Multi-Touch Interactions for Customizable Message Conversations and Message Threading

Jared Branscum
Georgia Institute of Technology
Atlanta, GA
jbranscum3@gatech.edu

Sean Lu
Georgia Institute of Technology
Atlanta, GA
slu81@gatech.edu

ABSTRACT

With the rising presence of mobile devices, mobile messaging is now one of the most prevalent ways to communicate with other people. People communicate every day using traditional SMS texting or instant messaging applications. However, even given the technological advancements we have in mobile devices themselves, the user experience regarding messaging has stayed relatively stagnant. Messaging still follows the same basic flows and interactions. The goal of this work is to investigate alternative ways for users to interact with mobile messaging systems. We propose new interactions that will change the ways users chat, and also look to gain a deeper understanding of other related works regarding user interactions in the mobile messaging space.

Keywords: Messaging, mobile, interface, user interaction, multitouch, messaging threading, customization, personalization.

INTRODUCTION

Mobile messaging attempts to take the social aspect of humans through conversations, and put it into a more convenient and accessible medium. However, putting conversations into a mobile and textual format comes with some compromises as well. Face-to-face conversations often involve underlying factors such as gestures, expressions, and emotions. In a text-based environment, a lot of this is lost, resulting in confusion and degradation in quality of conversation.

Current mobile messaging systems all follow a similar unanimously agreed-on format which stem from their predecessors such as e-mail or SMS. In the related works section, we explore the affordances of mobile devices which have lead to the design of current mobile messaging. We consider the limitations of mobile devices and how that impacts the user experience and how users are able to interact with mobile devices. We also investigate existing systems which attempt to augment mobile messaging to be more similar to face-to-face conversations. Finally, we search for important factors in mobile messaging which affect the user experience.

In the proposal sections, we present new interactions for mobile messaging which take advantage of a threaded messaging format in order to most similarly mirror face-to-face conversations. By analyzing factors which influence messaging efficiency, we can create interactions which aim to solve these common pain-points in messaging.

MOTIVATION

There are various communication platforms, such as GroupMe, Discord, WhatsApp, iMessage, etc. Their core purpose of providing an environment for users to interact with one another through their digital device remains roughly the same across these platforms and their interfaces tend to mimic the SMS interface. These communication platforms still use the same basic interface components of buttons, scrollbars, pinch zooming, text fields, etc. We have grown accustomed to the format of SMS without exploring alternative user interactions to make communication easier, more intuitive, and user-centered.

However, the intended use of SMS texting when it first became popular is much different from the way it is used today. In the present day, messaging and texting are ubiquitous. It is even the first thing some people do when they wake up in the morning. Even though messaging has evolved and grown far from its original use, the designs and interactions have not evolved with it. This inspired us to further investigate current research on messaging user interactions. We expect to develop a prototype application that will introduce new ways to interact with a standard messaging interface in order to enhance the conversational experience.

RELATED WORKS

Messaging Background

Messaging on mobile devices has extended much further than traditional SMS. These mobile apps allow users to send messaging instantaneously over the internet without needing to go through the downsides of SMS, such as cost. More recently, there has been a shift in communication from face-to-face interactions to instant messaging applications [11]. Coordination between groups used to assume interactions through SMS or mobile calls, but have increasingly turned to mobile messaging apps [8]. Church and Oliveira conducted two studies through interviews and a large-scale survey regarding these mobile messaging apps. They found that mobile messaging apps such as WhatsApp offer advantages such as a more social environment [2]. The way that messaging apps are used tend to be more conversational, which makes perfect sense considering messaging is meant to emulate conversations without being in person. The main takeaway is that a successful and immersive messaging app captures as many elements of an in-person conversation as possible.

Mobile User Interactions Background

Designing interfaces and interactions under a mobile environment requires a significant shift from what is normally seen in desktops. Most importantly compared to desktop, mobile devices have much less screen real estate. Traditional input devices are also limited, resorting mostly to touch interfaces [9]. The lack of screen real estate also puts limitations on the range of touch interactions that can be possible. Caro-Alvaro et al. [1] performed systemic evaluations using mobile devices on mobile instant messaging apps. The evaluations were performed using Keystroke-Level Modeling (KLM) to measure time to complete tasks and Mobile Usability Heuristics (MUH) to detect usability problems. The results of the evaluation showed that the most important recommendation for mobile messaging is to avoid deep navigation. That is, messaging should require few interactions in order to complete a task and should not exceed more than 5 or 6 interactions.

In a study by Park and Han, they found that touch key size and location were important to users being able to complete tasks. The results showed that a touch key size (a square shape where users touch) of 7mm and 10mm were more optimal than 4mm for user performance [10]. Although screen real estate on mobile devices is limited, it is difficult for users to be extremely precise when using fingers to touch the screen. Colley and Hakkila explored using finger-specific touch interactions on mobile phone interfaces. In this scenario, specific fingers were able to trigger specific actions, providing fast shortcuts to various functions [3].

Systems Enhancing Messaging

Many alternative interactions for messaging have been explored. In most of these cases, the new interactions sought to make messaging more ”real”. One of the main limitations of messaging on mobile devices is that it is difficult to fully reflect a real conversation. Using only text, aspects of conversation such as emotion or intent can be difficult to capture.

For example, HeartChat integrated heart rate into messaging to show the people who you are talking to what your heart rate is when sending the message. Study results showed that providing heart rate info helped understand conversational context and emotion better [6]. Similarly, ConductiveChat incorporated users’ skin conductivity levels in the messages being sent. When a user is aroused based on skin conductivity, the color and size of the messages being sent fluctuates [5]. Wang, Prendinger, and Igarashi built a chat system using physiological sensors attached to a user’s body to show affective state of the user [14]. Another messaging system used a brain-computer interface and gestures in order to capture emotions and facial gestures [7]. Each of these different messaging interactions attempted to make messaging more conversational and life-like.

Messaging Effectiveness

We have established that messaging lacks what is present in real-world face-to-face conversations. We have also explored some systems which use biological factors to help enhance chat messaging. However, messaging in the present day using mobile devices is fast and efficient. Requiring sensors or heart measures in order to obtain metrics to help influence messages is clunky and infeasible. However, we can look at factors within the structure of standard chat messaging which can help us achieve conversational effectiveness.

Vronay, Smith, and Drucker studied what factors may influence the efficiency of a chat conversation. Some highlights from these factors include lack of intention indicators, lack of context, lack of recognition, and high signal-to-noise ratio [13]. Lack of intention indicators refers to the fact that it is difficult to tell if certain messages are directed to one user or another. Lack of context means that it is difficult for a user to understand the conversation from before he or she has ”arrived”. Lack of recognition refers to it being difficult to recognize who is sending what message. High signal-to-noise ratio states that there is a lot of messages from users which do not greatly contribute to the main conversation. In relation to context and noise, some messages can be classified as announcements or spam, which may not generate a message response from other users in the conversation [4].

One solution which helps to avoid noise and provide meaningful context is threaded messaging. Threaded messaging addresses the previously mentioned problems regarding confusing message history and out-of-turn responses [12]. In a threaded structure, messages are more coherent as they are in shown to be in response to a specific message. A study conducted by Wold had participants interact in conversations using two similar messaging applications, one standard application with no extraneous features, and one application with message threading capability. The results showed that study participants had more coherent conversations with less interruptions or noise using threaded messaging [15].

Recognition can be achieved by making a user’s contacts and those contacts’ respective messages more customizable. Adding unique features to contacts and their respective messages can make it easier to differentiate between users in a conversation. We expand on this subject further in our proposal.

SYSTEM DESIGN

Our project proposes to implement message threading as a core feature in messaging applications, but we also propose designing a sandbox platform for users to manipulate their message and conversation backgrounds to add more flavor and emotion to the conversational experience. We believe that leveraging multi-touch functionality will make messaging more intuitive and more interactive for users, compared to existing single-touch systems.

Design Specifications

- The end system is built as a mobile web application in HTML, CSS, and JavaScript.

- Message Threading to compress messages into threads, so chat history is less cluttered with irrelevant messages.
- Messages and conversation backgrounds can have dynamically painted backgrounds that utilizes multi-touch to manipulate color patterns used in the background.
- These user-designed backgrounds are created in a sandbox-like interface, and are displayed locally per user. Other users within the same message chat will be able to customize message backgrounds their own way.
- Users can select only one message background for each contact, so messages from that contact will only have the selected message background. The default for contacts is having no background, using a default solid color.

Message Threading

We define message threads as a collection of messages that are organized into an object within a messaging conversation. Message threading is generally implemented by creating a global thread that's shared between the users in a conversation across separate devices and the users have to intentionally send messages in the thread. Slack is an example application that uses this kind of global thread implementation. We propose for users to create their own local customized threading, so they can personalize how their messages are compressed.

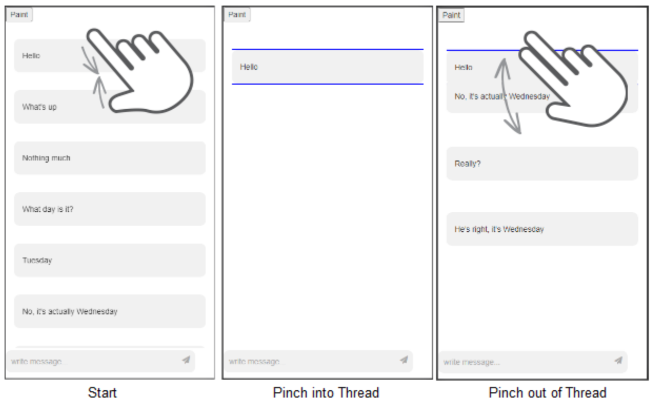

Hello
What's up
Nothing much
What day is it?
Tuesday
No, it's actually Wednesday
Really?
He's right, it's Wednesday

Figure 1: Message Threading Interactions
User pinches in messages to create a thread, then pinches outwards to expand messages out of the thread.

The threading interaction begins by the user pinching in our pinching out messages in the chat display. Pinching in causes the messages between the fingers to compress into a message thread. While the pinch interaction is occurring, the messages affected messages move with the touch interaction to provide active feedback to the user. The pinched messages will center around one main anchor message which will be the one message of the thread that is displayed. The anchor message is created during the pinch in interaction, and is designated by the first message that collides with another message as a result of the pinch. Whenever the user pinches out around the anchor message, messages that were in the thread are removed from the thread and placed back into the normal conversation. After a message thread is created, the user is free to pinch in additional messages into the thread, or remove messages using the pinch out interaction. The user can freely manage how many messages above or below the thread can be compressed into/out of the thread depending on how the fingers for the pinch interaction are positioned. The user is able to interact with messages above and below the threaded anchor message at the same time.

Paintable Message Backgrounds

The Paint button at the top left of the screen begins the process to customize backgrounds for a contact's messages. Pressing the paint button allows the messages within the chat to be clicked. This affordance is given by the messages highlighting in a blue aura, similar to that of a hyperlink. Clicking a highlighted message will bring up the paint canvas screen. Users access a color palette picker where the user can select a primary and secondary color to draw a background with. After the user selects their choice of colors, the user can paint the background of their message using multiple colors in a preset blend pattern. The way the colors are blended together will be described in the subsection **Color Blend**. The blend patterns are selected from the Color Blend Pattern Wheel, which the user can interact with using a two-finger rotate interaction. The user will also have a background color option that fills the entire background canvas the selected color. When the the user is finished customizing the painted background, the Apply button brings the user back to the main chat messaging interface, with the new customized background displayed for the appropriate messages.

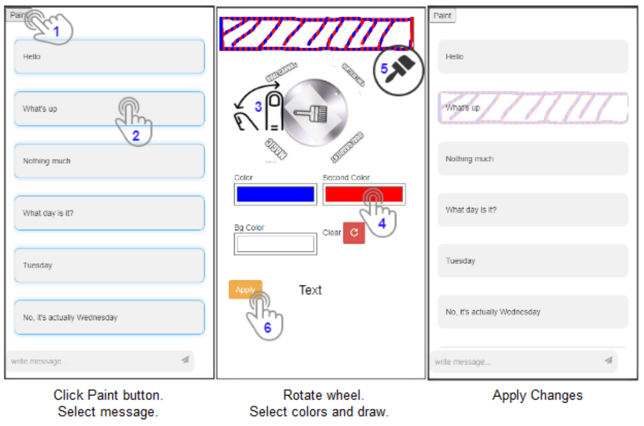

Hello
What's up
Nothing much
What day is it?
Tuesday
No, it's actually Wednesday

Figure 2: Paintable Message Background Interactions

Color Blend

Color blending is a challenging interaction for users to customize, so we decided that the interaction would be easier for users and be more intuitive if the application comes with pre-built color blending patterns. The user can use the two-finger rotate interaction to select which color blending pattern they like. There are four selections: Vertical Lines, Dual Gradient, Magic, and Dual Canvas. The vertical lines pattern alternates the two colors in the form of vertical lines through the draw stroke. The dual gradient pattern draws strokes with a mixed color using both gradients. The magic pattern draws an evolving stroke that changes color as the line is being drawn. The dual canvas pattern splits the message canvas in half down the middle, designating each half for one color. The desired pattern is selected based on which quadrant the wheel is pointing towards. The wheel direction is based on the direction the paintbrush is pointing. There is also a circular hole in the wheel as an additional marker of where the wheel is pointing.

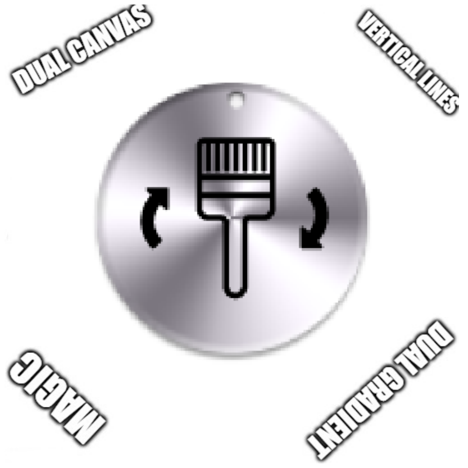


Figure 3: Color Wheel Pattern Tool

Implementation

The system is a web application using HTML, JavaScript, and CSS. The main design and layout of the application involved HTML and CSS to build a system resembling typical messaging environments. The multi-touch interactions involving message threading and customized paint backgrounds were implemented using JavaScript and the interact.js framework. Additional third-party libraries used include jQuery to perform robust JavaScript actions, and Font-Awesome and Bootstrap for button icons.

The interact.js framework allows for easy movement and manipulation of HTML elements based on touch interactions. Interact.js allows for creation of functions to have listeners on touch interactions for a designated HTML class. When the listener is activated, the function goes through a start, move, and end event to handle the the interaction.

For the pinch-in threading functionality to create a message thread, there is an interact.js listener for when the message class is dragged. Based on the touch points of the pinch interaction, the messages in between the touch points combine into a thread. The appropriate messages all move with the pinch interaction for immediate feedback. The distance that the messages move is based on the vertical distance that the fingers travel. When messages collide with the main anchor message of the thread, the message is hidden using CSS.

For the pinch-out threading functionality to pull messages out of an existing thread, there is an interact.js listener for when a pinch-out interaction is performed on a thread of messages. Similar to the pinch-in interaction, the messages within the thread expand outwards depending on the vertical distance of the pinch-out interaction. The hidden messages within the thread will appear out of the thread if the distance of the pinch-out interaction is large enough to fit the reappearing messages on the screen. The amount of messages pulled out of the thread depends on the length of the pinch-out interaction, so not all messages will be expanded out of the thread at all times.

The painting and drawing interaction follows a standard implementation. As a user draws a stroke on the paint canvas, the canvas fills in a point of the designated color using the x and y coordinate of the touch stroke. The finished canvas is then saved as an image. The image is applied to the background of the selected messages at a low opacity to allow for visibility of the message text.

The color blend pattern wheel has an interact.js listener that detects a rotate gesture on the wheel element. Both a multi-touch rotate and a single-touch rotate can perform this action. When a gesture is detected, an angle of the gesture is calculated using the tangent of the starting and ending (x, y) coordinates of the touch points. The wheel then rotates by the calculated angle through a CSS transform. The selected color pattern is based on the current angle that the wheel is pointed at. The top right quadrant (1-90 degrees) is the vertical lines pattern, the bottom right quadrant (91-180 degrees) is the dual gradient pattern, the bottom left quadrant (181-270 degrees) is the magic pattern, and the top left quadrant (271-360 degrees) is the dual canvas pattern.

Weaknesses

The web application is built mainly to introduce the new multi-touch interactions that we have described previously. In its current state, it does not actually allow for any messaging functionality such as sending/receiving messages or adding new contacts. As a result, the paint message background feature can be a little ambiguous. It is meant to apply the background to all messages of a specific contact. However, without contacts functionality, it simply applies only to the selected message.

Currently when doing a pinch-in interaction to create a thread, the anchor message of the thread is designated based on which message is the first to collide with another message. This leads to inconsistency, because the user is not able to directly control which messages becomes the anchor message of a thread. In our initial design, we required a swipe gesture before the pinch interaction, where the swipe was used to the mark the swiped message as the anchor. However, we removed the swipe interaction in order to make the threading functionality more simplistic and less complex. A potential solution could be to allow the user to manually select which message in the thread should be marked as the anchor after the thread is created.

Messages are added into a thread based on which messages are located within the two fingers of a user pinch interaction. However, this means that an unwanted message in between the two fingers will still be added to the message thread. This scenario is an edge case which may not happen all the time, and should not affect the foundation of the pinching interaction. A proper solution to this edge case would be to allow users to manually inspect and remove specific messages out of a thread after it is created.

The color pattern wheel rotates and selects a pattern based on the quadrant that the wheel is pointing at. Although the user can tell which way the wheel is pointing based on the direction of the brush and hole in the wheel, there is no feedback for when a pattern is selected. The wheel simply spins, and the user has to trust that the correct selection has been made. A solution to this would be to add instantaneous feedback by highlighting the text of the pattern when it is selected.

Finally, a new user may not completely understand what each of the color blend patterns actually are. With no visual examples or indication of the patterns, the user will now know what the patterns do until attempting to draw on the canvas. This will lead to the user constantly clearing the canvas if an undesired pattern selection is made. A solution to this could be to have the text of the color pattern be drawn using the pattern itself, so the user has a preview of what it would look like.

EVALUATION

While no formal evaluation was conducted, some feedback was received through the virtual poster session based on a poster representing a simplified storyboard. The poster was minimal, and did not display all the steps and interactions possible. The feedback gathered was valuable, but not as valuable compared to other methods such as interviews or cognitive walkthroughs.

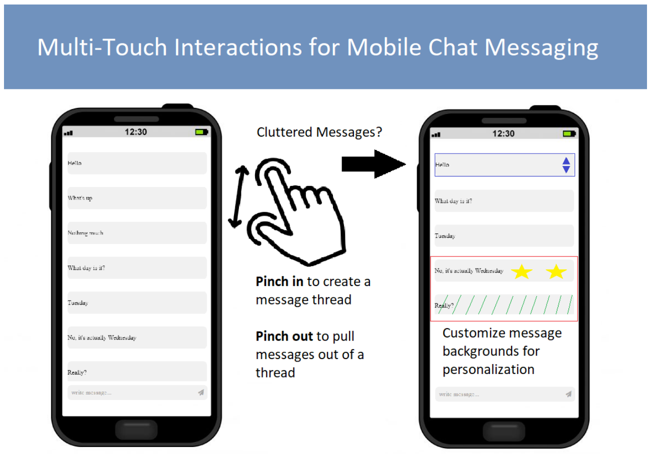

Multi-Touch Interactions for Mobile Chat Messaging
Cluttered Messages?
Pinch in to create a message thread
Pinch out to pull messages out of a thread
Customize message backgrounds for personalization

Figure 4: Multi-touch Messaging Poster

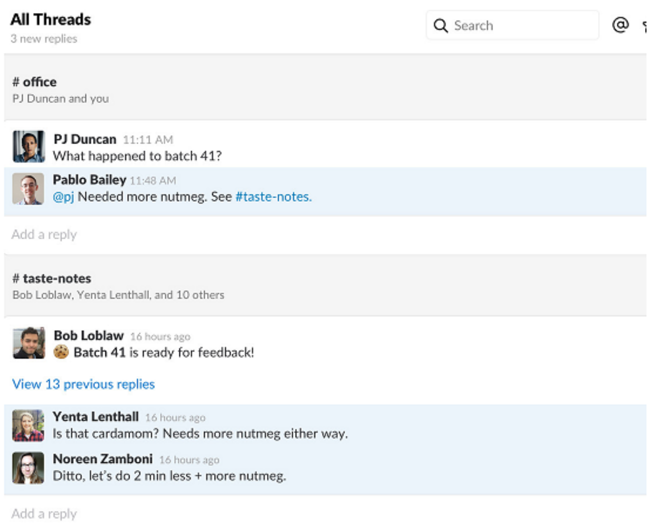
The main concern and question that occurred frequently was the ambiguity of which message is the anchor of a newly created thread. This is also stated as a weakness in the **Implementation** section of the report. There were concerns of how the pinch-in interaction would be able to tell which message becomes the anchor of the thread. There were also questions of whether or not the anchor could be changed after the thread is created, in case the user wanted to change it at a later time.

There was positive feedback for the customizable message backgrounds feature. As a novel feature, people were not sure exactly what to expect, but were excited to see how it could positively impact their messaging experience. The fact that message backgrounds and threaded messages were local to a user was also a popular detail, due to the fact that individual users may want to organize things their own way, or have certain personal preferences.

DISCUSSION

Some main problems with current chat messaging interfaces include lack of context, lack of recognition, and presence of noise as stated before in the **Related Works** section. Lack of context refers to the difficulty of understanding the intention and background behind certain messages without being fully immersed in the conversation. Lack of recognition means that it is difficult to discern who sends what message. Present-day systems have alleviated this problem by displaying names and profile pictures in order to hopefully give a better identity to who is behind the message. The presence of noise can be a big problem in messaging, and it is a problem that even builds off of the lack of context issue. The presented system aims to help solve these problems with novel interactions and features which help give the user a more personalized messaging experience.

Messaging applications often focus on the experience of sending messages, but forget about the other half of the transaction: reading the messages. A poor user experience for reading messages will lead to growing issues with context and noise. Existing applications such as Slack or Microsoft Teams implement message threading, which help give more context to messages. However, this is under the assumption that the threading functionality is used correctly. There can still be out of context messages or noise inside of a message thread. There can also be messages that were meant to be within a thread, but were sent as a regular chat message outside of the thread. These are just two examples of user error that can lead to ineffective message threading in current implementations. Applications with message threading can help reduce potential context and noise problems, but are not completely immune to them.


All Threads
3 new replies
Search

office
PJ Duncan and you

PJ Duncan 11:11 AM
What happened to batch 41?
Pablo Bailey 11:48 AM
@pj Needed more nutmeg. See #taste-notes.
Add a reply

taste-notes
Bob Loblaw, Yenta Lenthall, and 10 others

Bob Loblaw 16 hours ago
🍪 Batch 41 is ready for feedback!
View 13 previous replies
Yenta Lenthall 16 hours ago
Is that cardamom? Needs more nutmeg either way.
Noreen Zamboni 16 hours ago
Ditto, let's do 2 min less + more nutmeg.
Add a reply

Figure 5: Slack Message Threading

Our presented implementation solves these issues with message threading functionality that is local to users. Since noise, context, and recognition can be different depending on individual users, the best way to rid of these problems is to let the user solve them based on personal preferences. With local client-sided threading, one user can create a thread to his/her liking, whereas another user may prefer to create a thread with different messages included. Giving the power to each individual user to organize messages their own way ensures that the messages are easily readable to everyone regardless of different preferences.

The pinching interactions for message threading are a simple and intuitive way to support threading. Similar to the standard pinch-to-zoom interaction that everyone is accustomed to, pinching in makes things bigger while pinching out makes things smaller. This relationship is easily transferred to message threading. When pinching in on multiple messages, the amount of messages become ”smaller” because the pinched messages compress into a single message thread. When pinching out on a message thread, the amount of messages becomes ”bigger” because the hidden messages within the thread are expanded and pulled outwards. When messages are manipulated using the pinching interactions, the appropriate messages provide responsive real-time feedback and move across the screen along with the multi-touch interaction. The user can feel as if he/she is dragging the messages themselves into or out of the thread.

Allowing users to customize their own message backgrounds per contact is a completely novel interaction that no other application currently supports. The main goal of customized message backgrounds is to help alleviate the problem of lack of recognition, where it is difficult to tell who the message sender is. The current solution to lack of recognition is to display a name and picture next to messages. However, these can only be controlled by that person. If someone in a message conversation frequently changes their profile picture, it can be confusing to other people who are reading the messages and trying to figure out who it is sent by. Our proposed solution is to allow a user to customize the message background for everyone. These customizations are local to the user, meaning other people in a message will not see the same backgrounds. Being able to customize other users' message backgrounds makes the messaging experience more personal. As a user, I will be able to more easily recognize a message background that I personally draw and create myself as opposed to something that the message sender sets up.

In addition, message backgrounds that display behind the text within a message bubble are an area of the messaging experience that is overlooked. WhatsApp allows users to set the wallpaper of a conversation to appear behind all of the messages, but no application supports backgrounds for each individual message. Currently, messaging applications simply leaves it a default white background, likely so that the text is readable. Our implementation of customized message backgrounds apply the backgrounds at a low opacity, and the text is equally readable compared to a plain white background. Our novel feature shows that there is room and opportunity to use this blank white space for new and interesting features.

FUTURE WORK

In addition to the improvements detailed in **Weaknesses**, we have also identified more areas of our project that can be further explored. Our current implementation requires users to add a message background to each individual message they wish to customize, so we would design a fully functional contact system. Users would be able to choose to customize all the message backgrounds for individual contacts, where the background would be applied to all of the contact's messages. Furthermore, we would conduct a formal user evaluation to test the effectiveness of our novel user interactions. We would test the effectiveness of the application's affordances, feedback, mapping, and signifiers. We would also leverage more multi-touch functionality to edit existing message threads so users can add/remove specific messages for message threads and set a new anchor message for the thread. There are also opportunities to explore different multi-touch interactions for the paintable message backgrounds and evaluate which features are important to users.

CONCLUSION

Literature helped guide us in exploring different approaches for users to interact with messaging applications. We learned about various key factors that are vital to effective messaging: lack of intention indicators, lack of context, lack of recognition, and high signal-to-noise ratio. We designed new user interactions that alleviate these problems by leveraging multi-touch technology. We also incorporated user interface design principles such as affordances and real-time feedback, but found room for improvement. The interactions we implemented are novel to the messaging application space and introduce interesting opportunities for applications like WhatsApp, SnapChat, and Instagram. It can be concluded that despite messaging applications having similar interfaces, there are still opportunities for UI/UX researchers to investigate the effectiveness of chatting applications.

REFERENCES

1. Caro-Alvaro, S., Garcia, E., Garcia-Cabot, A., de-Marcos, L., Gutierrez-Martinez, J.M. A systematic evaluation of mobile applications for instant messaging on iOS devices. *Mobile Information Systems*. 2017.

2. Church, K., Oliveira, R. What's up with whatsapp? comparing mobile instant messaging behaviors with traditional SMS. *In Proceedings of MobileHCI '13*. Association for Computing Machinery, New York, NY, 2013.

3. Colley, A., Hakkila, J. Exploring finger specific touch screen interaction for mobile phone user interfaces. *In Proceedings of OzCHI '14*. Asoociation for Computing Machinery, NY, USA, 2014, pp. 539–548.

4. Cui, Y. Messaging design and beyond: Learning from a user study on holiday greeting messages. In *In Proceedings of Mobility '07*, Association for Computing Machinery, New York, NY, USA, 2007, pp. 159–166.

5. DiMicco, J. Conductive chat: instant messaging with a skin conductivity channel. MIT Media Lab, 2002.

6. Hassib, M., Buschek, D., Wozniak, P., Alt, F. HeartChat: heart rage augmented mobile chat to support empathy

and awareness. *n CHI '17*. Association for Computing Machinery, NY, USA, 2017, pp. 2239–2251.

7. Kuber, R., Wright, F.P. Augmenting the instant messaging experience through the use of brain-computer-interface and gestural technologies. *International Journal of Human-Computer Interaction 29(3)*, 2013, pp. 178–191.

8. Ling, R., Lai, C.H. Microcoordination 2.0: social coordination in the age of smartphones and messaging apps, Journal of Communication, 66(5), 2016, pp. 834–856.

9. Lumsden, J., Brewster, S. A paradigm shift: alternative interaction techniques for use with mobile and wearable devices. *In Proceedings of CASCON '03.* IBM Press, 2003, pp. 197–210.

10. Park, Y.S., Han, S.H. Touch key design for one-handed thumb interaction with a mobile phone: effects of touch key size and touch key location. *International Journal of Industrial Ergonomics, 40(1)*, 2010, pp. 68–76.

11. Schwarz, O. Who moved my conversation? instant messaging, intertextuality, and new regimes of intimacy and truth. Media, Culture, and Society 33(1), 2011, pp. 71–87.

12. Smith, M., Cadiz, J., Burkhalter, B. Conversation trees and threaded chats. eq In Proceedings of the ACM Conference on Computer Supported Cooperative Work. 2000, pp. 97–105.

13. Vronay, D., Smith, M., and Drucker, S. Alternative interfaces for chat. In *Proceedings of the 12th annual ACM symposium on User interface software and technology (UIST '99).* New York, NY, USA, 1999, pp. 231–240.

14. Wang, H., Prendinger, H., and Igarashi, T. Communicating emotions in online chat using physiological sensors and animated text. *In CHI EA '04*. Association for Computing Machinery, NY, USA, 2004, pp. 1171–1174.

15. Wold, A. Multi-threaded conversations in instant messaging clients. University of Oslo. 2007.