CHAPTER 2

# Working with Media

GLOBAL EDITION
Starting Out with App Inventor for Android
Tony Gaddis • Rebecca Halsey
ALWAYS LEARNING
PEARSON

## Topics

- Displaying images
- Duplicating Blocks and Using Drop Downs
- Sounds
- Color blocks
- Layout Components
- Commenting Blocks

## Displaying Images

Displaying an Image as a Screen Background

- There are various ways to display an image in an App Inventor app.
- Images can be displayed as the background for a screen in an Image `Component` or on a `Button` component.
- An image must be uploaded to a project before it can be displayed.
- The media column in the Designer let's you manage the image files.

## Displaying Images

Displaying an Image as a Screen Background

•Before you can display an image, It must be uploaded to your project on the App Inventor server.

•Recommended formats are `.png` and `.jpg`.

•To display an image as a screen's background image, select the `Screen1` component.

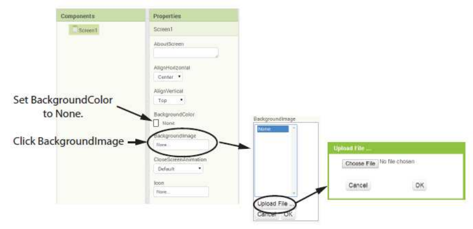
•In the properties column set the `BackgroundColor` property to *None*.

•Click the `BackgroundImage` property.

•Select a previously uploaded image, or upload a new image.

## Displaying Images

Figure 2-1 Changing the BackgroundImage Property (Source: MIT App Inventor 2, Pearson Education, Inc.)

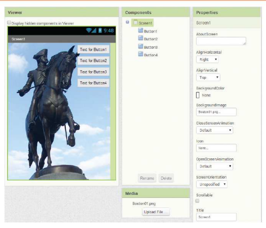


## Displaying Images

Displaying an Image as a Screen Background

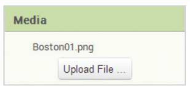
You can continue to place components on the screen after setting the screen's `BackgroundImage` property.

Figure 2-3 A Screen with a Background Image and Four Button Components (Source: MIT App Inventor 2, Pearson Education, Inc.)



## Displaying Images

Figure 2-4 The Media Column (Source: MIT App Inventor 2)

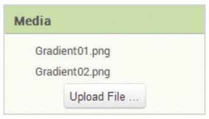


Using the Media Column to Upload Files

- As seen in Figure 2-4, when you upload an image to a project, the image file's name will appear in the media column.
- The media column has an *Upload File*… Button.
- You can select and upload media files to your project without assigning them to any specific property.

## Displaying Images

Figure 2-11 Two `.png` Files Uploaded (Source: MIT App Inventor 2)



Switching the Screens Background Image in Code

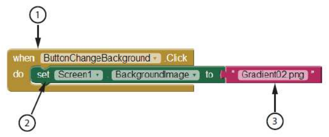
- The Blocks Editor sets the screen's background image property while the app is running.
- Suppose you have used the Media column to upload the two `.png` files.

## Displaying Images

Switching the Screens Background Image in Code

- Now you want the user to be able to click a button to change the background image to `Gradient02.png`.
- Add a `Button` component named `ButtonChangeBackground`.
- In the Blocks Editor you create the event handler shown in figure 2-12.

Figure 2-12 The `Click` Event Handler for the `ButtonChangeBackground` Button (*Source:* MIT App Inventor 2)


1
when ButtonChangeBackground Click
do set Screen1 BackgroundImage to
" Gradient02.png "
2
3

## Displaying Images

Switching the Screens Background Image in Code

A closer look at the blocks shown in figure 2-12

1. This is the `when ButtonChangeBackgroud.Click do` event handler.
2. This is the `set Screen1.BackgroundImage to` block. The purpose of this block is to set the `Screen1` component's `BackgroundImage` property to a value.
3. This is the text string block and its value is set to `Gradient02.png`.

When the `ButtonChangeBackground` button is clicked, set the `Screen1` component's `BackgroundImage` property to `Gradient02.png`.

## Displaying Images

The Image Component

- Another way to display an image is with the `Image` component found in the *User Interface* section of the Designer's pallet.
- The `Image` component allows you to specify the image's size with its `Width` and `Height` properties.
- The Image component has the following properties:

***Picture*** – Specifies the image file that the component displays.

***Visible*** – Can be set to *showing* or *hidden*.

***Width and Height*** – Specifies the images with on the screen. It can be set to Automatic, Fill parent, or a specific number of pixels.

## Displaying Images

Making Clickable Images with `Button` components

- A *clickable image* is an image that the user can click to make an action happen.
- `Button` components have an `Image` property. The `Image` property causes an image to be displayed on the button.
- In Figure 2-28 notice the button named `ButtonSwitch` is selected.
- In the Properties column it's Image property is set to `SwitchUp.png`.
- Buttons can display both text and an image.

## Displaying Images

When the user clicks the `ButtonSwitch` component, the app performs two actions:

1. It changes the image displayed on the button to the switch in the down position and
2. It changes the text that is displayed in the `LabelOutput` component to the *The switch is down.*

The button's *Click* event handler is shown in Figure 2-29.

Figure 2-28 A Button Component Displaying an Image (*Source:* MIT App Inventor 2)

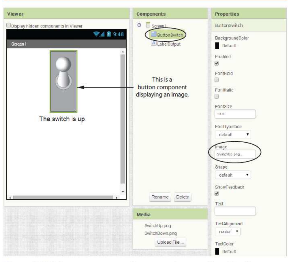
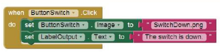


Figure 2-29 The `ButtonSwitch` Component's `Click` Event Handler (*Source:* MIT App Inventor 2)

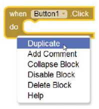

when ButtonSwitch .Click
do
set ButtonSwitch . Image to " SwitchDown.png "
set LabelOutput . Text to " The switch is down. "

## Duplicating Blocks and Using Drop Downs

Sometimes the easiest way to create a block is to duplicate one that you already have.

- By right-clicking a block, the menu shown in Figure 2-41 appears.

Figure 2-41 The Block Menu (*Source:* MIT App Inventor 2)

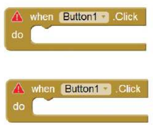


- Another way to duplicate a block is by copying and pasting it.
- Select the block, then click Ctrl + C to copy, then Ctrl + V to paste.

## Duplicating Blocks and Using Drop Downs

A red triangle with an exclamation mark (⚠) on both blocks.

Figure 2-42 Duplicate Blocks (*Source:* MIT App Inventor 2)



And error message will appear if you click the symbol (⚠).

## Duplicating Blocks and Using Drop Downs

•In Figure 2-43 notice the error massage: *This is a duplicate event handler for this component.*

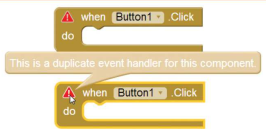
•Each component can only have one event handler.

•To fix the error change one of the blocks into a Click event handler for a component other than Button1.

•Click the down-arrow and a dropdown menu will appear.

•The error symbol is no longer shown.

Figure 2-43 Error Message for Duplicate Blocks (*Source:* MIT App Inventor 2)



## Duplicating Blocks and Using Drop Downs

**Figure 2-44** Changing the Duplicate Block's Component (*Source:* MIT App Inventor 2)

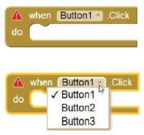
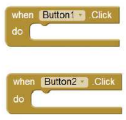


**Figure 2-45** The Modified Block (*Source:* MIT App Inventor 2)

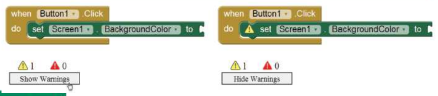


## Duplicating Blocks and Using Drop Downs

Errors and Warnings

- At the bottom of the workspace in the Blocks Editor is a set of "counters" that report the number of warnings and errors.
- To see which blocks have warnings, you must click the *Show Warnings* buttons.
- When you click *Show Warnings*, it changes to *Hide Warnings*.

**Figure 2-47** Showing Warnings (*Source:* MIT App Inventor 2)


when Button1 .Click
do set Screen1 . BackgroundColor to
1 0
Show Warnings
when Button1 .Click
do set Screen1 . BackgroundColor to
1 0
Hide Warnings

## Sounds

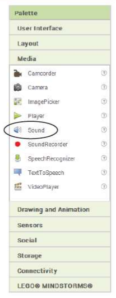
App inventor provides two components for playing sound files

1. The `Sound` component is recommended for small files such as those containing short sound affects.
2. The `Player` component is recommended for larger files such as those containing music.

Recommended sound formats are `.mp3` and `.wav`.

## Sounds

**Figure 2-49** The Sound Component is in the Media Palette (*Source:* MIT App Inventor 2)



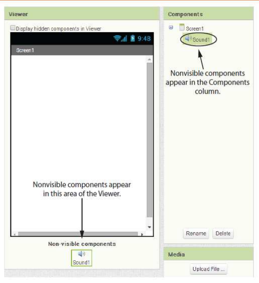
## Sounds

The Sound Component

- In the Designer, the `Sound` component is found in the Media section of the Palette.
- Drag it from the Palette to the Viewer.
- The `Sound` component is a non-visible component.
- When you drop a `Sound` component it appears in the area below the screen (Figure 2-50).

## Sounds

Figure 2-50 The Sound Component is a Nonvisible Component (Source: MIT App Inventor 2)


Viewer
Display hidden components in Viewer
9:48
Screen1
Nonvisible components appear in this area of the Viewer.
Non-visible components
Sound1
Components
Screen1
Sound1
Nonvisible components appear in the Components column.
Rename
Delete
Media
Upload File ...

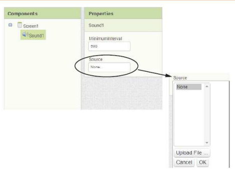
## Sounds

The Sound Component

•Clicking the `Source` property in the Properties column causes a dialog box to appear.

•You can either select a previously uploaded file or you can click on the *Upload File…* button.

Figure 2-51 The Source Property (*Source:* MIT App Inventor 2)

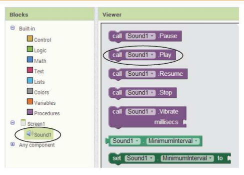


## Sounds

The Sound Component

Find the block for the `Play` method by clicking the name of the `Sound` component in the Blocks column.

Figure 2-52 The Sound Component's `Play` Method (*Source:* MIT App Inventor 2)



## Sounds

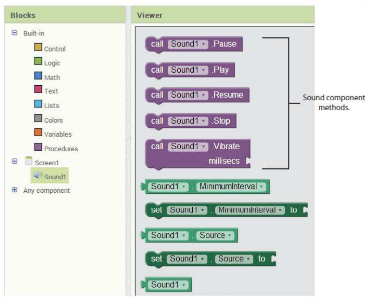
Pausing, Resuming, and Stopping the Sound

Other methods to control a sound bar:

- `Pause` – This method pauses an audio file.
- `Resume` – After you have use the `Pause` method to pause an audio file, you can use the `Resume` method to start playing again.
- `Stop` – This method stops the audio file that is currently playing.

## Sounds

Figure 2-57 The Sound Component Methods (*Source:* MIT App Inventor 2)

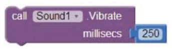


## Sounds

Vibrating the Phone

- You may also use the `Sound` component to vibrate the phone.
- The emulator does not vibrate.
- To vibrate the phone, call the `Sound` components `Vibrate` method.
- The `Vibrate` method will cause the phone to vibrate for a specified number of milliseconds.
- 1000 ms equals one second.

Figure 2-58 The Sound Component's `Vibrate` Method (*Source:* MIT App Inventor 2)

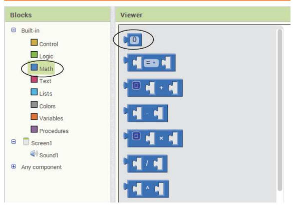


## Sounds

Vibrating the Phone

- Use the number block anytime you need to specify a number in a program.
- The number block is found under the *Built-In*, In the *Math* drawer.

Figure 2-59 The Number Block (*Source:* MIT App Inventor 2)



## Sounds

The `Player` Component

- To play a long audio file, such as an entire song, it is recommended that you use the `Player` component.
- Here is a summary of its properties:
- ***Source*** – Designates an audio file.
- ***Volume*** – Set a value from zero through 100.
- ***Loop*** – Can cause the audio file to loop, or play repeatedly.

## Sounds

The Player components methods:

- `Start` – Starts the audio file playing.
- `Pause` – Pause an audio file that is currently playing.
- `Stop` – Stop an audio file that is currently playing.
- `Vibrate` – Vibrates the phone a specified number of milliseconds.

## Color Blocks

Use Color blocks to represent and work with colors.

- Many of the user interface components in App Inventor have properties that determine the components color.
- For example:
  - `Screen`, `Button` and `Label` components all have a `BackgroundColor` property that determines the component's background color.

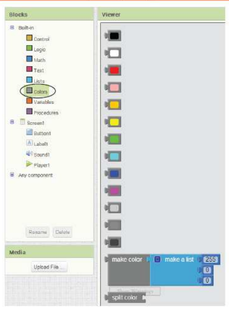
## Color Blocks

Use Color blocks to represent and work with colors.

- `Button` and `Label` components also have a `TextColor` property that determines the color of the component's text.
- There are a selection of `Color` blocks in the Blocks Editor to set the value of the color property.

## Color Blocks

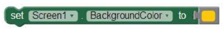
Figure 2-63 The `Color` Blocks (Source: MIT App Inventor 2)



## Color Blocks

Figure 2-64 shows the blocks for programmatically setting `Screen1`'s `BackgroundColor` property to `Orange`.

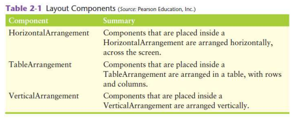
Figure 2-64 Setting `Screen1`'s BackgroundColor Property to Orange
(*Source:* MIT App Inventor 2)



## Layout Components

- A layout component is a container that governs the position of the components it contains.
- You did not specify the exact location of a component on the screen.
- Let a *layout component* control the position of the components for you.
- The layout components are found in the *Layout* section of the Palette.

**Table 2-1** Layout Components (*Source:* Pearson Education, Inc.)



| Component | Summary |
|---|---|
| HorizontalArrangement | Components that are placed inside a HorizontalArrangement are arranged horizontally, across the screen. |
| TableArrangement | Components that are placed inside a TableArrangement are arranged in a table, with rows and columns. |
| VerticalArrangement | Components that are placed inside a VerticalArrangement are arranged vertically. |

## Layout components

`HorizontalArrangement`

- Figure 2-67 shows a screen that has a `HorizontalArrangement` component.
- The `HorizontalArrangement` component has `Width` and `Height` properties and each is set to *Automatic*.
- The width and height of the component is determined by the width and height of the components inside of it.
- The `HorizontalArrangement` component also has `AlignHorizontal` and `AlignVertical` properties. They determine the alignment components inside the `HorizontalArrangement` component.

## Layout Components

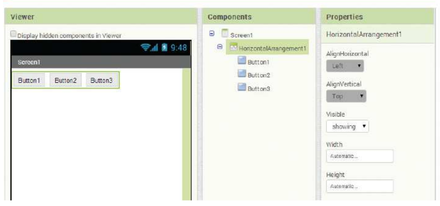


Figure 2-67 Three Buttons Inside a HorizontalArrangement (*Source:* MIT App Inventor 2)

Screen1
HorizontalArrangement1
Button1
Button2
Button3

HorizontalArrangement1
AlignHorizontal: Left
AlignVertical: Top
Visible: showing
Width: Automatic...
Height: Automatic...

## Layout Components

`TableArrangement`

- Components placed inside a `TableArrangement` are arranged in a grid.
- `TableArrangements` have a:
- `Row` property – That determines the number of rows
- `Columns` property – That determines the number of columns.

## Layout Components

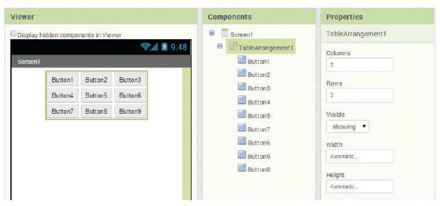


Figure 2-69 A TableArrangement with 3 Rows and 3 Columns (*Source:* MIT App Inventor 2)

Screen1
TableArrangement1
Button1
Button2
Button3
Button4
Button5
Button7
Button6
Button9
Button8

TableArrangement1
Columns: 3
Rows: 3
Visible: showing
Width: Automatic...
Height: Automatic...

## Layout Components

`VerticalArrangement`

Figure 2-70 shows a screen that has a `VerticalArrangementComponent`

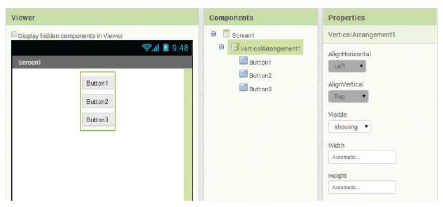


Figure 2-70 Three Buttons Inside a VerticalArrangement (*Source:* MIT App Inventor 2)

Screen1
VerticalArrangement1
Button1
Button2
Button3

VerticalArrangement1
AlignHorizontal: Left
AlignVertical: Top
Visible: showing
Width: Automatic...
Height: Automatic...

## Layout Components

Using Multiple Components in the Same Screen

- Quite often you will have to use multiple layout components.
- You can nest a layout component inside of another component.
- Figure 2-71 shows a screen that has `TableArrangement` component with a `VerticalArrangement` component .

## Layout Components

Here are details details about the components in figure 2-71.

- The `TableArrangement` has two columns and one row.
- The `Button` component is in the `TableArrangementsLeft` column.
- The `VerticalArrangement` is in the `TableArrangement`'s right column.
- The `Image` components are in the `VerticalArrangement`.

## Vertical Components

**Figure 2-71** A Screen with Nested Arrangement Components (*Source:* MIT App Inventor 2)

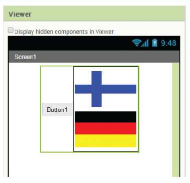
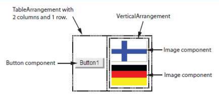


**Figure 2-72** The Screen Layout (*Source:* MIT App Inventor 2)

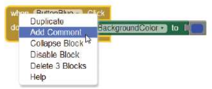


## Commenting Blocks

A *comment* is a note that programmer writes into the program, explaining some part of the code.

- In the Block's Editor, you can add a comment by right-clicking the block, then selecting *Add Comment* from the menu that pops up.
- A small question mark will appear on the block.
- Click the question mark.
- A small note editor will appear.

**Figure 2-79** Right-click a Block to Get this Menu



## Commenting Blocks

- Comments did not affect the execution of your app in any way.
- They make you’re program more understandable.

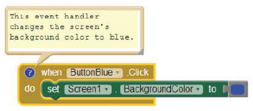


Figure 2-80 Note Editor for Comments (*Source:* MIT App Inventor 2)

```
This event handler
changes the screen's
background color to blue.
```

```
when ButtonBlue .Click
do set Screen1 . BackgroundColor to
```