Arpège: Learning Multitouch Chord Gestures Vocabularies

Emilien Ghomi[1,2] **Stéphane Huot**[1,2] **Olivier Bau**[2,3] **Michel Beaudouin-Lafon**[1,2] **Wendy Mackay**[2,1]

{ghomi, huot}@lri.fr olivier.bau@gmail.com {mbl, mackay}@lri.fr

[1]Univ. Paris-Sud & CNRS (LRI) F-91405 Orsay, France

[2]Inria F-91405 Orsay, France

[3]Disney Research, Pittsburgh Pittsburgh, PA 15213 USA

ABSTRACT

This paper presents *Arpège*, a progressive multitouch input technique for learning chords, as well as a robust recognizer and guidelines for building large chord vocabularies. Experiment one validated our design guidelines and suggests implications for designing vocabularies, i.e. users prefer *relaxed* to *tense* chords, chords with fewer fingers and chords with fewer *tense* fingers. Experiment two demonstrated that users can learn and remember a large chord vocabulary with both *Arpège* and cheat sheets, and *Arpège* encourages the creation of effective mmnemonics.

Author Keywords

Gestural interaction; chords; progressive learning technique; tabletop interface; multitouch interaction; gesture vocabulary

ACM Classification Keywords

H.5.2. Information Interfaces and Presentation: User Interfaces - Graphical user interfaces; Input devices and strategies.

INTRODUCTION

Although multi-finger interaction is readily available on multi-touch devices, the number of multi-finger gestures used in commercial systems and reported in the research literature [18, 22] is actually very small. We argue that larger gesture vocabularies can increase expressive power and have the potential for enriching the user experience. Unfortunately, users find large gesture sets correspondingly more difficult to learn. We are thus interested in how to create a large chord vocabulary that is both *comfortable* to perform and *easy to learn*.

This article focuses on a particular case of multi-finger gestures – *chords* – in which the user places two or more fingertips on a multi-touch surface and then lifts them in unison, with no additional movement. The most common use of chords is, of course, in music. Piano keyboards are adapted to the human hand and even novices can learn to perform basic chords simply by watching expert piano players [12]. Many piano instruction methods support step-by-step understanding of the structure of chords and highlight the similarities and differences among them [10]. We argue that novices can learn chord gestures and transition to expert performance in the same way, if we ensure the chords are *understandable* and *comfortable* to perform and if we provide a *progressive method* that helps novices learn how to perform the chords.

After presenting related research, we present chord design guidelines based on studies of the motor abilities and biomechanical constraints of the human hand. We next describe how to create a large, comfortable chord vocabulary, followed by an experiment that assesses the perceived *understandability* and *comfort* of a chord set derived from our guidelines.

We introduce *Arpège*, a contextual dynamic guide inspired by teaching piano, in which users progressively lay down each finger of a chord. *Arpège* gives feedforward, to reveal remaining potential commands, and feedback, to help correct finger positions. Novices can thus safely explore the chord vocabulary, and, due to progressive memorization, transition smoothly from novice to expert performance [19]. We describe a second experiment comparing *Arpège* to a *cheat sheet* similar to FingerWorks[1], the most common static method for learning chords, and conclude with implications for design.

RELATED WORK

Although Engelbart et al. introduced a five-finger chord keyboard for expert computer users [7] in the 1960's, few modern multitouch systems use chords. Instead, they focus on the number and motion of touches[2] and ignore the relative position of each touch. Our goal is to define specific guidelines for generating a large vocabulary of usable chords.

Designing chord vocabularies

One strategy for designing multi-finger gestures is to ask users to design their own [22]. Morris et al. [18] used this method, "*first portraying the effect of a gesture and then asking users to perform its cause*". The resulting gesture set cannot really be considered "natural", since participants were likely influenced by existing commercial multi-finger gesture vocabularies whose effectiveness and comfort have not been tested. Also, even though these gestures are easy to understand, can be performed without training, and map to existing

ITS'13, October 6–9, 2013, St. Andrews, United Kingdom.

http://dx.doi.org/10.1145/2512349.2512795

[1] *http://web.archive.org/web/20031013033746/http://www.fingerworks.com/pdf/TouchStream_QuickGuides.pdf*

[2] *http://www.billbuxton.com/input06.ChordKeyboards.pdf*

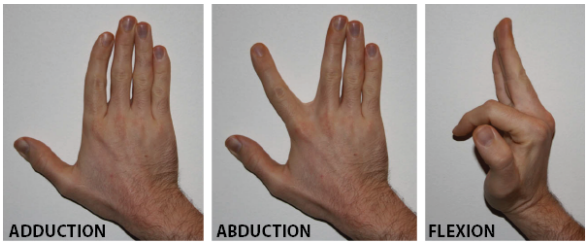

Figure 1. The index finger performs three common finger movements: *adduction*, *abduction* and *flexion*

commands, the command vocabulary is very small and favors novice rather than expert performance.

Most multitouch interaction techniques detect how many fingers touch the surface, but do not consider the relative positions of those fingers [2, 15, 23]. A notable exception is the Multitouch Menu [1], designed to account for relative finger positions while considering the mechanics of the human hand. Our goal is to generalize this approach, taking advantage of the three common finger movements shown in Figure 1 (adduction, abduction and flexion) to design a set of comfortable and easy-to-perform multi-finger chords.

Learning chords

Ensuring that chords are efficient for experts is not enough, since we must still ensure that novices can learn them. Early commercial multitouch systems, e.g., FingerWorks, display offset "cheat sheets" with text, images or video instructions showing how to perform each gesture. These summarize the complete vocabulary, but force users to switch context and make complex chords or dynamic gestures hard to reproduce.

Apple uses movie clips to show how to perform multi-finger gestures whereas TouchGhost's [20] two-handed animated character demonstrates each gesture and simulates its result in the context of the application. Both clearly illustrate *what* to do but are slow, since users must watch and interpret the entire animation before performing the gesture.

Dynamic guides break down single-finger gestures, providing progressive, in-context (i) *feedforward* to show the currently available gestures and how to perform them correctly; and (ii) *feedback* to indicate if the gesture has been recognized. For example, Marking Menus [11] help users learn multi-stroke gestures by exposing each successive level of a hierarchy, whereas OctoPocus [3] reveals the possible paths of each remaining potential gesture, based on the user's currently drawn mark. As the user follows the desired path, alternative paths become irrelevant and progressively disappear.

Both are activated as the user pauses, enabling experts to perform gestures quickly, without interruption, while helping novices learn complex gesture vocabularies. Both support the smooth transition from novice to expert performance, since novices who follow the guide perform identically to experts, who simply perform faster, obviating the need for the guide. Dynamic guides are particularly well suited for multitouch surfaces with co-located input and output and offer a promising approach for novices to learn large chord vocabularies.

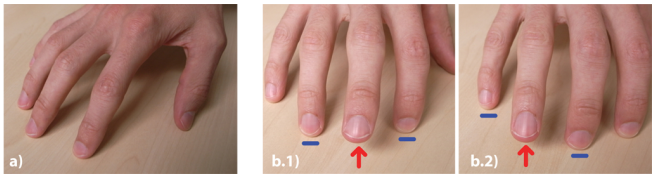

Figure 2. *(a)* Relaxed hand position on a flat surface. *(b)* Lifting the middle or ring finger without its neighbors can be uncomfortable.

ShadowGuides [8] are among the few dynamic guides created for multiple fingers. They help guide finger and hand movements by showing "*the user shadow expected by the system*", including the direction of movement and the eventual evolution of the contact shape. However, the only resource for learning *chord* gestures is an offline "registration pose guide", similar to a cheat sheet. Gesture Play [5] uses physical metaphors to motivate the user to learn and rehearse multi-finger gestures, with a step-by-step "button widget affordance" that supports chord gestures. However, neither system supports multiple finger positions, resulting in a limited chord vocabulary.

The next sections describe our design guidelines for creating a large, but still-comfortable gesture vocabulary, followed by a description of our *Arpège* technique, created to help novices learn and perform large chord gesture sets.

DESIGNING CHORD GESTURES

Fingers have different motor capabilities. For example, the index finger is strong, whereas the little finger is weak and difficult to lift. The thumb can move independently, whereas the middle and ring fingers' movements are restricted by their neighbors. These constraints affect the design space for chord gestures, dictating the movement and placement of each finger so as to avoid uncomfortable finger combinations.

Using Baudel et al.'s terminology [4], we first consider the *relaxed* position of the hand on a multitouch surface. Similar to the reference position when playing the piano, the palm is parallel to the surface, the fingers are slightly curved and all fingertips touch the surface (Fig. 2a). Fingers are neither tight against each other nor spread apart, which reduces muscle tension. Lifting one or more fingers from this relaxed position, but excluding single-finger configurations, leads to $2^5 - 1 - 5 = 26$ possible chords.

Mechanical Constraints For Finger Combinations

In order to identify which of these finger combinations are *comfortable*, we first identify which fingers cannot flex or extend independently of their neighbors. Lee et al. [14] measured the interdependence among fingers during flexion and extension and found that the middle and ring fingers have the strongest angular constraints. Lifting either of these fingers without also lifting its neighbor requires extra muscular effort (Fig. 2b) and leads to our first guideline:

1. Avoid chords in which the middle or ring finger is lifted while its neighbors touch the surface.

Applying this guideline to fingers in a *relaxed* position yields 26 chords, including eight that are potentially uncomfortable.

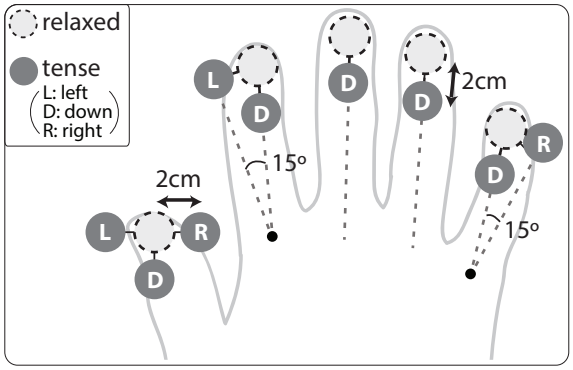

Figure 3. New finger positions derived from guideline 2. Pale grey dots indicate *relaxed* positions. Dark grey dots indicate *tense* positions.

Extended Finger Positions

The human hand has multiple degrees of freedom that permit fingers to perform chords from various positions, such as flexed or spread apart (Fig. 1). Lang et al.'s [13] studies of the mechanical coupling of fingers show that fingers can be ordered by their level of independence, from best to worst: thumb, index, little, middle, and ring finger. The most independent (thumb, index and little fingers) can reach several additional positions, whereas the middle and ring fingers cannot make lateral movements without affecting their neighbors. These considerations led to the second guideline:

2. Create additional chords by flexing or moving the most independent fingers sideways (thumb, index and little finger).

Figure 3 shows the additional chord positions produced by this guideline. The thumb is the most independent, with three additional positions, *down*, *left* and *right*, each 2 cm from the relaxed position. All other fingers can be flexed to reach a position 2 cm below the relaxed position along the finger's up-and-down axis. This distance corresponds to the standard size of keys on a computer keyboard [17]. The index and little fingers can also be abducted by rotating the relaxed position by 15° from the joint between the finger and the palm [16], which generates two additional positions.

Chords derived by displacing the fingers from the *relaxed* position are defined as *tense* [4] since they require additional muscle tension (Fig. 4). By incorporating these positions and assuming that all fingers can be lifted, we can define 720 ($5 \times 4 \times 3 \times 3 \times 4$) unique chords. Removing chords specified by guideline 1 still leaves a large vocabulary of 480 chords that can be performed easily with minimal discomfort.

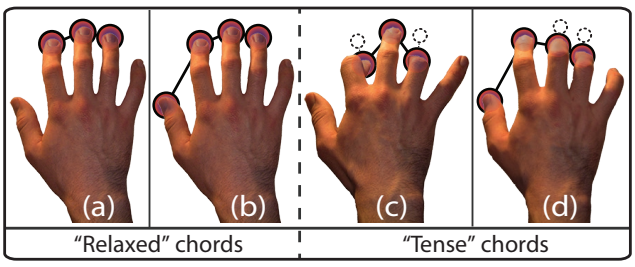

Figure 4. Chords that respect both guidelines: *(a)* and *(b)* include fingers in a *relaxed* position; *(c)* and *(d)* include fingers in a *tense* position.

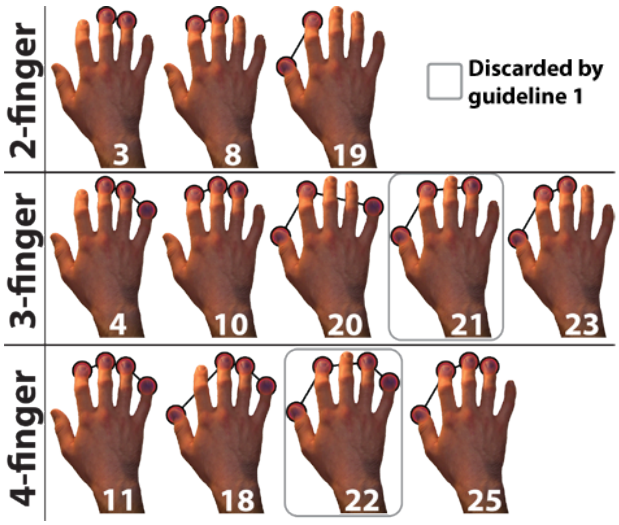

Figure 5. The set of preferred *relaxed* chords from the pilot study.

DESIGNING AND ASSESSING A CHORD VOCABULARY

We conducted a pilot calibration study with 12 participants to determine which of the *relaxed* chords users preferred most (Fig. 5). We created a representative set of 26 *tense* chords (Fig. 6) drawn from these preferences, including three 2-finger chords, ten 3-finger chords, eight 4-finger chords, and five 5-finger chords. We intentionally included a disproportionate number of 3-finger chords, since novices should perform and remember them easily, yet they represent chord-based interaction more clearly than 2-finger chords. We created a test vocabulary of 52 chords, composed of these 26 *tense* chords and the original 26 *relaxed* chords, balanced for number of fingers. We included chords that lift the middle or ring finger in order to test guideline 1.

Experiment one: Assessing the chord guidelines

We conducted an experiment to assess whether and how the above guidelines affect user preferences. We formulated four hypotheses, in which we expect users to prefer:

1. *relaxed* over *tense* chords;
2. chords with fewer fingers;
3. chords with fewer *tense* fingers; and
4. chords that do not lift middle or ring fingers (guideline 1).

Participants

We recruited ten men and two women, median age 27 (range: 21-39), all right-handed, who had not participated in the calibration study. Two had never used a multitouch device, two were casual users, and eight were daily users of multitouch smartphones or tablets.

Apparatus

The software was implemented in Java and run on an Apple MacBook Pro with an external 3M 27" multitouch screen installed horizontally on a table (95 cm high).

Procedure

We used a within-participant design with one main factor (CHORD) to compare RELAXED and TENSE chord vocabularies,

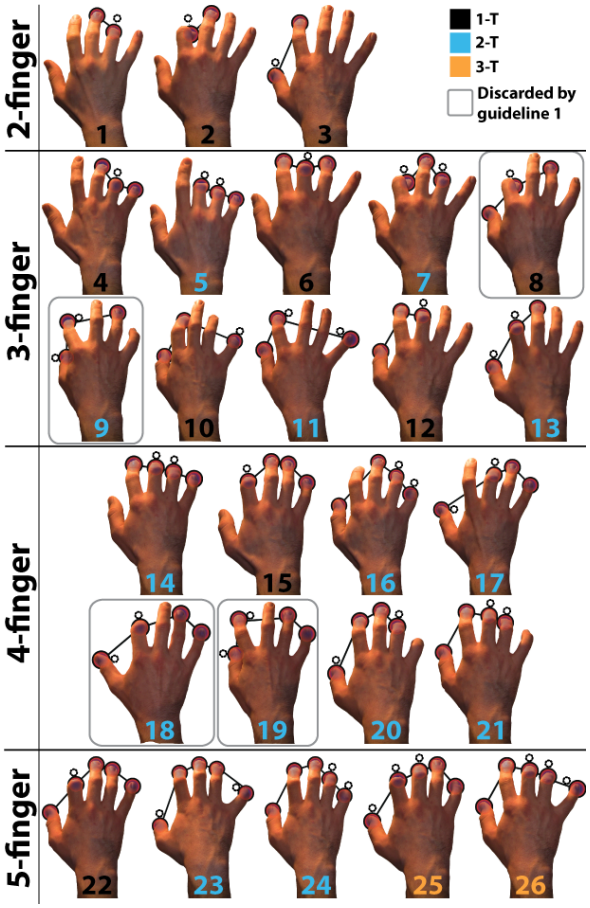

Figure 6. The set of 26 *tense* chords tested in the experiment. Dots indicate the *relaxed* position, the remaining fingers are in the *tense* position. Chords are labelled according to the number of *tense* fingers, e.g., *2-T* chords have two *tense* fingers. Guideline 1 recommends that chords surrounded by a rectangle should be discarded.

each with 26 unique chords matched according to NUMBER FINGERS and NUMBER TENSE FINGERS. Both sets contain equal numbers of chords that do not match GUIDELINE 1.

The experiment lasts approximately 40 minutes, organized into four phases that present the 52 unique chords in the test vocabulary. Each trial displays a picture of a hand performing the required chord, with colored circles to indicate where to place the fingers. For *tense* finger positions, a smaller dashed circle indicates the corresponding relaxed position (Fig. 6). Participants execute chords in the middle of the screen with their dominant hand. The practice phase calibrates the recognizer and ensures that participants can perform all 52 chords.

In phase A, participants reproduce each chord twice and then rate *understandability* – easy to understand and reproduce – and *comfort* to perform, on a five-point Likert scale. Each of the 52 chords are presented once, in random order, blocked in groups of seven. Feedback consists of linked red dots that appear under the fingers, as detected by the recognizer. Phase B presents each chord three times, in random order, blocked in groups of seven, with feedback limited to "success" or "error" messages as computed by our recognizer. Phase C occurs after phase B and is identical to phase A.

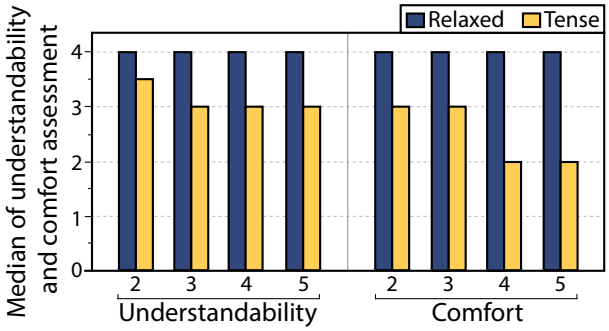

Figure 7. Median ratings of *understandability* and *comfort* for each chord vocabulary (*relaxed* or *tense*) by number of fingers.

Data Collection

We collected two preference ratings (*understandability* and *comfort*) for each chord, resulting in 52 × 2 measures for phase A and 52 × 2 measures for phase C, or a total of 208 preference ratings. We generated accuracy measures to provide feedback but did not analyze them in this experiment.

RESULTS

We analyzed[3] UNDERSTANDABILITY and COMFORT according to CHORD TYPE (*relaxed* or *tense*), NUMBER FINGERS, NUMBER TENSE FINGERS and GUIDELINE, i.e. if the chord follows guideline 1.

Hypothesis 1: Users prefer *relaxed* over *tense* chords.

Wilcoxon tests for CHORD TYPE reveal significant effects for both UNDERSTANDABILITY ($\chi^2(1) = 287.2, p < 0.0001$) and COMFORT ($\chi^2(1) = 282.3, p < 0.0001$) (Fig. 7). However, all median values are 3 or more – *agree* or better on a scale of 0 to 4 – suggesting that participants perceive this 52-chord vocabulary to be sufficiently *understandable* and *comfortable*.

Hypothesis 2: Users prefer chords with fewer fingers.

Kruskal-Wallis tests for NUMBER FINGERS reveal a significant effect of UNDERSTANDABILITY for both *relaxed* ($\chi^2(3) = 14.64, p = 0.0021$) and *tense* ($\chi^2(3) = 26.96, p < 0.0001$) chords, as well as a significant effect of COMFORT for both *relaxed* ($\chi^2(3) = 23.11, p < 0.0001$) and *tense* ($\chi^2(3) = 74.53,$

[3]We used SAS JMP Pro for all analyses.

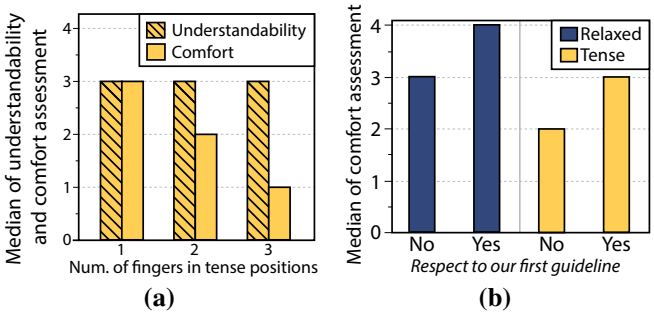

Figure 8. *(a)* Comfort and understandability by total fingers in *tense* positions. *(b)* Comfort according to guideline 1 for each vocabulary.

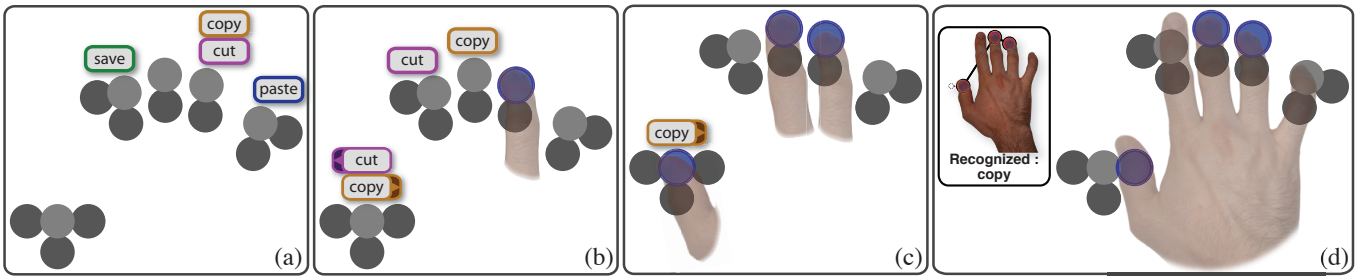

Figure 9. Triggering the *copy* command with *Arpège*. (a) Invoking *Arpège* displays fingerprints representing possible finger positions as well as command labels. (b) Touching the fingerprint under the *copy* label with the ring finger discards *paste* and *save* since they are no longer reachable. (c) If the thumb is laid on the fingerprint corresponding to the relaxed position, the remaining label indicates that it must be moved to the right. (d) The thumb reaches the *right* position, the chord is complete and the user is notified that *copy* will be triggered if all the fingers are lifted simultaneously.

$p < 0.0001$) chords. This effect is greater for *tense* than *relaxed* chords.

Hypothesis 3: Users prefer fewer *tense* fingers.

Kruskal-Wallis tests for NUMBER TENSE FINGERS reveal significant effects for UNDERSTANDABILITY ($\chi^2(2) = 65.68$, $p < 0.0001$) and especially COMFORT ($\chi^2(2) = 147.2$, $p < 0.0001$) (Fig. 8a).

Hypothesis 4: Users prefer chords that do not violate guideline 1: avoid lifting middle and ring fingers if their neighbors touch the surface.

Wilcoxon tests for GUIDELINE reveal a significant effect of COMFORT for *relaxed* chords ($\chi^2(1) = 105.4$, $p < 0.0001$), although not for *tense* chords ($\chi^2(1) = 3.81$, $p = 0.05$), supporting the hypothesis for relaxed chords only (Fig. 8b).

In a post-experiment questionnaire, all participants found it difficult to know exactly where to place their fingers because of the distance between the cheat sheet and the input area. Participants spontaneously developed two strategies for overcoming chord complexity. Seven placed their fingers in succession on the surface, like an arpeggio, and two placed their fingers in the relaxed position before moving them to a *tense* position. This decomposition of chords is a key element of our *Arpège* learning technique, described next. Four participants found 2-finger and 5-finger chords easier to perform than 3-finger and 4-finger chords because they could not easily tell which fingers were involved and what were their appropriate positions.

Discussion

Participants gave high ratings for both the *understandability* and *comfort* of this large chord vocabulary, which is encouraging for real applications. However, these results suggest that using and learning complex chords, especially those involving *tense* positions with three or more fingers, require additional training.

When creating a chord vocabulary, designers should begin with *relaxed* chords, which are perceived as both easier to understand and more comfortable to perform than *tense* chords. These findings also indicate that chords with fewer fingers are more comfortable and should be used for more common commands. Finally, designers should follow guideline 1 and avoid chords in which the middle or ring finger are lifted, since these are perceived as the most uncomfortable and difficult to perform.

ARPÈGE: A DYNAMIC GUIDE FOR CHORD GESTURES

Building on previous work and the results of experiment one, we identified three key requirements:

- *In-context Guidance* – Adapt the dynamic guide to handle chords, with progressive *in-context* guidance to break down the complexity of the gesture, as in OctoPocus [3].
- *Graphical occlusion and visual complexity* – Avoid occlusion from the user's hand and minimize visual complexity while displaying all relevant fingers and their positions (*relaxed*, *right*, *left* or *down*).
- *Chord decomposition* – Decompose chords so that users can place their fingers sequentially or in reference to the *relaxed* position.

Design

Based on these requirements, we designed *Arpège*, a dynamic guide that provides progressive guidance to novice users as they learn chords. *Arpège* displays the current possible chords and how to perform them (feedforward), and indicates which finger positions have been identified thus far or if it has already recognized a complete chord (feedback).

Figure 9 illustrates a simple scenario. Lea wants to use the chord that triggers the *copy* command in a drawing application. Since she is unfamiliar with chords, she double taps the screen to invoke *Arpège*, which displays groups of fingerprints representing all possible finger positions, along with labels indicating the first finger to lay on the surface for each command (Fig. 9a). When Lea places her ring finger on the corresponding fingerprint (Fig. 9b), *Arpège* displays new labels showing the remaining possible commands. Those no longer reachable disappear, e.g., the *save* command that starts with the index finger.

Lea places her middle finger and her thumb on the indicated fingerprints, which highlight as they are touched (Fig. 9c). The *copy* label has a right arrow to indicate that Lea must move her thumb to the right. When the *copy* chord is recognized, a picture appears (Fig. 9d) to show which command will be triggered if Lea lifts all of her fingers at once.

Figure 10. *Arpège*'s cartouches contain labels and arrows.

Using Arpège

Arpège requires a short calibration process for each new user, in order to adapt the dynamic guide and the chord recognizer to the user's hand shape and size. *Arpège* can be invoked in several ways, including performing a specific chord – e.g., the *relaxed* hand position with all fingers down – double tapping the screen or pressing an on-screen or physical button.

Visual Layout

Arpège consists of two dynamic graphical layers: fingerprints and command labels. Fingerprints are displayed as circles, with *tense* positions darker than *relaxed* ones. These targets show comfortable positions for the user and allow the system to identify individual fingers. These visual clues convey sufficient information for all users to reliably place their fingers in the correct position.

Labels consist of colored command names inside "cartouches" that indicate which action to perform for each command. Colors are associated with commands to reduce visual search. Cartouches appear above the fingerprints to specify which finger to place on the surface. Arrows on the side of the cartouche indicate which *tense* position to use, if any, and its appropriate position (Fig. 10). When one finger is involved in several chords, the labels of the corresponding commands are stacked on top of its fingerprints (Fig. 9a&b), in a consistent order across fingers.

How Arpège meets our requirements

Music students learn chords by first playing an *arpeggio*, pressing one key after another. This provides a step-by-step understanding of the chord [10] and helps them progress from basic to more complex chord patterns.

Arpège uses a similar approach, decomposing chords into a sequence of finger placements. The scenario illustrates how *Arpège* offers progressive guidance for an entire vocabulary within an application, using far less screen space than a video or cheat sheet. We limit occulusions by placing labels above the fingerprints [21] and requiring users to begin gestures with the outermost finger. We reduce visual complexity by indicating only the starting position of each chord (Fig. 9a), and displaying the subsequent finger only for the chords that remain possible (Fig. 9b&c).

Commands are triggered when the fingers are in the correct configuration and all the fingers are lifted simultaneously. This prevents issuing a command before the chord is complete. (Note that lifting all the fingers in sequence, like an arpeggio, does not trigger a command.) If one finger is lifted while *Arpège* is activated, e.g., to change a finger placement, the system reverts to the previous state and makes discarded commands available again. The user can thus explore the entire chord vocabulary and discover the corresponding commands by placing, moving and lifting fingers.

Finally, like Marking Menus [11] and Octopocus [3], *Arpège* should help novices transition to expert behavior. Users with intermediate knowledge of chords can also benefit from *Arpège*, for example, to help them remember how to start a chord and then complete the rest by simultaneously placing their remaining fingers. As with ShadowGuides [8], once a chord is recognized, the displayed picture should help users associate the chord with the resulting command.

Implementation

Arpège is implemented in Java in four modules:

- The input module captures touches;
- The output module displays feedforward and feedback;
- The *partial chord recognizer* infers potential target chords as well as required actions for completion, given the current finger positions; and
- The *chord classifier* compares the geometric features of the final chord to the templates in the vocabulary and returns the best match.

The input module uses a TUIO listener[4] that interprets touches, groups them into potential chords, and sends the list of current finger positions to the partial chord recognizer. When all fingers are lifted simultaneously (within 100 ms), the final finger positions are sent to the chord classifier. Visual feedback is implemented with custom Java2D graphics.

Chord templates include two represention. (i) The partial chord recognizer uses an abstract representation with the list of involved fingers and their positions (relaxed, down, left and right). (ii) The chord classifier uses a geometrical representation to describe the normalized distances and angles between the points of a chord. These geometrical features are computed based on the initial calibration of the user's hand and our definition of *tense* positions, i.e. 2 cm translations and 15° rotations.

Partial Chord Recognizer

The partial recognition algorithm provides *Arpège*'s step-by-step feedback and feedforward. The vocabulary is represented as a tree. Each node corresponds to a set of finger positions; its children correspond to the chords that can be reached from it by putting down, moving or lifting a finger. The input module sends an event when a finger reaches or leaves one of the available positions, which triggers a tree traversal. If a node exists for the detected finger movement, the current state of the recognizer and the display are both updated.

Chord Classifier

The chord classifier is independent of the dynamic guide. Our algorithm compares each input chord to templates with the same number of points in the vocabulary. For each template, it computes the sum of the squared distances between the points of all the permutations of the input chord and the template. In addition, it searches for the combination of scale and rotation that minimizes the distance between each permutation of the input chord and each template in the vocabulary.

[4] http://www.tuio.org/

Scale is tested between 0.9 and 1.1 times the size of the input chord, and rotation tolerance ranges from 15° clockwise to 15° counterclockwise. Our tests show that these thresholds permit both efficient chord recognition and sufficient comfort when performing chords. Finally, the algorithm compares the minimum distances between each permutation and each template and returns the best match from the full vocabulary.

Benchmark tests

The performance data from the previous experiment served as benchmark data. All the chords recognized by the partial recognizer were also recognized by the classifier. The chord classifier is thus appropriate for use in the *Arpège* novice mode, since it reliably detects finger positions.

For the performance data from phase B — with no graphical feedback — the classifier correctly recognized 76% of *relaxed* chords and 80% of *tense* chords. Even though novices were not instructed to focus on accuracy, they were able to successfully reproduce the majority of chords.

LEARNING AND RETAINING CHORDS

Before we can test whether or not *Arpège* is superior to a *Cheat Sheet* for learning chords, we must first generate a suitable test vocabulary. Based on our observations from experiment one and the hand mechanics literature, we developed the following rules to assign "penalties" to the 52 chords tested in experiment one:

- Additional fingers make chords more complex to perform and difficult to remember. Three- and four-finger chords thus receive *one penalty* and *two penalties*, respectively.
- The middle and ring fingers are particularly difficult to lift (guideline 1). Chords that lift either of these fingers, together with one or both neighbors, receive *one penalty*. Chords that lift either of these fingers while both neighbors remain on the surface, receive *two penalties*.
- *Tense* fingers are less comfortable than *relaxed* ones. Chords receive *one penalty* for each *tense* finger position.

We calculated the total number of penalties for each chord and classified them as "easy" (zero – two penalties), "medium" (three – four penalties) and "hard" (four – six penalties). Of the *relaxed* chords, 20 are easy and six are medium. Of the *tense* chords, five are easy, 14 are medium and seven are hard. This classification is consistent with the understandability and comfort ratings from experiment one.

We created a set of 12 chords from the four highest-rated chords in each category. We randomly assigned commands from three lexical categories (cities: *Amsterdam*, *Berlin*, *Paris*, *Rio*; fruits: *Apple*, *Banana*, *Cherry*, *Kiwi*; and objects: *Bike*, *Camera*, *Guitar*, *Shoe*) to these chords (Fig. 11).

Experiment two: Assessing learning and retention

We compared how *Arpège* and a *Cheat Sheet* help users learn to perform chords and remember chord-command pairs. We expect *Arpège* to improve retention, since it provides more information than the cheat sheet about *how* to perform a gesture and *why* it differs from other gestures. We formulated three hypotheses:

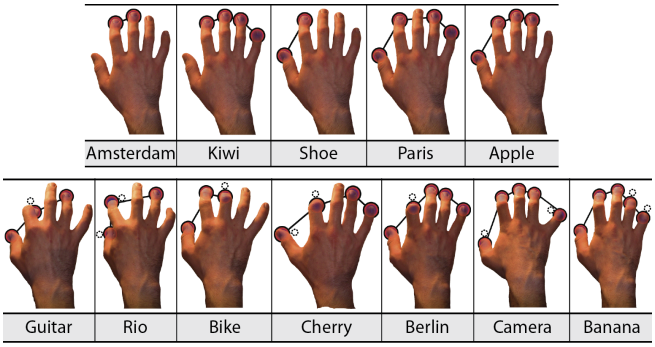

Figure 11. The 12 test chords are mapped to 12 arbitrary commands.

1. Participants can learn and retain a set of 12 chords;
2. Learning with *Arpège* is faster than with a cheat sheet; and
3. Intermediate retention is higher with *Arpège*.

Participants

We recruited 17 men and 7 women, median age 28 (range: 19-41), all right-handed. We allocated 12 to the *Arpège* group and 12 to the *Cheat sheet* group. Four participants from each group also participated in experiment one. Eight members of the *Cheat sheet* group use multitouch devices daily and four had limited or no experience. The *Arpège* group had less experience with multitouch devices. Only one was a daily user, eight had limited experience and three had no experience.

Apparatus

The hardware and sofware are identical to that of experiment one. Both help systems, *Arpège* and the cheat sheet, are invoked with the same physical button, a Griffin PowerMate USB Multimedia Controller. This avoids interference between invoking help and specifying chords and reduces contextual differences between them, since software cheat sheets are normally reached via menus. The experiment design thus favors cheat sheets.

Procedure

We used a between-participant design with one primary factor (TECH) to compare the *Arpège* and *Cheat Sheet* help systems. Day one begins with a five-minute practice session to calibrate the recognizer and ensure that participants can physically execute the 12 chords. *Learning* sub-sessions teach the 12 command-chord pairs; *Evaluation* sub-sessions test retention and are presented over two days. The experiment lasts about 40 minutes on day one and 10 minutes on day two.

Learning trials display a single command-chord pair, with either the corresponding *Arpège* dynamic guide or the *Cheat Sheet* photograph of a hand. The participant then tries to perform the corresponding chord.

Evaluation trials display a command name. Participants are told to execute a chord only if they are sure they know it. If they remember, they perform the chord *without* invoking help and then confirm their choice from a list of the 12 possible chords (Fig. 12), to distinguish performance and retention

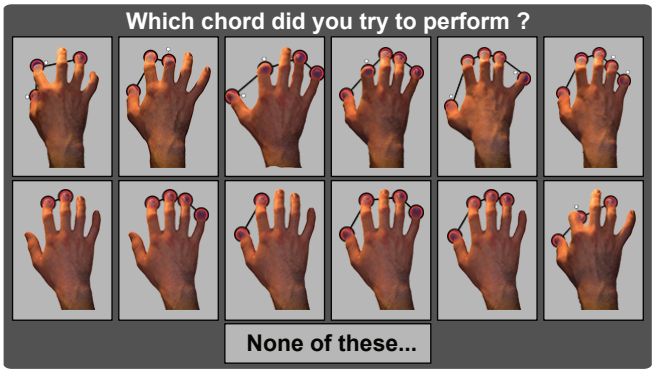

Figure 12. Confirmation screen after performing a chord.

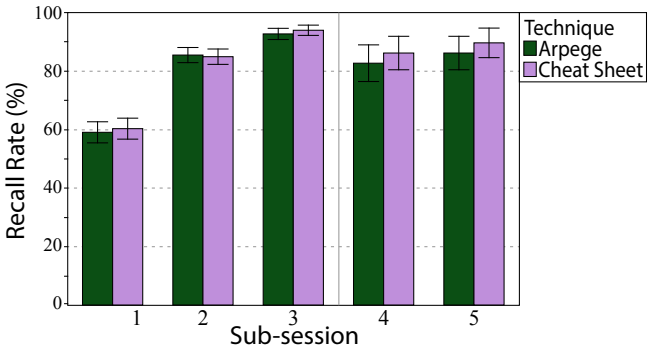

Figure 13. Recall rate for both techniques by sub-session.

errors [9]. Note that this also increases the participant's exposure to the chords, which increases overall learning. When participants forget, they push a physical button to invoke either the full *Arpège* dynamic guide or the *Cheat Sheet* static photograph of all 12 chords. All trials end with a "success" or "error" message.

The *Learning* sub-session presents each chord twice, for a total of 24 trials. In first three *Evaluation* sub-sessions, the frequency of presentation is determined according to a Zipf distribution (13,13,6,6,4,4,3,3,2,2,2,2), to simulate the varying frequencies of commands in real applications [9]. Frequency assignment is counterbalanced across participants, resulting in the same overall number of trials for each command, and is randomly distributed in over consecutive blocks.

On day one, participants see the complete set of commands three times according to the frequency distribution, for a total of 180 *evaluation* trials grouped into 12 blocks of 15 trials. On day two, each command appears twice, presented in random order, in two blocks of 12 trials.

Data Collection

In addition to the post-experiment questionnaire, we recorded three quantitative measures:

1. *Success rate*: Percentage of recognized chords among non-help trials, i.e. participants who believe they know the correct chord perform it correctly.

2. *Recall rate*: Percentage of correct confirmation answers, i.e. participants who believe they know the correct chord specify the correct command, regardless of whether or not they can perform it correctly.

3. *Help rate*: Percentage of help trials, i.e. participants who forgot the chord invoke help and then perform it correctly.

RESULTS

Sphericity tests were validated, indicating that unadjusted univariate F tests are appropriate [6]. We performed repeated-measures ANOVA.

Learning condition

Hypothesis 1: Participants can learn and retain a set of 12 chords.

All participants successfully and rapidly learned a large vocabulary of 12 chord gestures whatever the technique they used. We found a significant effect of SubSession on the *recall rate* ($F_{2,44} = 136.01, p < 0.0001$). The Tukey HSD post-hoc test reveals that the *recall rate* is significantly higher for each successive SubSession, indicating that they were quickly able to stop invoking help.

However, no significant effect obtains for Tech on the *recall rate* ($F_{1,22} = 0.024, p = 0.879$) or the Tech × SubSession interaction effect ($F_{2,44} = 0.120, p = 0.887$). Participants gave appropriate answers to the confirmation question more than 80% of the time in the second sub-session, i.e., after about 10 minutes, and more than 90% of the time by the third sub-session.

Hypothesis 2: Learning with *Arpège* is faster than with a cheat sheet.

We found a significant effect of SubSession on the *success rate* ($F_{2,44} = 146.6690, p < 0.0001$). The Tukey HSD post-hoc test reveals that the *success rate* is significantly higher for every successive SubSession, but there is no significant effect of Tech on the *success rate* ($F_{1,22} = 0.301, p = 0.589$), and no Tech × SubSession interaction effect ($F_{2,44} = 0.721$, $p = 0.492$). For the trials of the third sub-session in which participants did not invoke help, more than 80% of the chords they performed were recognized by the classifier.

Hypothesis 3: Intermediate retention is higher with *Arpège*.

For the *help rate*, an ANOVA reveals a significant effect of SubSession ($F_{2,44} = 98.88, p < 0.0001$), no effect of Tech ($F_{1,22} = 0.012, p = 0.914$), and no Tech × SubSession interaction effect ($F_{2,44} = 0.308, p = 0.736$). The Tukey HSD post-hoc test reveals that the *help rate* is significantly lower for every successive SubSession, decreasing to less than 5% of the trials in the third sub-session.

We found a significant effect of SubSession on the *recall rate* ($F_{4,88} = 68.94, p < 0.0001$) in five blocks of trials over two days. The Tukey HSD post-hoc test reveals that the *recall rate* in the first block is significantly lower than that of the four subsequent blocks (Fig. 13). Retention on day two remained high, at the same level as at the end of day one. Figure 13 shows that the participants remembered more than 80% of the chords on the second day.

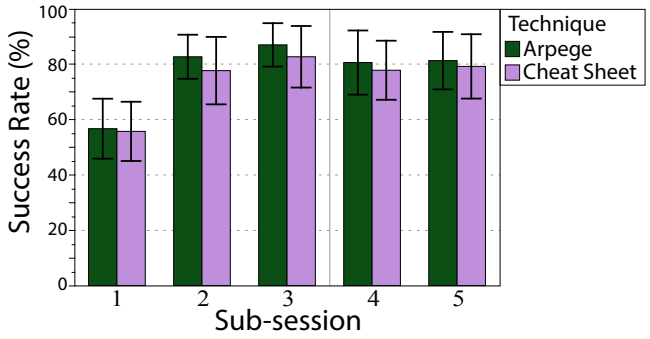

Figure 14. Success rate for both techniques by sub-session

We found a significant effect of SUBSESSION on the *success rate* ($F_{4,88} = 56.85, p < 0.0001$). The Tukey HSD post-hoc test reveals that the *success rate* in the first block is significantly lower than in later blocks (Fig. 14). The number of chords recognized by the classifier on day two is not significantly different from that at the end of day one.

Qualitative Results

The post-experiment questionnaire asked participants about their strategies for memorizing chord gestures and how they mapped to commands. Several participants created semantic associations between the hand position and its command name. For example, two participants in the *Arpège* group thought the "Paris" gesture represented the Eiffel tower and one participant in the *Cheat Sheet* group described the gesture for "guitar" as "like striking a guitar string". Five participants from the *Arpège* group and eight participants from the *Cheat Sheet* group reported using semantic associations, with 20 and 11 total associations respectively.

Two participants from the *Cheat Sheet* group felt that the cheat sheet conveyed too much information at a time. Four participants from the *Arpège* spontaneously told us that they were impressed by their own skill progression with respect to both memorization and ease-of-use. No participants in the *Cheat Sheet* group made such observations.

Discussion

Participants were able to successfully use both learning techniques to rapidly learn a large vocabulary of chord gestures and quickly stop using help, which suggests that larger, more complex chord vocabularies are suitable for real systems. In addition, participants were able to successfully retain what they learned, since performance was very high (over 90% correct) and not significantly different from one day to the next.

Given the high overall performance rates, it is perhaps not surprising that we did not find a significant difference between *Arpège* and *Cheat Sheet*s. However, our cheat sheet has several advantages over conventional ones. First, we show actual photographs of the hand, which suggests how to perform the correct chord position. Second, we provide additional information about relative finger positions, since the dashed white circles representing *relaxed* positions serve as a reference for *tense* positions. Third, our cheat sheet is displayed automatically, without requiring the user to switch context or navigate to a separate menu. Note too that the *Arpège* group produced almost twice as many mnemonic associations, which suggests the potential for a longer-term retention rate.

CONCLUSION

This paper addresses the problem of how to design large chord gesture sets and teach them to users. We present two guidelines for creating chord vocabularies that respect the biomechanical constraints of the hand, including the concept of *relaxed* versus *tense* fingers. The first guideline indicates which sets of fingers to avoid lifting within a chord and the second guideline describes how to move different fingers in order to generate comfortable new chords.

We identify a set of 480 possible chord gestures, from which designers may create a wide variety of chord-based applications. We propose a strategy for comparing and evaluating chords through the use of rules and "penalties", which can help designers select appropriate chord vocabularies that are both comfortable and efficient.

We conducted an initial experiment to evaluate these guidelines. We verified that *relaxed* chords are more understandable and comfortable than *tense* chords. Adding additional fingers, especially tense ones, makes chords more difficult to understand and perform.

We introduced *Arpège*, a dynamic guide that offers a progressive approach to learning and performing chord gestures. *Arpège* breaks down the complexity of a chord and can ease the transition between novice and expert use. *Arpège* uses very little screen real estate, making it appropriate not only for large tabletops, but also for smaller tablets and smartphones. To use it, users need not switch contexts but can learn and perform all their actions in place.

Arpège ensures that the user performs the chord appropriately and fluidly accommodates "undo" if the user makes an error, which encourages novices to freely explore the command vocabulary. Intermittent users, who may only remember some commands, can quickly perform the chords they know and only spend time learning those they have forgotten.

We conducted a second experiment that demonstrated that users can easily learn a relatively large set of chords associated with randomly determined commands. Although we did not find significant performance differences between *Arpège* and a cheat sheet, we demonstrated that both can quickly teach command sets that can be remembered from one day to the next. *Arpège* also inspires a richer set of mnemonic strategies and should perform even better in real-world settings, since it does not require users to switch context while performing a chord.

In future work, we plan to explore how well *Arpège* scales with larger chord vocabularies, which will require new strategies for handling visual complexity in the early stages. For example, a hierarchical version of *Arpège* could map command categories to individual fingers, which would be successively revealed as each finger is placed on the surface. We

are also interested in conducting a long-term field study with *Arpège*, to determine its relevance in a real-world setting.

In conclusion, *Arpège* offers a strategy for helping designers to incorporate relatively large sets of multitouch chord gestures into their applications. Designers can use our guidelines to generate chord vocabularies that are both understandable and comfortable. Users can also enjoy exploring and learning these new chord sets, in the context of their daily activities.

ACKNOWLEDGMENTS

We would like to thank to Clément Pillias, who helped implement *Arpège*'s chord classifier, and Mathieu Nancel who helped design the visual feedback.

REFERENCES

1. Bailly, G., Demeure, A., Lecolinet, E., and Nigay, L. Multitouch menu (mtm). In *Proceedings of the 20th International Conference of the Association Francophone d'Interaction Homme-Machine*, IHM '08, ACM (2008), 165–168.
2. Bailly, G., Lecolinet, E., and Guiard, Y. Finger-count & radial-stroke shortcuts: 2 techniques for augmenting linear menus on multi-touch surfaces. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, CHI '10, ACM (2010), 591–594.
3. Bau, O., and Mackay, W. E. Octopocus: a dynamic guide for learning gesture-based command sets. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*, UIST '08, ACM (2008), 37–46.
4. Baudel, T., and Beaudouin-Lafon, M. Charade: Remote control of objects using free-hand gestures. *Comm. ACM 36* (1993), 28–35.
5. Bragdon, A., Uguray, A., Wigdor, D., Anagnostopoulos, S., Zeleznik, R., and Feman, R. Gesture play: motivating online gesture learning with fun, positive reinforcement and physical metaphors. In *ACM International Conference on Interactive Tabletops and Surfaces*, ITS '10, ACM (2010), 39–48.
6. Davis, C. *Statistical Methods for the Analysis of Repeated Measurements*. Springer, 2002.
7. Engelbart, D. C. Augmenting human intellect: A conceptual framework. *SRI Summary Report AFOSR-3223* (1962).
8. Freeman, D., Benko, H., Morris, M. R., and Wigdor, D. Shadowguides: visualizations for in-situ learning of multi-touch and whole-hand gestures. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, ITS '09, ACM (New York, NY, USA, 2009), 165–172.
9. Ghomi, E., Faure, G., Huot, S., Chapuis, O., and Beaudouin-Lafon, M. Using rhythmic patterns as an input method. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, ACM (2012), 1253–1262.
10. Koh, J. *Scales And Arpeggios For Piano - Fingering Method*. Wells Music Publishers, 2010.
11. Kurtenbach, G. P. *The design and evaluation of marking menus*. PhD thesis, University of Toronto, 1993.
12. Lahav, A., Boulanger, A., Schlaug, G., and Saltzman, E. The power of listening: auditory-motor interactions in musical training. *Annals of the New York Academy of Sciences 1060* (2005), 189–94.
13. Lang, C. E., and Schieber, M. H. Human finger independence: limitations due to passive mechanical coupling versus active neuromuscular control. *Journal of neurophysiology 92*, 5 (Nov. 2004), 2802–2810.
14. Lee, J., and Kunii, T. L. Model-based analysis of hand posture. *IEEE Comput. Graph. Appl. 15*, 5 (1995), 77–86.
15. Lepinski, G. J., Grossman, T., and Fitzmaurice, G. The design and evaluation of multitouch marking menus. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, ACM (2010), 2233–2242.
16. Lin, J., Wu, Y., and Huang, T. S. Modeling the constraints of human hand motion. In *Proceedings of the Workshop on Human Motion (HUMO'00)*, HUMO '00, IEEE (2000), 121–127.
17. Miller, F. P., Vandome, A. F., and McBrewster, J. *Keyboard Layout*. Alpha Press, 2009.
18. Morris, M. R., Wobbrock, J. O., and Wilson, A. D. Understanding users' preferences for surface gestures. In *Proceedings of Graphics Interface 2010*, GI '10, CIPS (2010), 261–268.
19. Scarr, J., Cockburn, A., Gutwin, C., and Quinn, P. Dips and ceilings: understanding and supporting transitions to expertise in user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, ACM (2011), 2741–2750.
20. Vanacken, D., Demeure, A., Luyten, K., and Coninx, K. Ghosts in the interface: Meta-user interface visualizations as guides for multi-touch interaction. In *3rd IEEE International Workshop on Horizontal Interactive Human Computer Systems. TABLETOP 2008.*, IEEE (2008), 81–84.
21. Vogel, D., and Casiez, G. Hand occlusion on a multi-touch tabletop. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, ACM (2012), 2307–2316.
22. Wobbrock, J. O., Morris, M. R., and Wilson, A. D. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, ACM (2009), 1083–1092.
23. Zeleznik, R., Bragdon, A., Adeputra, F., and Ko, H.-S. Hands-on math: a page-based multi-touch and pen desktop for technical work and problem solving. In *Proceedings of the 23rd annual ACM symposium on User interface software and technology*, UIST '10, ACM (2010), 17–26.