Boston University
Electrical & Computer Engineering
EC464 Capstone Senior Design Project

User's Manual

VOCL

Submitted to

Andrey Vyshedskiy
vysha@bu.edu

by

Team #12
Vocl

Team Members

Mia Hernandez erojash@bu.edu
Charles Mo chmo@bu.edu
William Nilsen williamn@bu.edu
Krishan Eskew keskew@bu.edu

Submitted: April 17th, 2023

Vocl

Table of Contents

Executive Summary

Currently, there is a lack of educational and recreational resources for children with nonverbal autism that are engaging and informative. VOCL seeks to bridge that gap by developing virtual assets and sound processing algorithms in an app integrating educational software to provide a fulfilling experience for children that cannot receive it through traditional means. We will develop and modify Unity assets to provide an engaging experience for children in our product, as existing applications lack entertaining interactive character models specifically designed for children. The application will contain nonverbal communication processing algorithms, like clapping or moving the device, allowing children with any level of communication to have a fulfilling and interactive experience.

1 Introduction

Autism is a disability characterized by "persistent deficits in social communication and social interaction across multiple contexts" and "restricted, repetitive patterns of behavior, interests, or activities"[1]. In the United States today, about 1 in 44 children has been diagnosed with Autism according to the Centers for Disease Control and Prevention[2]. Moreover, approximately 63% of children who have been diagnosed with Autism also experience language impairment[3]. With a growing population being affected by the disability, the need for early intervention for educational and social purposes also grows.

Autism's impact on a child's ability to develop social and language skills interrupts their development in many areas, including relationships, education, career, etc. This delay becomes more pronounced if the child continues not to interact with their surroundings and peers. By creating an educational, engaging, and fun mobile application that targets nonverbal Autistic children, we lessen the gap between neurotypical and neurodivergent learning and social abilities.

The application needs to be appealing enough for the child to consistently use the app and continue developing their language and communication skills. Such an application is not readily available in the market today. Thus, our design will deliver a game that runs on the user's device that will focus on bridging the gap between education and entertainment via nonverbal cues accompanied by engaging games with 3d assets. On the suggestion of our client, our game is built in the Unity engine and designed for iOS. We have developed audio and touch algorithms that can detect the user's responses to various non-verbal cues. These responses are key to improving the comprehension and communication skills of nonverbal autistic children, and, accompanied by the 3d visuals and variety of game activities, keeps the child engaged.

The following user's manual gives an in-depth description of our project in its current version. First, there is a discussion of the system we have created, giving a brief overview. Following that is a discussion on the operation of our product, proceeding

---

[1] "Diagnostic criteria," *Centers for Disease Control and Prevention*, 06-Apr-2022. [Online]. Available: https://www.cdc.gov/ncbddd/autism/hcp-dsm.html. [Accessed: 12-Oct-2022].

[2] "Data & statistics on autism spectrum disorder," *Centers for Disease Control and Prevention*, 02-Mar-2022. [Online]. Available: https://www.cdc.gov/ncbddd/autism/data.html. [Accessed: 12-Oct-2022].

[3] N. Georgiou and G. Spanoudis, "Developmental language disorder and autism: Commonalities and differences on language," *Brain sciences*, 30-Apr-2021. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8147217/#:~:text=Although%20structural%20language%20impairment%20as,have%20language%20impairment%20%5B10%5D. [Accessed: 12-Oct-2022].

which is the requisite technical background. Finally, there is a brief discussion of both engineering standards and the cost breakdown of our project.

2 System Overview and Installation

Figure 2.1 shows the block diagram of the system, which contains the moving parts that have been implemented so far. The application exists on an iOS capable device, and interacts with the input sources and other hardware on the device along with the OS itself. Audio and touch inputs are sent from the input hardware to the OS, which handles them and sends the data to Vocl. The Unity game framework implemented on iOS is rather self-contained, and the application does not interface with other apps on the target device.

2.1 Overview block diagram.

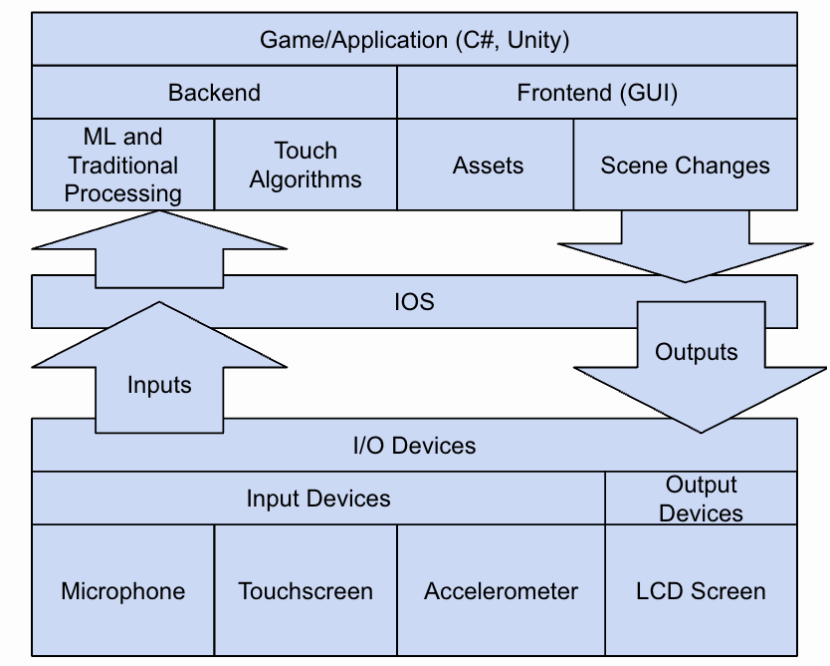


Figure 2.2 shows the user interface setup; since it is an iOS application, the only setup needed is opening the app upon downloading.

2.2 User interface.

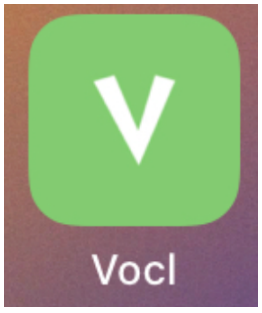


Figure 2.3 shows a physical description of the system; simply put, it requires the target device to be an Apple iPhone or iPad running iOS 11.0 or higher.

2.3 Physical description.

2.4 Installation, setup, and support

Vocl will be available for download on the Apple App Store. Prospective users will need to search for “Vocl” in the App Store on their iOS device and download from there, provided there is enough space on their iPhone or iPad. Refer to figure 2.2 to ensure that the correct application is installed, as it will match that spelling and application icon.

3 Operation of the Project

3.1 Operating Mode 1: Normal Operation

Under normal operation, the user will be greeted by the main loading screen for 10 seconds. The main loading screen consists of two animations: our rabbit asset blowing a bubble and 2 gears rotating.

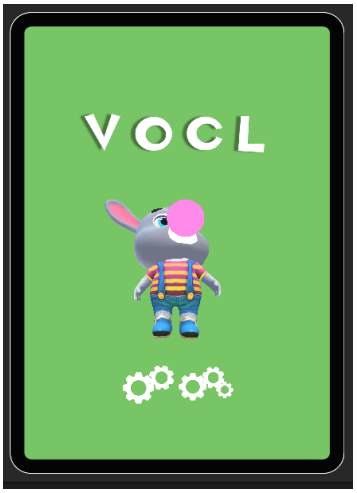


Figure 1.1 Main Loading Screen

The user will then be presented with a Menu screen where the title is ‘Choose a Game’ and two game options are displayed. Pressing each game button will launch our ‘Loading Game’ screen.

Figure 1.2 Main Menu Screen

Once the game is done loading, the scene will be changed to either Game 1 or Game 2 depending on the user’s button input.

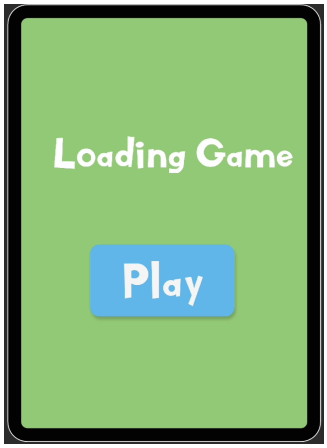


Figure 1.3 Loading Game Screen

3.12 Game 1

The first task involves the user tapping on the correct object. In this case, the correct object is the largest apple and there are two incorrect objects including a small apple and a banana.

Figure 1.4 First Game Task

1. If the user taps on either the small apple or banana, they are prompted to try again.

Figure 1.5 User prompted to try again

2. If the user taps on the largest apple, they are congratulated, and the rabbit runs to the next destination.

Figure 1.6 User congratulated and rabbit continues in the game

The second task asks the user to order the strawberries from smallest to largest.

Figure 1.7 Second Game Task

1. If the user clicks on the submit button and the strawberries are not in the correct order, they are prompted to try again.

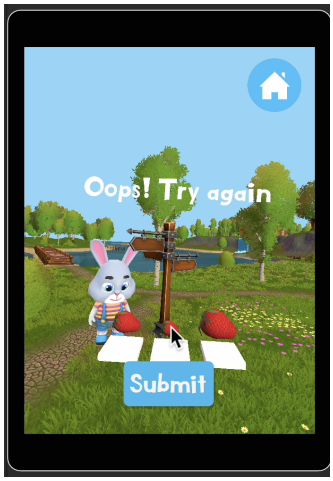


Figure 1.8 User prompted to try again

2. If the user clicks on the submit button and the strawberries are in the correct order, they are congratulated, and the rabbit runs to the next destination.

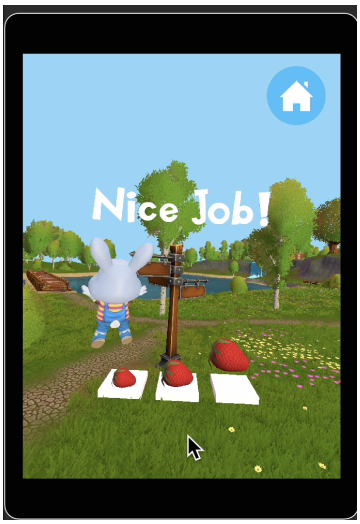


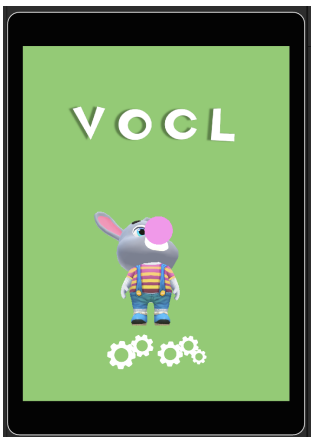
Figure 1.9 User congratulated and rabbit continues in the game

The third task prompts the user to clap three times.

Figure 1.10 Third Game Task

1. Once the audio algorithm detects the user has clapped three times, the rabbit congratulates the user. The user is not prompted to try again if they clap an incorrect amount.

If at any time the user wishes to quit the game and go back to the Main Menu, they can tap on the Home button which will launch the Main loading screen that then leads to the Main Menu screen.



Figures 1.11-1.13 Home Button Execution

Finally, the user is shown an end-of-game menu where they can either play again or go back to the Main Menu.

Figures 1.14 End of Game Screen

3.13 Game 2

Game two follows a similar structure as Game 1, however with an added goal of collecting all the candy. The first task prompts the user to stomp two times.

Figures 1.15 - 1.16 Game 2 Goal and First Task

1. Once the audio algorithm detects the user has stomped two times, the rabbit congratulates the user. The user is not prompted to try again if they stomp an incorrect amount. The lollipop is also animated to seem as if collected in the basket.

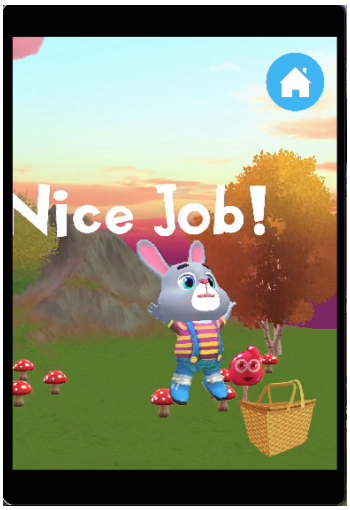


Figure 1.17 User congratulated and rabbit continues in the game

2. The second task prompts the user to tap the rabbit's ear three times. Once the user completes the task, the rabbit runs to the next destination.

Figure 1.18 Second Game Task

3. The third task will prompt the user to clap three times. Same as Game 1's last task. (See Figure 1.10)
4. The same end of game screen from Game 1 will be displayed once the user completes all three tasks. (See Figure 1.14)

3.2 Operating Mode 2: Abnormal Operations

A possible abnormal state for our project would include the application unexpectedly crashing and closing. To account for this, we want to implement crash reporting as well as keeping these records so that we may have an idea of what may cause this to happen. Typically, this has occurred when the game uses more memory than allocated for it, so that is the number one metric we will keep track of. The only user intervention needed would be the user re-launching the application on their device. Since our target user is a child with nonverbal Autism, we hope that a parent can access this manual or do some basic troubleshooting if an issue were to arise. Troubleshooting would include closing and reopening the app, checking that they are connected to wifi, and restarting the device.

3.3 Safety Issues

There are currently no safety concerns regarding our project. As a software-based product, our application presents no physical threats to either the user or bystanders. With regard to data privacy, the application asks for permission to access the device's microphone. The audio is then sent to our server hosting the machine learning model but is not saved. The user's audio data is therefore private and not stored or distributed in any way. Professor Vyshedskiy and his company - Imagiration - reserve the right to change this policy.

4 Technical Background

4.1 Unity Game

Vocl is built on Unity's cross-platform game engine to allow easy portability to iOS. The Unity API provides abstraction over many platform's firmware, allowing us to use C#/.NET and the Unity API library rather than having to write iOS-specific firmware. Vocl employs Unity's graphical interface to deliver game assets. The graphical interface defines the scenes and positions of game objects in each scene. After the definition of each scene and object, the Unity API interfaces with device drivers to perform the game logic defined in C#/.NET which these assets.

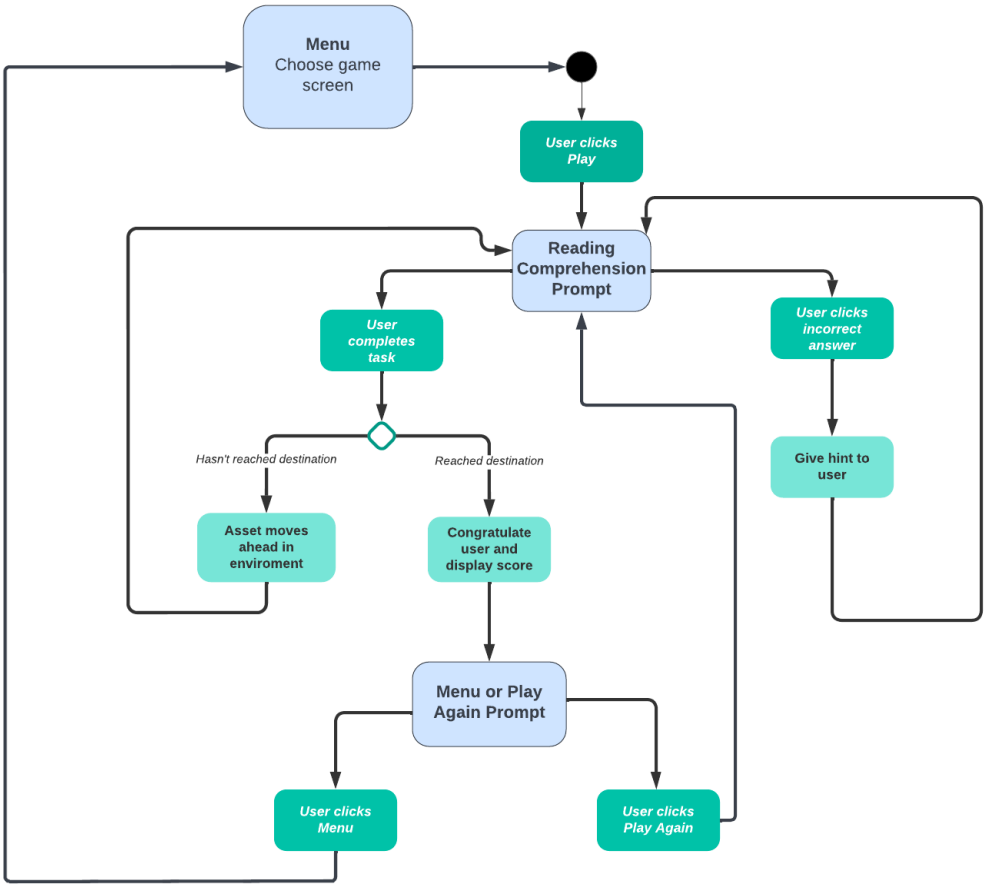


Figure 1.18 Game Logic

4.2 Audio Processing

Vocl uses conventional audio processing techniques to detect the start of a non-verbal cue and subsequently count the number of those cues in a fixed window of time.

Prior to detection, a filter is applied to the audio to remove low frequency noise. Next, thresholding based on RMS (root mean square) average is used to detect the start of a non-verbal cue. Then, a three-second window of audio around the detection is extracted to begin the cue counting.

To begin counting, a set of filters is applied to remove low and high frequency noise from the audio. Subsequently, a peak finding algorithm is applied to count the number of relatively large audio spikes within the window. Lastly, a peak selection algorithm is applied to determine which peak(s) can be considered an instance of a clap.

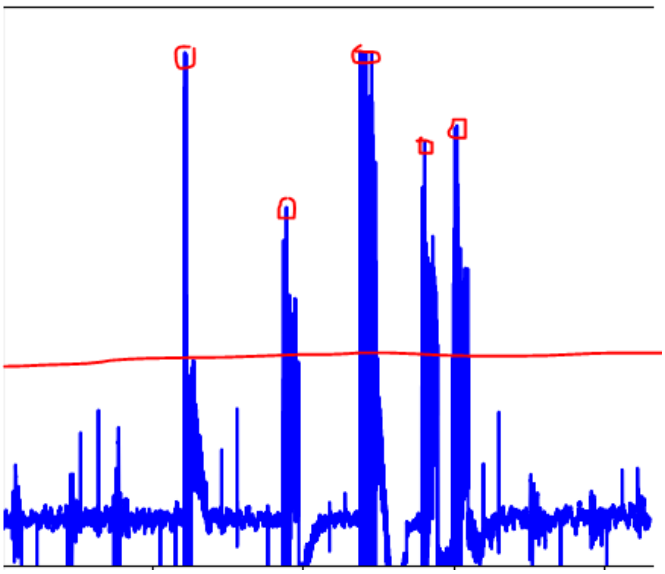


Figure 1.19 Conventional audio processing visualization. Red line: threshold level. Red circle: detect peaks.

4.3 Machine Learning

Vocl employs a machine learning model to classify the type of non-verbal cue detected by the traditional audio processing algorithm.

The model is built with the Tensorflow Python API. The model is pretrained on the AudioSet-Youtube corpus, consisting of 521 distinct audio classes. The pretrained model is then fine-tuned on data from the desired non-verbal cues as distinct audio classes.

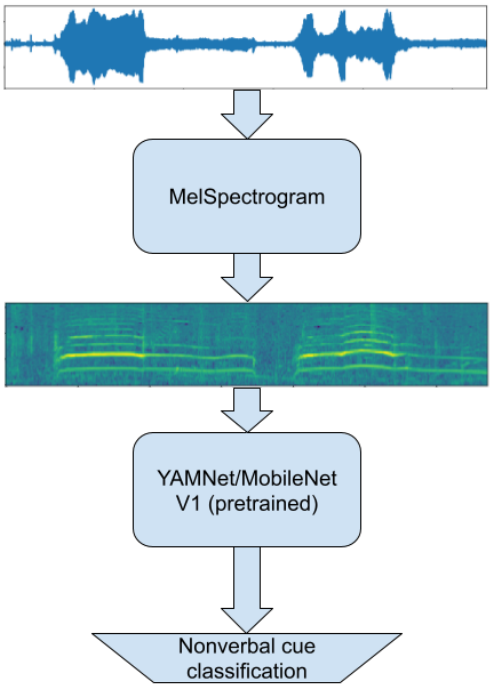


Figure 1.20 Tensorflow model pipeline

As highlighted in 1.20, the model takes an audio signal sampled at 16 kHz, converts it into a Mel Spectrogram, and fed into our fine-tuned YAMNet model which gives our desired classification. In our application, the fine-tuned model is converted to Tensorflow Lite format which reduces its computational complexity.

The model is hosted on a Python Flask server which uses HTTP to pass the window of audio detected by the conventional processing algorithm in Unity to the model and subsequently passes the audio classification to the algorithm to determine the type of non-verbal cue.

5 Relevant Engineering Standards

As a software-only project, we did not find ourselves bound to any particular set of standards when creating our project. That said, we did attempt to abide by many good coding practices. When possible, we looked to code in ways that could be described as efficient, self-documenting, modular, and safe. The Unity environment was mostly or entirely new to all of the members of our team, and so it was important to us that we take our time to properly learn our environment. We tried to keep the eventual size and power draw of our application lightweight in order to allow users to be able to play for extended periods of time on mobile devices without the need to charge.

Additionally, when we were beginning to research for our project, we wanted to follow relevant standards to allow our final product to be considered therapeutic programming for children with non-verbal autism. Ultimately, the scope of our project did not grow to such proportions that these standards could be applied.

6 Cost Breakdown

Consider your EC464 prototype to be the *alpha* version. The next unit made, according to your engineering specifications and design, would be the *beta* version. Later a manufacturing version or release-version would be made.

What would be the cost of your ***beta*** unit when it is created? This should assume market costs, i.e. no donations, no picking through the customer's parts closet.

You can edit the table below to describe the project expenses for the beta version. It is not necessary to provide every detail about parts, labor, and services in your cost breakdown. Decide upon a level of aggregation of investment and group costs accordingly.

Project Costs for Production of Beta Version (Next Unit after Prototype)				
Item	Quantity	Description	Unit Cost	Extended Cost
1 Additional Assets	Several	Assets (models, environments, and the like) purchased from the Unity store to be used in our game.	~$5-25	~$100
2				
3				
4				
5				
			Beta Version-Total Cost	$100

As we are a software-only project, we don't have any material costs to consider for our product. As such, every "unit" of our product can be made and distributed for free once it is developed. A "Beta Version" of our product could be considered to be an improved and expanded version of our product, though no less easily replicated than the previous version. For a beta version, we might assume that the game would be improved and expanded upon, therefore we would budget around $100 to bring more assets into our game to improve user experience. However, as we are a software project, the $100 cost is only for improving our product to the beta version, is not part of the cost for each individual beta distribution of the game. This number was calculated by taking the sum of assets purchased for the alpha release, totaling $113, and rounding down to the nearest hundred.

7 Appendices

7.1 Appendix A - Specifications

Specifications	Values & Units
Application Size	442.2MB.
OS	iOS 11.0 or higher.
Audio Detection	80% detection accuracy for both machine learning model and conventional peak detection algorithm. - This number includes both counting occurrences of specific inputs and differentiation between multiple different types of inputs (i.e. clap vs. shout).
Internet Connection	Minimum upload speed of 1.5 Mbps.
Latency	Under 500ms delay for communication with Flask server and Tensorflow lite model data processing.

7.2 Appendix B – Team Information

We are Team 12, developing audio and touch algorithms for non-verbal autistic children and our project is called Vocl, an iOS learning game. Our team is made up of four qualified computer engineers, Mia Hernandez, Krishan Eskew, Charles Mo, and William Nilsen. When selecting a project, all four team members were seeking a software-oriented project with a clear societal impact and this project, introduced by client Andrey Vyshedskiy, fulfilled our expectations. As we understood the lack of engaging applications, useful content, and intriguing game experiences for autistic youth we became more committed to the project. The tech stack that the project utilized fit well with our individual strengths, from William's work on machine learning to Mia's excellent front-end designs and animations. Overall, designing, developing, and introducing Vocl to our fellow students and engineering community has been a fulfilling and deeply rewarding educational experience, and we hope to be able to continue work on the project despite the Senior Design class ending.

Krishan Eskew is a graduating Computer Engineer. His passions inside the classroom include the software engineering process, computer architecture, and operating

systems. He plans to use his skills as an engineer in the software engineering field, building efficient and scalable products. Outside of his schooling and career, he enjoys cooking, powerlifting, biking, and keeping up with new music releases.

Mia Hernandez is a graduating senior majoring in Computer Engineering with a concentration in Machine Learning. Her areas of interest include software development and artificial intelligence. She plans on working as a software engineer at Raytheon Missiles & Defense after graduating.

Charles is a graduating senior majoring in Computer Engineering. His interests include real-time signal processing, software engineering, and internet of things. He plans to pursue a career in software engineering while continuing his passion of prototyping cyber physical systems.

William Nilsen is a graduating senior majoring in Computer Engineering. His work has brought him into the field of embedded systems and low-level programming. He plans to continue working in software development for the foreseeable future, and begins working at The Clearing House Payments Company this summer.