

Adobe Flash Lite LBS Application Tutorial

Version 1.0; 25 March 2010

Flash

NOKIA

Copyright © 2010 Nokia Corporation. All rights reserved.

Nokia and Forum Nokia are trademarks or registered trademarks of Nokia Corporation. Other product and company names mentioned herein may be trademarks or trade names of their respective owners.

Disclaimer

The information in this document is provided 'as is', with no warranties whatsoever, including any warranty of merchantability, fitness for any particular purpose, or any warranty otherwise arising out of any proposal, specification, or sample. This document is provided for informational purposes only.

Nokia Corporation disclaims all liability, including liability for infringement of any proprietary rights, relating to implementation of information presented in this document. Nokia Corporation does not warrant or represent that such use will not infringe such rights.

Nokia Corporation retains the right to make changes to this document at any time, without notice.

Licence

A licence is hereby granted to download and print a copy of this document for personal use only. No other licence to any other intellectual property rights is granted herein.

Contents

1	Introduction.....	5
1.1	Prerequisites.....	5
1.2	Requirements.....	5
1.3	Flash Lite skills taught.....	5
1.4	Running the applications.....	6
2	About the author	6
3	Open Screen Project Fund.....	6
4	Application description.....	7
4.1	Application screens.....	7
4.1.1	Home screen.....	7
4.1.2	Search results.....	7
4.1.3	About screen.....	8
4.2	Code	8
5	Skinning	11
5.1	Introduction.....	11
5.2	Example skins.....	12
5.3	Component skinning using preferences.....	13
5.4	Content skinning	14
5.5	Button text.....	14
5.6	Backgrounds	14
6	Additional resources.....	14
7	Evaluate this resource	15

Change history

25 March 2010	Version 1.0	Initial document release
---------------	-------------	--------------------------

1 Introduction

This tutorial, which was made possible by the Open Screen Project Fund, is one in a series of seven. Each tutorial in this series is designed around a user experience where Adobe Flash Lite and Nokia devices are a great fit: rich media, personalised experiences, social media, photo sharing, location based information, music, news, video, advertising, and marketing.

The focus of this tutorial is on using the platform integration features of Flash Lite, using an application that provides users with information about nearby coffee shops based on the location of the device. This application is designed to be shown as-is or modified easily. By modifying the application it can be used to quickly present a concept within your company or to pitch to outside clients.

An additional feature of the demonstration application is that it can be reskinned easily: The UI elements and the background can be changed in a matter of minutes.

Please check out the other tutorials in this series, as the concepts taught can be used together in a single application. The other tutorials are:

- Event — Displays information about an event and enables the user to send an SMS message containing text event information or call an event telephone number.
- Video Player — Plays an external *.flv video file in both landscape and portrait modes.
- Music Player — Plays external *.mp3 audio files.
- RSS News — Reads and displays RSS feeds.
- Social Media — Leverages Twitter for promotional purposes.
- Photostream — Displays Flickr photos.

1.1 Prerequisites

These tutorials have been designed for developers with intermediate Flash skills or above.

1.2 Requirements

To modify the applications Adobe Flash CS 3 or CS4 is required. A [S60 5th Edition device by Nokia](#) is recommended to show the demonstrations. Use of a device is more effective than presenting on a computer.

If you don't have a device, you can run the application using the Forum Nokia [Remote Device Access or Virtual Developer Lab](#) services. These services provide access to a range of Nokia devices over the internet.

You can use Adobe Device Central in Flash CS4 also.

1.3 Flash Lite skills taught

This tutorial is designed to teach you the skill to use the following capabilities of Flash Lite on Nokia devices:

- Accessing a device's location using Platform Services.
- Loading Google Maps.
- Initiating a phone call.

1.4 Running the applications

The following methods can be used to run the demonstration applications on a S60 5th Edition device:

- Use a Bluetooth connection to send the *.swf files to your device. The Flash Lite application can then be run by opening it from the device's messaging application.
- Create a new folder called *trusted* in the *other* folder on any drive of your S60 device. Copy the *.swf files to the *trusted* folder. Open the folder in the device's file manager application and tap any *.swf file to run it.

2 About the author

Omega Mobile is a technology-driven design studio that obsesses about mobile and devices. The company is passionate about clean design, simple user interfaces, and targeted experiences. Its specialty is mobile design, user interfaces, user experiences, prototypes, and Flash Lite. Omega Mobile has been producing mobile Flash applications since 1999. For help on a mobile project or for more information, please visit <http://www.omegamobile.com> or contact Omega Mobile at +1-415-596-6342 or contact2010@omegamobile.com.

3 Open Screen Project Fund

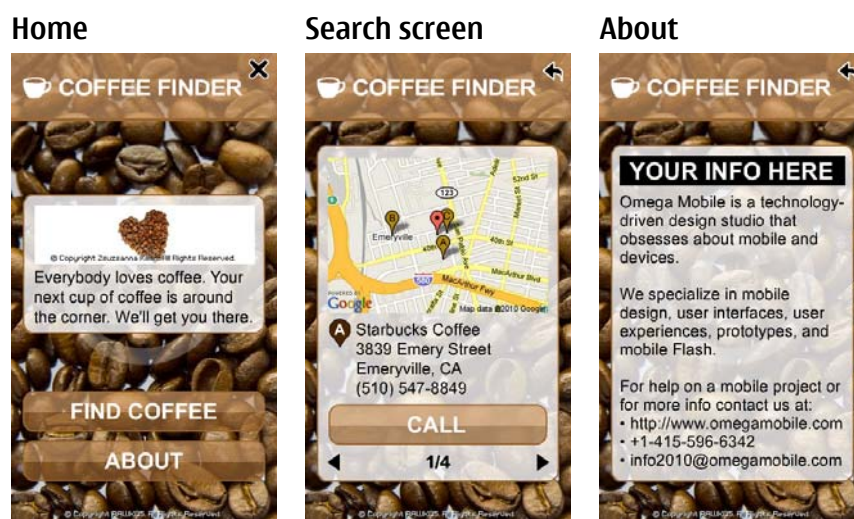
Announced by Nokia and Adobe in 2009, the Open Screen Project Fund awards grants to help developers create exciting new applications and content. Grants are made from a fund of US\$10 million and the fund will remain available until December 2010. The fund also provides marketing and educational support for the Open Screen Project, which aims to establish cross-platform runtimes, remove development and distribution barriers, and innovate through industry collaboration. To learn more about the Fund, visit [Get Started page of the Fund's website](#).

4 Application description

The LBS demonstration application is designed to teach you how to make use of the Platform Services ActionScript APIs offered on S60 devices from Nokia, in this case the APIs associated with determining the device's location. In addition, the application includes code that illustrates how location information can be used to display a Google Map of the user's location and add pins for nearby points of interest.

4.1 Application screens

The application provides the following screens:



4.1.1 Home screen

This screen provides a panel containing an image and text introducing the search feature of the application and enables the user to access the feature of the application through three buttons:

- Find Coffee — Opens the search results screen.
- About — Opens the about screen.
- Quit icon — Exits the application.

4.1.2 Search results

This screen displays the search results showing the nearest coffee shop's location name, address, and phone number. If there is more than one location found a spinner enables the results to be traversed. The spinner includes information on the displayed item number and total number of locations found in the format #/#. The screen includes the following buttons:

- Call — Initiates a phone call using the phone number of the displayed location.
- Left icon — Shows the previous location. If used while the first item is displaying, shows the last item.
- Right icon — Shows the next location. If used while the last item is displaying, shows the first item.
- Back icon — Returns to the home screen.

4.1.3 About screen

The about screen is designed provide a space to describe the application and the company that created it. It is recommended that information about your company, including your contact details, is used when distributing this application to potential clients. The screen contains a back icon button, which takes the user back to the home screen.

4.2 Code

The bulk of the code (see Example 1) dealing with location is used by the search results screen. The code is found in the LBS block.

```

/*
Handles the display of the ui items on the results screen
isResultsShowing:Boolean - true=data is done loading, false=data is
still loading
*/
function displayResults(isResultsShowing){
    marker._visible = isResultsShowing;
    markerInfoTxt._visible = isResultsShowing;
    btn1._visible = isResultsShowing;
    rightArrow._visible = isResultsShowing;
    leftArrow._visible = isResultsShowing;
}

/*
Uses the device sensors to recieve GPS data
*/
function getGPSData(){
    curLocation = new Service("Service.Location", "ILocation");
    var gpsParams = {LocationInformationClass:"GenericLocationInfo"};
    var gpsResults = curLocation.GetLocation(gpsParams);

    mapLoader = new MovieClipLoader();

    //Set default location
    latitude = DEFAULT_LATITUDE;
    longitude = DEFAULT_LONGITUDE;

    var isSucessResults = (gpsResults.ErrorCode == 0);

    if (isSucessResults)
    {
        var sensorResults = gpsResults.ReturnValue;
        latitude = sensorResults.Latitude;
        longitude = sensorResults.Longitude;
    }

    getSearchData();
}

/*
Uses Google maps api to get the map of the current location determined
by the gps with the markers for the locations found.
*/
function getGoogleMap()
{
    textForMarkers = "";
    lettersForMap = ["A", "B", "C", "D", "E", "F"];

    //Goes throught the markers data to created the needed string in the
    url call for the static Google map.
    for(var i = 0; i < markers.length; i++)
    {

```

```

        textForMarkers += "&markers=color:brown|label:" + lettersForMap[i]
+ " |" + markers[i][LOCATION_LAT] + "," + markers[i][LOCATION_LON];
    }

    mapURL =
"http://maps.google.com/maps/api/staticmap?center="+latitude+", "+longitude+
de+"&zoom="+GOOGLE_MAP_ZOOM+"&size="+GOOGLE_MAP_WIDTH + "x" +
GOOGLE_MAP_HEIGHT
+"&markers=color:red|"+latitude+", "+longitude+textForMarkers+"&key="+GOO
GLE_API_KEY+"&format=jpg-baseline&sensor=true";
    mapLoader.loadClip(mapURL, map_mc);
    spinnerOff();
    curLocNum = 0;
    updateMarkerInformation();
    displayResults(true);
}

/*
Uses the Google local search to find coffee shops using the gps data.
*/
function getSearchData()
{
    myLoadVars = new LoadVars();

    myLoadVars.load( GOOGLE_SEARCH_URL+ GOOGLE_SEARCH_TERM+"&sll="+
latitude+", "+longitude);
    myLoadVars.onLoad = function (success)
    {
        results_o = unescape(myLoadVars);
        TITLE_SEARCH_STRING = '"titleNoFormatting":'";
        LAT_SEARCH_STRING = '"lat":'";
        LONG_SEARCH_STRING = '"lng":'";
        ADDRESS_SEARCH_STRING = '"streetAddress":'";
        CITY_SEARCH_STRING = '"city":'";
        REGION_SEARCH_STRING = '"region":'";
        PHONE_SEARCH_STRING = '"number":'";

        markers = new Array();
        while(results_o.indexOf(TITLE_SEARCH_STRING) != -1){

            var locOfLat = results_o.indexOf(LAT_SEARCH_STRING);
            results_o = results_o.substring(locOfLat +
LAT_SEARCH_STRING.length);
            var endOfLat = results_o.indexOf('"');
            var curLat = results_o.substring(0 ,endOfLat );

            var locOfLon = results_o.indexOf(LONG_SEARCH_STRING);
            results_o = results_o.substring(locOfLon +
LONG_SEARCH_STRING.length);
            var endOfLon = results_o.indexOf('"');
            var curLon = results_o.substring(0 ,endOfLon );

            var locOfTitle = results_o.indexOf(TITLE_SEARCH_STRING) ;
            results_o = results_o.substring(locOfTitle +
TITLE_SEARCH_STRING.length);
            var endTitleLoc = results_o.indexOf('"');
            var curTitle = results_o.substring(0,endTitleLoc );

            var locOfAddress = results_o.indexOf(ADDRESS_SEARCH_STRING) ;
            results_o = results_o.substring(locOfAddress +
ADDRESS_SEARCH_STRING.length);
            var endAddressLoc = results_o.indexOf('"');
            var curAddress = results_o.substring(0,endAddressLoc);

            var locOfCity = results_o.indexOf(CITY_SEARCH_STRING) ;
            results_o = results_o.substring(locOfCity +
CITY_SEARCH_STRING.length);

```

```

        var endCityLoc = results_o.indexOf('');
        var curCity = results_o.substring(0,endCityLoc);

        var locOfRegion = results_o.indexOf(REGION_SEARCH_STRING) ;
        results_o = results_o.substring(locOfRegion +
REGION_SEARCH_STRING.length);
        var endRegionLoc = results_o.indexOf('');
        var curRegion = results_o.substring(0,endRegionLoc);

        var locOfPhone = results_o.indexOf(PHONE_SEARCH_STRING) ;
        results_o = results_o.substring(locOfPhone +
PHONE_SEARCH_STRING.length);
        var endPhoneLoc = results_o.indexOf('');
        var curPhoneString = results_o.substring(0,endPhoneLoc);
        var phoneNumsOnly = stripPhoneNum(curPhoneString);
        var titleToShow = cleanUnicodeAmpersand(curTitle);

        markers.push([titleToShow, curLat, curLon, curAddress,curCity,
curRegion, curPhoneString, phoneNumsOnly]);
    }

    totalMarkers = markers.length;
    getGoogleMap();
}

function cleanUnicodeAmpersand(titleToShow)
{
    while(titleToShow.indexOf("u0026") != -1)
    {
        var locOfUnicode = titleToShow.indexOf("u0026") - 1;

        var startReplace = titleToShow.substring(0, locOfUnicode);
        var lastReplace = titleToShow.substring(locOfUnicode+6);

        titleToShow = startReplace + "&" + lastReplace;
    }
    return titleToShow;
}

/*
Removes non-numerical characters from the phone number string
phoneString:String - raw string data recieved from the google search
returns cleaned phone number
*/
function stripPhoneNum(phoneString)
{
    var phoneNumToUse = "";

    for( var i=0; i<phoneString.length; i++)
    {
        var curCharacter = phoneString.charAt(i);
        if(Number(curCharacter) != NaN)
        {
            phoneNumToUse += curCharacter;
        }
    }
    var phoneNum = phoneNumToUse ;
    return phoneNum;
}

/*
Updates the detail information shown on the screen. The name, address,
phone number and marker letter are shown
*/
function updateMarkerInformation()

```

```

{
    marker.markerIndicator.text = lettersForMap[curLocNum];
    markerInfoTxt.html = true;
    markerInfoTxt.htmlText = markers[curLocNum][LOCATION_TITLE] +
"\r"+markers[curLocNum][LOCATION_ADDRESS] + "\r" +
markers[curLocNum][LOCATION_CITY] + ", " +
markers[curLocNum][LOCATION_REGION]
+ "\r"+markers[curLocNum][LOCATION_PHONE_STRING];
    statusTxt.text = (curLocNum+1) + "/" + totalMarkers;
}

/*
Invoked when the call button is pushed and places a phone call with the
number of the marker showing
*/
function placeCall()
{
    getURL("tel:" + markers[curLocNum][LOCATION_PHONE_NUMBER]);
}

/*
Called when the arrow buttons are pushed and sets the new current marker
to show
direction:Number - 1=advancing forward, -1=goes backwards
*/
function switchMarker(direction){
    curLocNum = curLocNum + direction;
    if(curLocNum < 0)
    {
        curLocNum = totalMarkers-1;
    }
    else if(curLocNum > totalMarkers-1)
    {
        curLocNum = 0;
    }
    updateMarkerInformation();
}

```

Example 1: The code employed by the LBS application's search results screen.

5 Skinning

5.1 Introduction

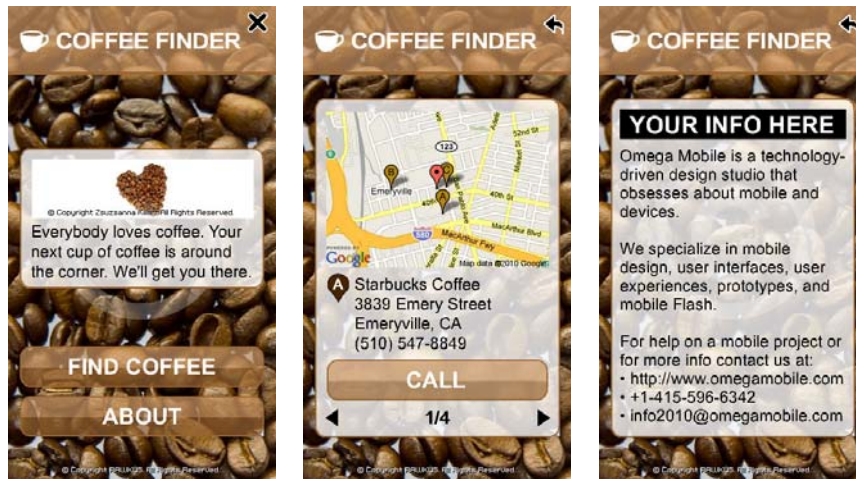
The application illustrates a number of methods that can be used to skin UI elements:

- Buttons, panels, the header, and scrollbars are skinned using preferences set in code.
- The background graphic can be changed by replacing the graphic.
- The text and the graphics used within the application's panels can be changes in code.

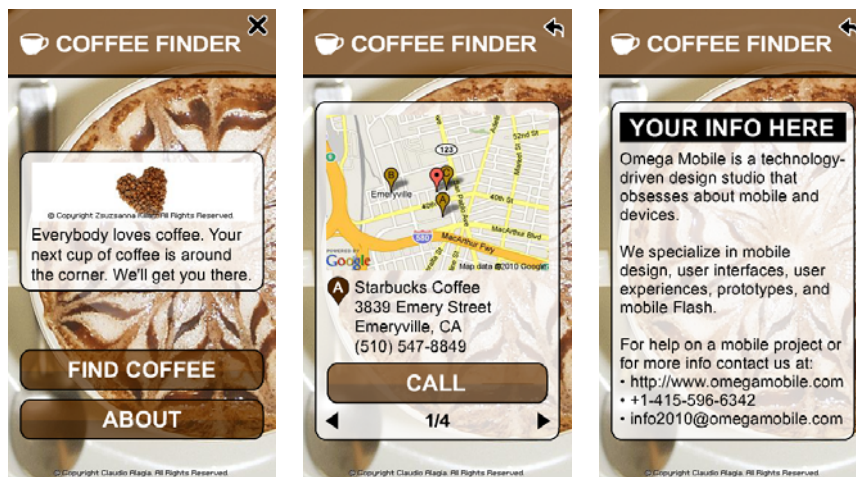
5.2 Example skins

The application has been build using three example skins:

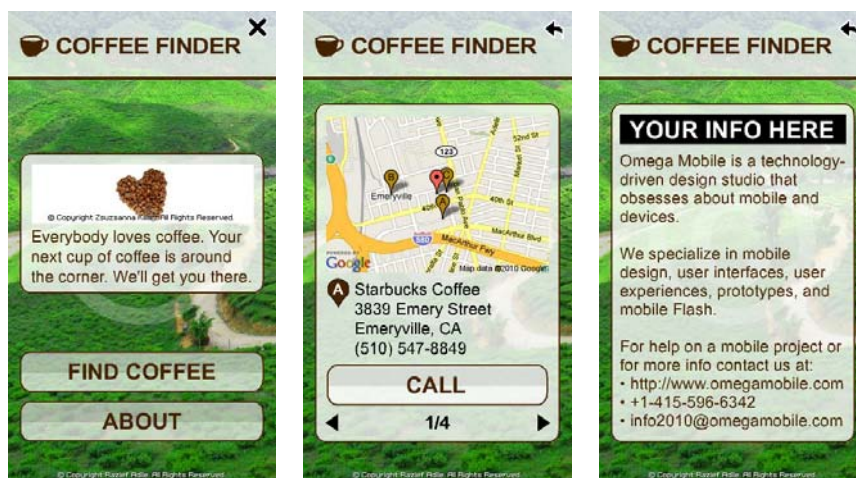
- Skin 1



- Skin 2



- Skin 3



5.3 Component skinning using preferences

The `setPrefs` function (see Example 2) is used to define the application's main skinning preferences. These preferences define:

- Colour and alpha levels for the buttons, panel, and header.
- Map Zoom Level, from 1 to 20; with 1 being the most zoomed out and 20 being the most zoomed in.
- Search term, this can be anything, but is set to 'coffee'.
- Default longitude and latitude, these values are passed to the Google search if a GPS response is unavailable.
- Google API Key, this key is required to make use of Google maps in a third-part application. For more information see [apply for an API key](#).

```
/*
Declare colors for the ui elements and skinning code.
Declare google api and search string. Also, the default longitude and
latitude if gps data is not available
If you're reskinning the application, most of the reskinning can be done
here.
*/
function setPrefs()
{
    //IMPORTANT: YOU MUST GET YOUR OWN API KEY AND INSERT IT HERE TO GET
    THE DEMO TO WORK
    //Apply for your own Google API key here:
    http://code.google.com/apis/maps/signup.html
    //Insert your API Key here
    GOOGLE_API_KEY =
    "insert_your_86_digit_Google_API_key_here_as_a_string";
    GOOGLE_SEARCH_URL =
    "http://ajax.googleapis.com/ajax/services/search/local?v=1.0&q=";

    GOOGLE_MAP_ZOOM = 14;
    GOOGLE_SEARCH_TERM = "Coffee";

    //Alter to your preferred location. If no GPS data is found, it will
    default to this
    DEFAULT_LATITUDE = 37.831423;
    DEFAULT_LONGITUDE = -122.282443;

    /**/
    //SKIN 1
    COLOR_BUTTON_UP_STROKE = 0x391C16;
    COLOR_BUTTON_UP_TEXT = 0xFFFFFFFF;
    COLOR_BUTTON_UP_OVERLAY = 0x663300;
    COLOR_BUTTON_UP_BG = 0x492400;
    COLOR_BUTTON_UP_STROKE_ALPHA = 100;
    COLOR_BUTTON_UP_BG_ALPHA = 70;
    COLOR_BUTTON_UP_OVERLAY_ALPHA = 70;

    COLOR_BUTTON_DOWN_STROKE = 0x391C16;
    COLOR_BUTTON_DOWN_TEXT = 0x000000;
    COLOR_BUTTON_DOWN_OVERLAY = 0xFF9933;
    COLOR_BUTTON_DOWN_BG = 0xFF9933;
    COLOR_BUTTON_DOWN_STROKE_ALPHA = 100;
    COLOR_BUTTON_DOWN_BG_ALPHA = 100;
    COLOR_BUTTON_DOWN_OVERLAY_ALPHA = 100;

    COLOR_PANEL_STROKE = 0x391C16;
    COLOR_PANEL_BG = 0xFFFFFFFF;
    COLOR_PANEL_BG_ALPHA = 80;
    COLOR_PANEL_STROKE_ALPHA = 100;
```

```

COLOR_HEADER_BG = 0xCC9966;
COLOR_HEADER_BG_ALPHA = 70;
COLOR_HEADER_STROKE = 0x996633;
COLOR_HEADER_TEXT = 0xFFFFFFFF;
COLOR_HEADER_ICON = 0xFFFFFFFF;

COLOR_TEXT = 0x000000;

```

Example 2: The code used to set the skinning preferences for the LBS application.

5.4 Content skinning

The text and picture in the main panel of the following screens can be changed by modifying the *.fla file:

- Home screen.
- About screen.

If you plan to modify and distribute this application, it is recommended that you replace the content in the about screen.

5.5 Button text

The text used on any button can be changed by updating the code on the movie clip that contains the button. You will see code similar to this on each movie clip that contains a button:

```

onClipEvent(load){
    txt.btnText.text = "BUTTON TEXT";
}

```

Replace `BUTTON TEXT` with the text you would like displayed on the button.

5.6 Backgrounds

You can choose a different background by:

- Going into the bg_mc movie clip on the main timeline.
- Guiding out the images you don't want.
- Turn off the guides on the image you do want.

You can also import a new background image and place it in the bg_mc movie clip.

6 Additional resources

The [Displaying GPS position using Google Maps images in Flash Lite](#) article from the [Forum Nokia Flash Lite Wiki](#) relate to the code discussed in this tutorial. In addition, the following Forum Nokia resources provide related information:

- [Essential S60: Creating Location-Aware Applications.](#)
- [Flash Lite: Live XML Data Integration Example.](#)

You can also find extensive information on using Flash Lite in Nokia devices in the [Flash Lite Developer's Library](#). There are additional documents and code examples available from the [Documentation](#) section of the Forum Nokia website.

You can also find a wealth of information on the [Flash Lite on Nokia Devices discussion board](#).

7 Evaluate this resource

Please spare a moment to help us improve documentation quality and recognise the resources you find most valuable, by [rating this resource](#).