

I Have a Dream, a Two-Part Introductory Tutorial

Most people have never even dreamed of building an app; software is this mysterious world of 0s and 1s and computer nerds. The goal of this lesson is to show people that they can create in this medium, that they can program software and contribute in today's highly digitized society.

This tutorial will get you building an app in minutes! You don't need to install anything on your computer-- App Inventor runs in the cloud! You will build the app on your computer and test it on your Android device. If you don't have a device, you can even test on an emulator. This tutorial is meant as a starter app. You will build the "I Have a Dream" app in two parts.

Part I: the app will have a single picture of MLK that, when clicked, plays a clip from the famous speech given to 250,000 on the steps of the Lincoln Memorial in 1963.

Part ii: You will add an image of Malcolm X and one of his speeches, turning the app into a soundboard with these great leader's perspectives on the civil rights movement. You'll learn about conditional "if" blocks in this second part.

The two parts of the tutorial should take you less than an hour. When you complete the two parts, you'll be ready to build soundboard apps on any topic.

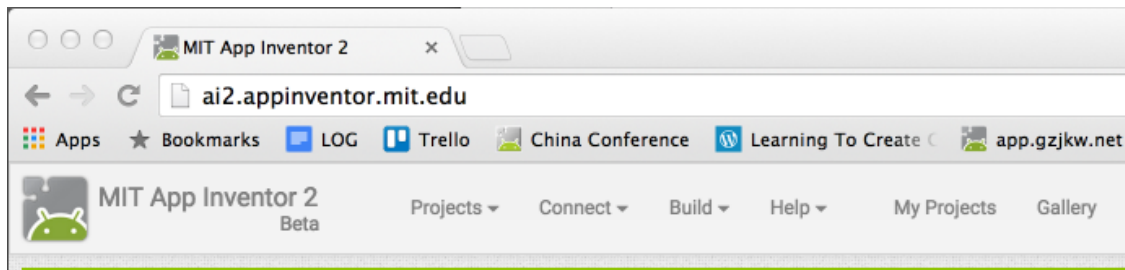
[Video version of this tutorial](#)

Check out the app (part 1)



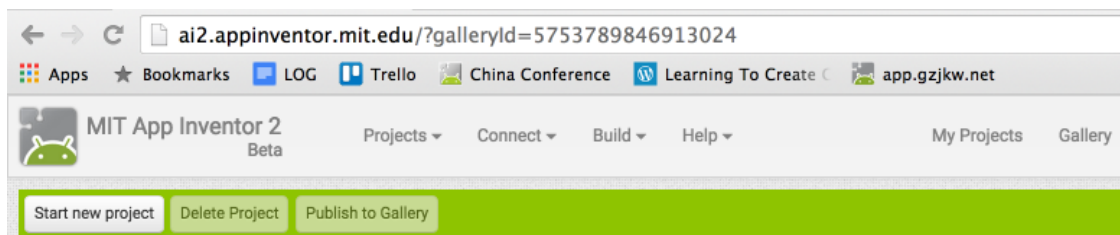
This is the way your I Have a Dream app will appear. When the user clicks on the images, the app will play a speech by Martin Luther King.

Begin by opening a browser window to App Inventor: ai2.appinventor.mit.edu.



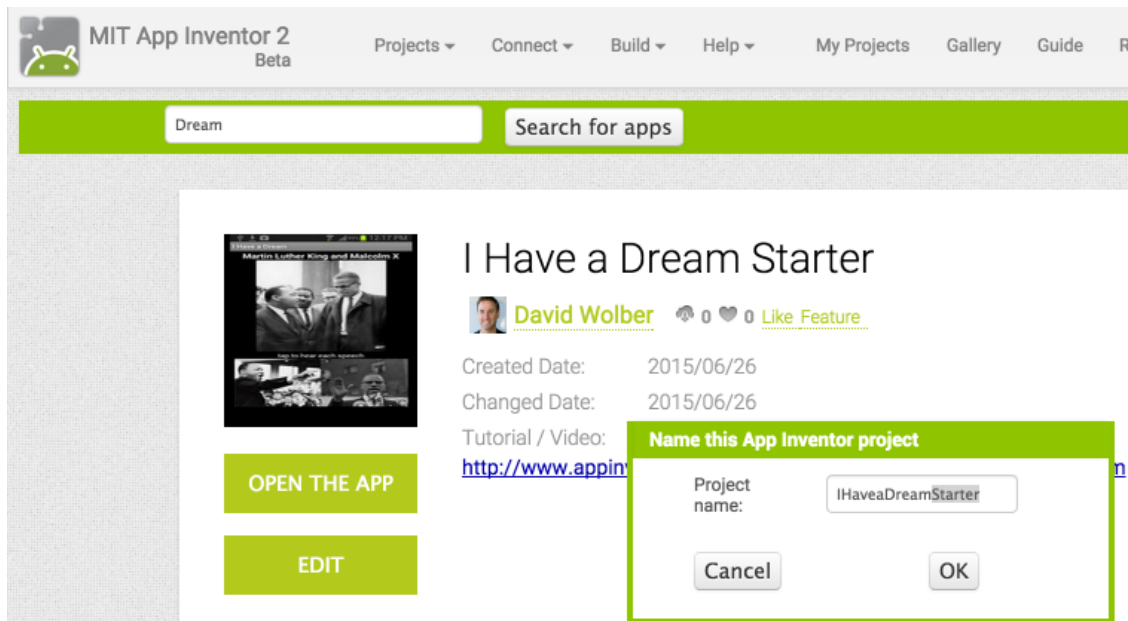
If you have WiFi and an Android device, you can build apps with App Inventor completely "in the cloud", i.e., without downloading any software to your computer. You login to the system with a Google account. If you were going to start an app from scratch, you would begin from here. But we'll use a "starter" app (see below).

For this tutorial, you'll begin with a "starter app" from the App Inventor Gallery



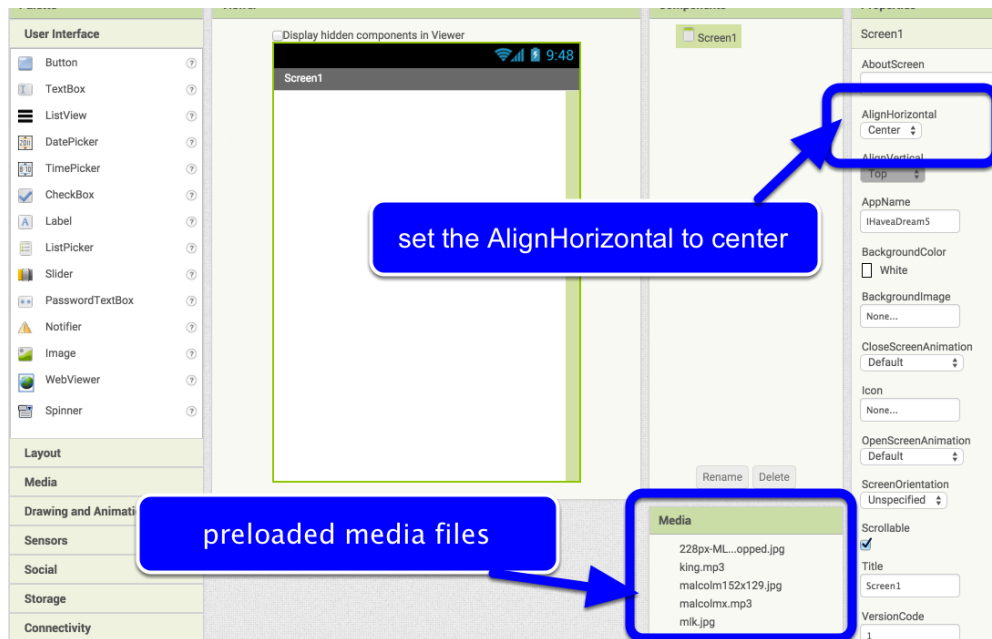
App Inventor has a gallery for sharing apps. Click this link <http://ai2.appinventor.mit.edu/?galleryId=5753789846913024> and App Inventor will open to the I Have a Dream Starter app. The starter app has no code-- it just contains the image files and sound clips you need for the app. Its provided so you can learn the coding without spending time finding media files on the Internet. Later, you can build your own soundboard app with your own images and sound clips.

Open the starter app



Click "Open the App" to open the starter app. You'll be prompted to name your app. Name it something like "IHaveADream", e.g., remove the "Starter" from the name. Note that app names cannot have spaces or other punctuation.

Check out the Designer and align the screen

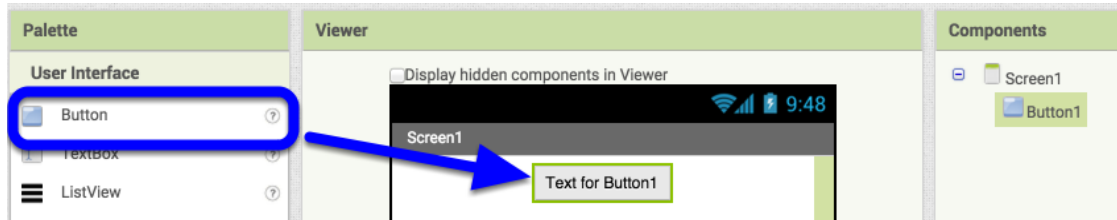


You'll specify how the app should look, and all the components you'll use, in the App Inventor Designer, which is shown above. As you can see, the screen starts out as blank, but the starter app has some media files preloaded to help you get going. Note that you can add your own media files

as well, you're just not doing that in this tutorial.

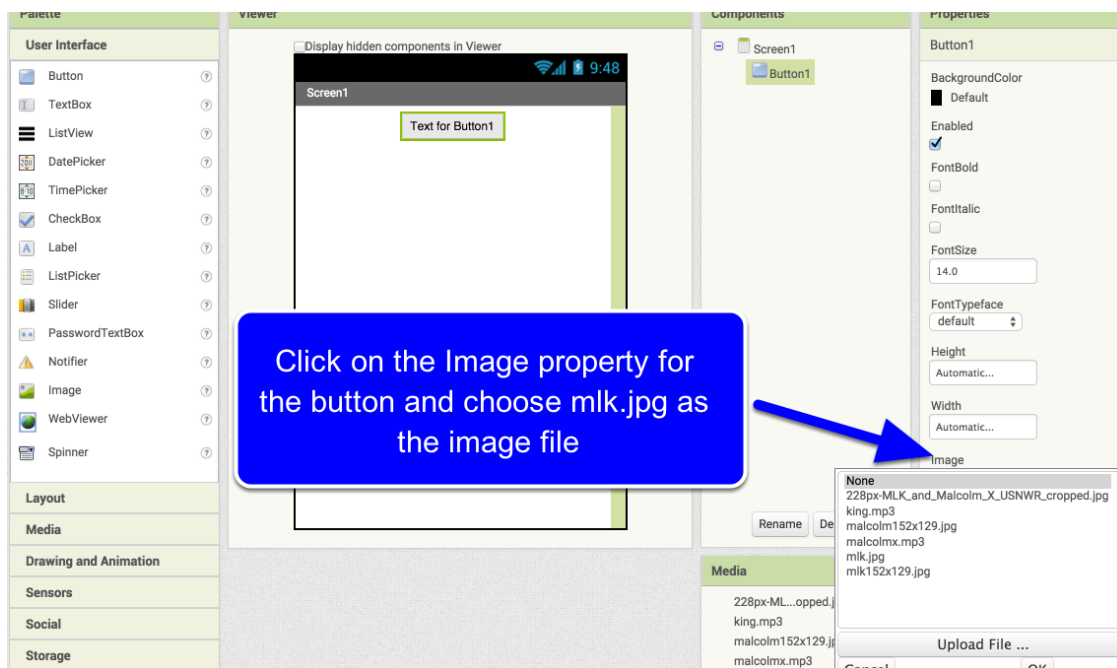
To get things started, set the Screen's AlignHorizontal property to "Center". When we add the pictures of the app, they'll automatically be centered within the screen.

Drag a button onto the screen

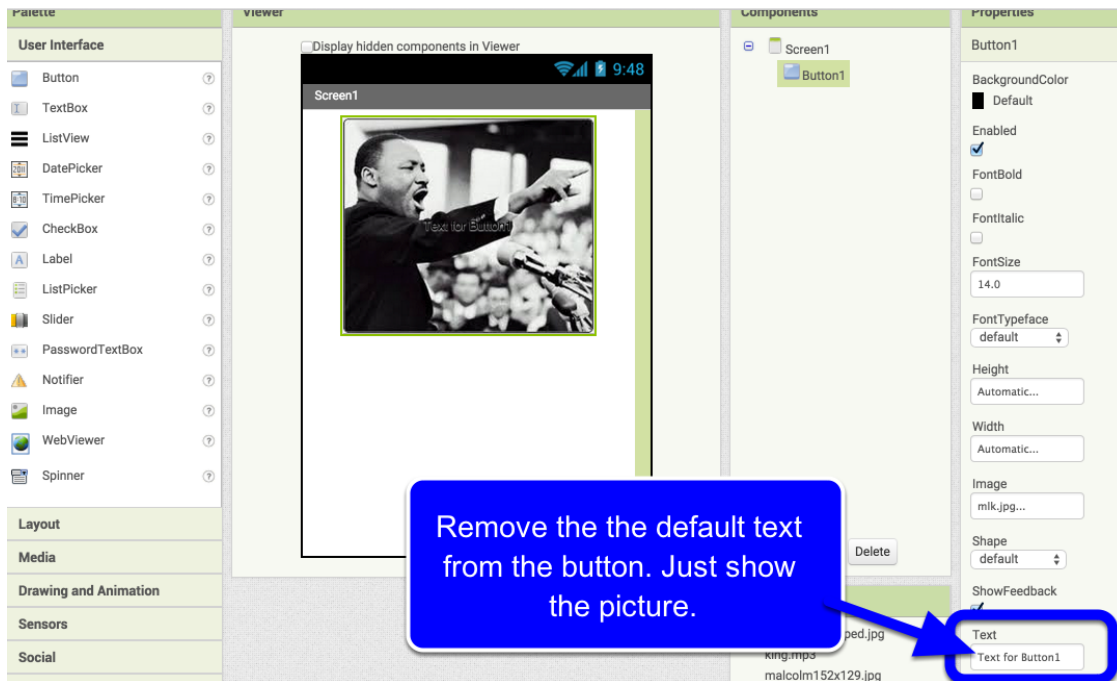


Next, add a Button to the screen. Click down on the Button in the palette, then drag one out and into the screen. The button will appear with some default text. In the next step, you'll change it so it shows a picture.

Set the button's image property so that it shows MLK

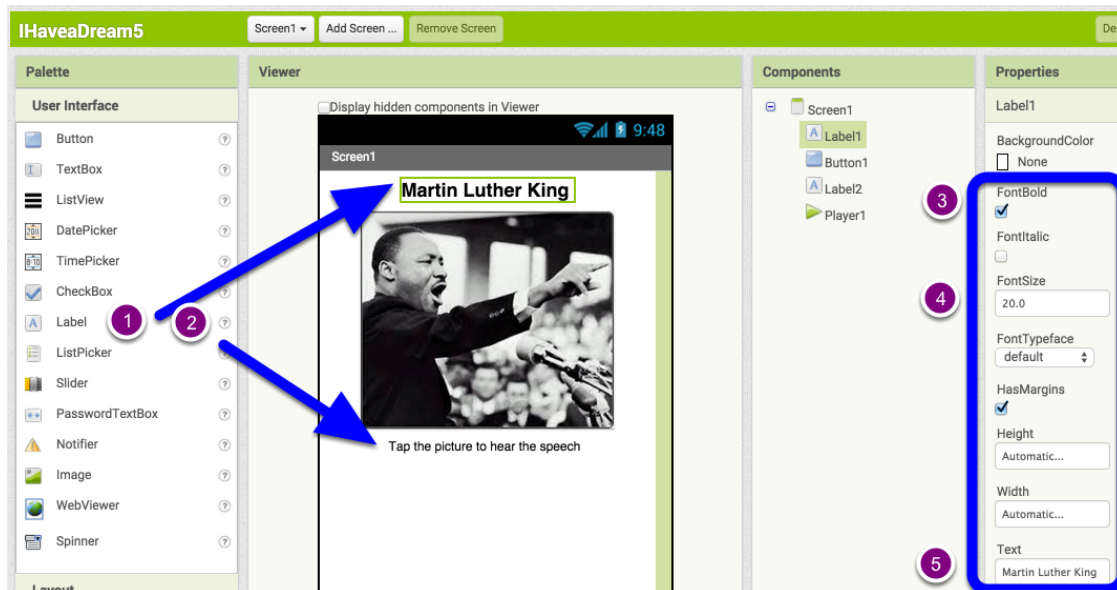


Remove the default text from the button



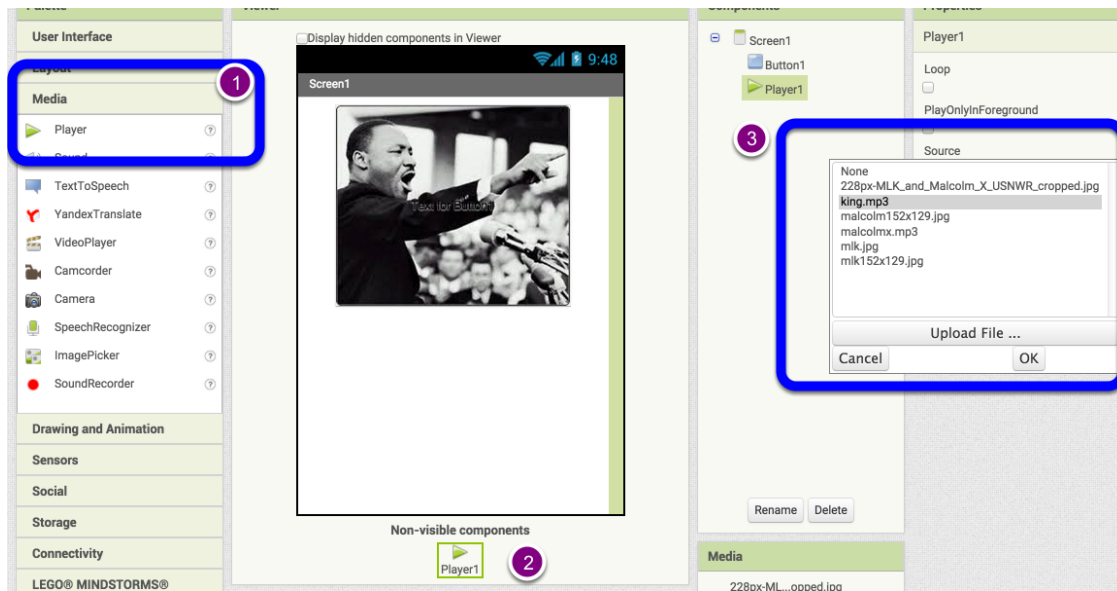
When you set the Image property for the button, the picture appears but the default text for the button is still there. Click on the Text property and set it to blank (backspace to beginning).

Add two labels above and below the button.



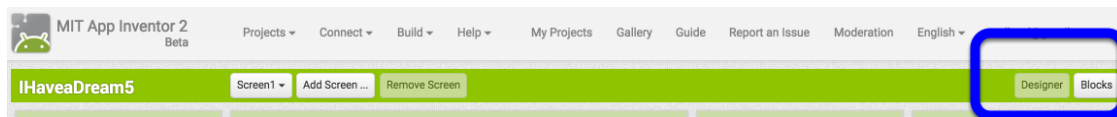
Drag two labels onto the screen, one above the button and one below. Set the Text property of the labels as shown. For the top label, check the FontBold property and change the FontSize to 20.0.

Add a Player component for playing the speech



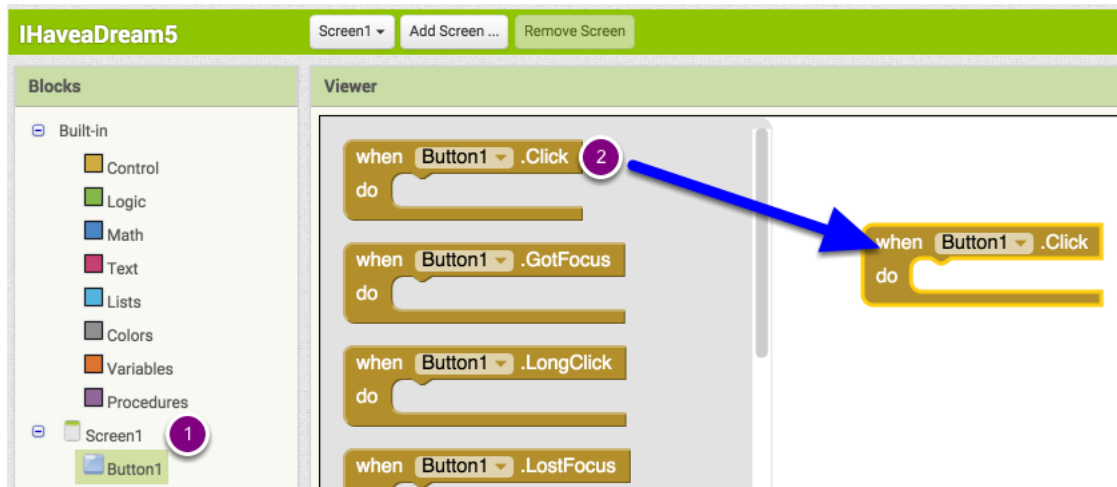
Open the Media drawer (1) and drag a Player component into the app (2). A Player component knows how to play sound clips and is considered a "non-visible" component because it doesn't show up in the user interface of the app. After dragging in the Player component, set its Source property to king.mpg (the sound clip of MLK's speech).

Open the Blocks Editor to program the interactive behavior



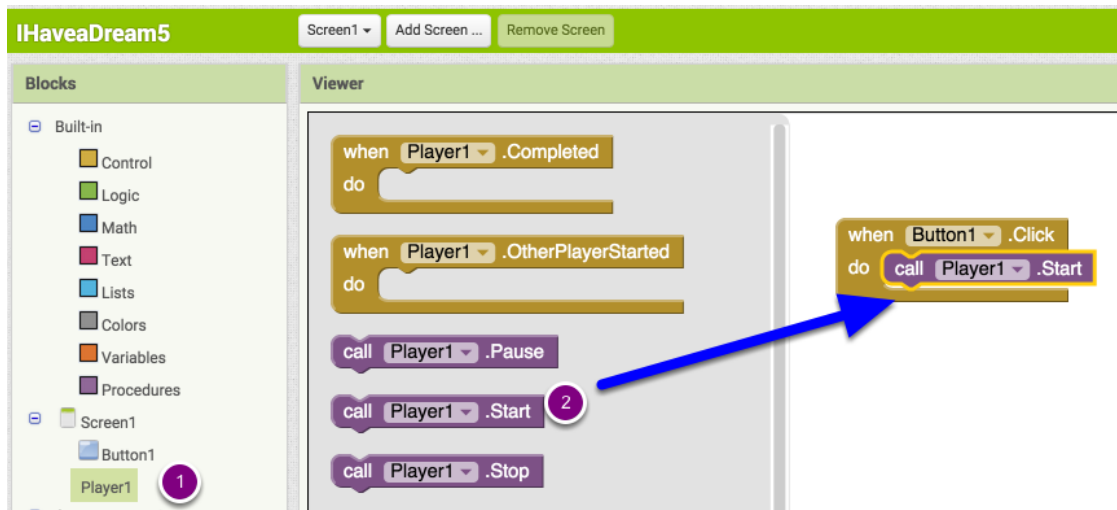
Thus far, you have completed the design of part 1 of the app, but you still need to code the interactive behavior of the app. In the right top-corner, you can switch between the Design view and the Blocks view. Click on "Blocks" and open the empty Blocks view.

Drag out a Button.Click event handler



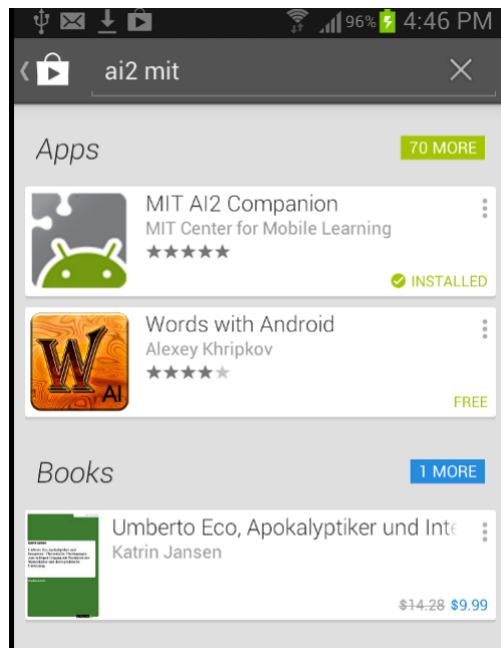
In the Blocks editor, click on Button1 (1) and then drag out a Button1.Click event handler (2). You are preparing to show that when the button is clicked, the speech will be played.

Place a Player1.Start block within the Button1.Click event handler.



Click on the Player1 drawer (1) and then drag out a "call Player1.Start" block and place it in the event handler. You are specifying that when the button is clicked, the speech should be played. Remember, the Player1.Source property is set to the king.mpg file, so that will play.

Install the MIT AI2 Companion on your phone.



To test your app, you first need to install an app on your Android device (not your computer). On your device, open the Android Play Store and search for the MIT AI2 companion. Then install it.

Connect your device for live testing

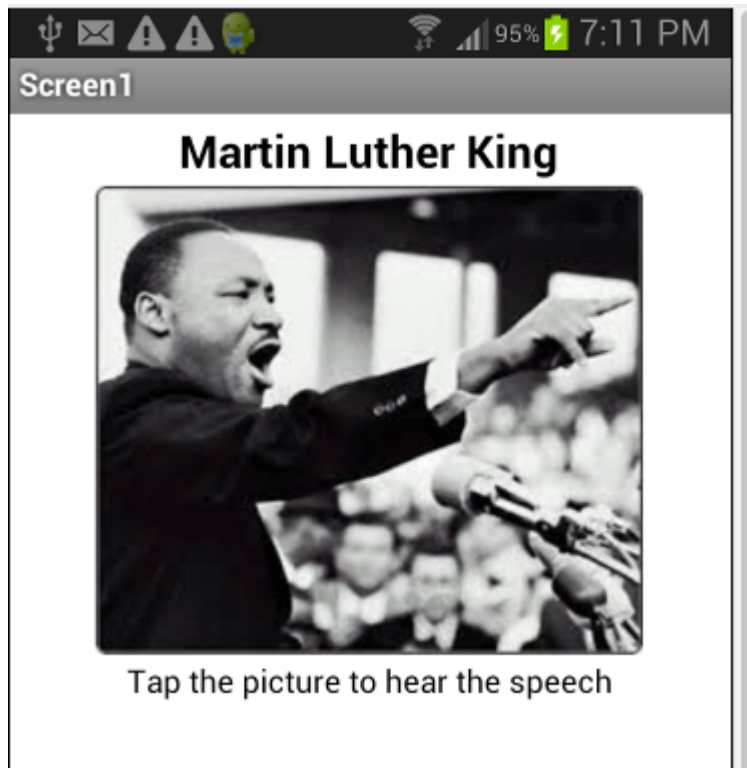


When you start the Companion on your phone, it will look like (1). Back in App Inventor, choose "Connect" (2) then "AI Companion. This will cause a QR code to appear (3) You can then scan the QR code (4) with your phone to see your app live.

NOTE: for live testing to work, both your computer and phone/tablet must be connected to the same WiFi station. Using WiFi is the easiest way to connect, but if you're at school/work you may have firewall issues. If so, you can

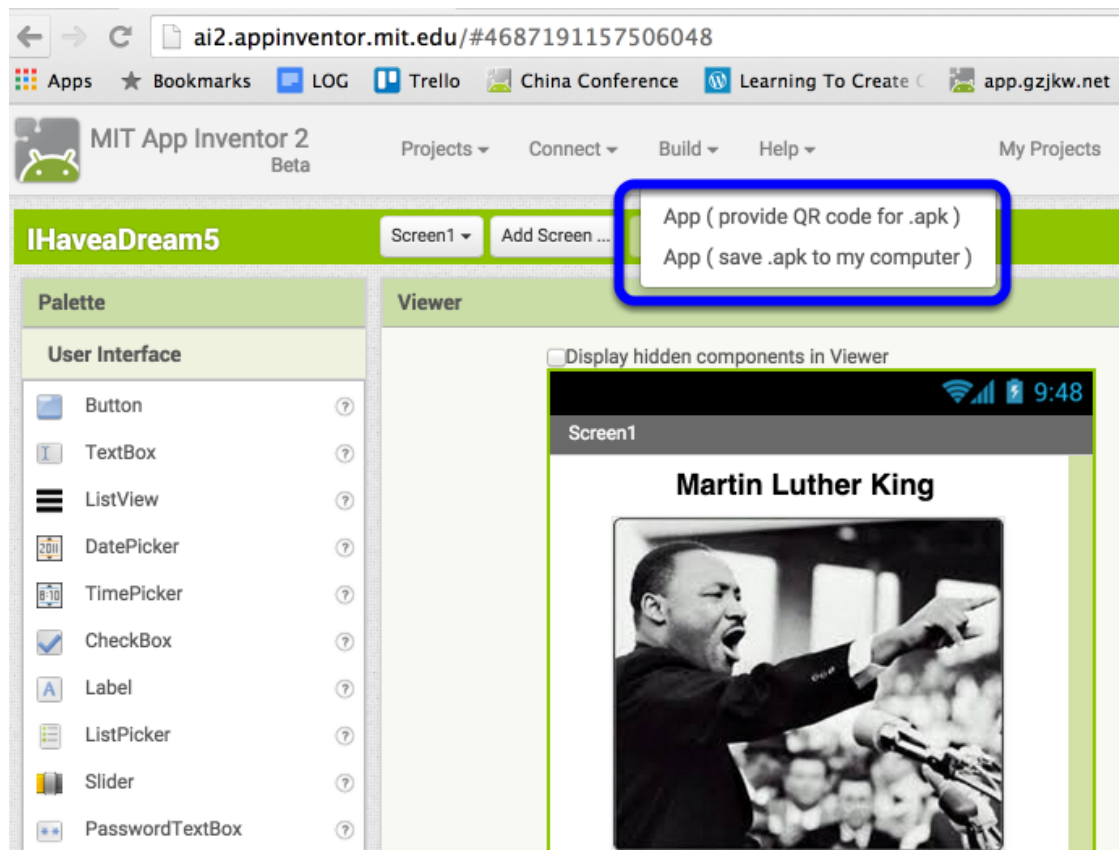
connect using a USB cable. If you don't have an Android device, you can connect your app to an emulator that runs on your computer. For any setup issues, please see <http://appinventor.mit.edu/explore/ai2/setup.html>

Test your app!



When you scan the QR code, your app should appear on your device. If you've programmed the app correctly, you should now be able to click on the picture and hear the speech.

Download the app to your computer or device



Your app is only working within the testing app, the Companion. If you disconnected from WiFi, the app wouldn't be on your phone, and so far you can't send it to a friend. So go back to the Designer and choose "Build | App (save to my computer)" This will download a ".apk" file, an executable Android app. Once you download it you can email it to yourself or a friend. Then you and your friends can open the email on their Android phone and install the attachment.

You can also choose the top choice and have a QR code appear. If you scan the QR code on your device, it will download the app (the .apk) directly to your device for installation.

Note: on some Android phones, you need to change a security setting to allow "Unknown Sources". This allows the device to install apps from places other than the Play Store.

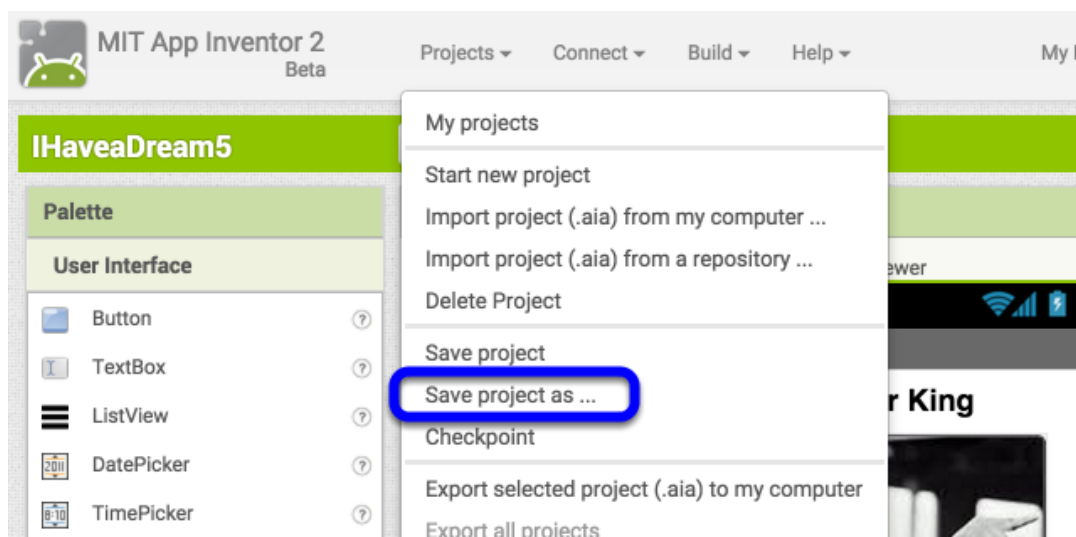
Congratulations! You've built your first app!

Part II. Add Malcolm X and his speech



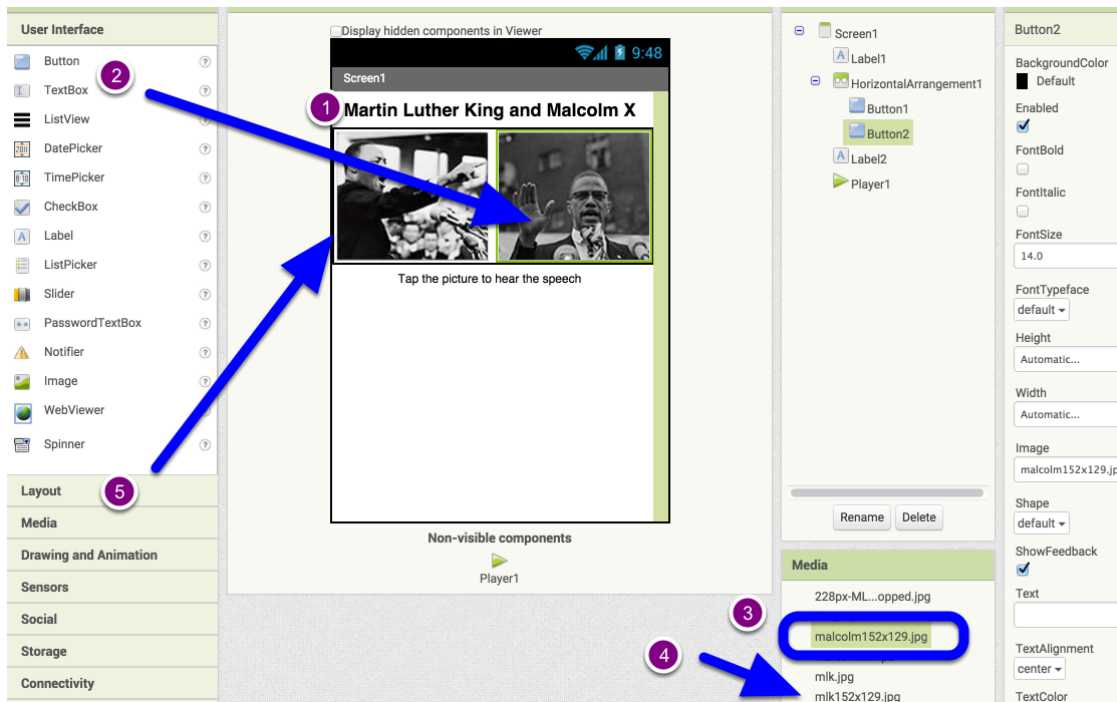
Now that you have your feet wet, let's make the app more fun. MLK and Malcolm X were Civil Rights leaders of great contrast. Let's build an app that also plays an app by Malcolm X in order to show this contrast. You'll need to modify the UI, then the behavior. Behavior-wise, things will become more complicated as you'll need to make sure the speeches don't overlap.

SaveAs your project to create a new project



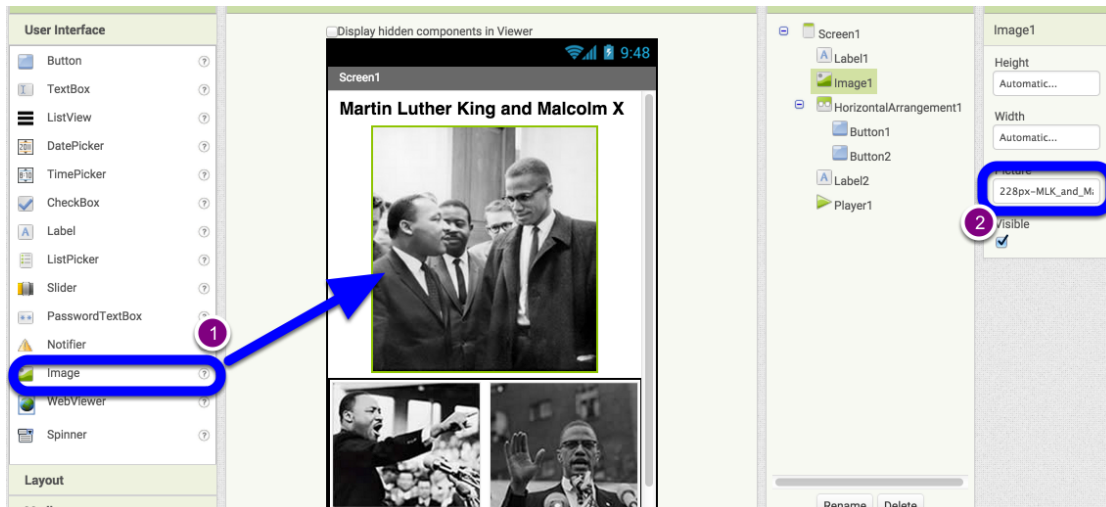
Choose Project | SaveAs and name your new app something else (e.g., DreamWithNoName).

Choose a smaller image for MLK and add an image of Malcolm X.



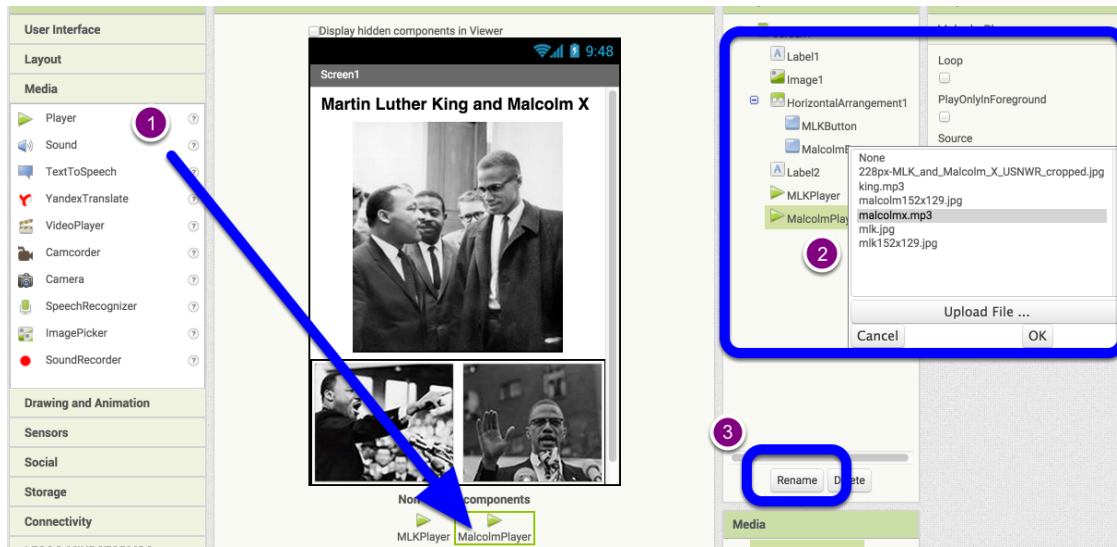
In the designer, modify the user interface. First, change the text of the top label so it mentions Malcolm X (1). Then drag out a second button for Malcolm X (2) and set its picture property as shown (3). Then change the Image for MLK's button to the smaller image (4). Finally, drag out a HorizontalArrangement from the Layout drawer (5), and drag the two buttons into it so that they appear side-by-side as shown.

Add the image of both leaders



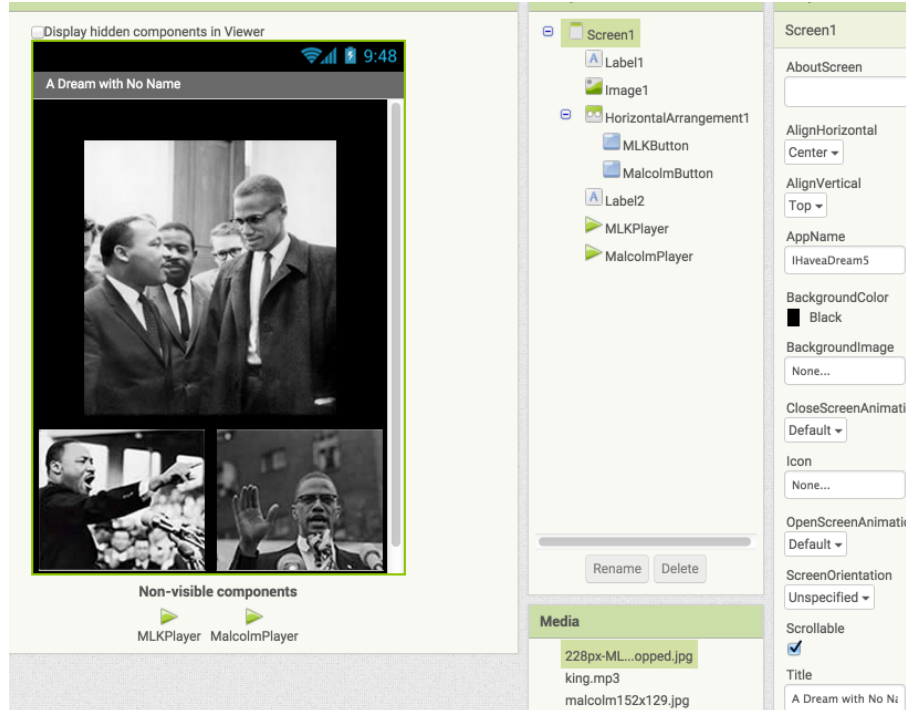
Add an Image component (1) then set its Picture property to the picture of both leaders. Note that the top component is an Image not a Button because nothing should happen when the user clicks it.

Add a second player and rename components



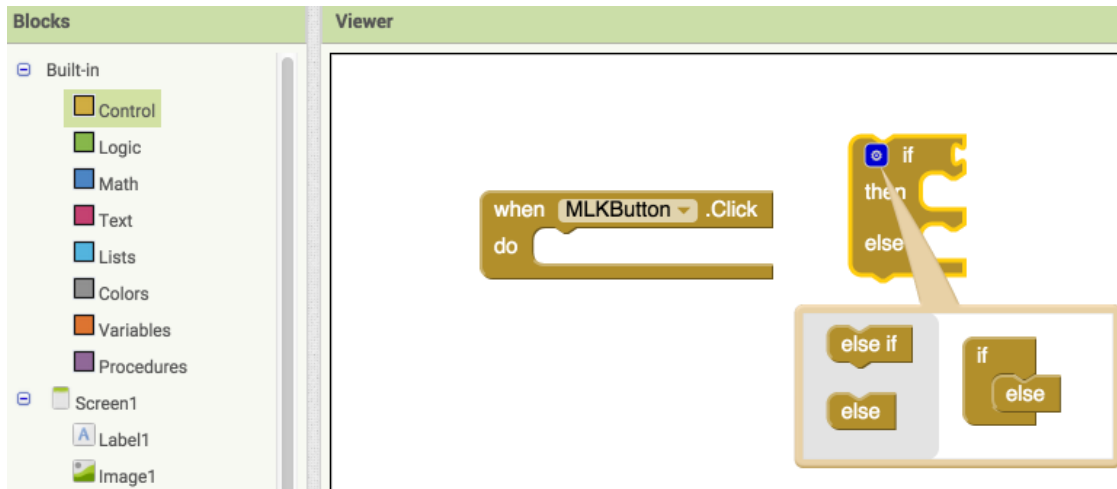
Drag out another Player component (1) and set its Source property to Malcolm's speech (2). Then rename the Button and Player components (3) so we can distinguish them easily in the blocks editor. A rule-of-thumb is to give a descriptive name with a suffix which is the component type, e.g., MalcolmPlayer.

Add a Screen Title and Background Color



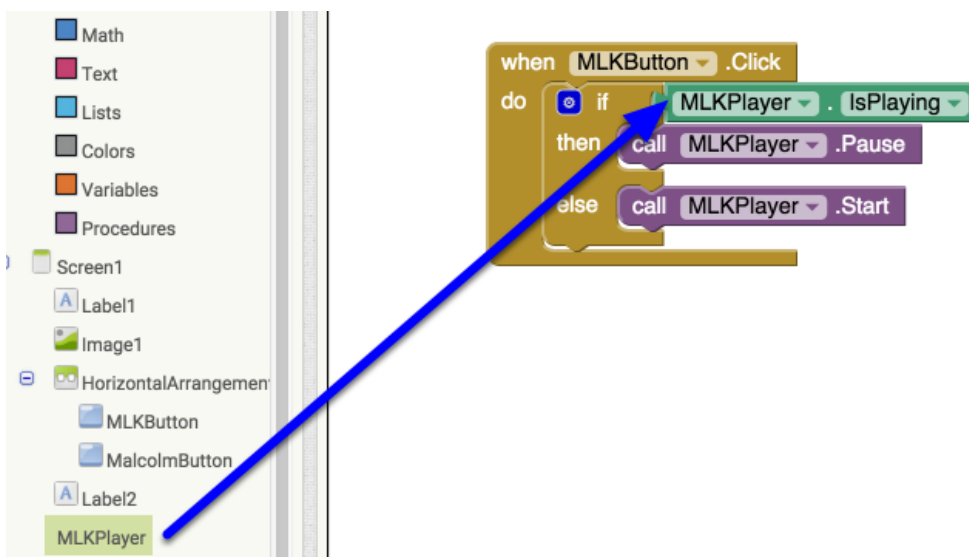
To complete the UI, set the BackgroundColor to black and set the screen's Title property. You can also set the Screen's Icon property to an image that you want to show on a device representing the app. Now you're ready to program the behavior.

Code it so you start/pause the MLK speech



Open the Blocks editor. Note that the Button1.Click event and other blocks have been renamed based on your renames in the designer. First, focus on the MLK button. The behavior we want is for the first click to start the speech, the next one pause it, next one start it, and so on. So you don't always want the same thing to happen when the button is clicked. To program this, you use an if-else block. If-else allows the app to ask questions, such as, "is the speech already playing?". To code this, drag out an if-block from the Control drawer, then click on the blue "modifier" icon. The modifier allows you to add branches. In this case, there are two possibilities, start or pause, so drag an "else" within the picture to its right so that there are two branches as shown.

Code it so the if-else starts/pauses MLK's speech.



Place the if-else block in the event handler. Then drag out an MLKPlayer.IsPlaying property block from the MLKPlayer drawer. This block is true if the speech is playing, false if not. If it is playing, you want to pause the speech, so drag out this block from MLKPlayer. If it is not playing, the "else"

branch will be taken, so `MLKPlayer.Start` is called.

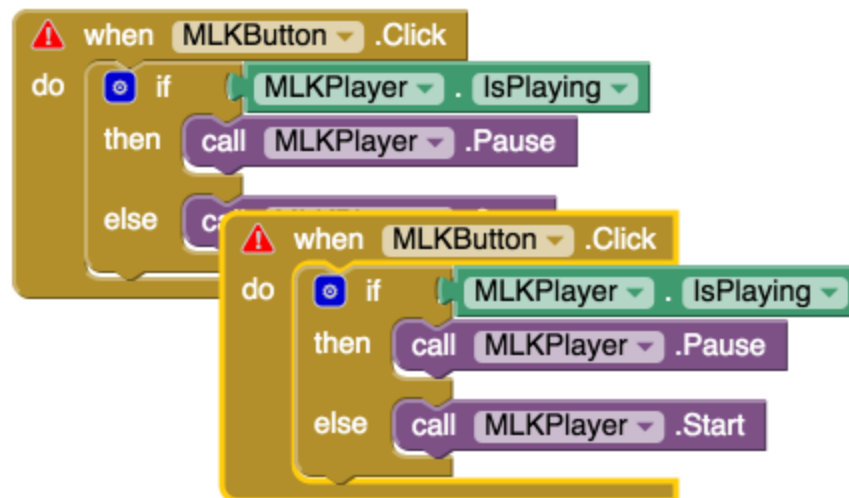
In this code, you're seeing two new things, both fundamental to software. First is the conditional (if-else) block. Conditional blocks provide the mechanism for the app to ask questions, the basis of artificial intelligence. The second new thing is you are checking a component property in the blocks. You set a few properties in the designer (e.g., the `Screen.Title`). Here you are checking the `Player's IsPlaying` property in the blocks.

For more on these fundamental topics, see these notes on [conditionals](#) and [component properties](#).

Test this new behavior

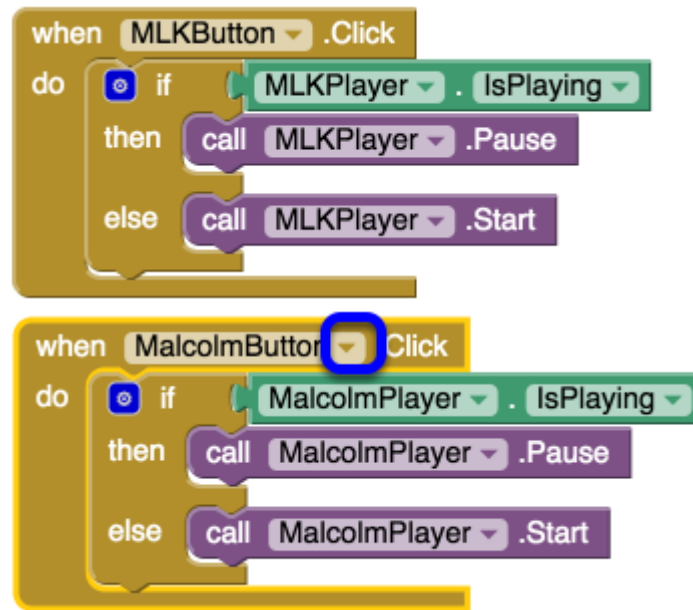
Your device should still be connected. Test this updated behavior to see if the speech starts and pauses.

Copy-Paste the MLKButton event handler



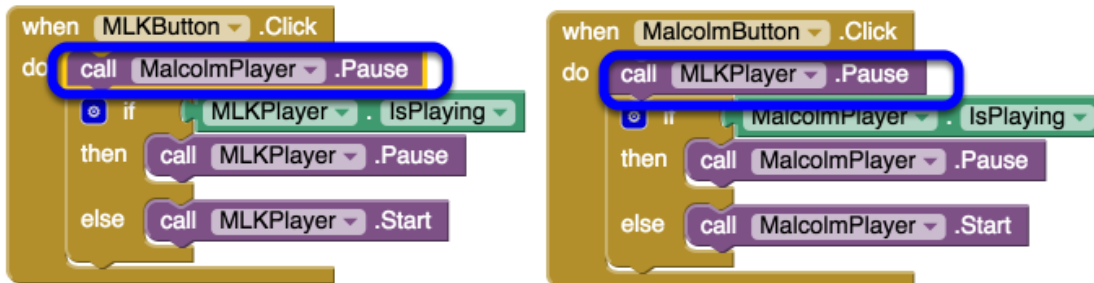
You want the Malcolm button to behave similarly to the MLK button, so copy-paste the event handler. Select the `MLKButton.Click` event handler and copy/paste it using `command-c` and `command-v` on Mac (`ctrl-c` and `v` on Windows). A red error triangle will appear because you can't have two handlers for the same event. Next, you'll fix this.

Change the blocks to refer to MalcolmButton and MalcolmPlayer



Using the upside down triangle widget circled above, switch the copied blocks to refer to MalcolmButton and MalcolmPlayer.

Program it so that you pause the "other" speech



You just need to pause Malcolm when MLK speaks, and vice-a-versa. The final blocks for the app are shown above with the additional blocks circled. Test the app again (you should still be connected). Does it work how you'd expect?

This app is simple, but indicative of the architecture of most apps. This one consists of two event handlers. Each event handler consists of an event (a click in this case) and a response which is a sequence of blocks (they're executed in order). Some blocks in the response are only conditionally executed.

So you've not only created an app, you've learned some programming lingo. Next time your hanging out with your friends, smack talk a bit about "event handlers" and "if-else conditionals".