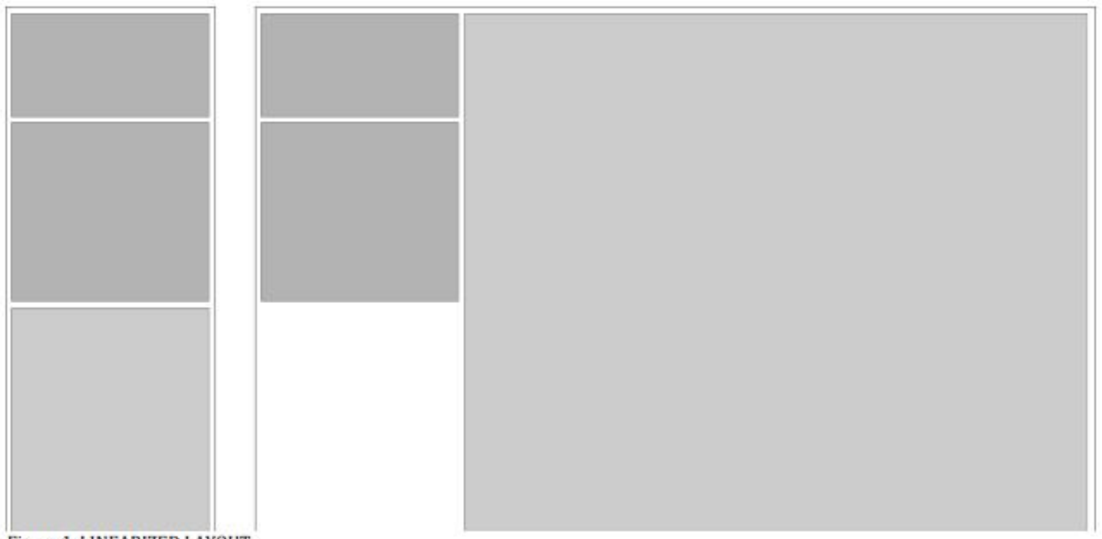
Consequences
– The user can quickly react to context changes and reconfigure the system accordingly in one interaction step:
• Input channels such as speech input can be deactivated when necessary (in loud environments) by simply clicking the respective button.
• In order not to disturb bystanders in public environments, the user can deactivate audio output with one click.
– Users feel better when exercising control over the system instead of being delivered to non-predictive system adaptation. – Users have to do at least one additional interaction step, which might be annoying anyway.
– Users might by fault select an inappropriate interaction profile.
– The users might not be able to rembember the current configuration state of the system and forget to reconfigure it at each situation change.

Rationale.
According to user preferences, abilities, environment and situation, different interaction modalites are appropriate and preferred [133, 6, 7]. Users learn to estimate and select the most context-appropriate modality. After recognition errors, users tend to switch the interaction modality [133, 199, 200, 171].

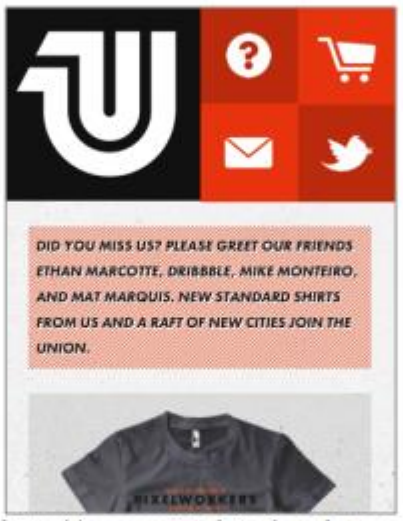
Known Uses.
The multimodal map-based system SmartKom Mobile covers the usage scenarios of pedestrian and automotive navigation as well as map-based queries [108] and allows the user to switch between several interaction modes [cf. 204, p. 59]:
– Default: All input and output modalities are supported.
– Listener : Speech and graphics are supported for output, for input only pen gestures are possible.
– Silent: Only graphics for output and pen gestures for input are supported.
– Speech Only: Only speech interaction is active.
Some mobile phones – e.g. Nokia E71 – offer the user to select profiles such as office or home. In addition to different startup screens, these profiles can be set to an appropriate context-dependent notification mode (ringing, vibrating). Some desktop applications, operating system environments, and multimedia applications provide an audio icon in the system tray for setting the system’s audio characteristics. With some restrictions, these can be seen as examples for this pattern, too.

Related Patterns.
This pattern is an alternative to Multiple Ways of Input and Context Adaptation but can be used in combination with them, too. In contrast to Multiple Ways of Input, which offers the user to select among several alternative input modalities without having to perform additional configuration actions, this pattern requires one additional interaction step. In addition to Multiple Ways of Input this pattern also addresses the adaptation of system output.

S21 - ARTICLE: WEB DESIGN PATTERNS FOR MOBILE DEVICES

PATTERN: #86
4.1 Linearized Layout  Figure 1. LINEARIZED LAYOUT. Problem On smaller devices with narrow screens, more complex designs with multiple columns do not fit perfectly. In most cases you end up with a scaled-down page that is unreadable. Solution Linearize the content, by stacking all blocks of information on top of each other, spanning the width of the screen. *** Layouts with multiple columns are a common pattern on web design. However, they do not work properly on devices with narrow viewports. When this type of websites are accessed on these devices, browsers will zoom out the page until it fits on the width of the screen; or they will add horizontal scrolling. Both alternatives are far from satisfying. Column layouts should be optimized for mobile viewing. Therefore, you should design your mobile site by assembling the page vertically and by stacking each page section on top of each other, which is already the default behavior of a page when no style is defined. Because we cannot foresee the size and resolution of the devices that will access our website, blocks with fixed widths are not a good practice. The LINEARIZED LAYOUT should have a fluid layout that adapts to the width of the browser whatever is its effective size. This pattern is also a requirement for many of the other patterns proposed, particularly VERTICAL LIST, i.e., if you want to use it you will invariably need to linearize the content.

Examples


DID YOU MISS US? PLEASE GREET OUR FRIENDS ETHAN MARCOTTE, DRIBBBLE, MIKE MONTEIRO, AND MAT MARQUIS. NEW STANDARD SHIRTS FROM US AND A RAFT OF NEW CITIES JOIN THE UNION.

http://www.unitedpixelworkers.com

http://colly.com
http://ucsd.edu
http://foodsense.is
http://cironline.org

Related Patterns

- GRID LAYOUT
- VERTICAL LIST
- Vertical Stack in Designing Interfaces (Tidwell 2011)

PATTERN: #87

4.2 Grid Layout

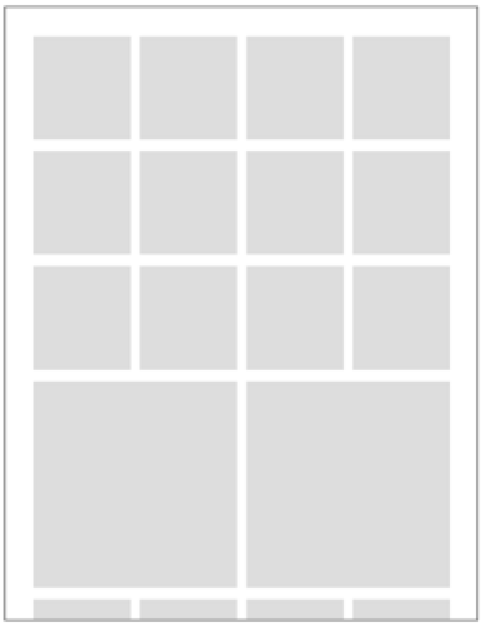


Figure 2. GRID LAYOUT.

Problem
Although a LINEARIZED LAYOUT works in most cases, sometimes you have content that does not need to,
or should not, fill the entire width of screen.
Solution
Arrange elements in a matrix of one or more rows.
***
While not as recurrent as the LINEARIZED LAYOUT, a GRID LAYOUT can be quite effective with some types
of content. It is typically used with image content such as photos, illustrations or icons. The most common case
is to present a gallery of images, but you can still use it with text, as long you keep it to a few words. You can
still use it with text, as long you keep it to a few words. Long paragraphs of text can become almost unreadable
even if your grid only has two columns.
A GRID LAYOUT can have as many divisions as needed; however, if the grid blocks work as buttons they
should be large enough and adequately spaced so they are easily triggered — use a TOUCH FRIENDLY BUTTON
or an ICEBERG TIP on those cases. Although in most examples blocks have the same height and width, they can
still assume irregular forms. That is, blocks can span multiple columns or rows — Figure 3.

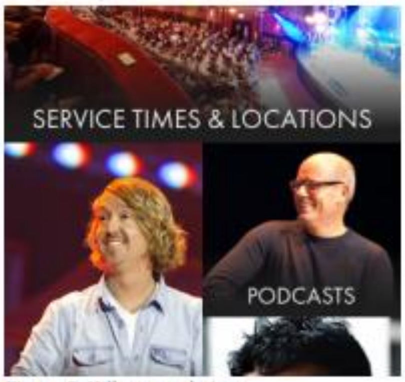


Figure 3. Hillsong website.

Because with this pattern you are displaying multiple items side by side, in some cases it can be a more
efficient use of the vertical space than a LINEARIZED LAYOUT. For example, if you have a VERTICAL LIST
where each item is composed by only one word, you can save space by displaying several items on the same
line.

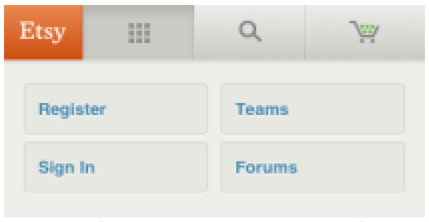


Figure 4. The TOGGLE MENU on Etsy website reveals a menu that list items on a grid.

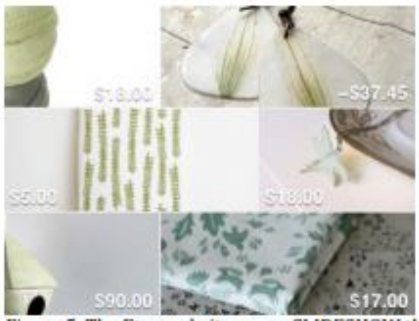
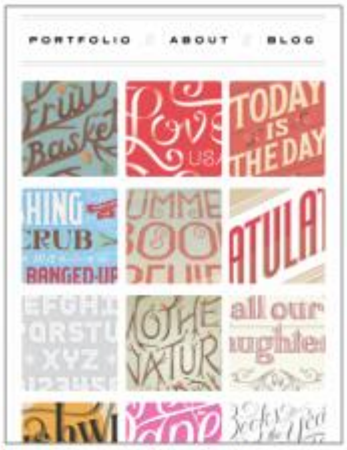


Figure 5. The Etsy website uses a SLIDESHOW that presents on the same slide multiple images on grid, each one working as a link.

The GRID LAYOUT can be used as a more structural pattern, or combined with other patterns to extend their capabilities. For example it can be used with a TOGGLE MENU to increase the number of items visible on the screen — Figure 4; or with a SLIDESHOW to increase the number of items by slide — Figure 5.
Examples



http://jessicahische.is,

http://m.microsoft.com
http://2012.dconstruct.org

Related Patterns

– LINEARIZED LAYOUT

– TOGGLE MENU

– SLIDESHOW

– Grid in Designing Mobile Interfaces (Hoober and Berkman 2011)

PATTERN: #88

4.3 Vertical List

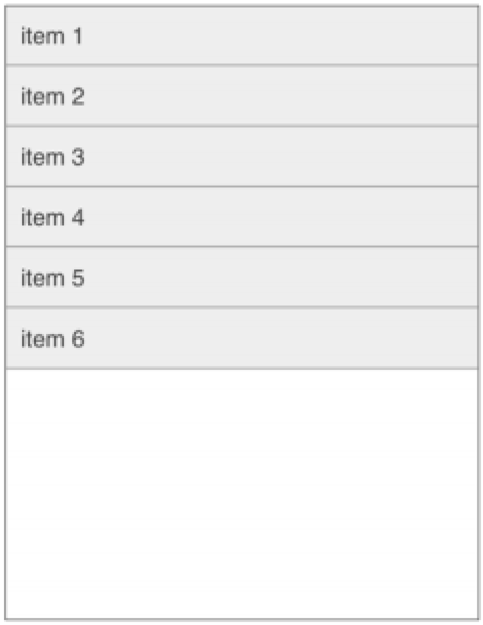


Figure 6. VERTICAL LIST

Problem A LINEARIZED LAYOUT is a quite common layout that gives much emphasis to the vertical orientation. If you have information that is organized as a list, you need to have a method to efficiently display that content. Solution Display these chunks of information stacked vertically, spanning all the width of the screen and graphically dividing each item. *** Along with the LINEARIZED LAYOUT this is perhaps the most common pattern that you will encounter when designing for mobile. Given that the width of the devices is not generous, you will invariably need to rely on the vertical space to accommodate most of the content. For most of these situations, this pattern or one of its variations will be used, such as: INFINITE LIST, THUMBNAIL LIST or EXPANDING LIST. Therefore, most of the recommendations given for this pattern also apply to the related patterns. Generally, there are two different alternatives to the implementation of this pattern: one that only displays information; another where each list item works as a link — which sometimes can be a LINEARIZED MENU. When the list items work as links you should optimize them for touch, you can use a TOUCH FRIENDLY TARGET for this. Each item ideally should only contain a link. Even if the item contains text with different hierarchies — Figure 7 —, you can wrap all those elements on an anchor tag rather than using only the title as a link. The result is similar to what can be attained with an ICEBERG TIP.

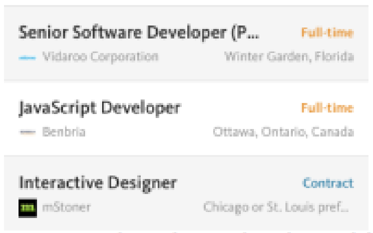
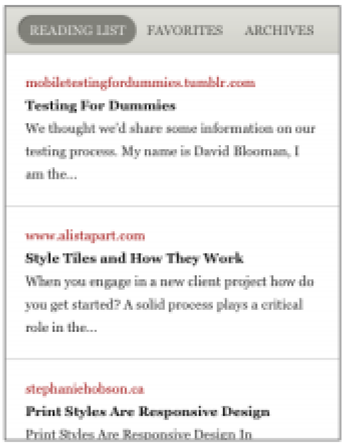

Senior Software Developer (P... Full-time
Vidaroo Corporation Winter Garden, Florida
JavaScript Developer Full-time
Benbria Ottawa, Ontario, Canada
Interactive Designer Contract
mStoner Chicago or St. Louis pref...

Figure 7. On the *Authentic Jobs* website, while each list item contains a diverse range of information with different hierarchies, the entire rectangle works as a link.

For this pattern to work effectively, each item should be clearly separated. You can use any design that visibly distinguishes items, such as: a horizontal line spanning the width of the screen; alternative color rows; typographic hierarchy; or just white space.

Examples



mobiletestingfordummies.tumblr.com
Testing For Dummies
We thought we'd share some information on our testing process. My name is David Blooman, I am the...
www.alistapart.com
Style Tiles and How They Work
When you engage in a new client project how do you get started? A solid process plays a critical role in the...
stephaniehobson.ca
Print Styles Are Responsive Design
Print Styles Are Responsive Design In

http://readability.com/mobile

http://cognition.happycog.com/topics
http://bostonglobe.com

Related Patterns

– INFINITE LIST

– THUMBNAIL LIST
– EXPANDING LIST
– LINEARIZED MENU
– Vertical List in Designing Mobile Interfaces (Hoober and Berkman 2011)
– List Menu in Mobile Design Pattern Gallery (Neil 2012)

PATTERN: #89

4.4 Infinite List

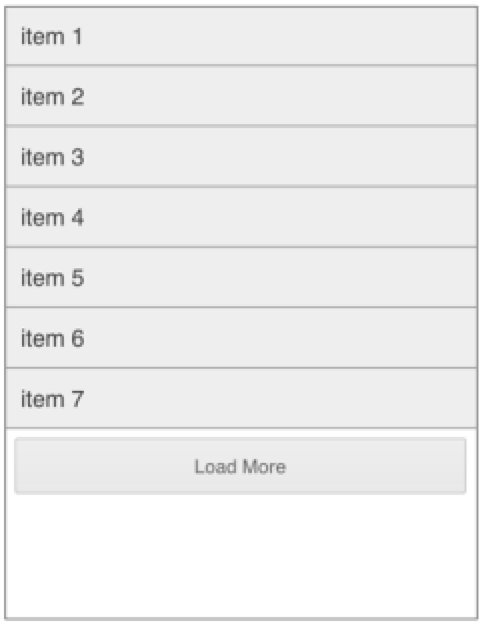
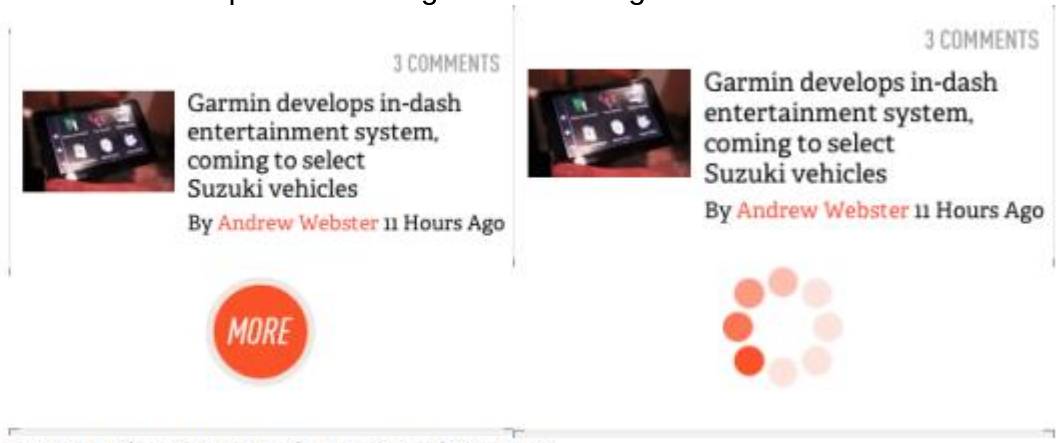


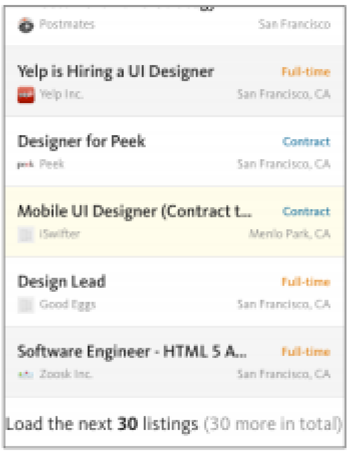
Figure 8. LINEARIZED MENU

Problem A very extensive VERTICAL LIST can become quite heavy on mobile devices. It would not be advisable to load and display all information in the list at once. Solution Design a normal VERTICAL LIST or any of its variations but only fetch the beginning of the list, loading the rest as the user scrolls through the page. *** This pattern is based on a homonym pattern from Designing Mobile interfaces (Hoober and Berkman 2011). It is very similar to the VERTICAL LIST with the main difference that only a portion of the list is initially loaded. It is most adapted for displaying a list of data with an uncountable number of items. For example, when loading all the news of a site in the same page we can be dealing with hundreds of items. It would not be a good idea to load all that information at once. That would be extremely heavy in terms of download size and load time, and probably, users would not need all that information from the beginning. Thus, we can start by loading only a few items, and only load subsequent items when the user provides some signal that he wants to keep browsing through the page. The number of items to load on each action varies, but it will depend on the length and download size of the content. This pattern takes advantage of methods for asynchronously loading additional content without the need to refresh the page: new items are just appended to the interface following the previous ones. Because the browser does not need to reload the page, we can get a perceivably faster response. It can be used as an alternative to a common pagination, which

normally implies more touches and a refresh on each load. Since there is not any refresh, users never lose the context of where they are at the list. There are two main alternatives that can be used for its implementation: explicit and implicit loading. On the first, the content is loaded with a direct action of the user; on the second, the list is loaded automatically as the user reaches its end.
When designing a list in which the user has to explicitly load new content, place at the bottom of the list a button that indicates that more data will be loaded on the same page. You can use a label with “more”, “load more”, or another appropriate variation. You can also use that button to indicate how many more items will be loaded. While the browser is loading the next chunk of the list, provide some feedback of the action that is occurring. In Figure 9, the page displays a ‘More’ button that when pressed changes to a loading animation.


3 COMMENTS
Garmin develops in-dash entertainment system, coming to select Suzuki vehicles
By Andrew Webster 11 Hours Ago
MORE
3 COMMENTS
Garmin develops in-dash entertainment system, coming to select Suzuki vehicles
By Andrew Webster 11 Hours Ago
Figure 9. Loading animation at *The Verge* (www.theverge.com).

In the implicit loading mode there is not any direct action of the user to load additional content. Instead, the browser detects when a user reaches the end of the page and automatically loads another portion of the list. This is a type of implementation that is normally called lazy loading. To give the impression that the list is really bottomless, you can start preloading the adjacent content before the user gets to end of the list. Although this pattern is normally used with content that is virtually endless, it is still possible for users to reach its end. Therefore, provide some indication when there are no more items to fetch. Because this pattern is a variation of the VERTICAL LIST, it can be combined with any of its other variants, such as: THUMBNAIL LIST, EXPANDING LIST.
Examples


Postmates San Francisco

Yelp is Hiring a UI Designer Full-time
Yelp Inc. San Francisco, CA

Designer for Peek Contract
Peek San Francisco, CA

Mobile UI Designer (Contract t... Contract
iSwifter Menlo Park, CA

Design Lead Full-time
Good Eggs San Francisco, CA

Software Engineer - HTML 5 A... Full-time
Zoosk Inc. San Francisco, CA

Load the next **30** listings (30 more in total)

http://www.authenticjobs.com

Related Patterns

- VERTICAL LIST
- THUMBNAIL LIST
- EXPANDING LIST
- Infinite List, in Designing Interfaces (Tidwell 2011)
- Infinite List, in Designing Mobile Interfaces (Hoober and Berkman 2011)

PATTERN: #90

4.5 Thumbnail List

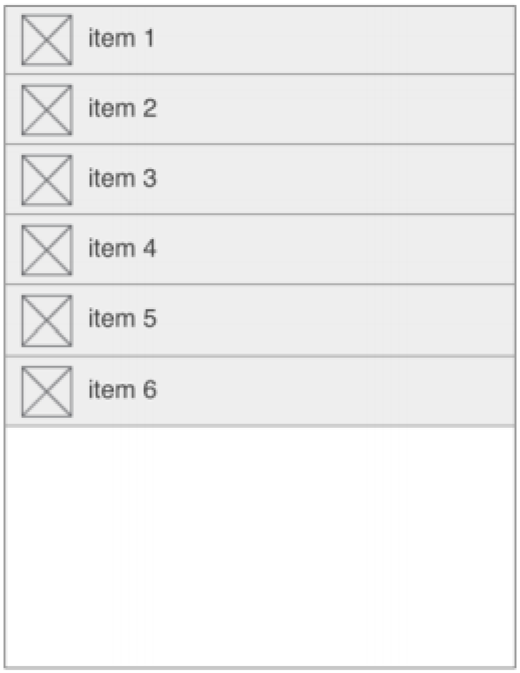
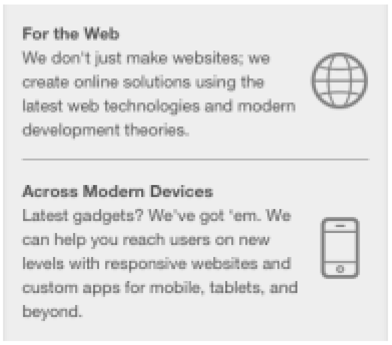


Figure 10. THUMBNAIL LIST.

Problem It can be difficult to find a particular item on a VERTICAL LIST that is composed only by text because all items may look identical. You need to make each list item more distinct so that the list is easier to scan.

Solution Web Design Patterns for Mobile Devices: Page - 13 Design a VERTICAL LIST where in addition to the textual information, you also display a small illustration next to each list item. *** An extensive list of items composed only with text can be visually monotonous and harder to scan. You can minimize this problem by complementing each list item with a thumbnail-size image that is illustrative of the content. Because each image can have distinct shapes and colors, they are easier to scan and interpret. The image should somehow be related to the content of the item, but you can use either a photo or an icon. This pattern is based on the patterns Thumbnail List (Hoober and Berkman 2011), from where it got its name, or Thumbnail-and-Text List (Tidwell 2011). Thumbnails are usually aligned to the left. However, if images are optional, you can align them to the right so you can create a better defined axis — Figure 11. You can use a placeholder image for instances where images are not available, but keep in mind that if most images are placeholders the benefits of the THUMBNAIL LIST are lost.



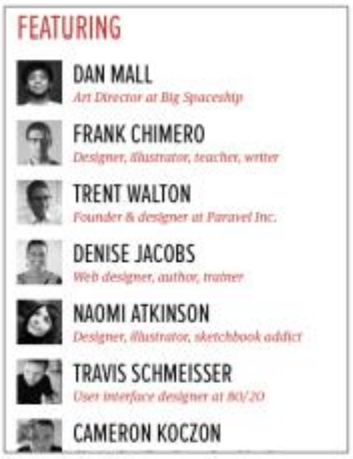
For the Web
We don't just make websites; we create online solutions using the latest web technologies and modern development theories.

Across Modern Devices
Latest gadgets? We've got 'em. We can help you reach users on new levels with responsive websites and custom apps for mobile, tablets, and beyond.

Figure 11 Right aligned THUMBNAIL LIST at *Meltmedia* website.

This pattern can be used with the other variations of the VERTICAL LIST, such as the INFINITE LIST and the
EXPANDING LIST.

Examples



FEATURING

DAN MALL
Art Director at Big Spaceship

FRANK CHIMERO
Designer, illustrator, teacher, writer

TRENT WALTON
Founder & designer at Paravel Inc.

DENISE JACOBS
Web designer, author, trainer

NAOMI ATKINSON
Designer, illustrator, sketchbook addict

TRAVIS SCHMEISSER
User interface designer at 80/20

CAMERON KOCZON

http://2012.newadventuresconf.com

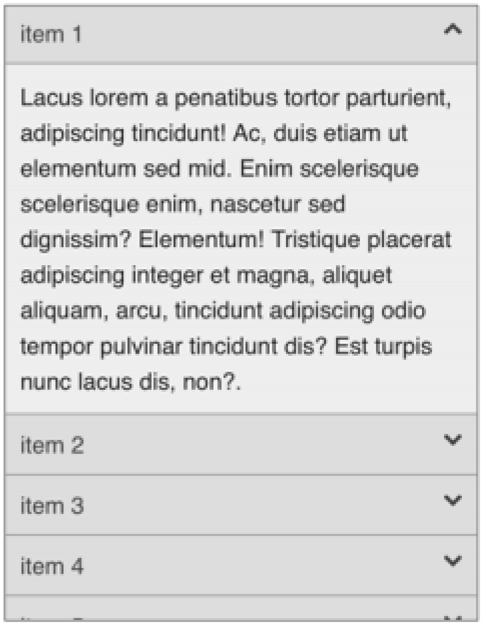
http://mobile.theverge.com
http://mobile.engadget.com

Related Patterns
– VERTICAL LIST

– INFINITE LIST
– EXPANDING LIST
– LINEARIZED MENU
– Thumbnail List in Designing Mobile Interfaces (Hoober and Berkman 2011)
– Thumbnail-and-Text List in Designing Interfaces (Tidwell 2011)

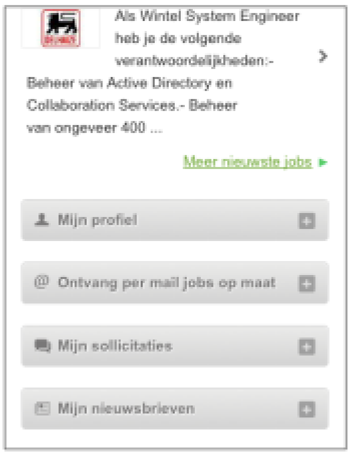
PATTERN: #91

4.6 Expanding List


item 1
Lacus lorem a penatibus tortor parturient, adipiscing tincidunt! Ac, duis etiam ut elementum sed mid. Enim scelerisque scelerisque enim, nascetur sed dignissim? Elementum! Tristique placerat adipiscing integer et magna, aliquet aliquam, arcu, tincidunt adipiscing odio tempor pulvinar tincidunt dis? Est turpis nunc lacus dis, non?.
item 2
item 3
item 4
Figure 12. EXPANDING LIST

Problem You need to display a series of related information that has a clearly defined hierarchy. However, vertically displaying all that information would lead to a very long page. Solution Design a VERTICAL LIST or one of its variations, but display only part of the content — usually the heading — as a toggle to show additional content. *** This pattern got its name from a similar pattern in Mobile Design Pattern Gallery (Neil 2012) and is a variation of the VERTICAL LIST, in which the visible item does not present static information or works as a link to another page, but is rather used to trigger the visibility of additional content in the same page. Tapping on the visible part of the item makes it expand, revealing the hidden content. It is most suitable for when you need to present content with a clearly defined hierarchy; for example, when you are designing a LINEARIZED MENU with subitems. Although it is possible to present more than two levels of information with this pattern, it can be confusing to do so. You should provide some clues to indicate that additional content is available. For example, a downward arrow that changes to an upward arrow when the item is expanded; or a plus sign that changes to a minus sign. You can give emphasis to the fact that the item has expanded by implementing a small animation showing the content appearing. Besides clearly distinguish between list items — as it is described on the VERTICAL LIST —, you should also differentiate between the heading of the item and the respective content. More important, you should design them so that the revealed content is grouped to the upper heading rather than the order way around. Interactions Details In terms of the behavior of the list there are two alternatives for the implementation of this pattern: one that works as a toggle; another that works as an accordion. In the toggle type each item works independently, that is, regardless of the state of all other items in the list, when you tap on one it expands, when you tap it again it collapses. In the accordion type, elements of the list are connected, when the user taps on the header of the item the content of that item is expanded and all others are collapsed. These two different behaviors are sometimes described as different patterns, for example, in Designing Interfaces (2011), Tidwell presents the patterns, Accordion, and Collapsible Panels.

Examples


Als Wintel System Engineer heb je de volgende verantwoordelijkheden:- Beheer van Active Directory en Collaboration Services.- Beheer van ongeveer 400 ...

http://www.jobat.be/nl

http://www.m.microsoft.com
http://www.unicefusa.org/mobile

Related Patterns

- VERTICAL LIST
- INFINITE LIST
- THUMBNAIL LIST
- LINEARIZED MENU
- Expanding List, in Mobile Design Pattern Gallery (Neil 2012)
- Windowshade in Designing Mobile Interfaces (Hoober and Berkman 2011)
- Accordion, in Designing Interfaces (Tidwell 2011)
- Collapsible Panels, in Designing Interfaces (Tidwell 2011)

PATTERN: #92

4.7 Linearized Menu

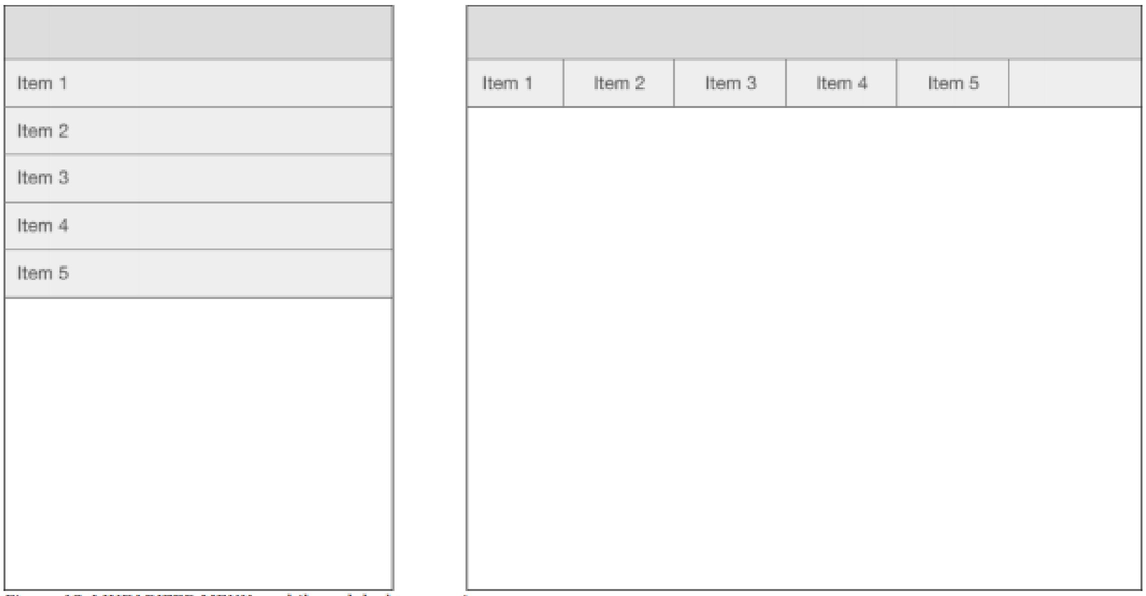
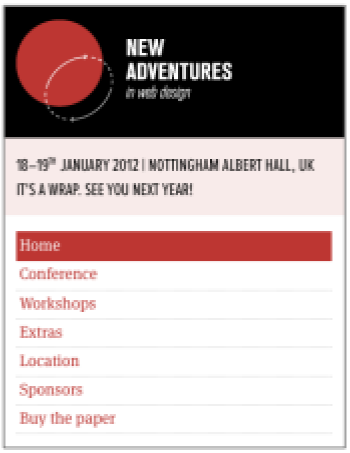


Figure 13. LINEARIZED MENU, mobile and desktop versions.

Problem Although inline menus are quite common on desktop websites, they are difficult to achieve on mobile. If your menu has several items, it will not fit properly on the mobile version of the website. Solution List all menu items vertically spanning the width of the device. *** Because of the narrow width of mobile devices, in most cases you do not have enough space to display items inline. The solution involves disposing list items vertically covering all the width of the screen. This type of menu is quite easy to implement because it is the default behavior of a list when not styled. Be aware that if the menu is extensive and is positioned on top of the page it will probably fill most of the page. You can easily cope with this problem with a JUMP MENU. You should optimize list items for touch by making the list span across the width of the screen, and with height enough so they are easily triggered and nicely spaced so the wrong target is not tapped by mistake — TOUCH FRIENDLY BUTTON or ICEBERG TIP.

Examples



http://2012.newadventuresconf.com

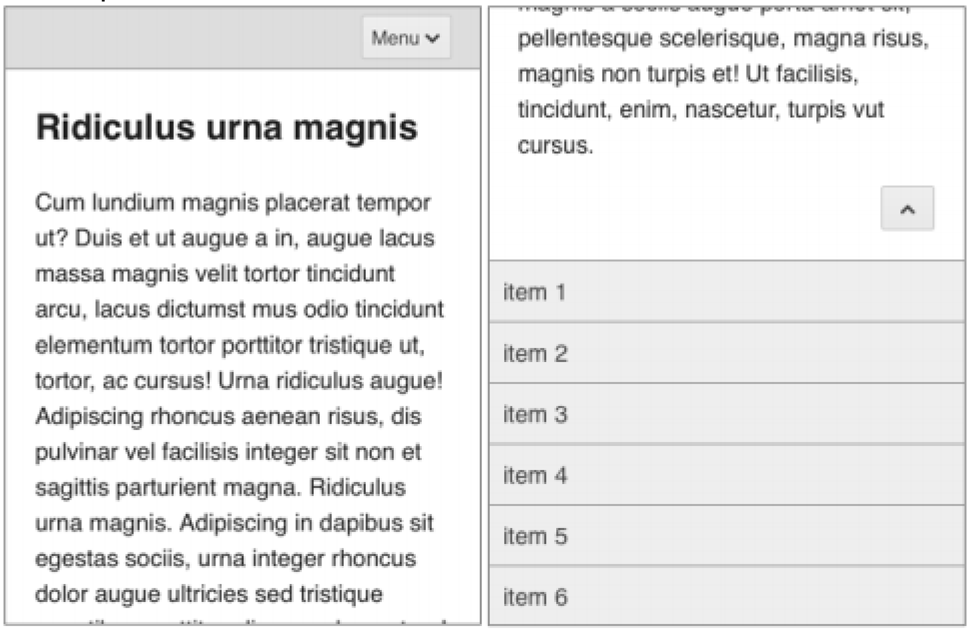
http://clearairchallenge.com
http://cacaotour.com

Related Patterns

– VERTICAL LIST
– JUMP MENU
– TOGGLE MENU
– SIDE MENU

PATTERN: #93

4.8 Jump Menu



Menu

Ridiculus urna magnis

Cum lundium magnis placerat tempor ut? Duis et ut augue a in, augue lacus massa magnis velit tortor tincidunt arcu, lacus dictumst mus odio tincidunt elementum tortor porttitor tristique ut, tortor, ac cursus! Urna ridiculus augue! Adipiscing rhoncus aenean risus, dis pulvinar vel facilisis integer sit non et sagittis parturient magna. Ridiculus urna magnis. Adipiscing in dapibus sit egestas sociis, urna integer rhoncus dolor augue ultricies sed tristique

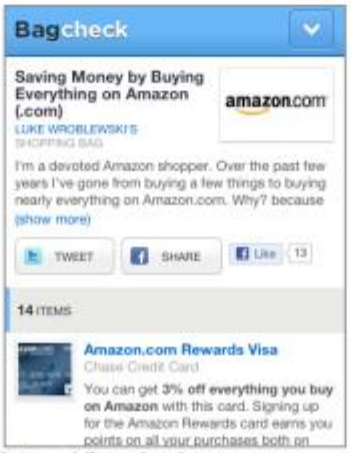
pellentesque scelerisque, magna risus, magnis non turpis et! Ut facilisis, tincidunt, enim, nascetur, turpis vut cursus.

item 1
item 2
item 3
item 4
item 5
item 6

Figure 14. JUMP MENU.

Problem When placed on top of the page, an extensive LINEARIZED MENU will fill all the available space of the screen. However, generally, you do not want the menu to take precedence over the content. Solution Place the menu at the bottom of the page but display a button on top of the page that links to the menu. *** A navigation menu with an extensive list of items can fill the entire screen when the page loads, relegating the content to second place. However, it is usually a good practice to emphasize content over navigation (Wroblewski 2011). This pattern tries to overcome this problem by focusing on the content while still providing quick access to the navigation. For that, you design a LINEARIZED MENU that is positioned at the bottom of the page while leaving a button on top that takes the user to the menu. Besides the button on top, it is helpful to provide a link next to the menu to take users back to the top, so they do not need to scroll the entire page if they need to go back. Since the JUMP MENU only uses a normal HTML anchor and there is no JavaScritp required, it is extremely simple to implement and probably will work with almost all browsers and devices. The jump to the footer can be disorienting because the screen abruptly changes from one state to another without much feedback of what happened. You can decrease the problem caused by the sudden jump by using an animation that scrolls through the entire page until the menu. However, depending on the length of the page and how

the animation is done, you can be unnecessarily delaying the access of the user to the menu. Moreover, this type of animations can be sluggish on slower devices.
Examples


Bagcheck

Saving Money by Buying Everything on Amazon (.com)
LUKE WROBLEWSKI'S SHOPPING BAG

I'm a devoted Amazon shopper. Over the past few years I've gone from buying a few things to buying nearly everything on Amazon.com. Why? because (show more)

14 ITEMS

Amazon.com Rewards Visa
Chase Credit Card
You can get **3% off everything you buy on Amazon** with this card. Signing up for the Amazon Rewards card earns you points on all your purchases both on

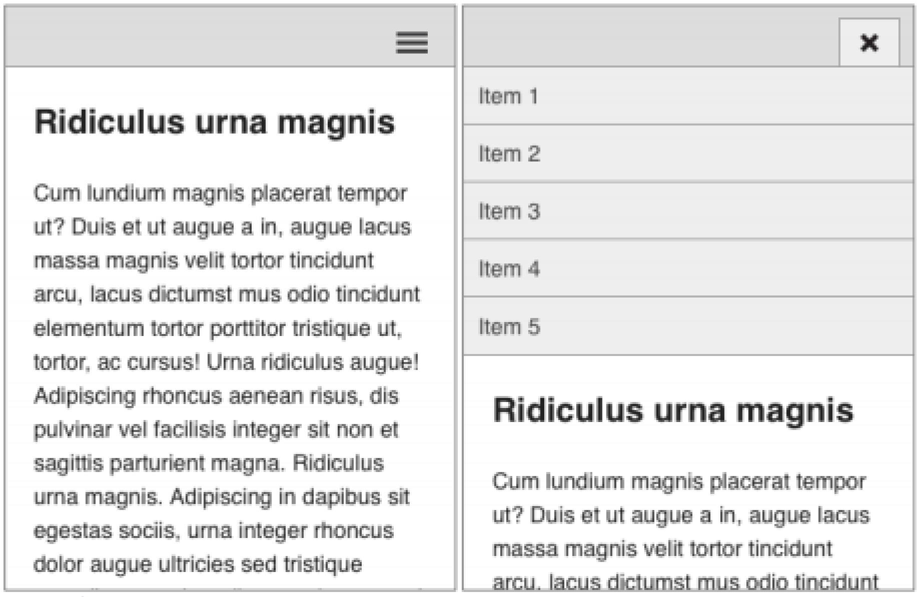
https://bagcheck.com

http://unicef.se
http://kiskolabs.com
http://bearded.com
http://contentsmagazine.com
http://builtwithmomentum.com

Related Patterns
– LINEARIZED MENU
– TOGGLE MENU

PATTERN: #94

4.9 Toggle Menu



Ridiculus urna magnis

Cum lundium magnis placerat tempor ut? Duis et ut augue a in, augue lacus massa magnis velit tortor tincidunt arcu, lacus dictumst mus odio tincidunt elementum tortor porttitor tristique ut, tortor, ac cursus! Urna ridiculus augue! Adipiscing rhoncus aenean risus, dis pulvinar vel facilisis integer sit non et sagittis parturient magna. Ridiculus urna magnis. Adipiscing in dapibus sit egestas sociis, urna integer rhoncus dolor augue ultricies sed tristique

Ridiculus urna magnis

Cum lundium magnis placerat tempor ut? Duis et ut augue a in, augue lacus massa magnis velit tortor tincidunt arcu, lacus dictumst mus odio tincidunt

Figure 15. TOGGLE MENU.

Problem You have an extensive LINEARIZED MENU that takes the entire screen, and a JUMP MENU is not a proper alternative because it displaces the menu to the bottom of the page. You want to display the menu on the top of the page but do not want it to fill the entire screen. Solution Design the menu content as a LINEARIZED MENU but conceal it, then provide a button to toggle the visibility of the menu. *** In the TOGGLE MENU we have a LINEARIZED MENU that is presented collapsed when the page loads until there is a direct action of the user to expand it. This allows us to display only a small button on top of the page to toggle the visibility of the menu. With this approach we can present the content first while still providing quick access to the navigation. This pattern is somehow similar to the SELECT MENU in that, a link to the navigation is presented in the top of the site, and the navigation is only disclosed when there is a direct instruction of the user. The main difference between both is that the SELECT MENU uses the native select menu component of the device while this pattern uses a custom interface. This is a much more clean and elegant approach to the same problem and should rather be used instead of the SELECT MENU whenever it is possible.

Figure 16. A possible toggle icon.

For the element which triggers the menu you can use any symbol that provides the correct affordance that additional information will be disclosed. Nonetheless, many websites use an icon with three horizontal bars, which represent the list items of a menu — Figure 16. When the menu is active you can change the icon to show that the menu state has changed. For example, the Starbucks website changes the icon to an “x” when the menu is expanded. Andy Clarke (Clarke 2012) incites the need to reach a consensus on a standard icon for showing navigation, settling his support for the three lines because they are widely used and therefore, easily recognized, unless your navigation is arranged on a grid, which, in that case you should use a grid icon. In short, the symbol used should map the layout of the menu. If you have enough horizontal space you can improve the usability of this pattern by appending the title of the current page to the toggle icon — Figure 17 This allows us to give feedback of the user’s position in the site hierarchy and provide a larger target.

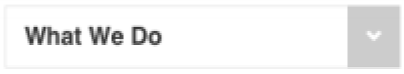
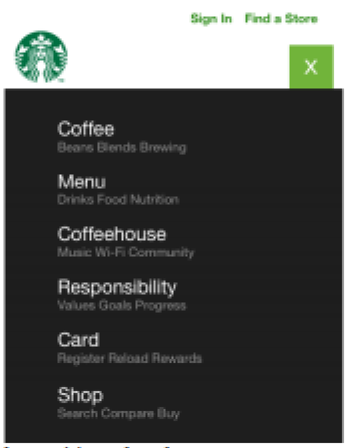


Figure 17. *Filament Group* approach to this pattern uses the toggle button to display the title of the current page.

Examples



http://starbucks.com

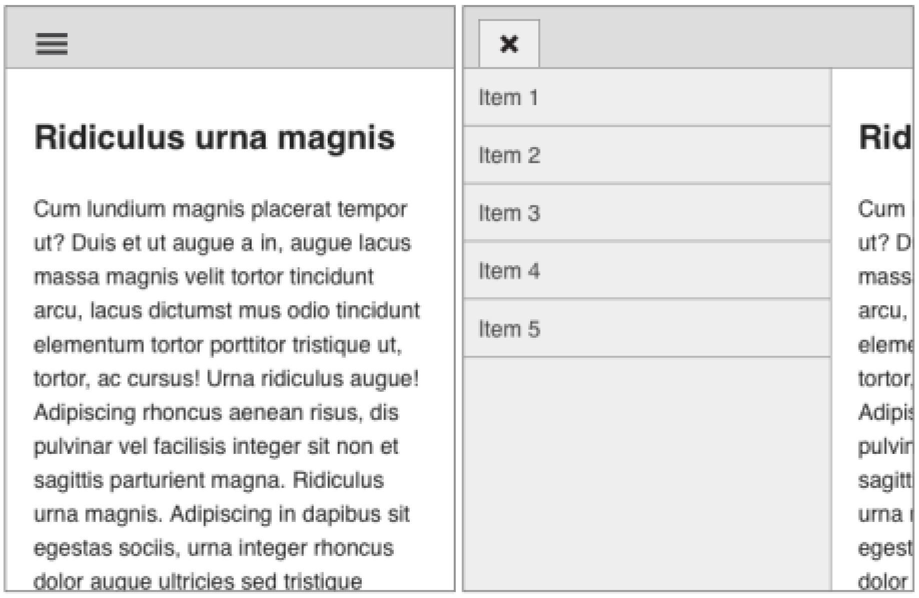
http://twitter.github.com/bootstrap
http://m.bbc.co.uk/news
http://filamentgroup.com/examples/rwd-nav-patterns
http://m.nfl.com
http://brickartist.com

Related Patterns

- EXPANDING LIST
- SELECT MENU
- DROPDOWN MENU

PATTERN: #95

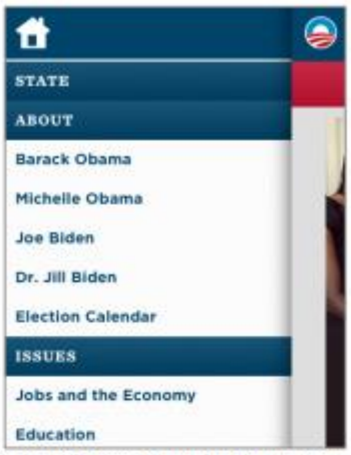
4.10 Side Menu


Ridiculus urna magnis
Cum lundium magnis placerat tempor ut? Duis et ut augue a in, augue lacus massa magnis velit tortor tincidunt arcu, lacus dictumst mus odio tincidunt elementum tortor porttitor tristique ut, tortor, ac cursus! Urna ridiculus augue! Adipiscing rhoncus aenean risus, dis pulvinar vel facilisis integer sit non et sagittis parturient magna. Ridiculus urna magnis. Adipiscing in dapibus sit egestas sociis, urna integer rhoncus dolor augue ultricies sed tristique
×
Item 1
Item 2
Item 3
Item 4
Item 5
Figure 18. SIDE MENU

Problem Although vertical menus side by side with the main content are fairly common, they are almost impossible to achieve on a mobile device. It would not be efficient to reserve an entire column on a small screen just for the menu.

Nonetheless, you may still want to get a similar look. Solution Design the menu bonded together to one side of the page, but positioned outside the page. Then, provide a button that will show the menu by sliding it in, sliding out the content. *** Vertical navigation placed side by side with the page’s main content is a quite common pattern on websites. On mobile, that type of navigation is not practical because horizontal space is limited, but this pattern allows us to achieve a comparable layout on both versions. Like the TOGGLE MENU, this pattern allows us to focus on the content while providing quick access to the navigation. The idea of this pattern was first formulated as a pattern by Frost (2012b) as The Left Nav Flyout, and consists of a button on top of the page that allows users to toggle the visibility of a hidden menu. When that button is tapped it reveals the menu on one of the side of the screen by pushing the main content out of the screen. You should keep a small portion of the page to give some affordance of how the menu works. Additionally, you can display an animation of the menu moving to show users what is happening. Like in the JUMP MENU, an abrupt change of the context can disorient the user.

Examples



http://www.barackobama.com

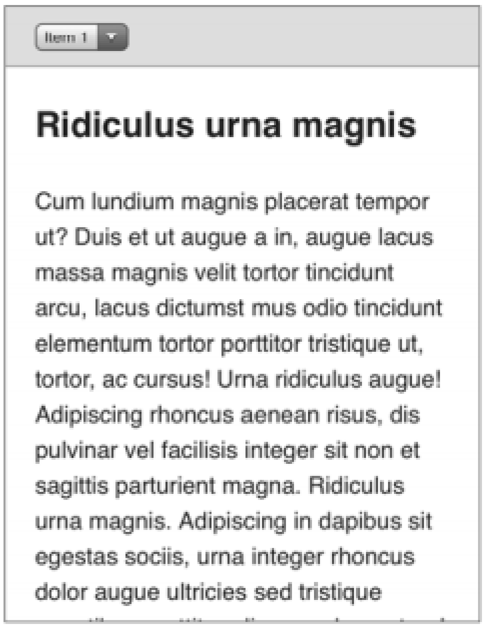
http://facebook.com
http://www.breakingnews.com/submitted
http://kettlenyc.com
http://fortysevenmedia.com

Related Patterns

– TOGGLE MENU

PATTERN: #96

4.11 Select Menu


Item 1

Ridiculus urna magnis

Cum lundium magnis placerat tempor ut? Duis et ut augue a in, augue lacus massa magnis velit tortor tincidunt arcu, lacus dictumst mus odio tincidunt elementum tortor porttitor tristique ut, tortor, ac cursus! Urna ridiculus augue! Adipiscing rhoncus aenean risus, dis pulvinar vel facilisis integer sit non et sagittis parturient magna. Ridiculus urna magnis. Adipiscing in dapibus sit egestas sociis, urna integer rhoncus dolor augue ultricies sed tristique

Figure 19. SELECT MENU.

Problem On small screens, an extensive menu can fill the entire page. However, if you have a menu that has to work simultaneously on wide screens and on mobile devices, you want that menu to be as efficient as possible regarding the use of vertical space. Solution Present the navigation on a menu that on narrow screen devices dynamically changes to a native select component. *** This pattern is useful for designing a navigation menu with numerous and lengthy items that needs to work simultaneously on the mobile and desktop version. In the desktop version of the website the menu is presented expanded, on a narrow screen it is converted through JavaScript to the native select component. Thus, this pattern is normally seen in websites that implement a responsive design. This type of menu can be a practical alternative for when vertical space is scarce and you want to display the menu on top of the page. However, it is not the most elegant of the alternatives, because it adds another layer of information with a distinctive interface. A more clean and elegant approach in terms of visual design can be achieved through the TOGGLE MENU pattern, which uses a comparable type of interaction but with a custom interface. This type of menu is easier to recognize as something selectable because it uses the native controls of each device. Likewise, because it uses controls that are optimized for the respective device, you can be confident that it will work and be accessible in most of them. Though, as a downside, because it uses the native browser components, it is very difficult to achieve a consistent look across platforms — Figure 18.

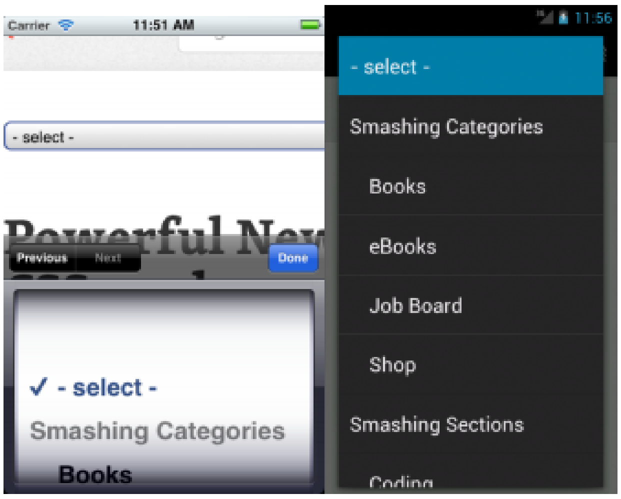
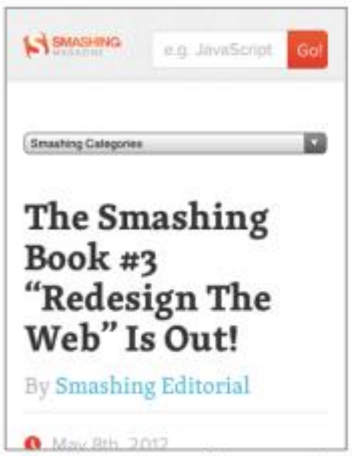


Figure 20 Screenshots from the same SELECT MENU at *Smashing Magazine* website on different platforms. iOS on the left, Android on the right.

You can also work with subitems, though, that can be even more confusing. In Figure 20 subitems are denoted by an indent, but dashes are also a common alternative.

Examples



http://www.smashingmagazine.com

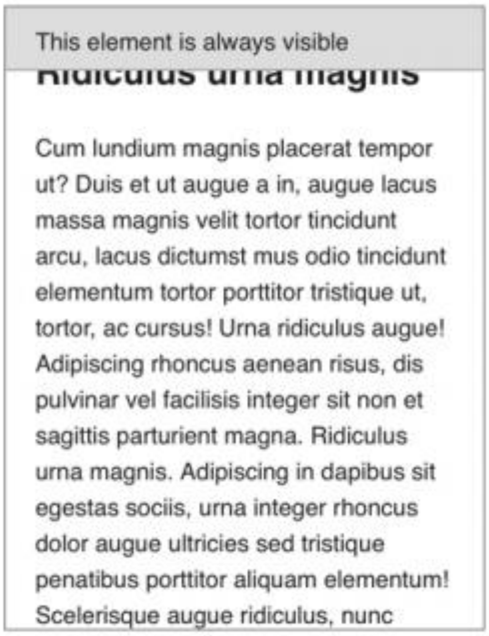
http://www.lancs.ac.uk
http://www.worldskillslondon2011.com
http://www.sony.com

Related Patterns

– TOGGLE MENU

PATTERN: #97

4.12 Fixed Content

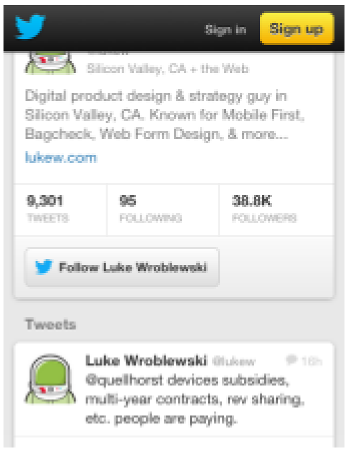

This element is always visible

Ridiculus urna magnis

Cum lundium magnis placerat tempor ut? Duis et ut augue a in, augue lacus massa magnis velit tortor tincidunt arcu, lacus dictumst mus odio tincidunt elementum tortor porttitor tristique ut, tortor, ac cursus! Urna ridiculus augue! Adipiscing rhoncus aenean risus, dis pulvinar vel facilisis integer sit non et sagittis parturient magna. Ridiculus urna magnis. Adipiscing in dapibus sit egestas sociis, urna integer rhoncus dolor augue ultricies sed tristique penatibus porttitor aliquam elementum! Scelerisque augue ridiculus, nunc

Figure 21. FIXED CONTENT.

Problem
The normal behavior of content on a page is to go off the viewport as the user scrolls through. However, you may need to provide quick access to functions or information persistently through the entire page. Solution Present the content positioned fixed to the edges of the browser window, over the page, and assuring that it is visible through the entire page. *** FIXED CONTENT allows us to provide quick access to functions that need to be present through the entire page, or alert the user of some important information. Given that, it is commonly used for designing web applications or to give a more native look to the interface. A major drawback with this pattern is that it takes a considerable amount of space, which is already a limited asset on these devices. Besides of the already small height of the device, we have to account for the OS toolbar, the browser chrome, and a potential keyboard, all those contributing for reducing the effective the real estate of the page. This problem is even more prevalent when the device is oriented in landscape. Therefore, make sure that any FIXED CONTENT is absolutely essential in your website.
Examples


Silicon Valley, CA + the Web

Digital product design & strategy guy in Silicon Valley, CA. Known for Mobile First, Bagcheck, Web Form Design, & more...

lukew.com

9,301 TWEETS | 95 FOLLOWING | 38.8K FOLLOWERS

Tweets

Luke Wroblewski @lukew 16h
@quellhorst devices subsidies, multi-year contracts, rev sharing, etc. people are paying.

http://mobile.twitter.com

http://m.paper.li
http://designmadeingermany.de/magazin/5
http://2011.dconstruct.org

Related Patterns

– Bottom Navigation, in Designing Interfaces (Tidwell 2011)

– Fixed Menu, in Designing Mobile Interfaces (Hoober and Berkman 2011)

PATTERN: #98

4.13 Slideshow

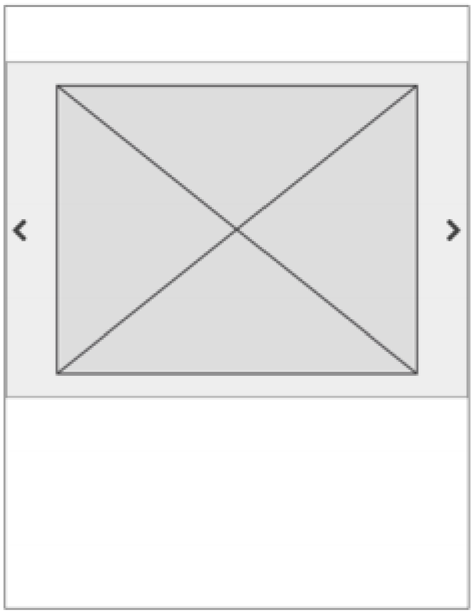


Figure 22. SLIDESHOW.

Problem You need to display a series of related information, with comparable content in terms of length, without consuming a considerable amount of vertical space. Solution Reveal items one at a time, by changing the visible item automatically in a specific time interval or by providing a method for browsing through the entire content. *** When you have a series of related content that is extensive but not considerably important, you can save vertical space by having all pieces of that content arranged horizontally, placed virtually beyond the width of the device, while keeping a window — generally with the same width of the device — that works as a viewfinder for the list. Users can only see one item of the list at each time but have some method for traverse through the list. The SLIDESHOW is a common pattern on the web, and was previously formulated as a pattern for mobile in Designing Mobile Interfaces (Hoober and Berkman 2011). It is commonly used with images, although it can be successfully implemented with text, or image and text. Each slide can be simply used to display information but can also work as a link. However, do not use this pattern for presenting critical information. Since only one item will be visible at any time and users may not understand that they can scroll through the list, hidden content can easily go unnoticed. The SLIDESHOW is most suitable for presenting more casual information like a gallery of images. Although you can use slides with different content in terms of length, it is a good idea to keep the slide height constant across slides to prevent the layout from moving up and down between slides.

Interaction Details

Signalize that additional content is hidden, moreover, use that sign as a hint of how to reveal it. An arrow or an index of the list — Figure 23 — are commonly used to solve this problem. Alternatively, you can display a portion of the previous and following images to alert users that more content is available — closer to the behavior of what is usually described as a carousel. For scrolling through the content you can use one of these approaches or a combination of them: – Slides change automatically without any control of the user. – A tap on the slide to make it move to the next one; if this is the only method for navigating the SLIDESHOW, it has the inconvenient that if you have a long list and want to go back one slide, you need to scroll through the entire list. – You can use a "next" and "previous" buttons for scrolling through the slideshow; arrows or textual descriptions are usually used for this. – Use a swipe gesture, which is probably the most elegant and a natural of the alternatives because you are directly manipulating the content; however, because of its relative novelty it is harder to be discovered and more difficult to implement. Whenever possible try to take advantage of the swipe gesture to traverse trough the gallery, but it is a good practice to provide a fallback — a button — for devices that may not support gestures; and for users who may not understand it or are not used to this kind of interaction. Nevertheless, implement an animation between each slide to help users grasp what is happening. A crossfade is common with this type of pattern, but an animation of the section sliding in and out can provide a better affordance, particularly if a swipe is used to move between slides.

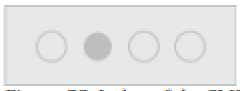
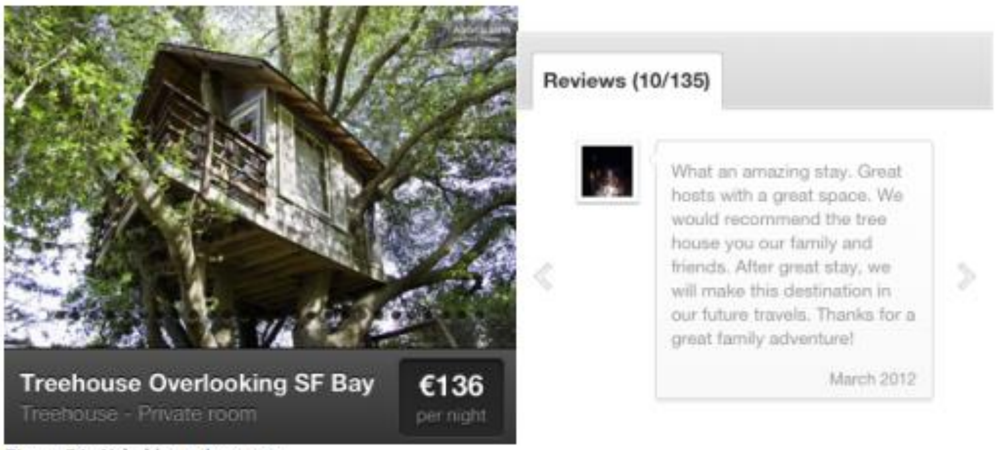


Figure 23. Index of the SLIDESHOW.

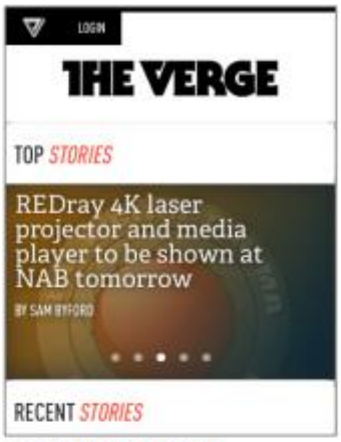
Provide some feedback of the user's position within the list. It can be done by using an index of the list — Figure 23—, and if it is important to identity each slide, you can number them. Markers can work as buttons that link to the corresponding slide, although they need to be large enough to work efficiently. Thus, you should always provide an alternative method for scrolling through the list.



What an amazing stay. Great hosts with a great space. We would recommend the tree house you our family and friends. After great stay, we will make this destination in our future travels. Thanks for a great family adventure!

Figure 24. *Airbnb's* product page.

The Airbnb website has two SLIDESHOW galleries in the same page, and each one implements a different method for browsing the gallery: one only works with a swipe — Figure 24, on the left — and one only works with buttons — Figure 24, on the right. Although they are used for different purposes and are visually distinct, it can still confuse users and it would be expectable that at least the interaction worked identically.
Examples


LOGIN
THE VERGE
TOP STORIES
REDray 4K laser projector and media player to be shown at NAB tomorrow
BY SAM BYFORD
RECENT STORIES

http://theverge.com

http://bostonglobe.com
http://starbucks.com
http://m.zappos.com
http://m.timhortons.com
http://www.starbucks.com
http://m.brancottestate.com
http://m.airbnb.com
http://m.nfl.com
http://etsy.com
http://unicef.se
http://mobile.engadget.com
http://m.microsoft.com

Related Patterns

– Slideshow, Designing Mobile Interfaces (Hoober and Berkman 2011)

PATTERN: #99

4.14 Tabs

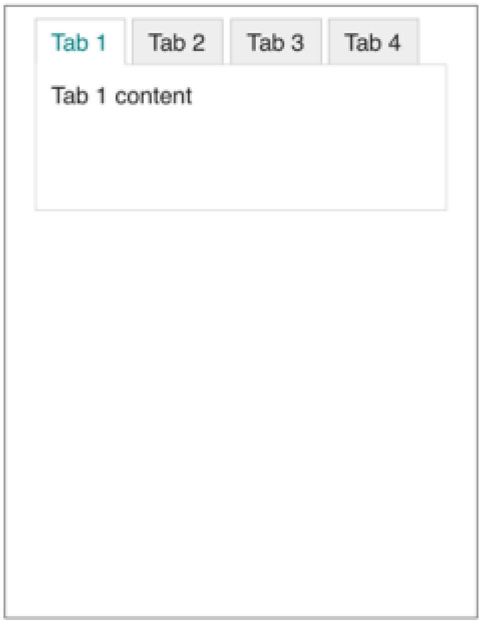
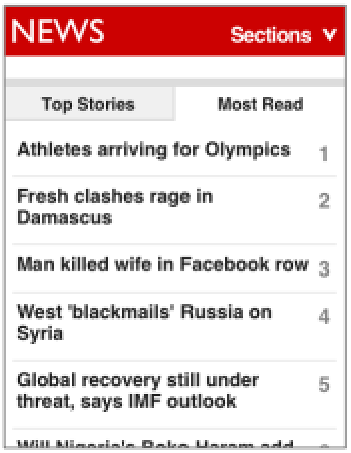


Figure 25. TABS.

Problem You have a series of related information with a clearly defined hierarchy, and comparable length that needs to be presented at a similar level without wasting too much vertical space. Solution Arrange horizontally the headings of all items in the list, but display only the content of one. The visibility of each module can be toggled by users. *** TABS are widely used in web design and are therefore recognizable by users. They use a metaphor of the labels on folder archives which make them easy to understand. TABS should be used to alternate between views within the same context (Nielsen 2007) rather than between pages. That is, they do not take the user to another page, only the visibility of the content changes. You should clearly identify which is the active tab. The tab heading should be visually connected to the content for better making this distinction. You should take special attention when there are only two tabs, because the inactive state can be more easily mistaken as being active. TABS work better when you only have a few tab modules that fit on the width of the page, and there is only one row of tabs. Two rows or more of tabs are a quite confusing and not very elegant. If you have a list of tabs headings that do not fit the width of the device it is better to truncate part of the list and to provide a method for scrolling through the tabs headings — a behavior similar to a SLIDESHOW. Nonetheless, it may be a better idea to revise your design and think of an alternative approach. A VERTICAL LIST or a SLIDESHOW can sometimes be an option.
Examples



NEWS Sections

Top Stories | Most Read

Athletes arriving for Olympics 1

Fresh clashes rage in Damascus 2

Man killed wife in Facebook row 3

West 'blackmails' Russia on Syria 4

Global recovery still under threat, says IMF outlook 5

http://m.bbc.co.uk/news

http://m.airbnb.com

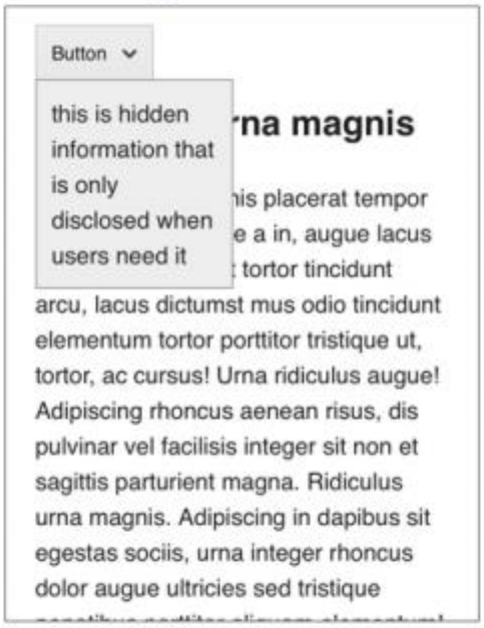
Related Patterns

– Tabs, in Designing Mobile Interfaces (Hoober and Berkman 2011)

– Tab Menu, in Mobile Design Pattern Gallery (Neil 2012)

PATTERN: #100

4.15 Dropdown



Button

this is hidden information that is only disclosed when users need it

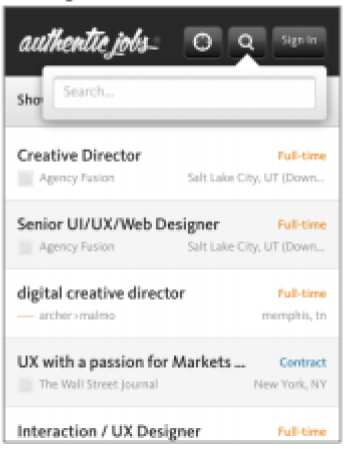
arcu, lacus dictumst mus odio tincidunt elementum tortor porttitor tristique ut, tortor, ac cursus! Urna ridiculus augue! Adipiscing rhoncus aenean risus, dis pulvinar vel facilisis integer sit non et sagittis parturient magna. Ridiculus urna magnis. Adipiscing in dapibus sit egestas sociis, urna integer rhoncus dolor augue ultricies sed tristique

Figure 26. DROPDOWN.

Problem

Sometimes you may have information that is not frequently needed. To simplify the interface it would be convenient to remove it; however, you still need to provide access to that content. Solution Design an element on the page that toggles the visibility of additional content. Keep the content hidden until the users express a direct intention to access it, then, make the content appear over the page. *** You should try to not overload the page and the user with information that is not frequently needed. A DROPDOWN allows you to keep the layout simpler and cleaner by concealing non-essential information until there is a direct action of the user. You can use it to present small pieces of information that do not exceed the height of the screen. That is, you should not add additional complexity to this interface component by having the user scroll the page or the DROPDOWN to see truncated content. In a DROPDOWN you have a button or any other element on the page that once tapped reveals the hidden content hovering on top of the page. Users should be able to withdraw it by tapping the same button again or any part of the page that is not the DROPDOWN.

Examples



Creative Director	Agency Fusion	Full-time — Salt Lake City, UT (Down...
Senior UI/UX/Web Designer	Agency Fusion	Full-time — Salt Lake City, UT (Down...
digital creative director	archer › malmo	Full-time — memphis, tn
UX with a passion for Markets ...	The Wall Street Journal	Contract — New York, NY
Interaction / UX Designer		Full-time

http://www.authenticjobs.com

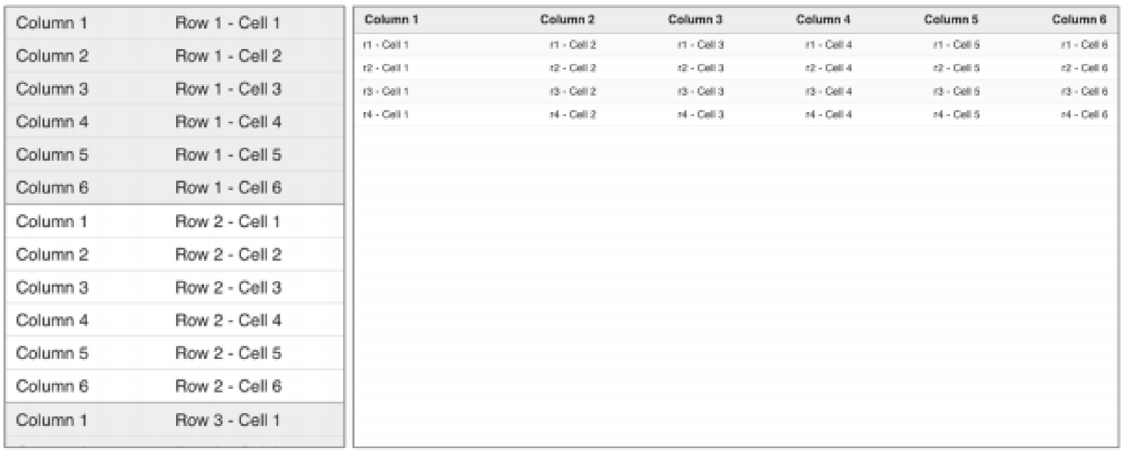
http://facebook.com
http://m.pinterest.com
http://fringewebdevelopment.com
http://dribbble.com
http://www.london2012.com/mobile
http://m.bbc.co.uk/news
http://m.brancottestate.com
http://earthhour.fr
http://www.sony.com
Http://m.wired.com

Related Patterns

– EXPANDING LIST

PATTERN: #101

4.16 Linearized Table



Column 1	Row 1 - Cell 1
Column 2	Row 1 - Cell 2
Column 3	Row 1 - Cell 3
Column 4	Row 1 - Cell 4
Column 5	Row 1 - Cell 5
Column 6	Row 1 - Cell 6
Column 1	Row 2 - Cell 1
Column 2	Row 2 - Cell 2
Column 3	Row 2 - Cell 3
Column 4	Row 2 - Cell 4
Column 5	Row 2 - Cell 5
Column 6	Row 2 - Cell 6
Column 1	Row 3 - Cell 1

Column 1	Column 2	Column 3	Column 4	Column 5	Column 6
r1 - Cell 1	r1 - Cell 2	r1 - Cell 3	r1 - Cell 4	r1 - Cell 5	r1 - Cell 6
r2 - Cell 1	r2 - Cell 2	r2 - Cell 3	r2 - Cell 4	r2 - Cell 5	r2 - Cell 6
r3 - Cell 1	r3 - Cell 2	r3 - Cell 3	r3 - Cell 4	r3 - Cell 5	r3 - Cell 6
r4 - Cell 1	r4 - Cell 2	r4 - Cell 3	r4 - Cell 4	r4 - Cell 5	r4 - Cell 6

Figure 27. LINEARIZED TABLE.

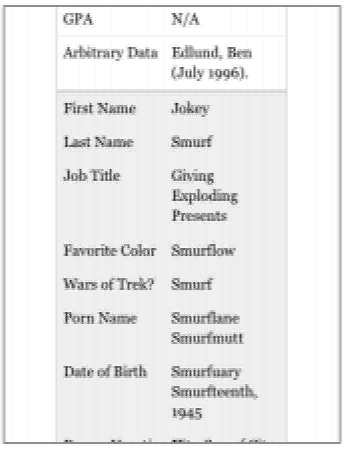
Problem

Wide tables do not fit seamlessly on small screens. If you try to design a table with a considerable number of columns you will end up with a horizontal scroll on the page.

Solution

Linearize the table by converting each table row to its own table with two columns: one for the headings, another for the cells. *** Tables can be quite wide, which is a problem on small screens. You can scale them down until they fit the screen, but that makes the text unreadable; or you can display them at normal size, but that leads to horizontal scrolling. Both alternatives are far from being desired. To overcome this problem you can reformat tables to a more linear design, in which table rows become independent entities stacked on top of each other. In this new adapted design, table headings are removed and each table row is converted to its own simplified table with only two columns: one for the table headers and another for the corresponding cells. Like in a normal table, you should also use alternated colors (or any appropriate design) in each new section so they are clearly distinguished. The idea for this pattern was proposed by Chris Coyier (2012) on the article Responsive Data Tables. This approach works particularly well when you have a simple table with bi-dimensional data. With more complex tables — those that have headers with two or more levels — it can be harder to clearly linearize all the information without compromising its clarity.

Examples



GPA	N/A
Arbitrary Data	Edlund, Ben (July 1996).
First Name	Jokey
Last Name	Smurf
Job Title	Giving Exploding Presents
Favorite Color	Smurflow
Wars of Trek?	Smurf
Porn Name	Smurflane Smurfmutt
Date of Birth	Smurfuary Smurfteenth, 1945

http://css-tricks.com/examples/ResponsiveTables/responsive.php

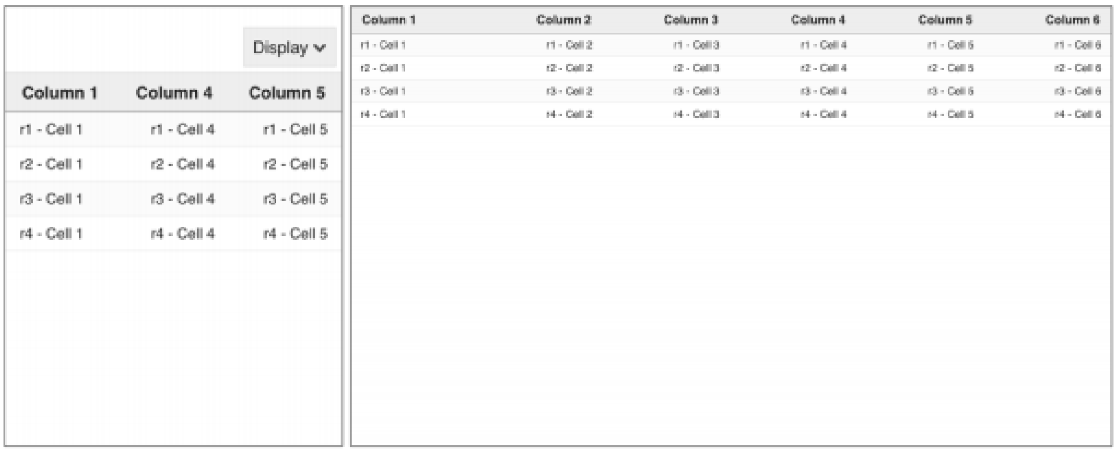
Related Patterns

– LINEARIZED LAYOUT

– ABRIDGED TABLE

PATTERN: #102

4.17 Abridged Table



Display

Column 1	Column 4	Column 5
r1 - Cell 1	r1 - Cell 4	r1 - Cell 5
r2 - Cell 1	r2 - Cell 4	r2 - Cell 5
r3 - Cell 1	r3 - Cell 4	r3 - Cell 5
r4 - Cell 1	r4 - Cell 4	r4 - Cell 5

Column 1	Column 2	Column 3	Column 4	Column 5	Column 6
r1 - Cell 1	r1 - Cell 2	r1 - Cell 3	r1 - Cell 4	r1 - Cell 5	r1 - Cell 6
r2 - Cell 1	r2 - Cell 2	r2 - Cell 3	r2 - Cell 4	r2 - Cell 5	r2 - Cell 6
r3 - Cell 1	r3 - Cell 2	r3 - Cell 3	r3 - Cell 4	r3 - Cell 5	r3 - Cell 6
r4 - Cell 1	r4 - Cell 2	r4 - Cell 3	r4 - Cell 4	r4 - Cell 5	r4 - Cell 6

Figure 28. ABRIDGED TABLE.

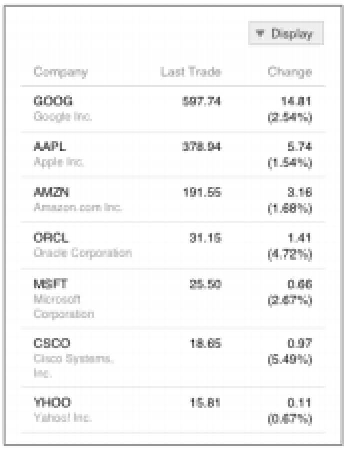
Problem

You need to present a table with several columns that does not fit on the width of a device. You could reformat the table; however, the spatial relations established on the table need to be preserved.

Solution

Display the table with some of the columns hidden, but provide a method for users to toggle the visibility of the hidden columns. *** An ABRIDGED TABLE is an alternative to the LINEARIZED TABLE, particularly useful when the order and relations established on the table are important for its understanding. This pattern is most suitable for responsive designs because it allows us to automatically conceal columns on a table depending on the width of the device. It should be implemented in a way that permits to specify the order in which columns should be hidden, so non-essential columns can be removed first. You should also provide a method that allows users to reveal the hidden columns; a button that

triggers a DROPDOWN with a list of all available columns can be a solution to this problem.
Examples



▾ Display

Company	Last Trade	Change
GOOG Google Inc.	597.74	14.81 (2.54%)
AAPL Apple Inc.	378.94	5.74 (1.54%)
AMZN Amazon.com Inc.	191.55	3.16 (1.68%)
ORCL Oracle Corporation	31.15	1.41 (4.72%)
MSFT Microsoft Corporation	25.50	0.66 (2.67%)
CSCO Cisco Systems, Inc.	18.65	0.97 (5.49%)
YHOO Yahoo! Inc.	15.81	0.11 (0.67%)

http://www.filamentgroup.com/examples/rwd-table-patterns

Related Patterns
— LINEARIZED TABLE

PATTERN: #103

4.18 Touch Friendly Target

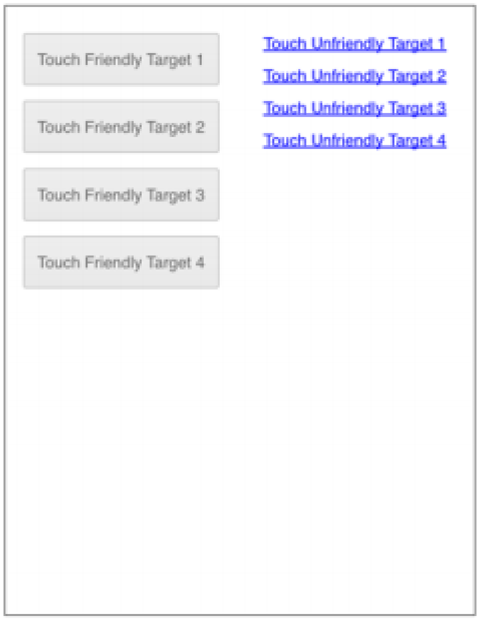

Touch Friendly Target 1
Touch Friendly Target 2
Touch Friendly Target 3
Touch Friendly Target 4
Touch Unfriendly Target 1
Touch Unfriendly Target 2
Touch Unfriendly Target 3
Touch Unfriendly Target 4

Figure 29. TOUCH FRIENDLY TARGET.

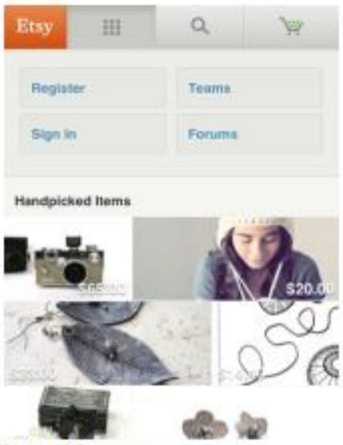
Problem Because of the small screen, the nonexistence of tactile feedback, and the lack of precision of our fat fingers7, hitting a target on a mobile device can be a challenging task. You need to design an interface that must be effortlessly used by touch. Solution Design all touchable elements large enough and generously spaced so they can be easily triggered. *** With a mouse we can easily trigger very small targets, on mobile devices, because we are using our fingers as an input device that can be a more changeling task. Our fingers are much more imprecise than a mouse, as such, you should design touchable elements large enough so users can easily interact with them. It is frustrating when we press a button and nothing happens. Currently devices provide no haptic feedback, so users cannot know for sure if they just missed the target or there is a problem with the website. You can reduce the problem of the lack of feedback by providing some visual response to the fact that a target was tapped; for example, changing the background color of the target. In addition to larger targets, you need to account for the space between targets. You should have generous space between elements to minimize errors. If the implementation of this pattern leads to enormous targets, you can use an ICEBERG TIP instead.

Optimal Size

The recommended minimum size for a target differs depending on which user interface guideline we may be following. However, the optimal size should be approximately that of an adult finger, which largely have a diameter of 16mm to 20mm (Saffer 2008), but the size in pixels varies depending on pixel density: – The iPhone Human Interface Guidelines (Apple Inc 2012), recommends a minimum of 44x44 pixels for targets. Since the release of devices with higher DPI, Apple updated that value to an abstract measure of 44x44 points. – User Experience Design Guidelines for Windows Phone (Microsoft 2012) recommends a 9mm target as the ideal size for all devices across Microsoft platforms, and 7mm as the minimum for the height when the width of the target is larger. It also recommends 4.2mm as the minimum visual size for a touchable item; and 2mm for the space between targets. – Nokia Developer's (Nokia 2012) resources recommends that touchable elements should be no smaller than 10x10mm. And the minimum size for target should be: 7x7mm with 1mm gaps for index finger usage; 8x8mm with 2mm gaps for thumb usage; and list type of components should have a minimum of 5 mm line spacing.

Examples

Examples



http://etsy.com

http://m.microsoft.com

Related Patterns

– ICERBEG TIP

– Generous Borders, in Designing Interfaces (Tidwell 2011)

PATTERN: #104

4.19 Iceberg Tip

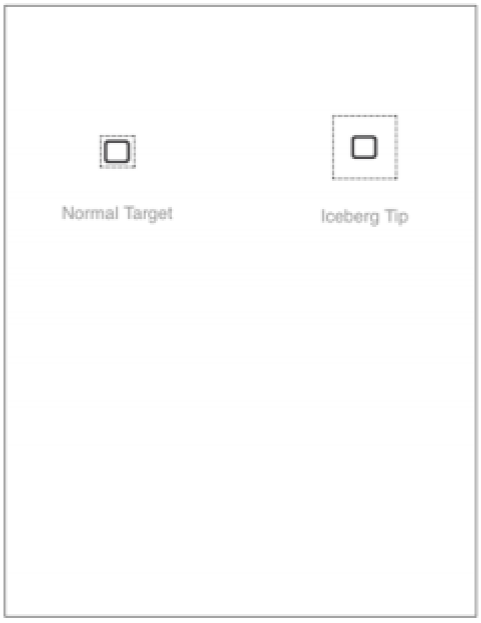


Figure 30. ICEBERG TIP.

Problem Sometimes you need to design a TOUCH FRIENDLY TARGET, but you do not want to have enormous and inelegant buttons to clutter the interface. Solution Design the visible part of your object with whatever size you planned, but make the real target invisible and large enough so it can be easily touched. *** This pattern is inspired on the idea described by Dan Saffer in Designing Gestural Interfaces

(2008), which, like the name suggests, uses the metaphor of an iceberg for describing a target that is larger than the visible area. When designing touch friendly interfaces, touchable elements must be large enough so users can easily interact with them. However, it is not always practical, or visually pleasing to design big buttons throughout the entire interface, in fact, touch friendly buttons may sometimes look clumsy. This pattern is valuable for when we have any element, either text or image, that by itself is not large enough to be triggered without effort; or when designing big buttons may go against what was envisioned for the visible design of the interface. The solution involves having the visible part of the element surrounded by an invisible padding area that also works as a target, i.e., only a portion of the target is visible, the rest is hidden. One limitation of this pattern is that it cannot be used when we have several touchable elements side by side, without creating additional white space between them. We should always try to provide the correct affordances for the interface, but because of the small scale that the visible component can reach, the simpler design, or just because users were not expecting that elements that small would work as buttons, it may happen that these elements are not be perceived as touchable. Microsoft (2012) recognizes this downside and recommends that the graphic component should be at least 4.2mm.

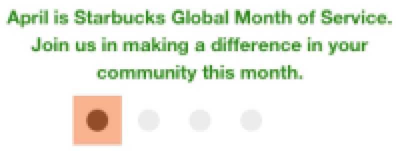

April is Starbucks Global Month of Service.
Join us in making a difference in your
community this month.

Figure 31. ICEBERG TIP on Starbucks <www.starbucks.com>, the color overlay represents the real target.

Starbucks — Figure 31 — uses small circles with a diameter of 16px as buttons for navigating through a SLIDESHOW. Those buttons do not look large enough for being easily touched; however, the real target is a 36px square, which improves considerably its efficacy, though, it is still below what is usually recommended by TOUCH FRIENDLY TARGET.

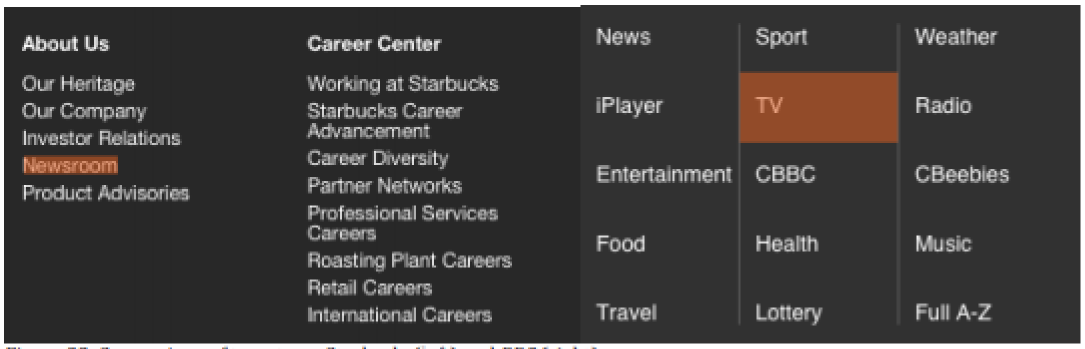
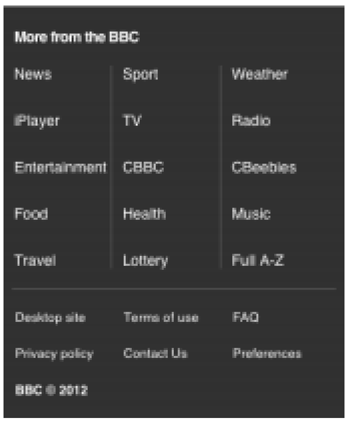


Figure 32. Comparison of targets on *Starbucks* (left) and *BBC* (right).

On the Figure 32, on the left image, we have a menu with links that could be improved by using this pattern. The text is too small, which by itself is not necessarily a problem, but the target occupies the same space as the text, in this case, as low as 8 pixels height; and there is not adequate space between links, which makes it almost impossible to hit the intended target on the first try. On the contrary, on the right image, we have a comparable menu, but here, while the visible part is still small, links have plenty of space between them and the real target occupies the maximum space possible.

Examples



More from the BBC

News	Sport	Weather
iPlayer	TV	Radio
Entertainment	CBBC	CBeebies
Food	Health	Music
Travel	Lottery	Full A-Z
Desktop site	Terms of use	FAQ
Privacy policy	Contact Us	Preferences

BBC © 2012

http://m.bbc.co.uk/news

http://www.starbucks.com
http://www.pepatostudio.com
http://www.earthhour.fr

Related Patterns

– TOUCH FRIENDLY TARGETS

PATTERN: #105
4.20 Dynamic Filtering

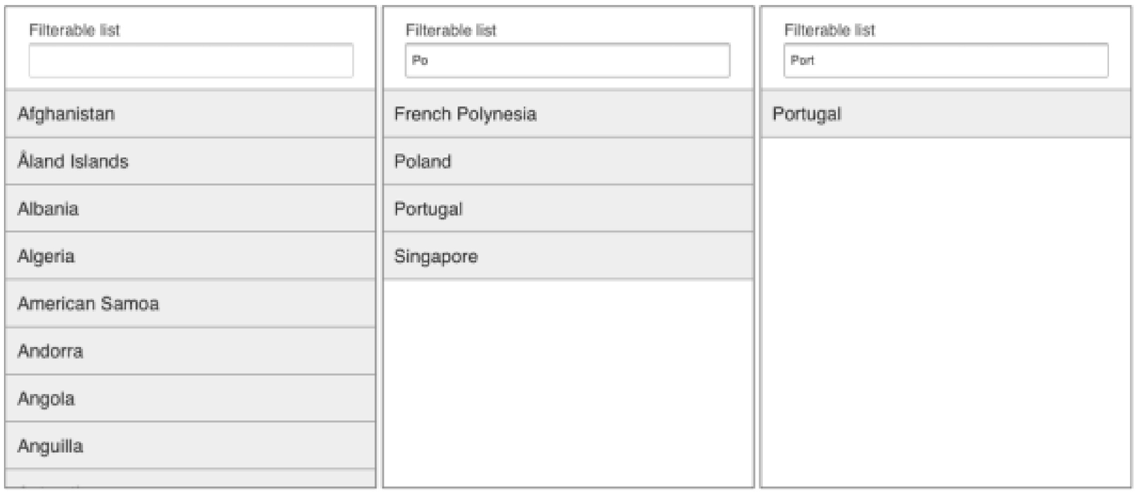


Figure 33. DYNAMIC FILTERING.

Problem
Searching through an extensive list of items within a page can be a tiresome task. Likewise, searching on a long dataset, especially if the user does not remember exactly the search term, can also be very time consuming.

Solution
Provide a search form that dynamically filters the results as the user is typing. *** The idea for the DYNAMIC FILTERING can be found in patterns such as, Dynamic Search (Neil 2012) or Search Within (Hoober and Berkman 2011), and you can implement it on forms in order to present faster results, by minimizing the users' need to type and scroll. Unlike an explicit search that forces users to type the complete search term and press a button confirm it, with a DYNAMIC FILTERING they can get the intended result by typing only a few letters. Because users do not need to type everything, this pattern can also be helpful for cases when they do not remember accurately the desired query. When the user types a letter the Dynamic Filtering removes entries that do not contain that letter. As the user keeps typing, the system keeps eliminating entries that do not fit the pattern entered.

Examples
http://www.google.com
http://www.bing.com

Related Patterns
- Dynamic Search, in Mobile Design Pattern Gallery (Neil 2012)
- Search Within, in Designing Mobile Interfaces (Hoober and Berkman 2011)

PATTERN: #106

4.21 Clear Entries

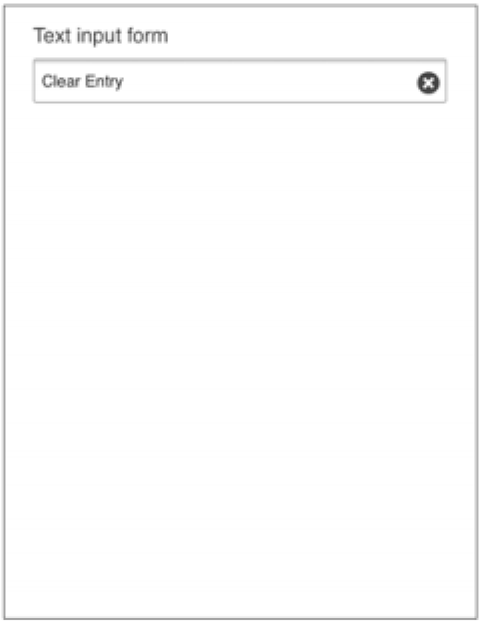


Figure 34. CLEAR ENTRIES

Problem

Although typing on a virtual keyboard is a difficult task, resetting an input form to the default state may be no less easy. Deleting a long string of text letter by letter can be a very tedious error prone task.

Solution

Provide a button that resets the input form with one tap.

***

You should always strive for minimize users' need to input text on mobile. Like text entry, deleting long strings of text can be a very tedious task and propitious to mistakes. While, generally, OS have some sort of method to facilitate the clearing of input fields, such as, faster deleting on long presses, you may still provide a better and faster method for this task. This pattern is based on the patterns Text Clear Button (Tidwell 2011) and Clear Entry (Hoober and Berkman 2011), so you can find additional information there. Therefore, provide a button on all free-text input fields that allows users to quickly remove previous composed text. Place that button inside the input field aligned to the right and farther enough from other Web Design Patterns for Mobile Devices: Page - 41 targets so it is not tapped by mistake. If it is needed you can use and ICEBERG TIP to improve the efficacy of that button. A button with an "x" is succinct, unambiguous and almost a standard so it is usually favored, but if you have the space you can use a label like "Clear" or "Reset".

Examples

http://google.com

http://bing.com

Related Patterns

- Clear Entry, in Designing Mobile Interfaces (Hoober and Berkman 2011)
- Text Clear Button, in Designing Interfaces (Tidwell 2011)