|  |  |
| --- | --- |
| This is the "Hello World" of App Inventor, the first app most people build. You'll learn that app building is a creative process. You will write code that plays Martin Luther King’s “I have a dream” speech when the user touches the phone’s screen.      **Objectives:** In this lesson you will:   * follow an instructor-led walkthrough to create the *I Have a Dream* app on a mobile device; * learn how to use *if-else selection* blocks to improve the app's behavior. | UIShot.PNG    ***[Click to watch Preview Video](https://www.youtube.com/watch?v=qvimSRfiOKk)*** |

# 

# 

# Getting Ready

1. Open [App Inventor 2](http://ai2.appinventor.mit.edu/).
2. Go to *Project* and select the I Have A Dream app that you created in the first tutorial.

I Have a Dream Tutorial Part 2

## The I Have a Dream User Interface

**User Interface**

## UIShot.PNG

We are going to update our user interface to include some extra components to improve our app. Our new user interface (UI) for our I Have a Dream app will consist of eight *Components:* two Buttons, two Players, two Labels, an Image, and a HorizontalArrangement component. Remember that this is done in the Designer View.

Your Viewer panel should now look like this when your app is complete.

### 

### 

### 

### Adding a Button Component

1. *Drag and drop* a Button component from the Palette’s User Interface category to the Viewer. It will be named *Button2.*
2. Select *Button2* by clicking on it in the Viewer or the Components panel.
3. Click on its *Image* property and select the “malcolm152x129.jpg” image from the drop-down list.
4. Click on its *Text* property and change it to an empty string by deleting “Text for Button2”.
5. You should also update the button for Martin Luther King. Do this in the same way as before. Select *Button1* in the Components panel and change its *Image* property to “mlk152x129.jpg”

**Organizing the User Interface**

1. *Drag and drop* a HorizontalArrangement component from the Palette’s Layout category to the Viewer
2. *Drag and drop* the Martin Luther King button and Malcolm X button into the HorizontalArrangement component so they are side-by-side.

**Adding an Image Component**

1. *Drag and drop* an Image component from the Palette’s User Interface category to the Viewer. It will be named *Image1.*
2. Select *Image1* by clicking on it in the Viewer or the Components panel.
3. Click on its *Image* property and select the “228px-MLK\_and\_Malcolm\_X\_USNWR\_cropped.jpg” image from the drop-down list.

### Adding a Player Component

1. Click on the Palette’s *Media* category to open it.
2. Drag and drop a *Player* component from the Media category to the Viewer. It will be named *Player2* and it will appear at the bottom of the Viewer under Screen1 as a *non-visible component.*
3. Select *Player2* by clicking on it in the Viewer or the Components panel.
4. Click on its *Source* property and select the “malcolmx.mp3” sound file from the drop-down list.

**Renaming the Components**

We want to rename our components to make coding our app easier. Component names should include the component type and some sort of identifier.

To rename a component

1. Select *Button1* in the Viewer or the Components panel
2. Click on *Rename*
3. In the *New Name* text box, type the desired name, and click OK
   1. Rename the *Button1* component to *MLKButton*
   2. Rename the *Button2* component to *MalcolmButton*
   3. Rename the *Player1* component to *MLKPlayer*
   4. Rename the *Player2* component to *MalcolmPlayer*

We also want to change the Label’s *Text* property to reflect the new changes we are making to the app

1. Select *Label1*
   1. Change the *Text* property from “Martin Luther King” to “Martin Luther King and Malcolm X”
   2. Put a check in the box under the *FontBold* property
   3. Change the *FontSize* property to 18
2. Select *Label2*
   1. Change the *Text* property from “Tap to hear the speech” to “Tap to hear each speech”

## Coding the App’s Behavior

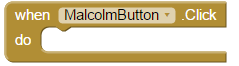
It’s time to get the app to play the speech when we touch its screen. For this we will be using the Blocks editor. So switch now to the *Blocks Editor View.*

### Event Driven Programming

Mobile apps use a style of programming known as *event driven programming.* An app’s behavior depends on how the user programs the app to respond to various events. An example of an *event* would be when the user clicks on a button or when the phone’s location changes or when a text message is received. We’ll write apps that respond to all of these events.

For the I Have a Dream app, there is only *one* event that we care about, the *Button click event.*  This event will be called when Martin Luther King or Malcolm X is clicked.

1. Click on the *MalcolmButton* component in the Toolbox.
2. Drag and drop the *When MalcolmButton Click* block into the Workspace.



This is an example of a *when event block*, which is also called an *event handler block.*

Notice that it has a empty *do slot.*  When *MalcolmButton* is clicked, the app will *do* whatever code we put into this slot.

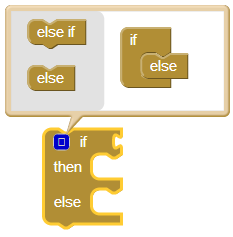
1. Click on *MalcolmPlayer* component in the toolbox
2. Drag and drop the *MalcolmPlayer*.Start block into the *do* slot of the *MalcolmButton*.Click event

For the I Have a Dream app, we want it to play only one speech at a time and pause each speech when the pictures are clicked. To do this, we will need to use *Logic*

**Adding an If block**

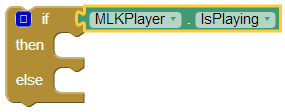
If blocks allow the app to make decisions. In our app, we want to determine whether the sound should be played or paused based on whether something is already playing. To do this, we need to use an *If* block.

1. Open *Controls* in the toolbox
2. Drag and drop the block that reads *If/then* into the workspace.
3. Click on the blue box in the corner of your *If* block (called a *Mutator*)
4. Drag and drop the *else* block from the left into the *if* block on the right



Now that we have our *if* block, we will add the question we want it to ask

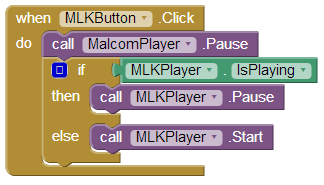
1. Click on the *MLKPlayer* component from the toolbox
2. Drag the *MLKPlayer.IsPlaying* block and drop it into its slot on the *if* block



When the *MLKButton* is clicked, the app now asks if the player is playing. If it is, the code under *then* will be executed. If not, it will execute the *else* code instead. Next we will tell the app what to do in each case.

When *MLKButton* is clicked, we want the program to play *MLKPayer* if it is not playing, pause *MLKPlayer* if it is playing, and always pause *MalcolmPlayer*. To do this:

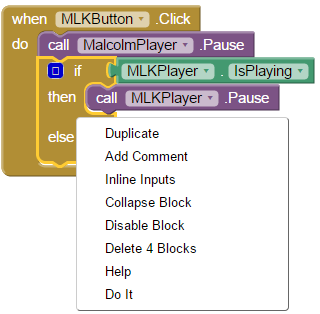
1. Drag and drop the *MLKPlayer.Start* from *MLKButton.Click* to the *else* slot in our new *if* block.
2. Open the *MLKPlayer* component from the toolbox
3. Drag the *MLKPlayer.Pause* from the toolbox and drop it in the *then* slot of our *if* block
4. Drag and drop the *if* block into the *MLKButton.Click* event block
5. Open the *MalcolmPlayer* component from the toolbox
6. Drag the *MalcolmPlayer.Pause* block and insert it on the outside the *if* block



**Duplicating a block of code**

The next step is to code the behavior for *MalcolmButton*. Because it is similar to the behavior of *MLKButton*, we are able to make a shortcut by duplicating the code.

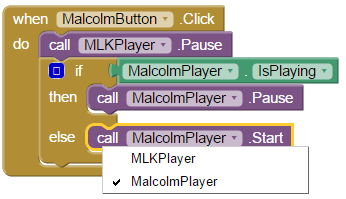
1. Drag the *MalcolmPlayer.Start* block from the *MalcolmButton.Click* event and drop it in the trash icon in the bottom right corner of the workspace. Because we are duplicating the other code, we will not need this extra block.
2. Right click on the *if* block that is in the workspace and select “Duplicate”



1. Drag and drop the copy of our *if* block into the *MalcolmButton.Click* event block
2. Right click on the *MalcolmPlayer.Pause* and select “Duplicate”
3. Drag and drop the copy of *MalcolmPlayer.Pause* into the *MalcolmButton.Click* event, inserting it above the *if* block

We have now copied all the code needed for *MalcolmButton*, but it does not yet apply to the correct player. We want *MalcolmButton.Click* to pause *MLKPlayer*, then either player or pause *MalcolmPlayer*. We can change this easily by using the drop-down lists.

1. Click the small down arrow located in each block to open the drop down list
2. Where the blocks read “MLKPlayer” change to “MalcolmPlayer”
3. Where the blocks read “MalcolmPlayer” change to “MLKPlayer”



To summarize the behavior, whenever the user clicks on *MLKButton*, the app will pause *MalcolmPlayer*, then either pause or play *MLKPlayer*, and vice versa for *MalcolmButton*. This completes the coding of the I Have a Dream app.

# Running the App on the Phone (or Tablet)

Did you remember to set up your device? If yes, continue. If not, follow these [setup instructions](http://appinventor.mit.edu/explore/ai2/setup-device-wifi.html) and come back to continue.

1. Start the the *MIT AI2 Companion* app on your phone or tablet.
2. In App Inventor, click Connect and then select AI Companion. This will display a 6-letter code both as a barcode and as plain text.
3. On the companion app, either type in the 6-letter code and click the “Connect to App Inventor” button or scan the barcode.

If everything is configured properly, you should see the IHaveADream app on the phone and when you click a button, you should hear the appropriate speech.

***Nice work! Complete the Self-Check Exercises and Portfolio Reflection Questions as directed by your instructor.***

# 

# 