CHAPTER 13

Other App Inventor Capabilities

Topics

- Recording Audio
- Taking a photo with the phone's camera
- The `Camcorder` component
- Using the `ImagePicker` component to select an image from the phone's gallery
- Playing video
- Selecting contacts from the contact list and placing phone calls
- Scanning a bar code
- Using voice recognition
- Connecting to a Twitter account
- Using `TinyWebDB` to create a web database

Recording Audio

- App Inventor provides a `SoundRecorder` component.
- The `SoundRecorder` component is found in the Media Palette. It is a non-visible component.
- The component provides three events and two methods. The two methods are:
  - `SoundRecorder.Start`
  - `SoundRecorder.Stop`.

Recording Audio

The events are:

- `StartedRecording`
- `StoppedRecording`
- `AfterSoundRecorded`

Figure 13-1 Start and Stop Recording Methods (Source: MIT App Inventor 2)

Recording Audio

`SoundRecorder.StartedRecording` event — To perform an action as soon as possible.

`SoundRecorder.StopRecording` event — To perform an action once the recording has stopped.

Recording Audio

Figure 13-2 shows an example of using the `StartedRecording` and `StoppedRecording` event handlers to enable and disable the appropriate buttons.

Figure 13-2 Started Recording and Stopped Recording Events (Source: MIT App Inventor 2)

Recording Audio

- `SoundRecorder` has an `AfterSoundRecorded` event.
- This event has one argument, the `Sound` component.
- Figure 13-3 shows how to store a recording to a variable in the `AfterSoundRecorded` event handler.

Figure 13-3 Store a Recording to a Variable (Source: MIT App Inventor 2)

Taking a Photo with the Phone's Camera

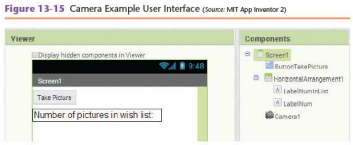
App Inventor has a camera component in the *Media* Palette.

- The `Camera` component — is non-visible and has one property, one method, and one event.
- `UseFront` property — If enabled the front-facing camera will open on the device.
- `Camera.TakePicture` method — Is used to invoke the device's camera.
- `Camera.AfterPictureEvent` event — Allows developers to program actions after the picture is taken.

Taking a Photo with the Phone's Camera

To demonstrate how to use the `Camera` component, look at the "Which List" in Figure 13-15

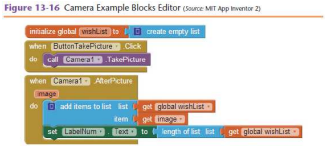
Figure 13-15 Camera Example User Interface (Source: MIT App Inventor 2)



Taking a Photo with the Phone's Camera

- Figure 13-15 demonstrates the `ButtonTakePicture.Click` event to invoke the `Camera` component using the `Camera1.TakePicture` event handler.
- In Figure 13-16 a list variable, `wishList`, is created using the list `create empty list` block.

Figure 13-16 Camera Example Blocks Editor (Source: MIT App Inventor 2)



The `Camcorder` Component

- App Inventor has a `Camcorder` component in the Media Palette.
- The `Camcorder` component is non-visible and has one method and one event.
- The method of the `Camcorder` component is `Camcorder.RecordVideo`. It is used to invoke the device's camcorder.
- The event is `Camcorder.AfterRecording` event. This event allows developers to program actions after the video is taken.

Using the `ImagePicker` Component

- The `ImagePicker` accesses your device's photo gallery and allows you to select a picture.
- `Selection` property — Allows you to use the selected image in your application.
- Open the picker by clicking an `ImagePicker` on a user interface or by calling the `ImagePicker.Open` method programmatically.
- `ImagePicker` has `BeforePicking` and `AfterPicking` events.

Playing Video

- App Inventor has a `VideoPlayer` component in the *Media* palette.
- Upload your video the same way you upload images and sound files.
- Videos should be smaller than 1MB in size.
- Supported formats:
  - .wmv
  - .3gp
  - MPEG-4.

Playing Video

`VideoPlayer` properties:

- `Source` – The name of the video that is uploaded.
- `Visible` – Will tell your application to show video if set to `true`.
- `Height and Width` – Control the size of the player on the Screen.
- `Volume` – Controls the video's volume.
- `FullScreen` – Will take over the entire screen if set to `true`.

Selecting Contacts from the Contact List and Placing Phone Calls

App Inventor provides three components that are found in the Social Palette:

- `ContactPicker`
- `EmailPicker`
- `PhoneNumberPicker`

Selecting Contacts from the Contact List and Placing Phone Calls

The `ContactPicker` and `PhoneNumberPicker` Components

- The `ContactPicker` and `PhoneNumberPicker` allow a user to select a contact and use the information about that contact in an application.
- For example, the `PhoneNumberPicker` will show a list of the phone numbers in the contacts list.
- The `ContactPicker` will show a list of contact names.

Selecting Contacts from the Contact List and Placing Phone Calls

The `ContactPicker` and `PhoneNumberPicker` Components

- You can open these pickers by clicking the `ContactPicker` or `PhoneNumberPicker` button placed on the user interface.
- They can also be opened by calling their `Open` method programmatically in the Blocks Editor
- These pickers have `BeforePicking` and `AfterPicking` events.

Selecting Contacts from the Contact List and Placing Phone Calls

`EmailPicker` Component

- As you type an email into the `EmailPicker`, the email addresses in the contact list are searched and filtered.
- Often, the `EmailPicker` is used in conjunction with a `Button`.

Figure 13-40 Example Email Picker User Interface (Source: MIT App Inventor 2)

Selecting Contacts from the Contact List and Placing Phone Calls

`EmailPicker` Component

- Notice that a button is used in conjunction with this picker.
- Figure 13-41 shows how you might use a `Button Click` event handler to process the chosen email addresses.

Figure 13-41 Example Email Picker Blocks (Source: MIT App Inventor 2)

Scanning a Barcode

- App Inventor has a `BarcodeScanner` component in the Sensors Palette.
- The device needs a barcode scanner program installed, such as the Barcode scanner application ZXing.
- An application invokes the `BarcodeScanner` by the `DoScan` method block.
- The `BarcodeScanner` has an `AfterScan` event that will allow an application to retrieve and process the result of a scan.

Scanning a Barcode

- Figure 13-42 shows an example user interface which uses a `BarcodeScanner`.
- Figure 13-42 shows how you can use a `Button Click` event handler to invoke the `BarcodeScanner` by calling the `DoScan` method.

Scanning a Barcode

Figure 13-42 Example BarcodeScanner User Interface (Source: MIT App Inventor 2)

Figure 13-43 Example BarcodeScanner Blocks (Source: MIT App Inventor 2)

Using Voice Recognition

- App Inventor's `SpeechRecognizer` component coverts speech to text.
- `SpeechRecognizer` has one property, the `Result`.
- After the recording stops, the `AfterGettingText` event is triggered.
- There is also a `BeforeGettingText` event handler in which developers can perform any needed activities before the recording starts.

Connecting to a Twitter Account

- A Twitter component can be found in the Social Palette.
- You can search for tweets and tags, tweet messages and more.
- To use this component visit http://twitter.com/oauth_clients/new
- Once registered, you will receive a consumer key.

Connecting to a Twitter Account

- Once the key and secret are set, calling the `Twitter.Authorize` method will transfer control to Twitter and ask the user to login.
- We will look at the `Twitter.IsAuthorized` event, which is triggered after a user logs in, in Tutorial 13-6.

TinyWebDB

- The `TinyWebDB` component allows applications to share data stored on the Web.
- It stores data in tag-value pairs.
- The database is on the Web and therefore multiple applications can access it.
- The `TinyWebDB` component requires that you set up a custom Web service to host the database.
- You can find it at http://appinventor.mit.edu/explore/content/custom-tinywebdb-service.html

TinyWebDB

- There are three events: `GotValue`, `ValueStored` and `WebServiceError`.
- `GotValue` event — Is triggered if the `GetValue` is successful and will provide the tag and value for use in the event handler.
- `ValueStored` event — is triggered when a successful store has been processed.
- `WebServiceError` event — Is triggered when a successful store has been processed.