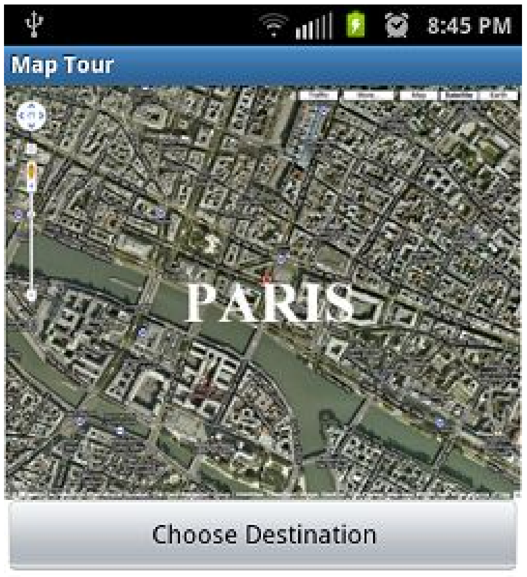
Map Tour Tutorial

The Map Tour allows a user to select a location from the list picker. When a destination is selected, the activity starter component is used to launch Google maps and shows the chosen location on the map. This tutorial introduces how to use a list picker and introduces how to use an activity starter to launch another application outside of the Map Tour app. **Objectives:** In this lesson you will learn to: ● create an app that ○ displays locations on a Google map; ○ uses App Inventor's *List Picker* component to select from a menu of items; ○ uses App Inventor's *Activity Starter* component to start an Android activity.	 ***Click to watch Preview Video***

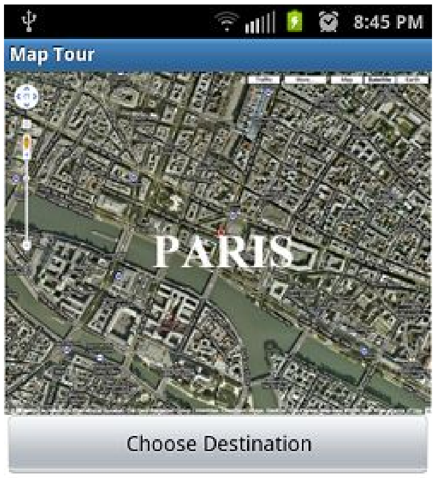
Getting Ready

To get started, open App Inventor with the Map Tour Media Only Template in a separate tab. When the project opens, use the *Save As* option to rename it ***MapTour***. Then follow along with the following tutorial.

Map Tour Tutorial

(Video Tutorial)

The Map Tour UI



The UI for the Map Tour app is very simple and contains only three components: an Image, a *List Picker*, and an *Activity Starter.* The Image is used to display a map of Paris (this is just for show). The *List Picker* is used to let the user choose a location in Paris that they would like to visit. Once the user chooses a destination, the *Activity Starter* opens that location in Google Maps (a separate app).

Adding an Image Component

1. Drag and drop an Image component from the Palette's User Interface category to the Viewer.
2. Change the Image's *Picture* property to the "parismap.png" image which is provided in the template.
3. Set the width to Fill Parent.
4. Set the height to Fill Parent.

Adding a ListPicker Component

The *List Picker* component can be used to select an option from a list of choices. The List Picker appears as a button that, when clicked on, displays a list of texts for the user to choose from. The list can be a list of text or a list of numbers.

1. Drag and Drop a List Picker component from the Palette's User Interface category to the Viewer.
2. Set the List Picker's *Text* property to say "Choose Destination"
3. Set the width to Fill Parent.
4. Set the height to Fill Parent.

To learn more about the List Picker, read about the List Picker in the App Inventor glossary.

Adding an Activity Starter Component

The *Activity Starter* component can be used to launch an application outside of the current app. This can be another application that is included on the device (such as the camera or Google Maps) or another App Inventor app that is installed on the device.

1. Drag and Drop an Activity Starter component from the Palette's Connectivity category to the Viewer.
2. Set the Activity Starter's *Action* property to say "android.intent.action.VIEW" without the quotes. (You may copy and paste.) This sets the action of the activity to be launched.
3. Set the Activity Starter's *Activity Class* property to say "com.google.android.maps.MapsActivity". This sets the class name of the activity to be launched.
4. Set the Activity Starter's *Activity Package* property to say "com.google.android.apps.maps". This sets the package name of the activity to be launched.

To learn more about steps 2-4, read about the Activity Starter in the App Inventor glossary.

Coding the App's Behavior

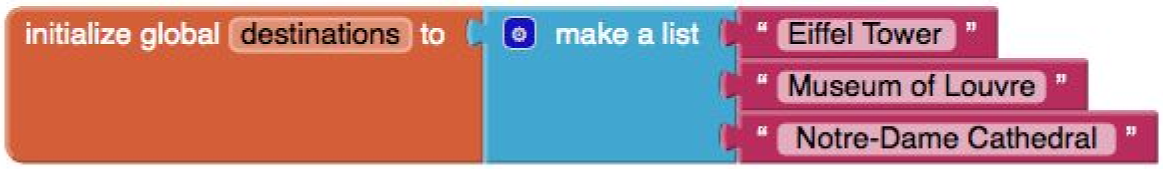
This app uses a List Picker component and an Activity Starter component to open Google Maps and view a desired location on the map. To do this in App Inventor, you can provide the user with a list of locations that they can then choose from. Once a destination is chosen from the list, you can tell the activity starter to show that location in Google Maps.

Creating a List of Destinations

Begin by creating a *list* of places that one might want to visit while in Paris.

1. Initialize a list variable called *destinations* by using an *initialize global variable* block from the Variables drawer and a *make a list* block from the *Lists* drawer in the Toolbox
2. Use *text* blocks to add items to the list. If you need more than two items, remember you can use the mutator (the blue plus sign) to allow there to be more items to the list.I
   a. For example, you could use the following destinations.

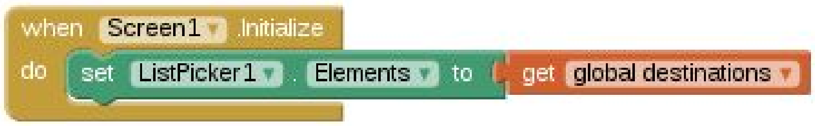
Your *destinations* list should look something like this:



```
initialize global destinations to  make a list  " Eiffel Tower "
                                                " Museum of Louvre "
                                                " Notre-Dame Cathedral "
```

Setting the List Picker's Elements

Now that you have a list of destinations, you'll want to display this list to the user. Here is where you should make use of the List Picker. When the screen initializes, set the List Picker's *elements* to be the global list variable *destinations*. That is, when the app opens, set the List Picker's list of choices to be the list of destinations - this will make each item in *destinations* a choice in the List Picker.
Your code should look like this:



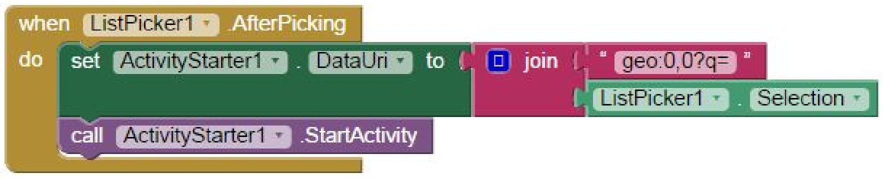
Opening Another Application with the Activity Starter

So far you have created a list of possible destinations and then displayed that list in the List Picker. Now, what happens when the user selects one of the destinations in the List Picker? The Map Tour app should now use the Activity Starter component to open the selected location in Google Maps.

1. Get a *ListPicker1.AfterPicking* event handler from the ListPicker1 drawer in the Toolbox
2. Get a setter block from the ActivityStarter1 drawer in the Toolbox and change the mutator to select the *DataUri* property. **The *DataUri* property is used to specify the URL**.

You would generally use a text block that contains the specific url (e.g. http://www.facebook.com). However, in this case, our url is actually a location and therefore we will use *url encoding* to refer to the select location.

3. Get a *join* text block from the Text drawer in the Toolbox.
4. Get an empty string text block from the Text drawer and type "geo:0,0?q=" without the quotes. (You may copy and paste.) This is the code that is used for *url encoding.*
5. Get a getter block from the ListPicker1 drawer and change the mutator to select the Selection property.
6. Get a call *ActivityStarter1.StartActivity* block from the ActivityStarter1 drawer. This block launches the activity that is specified with the activity starter. In Map Tour, this will open the chosen destination in Google Maps.


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

You can read more here on how to use the Activity Starter to launch other activities.

Run and Test Your App

That's It! You're ready to test your app on your device (phone or tablet) or on the emulator. After you've made sure everything is working properly, try adding some additional destinations of your choice. Your choices should be added to the list so that they can be displayed in the List Picker.

Still Curious?

In this app, you made use of Google Maps, an existing Web application that was created by Google. It is typical of many web applications in that it provides ways that programmers can ***interface*** with it.

An Application Programming Interface or API, is a specification that describes exactly how programs can interact with each other. For example, in this case, the API specified that if you wanted to tell Google Maps how to go to a specific location on a map, you would code it as "geo:0,0?q=Eiffel Tower". In other words, it specified what information you need to provide and in what specific format.

In other words, if you want to interface with Google Maps, you would to follow the specifications set forth in the Google Maps API. The Google Maps API provides documentation for
programmers and app developers for how to interact with its application. The "geo" tag is an example of a ***URL parameter*** and the API contains lots of other parameters for controlling how the map would appear in your app. For example, you can control how far it zooms in and other features. If you want to learn more about the Google Maps ***geo*** API, visit the this online documentation.

So, one interesting implication of this is that because of APIs, programmers see the Web very differently than other users. Rather than seeing it merely as something to view or to search, programmers see the Web as something they can ***control*** by using APIs provided by Google, Amazon, Twitter, and other Web companies.