Advanced Elective Module (AEM)
Creating Cool Mobile App

Module 07
Location-based Development

What You'll Learn

Location-based Development with App Inventor

- Introduction to Google Maps
- Introduction to location-based development with App Inventor
- Activity starter component for launching other Android applications
- List Picker component for choosing list of locations

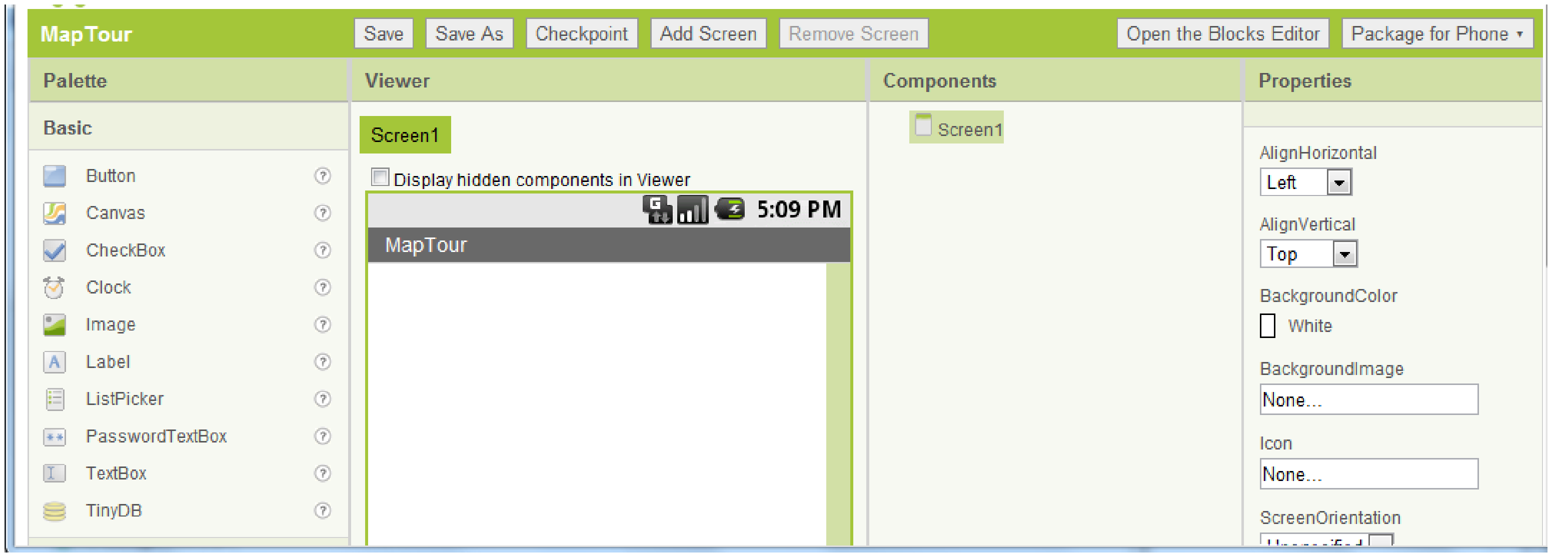
What You’ll Create

This tutorial introduces the ActivityStarter component for launching arbitrary Android Apps and the ListPicker component for allowing a user to choose from a list of items. You'll build MapTour, an app for visiting Singapore tourist destinations with a single click. Users of your app will be able to visit the Esplanade, zoo and the Flyer in quick succession.

Starting App Inventor Environment

Getting Started

1. Go to “http://ai2.appinventor.mit.edu/
2. Sign-in with your gmail account
3. Click ‘new’ project
4. Enter name for project : **MapTour**
5. Set the screen’s title to “MapTour”.
6. Open the Blocks Editor and connect to the phone.



AlignHorizontal
Left

AlignVertical
Top

BackgroundColor
White

BackgroundImage
None...

Icon
None...

ScreenOrientation

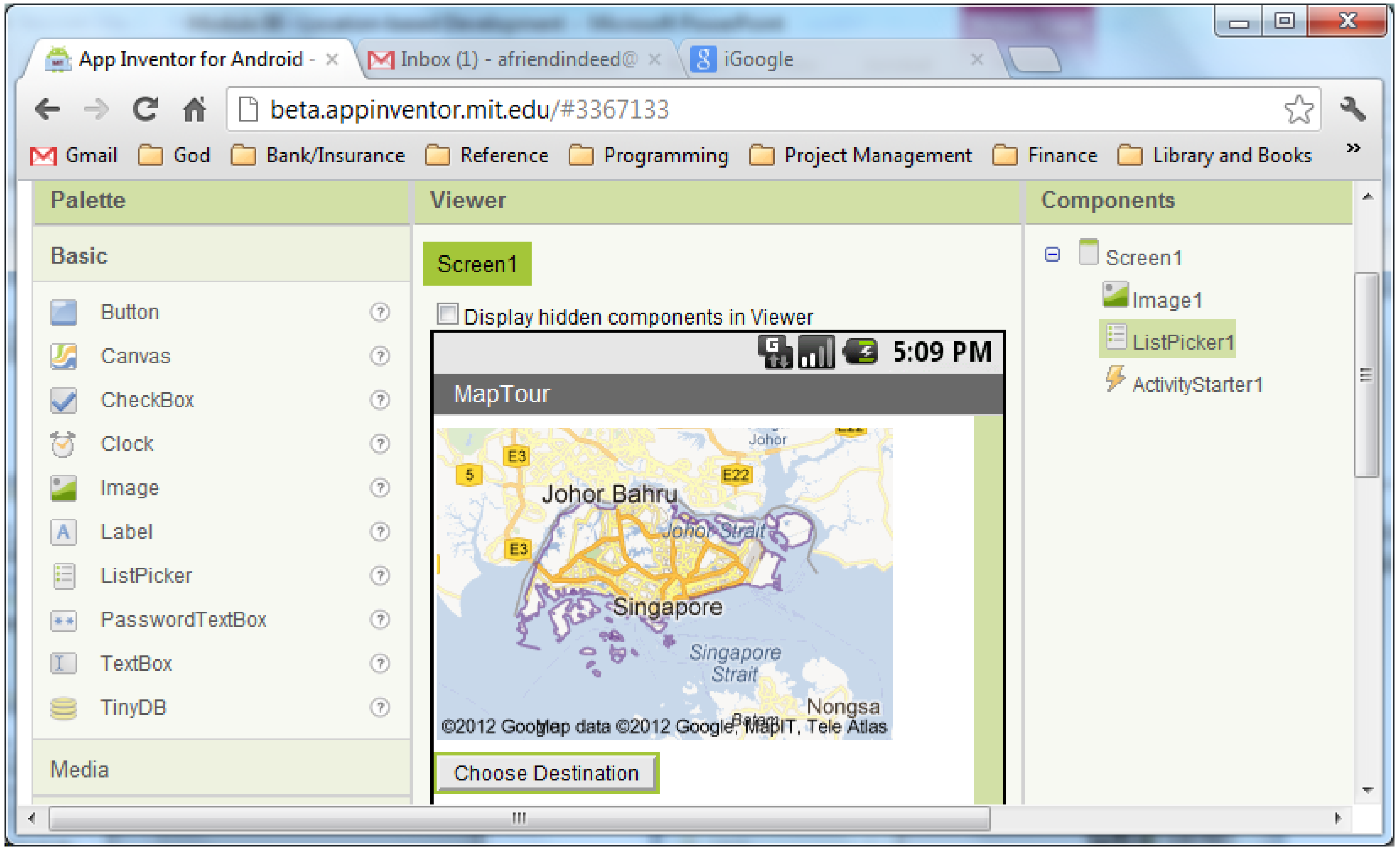
Building MapTour

Overview

You'll design the app so that a list of destinations appears. When the user chooses one, the Google Maps app is launched to display a map of the destination.

Components for MapTour

The user interface for MapTour is simple: you'll have a single ListPicker component and an ActivityStarter (non-visible) component. The design view should look like this when you're done:



Components for MapTour

The components listed below were used to create this designer window. Drag each component from the Palette into the Viewer and name it as specified:

Component Type	Palette Group	What you'll name it	Purpose of Component
Image	User Interface	Image1	Show a static image of Singapore map on screen
ListPicker	User Interface	ListPicker1	Display the list of destinations
ActivityStarter	Connectivity	ActivityStarter1	Launches the maps app when a destination is chosen

With the map03.gif" file in your computer desktop "App Inventor" folder, choose **Add** in the **Media** section to load it into your project. For it to appear, you'll also need to set this as the Picture property of Image1.
http://math.nie.edu.sg/atcm/images/map03.gif

The **ListPicker** component has an associated button-- when the user clicks it, the choices are listed. Set the text of that button by setting the **Text** property of **ListPicker1** to "Choose Destination".

Setting properties of ActivityStarter

ActivityStarter is a component that lets you launch any Android app - browser, Maps or even another one of your own apps. When another app is launched from your app, the user can click the back button to get back to your app.

MapTour will launch Maps application to show the map user chooses. The user can then hit the back button to return to MapTour and choose a different destination.

ActivityStarter is a low-level component in that you'll need to set some properties that an Android programmer is familiar with, but many of us are unfamiliar. Not to worry, however, you just need to get the sample information provided in the next slide and you too can easily learn how to launch apps like Maps from apps.

Setting properties of ActivityStarter

For ActivityStarter to launch the Maps application, set the following properties of ActivityStarter in the Component Designer:

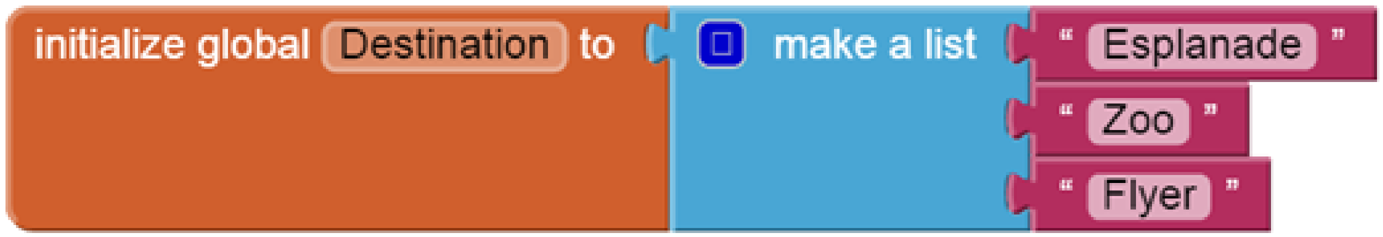
Property	Value
Action	android.intent.action.VIEW
ActivityPackage	com.google.android.apps.maps
ActivityClass	com.google.android.maps.MapsActivity

In the Blocks Editor, you'll set one more property, **DataUri** , which will allow you to launch Maps with a particular map being displayed. This property must be set in the Blocks Editor because its value is based on what the user chooses - Esplanade, zoo or the Flyer.

Create a list of destinations

Open the blocks editor and create a variable with the list of destinations for the MapTour.

The blocks will look like this:



```
initialize global Destination to  make a list  " Esplanade "
                                               " Zoo "
                                               " Flyer "
```

Add behaviours to the components

MapTour has two behaviours:

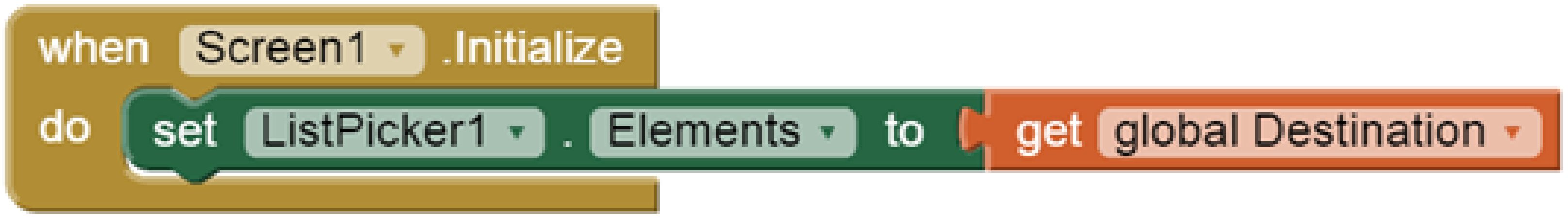
1. MapTour loads the destinations into the ListPicker component when the app starts so user can choose.
2. When the user chooses a destination from the ListPicker , the app should open Maps and make it search for the destination the user chooses. A map of the destination would then be shown.

The ListPicker component displays a list of items when the user clicks on it. ListPicker has a property named Elements . If you set Elements to a list, the items in the list will appear in the ListPicker . For this app, you want to set the ListPicker' s Elements property to the destinations list you just created.

Because destinations should be available when the app starts, we shoulc define this behavior in Screen1.Initialize event.

Add behaviors to the components

The blocks should look like this:



```
when Screen1 .Initialize
do  set ListPicker1 . Elements to  get global Destination
```

How the Blocks Work

Screen1.Initialize is triggered when the app begins. The event-handler just sets the Elements property of ListPicker so that the three destinations will appear.

Test this behavior. On the phone, click the button labelled "Choose Destination". The list picker should appear with the three items.

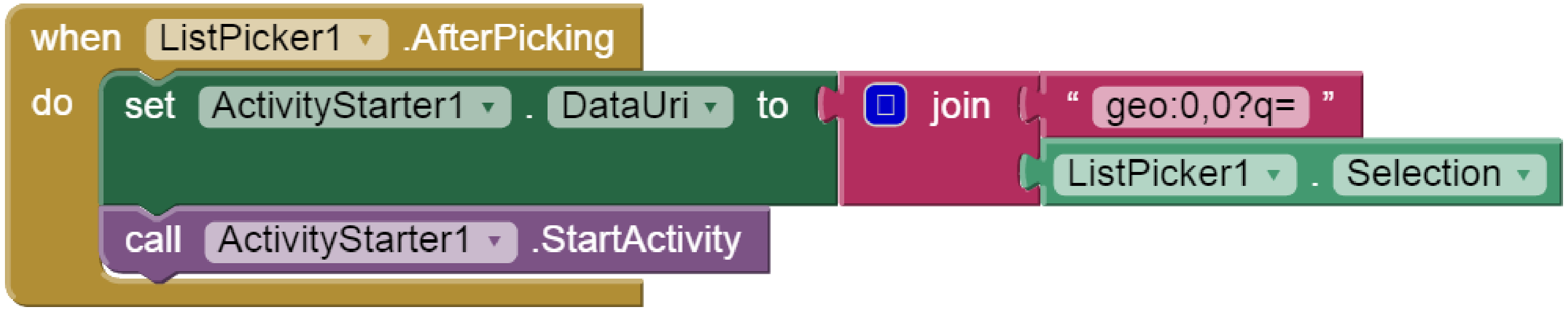
Maps with search for destination

We would proceed to program the behavior after the user has chosen one of the destinations, ie, the ActivityStarter should launch Maps to search for the selected destination.

After the user has chosen from the ListPicker, ListPicker.AfterPicking event will be triggered. In the event-handler for this event, we need to set the DataUri of the ActivityStarter component so it knows which map to open, then we call StartActivity to launch the Maps app.

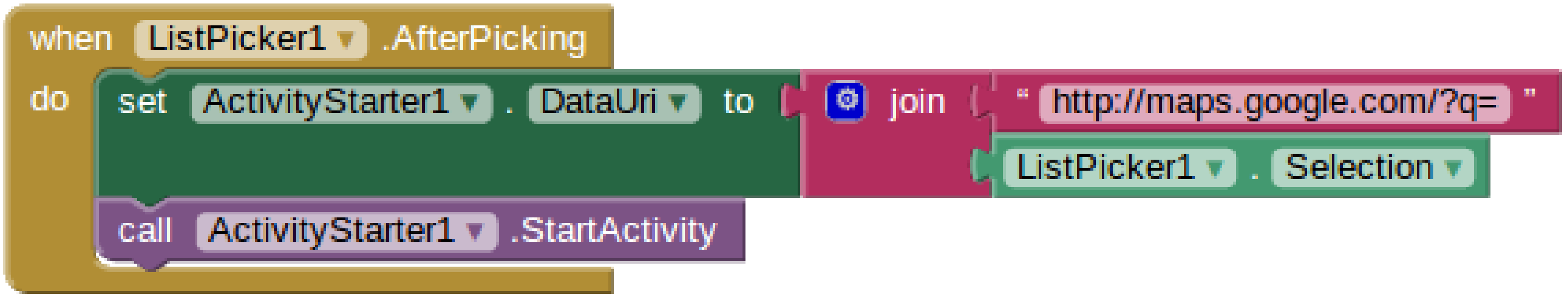
Maps with search for destination

You'll need the following blocks:



```
when ListPicker1 .AfterPicking
do  set ActivityStarter1 . DataUri to join " geo:0,0?q= "
                                           ListPicker1 . Selection
    call ActivityStarter1 .StartActivity
```

Or could also try this link



```
when ListPicker1 .AfterPicking
do  set ActivityStarter1 . DataUri to join " http://maps.google.com/?q= "
                                           ListPicker1 . Selection
    call ActivityStarter1 .StartActivity
```

How the Blocks Work

When the user chooses from the ListPicker , the AfterPicking event is triggered. ListPicker1.Selection property contains the destination user chooses. So if the user chose “Esplanade", that value of ListPicker1.Selection property is “Esplanade”.

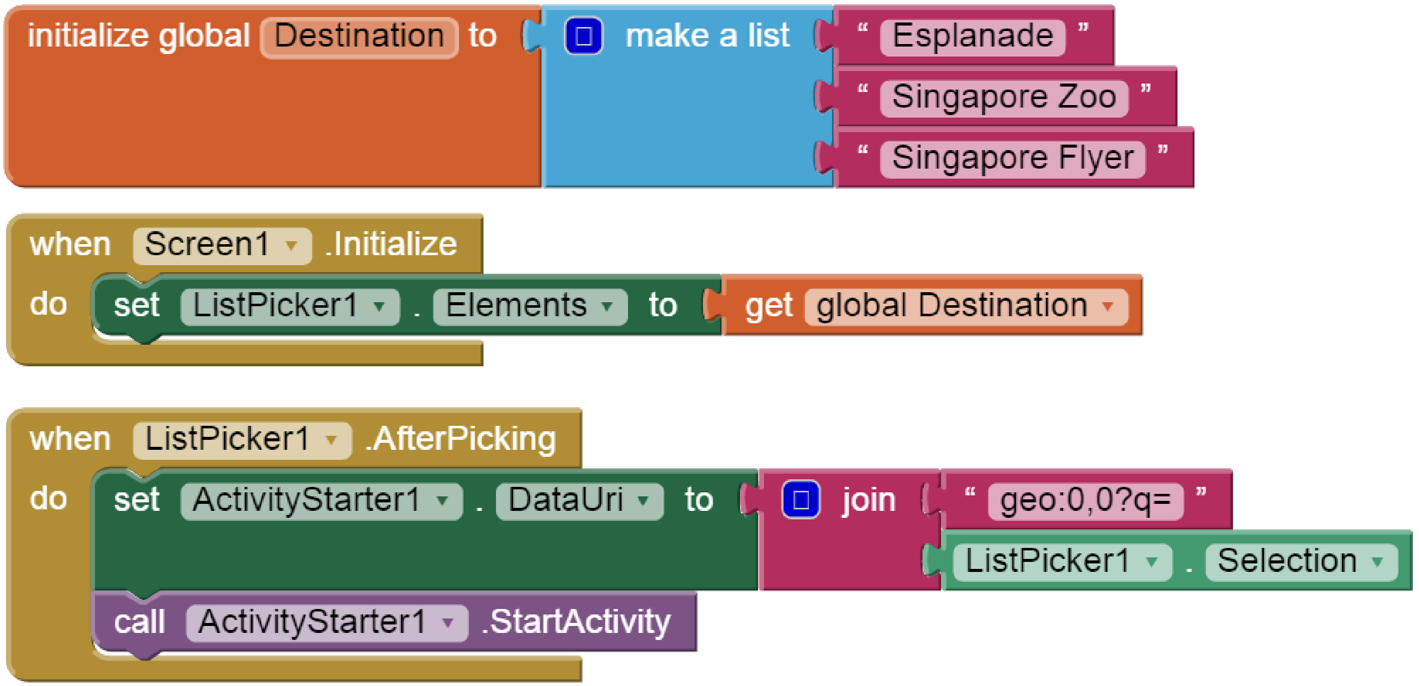
In the Component Designer, you already setup the ActivityStarter componnent with properties so that it will launch the Maps application. Here, you just need to tell it which map to show. The DataUri property of ActivityStarter allows you to specify this using a special protocol.

In this case, you want to show the map that would appear if you typed in “Esplanade" in the search box of the Maps application. To do this, the DataUri should be set to: geo:0,0?q=‘Esplanade'

The make a text block creates a text exactly like this, only the text after "q=" will be the choice made by the user.

Once the DataUri is set, ActivityStarter1.StartActivity launches the Maps app.

Final MapTour App



```
initialize global Destination to make a list "Esplanade"
                                             "Singapore Zoo"
                                             "Singapore Flyer"

when Screen1 .Initialize
do  set ListPicker1 . Elements to get global Destination

when ListPicker1 .AfterPicking
do  set ActivityStarter1 . DataUri to join "geo:0,0?q="
                                           ListPicker1 . Selection
    call ActivityStarter1 .StartActivity
```

Advanced Topic: Web Viewer

A Virtual Tour with the Web Viewer :
The ActivityStarter is an important component because it provides access to any other app on the device. But, there is another way to build a tour guide that uses a different component, instead; the WebViewer. WebViewer is a panel you place directly within your app that behaves like a browser. You can open any web page, including a Google Map, in the viewer, and you can programmatically change the page that appears. Unlike with an ActivityStarter, your user doesn’t ever leave your app, so you don’t have to count on them hitting the back button to get back.

Advanced Topic:

In this second version of the app, you’ll use the WebViewer and you’ll also spice up the app so that it opens some zoomed-in and street views of the Esplanade, Singapore Zoo and Singapore Flyer. You’ll define a second list and use a more complicated scheme to decide which map to show. To begin, you’ll first explore Google Maps to obtain the URLs of some specific maps.

You’ll still try to use the same landmarks for the destinations, but when the user chooses one, you’ll use the *index* (the position in the list) of her choice to select and open a specific zoomed-in or street-view map.

Add the Web Viewer

In the designer, delete the ActivityStarter and image components. Then, from the User Interface drawer, drag in a WebViewer component and place it below the other components.

Uncheck the Screen1.Scrollable property so the WebViewer will display pages correctly.

Finding the URL for Specific Maps

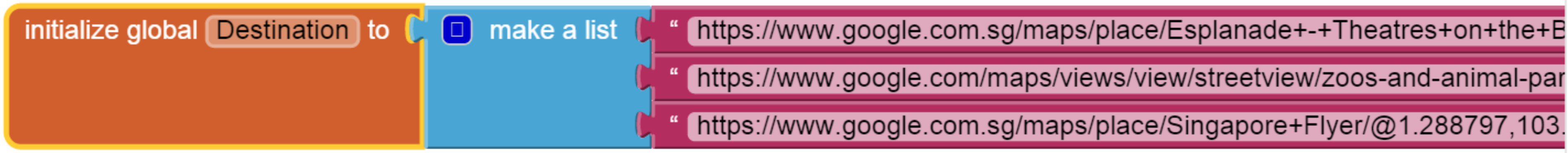
The next step is to open Google Maps on your computer to find the specific maps you
want to launch for each destination:

1. On your computer, browse to *http://maps.google.com*.
2. Search for a landmark (e.g., the Esplanade).
3. Zoom in to the level you desire.
4. Choose the type of view you want (e.g., Street View).
5. Grab the URL. In the classic version of Maps, you click the Link button near the top right of the Maps window and copy the URL for the map. In the newer version of Google Maps you can just grab the URL from the address bar.

Blocks:

The URLs could be too long.

We can try to use bitly services



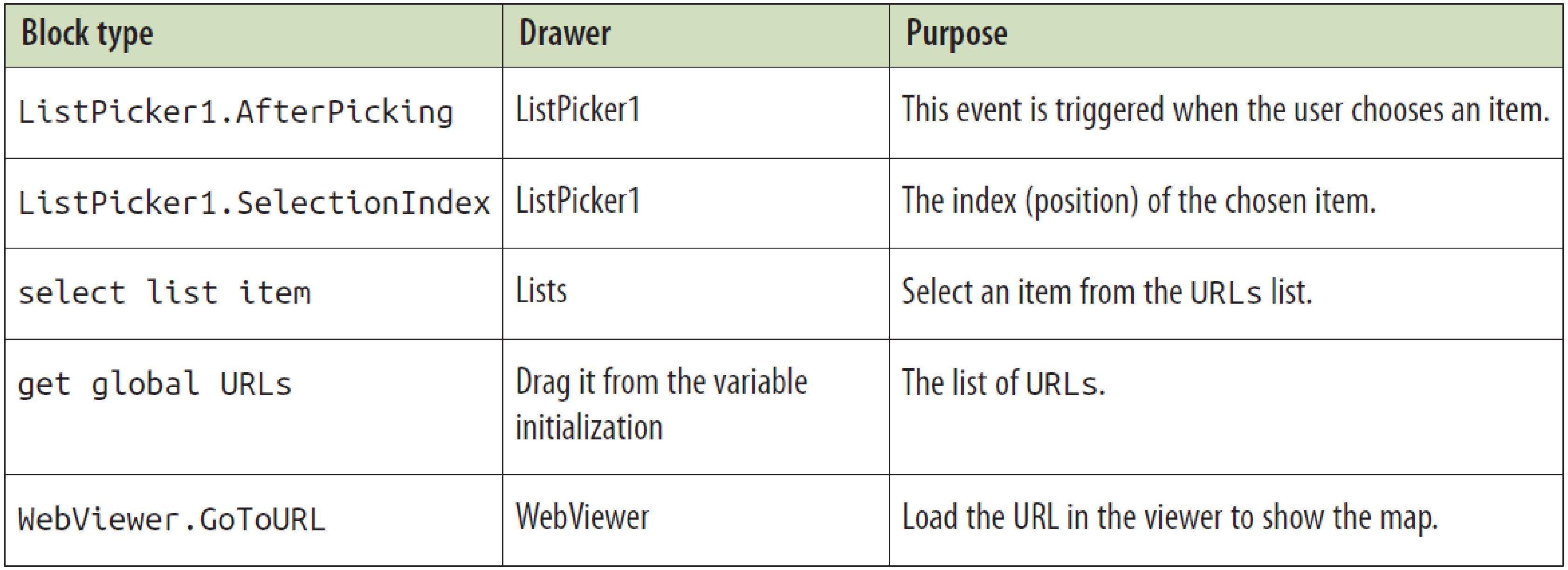
```
initialize global Destination to   make a list   " https://www.google.com.sg/maps/place/Esplanade+-+Theatres+on+the+E
                                                 " https://www.google.com/maps/views/view/streetview/zoos-and-animal-par
                                                 " https://www.google.com.sg/maps/place/Singapore+Flyer/@1.288797,103.
```

Modifying the List Picker Behavior

In the first version of this app, the ListPicker.AfterPicking behavior set the DataUri to a combination of “http://maps.google.com/?q=” and the destination the user chose from the list (e.g., “Tour Eiffel”). In this second version, the AfterPicking behavior must be more sophisticated, because the user is choosing from one list (destinations), but the app is choosing from the URLs list for the URL. Specifically, when the user chooses an item from the ListPicker, you need to know the index of the choice so you can use it to select the correct URL from the list. We’ll explain more about what an index is in a moment, but it helps to set up the blocks first to better illustrate the concept.

There are quite a few blocks required for this functionality, all of which are listed in Table.

Blocks:

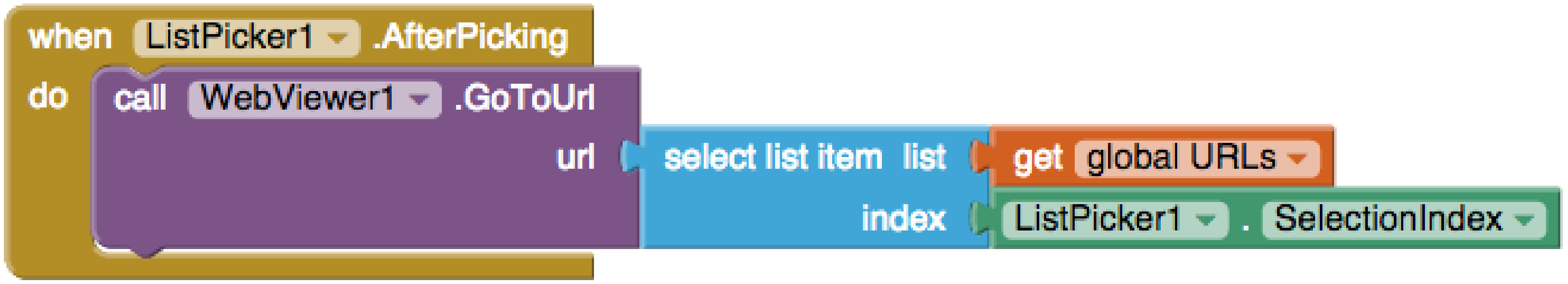


Block type	Drawer	Purpose
`ListPicker1.AfterPicking`	ListPicker1	This event is triggered when the user chooses an item.
`ListPicker1.SelectionIndex`	ListPicker1	The index (position) of the chosen item.
`select list item`	Lists	Select an item from the `URLs` list.
`get global URLs`	Drag it from the variable initialization	The list of `URLs`.
`WebViewer.GoToURL`	WebViewer	Load the URL in the viewer to show the map.

How the Blocks work:

When the user chooses an item from the ListPicker, the AfterPicking event is triggered, as shown in Figure. The chosen item—for example, “Esplanade”—is in List Picker.Selection. You used this property in the first version of this app. However,ListPicker also has a property SelectionIndex, which corresponds to the position of the chosen destination in the list. So, if “Esplanade” is chosen, the SelectionIndex will be 1; if “Singapore Zoo” is chosen, it will be 2; and if “Singapore Flyer” is chosen, it will be 3.

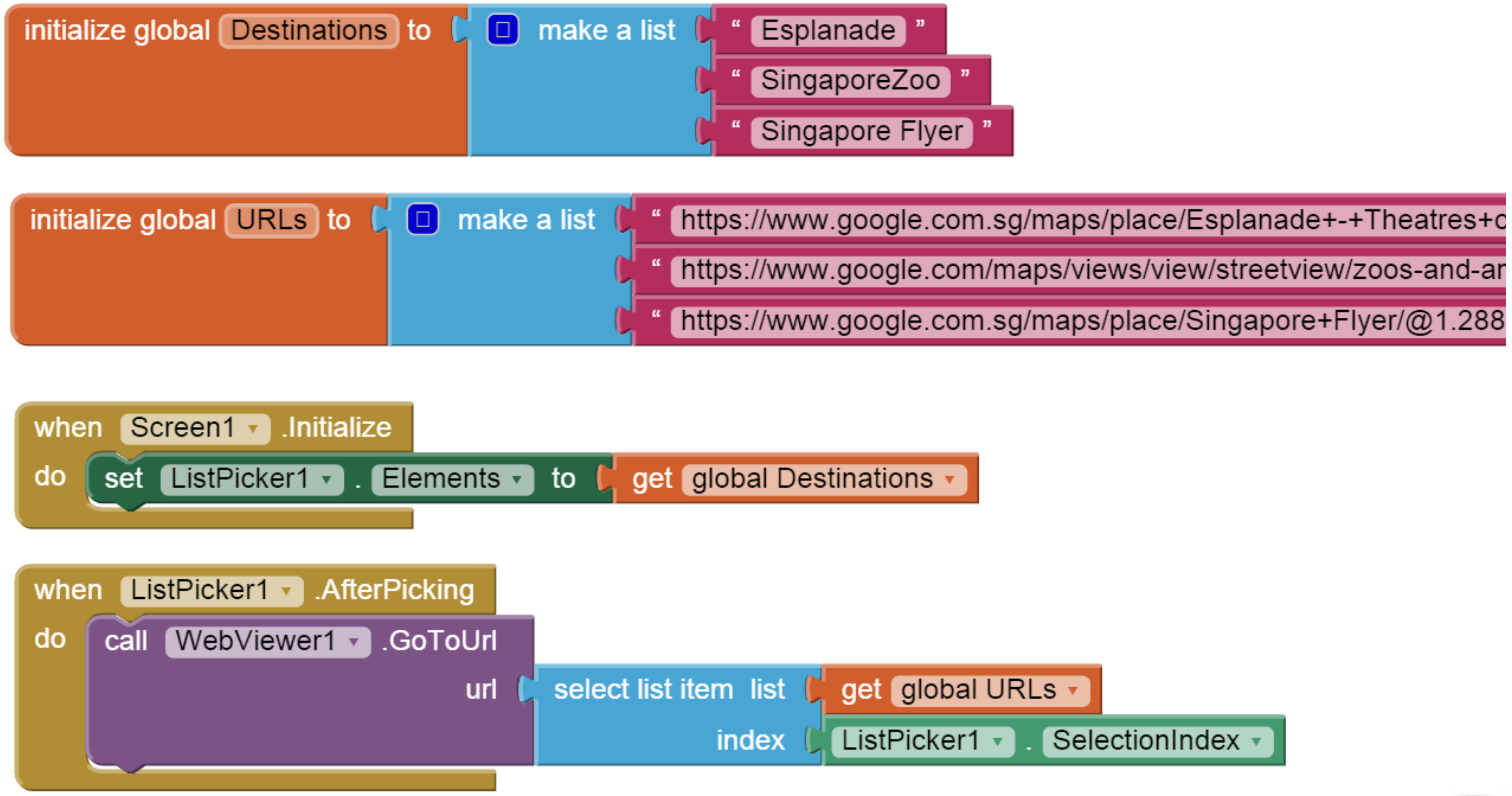
How the Blocks work:



```
when ListPicker1 .AfterPicking
do  call WebViewer1 .GoToUrl
                    url  select list item  list  get global URLs
                                          index  ListPicker1 . SelectionIndex
```

You use ListPicker.SelectionIndex to select an item from the URLs list. This works because the items on the two lists, destinations and URLs, are in sync: the first destination corresponds to the first URL, the second to the second, and the third to the third. So, even though the user chooses an item from one list, you can use their choice (well, the index of their choice) to select the right URL to show.
Test

The Complete App:



```
initialize global Destinations to  make a list  " Esplanade "
                                                " SingaporeZoo "
                                                " Singapore Flyer "

initialize global URLs to  make a list  " https://www.google.com.sg/maps/place/Esplanade+-+Theatres+c
                                        " https://www.google.com/maps/views/view/streetview/zoos-and-ar
                                        " https://www.google.com.sg/maps/place/Singapore+Flyer/@1.288

when Screen1 .Initialize
do  set ListPicker1 . Elements to  get global Destinations

when ListPicker1 .AfterPicking
do  call WebViewer1 .GoToUrl
                         url  select list item  list  get global URLs
                                                index ListPicker1 . SelectionIndex
```

Variations to try:

1. Create a virtual tour of your workplace or school, or for your next vacation
destination.
2. Explore ActivityStarter and use it to send an email or launch an app such as YouTube (see http://bit.ly/1qiFx8Z for help).
3. Difficult: Create a customizable Virtual Tour app that lets a user create a guide for a location of her choice by entering the name of each destination along with the URL of a corresponding map. You’ll need to store the data in a TinyWebDB database and create a Virtual Tour app that works with the entered data.

Summary

Summary

Here are some of the ideas we covered in this chapter:

• You can use list variables to hold data such as map destinations and URLs.

• The ListPicker component lets the user choose from a list of items. The ListPicker's Elements property holds the list, the Selection property holds the selected item, the SelectionIndex holds the position of the selected item, and the AfterPicking event is triggered when the user chooses an item from the list.

• The ActivityStarter component makes it possible for your app to launch other apps. This chapter demonstrated its use with the Google Maps application, but you can launch a browser or any other Android app as well, even another one that you created yourself.

End of Workshop