starting out with >>> APP INVENTOR FOR ANDROID

CHAPTER 4

Decision Blocks and Boolean Logic

TONY GADDIS · REBECCA HALSEY

Topics

- Introduction to Decision Blocks
- Relational Operators and the *if* Block
- The *if then else* block
- A First Look at Comparing Strings
- Logical Operators
- Nested Decision Blocks
- The *if then else if* block
- Working with Random Numbers
- The *Screen's Initialize* Event
- The *ListPicker* Component
- The *Checkbox* Component

The `Screen`'s `Initialize` Event

- If you need to perform set up operations when the app starts, you can create an event handler for the *`Initialize`* event.
- To create an *`Initialize`* event handler for the *`Screen1`* component, go to the *`Screen1`* drawer in the Blocks column and select the *`when`* `Screen1.Initialize` *`do`* block.

The Screen's Initialize Event

Figure 4-79 shows an example of a *Screen1.Initialize* event handler. It sets the screen's background color randomly to either blue, yellow, or green.

Figure 4-78 The Screen1 Component's Initialize Event Handler
(*Source:* MIT App Inventor 2)

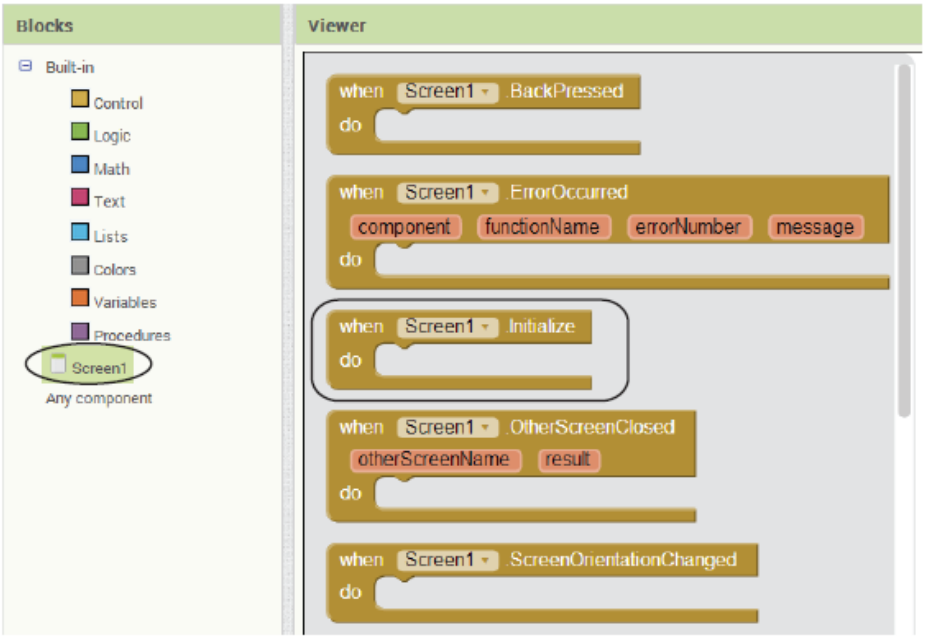
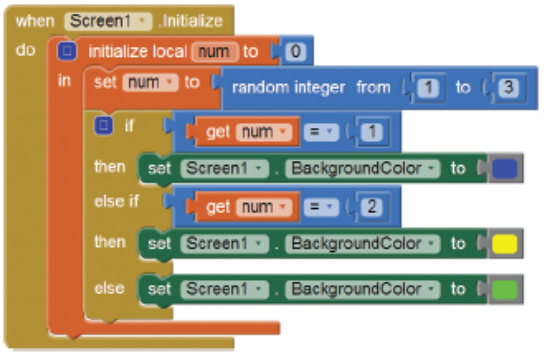


Figure 4-79 Example Screen1.Initialize Event Handler (*Source:* MIT App Inventor 2)



The `ListPicker` Component

A *`ListPicker`* component displays a list of items and allows the user to select an item from the list.

Figure 4-80 shows an example app (ListPickerDemo) running in the emulator.

Figure 4-80 The ListPickerDemo App Running in the Emulator (*Source:* MIT App Inventor 2)

The `ListPicker` Component

`ListPicker` has all the same properties as a *`Button`* component, plus a couple of extra ones:

- *`ElementsFromString`*: This property holds the list of items that is displayed when the user clicks the *`ListPicker`* as seen in Figure 4-81.
- *`Selection`*: Once the user selects an item from the list, the selected item is copied into the *`Selection`* property.
- When the user selects an item from a *`ListPicker`*'s list, an *`AfterPicking`* event is triggered.
- Figure 4-82 shows the *`AfterPicking`* event handler for the *`ListPicker`*.

The ListPicker Component

Figure 4-81 The ListPicker Component's ElementsFromString Property
(*Source:* MIT App Inventor 2)

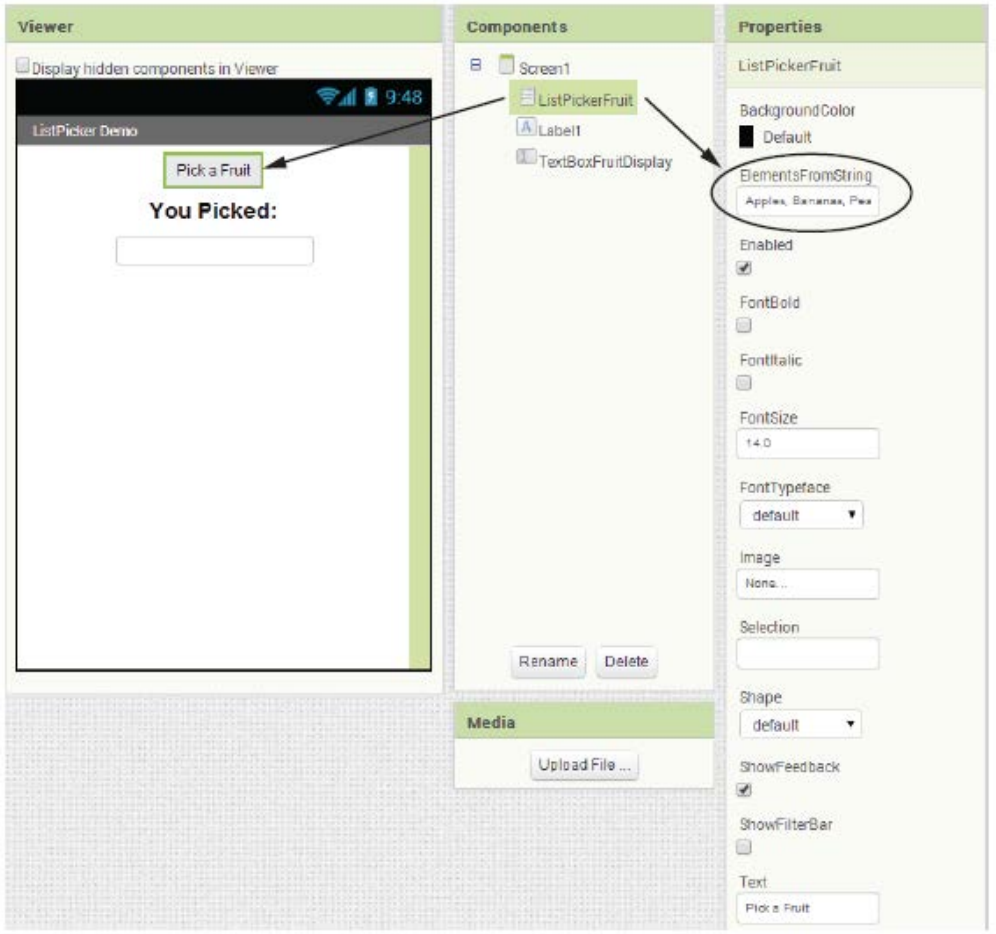


Figure 4-82 The `AfterPicking` Event Handler (*Source:* MIT App Inventor 2)

The CheckBox Component

- The *CheckBox* component appears as a small box with some accompanying text.
- In the *Designer*, *CheckBox* components are found in the *User Interface* section of the Pallet.
- You will mostly be concerned with the *Text* and *Checked* properties.
- The *Text* property determines the text that is displayed next to the small box.
- The *Check* property indicates whether the component is checked or unchecked.

The CheckBox Component

Figure 4-89 shows the PizzaToppings app in the Designer and Figure 4-90 shows how it initially appears in the emulator.

Figure 4-89 The PizzaToppings App in the Designer (*Source:* MIT App Inventor 2)

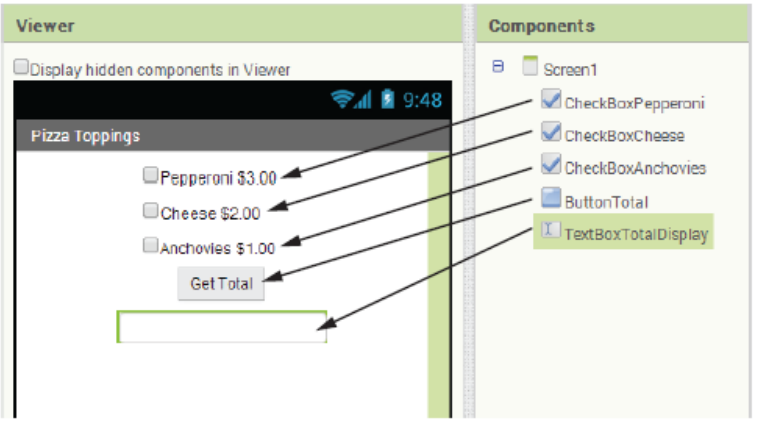
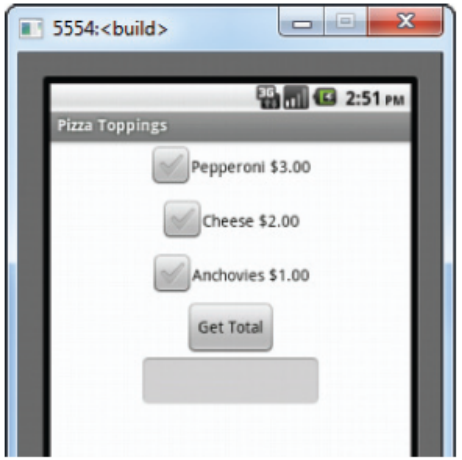


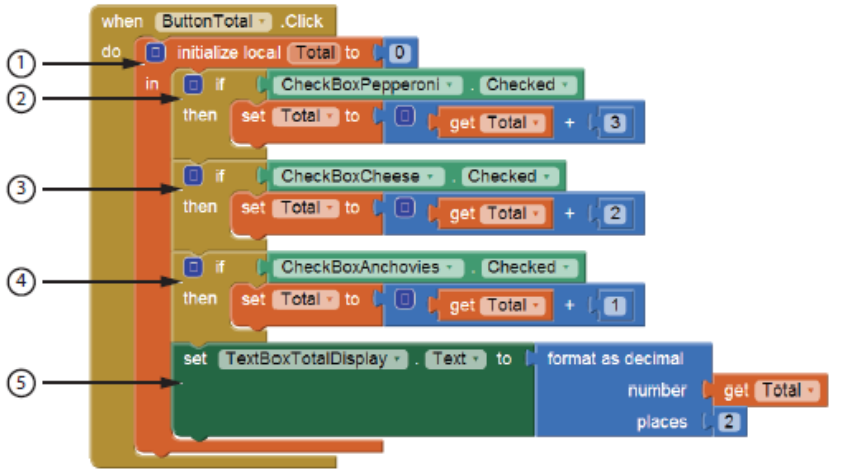
Figure 4-90 The PizzaToppings App Initially in the Emulator (*Source:* MIT App Inventor 2)



The CheckBox Component

Figure 4-91 shows the apps workspace in the Blocks Editor.

Figure 4-91 The App's Workspace in the Blocks Editor (*Source:* MIT App Inventor 2)



The CheckBox Component

1. This block initializes a local variable named *Total*, with the initial value of 0.
2. This *if then* block determines whether the *CheckBoxPepperoni* component is checked. If so, 3.00 is added to the *Total* variable.
3. This *if then* block determines whether the *CheckBoxCheese* component is checked. If so, 2.00 is added to the *Total* variable.

The CheckBox Component

4. This *If then* block determines whether the *CheckBoxAnchovies* component is checked. If so 1.00 is added to the *Total* variable.
5. This block displays the value of the *Total* variable, rounded to two decimal places, in the *TextBoxTotalDisplay* component.

The CheckBox Component

The Changed Event

- Any time a *CheckBox* component's *Checked* property changes, a *Changed* event happens.
- In the Blocks column, click the name of the *CheckBox* component then select the block for the *Changed* event handler.

The CheckBox Component

The Changed Event

- Figure 4-93 shows a PizzaToppings2 app in the Designer and Figure 4-94 shows how it initially appears in the emulator.
- This app serves the same purpose as the Pizza Toppings app except it does not require the user to click a button to calculate the total cost.

The CheckBox Component

Figure 4-93 The PizzaToppings2 App in the Designer (*Source:* MIT App Inventor 2)

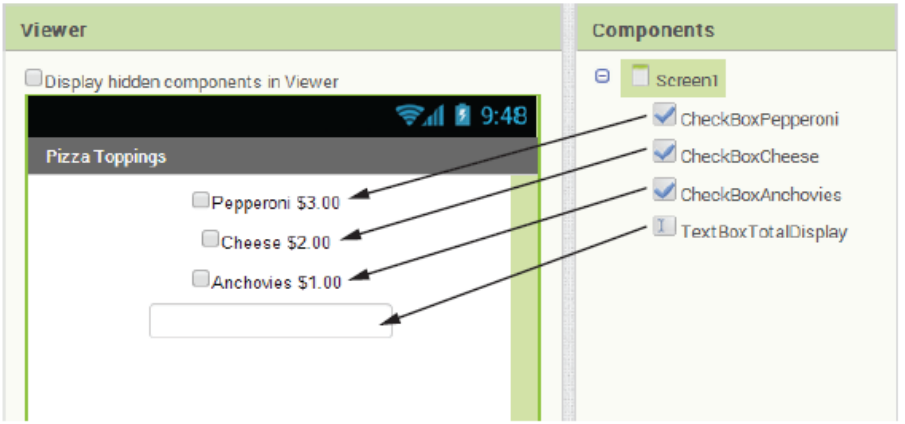
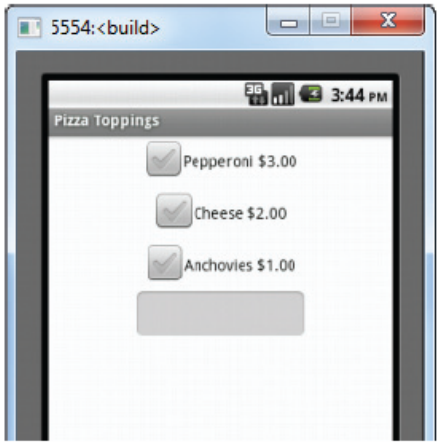


Figure 4-94 The PizzaToppings App Initially in the Emulator (*Source:* MIT App Inventor 2)

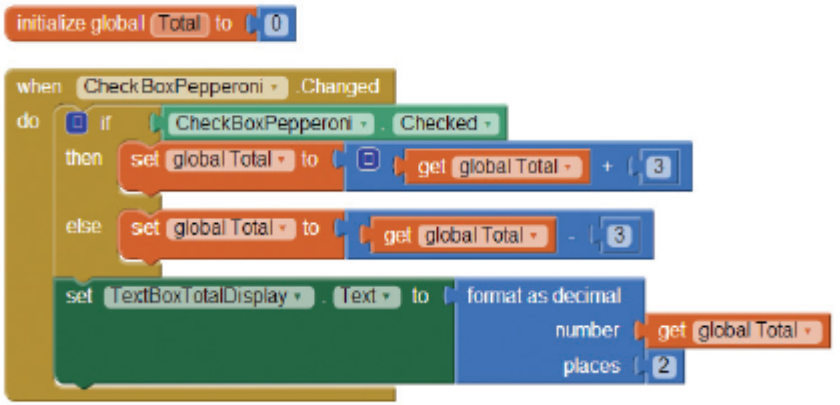


The CheckBox Component

The Changed Event

- This app creates a global variable named *Total*, initialized to 0.
- The app also has a Changed event handler for each of the *CheckBox* components.

Figure 4-95 The `CheckBoxPepperoni.Changed` Event Handler
(*Source:* MIT App Inventor 2)



The CheckBox Component

The Changed Event

- The event handler works like this:
- If *CheckBoxPepperoni* is checked, then add 3.00 to the *Total* variable.
- Otherwise, subtract 3.00 from the *Total* variable.
- Display the *Total* variable rounded two decimal places.

The CheckBox Component

The Changed Event

Figure 4-96 shows the *CheckBoxCheese.Changed* and *CheckBoxAnchovies.Changed* event handlers, which worked in a similar fashion.

Figure 4-96 The `CheckBoxCheese.Changed` and `CheckBoxAnchovies.Changed` Event Handler (*Source:* MIT App Inventor 2)