

# Developing Visual Accessibility Options to Empower Grade School Students in Designing Inclusive Mobile Applications

by

Murielle Dunand

S.B., Massachusetts Institute of Technology (2020)

Submitted to the Department of Electrical Engineering and  
Computer Science

in partial fulfillment of the requirements for the degree of  
Master of Engineering in Electrical Engineering and Computer  
Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2021

© Massachusetts Institute of Technology 2021. All rights reserved.

Author.....  
Department of Electrical Engineering and Computer Science  
May 20, 2021

Certified by.....  
Harold Abelson  
Class of 1922 Professor of Computer Science and Engineering  
Thesis Supervisor

Accepted by.....  
Katrina LaCurts  
Chair, Master of Engineering Thesis Committee

# Developing Visual Accessibility Options to Empower Grade School Students in Designing Inclusive Mobile Applications

by

Murielle Dunand

Submitted to the Department of Electrical Engineering and Computer Science

on May 20, 2021, in partial fulfillment of the

requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

## Abstract

As technology becomes more available to the general public, it is important that it be as accessible as possible. Accessibility for mobile apps is crucial given how pervasive smartphone use has become. For students who use MIT App Inventor -- an online platform that enables users to make their own mobile apps -- part of being an effective app designer is to appreciate the importance and practice of inclusive design.

I have empowered young students to make their apps more visually accessible by: (1) offering new options for larger text, higher contrast, and alternate text in App Inventor apps, (2) creating a curriculum for students aged 13-18 about the principles of visually accessible design, and (3) running the workshop three times and collecting student feedback on the curriculum. After the workshop, students reported a more accurate understanding of the nature of low vision as well as increased comfort with making visually accessible apps. Overall, this work has shown that it is both simple and effective to teach the principles of accessible design to students as young as middle-school age. The code changes have been added to MIT App Inventor and the curriculum is available on the App Inventor website.

Thesis Supervisor: Harold Abelson

Title: Class of 1922 Professor of Computer Science and Engineering

## Acknowledgements

I would like to thank the members of the MIT App Inventor team for their support and advice. I have learned more from them over the years than I can list and I am deeply grateful.

Specifically, I thank my advisor Hal Abelson for pushing me and making me think big, Evan Patton for patiently meeting with me and helping me with the code section, Jeffrey Schiller for reviewing my complete code changes, Selim Tezel for the advice on education and connecting me to teachers, and Natalie Lao and Jessica Van Brummelen for coming before me and giving advice on thesis writing.

In addition, I thank Kyle Keane and Judy Brewer for their invaluable advice on low vision and education.

Finally, I would like to thank my parents for their support and encouragement in thesis writing, and my little brother for his honest high-school perspective.

# Table of Contents

<b>Chapter 1: Introduction</b>	<b>6</b>
1.1 Web Accessibility	6
1.2 Visual Accessibility	6
1.3 MIT App Inventor	7
1.4 Research Overview	7
<b>Chapter 2: Prior Work</b>	<b>9</b>
2.1 Accessibility with block-based programming	9
2.2 Ways of evaluating accessible technologies	9
2.3 Current accessibility features on mobile phones	10
2.4 Existing student education about accessibility	10
<b>Chapter 3: Code changes to MIT App Inventor</b>	<b>11</b>
3.1 Design considerations	11
3.2 Alt text field in images	13
3.3 High contrast mode	13
3.4 Larger default text	16
3.5 Improvements shown with the Google Accessibility Scanner	16
<b>Chapter 4: Workshop Methodology</b>	<b>20</b>
4.1 Curriculum plan	20
4.2 The CoinFlipGame App	22
4.3 Data collection	24
<b>Chapter 5: Results</b>	<b>25</b>
5.1 Prior experience of the population	25
5.2 Students learned how to make visually accessible apps	26
5.3 Students gained a better understanding of what low vision is	28
5.4 Students used the new accessibility options	29
5.5 Some students went above and beyond	31
5.6 Students understood the importance of visual accessibility	32
<b>Chapter 6: Discussion</b>	<b>34</b>
<b>Chapter 7: Future Work</b>	<b>36</b>
<b>Bibliography</b>	<b>38</b>
<b>Appendix A: Code for the new features</b>	<b>40</b>
A.1 App Inventor Button component (companion app)	41
A.2 App Inventor button component (web)	43
A.3 App Inventor Screen component	45

A.4 App Inventor Image Component	47
<b>Appendix B: Workshop Materials</b>	<b>48</b>
B.1 Pre-survey	48
B.2 Main workshop slides	54
B.3 Personas	63
B.4 Post-survey	69

# List of Figures

*Figure 1: An example app with user optionality. The left image is what the app looks like when the end user presses the “High Contrast ON” button. The right image is what the app looks like when the end user presses the “High Contrast OFF” button, and is also the default button coloring for App Inventor . . . . . 12*

*Figure 2: The code required for the app in Fig. 1 to function . . . . . 12*

*Figure 3: Left: An image of an app with high contrast mode turned off. Right: An image of the same app with high contrast mode turned on. Note that because the pink button is not the default button color, it was not affected by the high contrast mode change. However, the top two buttons were default and so were set to high contrast in the right image. . . . . 14*

*Figure 4: A diagram of the positions of the HighContrast and BigDefaultText options in the Screen properties tab. . . . . 15*

*Figure 5: An image of a test app with no added accessibility features. . . . . 17*

*Figure 6: An image of a test app with high contrast on. . . . . 18*

*Figure 7: An image of a test app with high contrast and large text mode on. . . . . 19*

*Figure 8: A screenshot of the main screen of CoinFlipGame. . . . . 23*

*Figure 9: An example of a shared slide for students to write on. Students are assigned by name to each slide so that they know where to write, and they answer the set questions as a group. . . . . 24*

*Figure 10: A chart detailing overall student answers to the pre-survey question: “How much experience do you have with MIT App Inventor?” . . . . . 25*

*Figure 11: A chart detailing overall student answers to the pre-survey question: “What exposures have you had to visually accessible design in the past?” . . . . . 26*

*Figure 12: A chart comparing all pre-survey responses to all post-survey responses of how comfortable students felt making basic App Inventor apps. . . . . 27*

*Figure 13: A chart comparing all pre-survey responses to all post-survey responses of how comfortable students felt making visually accessible App Inventor apps. . . . . 28*

*Figure 14: The original CoinFlipGame (left) and a student’s new version using large text and high contrast (right). . . . . 30*

*Figure 15: The original CoinFlipGame (left) and a student’s new version using large text and a larger coin image (right). . . . . 31*

*Figure 16: A student version of the CoinFlipGame with a font size incrementer (left) and a different student’s version with a large button to enable dark mode that has just been pressed (right) . . . . . 32*

*Figure 17: The original DatePicker interface (left) and the proposed new DatePicker interface (right). . . 36*

# Chapter 1

## Introduction

According to the World Health Organization, there are 285 million people with visual impairments worldwide [1]. Of these, 39 million people with visual impairments are categorized as blind and 246 million are considered Low Vision (LV). The number of children with LV is three times higher than the number of children with blindness.

MIT App Inventor seeks to empower learners to make an impact in their communities [2], but impact is lessened if members of those communities are unable to use those apps due to vision impairments. However, many students may not be aware of this facet of design. Given that low vision is one of the most common chronic disabilities, addressing it offers a strong base for accessible education; my hope is that including a full App Inventor curriculum about the importance of visual accessibility will impress upon students the importance of general inclusivity in their app design.

### 1.1 Web Accessibility

Web and mobile accessibility focus on ensuring that people with disabilities can use online services easily and effectively. Currently, many sites and tools are developed with accessibility barriers that make them difficult or impossible for some people to use. The WAI (Web Accessibility Initiative) offers strong standards of accessibility. The WAI states that the types of disabilities that affect web use are: “auditory, cognitive, neurological, physical, speech, and visual” [3]. Accessibility is also helpful for web users without disabilities, such as people using small screens or a slow internet connection. For websites and mobile apps, accessibility is morally important and beneficial for business, as well as often required by law. Many aspects of accessibility are easy to implement for designers, such as setting a larger font size or ensuring that an image has alternate text.

### 1.2 Visual Accessibility

Visual accessibility centers on accessibility for users with a visual disability. There are many types of visual disabilities, including blindness, low vision, and colorblindness. Each type has different considerations in terms of web accessibility. This thesis focuses on visual accessibility

for app users with low vision. Low vision is generally defined as a condition in which a person's vision cannot be fully corrected by glasses and interferes with daily activities such as reading and driving [4]. Low vision often develops with age, and makes itself known in conditions such as macular degeneration, glaucoma, cataracts, and retinoplasty. This can cause blurring or distortion at the center or edges of vision, and make screen use challenging without assistive devices and accessibility standards.

### 1.3 MIT App Inventor

MIT App Inventor is an online platform that empowers learners to make their own mobile apps. App Inventor users can design their apps on the web by going to the App Inventor platform and making their app out of “components” on the Designer tab of App Inventor. Components are different functions of apps, and are separated into visible and non-visible components. Visible components show up on the completed app screen, while non-visible components are not part of the user’s interface , but provide access to app functions. A visible component might be a button, a label, or a check-box. A non-visible component might be a text-to-speech function, an internal clock, or a cloud database. To code the functionality of apps, designers switch to the “Blocks” section of the platform and snap together different preset blocks of code to program different behaviors. This approach is also known as block-based programming. Once the app has been designed on the web, designers can test out and distribute their apps on mobile devices for community use. These apps often reach a wide audience, and for everyone to use them easily, they must be accessible.

### 1.4 Research Overview

In this work, I contribute (1) new options for larger text, higher contrast, and alternate text in App Inventor apps, (2) a curriculum for students aged 13-18 about the principles of visually accessible design, (3) results of three workshops run with the curriculum, and (4) discussion of the most effective ways to teach visual accessibility to young students. In chapter 2, I discuss the prior research on visual accessibility and block-based programming. In chapter 3, I explain my coding methodology as well as the features of high contrast, large text, and alternate text that I added to App Inventor. In chapter 4, I detail the structure of my workshop, the premade app that students modified, and the data collection procedure. I go over my results in chapter 5. Namely, students gained significant confidence in making visually accessible apps, having overall come in with little App Inventor experience and no accessibility experience. The



workshops empowered students to understand the importance of visually accessible design, with several students adding advanced accessibility features. I discuss the further implications of the results in chapter 6, and conclude that teaching the basics of visually accessible design to young students is simple and effective. In Chapter 7, I complete the thesis by discussing avenues for future projects that arose from my work.

# Chapter 2

## Prior Work

### 2.1 Accessibility with block-based programming

Block-based languages tend to already be more accessible than other programming languages due to their more intuitive methods of snapping relevant blocks together to make code [4]. For individuals with motor impairments, the program Myna [5] offers a vocal user interface so that users can program block-based languages by voice. Although originally developed to work with Scratch v1.4, Myna has been extended to work with additional block-based languages, including Lego Mindstorms, Scratch v2.0, and Blockly. By their very nature, block-based programming environments aim to be more accessible, especially to new learners, “by simplifying the mechanics of programming, by providing support for learners and by providing students with motivation to learn to program” [6]. Ludi presents the drawbacks of block-based programming for blind users due to the visually centered nature of the interface and mouse-centric input [7]. However, interfaces that use Blockly (such as App Inventor) are based on JavaScript and CSS, and so “widgets can be described...., keyboard navigation can be provided, [and] clearly articulated properties for drag-and drop, widget states, or areas of a page can be updated over time or based on an event”. This offers significant potential in improving App Inventor’s accessibility. Illustrating this, Giurleo developed keyboard interactions in App Inventor that allow users to navigate the interface using key commands rather than a mouse, aiding those with motor impairments [8]. In this research, I take advantage of App Inventor’s JavaScript base to add additional features that target people with visual impairments.

### 2.2 Ways of evaluating accessible technologies

General accessibility for tech has become a major concern for many groups [9]. A main accessibility tester for Android apps is the Google Accessibility Scanner, an application that runs in the background of the chosen app and offers suggestions based on the W3C accessibility guidelines [10]. Additionally, Vontell also developed Bility, another Android app scanner that streamlines the process compared to the Google Accessibility Scanner [11]. Testers with an Apple device can use VoiceOver to test out a screen reader, or run the Accessibility Inspector within iOS Simulator [12]. I will use these testers to try to get a unified assessment of the accessibility of apps and then move on to surveying humans.

## 2.3 Current accessibility features on mobile phones

Android has several options for accessibility, including high-contrast text and buttons. It also has a dedicated color modification mode. It is relevant to consider whether these accommodations already solve accessibility needs within App Inventor. While the color modification is effective for those who are colorblind, its high contrast modes do not greatly affect the appearance of the typical App Inventor app, making built-in high contrast in App Inventor apps a very desirable new feature. As shown in Chapter 3.5, I run the Google Accessibility Scanner on apps with having the main accessibility options enabled, namely high contrast text, high contrast buttons, and screen reader. Even with these features on, many App Inventor visual components fall short of the W3C guidelines the scanner uses.

## 2.4 Existing student education about accessibility

Many courses are aimed at teaching sighted students the importance of universal design. Wang details a holistic course for teaching web accessibility to undergraduate students. She raises the important point that accessibility functions should be “seen as an integral part of the website development process, rather than something extra to be added at the end of the process. Keeping accessibility in mind encourages students to make better design decisions along the way and seek optimal methods to ultimately achieve Web accessibility” [13]. With this approach, including both an accessibility curriculum and new accessibility features is likely the most effective method to teach students its importance. This curriculum includes lectures and projects, and focuses on visual accommodations that app developers should consider. Though Wang’s curriculum is aimed at undergraduate students in web development, I have referred to her course as I designed my middle- and high school curriculum due to its focus on visual impairment and its project options. For low vision specifically, the Fred Hollows Foundation offers a low vision simulator with varying severities of cataracts, glaucoma, and retinopathy [14] which offers a great experiential activity for students.

# Chapter 3

## Code changes to MIT App Inventor

The code changes made in this chapter aim to empower App Inventor designers to create apps that are accessible to low vision app users. While it is beneficial to also make the App Inventor platform itself accessible to low vision users, this is currently outside the scope of my work.

Chapter 7 details additional approaches to improve the accessibility of App Inventor.

To make the changes to MIT App Inventor, I forked the Github repository at <https://github.com/mit-cml/appinventor-sources>, where App Inventor offers its code in open source format. I first ran my changes on my local server, then hosted and tested the changes through the Google App Engine on a server that testers could access.

### 3.1 Design considerations

Before making the changes to App Inventor, I consulted with several low vision users and professionals. I met several times with lecturer Kyle Keane, instructor for the MIT course “6.811: Principles and Practices of Assistive Technology” [15]. He offered guidance on what components to prioritize and how to improve the preliminary code changes I made. I also met twice with Judy Brewer, Director of the Web Accessibility Initiative (WAI) at the World Wide Web Consortium (W3C) [16]. She helped focus my thesis scope on low vision and set me on the path to making personas for the curriculum section of this work. Both emphasized the importance of optionality for the wide range of low vision users.

Due to these valuable discussions, I prioritized coding features that had a high level of optionality for many types of low vision. Therefore, all changes are set in the properties of App Inventor components that the user can toggle on or off. Depending on the type of low vision for which the app designer is building, they can choose to use some or all of the new features. Similarly, it was important in my design that these new options be affected by the Blocks tab in App Inventor. This allows the end user to change the accessibility properties from the completed app, as long as the app designer codes that functionality into the app. For example, a user could press a button marked “High contrast mode” on the live app and it would enable the high contrast mode in the same manner as marking it in the preferences tab. This allows app designers to incorporate the accessibility options in the layout of their apps so that low vision users can pick their preference.

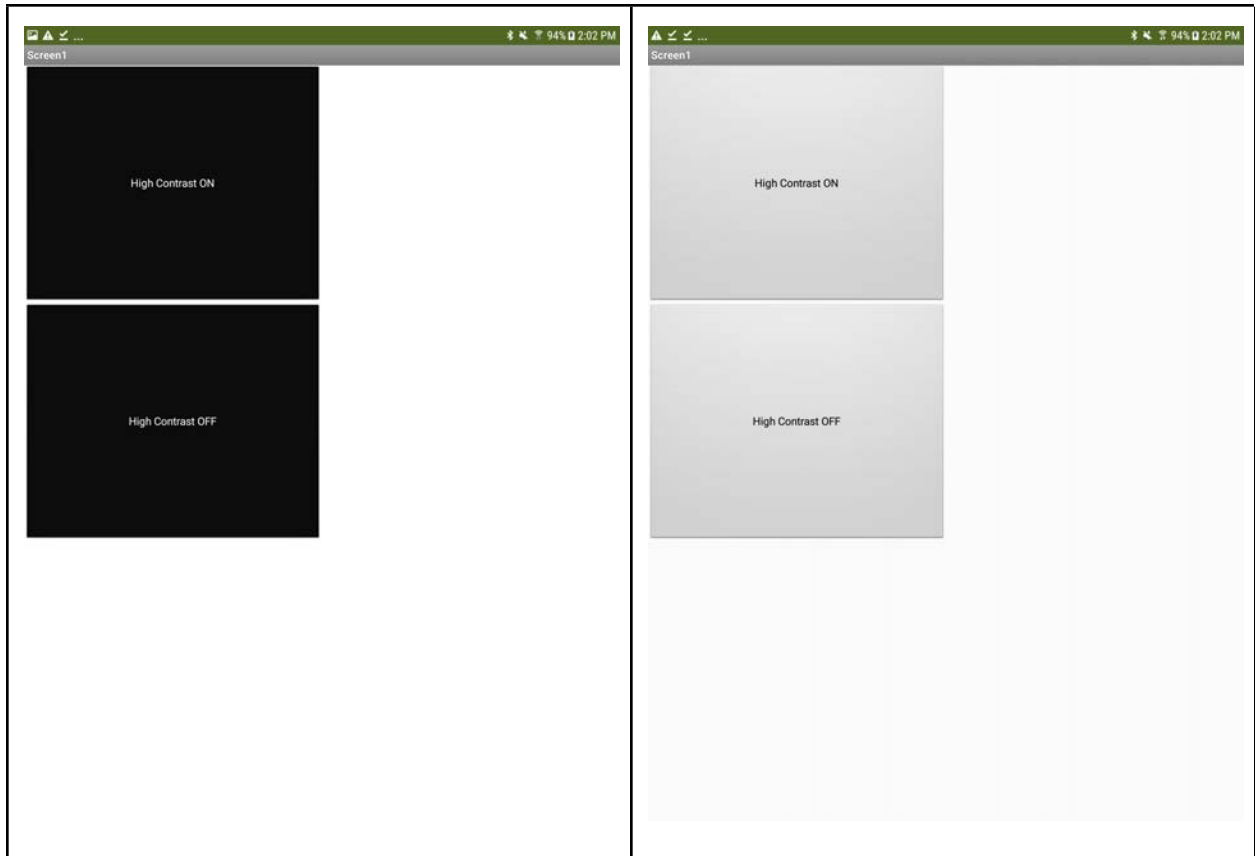


Figure 1: An example app with user optionality. The left image is what the app looks like when the end user presses the “High Contrast ON” button. The right image is what the app looks like when the end user presses the “High Contrast OFF” button, and is also the default button coloring for App Inventor.

```

when HighContrastButtonON .Click
do set Screen1 . HighContrast to true

when HighContrastButtonOFF .Click
do set Screen1 . HighContrast to false

```

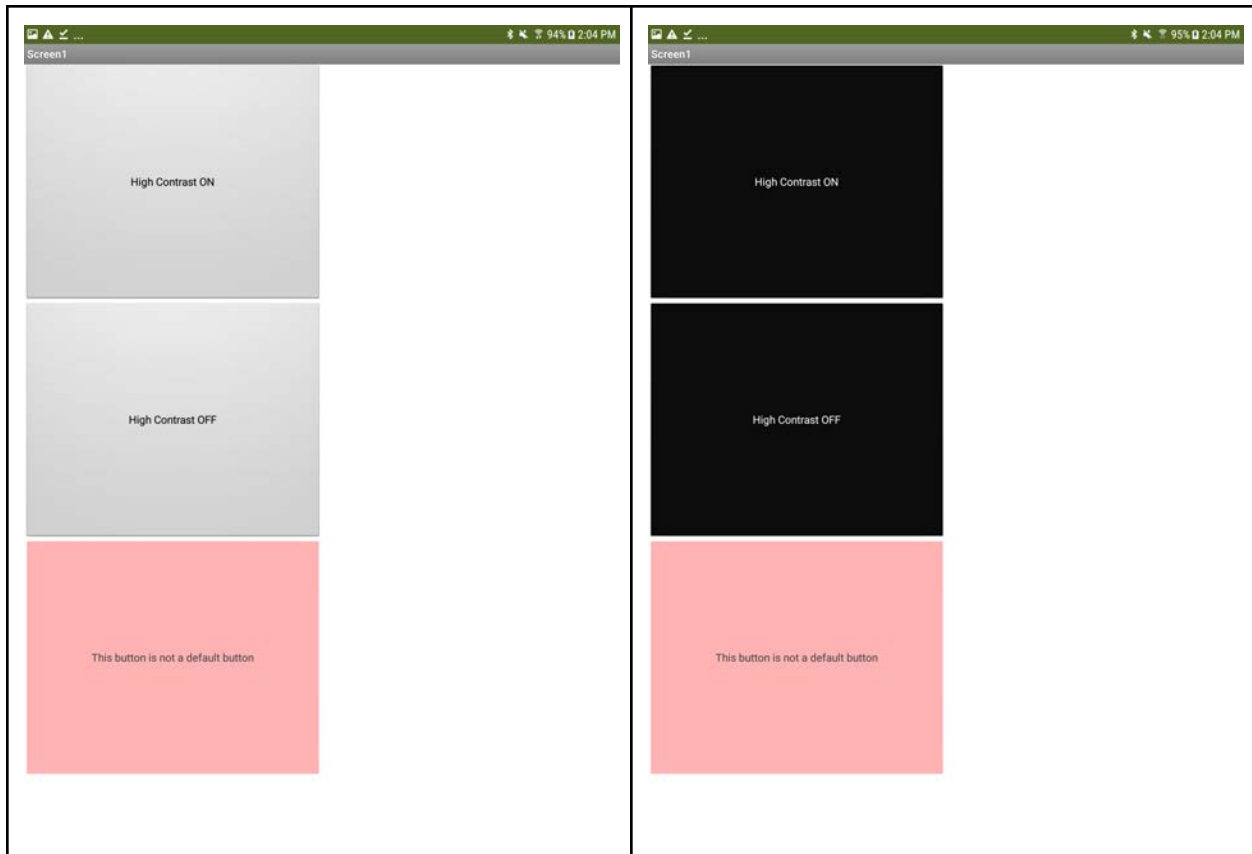
Figure 2: The code required for the app in Fig. 1 to function.

## 3.2 Alt text field in images

Alternate text for images is an important aspect of web and app accessibility. App users with low vision often use text-to-speech functions on their mobile devices to read out loud what may be too difficult to see. This is why important images need alternate text to be visually accessible. If an image is too small or too low-contrast for a user to distinguish, the text-to-speech function can read the alternate text for that image aloud. Typically, alternate text is a direct and concise description of the corresponding image. The Image component in App Inventor displays a specified image on the mobile app screen. I added a new property to the Image component called "AlternateText". The app designer can type text into the "AlternateText" field, and when an app user tries the built app with a text-to-speech assistive device, the "AlternateText" field is compatible and the assistive device is able to read the text provided for that image component.

## 3.3 High contrast mode

One of the goals of this research is to ensure that making apps more visually accessible is easy for the average app designer. The implemented high-contrast mode sets several visual components to a high contrast version if they were previously in their default state. Thus, if the color of a button was changed from the default grey, it will not change in appearance when high contrast mode is turned on. This is because I assume that a designer has a reason for changing text or button color, and would not appreciate that change being overridden by high contrast mode. This check-box, which can be toggled at will, is named HighContrast and is a property of the Screen component and thus native to every app.



*Figure 3: Left: An image of an app with high contrast mode turned off. Right: An image of the same app with high contrast mode turned on. Note that because the pink button is not the default button color, it was not affected by the high contrast mode change. However, the top two buttons were default and so were set to high contrast in the right image.*

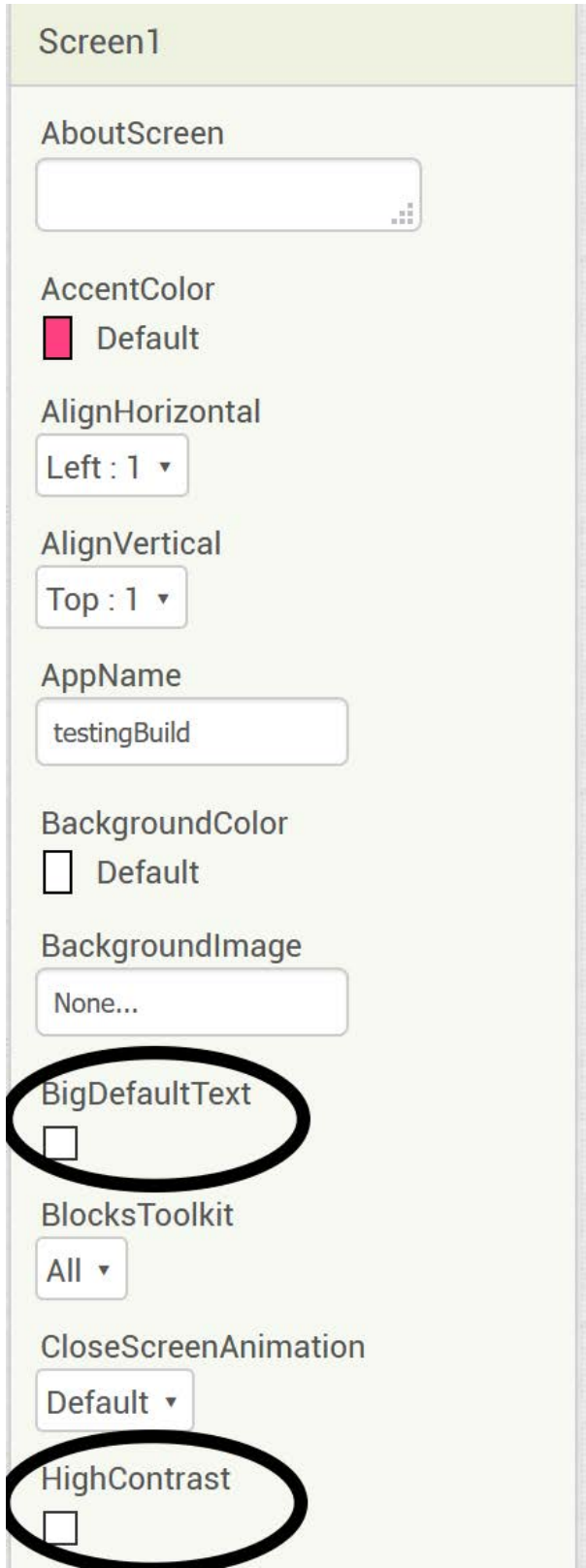


Figure 4: A diagram of the positions of the HighContrast and BigDefaultText options in the Screen properties tab.



### 3.4 Larger default text

Similar to the high contrast mode, this large text mode only affects text that is displayed at its default font size (14 point) and changes it to be much larger (24 point). This way, any fonts that were set to a specific size by the designer are not affected. Like the HighContrast field, this check-box is found in the properties of the Screen component and can be toggled at will.

### 3.5 Improvements shown with the Google Accessibility Scanner

As stated in section 1.4, I used the Google Accessibility Scanner to find App Inventor components that needed accessibility changes. Figures 5, 6 and 7 illustrate the results of running an example App Inventor app through the scanner.

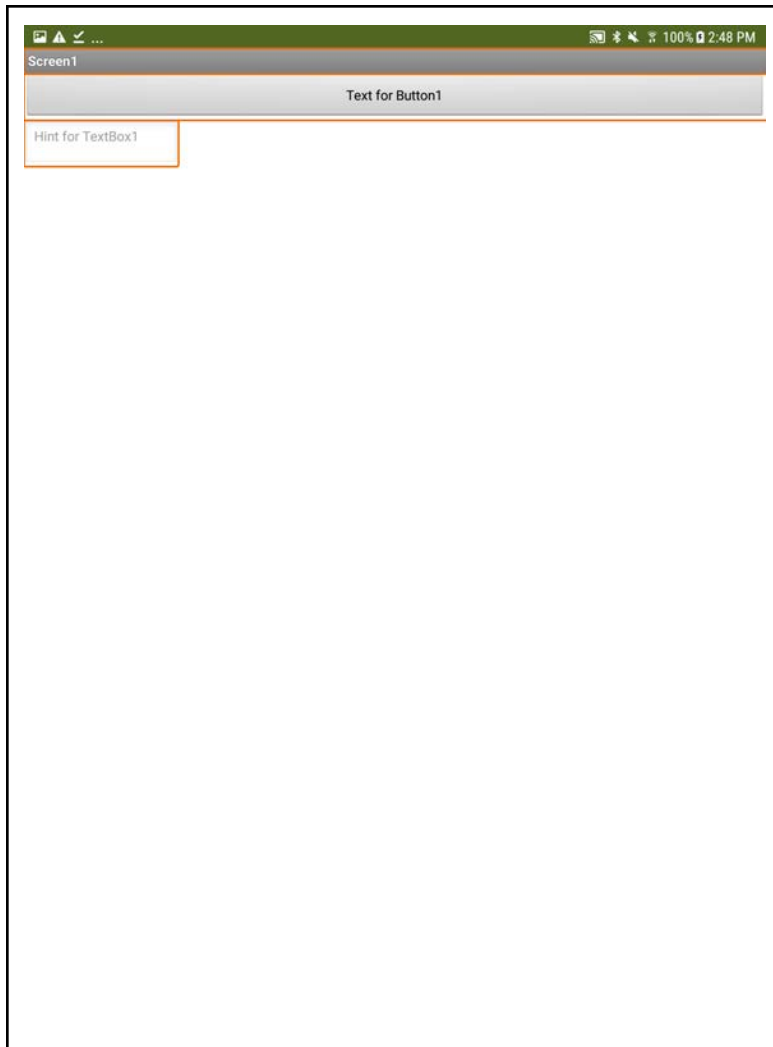


Figure 5: An image of a test app with no added accessibility features.

Figure 5 is a screenshot of an unmodified app in the original version of App Inventor. The accessibility scanner has identified three areas of possible poor accessibility by boxing them in orange. From the top of the app to the bottom, these are:

- The grey action bar at the top is low contrast
- The large button is low contrast
- The empty text box with grey hint text is low contrast

The results of the accessibility scanner after implementing and turning on the high contrast mode for the same app are shown in Figure 6.

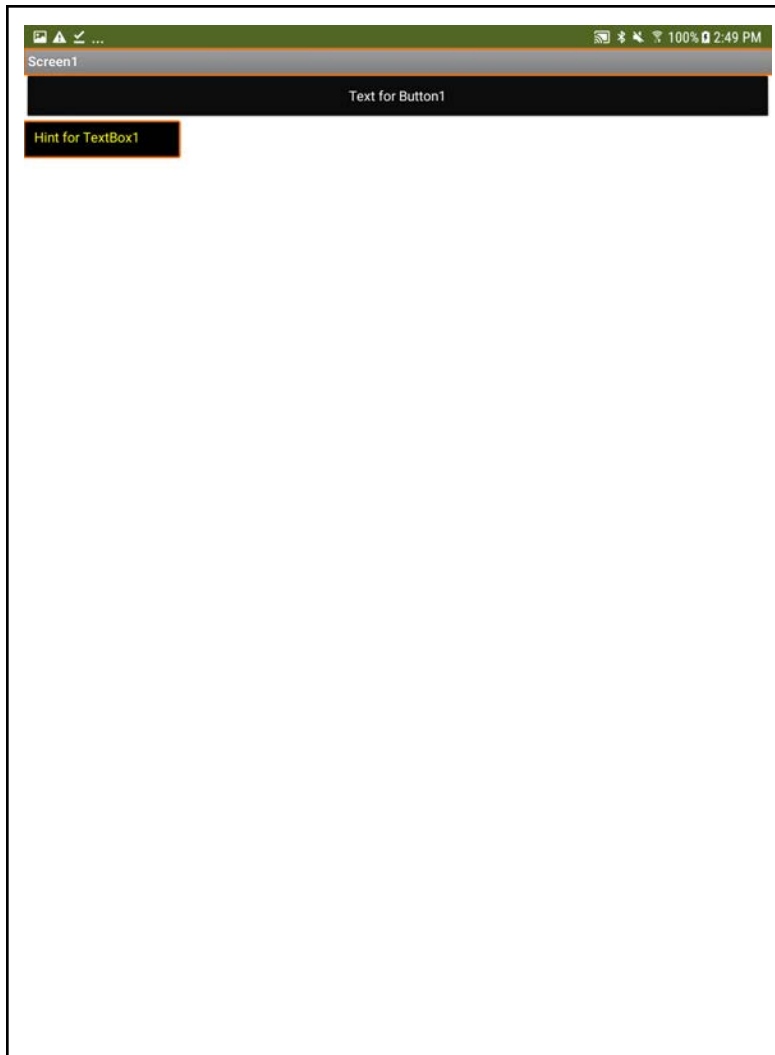
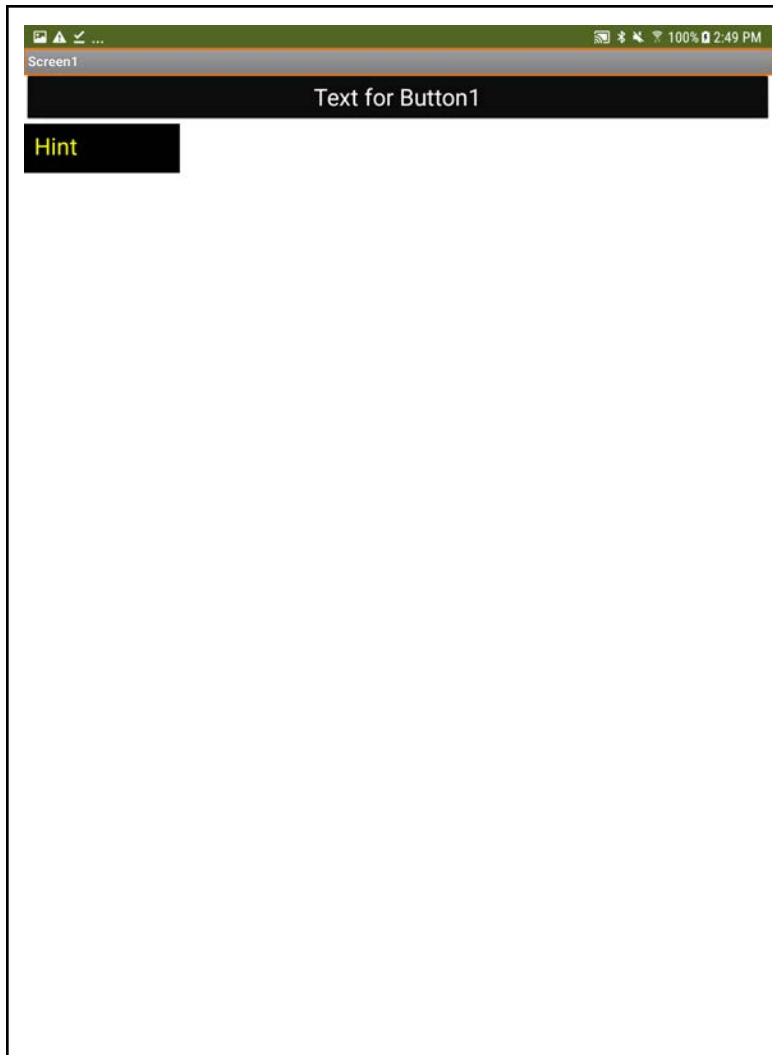


Figure 6: An image of a test app with high contrast on.

Figure 6 is a screenshot of the same test app with high contrast mode turned on. The accessibility scanner now identified only two areas for improvement. From the top of the app to the bottom, these are:

- The grey action bar at the top is low contrast
- The empty textbox with yellow hint text is too small

After implementing and turning on large text mode as well as high contrast mode, the final result is shown in Figure 7.



*Figure 7: An image of a test app with high contrast and large text mode on.*

With both large text and high contrast (Figure 7), the only issue that the accessibility raises is the low contrast of the top grey bar. Overall, it is clear that adding high contrast and large text mode improves app performance in accessibility checks. However, it is important to also test how these changes are received by app users. In Chapter 4, I discuss how the curriculum was designed to improve understanding of accessibility and the App Inventor changes.

# Chapter 4

## Workshop Methodology

To test the changes made to the App Inventor interface and discover if young App Inventor students, who were not experts on visual accessibility, would understand the above changes, I designed and taught three workshops consisting of a guide to visual accessibility and hands-on work with App Inventor.

I ran three such workshops, for a total of 30 students aged 12-18. The first workshop was on March 20, 2021 through the MIT eSpark program. This program is for students aged 12-13, and students pay a flat fee of \$40, then apply to courses through a lottery. The course I organized, named “Visual Accessibility with MIT App Inventor” got over 200 students in its lottery, indicating that there is a high level of interest for such content. However, due to staffing limitations, I could only accept 20 students. On the day of the workshop, 15 of these 20 students attended the workshop.

The other two workshops were on April 12, 2021, with two different teachers who are part of the Mobile Computer Science Principles program (Mobile CSP). Mobile CSP is a high-school NSF-funded effort to teach students computer science using App Inventor. The first workshop on April 12 had a class of 12 students, and the second workshop had a class of 4 students. Both of the Mobile CSP workshops consisted of high school students.

For all three workshops, the courses were designed for a 90-minute instruction time, for students aged 12-18. The workshops were conducted over Zoom, and students received and fulfilled consent forms and basic set-up instructions before the workshops.

### 4.1 Curriculum plan

For each workshop, the general schedule was:

Title	Description	Time
Welcome and Introduction	Class introduction. Students and teachers introduce themselves.	5 minutes

Low vision simulation	Students try out a low vision simulator on their browser and discuss what they see	5 minutes
Low vision overview	Instructor goes through definitions of low vision, visual accessibility, and App Inventor interface as needed	5 minutes
Persona Discussion	Students split into groups and receive a “persona” of a person with low vision for whom they will design the app.	10 minutes
App Testing	Students test out a pre-loaded app and discuss in groups how their assigned persona would like or dislike this app	10 minutes
Accessible App Design	Students use the additions to App Inventor to make the pre-loaded app more visually accessible for their persona	30 minutes
Presentation and Discussion	Students share with the class what changes they made to the pre-loaded app to improve visual accessibility, and turn in their changed apps.	10 minutes
Post-survey	Students take a short survey to give feedback on the quality of the course	10 minutes

Students were expected to set up before class by completing the pre-survey as well as setting up their Android devices with the experimental version of the MIT App Inventor companion. They also accessed the local server on the web, where a sample project, CoinFlipGame, was available in a repository.

The **Low Vision Simulation** asked students to visit the site <https://simulator.seenow.org/> on their mobile devices. This low vision simulator accesses the camera of the mobile device and places a virtual filter over it, simulating different types of low vision. Students were also encouraged to use the available sliders on the site to change the severity of low vision, and to swap between glaucoma, cataracts, macular degeneration, and diabetic retinopathy. Then, the class reconvened to discuss what students had seen.

For the **Persona Discussion**, students were split into groups of three, except for the class of four students where each student instead worked individually. Each “group” was assigned to

analyze a persona of a low vision user . The personas spanned a large range of age, gender, nationality, and low vision type to offer a diverse set of analyses (see Appendix section B.3). Each of the 6 available personas had a short profile and distinct goals. The population of personas consisted of:

- a 68-year-old male professor with cataracts;
- a 21-year-old female student with migraines and light sensitivity;
- a 43-year-old female accountant with retinopathy from diabetes;
- a 70-year-old female retiree with glaucoma and limited central field vision;
- a 12-year-old colorblind male middle-schooler;
- a 16-year-old male student with retinopathy of prematurity and patchy vision loss.

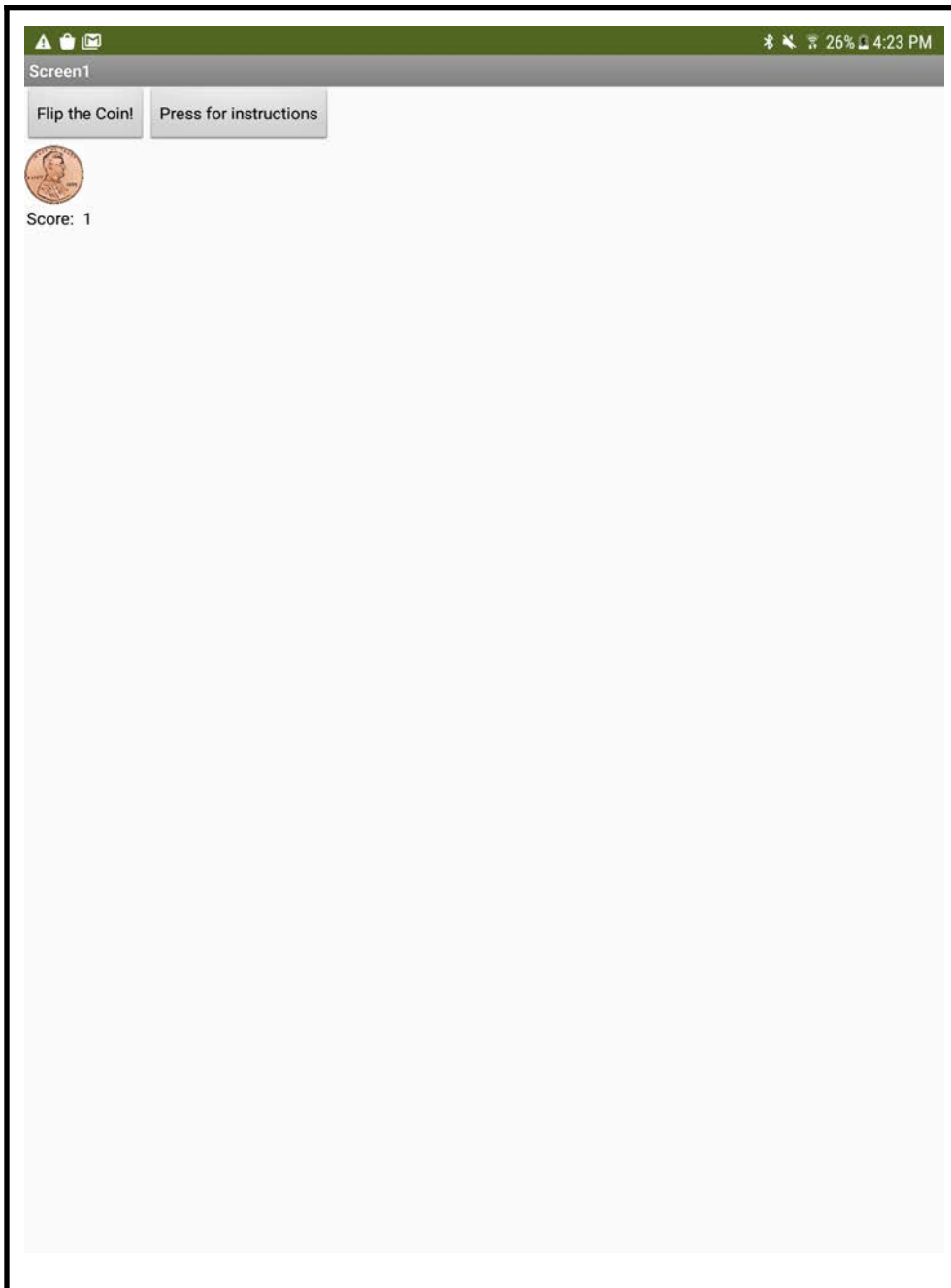
Since the workshops were handled remotely, students in the same group wrote on a shared Google Slides slide, and the entire slideshow was saved.

For the **App Testing**, students returned to the same groups as in the persona discussion. They then played with the premade app (CoinFlipGame) and critiqued its design with their low vision persona in mind. Like in the previous persona discussion, students wrote their opinions on a shared Google Slides document.

The **Accessible App Design** section started with my going over the new changes to MIT App Inventor. I demonstrated how the HighContrast, BigDefaultText, and AlternateText fields were used by sharing screens with the class with App Inventor open. Then, students worked individually to improve their local versions of CoinFlipGame. They were encouraged to discuss the process with their peers or ask an instructor for help if needed.

## 4.2 The CoinFlipGame App

The premade app that students critiqued was a simple game app. The game starts with a point tally of 0, and the player can press a button to flip a coin. If the coin shows heads, the player gains a point. If the coin shows tails, the player loses a point. At 3 total points, the player wins the game and the app shows a win screen. At -3 total points, the player loses the game and the app shows a loss screen. Regardless of win or loss, the player is presented with the option to play again.



*Figure 8: A screenshot of the main screen of CoinFlipGame.*

The CoinFlipGame app used all default values in its properties, including font size, button size, and screen color. It did not utilize any of the new accessibility options.

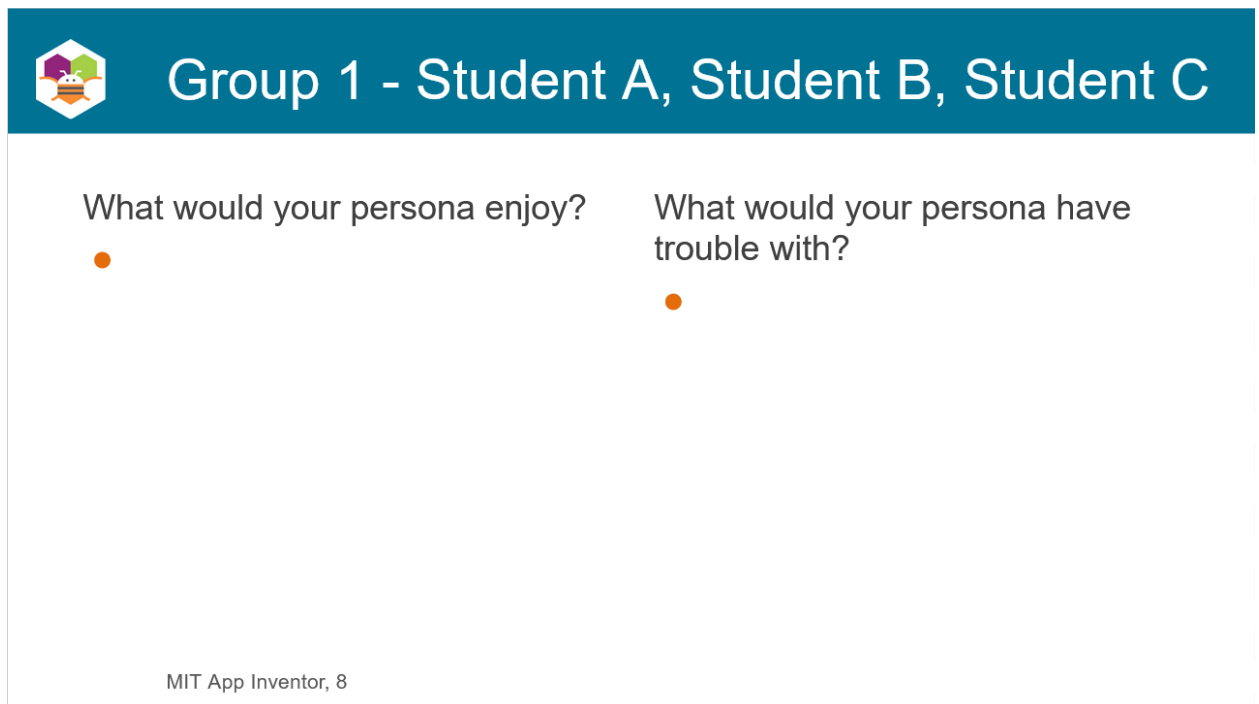



## 4.3 Data collection

Data were collected using the pre- and post-surveys, as well as written results on the slides completed during the workshop. Informed consent was obtained from all workshop participants. Since most of the workshop participants were under 18, parents signed the consent forms while the students signed assent forms per COUHES guidelines. The workshops had a population of 15 middle-schoolers and 15 high-schoolers. 2 total middle-schoolers signed the consent forms and did both surveys, while 13 high-schoolers did the same.

Two types of data were collected: written and App Inventor project files. The written data consisted of the two surveys as well as comments written during the workshop, and the App Inventor project files consisted of the built files of the improved app made by the students. The pre-survey contained 5 scale questions and 3 free response questions. The post-survey was made up of 3 scale questions and 6 free response questions.

Because the workshops were done remotely, students were requested to write group solutions on a set of shared Google Slides so that their work could be reviewed and discussed in real time. Data were only collected for groups of students who had all completed the consent forms.



 **Group 1 - Student A, Student B, Student C**

What would your persona enjoy?

What would your persona have trouble with?

MIT App Inventor, 8

*Figure 9: An example of a shared slide for students to write on. Students are assigned by name to each slide so that they know where to write, and they answer the set questions as a group.*

# Chapter 5

## Results

### 5.1 Prior experience of the population

All of the results shown in this chapter combine responses across all three workshops. There is one set of responses for all pre-surveys, and another one for all post-surveys. A question on the pre-survey asked students “How much experience do you have with MIT App Inventor?” Students could select values 1-4, where 1 is “None at all” and 4 is “a lot”.

Of the 23 valid pre-survey responses, 12 reported that they knew “a moderate” amount of App Inventor. Only 1 person said that they knew “A lot”, and two reported “none at all”. This is not surprising, because the workshops were made up of students just starting to learn App Inventor. Both of the “None at all” responses came from the middle-school group, who had had less time to be familiar with App Inventor and were not part of a Mobile CSP course.

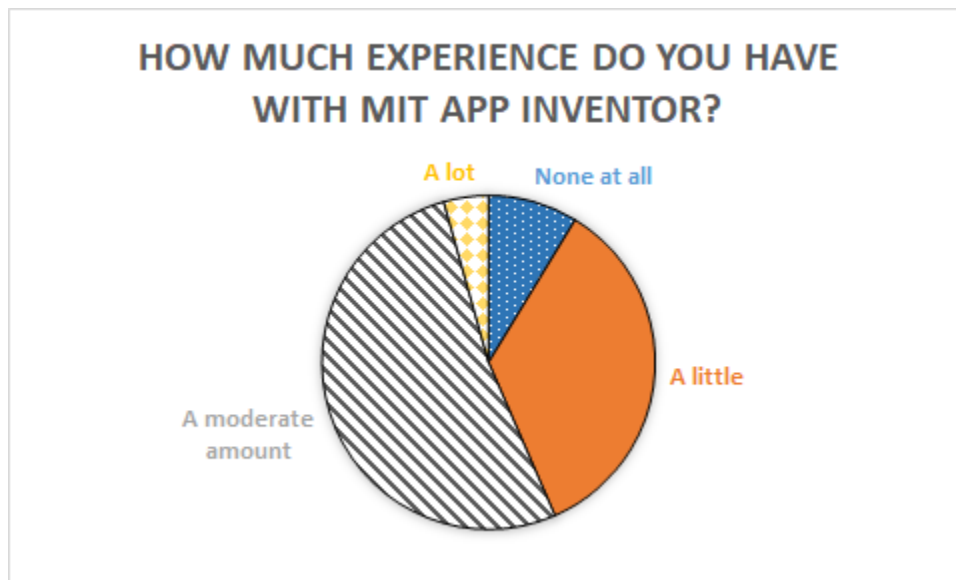


Figure 10: A chart detailing overall student answers to the pre-survey question: “How much experience do you have with MIT App Inventor?”

The pre-survey asked “What exposures have you had to visually accessible design in the past?”

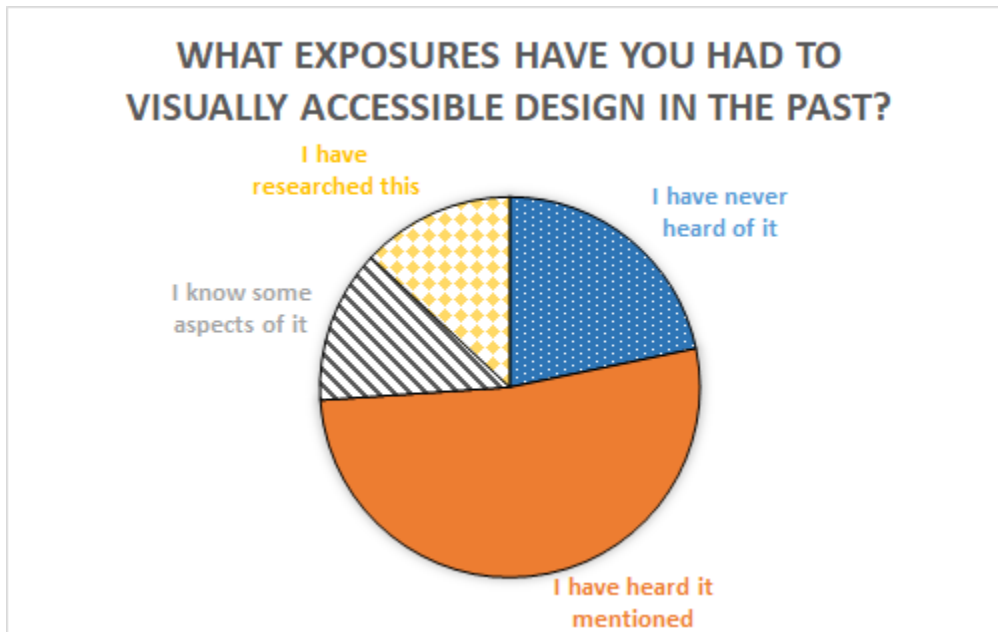


Figure 11: A chart detailing overall student answers to the pre-survey question: “What exposures have you had to visually accessible design in the past?”

74% of students had either never heard of visual accessibility or only heard it mentioned. This is partly why the workshop includes an overview of the nature of visual accessibility and its importance in app design. This result is also informative of later results, since the intended audience for these changes are App Inventor users who may not necessarily be professionals in accessible design. Overall, the surveyed population tended to have low to medium experience with App Inventor, and low experience with visually accessible design.

## 5.2 Students learned how to make visually accessible apps

Figures 12 and 13 illustrate the changes in student sentiment for several questions that were asked in both pre- and post-surveys. First, students were asked to rank how much they agreed with the statement: “I feel comfortable making basic App Inventor apps”.

## I feel comfortable making basic App Inventor apps

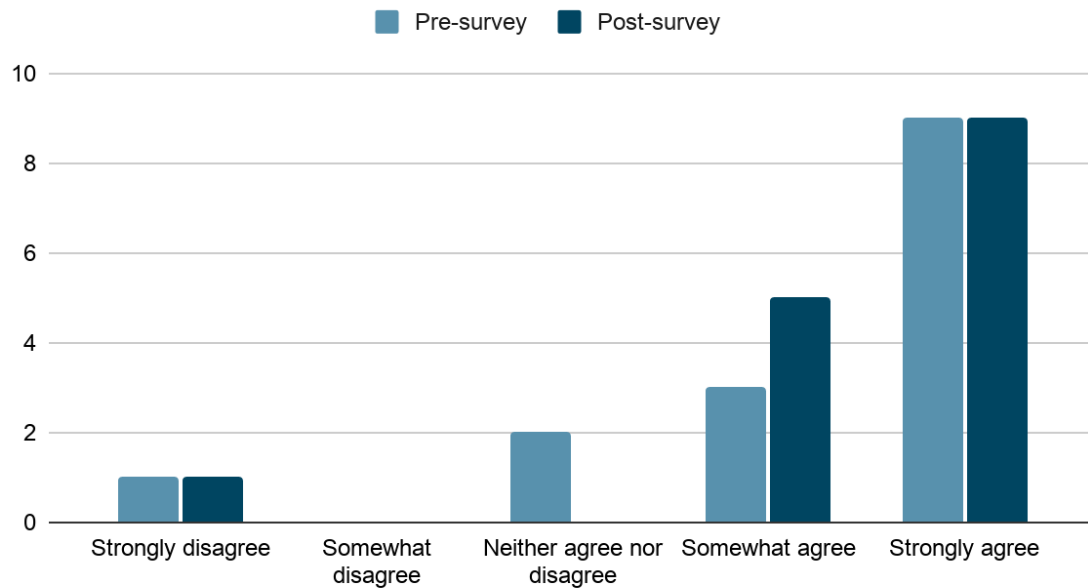


Figure 12: A chart comparing all pre-survey responses to all post-survey responses of how comfortable students felt making basic App Inventor apps.

As can be seen in Figure 12, there was little change in how comfortable students felt about making App Inventor apps. Several students went up from “Neither agree nor disagree” to “Somewhat agree” on their confidence, but this may not be significant. This lack of change is consistent with the fact that this workshop was not about designing basic App Inventor apps *per se*, but rather about visual accessibility within App Inventor.

Next, students were asked to report on each survey how much they agreed with this statement: “I feel comfortable making visually accessible App Inventor apps”.

## I feel comfortable making visually accessible App Inventor apps

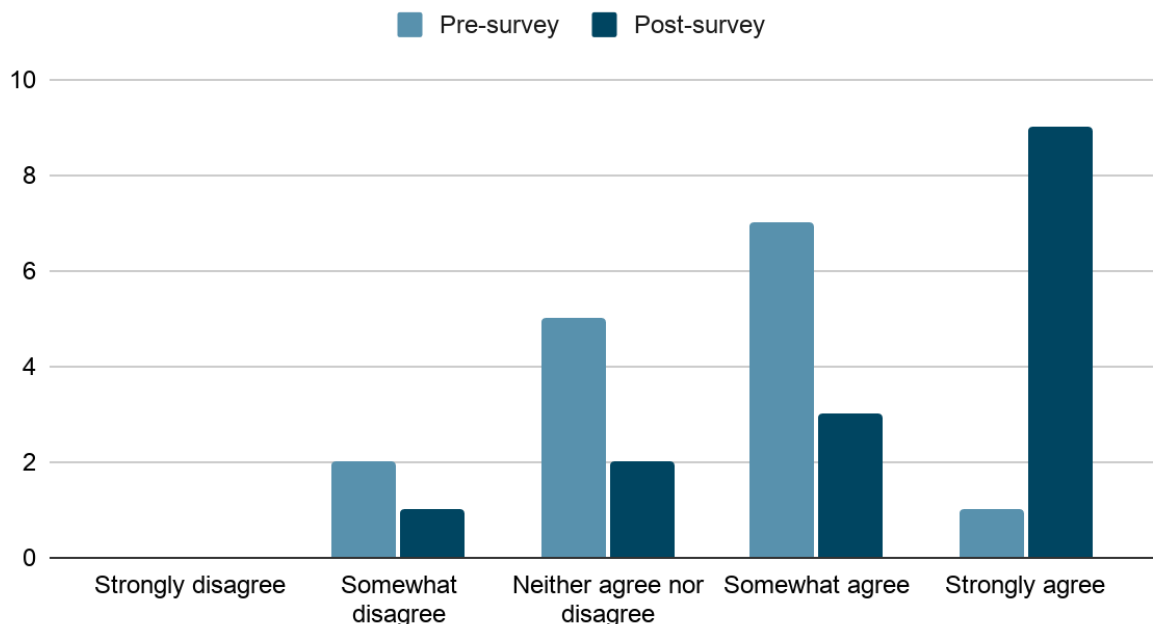


Figure 13: A chart comparing all pre-survey responses to all post-survey responses of how comfortable students felt making visually accessible App Inventor apps.

Here, we see a noticeable increase in students strongly agreeing that they are comfortable making visually accessible apps. This field was the only one to increase in population, from 1 to 9, between surveys. This result implies that students learned enough about visually accessible design to improve their confidence in it, which was the goal of the workshop. This also suggests that workshop materials were effective and could be used as future curriculums for visual accessibility.

### 5.3 Students gained a better understanding of what low vision is

Another question that was asked on both pre- and post-surveys was “What would you say it means for a person to have low vision? Responses in the pre-survey were fairly basic. Two responses mentioned needing glasses, and most of the other answers were variations on “not being able to see well”. Below are several samples from the pre-survey responses.

*“They have problems seeing things, either with color, distance, etc”*

*“Poor eyesight, maybe they need glasses can't see much at distances”*

*“I think it means that a person with low vision has a hard time seeing things that are right in front of them”*

It is clear that there is a range of different responses between students, pointing out various parts of low vision. After the post-survey, the answers became more general:

*“Someone whose vision is impaired in some way. It can be near- or far- sighted, blurriness, partial blindness, warped vision.”*

*“I would say that it means a variety of different things. They have trouble seeing things overall whether that be size color or any other area of functionality.”*

Several responses also included specific examples of low vision conditions, such as glaucoma or cataracts, which had been mentioned as examples during the workshop. Overall, answers became more holistic to encompass a wider spectrum of low vision conditions.

## 5.4 Students used the new accessibility options

Many methods exist to make apps more visually accessible. One of the considerations for this thesis was to see if students would use the newly implemented options (high contrast mode, big text) while improving the CoinFlipGame app. This was verified by asking students to submit their completed app files to the investigator for review. All of the students used some combination of the high contrast and large text modes for their final apps. The combinations of modifications varied depending on the persona for whom the students were making the app. For example, one student had a persona with light sensitivity, so their final app had a grey background instead of the default white background to make it gentler on the eyes.

We can see in Figure 14 that students used the high contrast mode to change the button color and increase the font size. Other common changes included increasing the size of the coin image.

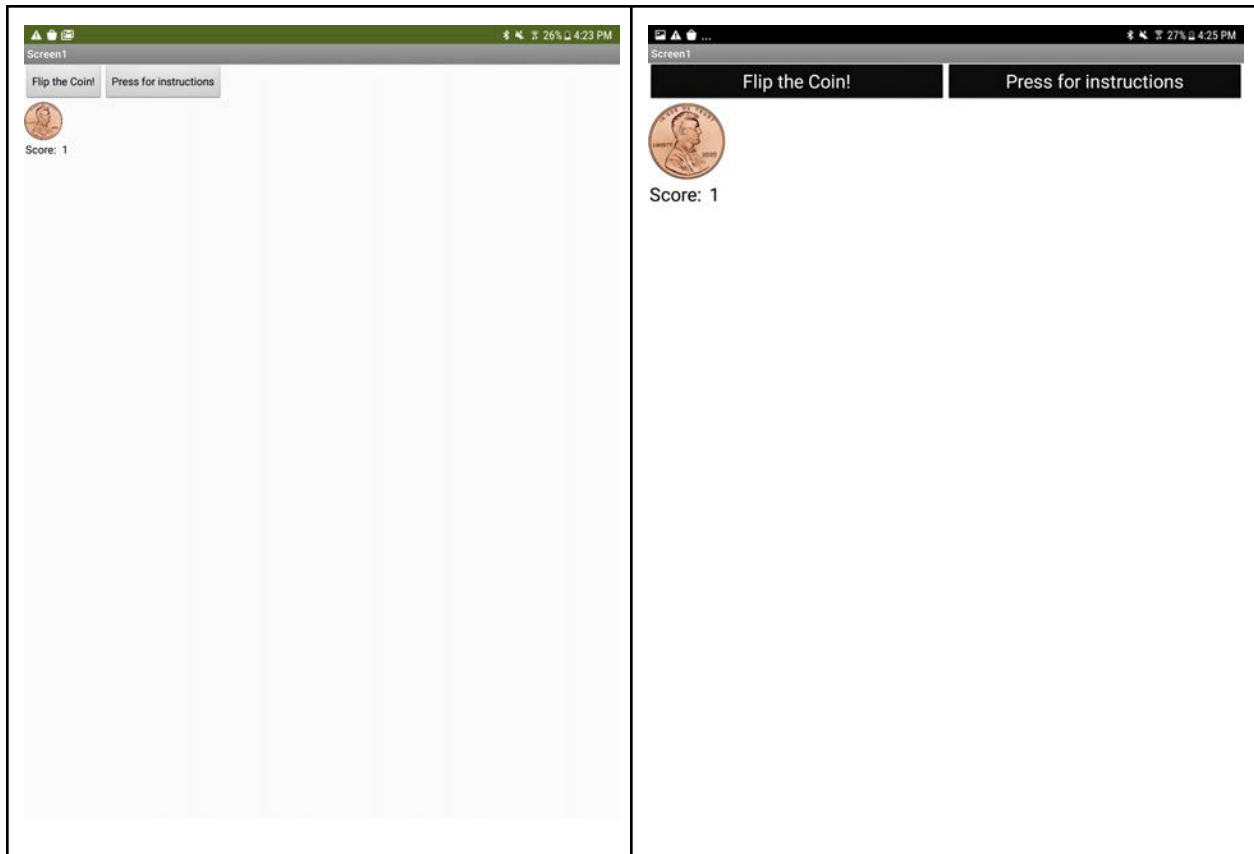


Figure 14: The original CoinFlipGame (left) and a student's new version using large text and high contrast (right)

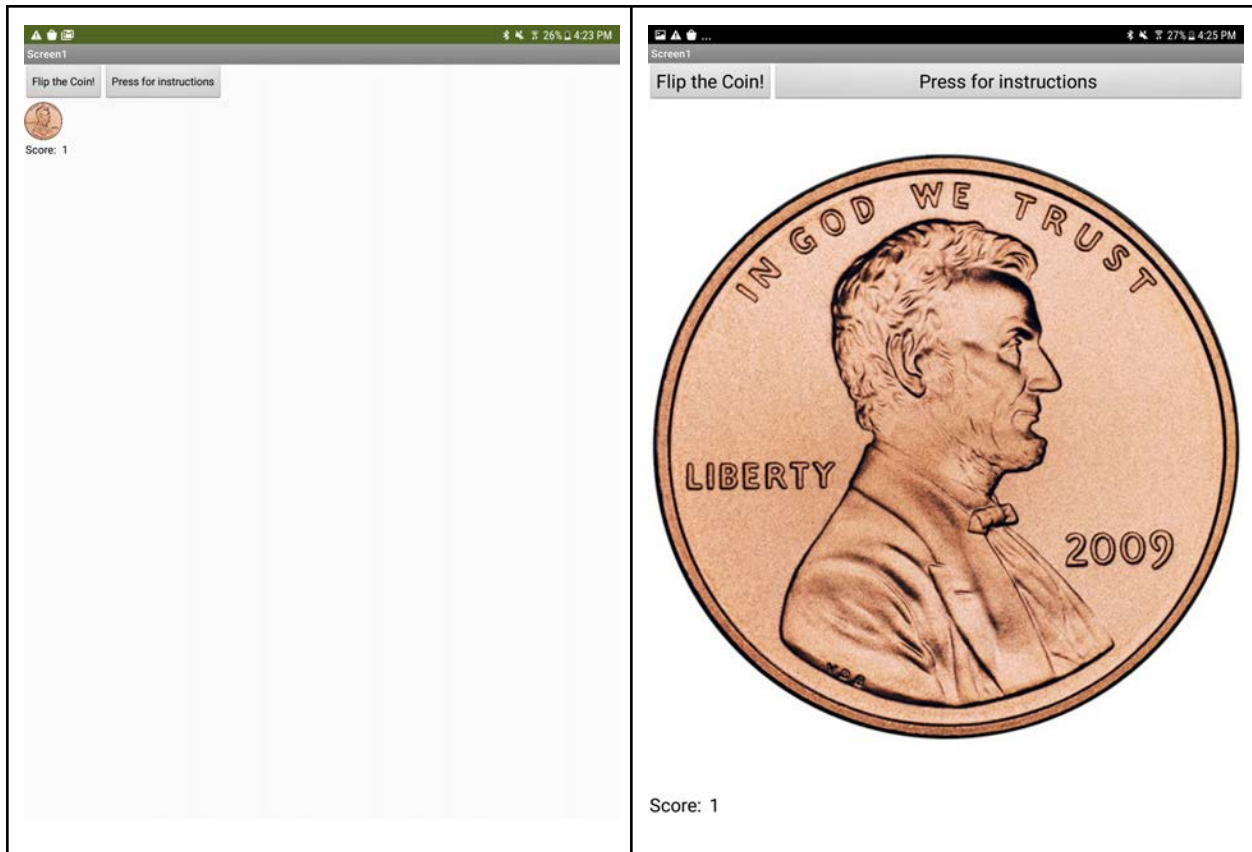


Figure 15: The original CoinFlipGame (left) and a student's new version using large text and a larger coin image (right).

After a short demonstration, students were interested in using the changes to App Inventor and demonstrated that interest by including the features in their personal versions of the CoinFlipGame app. Overall, students used a mixture of new features and existing features (e.g., increasing image size) to make their apps more visually accessible.

## 5.5 Some students went above and beyond

Since this workshop only required a basic knowledge of MIT App Inventor, students were able to make all basic changes directly in the Designer section of App Inventor. However, the option was left open for students to go into the Blocks section and change the actual programming behind the CoinFlipGame app to improve accessibility. Several of the more experienced students took this option. One student used the existing TextToSpeech component in App Inventor to read the text of buttons when they were pressed.



Adding new code blocks also allowed for increased optionality. One student made two new working buttons to increase and decrease the font size to personal preference. Another one set a toggle for a dark mode, since they were working with the light sensitive persona.

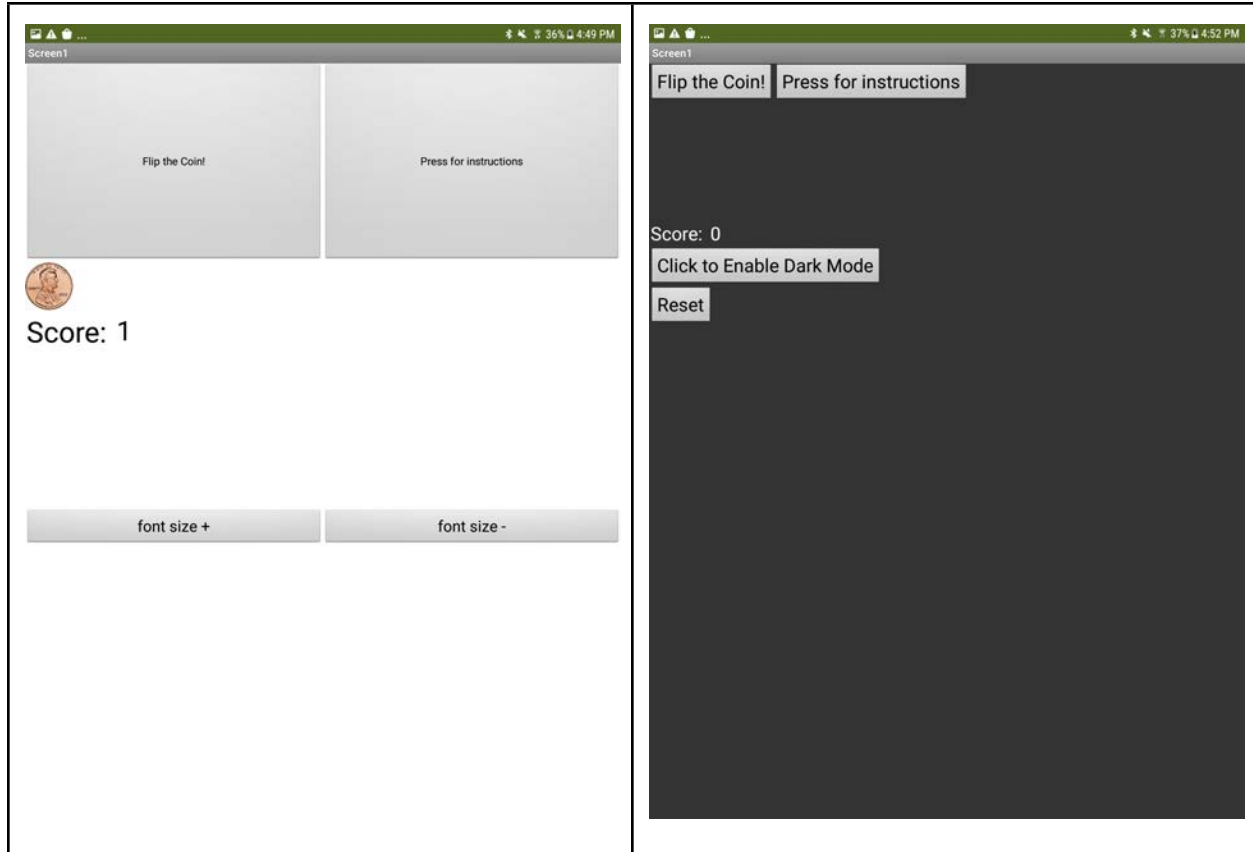


Figure 16: A student version of the CoinFlipGame with a font size incremter (left) and a different student's version with a large button to enable dark mode that has just been pressed (right).

This demonstrates that additional accessible options can be implemented in App Inventor with a knowledge of its blocks and components. I recommend a more bloc-based version of this curriculum as a course for advanced App Inventor users.

## 5.6 Students understood the importance of visual accessibility

An important consideration is whether students understand why visual accessibility is important, rather than simply how to build it. If students had the technical knowledge to make apps visually accessible but did not consider it important, they would be unlikely to actually put that

knowledge to use. I did not add an explicit question to the post-survey of how likely they would be to make accessible apps in the future, since answers may be skewed due to the desire to please the instructor. Instead, I checked the answers to the post-survey question “What were 2-3 things you learned over the course of this workshop?” to discover whether students would volunteer that information themselves. Of the 15 full responses to the post-survey, 7 of them included understanding that making visually accessible apps was important in general app design. This included quotes such as:

*“As a app creator you need to have your app accessible to people. Another thing is that sometimes you need to have someone critique your work.”*

*“Visually impaired people need help when it comes to apps It’s better to help more people be able to use your app”*

*“To be more attentive to the visual aspects of a project, and not give up easiliy.”*

Another six responses stated that they learned what visual accessibility or low vision was during the workshop, including:

*“vision impairment isn’t limited to blindness, blindness isn’t always a complete lack of vision, including visually accessible design does not need to be hard”.*

About half the students reported an understanding of the importance of inclusive app design, rather than simply an understanding of its definition. This is an important additional step to include in all related workshops to ensure that the accessibility skills taught are put to practice when students decide to make their own apps.

Overall, students showed a great increase in understanding of how to make accessible apps, what low vision is, and why it is important. This workshop is best suited for students with low experience with App Inventor and little to no prior knowledge of visual accessibility. For students with high experience with App Inventor, an alternate and more challenging version of this workshop using the text-to-speech component is recommended. In all, these results indicate that accessibility education for students using MIT App Inventor is highly effective and thus enables a greater output of accessible apps.

# Chapter 6

## Discussion

The vision of this research is to educate young students in visually accessible app design and empower them to use these techniques in future design projects and careers. To do this, I: (1) developed new accessibility features for completed App Inventor apps, (2) created workshop curriculum to teach visual accessibility principles and app design through the MIT App Inventor tool, (3) taught this workshop three times with groups of middle- and high-school age students. My major findings are summarized below:

- **Because low vision users are an extremely broad group, it is important to include optionality when offering accessibility options.** Thus, my changes include simple toggles and fully programmatic functionality for both app developers and users.
- **There is strong interest in visually accessible design from students.** The MIT eSpark program received over 200 applicants, and the Mobile CSP recruiting also received more interested teachers than I could work with. This is promising news for future accessibility courses using App Inventor.
- **Students can easily understand and utilize the coded changes to App Inventor.** Surveyed students showed a clear increase in their understanding of making accessible apps, and a vast majority of changed apps used the high contrast, big text, and alternate text options described in the workshop. Students did not have questions as to how to use these options and easily integrated them into their projects.
- **Students and educators are passionate about accessible design.** In addition to the high level of interest in the workshops, students reported an understanding of why visually accessible design is important and how increased inclusivity is an improvement to apps. Several students and teachers asked to be notified when the changes would be available on the live App Inventor platform. Some students considered accessibility in terms of themselves and their loved ones.

This work has shown that it is both simple and effective to teach the principles of accessible design to students as young as middle-school age. After the workshop, students reported a more accurate understanding of the nature of low vision as well as increased comfort with making visually accessible apps. Being able to consider accessibility is a powerful skill for young

students. This additional consideration for other populations can offer a head start in any design career; outside of careers, an inclusive view is consistently beneficial. Through the surveys, I found that students gained greater understanding of visually accessible app design in just one 90 minute workshop. Therefore, I recommend that these visually accessible features be included in current App Inventor curricula. I believe that, with proven student enthusiasm and early curriculum, we can foster a drive for inclusive design in tomorrow's engineers.

# Chapter 7

## Future Work

There are many possible extensions to the technical work done here as well as many other questions that can be asked from the educational research point of view. First, there are some App Inventor components that need visually accessible updates. Chief amongst them is the DatePicker component, a mobile phone widget that shows an interactive calendar when pressed. The current version of the DatePicker has lower contrast compared to my proposed alternative.

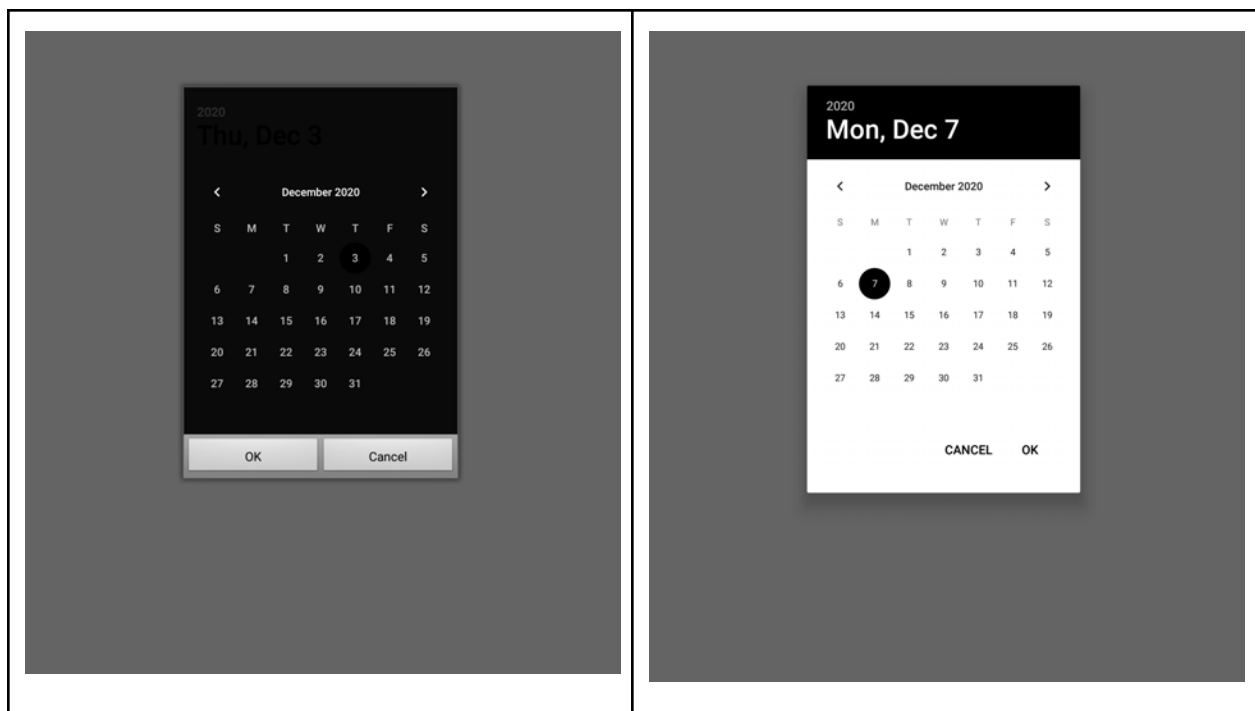


Figure 17: The original DatePicker interface (left) and the proposed new DatePicker interface (right)

In addition, the top bar of App Inventor apps was shown by the accessibility scanner to be too low contrast. Other App Inventor components, such as checkboxes and sliders, require increases in size as well.

Secondly, the scope of this project was focused on low vision app users, with the designers mostly being sighted. It is important to also make sure that the App Inventor

developer interface is visually accessible, and not only its completed apps. A project that includes visual optionality on the App Inventor web interface would surely be welcome.

Expanding the scope of this project, it is important for App Inventor to be accessible to all of its users with disabilities, not only those with low vision. This should include users with full blindness, motor impairments, and learning disabilities. A good place to start with other types of accessibility would be allowing apps designers to move components and code blocks using voice or key commands rather than requiring mouse input.

On the education side, these workshops were conducted entirely remotely due to COVID-19 concerns. It would be useful to make the workshop materials available in an in-person format, to allow teachers to use the materials in a traditional classroom environment. This would entail changing the shared slides to printable worksheets as well as several other considerations. Overall, however, the workshop is fairly flexible in terms of whether it is conducted in-person or remotely. A greater sample size of students would of course always be useful.

In addition, the workshop was well-received by middle-school students but less data were able to be collected about them. Running this workshop for middle-school students with more formal feedback could prove its effectiveness with that age group.

Finally, since these changes are being made available to the entire App Inventor curriculum, some form of guide to the new changes should be made available to new and existing users, so that they can take advantage of the new features. This guide could be in the form of workshop materials, live presentation, or a video guide. For the workshops to reach a larger population, translation into different languages would be welcome. Some form of integration into current App Inventor tutorials would go a long way to making these changes clear to App Inventor users.

# Bibliography

1. "Blindness and vision impairment - World Health Organization." 8 Oct. 2019, <https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment>. Accessed 8 Sep. 2020.
2. "About Us - MIT App Inventor." <https://appinventor.mit.edu/explore/about-us.html>. Accessed 4 April. 2021.
3. (WAI), W3C Web Accessibility Initiative. "Introduction to Web Accessibility." Web Accessibility Initiative (WAI), W3C, [www.w3.org/WAI/fundamentals/accessibility-intro/#main](http://www.w3.org/WAI/fundamentals/accessibility-intro/#main).
4. "Visual Disabilities - Low Vision." WebAIM, WebAIM, [webaim.org/articles/visual/lowvision](http://webaim.org/articles/visual/lowvision).
5. Wagner, Amber, et al. "PROGRAMMING BY VOICE: A HANDS-FREE APPROACH FOR MOTORICALLY CHALLENGED CHILDREN." Proceedings of the 2012 ACM Annual Conference Extended Abstracts on Human Factors in Computing Systems Extended Abstracts - CHI EA '12, 2012, doi:10.1145/2212776.2223757.
6. Kelleher, Caitlin, and Randy Pausch. "Lowering the Barriers to Programming: a Survey of Programming Environments and Languages for Novice Programmers ." ACM Computing Surveys, vol. 37, no. 2, 2005, pp. 83–137., doi:10.1145/1089733.1089734.
7. S. Ludi, "Position paper: Towards making block-based programming accessible for blind users," 2015 IEEE Blocks and Beyond Workshop (Blocks and Beyond), Atlanta, GA, 2015, pp. 67-69, doi: 10.1109/BLOCKS.2015.7369005.
8. E Giurleo, "Keyboard Interactions in Block-Based Coding Environments" <https://drive.google.com/file/d/0B8AUNoeCnCCASzUwMWtwWUhmN1E/view>
9. WebAIM. (2020, April 14). Introduction to Web Accessibility. WebAIM - Introduction to Web Accessibility. <https://webaim.org/intro/>.
10. "Get started with Accessibility Scanner - Google Support." <https://support.google.com/accessibility/android/answer/6376570?hl=en-GB>. Accessed 8 Sep. 2020.
11. Vontell, Richard Aaron. "Bility : Automated Accessibility Testing for Mobile Applications." MIT Libraries, 2019. <https://dspace.mit.edu/handle/1721.1/121685>. Accessed 8 Sep. 2020.
12. "About Accessibility Verification on iOS - Apple Developer." 23 Apr. 2013, <https://developer.apple.com/library/archive/technotes/TestingAccessibilityOfiOSApps/TestingtheAccessibilityofiOSApps/TestingtheAccessibilityofiOSApps.html>. Accessed 8 Sep. 2020.

13. Wang, Ye Diana. "A Holistic and Pragmatic Approach to Teaching Web Accessibility in an Undergraduate Web Design Course." Proceedings of the 13th Annual Conference on Information Technology Education - SIGITE '12, Oct. 2012, doi:10.1145/2380552.2380568.
14. Hollows, Fred. Fred Hollows - Sight Loss Simulator, The Fred Hollow Foundation, [www.hollows.org/sightsimulator/](http://www.hollows.org/sightsimulator/).
15. Keane, Kyle. "Kyle Michael Keane." KYLE MICHAEL KEANE, [www.kylekeane.com/](http://www.kylekeane.com/).
16. "Judy Brewer, Bio." W3C, W3C, 2018, [www.w3.org/People/Brewer/](http://www.w3.org/People/Brewer/).



## Appendix A: Code for the new features

All code snippets show what was modified for this thesis. Any highlighted section shows an addition to the original code. If no part is highlighted, then the whole selection was added for this thesis. Note that sections A.1 and A.2 show changes in code for the ButtonBase component only, but analogous changes (not included in this appendix) were made for the Textbox, PasswordTextbox, Label, and Sliders components.

## A.1 App Inventor Button component (companion app)

```
@Override
public void setHighContrast(boolean isHighContrast) {
    //background of button
    if (backgroundImageDrawable == null && shape == Component.BUTTON_SHAPE_DEFAULT
        && backgroundColor == Component.COLOR_DEFAULT) {
        if (isHighContrast) {
            ViewUtil.setBackgroundDrawable(view, drawable: null);
            ViewUtil.setBackgroundDrawable(view, getSafeBackgroundDrawable());
            view.getBackground().setColorFilter(Component.COLOR_BLACK, PorterDuff.Mode.SRC_ATOP);
        } else {
            ViewUtil.setBackgroundDrawable(view, defaultButtonDrawable);
        }
    }

    //color of text
    if (textColor == Component.COLOR_DEFAULT) {
        if (isHighContrast) {
            TextViewUtil.setTextColor(view, Color.WHITE);
        } else {
            TextViewUtil.setTextColors(view, defaultColorStateList);
        }
    }
}

@Override
public boolean getHighContrast() { return isHighContrast; }

@Override
public void setLargeFont(boolean isLargeFont) {
    if (TextViewUtil.getFontSize(view, container.$context()) == 24.0 ||
        TextViewUtil.getFontSize(view, container.$context()) == Component.FONT_DEFAULT_SIZE) {
        if (isLargeFont) {
            TextViewUtil.setFontSize(view, size: 24);
        } else {
            TextViewUtil.setFontSize(view, Component.FONT_DEFAULT_SIZE);
        }
    }
}

@Override
public boolean getLargeFont() { return isBigText; }
```

---

```

/**
 * Specifies the text color of the `%type%` as an alpha-red-green-blue
 * integer.
 *
 * @param argb text RGB color with alpha
 */
@DesignerProperty(editorType = PropertyTypeConstants.PROPERTY_TYPE_COLOR,
    defaultValue = Component.DEFAULT_VALUE_COLOR_DEFAULT)
@SimpleProperty
public void TextColor(int argb) {
    // TODO(user): I think there is a way of only setting the color for the enabled state
    textColor = argb;
    if (argb != Component.COLOR_DEFAULT) {
        TextViewUtil.setTextColor(view, argb);
    } else {
        if (isHighContrast || container.$form().HighContrast()){
            TextViewUtil.setTextColor(view, Color.WHITE);
        }
        else {
            TextViewUtil.setTextColors(view, defaultColorStateList);
        }
    }
}
}
}

```

## A.2 App Inventor button component (web)

```
@Override
public void onComponentPropertyChanged(MockComponent component, String propertyName, String propertyValue) {
    if (component.getType().equals(MockForm.TYPE) && propertyName.equals("HighContrast")) {
        setBackgroundColorProperty(getPropertyValue(PROPERTY_NAME_BACKGROUND_COLOR));
        setTextColorProperty(getPropertyValue(PROPERTY_NAME_TEXT_COLOR));
        updatePreferredSizeOfButton();
        refreshForm();
    }
    else if (component.getType().equals(MockForm.TYPE) && propertyName.equals("BigDefaultText")) {
        setFontSizeProperty(getPropertyValue(PROPERTY_NAME_FONT_SIZE));
        updatePreferredSizeOfButton();
        refreshForm();
    }
}
```

```
/*
 * Sets the button's FontSize property to a new value.
 */
private void setFontSizeProperty(String text) {
    float convertedText = Float.parseFloat(text);
    if (convertedText == 14.0 || convertedText == 24.0) {
        MockForm form = ((YaFormEditor) editor).getForm();
        if (form != null && form.getPropertyValue(name: "BigDefaultText").equals("True")) {
            MockComponentsUtil.setWidgetFontSize(buttonWidget, size: "24");
        } else {
            MockComponentsUtil.setWidgetFontSize(buttonWidget, size: "14");
        }
    } else {
        MockComponentsUtil.setWidgetFontSize(buttonWidget, text);
    }
}
```

## A.3 App Inventor Screen component

```
/**
 * HighContrast property getter method.
 *
 * @return true if we want high contrast mode
 */
@SimpleProperty(category = PropertyCategory.APPEARANCE,
    description = "When checked, we will use high contrast mode")
public boolean HighContrast() { return highContrast; }

/**
 * When checked, there will be high contrast mode turned on.
 *
 * @param highContrast true if the high contrast mode is on
 */
@DesignerProperty(editorType = PropertyTypeConstants.PROPERTY_TYPE_BOOLEAN,
    defaultValue = "False")
@SimpleProperty
public void HighContrast(boolean highContrast) {

    //this.scrollable = scrollable;
    this.highContrast=highContrast;
    setHighContrastRecursive( container: this, highContrast);
    recomputeLayout();
}

private static void setHighContrastRecursive(ComponentContainer container, boolean enabled) {
    for (Component child : container.getChildren()) {
        if (child instanceof ComponentContainer) {
            setHighContrastRecursive((ComponentContainer) child, enabled);
        } else if (child instanceof AccessibleComponent) {
            ((AccessibleComponent) child).setHighContrast(enabled);
        }
    }
}
}
```

```

/**
 * BigDefaultText property getter method.
 *
 * @return true if we are in the big text mode
 */
@SimpleProperty(category = PropertyCategory.APPEARANCE,
    description = "When checked, we will use high contrast mode")
public boolean BigDefaultText() { return bigDefaultText; }

/**
 * When checked, all default size text will be increased in size.
 *
 * @param bigDefaultText true if the big text mode is on
 */
@DesignerProperty(editorType = PropertyTypeConstants.PROPERTY_TYPE_BOOLEAN,
    defaultValue = "False")
@SimpleProperty
public void BigDefaultText(boolean bigDefaultText) {

    //this.scrollable = scrollable;
    this.bigDefaultText=bigDefaultText;
    setBigDefaultTextRecursive( container: this, bigDefaultText);
    recomputeLayout();
}

private static void setBigDefaultTextRecursive(ComponentContainer container, boolean enabled) {
    for (Component child : container.getChildren()) {
        if (child instanceof ComponentContainer) {
            setBigDefaultTextRecursive((ComponentContainer) child, enabled);
        } else if (child instanceof AccessibleComponent) {
            ((AccessibleComponent) child).setLargeFont(enabled);
        }
    }
}
}

```

## A.4 App Inventor Image Component

```
@DesignerProperty(editorType = PropertyTypeConstants.PROPERTY_TYPE_STRING, defaultValue = "")
@SimpleProperty(description = "A written description of what the image looks like.")
public void AlternateText(String description) { view.setContentDescription(description); }
```



# Appendix B: Workshop Materials

## B.1 Pre-survey

## Default Question Block

Welcome to the mandatory survey for "Visual Accessibility for MIT App Inventor"! Please input your name. (We are just using this to make sure that you have completed the survey, your name will not be used for any other reason.)

How much experience do you have with MIT App Inventor?

- None at all
- A little
- A moderate amount
- A lot

Please follow the instructions in the set-up document your teacher has to sign in to App Inventor, download the App Inventor Companion app, and upload the CoinFlipGame. Once you have followed these instructions, please play with the CoinFlipGame for a few minutes until you understand

what it does. If you're confused, be sure press on the "Press for Instructions" button on the app.

Please mark as many examples as possible of this app NOT being visually accessible.



Go on to the next page by pressing the arrow key in the lower right.

Hello! Thanks for taking the time to fill out this survey. We are collecting your response as part of a research study to try and create more fun and easy to understand learning materials. This survey is anonymous, which means that we will not be taking your name or linking the responses back to you. This survey is not graded in any way and is not an assessment of skill.

This survey will ask you some questions related to visual accessibility.

**Accessibility** is the practice of making products and services usable by as many people as possible.

**Visual accessibility** focuses on making tools more viewable for people with low vision.

What exposures have you had to visually accessible design in the past? (Please check all that apply)

- I have never heard of it.
- I have heard it mentioned a few times.
- I know some aspects of visually accessible design.
- I have done work or research about visually accessible design.

What are the top features (list up to three) that come to your mind when you think of a visually accessible mobile app?

What would you say it means for a person to have low vision?

Please indicate how much you agree or disagree with each statement.

	Strongly disagree	Somewhat disagree	Neither agree nor disagree	Somewhat agree	Strongly agree
I feel comfortable making basic App Inventor apps.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
I feel comfortable making visually accessible App Inventor apps.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

How important are the following aspects in the design of apps? Please rank them in order of importance, with 1 being most important. To rank the listed items please drag and drop each item.

Having a good idea for the app

Have the app be fun to use

Making the app nice to look at

Making the app quick to load

Making the app accessible to use

Thank you for taking this survey! Optionally, do you have any questions about the survey or the upcoming workshop, or anything the organizers should know?

## Block 1

Powered by Qualtrics

## B.2 Main workshop slides



# Visual Accessibility with MIT App Inventor

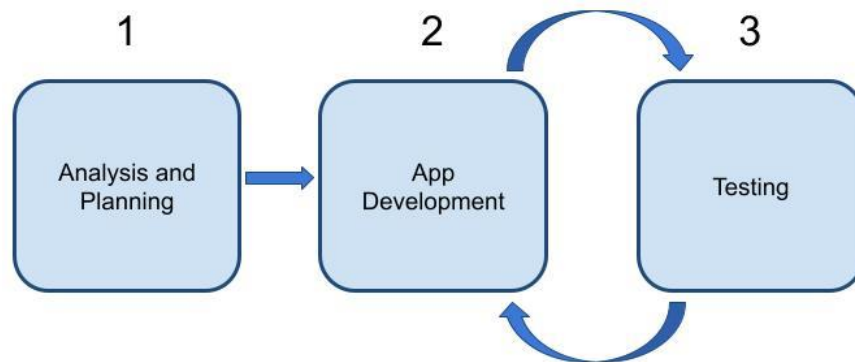
Murielle Dunand, MIT Masters Candidate  
MIT App Inventor Lab

Welcome, please open **simulator.seenow.org**  
on your phone/tablet and press “Launch Simulator”

MIT App Inventor, 1



## Workshop Plan



MIT App Inventor, 2



## What is MIT App Inventor?

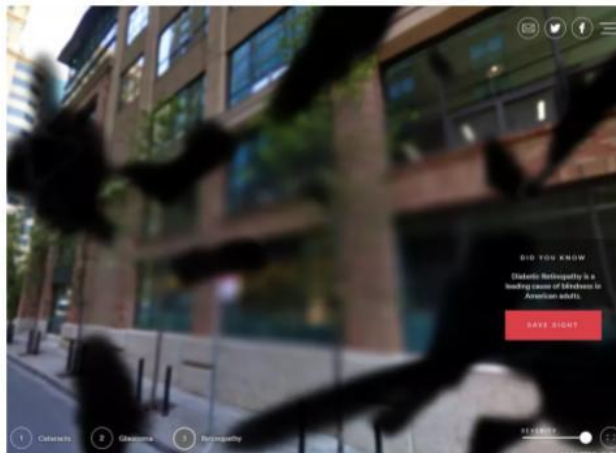


MIT App Inventor, 3



## Let's try it out!

[simulator.seenow.org](http://simulator.seenow.org)



MIT App Inventor, 4





## What is low vision?

- Central vision loss (not being able to see at the center of vision)
- Peripheral vision loss (not being able to see at the edges of vision)
- Night blindness (not being able to see in low light)
- Blurry or hazy vision
- And everything in between!

MIT App Inventor, 5



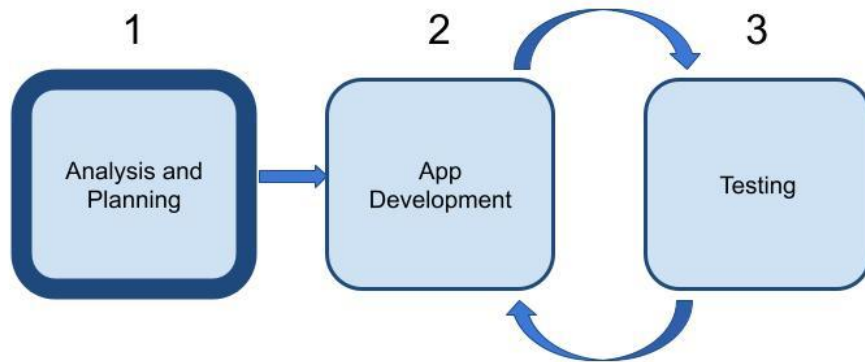
## Accessibility Devices



MIT App Inventor, 6



## App Design



MIT App Inventor, 7



## Analysis and Planning - Personas

*“**Personas** are the single most powerful design tool that we use. They are the foundation for all subsequent goal-directed design. **Personas** allow us to see the scope and nature of the design problem.”*

*— Alan Cooper, software designer, programmer and the “Father of Visual Basic”*

MIT App Inventor, 8



## Persona Analysis

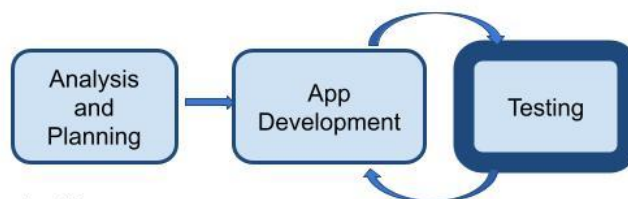
- What type of low vision does your persona have?
- What are the priorities of your persona?
- What parts of mobile apps might they struggle with?
- What steps are they taking to improve their experience?

MIT App Inventor, 9



## Test out a Completed App

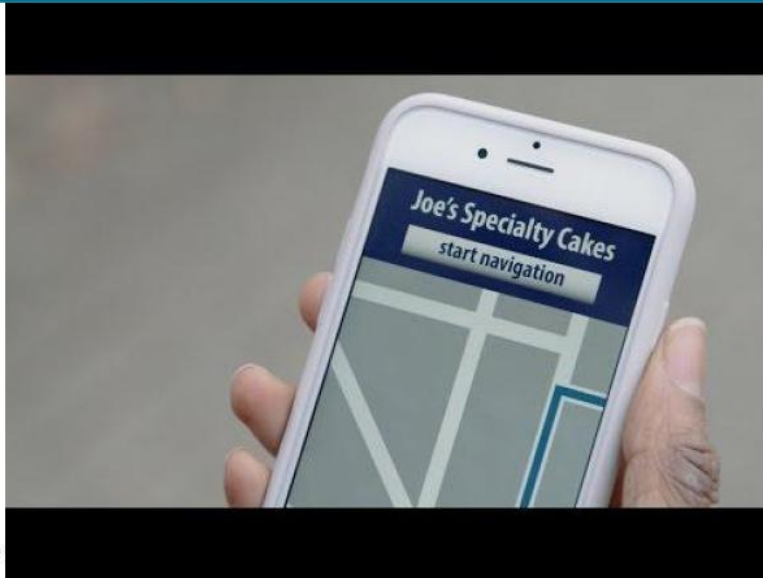
- What parts of the app would your persona enjoy?
- What parts of the app might they struggle with?



MIT App Inventor, 10



## W3C Tips: Contrast



MIT App



## W3C Tips: Text



MIT App



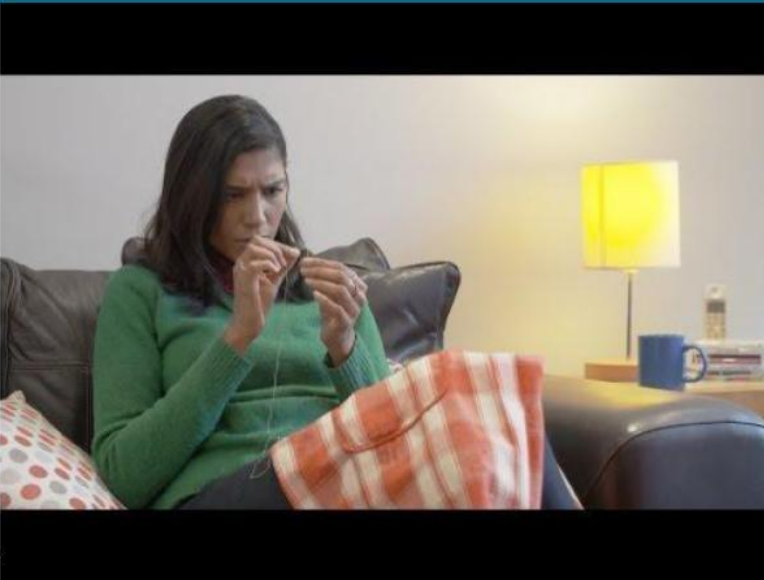
## W3C Tips: Text to Speech



MIT App



## W3C Tips: Sizing



MIT App





## Test out a Completed App

- What parts of the app would your persona enjoy?
- What parts of the app might they struggle with?

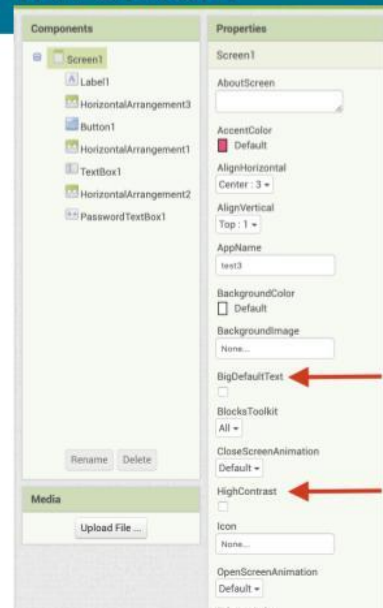


MIT App Inventor, 15



## What has been added to App Inventor?

- Larger default text option
- High Contrast component option
- Alternate Text field to images



MIT App Inventor, 16



## What can you do?

- Keep in mind the persona you were given
- Make sure important information is in high contrast
- Convey information in multiple ways
- Make sure your layout is large and understandable

MIT App Inventor, 17



## Improve on the coin-toss app!

- What components did you use?
- How do the features reflect the needs of your persona?
- What else would you include, given the time?

MIT App Inventor, 18

## B.3 Personas

### Persona #1

Name: Elijah

Gender: Male

Age: 68

Profession: University Professor

Country: South Africa

#### Biography:

Like many people his age, Elijah developed mild cataracts and now has low vision related to visual clarity. When he tries to read small text on a screen, it is generally too blurry for him to read. This gives him some trouble in his work, because most of the classes he teaches has his students return their homework in electronic form. To be able to grade his students' work more easily, Elijah uses a screen magnifier to make the text he needs to read bigger. However, he is concerned that as his cataracts worsen, he may need more than just a text magnifier to be able to read small text.

#### Goals and Challenges:

- Design good classes for his students
- Grade his students' work more easily
- Read academic papers quickly



## Persona #2

Name: Isabella

Gender: Female

Age: 21

Profession: College Student

Country: Spain

### Biography:

Isabella is a college student, currently studying biology in Spain. She started getting migraines in high school which have become chronic. One of the side effects is strong light sensitivity. This makes using fluorescent screens very straining for long periods of time, but her lab work often requires her to analyze data on screens. To lessen the effects of exposure to screen light, Isabella wears tinted glasses and has enabled dark mode on all of her devices. Sometimes, dark mode is incompatible with the apps that she uses, which gives her extra trouble and concern that the app may trigger a migraine, so she avoids sites with lots of bright colors regardless.

### Goals and Challenges:

- Work in a busy lab with her peers
- Read large spreadsheets of biological data
- Graduate from her school with her classmates

## Persona #3

Name: Ming

Gender: Female

Age: 43

Profession: Accountant

Country: China

#### Biography:

Ming is an accountant in Shanghai, China, and works with Google Docs and spreadsheets of data. She was diagnosed with type 2 diabetes in her 30s. One of the side effects of her diabetes is diabetic retinopathy. This means that her high blood sugar levels caused damage to her eyes. Her vision is a bit blurry and Ming has trouble distinguishing contrast in text and images. When she navigates websites, she sometimes misses important things like the "login" or "submit" buttons, or even pop-up messages because they don't stand out visually to her. She needs items to be marked with more contrast or a distinguishing visual border. Ming uses high contrast mode on her computer and mobile device, but this is not always enough for her. When she cannot read the text at all, she uses a screen reader to read the text to her aloud.

#### Goals and Challenges:

- Find an online calculator that is easy for her to use
- Read client reports more easily
- Advance in her job and prove herself as an accountant

## Persona #4

Name: Sarah

Gender: Female

Age: 70

Profession: Retired

Country: Australia

### Biography:

Sarah is a 70 year old grandmother who has recently developed severe glaucoma. She has retired in Australia but still has a lively social life and loves to give gifts to her grandchildren. Sarah has a limited field of vision, especially in the center of her vision. This is known as central field loss. This does not stop her from spending time with her grandchildren, but when she tries to use a mobile device, she has trouble seeing the center of her screen. She uses a screen reader to read the text and describe the images on her screen, and relies on images having alternate text that can be read by her screen reader. She mostly uses her mobile device to shop online for gifts, and wishes that the images of products were well described and compatible with her screen reader.

### Goals and Challenges:

- Shop online quickly and easily
- Video call her grandchildren often
- Read email advertisements to find deals on gifts

## Persona #5

Name: Rob

Gender: Male

Age: 12

Profession: Student

Country: United States

### Biography:

Rob is a 12 year old middle school student in the United States. Like 1 in 12 boys, Rob was born with colorblindness. Rob has red-green colorblindness, and so he has trouble distinguishing between the two colors. He loves to play soccer and phone games. When he plays on his phone, he used to have trouble distinguishing different parts of the game, especially when green means go and red means stop. His parents adjusted the colors on his phone so that they are more distinguishable for him, but not all apps are compatible with the color scheme that works for him. He hopes that he can keep playing, both alone and with others.

### Goals and Challenges:

- Find games that are compatible with his phone color scheme
- Distinguish between his soccer teammates' jerseys
- Win his next soccer game and make friends

## Persona #6

Name: Advik

Gender: Male

Age: 16

Profession: Student

Country: India

Advik is a 16 year old student from India. He has retinopathy of prematurity, which means that he was born before his eyes had finished developing, and so Advik has patches of vision loss across his vision. He has trouble noticing small details, especially text, which may be obscured by floating dark patches in his vision. To do his homework, he first scans the papers onto his mobile device. After scanning, he reads his homework with a screen magnifier so he can zoom in on specific words and images he has trouble seeing. Advik really does not mind spending the extra time to adjust the size to read, but it is stressful for him when teachers do not share homework in time for him to be able to read and keep up with his classmates.

### Goals and Challenges:

- Ask his teachers to assign his homework digitally in advance
- Build up experiences to apply for college
- Use an e-reader to read his favorite novels

## B.4 Post-survey

4/30/2021

Qualtrics Survey Software

### Default Question Block

Hello! Thanks for taking the time to fill out this survey. We are collecting your response as part of a research study to try and create more fun and easy to understand learning materials. This survey is anonymous, which means that we will not be taking your name or linking the responses back to you. This survey is not graded in any way and is not an assessment of skill.

This survey will ask you some questions related to visual accessibility.

**Accessibility** is the practice of making products and services usable by as many people as possible.

**Visual accessibility** focuses on making tools more viewable for people with low vision.

What is your name? (This is just to make sure you completed the pre-survey, we will not save this information)

During the workshop, you got to critique the CoinFlipGame app from the point of view of your personas. What were some aspects that were NOT visually accessible?

During the workshop, you got to make the CoinFlipGame more visually accessible. What did you change to achieve that? If you couldn't figure out how, what were some ideas you had?

What are the top features (list up to three) that come to your mind when you think of a visually accessible mobile app?

What would you say it means for a person to have low vision?

Please indicate how much you agree or disagree with each statement.

		Neither		
Strongly	Somewhat	agree nor	Somewhat	Strongly
disagree	disagree	disagree	agree	agree



	Strongly disagree	Somewhat disagree	Neither agree nor disagree	Somewhat agree	Strongly agree
I feel comfortable making basic App Inventor apps.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
I feel comfortable making visually accessible App Inventor apps.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

How important are the following aspects in the design of apps? Please rank them in order of importance, with 1 being most important. To rank the listed items please drag and drop each item.

Having a good idea for the app

Have the app be fun to use

Making the app nice to look at

Making the app quick to load

Making the app accessible to use

What were 2-3 things you learned over the course of this workshop?

A large, empty rectangular text input box with a thin grey border and a small diagonal slash icon in the bottom right corner.

Did your views on visually accessible design change because of this workshop? If so, how?

A large, empty rectangular text input box with a thin grey border and a small diagonal slash icon in the bottom right corner.

Thank you for taking this survey! Optionally, do you have any questions about the survey or the upcoming workshop, or anything the organizers should know?

A large, empty rectangular text input box with a thin grey border and a small diagonal slash icon in the bottom right corner.