A Tangible Programming Tool for Creation of Context-Aware Applications

Jisoo Lee1  Luis Garduño2  Erin Walker2  Winslow Burleson2
1School of Arts, Media + Engineering 2School of Computing, Informatics, and Decision Systems Engineering
Arizona State University, Tempe, AZ 85281
{jisoo.lee, luis.garduno, erin.a.walker, winslow.burleson}@asu.edu

ABSTRACT
End-user programming tools, if properly designed, have the potential to empower end-users to create context-aware applications tailored to their own needs and lives, in order to help them break bad habits and change their behaviors. In this work, we present GALLAG Strip, an easy to use mobile and tangible tool that allows users to create context-aware applications without the need of programming experience. It enables programming by physical demonstration of envisioned interactions with the same sensors and objects that users will later encounter in their finished application. After an initial pilot to verify the usability of GALLAG Strip, we conducted a user study to evaluate the effects of tangible programming in terms of ease of use, engagement, and facilitation of the ideation process. We found that tangibility has both benefits and drawbacks, and suggest a mixed tangible and non-tangible approach for better user experience.

Author Keywords
Context-aware computing, end-user programming, mobile programming, tangible programming.

ACM Classification Keywords
H.5.2 User Interfaces -- Interaction styles.

General Terms
Human Factors; Experimentation.

INTRODUCTION
Improvement of individuals’ lifestyles has been one of the main goals of ubiquitous computing in the home. An increasingly large number of systems are being developed in the HCI community to promote behavior change in a variety of domains such as physical health [16, 20], affective stability [23] and energy conservation [1].

Fogg, a pioneer of persuasive technologies, asserts that a user will not engage in a target behavior without an appropriate trigger, even if the user has high motivation and ability [11]. A context-aware application can promote the improvement of user habits and behaviors by providing the required triggers at the right time, as evidenced in the work by Intille et al. [13, 20]. However, these contextual cues will likely be highly idiosyncratic and therefore require individuals to conduct self-experimentation to find the best triggers to foster the desired behavior change of interest.

There is a need for a system that allows end-users to create applications that are more relevant to their lives and ultimately more effective at promoting behavior change. To address this issue, we developed GALLAG Strip, a visual, mobile, and tangible programming tool. GALLAG Strip allows end-users who have little or no programming experience to create context-aware applications that integrate sensing and actuating components to work for their particular needs. We intend to facilitate end-users in self-experimentation with strategies for creating better behavioral routines. A number of studies on end-user programming for context-aware applications suggest the importance of end-user involvement. Users have intimate knowledge about their ever-changing activities and environments. Therefore, they are often better positioned than a hired programmer to design context-aware software [7, 8, 24].

To allow end-users to create their own applications without previous programming skills, we employed a visual programming approach in developing GALLAG Strip. We were inspired by some previous systems that use a simplified menu-based or metaphor-based GUI interface [7, 15, 24] to allow end-users to specify applications visually without requiring them to write any code [7]. Furthermore, we hypothesized benefits of mobility and tangibility in end-user programming for context-aware applications.

While most end-user programming tools for context-aware applications imply desktop computers as their usage environments, GALLAG Strip is mobile. Its smartphone based user interface allows users to roam within a sensor-instrumented space while programming their applications. This design decision was inspired by our observation of participants’ use of environments while they brainstormed ideas for context-aware applications, demonstrating the potential utility of placing users in a location to which a target behavior is related. This mobile approach is validated by values of contextual design, rapid prototyping, and in situ creation acknowledged in the HCI community [21, 22].
Going beyond mobility, GALLAG Strip is tangible, in that its interface enables programming by physical demonstration of envisioned interactions with the same sensors and objects that users will later encounter in their finished application. Users manipulate objects that are part of their daily lives rather than the models (e.g., abstractive blocks [14] and miniatures of real-world objects [2]) that most tangible interface systems have adopted. We call this approach real-world tangibility. It may be frequently subtle for a person to identify the contextual cues that trigger their behaviors, especially when they are habits that occur in an automatic way [25]. Therefore, we assumed that users may be better reminded by doing.

While there are a few studies that propose end-user programming tools integrating real-world tangibility [6, 29], no substantial evaluation has been conducted to examine its effects on the user’s experience. Therefore, we conducted a controlled experiment, comparing it with two other conditions (mobile and non-tangible interface, and non-mobile and non-tangible interface), regarding ease of use, engagement, and the user’s ideation experience.

In this paper, we describe GALLAG Strip’s design and user experience. Then, we present results of a preliminary study to verify the basic usability of the system. Next, we describe our controlled experiment, findings, and limitations. We present design implications for end-user programming tools employing real-world tangibility.

RELATED WORK

Visual Programming Tools for End-User Creation of Context-Aware Applications

There has been considerable research and commercial efforts to enable users with little or no technical expertise to prototype context-aware applications. The majority of these efforts have employed visual programming methods [7, 15, 24, 28] by using either metaphor-based GUIs or simple input [24]. One such tool is the work by Humble and Crabtree [15], with their GUI based in the “jigsaw puzzle” metaphor. They allowed users to connect digital jigsaw-puzzle-like components that represented sensors and devices in various left-to-right combinations to form expressions. They believed that although their linear programming model constrained users in terms of expression possibilities, it allowed for easy reconfiguration and helped users to have a better sense of the information flow. Similarly, Truong et al. employed a GUI based in a magnetic poetry metaphor for CAMP [24], allowing users to define context-aware applications through the arrangement of fridge-magnet-like words. They found that users tended to define their applications in terms of high-level goals, rather than low-level details like devices and sensors. With a more traditional PC-based GUI in iCAP [7], Dey et al. enabled users to create context-aware applications by selecting menus and dragging and dropping graphical elements, like objects, activities, locations, people and time. In their system, users arrange these elements to define if-then rules with “and” and “or” conditional operators. Recently, a project called Twine [28] is trying to bring to market a user-programmable wireless module with embedded sensors for home automation. Their approach is a visual, rule-based web editor that allows users to create if-then rules to program sensor modules. Empowered by crowdfunding, Twine has raised more than half a million dollars in just over a month -- an indication of the strong current demand for sensor-based end-user programming.

Tangible Programming Tools

Several studies have developed tangible tools for context-aware programming. For example, SiteView by Beckmann and Dey [2], allowed users to build rule-based applications for home automation through tangible interaction with physical objects placed on a small-scale floor plan. Their system used RFID and a top-mounted camera to capture the rules that users wanted to program, and an environmental display to show images of how the real environment (represented by the floor plan) would look when the rules were applied. Beckmann and Dey describe how their interface is intuitive and lowers the programming difficulty for novice users.

While use of models is dominant in tangible interface systems like SiteView [9], some researchers have explored the use of real environments for context-aware programming. Chin et al. [6] proposed PiP (Pervasive Interactive Programming). It is an if-then rule system that lets users show the behaviors that they intend to program through physical interaction with a sensed environment. The programming interface in PiP was composed of several networked devices (e.g., lamp, phone, fridge), and a PC-based GUI called the PiPView which allowed users to define the devices that can be used and their capabilities. Users had the choice to program their applications solely through physical demonstration, through the GUI, or a combination of both. They report that the majority of the participants (72%) preferred to program through physical interactions. Likewise, the HomeMaestro project by Salzberg [27], allowed users to move about a sensed environment and interact with physical artifacts to program context-aware applications using a mobile phone. The HomeMaestro prototype uses Microsoft’s HomeOS [29] and focuses mainly on home automation. Our work is parallel with these two systems as all three employ real-world tangibility as a primary interface method, focusing on the potential benefits of its intuitive quality. However, there has been little, if any, effort to examine the effects of this approach through controlled experiments and in-depth analysis. Although Chin et al. conducted user evaluation, only their tangible tool was examined, and to our knowledge the HomeMaestro project did not perform a formal evaluation.

Our early user studies demonstrated the need to enhance a tool by reflecting on users’ actual creation processes. During informal interviews with six users, we asked about
their thought process when creating their own applications. We found that they typically pictured a particular location of their house (e.g., kitchen, living room), or looked around the space where they were currently located. Another common process was that they thought of what they usually do during a particular period of time (e.g., morning, evening); that is, they mentally placed themselves in situations of everyday life. Additionally, we observed that interacting with physical objects served as a cue to remind users of situations that they wanted to address. With these findings, taking a mobile and tangible approach to knit application creation closely with users’ environment and behaviors seemed advantageous.

**SYSTEM DESCRIPTION**

**User Experience**
We developed GALLAG Strip, an if-then, rule-based, programming-with-demonstration system with a mobile phone-based GUI. Here we explain the process of creating a new GALLAG application, its structure, and the customization options available.

*Example application*
As a sample application, imagine that a user wants to create an application that will sense when the TV is turned on and triggers an audio cue to remind that reading would be better than watching TV.

When GALLAG Strip starts, the user can see the list of system applications that she created previously. The user can enable and disable some of them according to her needs. To add a new application, she touches the ‘plus’ button and the demonstration screen is presented. This screen is where users demonstrate what they want to program. The demonstration screen has two modes: a recording mode and an edit mode. When creating a new application, the demonstration screen goes directly into recording mode, where the system listens for sensor events triggered by user actions.

Following our example, imagine that the demonstration screen is in recording mode and the user starts to demonstrate her application, so she first turns on the TV (i.e., with the TV’s remote control) and a frame with an icon of a TV turned on appears on the demonstration screen. Because that is the only event the user wants the application to listen to at the beginning, the user touches the pause button to stop recording. When the user touches the pause button, the application goes into edit mode, where a user can review and edit the current application. Now, the user wants to add the audio cues to remind her that she should read instead, therefore the user touches the plus button to add an audio response, selects the sound to play, and then adds another response to make the system speak (i.e., text to speech) the phrase: “You should read instead of watching TV”. At this point, the user has added two response frames (audio cues). Next, the user decides to make the application sense when she turns the TV off and be rewarded with an achievement sound cue. So the user touches the record button, the demonstration screen goes to recording mode again, the user turns off the TV using the TV’s remote control, and an action frame, with an icon of a TV turned off, is appended at the end of the application (see Figure 1). The final frame the user wants to add is an achievement sound as a reward for turning off the TV, so the user touches the pause button to switch to edit mode, then touches the plus button to add a response and selects the achievement sound (see Figure 2).

![Figure 1. Adding action frame in recording mode.](image1)

![Figure 2. Adding a response frame.](image2)

When the user finishes creating the application, she touches the save button and the system configures itself to do what the user just programmed. After the application has been configured in the server, it is ready to be run and can be tested simply by interacting with the sensed object (i.e., the TV) and performing actions previously defined in the application.

*Types of frames*
We employed a comic strip metaphor in our mobile phone GUI, where states are represented with frames. In our GUI, a GALLAG application is represented through a sequence of frames. We call this sequence of frames the “application strip” and it can have three types of frames: action, response, and time-date.

Action frames represent the user’s actions within the sensed space and are shown as blue frames in the application strip. These frames have a default text label and image depending on the type of sensor (see Figure 3).
Response frames represent actions that the system will perform and are set by the user. This type of frame is shown in orange and has a text label and image related to the type of response selected. Response frames can also have an additional parameter that is displayed in text above the frame’s image.

Time-date frames are conditions set by the user and they constrain the application’s execution to a particular time and/or date. These frames are shown in green and show the selected date or time as their text label. Time frames additionally have a parameter to show the selected days of the week (see Figure 4). Time and date frames can be combined to create conditions based both on a date and a time; that is, an application can have up to two date and time frames.

User customization
Action frames are displayed with a default text label and image depending on the sensor being activated. Users can customize them by changing the text and taking a picture with the phone’s built-in camera to make it easier to understand. Figure 5 shows an example, where a frame is represented with a captured book image and user-typed label, changed from a default motion sensor image and text.

Design Rationale
We designed GALLAG Strip as a visual programming tool. A visual representation of a program’s structure is easier to understand than in text form, especially if the programs are short [17, 19]. We target creation of relatively small and simple applications. Application scenarios generated by users in several prior studies [7, 24] tended to be small and simple, not containing nested loops or nested conditionals [18], and therefore are relatively easy to represent visually. In designing the graphic interface, we employed a comic strip metaphor (i.e., showing action states in a sequence of frames), inspired by the work of Modugno and colleagues in Pursuit [17].

Users define their applications in a linear fashion, using simple if-then conditions. In developing a tool as an attempt to support end-user experimentation with behavior change, we chose to start with simple but essential programming logic. In our early field studies, we found that participants quite frequently generated application scenarios that just involved simple if-then rules [7]. For example, "it plays a 2-3 minute song every time we walk by the dishwasher, with the intent of suggesting we clean just until the song ends." One that is considered fairly necessary next to the simple if-then rules in context-aware applications is logic for temporal relationships [12]. However, GALLAG Strip does not support this functionality, and thus, scenarios like the following cannot be programmed: “If I keep brushing my teeth for 2 minutes, it plays a kind of cheerful sound.” As a first pass, we incorporated simple and essential functionality to emphasize ease of use, and reserved further expansion for future work.

Implementation
GALLAG Strip has three main components: a physical sensing system, the GALLAG Strip server, and a mobile phone-based GUI.

Users’ interactions with objects and spaces are sensed through X10 [30] and Insteon [31] home automation sensors. The current implementation of GALLAG Strip supports four types of sensors: X10 wireless open/closed magnetic sensors, X10 wireless motion sensors, Insteon LampLinc modules and Insteon SynchroLinc modules. X10 open/closed sensors can be used to sense interactions with a wide variety of objects, like a drawer, a door, a thermostat and other objects for which position or location can be changed (see Figure 6a). X10 motion sensors are used to detect presence of an individual in a specific space.
They are also useful in situations where attaching an open/closed sensor to an object is not desired, like detecting when a user reaches to grab something from a chest with several objects or detecting if a user grabs a book from a shelf (see Figure 6b). Insteon LampLine and SynchroLinc devices are used to detect user's turning on/off lamps and electric appliances such as a TV respectively (Figure 6c).

Figure 6. Sensors supported in GALLAG Strip.

We use commercial home automation software called Indigo [32] for communication with X10 and Insteon hardware, as well as a platform for running GALLAG applications. Indigo receives information from X10 and Insteon devices either through RF or through AC power lines, and can send commands to these devices using these same channels. The GALLAG Strip server was developed so that it provides sensor event information to the phone and configures Indigo when the user saves a GALLAG application in the mobile phone interface. The mobile application was developed to run on the Windows Phone [33] mobile operating system. We explain this mobile interface in more detail in the following section.

USABILITY STUDY

We performed an initial evaluation of GALLAG Strip with novice users. The purpose of this study was to evaluate the basic usability of the system for end-user programming of context-aware applications.

A total of thirteen subjects volunteered to participate in this study, with ages ranging from 21 to 49, six male and seven female. Participants had a variety of educational backgrounds. They were required to have prior general exposure to smartphones. In addition to the thirteen study participants, two members of our research group volunteered to participate as expert users. They were the source of benchmark data to compare the performance of novice users creating applications with GALLAG Strip to expert users programming the same applications using AppleScript code and Indigo configuration. The expert participants had one or more years of exposure to the GALLAG system and were proficient in AppleScript programming. Study sessions were held in a laboratory setting that simulated a living room, with sensors placed around the space (see Figure 7).

The individual, one-hour user sessions consisted of an initial twenty-minute tutorial about GALLAG Strip, followed by four application programming tasks. The first two programming tasks were simple (one if-then rule, two actions, and two responses). By comparing user performance on the two tasks, we planned to test how much participants improved when asked to program an application of the same complexity as the one they just programmed. Thus, the order of the first and second applications was counterbalanced. The third task was more complex (a time-date condition, two nested if-then rules, three actions and three responses). The last task was for users to program an application of their choosing (free-form application), with no restrictions other than a fifteen-minute time limit.

Figure 7. Living room setting and sensors (yellow circles).

We found that average number of errors was low for all of the four applications that users were asked to program and errors decreased after the first application (see Figure 8). All participants completed the applications in a reasonable amount of time and improvement was evident for the second application (see Figure 8). Another interesting finding was that, even though the average number of requirements (and complexity) for the free-form application was higher than the third one, the average time to program each requirement was lower; this shows that GALLAG Strip has a low learning curve, as participants were able to program applications of increasing complexity in shorter amounts of time.

We then compared novice performance to expert performance. Although both experts had to debug their applications to make sure they worked, neither had errors in their final applications. Like the novices, the second application took less time than the first to program. However, the average time the experts required for the third
and more complex application was almost double (16.3 minutes) than the average time that the study participants needed (8.5 minutes). This finding demonstrates that users without knowledge about how to program a GALLAG application are able to program them with GALLAG Strip faster than expert programmers using traditional tools.

**COMPARATIVE EXPERIMENT**

We conducted a second user study to systematically explore how a real-world tangible programming method affects user experience in developing context-aware applications employing contextual cues as means for behavior change.

**Questions**

Based on our preliminary user study experience and comparative studies on tangible and graphical interfaces by Horn, et al. [14] and Xie, et al. [26], we identified three questions exploring the effects of real-world tangibility. Q1: Does a combined mobile and tangible end-user programming environment make use of the programming tool more difficult? Q2: Does it make end-users more engaged? Q3: Does it facilitate ideation processes?

We found that ease of learning by users of tangible systems is not yet demonstrated in research that compares graphical and tangible interfaces. For example, Horn, et al. [14] report that no significant difference was found between their two conditions, and in the study by Xie, et al. [26], participants had more difficulty in the GUI condition. Moreover, usability of tangible interactions in real environments has only rarely been examined. The effect of engagement of GUI and TUI conditions was found the same in both studies by Horn, et al. and Xie, et al., although it has long been claimed that increased engagement is one of the principal benefits of tangible interaction [10, 26]. A shortcoming in relying on existing studies to understand the benefits of tangible programming is that the target audience in most studies has been children.

Mobility in user’s conception of scenario ideas might positively affect users’ creativity in design by making them aware of their surroundings and giving them immediate contextual input during creative activities [21]. For effects of TUI in user’s ideation and creativity, there was no significant difference in the study by Horn, et al. On the other hand, several studies suggest potential benefits due to lower cognitive load and a natural and familiar mechanism [6], intuitiveness [2], and “liveness” [9].

**Conditions**

We compared three conditions: (1) mobile-tangible (MT), (2) mobile-menu (MM), and (3) stationary-menu (SM). We chose a between-subject design, most of all to avoid participants’ potential bias to “pleased the experimenter.” In all three conditions, participants were asked to complete the same tasks in the same laboratory setting as the first study. MT participants programmed their applications using the version of GALLAG Strip described above, by physically interacting with the sensed environment.

Participants in the MM condition programmed their applications through an equivalent menu-based GUI on a mobile device. The menu-based GUI for the MM condition differed from the GUI of the MT condition in that actions are added to the application manually by selecting them from a list of available sensors and related sensing features (e.g., a motion sensor detecting motion or the ends of a magnetic sensor being separated; see Figure 9).

![Figure 9. Manually adding a motion detected action](image)

With this new feature, users in the MM condition were able to create an application completely through the GUI, that is, without needing to physically demonstrate the application. In this mode, action frames and the desired sensors could be added entirely through menu options. Participants were given a list of the sensors with their IDs and a picture so they could locate them in the lab setting, however, they were not required to carry the list while programming. In both the MT and MM conditions, participants were able to move around while creating their applications.

Participants in the SM condition used the same menu-based GUI as the MM condition, but on a desktop computer through a Windows Phone emulator. Thus, they programmed their applications in a stationary manner. SM participants faced away from the living room, but were able to turn their heads and look at the living room setting. Through these conditions we hoped to isolate the effects of being able to move within an environment (as in the MM condition) from having the movement influence the developing program (as in the MT condition).

**Participants**

A total of 36 individuals were recruited through email lists and Craigslist. 17 were female, and 19 were male. Ages ranged from 18 to 39. Participants were required to know the basics of how to use a computer and a smartphone. 19 participants (6 men; 13 women) had non-engineering backgrounds (e.g., dance, industrial design), and 17 (13 men; 4 women) had engineering backgrounds (e.g., chemical engineering, civil engineering, computer science). Participants were randomly assigned to one of the three experimental conditions, for a total of 12 participants per condition. 5 non-engineering and 7 engineering background participants were in the MT condition, 6 non-engineers and 6 engineers were in the MM condition, and 8 engineers and 4 non-engineers were in the SM condition.
Procedure
The one and a quarter-hour sessions began with an initial video tutorial about GALLAG Strip. Participants were then provided with the programming tool corresponding to their assigned condition (i.e., a mobile phone for the MT and MM conditions, and a desktop computer for the SM condition). Participants were also given a printed list of all the sensors available, showing a picture of their location and their assigned sensor ID.

Each participant was shown a set of presentation slides on a computer describing the three applications that we wanted them to program: a simple, a complex, and a free-form one. For the simple application task, participants were asked to program an application with two actions and two responses: if you enter the living room and you turn the TV on, then make the system play the reminder sound and make the system say “Remember to take your pills”. In the complex application task, participants were asked to program an application with one time-date condition, three actions and three responses: if time is after nine in the morning and you turn on the AC and then open the front door, then make the system play the alarm sound, and make the system say “Turn off the AC”; if you then close the door (i.e., if you leave the house), then make the system send an SMS to your mobile phone with the message “You left the AC on!” For the free-form application, participants were asked to think about a personal scenario that they would like to program. They were given a blank piece of paper and they were asked to think for a couple of minutes and then describe the chosen scenario. After they gave their description of their envisioned application, they were asked to program it without time limitations.

After completing the three programming tasks, participants were given a questionnaire to complete on their subjective perceptions of the activity. After participants completed the questionnaire, we conducted an exit interview to get their feedback regarding usability and engagement. The interview lasted an average of fifteen minutes and included questions about fun and creativity, future use, and potential effectiveness of the system in improving their lives. As part of the interview, we asked participants if they had the system installed in their home and asked them to describe the chosen scenario. After they gave their description of their envisioned application, they were asked to program it without time limitations.

Measures
Below we present results related to perceived ease of use and engagement while using the system. Ease of use was measured using four items on the post-questionnaire. It asked for user responses on a 7-point Likert Scale (“Strongly disagree” to “Strongly agree”) to statements such as “I found the system unnecessarily complex.” Engagement was measured using three items on the post-questionnaire (e.g., “The experience was fun.”). Questions were adapted from the System Usability Scale (SUS) developed by John Brooke [4]. We supplement our discussion of ease of use and engagement with interview and observational data. We also discuss participants’ idea generation while using our system through observational data, interviews, and artifacts produced by participants while brainstorming. It should be noted that while we tracked time spent and errors made across tasks, we tested for and found no significant differences between conditions, and thus for space considerations we will not address those results further.

RESULTS
Ease of Use
We conducted a two-way ANOVA with ease of use as the dependent variable, and engineering background and condition as factors. While condition was not significantly related to ease of use ($F[2,30] = 1.36, p = 0.271$), engineering background was significantly predictive of ease of use ($F[1,30] = 5.41, p = 0.027$). Additionally, we found that engineering background interacted with condition to predict ease of use ($F[2,30] = 6.23, p = 0.005$; see Table 1 for means). Contrasts revealed that non-engineers found the MT condition to be significantly more difficult than the MM condition ($p = 0.006$) and the SM condition ($p = 0.003$). For the MT condition, non-engineers perceived the activity as significantly more difficult than the engineers ($p < 0.001$). To ensure that gender was not driving the effect, we then conducted a two-way ANOVA with ease of use as the dependent variable, and gender and condition as factors. Gender did not have a significant effect on ease of use ($F[1,30] = 2.63, p = 0.155$), and neither did the interaction between gender and condition ($F[2,30] = 1.27, p = 0.297$).

<table>
<thead>
<tr>
<th>Condition</th>
<th>Major (Number of participants)</th>
<th>Ease of Use</th>
<th>Engagement</th>
</tr>
</thead>
<tbody>
<tr>
<td>MT</td>
<td>Non-Engineering (5)</td>
<td>4.15</td>
<td>.74</td>
</tr>
<tr>
<td></td>
<td>Engineering (7)</td>
<td>6.12</td>
<td>.82</td>
</tr>
<tr>
<td>MM</td>
<td>Non-Engineering (6)</td>
<td>5.58</td>
<td>.63</td>
</tr>
<tr>
<td></td>
<td>Engineering (6)</td>
<td>5.39</td>
<td>.39</td>
</tr>
<tr>
<td>SM</td>
<td>Non-Engineering (8)</td>
<td>5.63</td>
<td>.88</td>
</tr>
<tr>
<td></td>
<td>Engineering (4)</td>
<td>4.75</td>
<td>1.27</td>
</tr>
</tbody>
</table>

Table 1. Descriptive statistics for ease of use and engagement.

Interviews supported the fact that some non-engineering MT participants found the activity “inconvenient” ($P33$) or “cumbersome” ($P35$). Part of the difficulty that participants had with the MT and MM conditions was the size of the mobile interface. P3 commented “I want to have a bigger size of screen,” while P17 commented “the current one [interface] is a little confusing for a small screen.” One participant with an engineering background ($P8$) better articulated some of the difficulties with the small screen:
“It was bothering to switch between the screen and the scene while programming. It might be cumbersome since people may need to modify -- change frequently.” Observation of participants confirmed that it was at times awkward to hold the mobile device while triggering the physical sensors, especially for the non-engineering participants. It is possible that engineers, more used to working with technology, had an easier time adapting to the difficulties that the tangible interface presented than non-engineers.

There is also evidence that the engineers appreciated the advantages of the MT condition in ways that the non-engineers did not. P1 commented on the benefits of the tangible environment for debugging: “I can test it, it's like a preliminary test to see whether sensors are working properly.” P26 described the advantages as follows: “I think physically making actions helped me remember -- follow a pattern I've created.”

It should be noted that using the menu-based tool also posed its own obstacles for some participants, especially with the need to use the correct sensor IDs and with participants not being sure of what sensor state to select. P36, in the MM condition commented: “…not easy to figure out what it is with the pictures in the list; I needed to find sensor ID, and it's a little inconvenient; especially because I was not sure if I should select 'open' or 'close' for a particular item.” P12 in the MM condition expressed concern about a potentially lengthy list of sensors when used in real situations including significantly more items, which may cause difficulty in finding intended ones.

**Engagement**

We then conducted a two-way ANOVA with engagement as the dependent variable, and engineering background and condition as factors. There were no significant differences between the effects of condition ($F[2,30] = 0.179, p = 0.837$) or engineering background ($F[1,30] = 1.023, p = 0.320$) on engagement, and no significant interaction effect ($F[2,30] = 2.018, p = 0.15$; see Table 1). We again looked at the effects of gender using a two-way ANOVA with engagement as the dependent variable, and gender and condition as factors. Gender did indeed have a significant effect on engagement ($F[1,30] = 5.16, p = 0.03$; Men: $M = 5.98, SD = 0.768$; Women: $M = 6.49, SD = 0.708$). The interaction between gender and condition was not significant ($F[2,30] = 0.319, p = 0.729$).

Interview data revealed the specific aspects of the activity that people enjoyed. People liked the immediate feedback they received after programming the application: “Almost instantly you could use an app. Really very fun!” (P17, MT condition); “I liked how quick it was to program, and how quick the responses were to the actions.” (P21; MM condition). Others liked particular features of the system: “It’s fun when music plays.” (P19; SM condition); “I liked the ability for the system to say something out loud” (P27; MM condition). The most engaging aspects of the system were not unique to a particular condition and engineering background.

**Ideation Experience**

**Effects of physical information**

**Gaining ideas.** Participants tended to look around the scene while developing their own scenario idea in the free-form task (59% of total participants). Even if they were in the SM condition, they turned their heads to face the scene. P13 from the SM condition said, “By looking at the room, I was able to get a picture of the whole area... I'm a visually oriented person so [I] like to see the whole picture...looking at the room, I thought of what is in my proximity, and whether it could be better done if I were sitting on the chair.” Participants also tended to look around the scene while brainstorming (64% of total participants; 92%, 50%, 50% of the MT, MM, SM participants respectively).

**Considering intangible aspects.** In addition, we found that the tangible interaction might have interfered with participant’s attention toward intangible elements. In the free-form task, only one person in the MT condition used time-date constraints, compared to five in the MM condition and nine in the SM condition.

**Intuitiveness**

**Ordering actions.** When participants in the MM and SM conditions had to mentally construct a series of actions, the resulting programs tended to be more unnatural. In adding actions for the phrase “leave the living room”, one participant (P13; SM condition) added first stepping off the mat and then stepping on, when the order should have been the opposite. Another participant (P16, SM condition) made her application send a SMS message when she closed the door at first. Then, she realized that it made more sense to receive a message before she shut the door, rather than after, as the message was a reminder that told her to turn off the air-conditioner before leaving for work. P19 (SM condition) remarked that the most difficult thing to her in doing the free-form task was to organize her actions (that is, add actions in the order that she actually does in her daily life). “It's hard to organize-- a couple of actions, picking up guitar, Turn on TV, Pick up dumbbells, but hard to order them...what do I do first?” It is likely that the MT condition made the organization of ideas more concrete.

**Finding circumstance of an action.** We further found that the tangible experience helped people to discover particular features of the system. P3 in the MT condition got puzzled since she did not have an idea for what sensor to use for the phrase, ‘leave the living room’. After wondering for a while, she decided to just walk to the entrance door hoping she might be able to discover a clue. She by chance noticed a sensor-augmented mat that was placed at the edge of the living room area, and realized that it was an appropriate item for the phrase.
DISCUSSION & CONCLUSIONS
In this paper, we presented GALLAG Strip, a tool designed to enable end-user programming of context-aware applications for behavior change, and results of a comparative experiment to explore the benefits of our tangible approach. We found that while our system was usable, people with a non-engineering background perceived it as less easy to use. In contrast, the participants who had some degree of programming skills considered it easier than the other conditions.

Richness brought by using real-world everyday objects can be considered as augmentations of ‘role expression’ and ‘hidden dependency’ dimensions among the Cognitive Dimensions suggested by Edge and Blackwell [9]. People are quite familiar with the uses of everyday objects. Their forms and operations naturally elicit people’s recognition and action. Furthermore, as objects are located in a living space, relationships between them build up. For example, the location of objects inside a container depends on where the container is placed. By sitting on a chair located at a particular spot, a user notices a picture frame in front of it. In our case, such richness of artifacts might have served as advantage for the participants with an engineering background as they are more able to handle both the visual interface on a mobile phone and rich information from an environment relatively easily.

Despite the difficulty the non-engineering participants had in using the system, we observed benefits of the tangible programming such as intuitive ordering of actions and diverse ideation with rich physical information. On the other hand, the tangible interface tended to distract the participants from the intangible elements of the system. These results imply that it may be useful to only encourage tangible programming interactions for tasks where the tangible medium is particularly beneficial. The tangible feature might best be used to support people’s creation of scenarios that are primarily related to object use and actions. In contrast, with a GUI, people might be better equipped to program applications with non-tangible or global states such as time or weather. Considering advantages of each method, we suggest that a mixed tangible and menu-based approach is appropriate to encompass user groups of different programming skills and use cases.

Our comparative experiment had some limitations. First, we conducted it in a lab setting. The participants may have showed more naturalistic responses in a place more familiar to them. Secondly, our sample size was too small to draw solid conclusions about the interaction between condition and engineering major. In addition, the numbers of participants of each educational background and gender were not evenly distributed over the conditions. An alternate explanation for the results could be that more women are non-engineers, and more men are engineers, so the results that we found might be driven by gender and not engineering background. However, the same results did not appear when we substituted gender for engineering background in our analyses, although we did find that women were more engaged than men overall. In spite of these limitations, we believe the combination of our qualitative and quantitative analyses provides interesting design insights that can inform future development of end-user tangible programming environments.

It was not always easy to determine whether problems we noticed were artifacts of our particular interface or tangible interfaces in general. Our results indicate that we should iteratively refine the interface to allow for even more naturalistic physical interactions. In our data, the small-sized screen and switching attention between the scene and the screen was identified as an issue in the tangible-mobile tool. We also found that the current interface design should be improved to minimize interruptions of natural user actions. Sound could be employed to supplement the visual interface to give information while the user is manipulating an object; for example, a sound could be played for a newly added action tile so that users do not have to look at the screen to check if it has been added successfully. It might be a better design to have the user record all actions first and then allow the user to insert responses between the recorded actions. The shortcomings of the present interface design may have influenced the results of our experiment, and further study is necessary with a tool improved to better support users’ natural performance.

Our work demonstrates benefits and disadvantages of a combined mobile and tangible programming tool for context-aware applications for end-users, with a particular focus on ease of use, engagement, and creativity. We also showed that our programming tool’s design has a low learning curve and that it is effective when compared to traditional ways of programming context-aware applications (i.e., coding). Our vision is to continue to develop GALLAG Strip as a means of facilitating programming of context-aware applications that are uniquely tailored to the needs of the end-user.

REFERENCES


