



HEWLETT-PACKARD DEVELOPMENT COMPANY, L.P.



User Help Guide

HEWLETT-PACKARD DEVELOPMENT COMPANY, L.P.

Mscape User Help Guide

© 2008 Hewlett-Packard Development Company, L.P.
All Rights Reserved

Table of Contents

| | |
|--|-----------|
| <u>Getting Started</u> | 3 |
| <u>Recommended Mobile Devices</u> | 4 |
| <u>Installing the Mscape Player</u> | 8 |
| <u>Downloading and Running Your First Mediascape</u> | 9 |
| <u>Exit the Mscape Player Running on a Device</u> | 11 |
| <u>Make a Simple Mediascape with the Mscape Wizard</u> | 12 |
| <u>Introduction to Mscape Maker</u> | 13 |
| <u>Creating a Mediascape with Mscape Maker</u> | 14 |
| <u>Testing an Mscape</u> | 19 |
| <u>Uploading an Mscape to mscapers.com</u> | 22 |
| <u>Mscape Maker Documentation</u> | 23 |
| <u>Articles</u> | 13 |

Getting Started

A quick introduction to getting Mediascapes running on a device: How to install the software, what devices are supported, playing Mscares, Creating a simple Mscape.

[Recommended Mobile Devices to Run Mscape Player](#)

To play a Mediascapes you need a mobile device that supports GPS. This is a list of such devices recommended for playing Mediascapes.

[Installing the Mscape Player](#)

The Mscape Player should be installed on any device you wish to use to play Mediascape.

[Downloading and Running Your First Mediascape](#)

The Mscape Library allows you to easily install downloaded Mediascapes onto your device

[Exit the Mscape Player Running on a Device](#)

Instructions on quitting the Mscape Player when it is running on a device.

[Make a Simple Mediascape with the Mscape Wizard](#)

You can make a simple Mediascape easily online now using this wizard.

Hardware Requirements

Basically, the Mscape Player runs on a device with Windows Mobile 2003 second edition or newer and GPS and touch screen.

In the Tables below you'll find the list of hardware that we've tested and that we've verified works correctly.






Handheld Devices Recommended By Mscape Users

| MANUFACTURER | DEVICE | IMAGE | NOTES |
|--------------|---------------|---|--|
| HTC | Touch Cruise |  | http://www.htc.com/www/product/touchcruise |
| HTC | TyTn II |  | Note that Static Navigation is enabled by default on this device - this impairs pedestrian GPS accuracy. See this forum topic for details. |
| ASUS | MyPal 696/686 |  | http://en.wikipedia.org/wiki/HTC_TyTN_II |

Handheld Devices Tested By Mscape Users

| MANUFACTURER | DEVICE | IMAGE | NOTES |
|--------------|---|---|--|
| HP | Travel Companion iPAQ rx5900 series iPAQ rx5700 series |  | Our platform of choice. Integrated SiRFstarIII chipset. |

RECOMMENDED MOBILE DEVICES

| MANUFACTURER | DEVICE | IMAGE | NOTES |
|--------------|---|---|---|
| HP | iPAQ hx2400 series |  | Requires a separate GPS device. Works great with the GlobalSat BC-337 Compact Flash GPS (SiRFstarIII?) . |
| HP | Mobile Messenger iPAQ hw6900 series iPAQ hw6500 series |  | Integrated GPS. |
| HP | iPAQ hx4700 |  | Menus look funny due to VGA but works fine. Requires a separate GPS device. Works great with the GlobalSat BC-337 Compact Flash GPS (SiRFstarIII?) . |
| HP | iPAQ rx3700 series |  | Requires a separate GPS device. We use the GlobalSat Bluetooth GPS receivers for this model. |
| Treo | 750 |  | Requires bluetooth GPS unit. Worked great. |

RECOMMENDED MOBILE DEVICES

| | | | |
|---------|-----------------|---|---|
| iMate | JAQ3 |  | Everything worked pretty well. Sometimes the shutdown had problems. |
| Siemens | Loox T830 |  | Requires some fiddling to get the GPS to connect. |
| HTC | p4350 |  | A constant clicking noise from sound card while Mscap player is running. Bluetooth fiddle to set up. |
| HTC | Advantage x7500 |  | Integrated GPS The font is too big so the text of the HELP button on the GPS select screen looks wrong. Also can't see the word "loading" when launching a Mediascapes. BMP on the screen looks strange. |

Supported GPS Units

We have also tested Compact flash and Bluetooth GPS devices from GlobalSat. For these we recommend the models that contain the SiRFstarIII Chipset. We found the SD card one to be less good and the one that combines SD card memory and GPS to be even worse to the point that we have since then avoided them. We have not tested many other GPS manufactures but have been burnt by buying products that default to static navigation and return to that setting every few days.

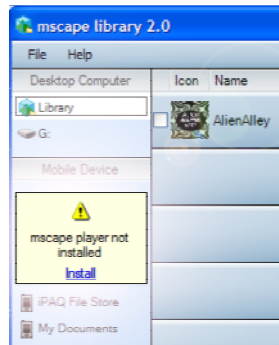
| MANUFACTURER | DEVICE | IMAGE | NOTES |
|--------------|-------------------------|--|--|
| GlobalSat | Compact Flash BC-337 |  | Compact Flash SiRFstarIII GlobalSat BC-337 |
| GlobalSat | Bluetooth BT-338S |  | Bluetooth SiRFstarIII GlobalSat BT-338S |

Installing the Mscape Player

Mscape player is the software that runs on your mobile device that enables you to experience Mediascapes. Before you can install Mscape player you need to have installed Mscape onto your desktop machine. **Download Mscape**

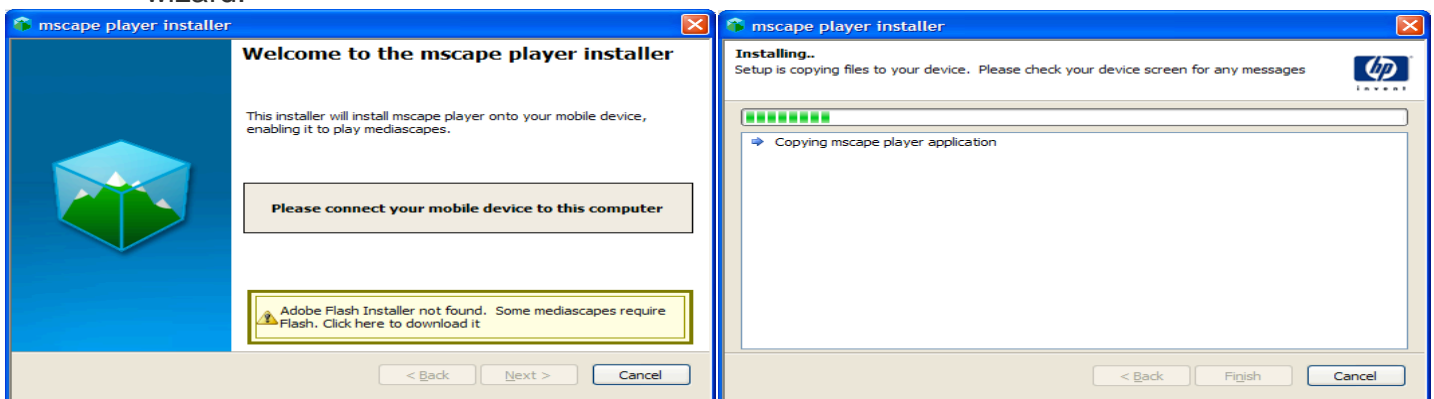
Mscape library is the application that allows you to manage your collection of Mediascapes. It is also used to install Mscape player onto your device - so first we need to start it up.

- On your desktop PC, start up Mscape library. Click *Start Menu / Programs / Mscape* and select **Mscape library**.
- Connect your mobile device to the desktop PC.
- If Mscape player is not already installed on your device you should see a yellow box appear under the *Mobile Device* heading on the left-hand panel. Click **Install** to bring up *Mscape player installer*. If the yellow box does not appear, or you want to force a reinstall, click the *File* menu and select *install Mscape player*.



Many Mediascapes make use of **Adobe Flash** - this includes any Mediascapes that uses video as well as interactive Adobe Flash interfaces. Due to licensing conditions we are not able to distribute the Flash installer with Mscape so you will need to download it yourself. The Mscape player installer will take you through the process of downloading the flash installer if you click the *Adobe Flash Installer not found* button on the installer interface. You only need to complete this process once.

- Ensure that your mobile device is connected and then click next to start the installation wizard.



If the install pauses or appears to have stopped, check your device screen for messages. On some devices various messages and / or security warnings may pop up. If the install fails, there is a manual alternative to the automatic installer. See [Resolving Installations Problems](#)

This document takes you through the process of getting a Mediascape onto your mobile device, so you're ready to experience it in the real world.

Downloading and Running Your First Mediascape

Before We Can Start...

In order to play a Mediascape on your mobile device, you first need to have performed two important steps.

- Mscape should be installed onto your desktop PC.
- Mscape Player should be installed onto your mobile device.
 - [How to Install Mscape Player?](#)
 - [Which Mobile Devices Can I Use?](#)

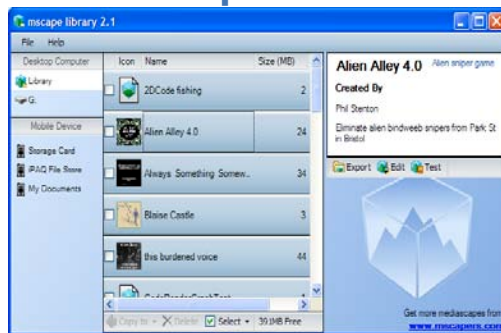
Downloading A Mediascape

Once you have completed these tasks you can use your desktop PC to browse the available Mediascapes on mscapers.com, select one of your choosing and download it. The file that you download is called an **MSZ** file (MediaScape Zipped) - it contains all of the maps, media, and interaction of the Mediascape.

When the download is completed, **double-click the downloaded file** and Mscape library will open and add the new Mediascape to your collection. Mscape library is a program that allows you to manage all of your Mediascapes on your desktop machine and choose which should be copied to a connected mobile device.

**A good Mediascape to start with is “Always Something Somewhere Else” - an audio walk that works anywhere on earth*

Copying the Mediascape to the Mobile Device



The Mscape Library Application

- If Mscape library is not already open, go to *Start Menu / Programs / Mscape* and select *Mscape library* to start it up.
- Connect your mobile device using the ActiveSync cable. The left-hand panel should display a list of folders on your mobile device under the *mobile device* heading.
- Select the Mediascapes you wish to play by checking the box to the left of the icon. You can copy multiple Mediascapes in one go, simply check the boxes of all the Mediascapes that you wish to copy.
- Click the **Copy To** button at the bottom of the screen, and select the destination folder on your mobile device. If your device has an SD card inserted, we recommend you copy to that.

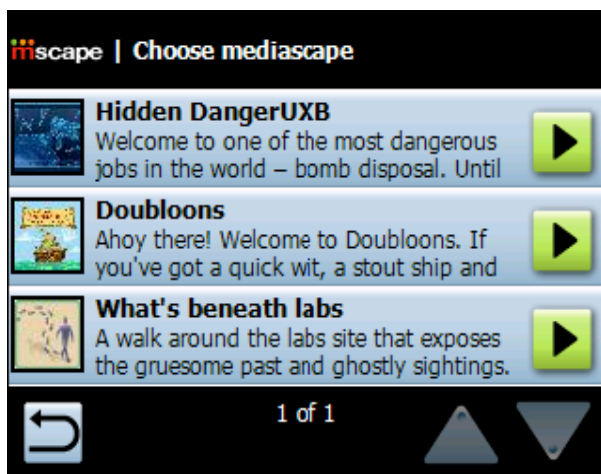
For more detailed instructions on using Mscape library to copy Mediascapes to your mobile device see [Copying Mediascapes to the Mobile Device Using Mscape Library](#)

Running the Mediascape on Mscape Player

Once you have the Mediascape on your mobile device, the next step is to run Mscape player so you can actually experience the Mediascape.

- On your mobile device, tap the **Start Menu** (top left), then select **Programs**, and then select **Mscape Player**.
- Mscape player will begin. The first thing that Mscape player does it to attempt to find the GPS on your device - if it is built-in, like on the iPAQ rx5900 or 610 series models, the program should find the GPS automatically. If not, then you will need to set up the GPS yourself. See [Setting-Up GPS in Mscape Player](#) for more details.

You will then be taken to the main menu of Mscape Player. To load a Mediascape, select **Load Mediascape**. This will show a list of all Mediascapes found on the device. Tap the name of the Mediascape to read detailed information about it, or tap the green 'Play' button to start it up.



the Mscape player load Mediascape screen

Getting A GPS Fix

Most Mediascapes will not start until your GPS is able to track your location - we call the process of getting a location 'getting a fix'.

While a Mediascape is running, there is an icon at the bottom left of the screen that shows the status of the GPS receiver. It will change color depending on the state.



GPS Good



GPS Does Not Have A Fix



GPS Is Not Connected

It can take several minutes for your iPAQ to see enough satellites to get a good GPS fix. It helps if you stand in an open space with a clear view of the sky. Once you get a GPS fix it may take awhile for the readings to settle down they may be quite erratic to begin with.

See [Having Trouble with GPS](#) for help on what to do the GPS cannot be found or has trouble getting a fix.

Exiting the Player

When you have finished playing a Mediascape you can either play another Mediascape or exit the player completely. Both are done using the Mscape player menu.

Click on the mountains icon in the bottom left-hand corner of the screen. This calls up the menu.

The first option on the menu is 'Load Mediascape'

The last option is 'Exit'. (You may have to scroll down using the down arrow button at the bottom of the screen).

Make a Simple Mediascape in Your Browser

You can start building your own Mediascapes on the web right away with the Mscape wizard. Easy and quick templates will let you create your own walks, guides and treasure hunts. Have a good story to tell about your neighborhood? Put it on the map, so that each chapter plays back in the right place.

Introductions to Mscape Maker

Make your own Mediascapes with the Tool-Kit! Learn about Mscape design concepts, the basics of the Mscape Tool-Kit and how to upload an Mscape to the Mscape website.

[Creating a Mediascape with Mscape Maker](#)

An Introduction to creating a Mediascape in the Mscape Maker software, from importing a map, to adding regions and media through to testing it.

[Testing an Mscape](#)

A comprehensive guide to the Mscape Tester and the Pros and Cons of using it as opposed to going out with a device and testing “in the field”.

[Uploading an Mscape to mscapers.com](#)

When you've made a Mediascape, why not share it with the world by uploading it to the Mscape website.

[Mediascape Experience Guidelines \(PDF\)](#)

A detailed document that explains how to design a successful Mediascape.

Creating a Simple Mediascape Using Mscape Maker

This document takes you through the process of building a simple Mediascapes using the tools included within the Mscape package.

Before We Can Start...

Before we can get started creating a Mediascape you need to ensure that you've installed Mscape onto your desktop computer.

Introduction

We are going to create a simple Mediascape that will trigger various pieces of media at real-world locations. Mscape supports audio, images, video, html (web) pages, and Flash Movies. To start off with we're going to keep things simple and just use audio and images.

I've made a package of media files you can use for this tutorial. It's a collection of media borrowed from **Doubloons!** A Mediascape trading game involving Pirates. It's a **zip** file, so you'll need to download it, save it to your desktop, and extract it before we can use the files inside it. In Windows XP you can just right-click the downloaded file, select 'Extract here' and follow the instructions.

- Start up Mscape Maker by going to the Start Menu, then Programs, then Mscape. Click the Mscape Maker Icon.
- Mscape Maker will start up and present you with a blank Mediascape.

Right now this Mediascape won't do anything interesting at all - first up we need to bring in those media files.

Import the Media Files

Before you can use any media in your Mediascape you need to import them into Mscape Maker.

- Click the *Import Audio* button on the toolbar at the top of the Mscape Maker screen
- Navigate to where you downloaded the pirate media files
- Select all the audio files by clicking and dragging out a selection rectangle. Alternatively hold the SHIFT key while clicking each file.
- Click OK, and you'll see the audio files appear in the Mediascape view of Mscape Maker (the left-hand panel).
- Click the *Import Images* button on the toolbar and repeat the process above to bring in the image files.

Importing a Map

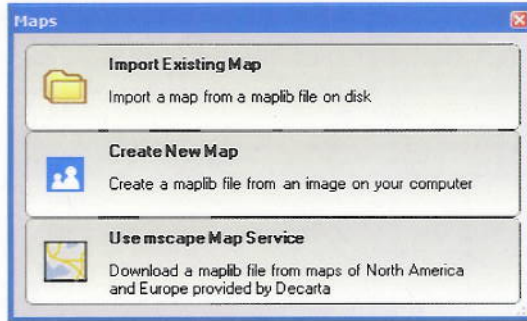
The next thing we need to do is to add a map that represents the area of the world we want our Mediascape to run in. What we recommend you do is to import a map of an area nearby you, so you can go right outside and experience the Mediascape you have created in the real world. Note that you need an internet connection for the following steps to work.

Some Mediascapes are designed to work anywhere in the world - these are known as **portable Mediascapes. A Mediascape that only works in one particular location is known as an **anchored***

CREATING A SIMPLE MEDIASCAPE USING MSCAPE MAKER

Mediascape. In this tutorial we are going to create an anchored Mediascape. See How to Make a Portable Game for more details on portable Mediascapes.

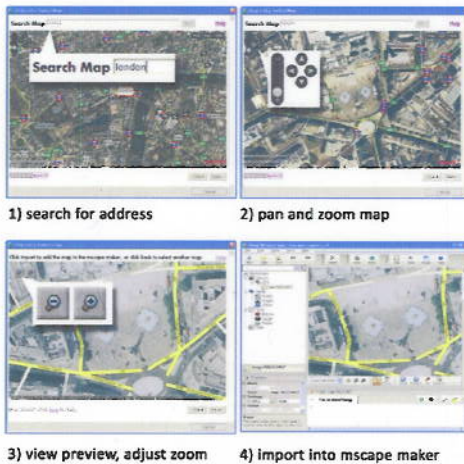
- On the top bar of the Mscape Maker click on the Map button. This will bring up a screen asking where you want to get your map from.



Map Service will not be available after March 31, 2010

Select Map Source

- Choose '**Mscape Map Service**' when prompted for the source of the map. The Mscape Map Service allows you to browse maps using an interface similar to Google Maps. With the click of a button you can capture that map and place it into your Mediascape.



1) search for address

2) pan and zoom map

3) view preview, adjust zoom

4) import into mscape maker

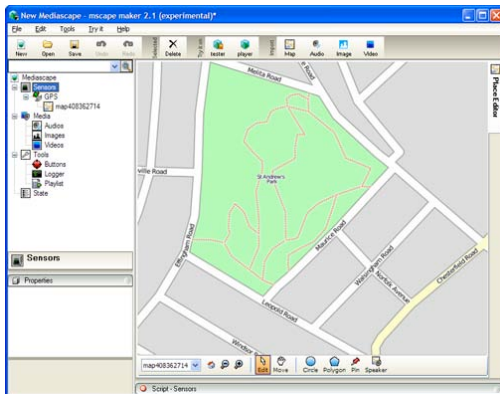
Map Service will not be available after March 31, 2010

How to use the Mscape Map Service

**The map service in Mscape 2.1 currently supports Europe and North America. Mscape 2.2 will have greater coverage as it integrates OpenStreetMap - a community-generated map of the whole world. If you want to create a map elsewhere with Mscape 2.1 see Using Maps in Mediascapes for more help.*

CREATING A SIMPLE MEDIASCAPE USING MSCAPE MAKER

- Once you are happy with the map, click Next to preview the image. The previewed image may look a little different to the image you saw while browsing the map, but this is normal.
- Click Import to bring the map into Mscape Maker.



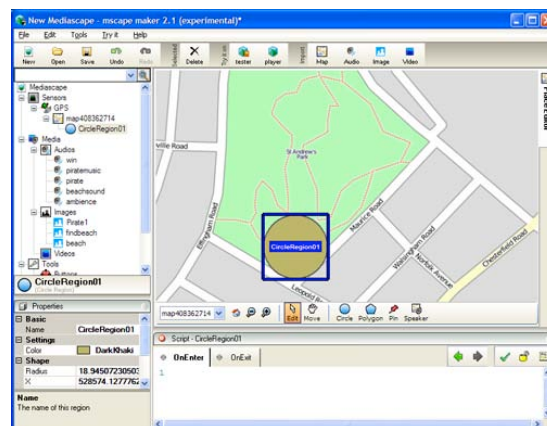
Mscape Maker with the newly-imported map

Add a Region

A *region* is simply an area of the real world that triggers actions when the user moves into it - these areas can be either circular in shape (Circle Region) or of arbitrary shape (Polygon Region). Regions are one of the main methods we have of associating pieces of media with a real-world location. Once you have a map in the main window you will see buttons appear below it.

- Click on 'Create Circle' and you will see a circular region appear in the centre of the main screen.

You can move it around by dragging it, or resize it by dragging the blue border of the shape. When the region is selected you may notice that the *Script* panel shows two tabs called *OnEnter* and *OnExit* respectively. The contents of these tabs describe what will happen when the user moves into the region (*OnEnter*) and out of the region (*OnExit*). These are known as the *events* of the region. Right now both of them are empty - so moving into the region will have no effect

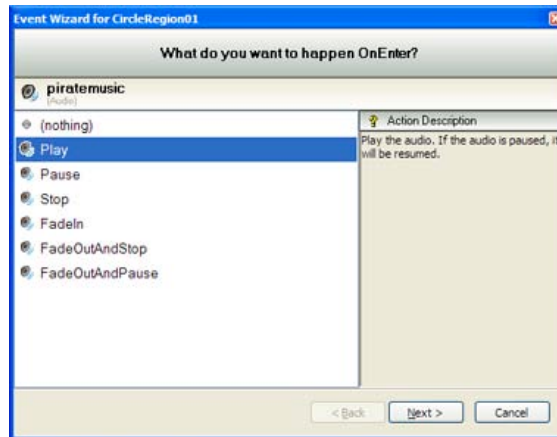


Make The Region Play A Sound

There are two main methods of filling in those *OnEnter* events - you can use simple drag-and-drop actions, or for more advanced users you can type commands directly in using the keyboard. For now we will stick to the simple method.

- Drag the audio object called *piratemusic* from the left-hand bar over to the region and drop it onto the region.

This will make a wizard appear that will ask you how you want the audio object to react to the enter event and the exit event.



The drag-drop wizard

- Choose Play for *OnEnter*, then click Next and choose Stop for *OnExit*
- Click Finish and you will see that the text *piratemusic. Play();* has appeared under *OnEnter* and *piratemusic. Stop();* has appeared under *OnExit*.

*The *piratemusic. Play();* and *piratemusic. Stop();* statements are simple examples of the Mediascape scripting language that is used in Mscape. Once you get to know the scripting language you may find it quicker to type these statements in yourself rather than using the drag-drop wizard. Try it for yourself by deleting *piratemusic. Play();* just as you would in a text editor or Word document. Type in *piratemusic* and then hit the (period) button. A list of options will appear - this is the list of the possible *actions* (play, stop etc) that can be performed on the object called *piratemusic*, and the list of *properties* that can be read. To find out more about the Mediascape scripting language see [The Mediascape Scripting Language: Introduction](#).

Make the Region Show an Image

If you drag and drop an image onto a region, the wizard will appear giving you options to show and hide the image on the *OnEnter* and *OnExit* events.

Drop the image called Pirate1 onto your circular region, and use the wizard to make the image be shown *OnEnter* and hidden *OnExit*.

Create More Regions

- Repeat the previous steps to create more regions, until you have used all of the sample content.

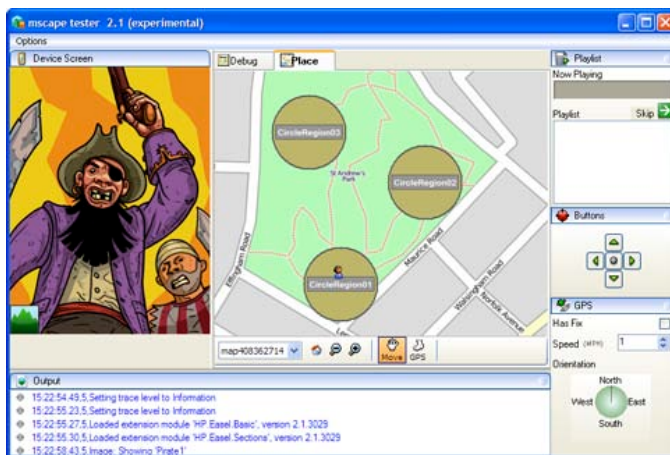
Save Your Mediascape

You can now save your Mediascape by clicking the Save button on the top bar (the little icon of a floppy disk) or choosing Save from within the File menu.

Try the Mediascape Out

Mscape tester is an application that allows you to test out the Mediascape you have created on your desktop PC.

- Start up Mscape tester simply by clicking on the 'try it on - PC' button in the top bar of the Mscape Maker.



The Mscape tester application

When it starts up, on the left you will see a simulation of the screen of the mobile device. On the right is the map with the regions you created. If you click anywhere in the map you will position a little figure at that point. That figure represents the person carrying the mobile device and experiencing the Mediascape.

If you click the figure into the region with the attached audio then you should hear the audio start to play and the images appear.

When you're done testing, click the Close button on the top-right of the window to return to Mscape Maker.

If you have a mobile device linked to the computer you can click on the 'try it on - PDA' button in the top bar of the Mscape maker and the Mediascape will be copied to the mobile device so that you can try it outside.

Remember that it will only work outside if you have built it for the map region where you are based. See the document [Running Your First Mediascape](#) for details on how to run your Mediascape using Mscape Player.

What's Next?

We recommend browsing the help links available on the [Mscape Help Pages](#) to find out more about how to progress from here.

Testing an Mscape

Using the Mscape Tester

You will often want to test a Mediascape out while you are in the process of making it. You can do this easily from within the Mscape Maker. Simply click on the button in the top bar under the 'Try it on' heading, that says 'a PC'. This will start up the Mscape tester with the Mediascape that you are currently editing.

The Basics

Once the Mscape tester is running you will see a window divided up into four main sections. The main area in the centre will be familiar from the Mscape Maker; it simply shows the map of the area and the regions.

To the left of it is a simulation of what will appear on the screen of your mobile device as you navigate through the Mediascape. Also any sounds that are part of your Mediascape will be played through the computer.

If you click somewhere on the map you will place a small figure at that point and on the screen simulation and the sound you will see and hear what they would see and hear at that point in your Mediascape.

By clicking elsewhere on the map you can move the figure to that point and see and hear what would happen if the user moves there.

That's the basics. For more detail read on.

More on the Mscape Tester

Below the map: Viewing Controls

The bars of buttons directly below the map are viewing controls. These are very similar to those in the Mscape Maker.

You can zoom in or out of the map view. Using the hand you can drag the map around to alter your view. Using the foot you can go back to placing your little figure in the map. The home icon resets the map view to the initial viewing position you started off with. Finally selecting a place from the menu of places that you have defined re-centers the map display on that place.

The right-hand screen area: Advanced Simulation

This currently contains two panels both of which are concerned with more advanced aspects of Mediascapes.

A Mediascape that reacts to pressing buttons

The top one simulates the standard button layout that accompanies the screen on many mobile devices. It is possible to make a Mediascape where you can interact with the Mediascape by means of the buttons on the mobile device. This panel of buttons in the Mscape tester enables you to test out such functionality.

A Mediascape that reacts to walking behavior

While the top panel allows you to simulate what happens when the user pushes buttons to interact with your Mediascape, the other panel allows you to simulate how your Mediascape reacts to other user actions connected with the GPS positioning.

When you are writing your script you can make things happen in reaction to the users speed of travel and their direction of travel. These are both simulated in this panel, you can input the speed that your little figure is travelling at and which direction they are travelling in to investigate how your Mediascape reacts.

There is also a check box where you can simulate whether or not the mobile device has got a GPS fix or not. This is not a user action but it is a situation that can easily happen, either when the user switches the mobile device on for the first time or if they lose the GPS signal due to obstructions. (If the mobile device has got a GPS fix it means that it 'knows where it is', i.e. it has picked up signals from the GPS satellites and has worked out where it is on the earth.)

The bottom screen area: Text Output

Finally, at the bottom of the screen is a text area that shows a continual listing of what the Mediascape is doing. This shows error messages and messages from the event scripts that the designer has requested be written there.

If there are any problems with your Mediascape then this is the first place to check. Although much of it is highly technical it will show you if anything unexpected has happened.

Using Mscape Tester on its own

It is also possible for you to run the Mscape Tester on its own without having to run Mscape Maker first. Simply choose 'Mscape Tester' from the 'Mscape' group of programs in the Windows 'Start menu'. Tester will start up and will prompt you to tell it which Mediascape you want to test.

Pros and Cons of Using the Mscape Tester

Using the Mscape tester instead of going out into the field to test Mediascapes has a few advantages and disadvantages.

Faster testing

It is a lot easier than doing the same thing outside in a real Mediascape, you can test things out without having to run around, you can try something in one corner and then test something else in the other corner without having to walk between the two.

Comprehensive testing

Mediascapes can quickly become complex and ensuring that complex Mediascapes do what you expect them to do is difficult if you are testing them out in the field. Using the Mscape tester enables you to rapidly do test routines that would take a long time in the field. For example you could have a media object that is triggered only if the user goes into region A and then region B and back into region A. To test this out in the field would require a great deal of co-ordination and movement. Testing it out in the Mscape tester is quick and simple.

Inaccurate user experience

The experience of a Mediascape is not just the media; it is the combination of media with place. Using the Mscape tester is a good way of evaluating the correct function of the media delivery, but it is difficult to get a feel for the real user experience of the Mediascape. So while it is a good tool for getting a Mediascape working, it is no replacement for actually going out into the field and trying your Mediascape out 'for real'.

Debugging

Sometimes you will be using the Mscape tester to try out a Mediascape you have downloaded before you go outside to use it. In this case the description above is sufficient.

There will be other times when you are using Mscape tester to test a Mediascape that you are in the process of developing. In this situation you may require access to more advanced controls to help with debugging (finding and fixing problems with your scripts).

You may have noticed that above the map in the Mscape tester there are two tabs. The map is shown in the 'Place' tab. The other tab is 'Debug'. This tab gives you access to extra functionality. This functionality is similar to the editor except that you are not really making permanent changes to your Mediascape you are just changing things as it is running to investigate what happens.

This ability allows you to try out 'what if...' scenarios. You can see how the Mediascape reacts if certain parameters are different 'What if we make this audio a looping audio? What if the volume of the first audio is less than the second?' etc.

Not all the properties can be edited; those that are grey are fixed.

Uploading a Mediascape to the Mscapers.com Website

Why Would I Want To Do This?

You would give others a chance to comment on the work that you had produced.

It could help you with your experiments if people give feedback on how you had done what you did.

It would be a chance to share your work with a much wider public.

Even if you are trying out something experimenting what you are doing could be useful to the many others like you who are just starting to get to grips with the Mscape technology, so share whatever you have.

How Do I Do It Then?

Get the Mediascape out of the library

The Mediascape you want to upload is in the Mscape Library.

Open the Mscape Library and click on your Mediascape to select it.

Click on the 'export' button in the right hand panel.

Choose where to put the file, (remember: this just exports a copy of your Mediascape, the original still stays in the library).

Upload it to the site

Click on 'login' on the mscapers.com web site.

Type in your login details (or click on the 'sign up!' link if you have not already registered - don't worry it won't take you long!).

Click on your user name in the top right of the page, this will take you to your personal page.

One of the tools there is a link; 'upload a new Mediascape'.

Click on this and fill out the form that describes your Mediascape, the more information you can supply the better. (For the image you can use one of the images from the Mediascape itself, or design your own poster-style promo image.) There is more about the fields associated with the Mediascape in the section on **Mediascapes on the Mscape Website**.

Now upload it and you are done, it is part of the web site for all to see and use.

Mscape Maker Documentation

This documentation is intended to supplement the “Online Help” in the Mscape Maker Tool-Kit. The latter provides explanations of the properties and events of available objects and the syntax for their use, but may not provide real-world examples of their application.

The intention of this section is to provide conceptual information on the available objects. For syntax please see the online help.

Mediascape

This top level element has two useful events, 'OnLoaded' and 'OnUnloading, which as their names suggest are triggered when an Mscape is loaded and when it is closed. This might, for example, be used to load saved State variables when an Mscape starts or to save them when it ends.

Sensors

Tools to get information from the world.



[GPS](#)



[Rfid Tags](#)



[Wi-Fi Base Station](#)



[Barcode2d](#)



[Bluetooth Devices](#)



[IR Beacons](#)

Media

When you've made a Mediascape, why not share it with the world by uploading it to the Mscape website.



[Audios](#)



[Images](#)



[Videos](#)



[Flash Movies](#)



[Slid-Shows](#)



[Web-Pages](#)

Tools

A detailed document that explains how to design a successful Mediascape.



[Buttons](#)



[Logger](#)



[Playlist](#)



[Timer](#)



[Map Displayer](#)



[Device](#)



[Word List](#)



[State Machine](#)



[Net Connections](#)



[Functions](#)



[HTTP](#)

State

When you've made a Mediascape, why not share it with the world by uploading it to the Mscape website.



[Variables](#)



[Text](#)



[Number](#)



[True / False](#)



[Group](#)



[State List](#)

Global Positioning System (GPS)

Put simply GPS is a system that uses satellites to determine your current position, which in civilian use is most commonly found in car navigation systems. Portable GPS receivers are available and can be found in a range of portable devices, even in some mobile phones. Mediascapes can use GPS to track the user's position and to trigger events when they enter defined locations.

Characteristics

Interaction Model

- The user should establish whether they have a GPS fix. The Mscape Player displays an icon to show the current GPS status. For more information see [Having Trouble with GPS?](#)
- Once a fix is confirmed no further interaction is required. The system will track the user as they walk around; though it should be noted that in some situations the GPS fix can be lost.

Coverage

GPS offers worldwide coverage; however, since it relies on signals from satellites, it will not work inside buildings and can be unreliable in heavily built up areas, particularly where there are many tall structures. As a rule of thumb, the more sky you can see the better and more reliable the signal and the more accurate the resulting location data. In cases where the number of satellite signals are reduced, or are bouncing off buildings, the location data may be subject to 'drift': it may become inaccurate or shift, even when the user is stationary.

Note also that the satellites are in constant motion, so reliability can change depending on the time of day, week or year. As such it is a good idea to test a Mediascape at different times, particularly if some sections are in areas with poor coverage (e.g. built-up areas).

Latency

Once a fix is established position information is updated frequently (usually once per second), so long as the fix is maintained. If the fix is lost, for example when someone enters a building, there will be a slight delay after they exit the building before the fix is re-established.

Initially establishing a fix - for instance after a unit has been switched off for a prolonged period - can take some time whilst the unit gathers data from satellites. Once this data has been gathered a fix can be re-established relatively quickly; for example when the device is switched off and then back on later the same day.

When time is a factor, for instance when a Mediascape is being used by the public, it is recommended that devices are placed in a location where they can gather satellite data well in advance and switched on with the GPS unit active (the simplest way of achieving this is launching the Mscape player software).

Hardware

See the list of [Recommended Hardware](#) for devices that include GPS or for add-on GPS units and also [Setting up GPS on Mscape Player](#).

Using GPS in Mscape Maker

The GPS sensor

The GPS sensor object has events that can be used to determine whether the unit has a GPS fix.

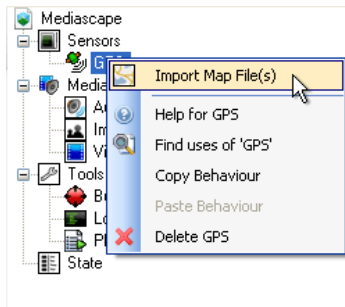
This can be used to ensure that GPS dependent functionality is not started until a fix is established, or to signal a warning when the GPS signal is lost.

The GPS sensor can also provide location information as well as the angle and speed of travel. It is important to understand that these are only updated when the user is in motion. The angle is not necessarily the angle they are facing, but is the angle between the last recorded point and the current point.

GPS and Maps

GPS comes into its own when combined with a map. Once you have [Acquired a Map](#) with associated coordinates you can add regions that use the GPS sensor to trigger events.

Add a map to your Mediascape by right-clicking the GPS sensor and selecting 'Import Map File(s)':



The PlaceEditor will now display your map, and along the bottom you will see controls to move and zoom the map in the editor and to add regions:



Related Articles

[Using Maps in Mediascapes](#)

[Importing a Map into the Mscape Toolkit](#)

[How to Create a Mediascape using a Blank Map](#)

[How to Replay a GPS Trace](#)

[Setting up GPS on Mscape Player](#)

[Simulating GPS on Mscape Player](#)

[Having trouble with GPS](#)

Using Rfid Tags to Trigger Mediascape Action

The **RfidTags** object allows authors to use RFID tags as part of their Mediascape experiences. RFID (Radio-Frequency Identification) tags consist of a chip and a radio aerial, and are small enough to be incorporated into mobile devices or clothing, or stuck onto signs, packaging, or documents. Each tag transmits a code (which we call the ID) that can be used to identify that particular tag without needing physical contact between reader and tag. The tag itself requires no power, as it harvests the radio frequency power of a nearby reader.



Image showing two different types of RFID Tags

In Mscape, reading a tag will trigger events that allow you to play media or trigger other interactions as a result of reading the tag.

Characteristics

Interaction Model

Reading an RFID tag using the system is a deliberate action by the user, as they must place the reader device close to the tag for it to be read. For this reason it is a good idea to try and draw the user's attention to where the tag devices are placed.

Range

The reader needs to be held around 1-2cms from the tag to ensure a successful read.

Latency

The time between the reader being in the correct place relative to the tag, and the action taking place is very short, under half a second.

Hardware

Mscape supports a tag reader manufactured by [ACG](#), which connects to the Compact Flash (or CF for short) port on compatible mobile devices.

[The ACG RFID Reader Compact Flash Device](#)

[Factsheet on the Reader \(PDF\)](#)

If you would like to buy this device, please contact ACG directly.

The [HP iPAQ 210](#) is an example of a currently-available device that this reader works with .

Using RfidTags in Mscape Maker

To add the RfidTags object to your Mediascape:

- Start Mscape Maker
- Right-click on 'Sensors' in the list of Mediascape objects.
- Select 'Add Rfid Tags'

The RfidTags object has a single event, OnTagRead, which carries a single parameter TagID - the identifier of the tag that has just been read. This event is useful when you do not know in advance the IDs of the tags that may be read during the execution of your Mediascape.

Example

This example will play one sound when a tag is read that has an id number that is *even*, and plays another when the id number is *odd*.

This code should go in the RfidTag's OnTagRead event, and requires that audio objects called *audio01* and *audio02* exist in the Mediascape.

```
int id;
int.TryParse(TagID, out id); // convert TagID to an
integer
if ( (id/2)*2 == id) // its even
{
    audio01.Play();
}
else
{
    audio02.Play();
}
```

Using RfidTag Object in Mscape Maker

If you do know in advance the IDs of RFID tags that will be used in your Mediascape, for example if you are deploying a set of known tags in the vicinity, you may find it useful to use the RfidTag object. This object is a child of RfidTags and essentially allows you to associate an OnRead event with a known RFID Tag ID.

- To add an RfidTag object, ensure your Mediascape contains a RfidTags object. If not, follow the instructions under *RfidTags Object* to add it.
- Right-click RfidTags and choose 'Add Rfid Tag'. The new object will be added as a child of RfidTags
- To set the ID of this tag, consult your RFID tag suppliers documentation to find the ID of the tag and enter it in the ID field of the properties box.
- Now when an RFID tag is read, the *OnRead* event will fire on the RfidTag object with the matching ID property.

*The RfidTag object will have no effect if you do not enter a correct ID. Do not make the mistake of renaming the RfidTag to the ID, or entering the ID into the Name property of the tag. The Name property of the tag should be used to remind the designer where the tag is located or what it represents.

Setting-Up Mscape Player to Use RfidTags

RfidTags does not require any specific setup inside Mscape Player, though you should follow the manufacturer's instructions to correctly set up the reader hardware on your mobile device.

WiFiBaseStations

The WiFiBaseStations sensor allows a Mediascape to trigger events when the user moves into range of a Wi-Fi base station or other Wi-Fi enabled device. The most common usage for this object is to provide your Mediascape with a fairly-course grain indoor positioning system.

This object does *not* allow you to actually send or receive data with the WiFiBaseStations themselves, but simply to respond to their presence. To work with networked data use the [HTTP](#) object, found under Tools.

Getting Started

This tutorial will explain how to use WiFiBaseStations as a rough location system.

- The first thing to do is put in place the Wi-Fi base stations to use. If you are going to use the existing wireless infrastructure then this step is not required. Mobile devices can be used provided their Wi-Fi cards can be set to ad-hoc mode.

We recommend using small cheap Wi-Fi routers (if you shop around they can be found for £15-£20). It may be possible to power some of the smaller devices using a battery pack for greater portability and an easier set up. Not that these routers do **not** need to be connected to the internet.

- Next, you'll need to survey the area to discover which base stations found, and especially to make sure any devices you have placed are up and running correctly.

Run the Mediascape and it'll give you a list of all the Wi-Fi base station SSIDs as they are found, along with their MAC addresses. This information is also saved into a text file inside the **_logger** directory within the Mediascape's content folder. Walk around your area until you are satisfied that you have seen all the devices you expected. Make a note of the locations where particular networks became available.

Next, we can create a Mediascape that uses the WiFiBaseStations you have found.

Using the WiFiBaseStations Object

- To add the object, Right-click on Sensors and select 'Add Wi-Fi Base Stations'.

This object holds a list of WiFiBaseStation objects, each of which represents a single Wi-Fi base station (or series of Wi-Fi base stations with the same SSID). You can think of each Wi-Fi base station as being equivalent to a Region on a map - except with Wi-Fi being based around radio, the 'edges' of the region covered will be rather less distinct.

- To add a WiFiBaseStation object, right-click WiFiBaseStations and select 'Add WiFiBaseStation'.

Each WiFiBaseStation object has a series of properties that define how they behave.

| Property Name | Description |
|---------------------|--|
| CombineBaseStations | If true, multiple base stations with the same SSID will count as one |
| MAC | The MAC address of the base station |
| SSID | The SSID used by the base station |
| Threshold | Signal strength required for the base station to count as in range |

This table will help you decide how to set up the properties on your WiFiBaseStation objects.

| Situation | Suggestion |
|--|---|
| A single base station has a unique SSID | Add a WiFiBaseStation object, fill in the SSID, leave MAC blank |
| Multiple overlapping base stations have the same SSID (e.g. a corporate Wi-Fi network) | Add a WiFiBaseStation object, fill in the SSID, leave MAC blank, set CombineBaseStations to true |
| Multiple base stations have the same SSID, but you want each base station to behave separately | Add a WiFiBaseStation object, fill in the SSID, leave MAC blank, set CombineBaseStations to false |
| The base station to use doesn't have a SSID | Add a WiFiBaseStation object, fill in the MAC address, leave the SSID blank |
| You don't know the SSIDs of base stations that the user may encounter | Don't add a WiFiBaseStation object, use the OnFoundBaseStation and OnLostBaseStation events on the WiFiBaseStations object, which are triggered when any base station is found or lost respectively - see below |

Use the MAC addresses and SSIDs recorded using WiFiBaseStationHarvester to correctly fill in SSID and MAC address properties

You can now use the WiFiBaseStation object's OnFound and OnLeave events to trigger behavior when that base station is found or lost.

Changing the Range of a WiFiBaseStation

You can influence the range at which the events will be triggered by setting the Threshold property for the WiFiBaseStation. This is roughly equivalent to changing the size of a circular region, though the exact range in practice will depend on the power of the signal being broadcast and by environmental factors such as walls, and interference. It is a good idea to test this range in situ, bearing in mind that environmental factors can change the result on a daily basis.

The higher the Threshold is the smaller the range of the WiFiBaseStation will be. e.g. 'Excellent' will give a short range, and 'Very Low' will give the widest range, though the coverage may be rather sporadic.

Advanced

Using the WiFiBaseStations OnFoundBaseStation and OnLostBaseStation events

The WiFiBaseStations sensor object can be used to trigger events when any Wi-Fi base stations are found or lost by the wireless driver. If you want to trigger events when particular known base stations (e.g. you know the SSID or MAC address) are found or lost, you should add WiFiBaseStation objects as described above.

- To trigger events when any base station is found use the *OnFoundBaseStation* event. Use the SSID and MAC Parameters to find details of the base station.
- Conversely, use the *OnLostBaseStation* event to trigger events when base stations go out of range. Use the SSID and MAC Parameters to find details of the base station.
- The OnScan event is triggered when a scan is made at intervals specified by the Scan Interval property, though in practice can take longer than the allocated interval depending on factors such as the number of base stations in a given area. The associated parameter 'APsFound' returns a list of all access points found.

Latency

The WiFiBaseStation sensor Scan Interval property - i.e. the rate at which it scans - can be set in the Properties panel. Note that the minimum value is once every 3 seconds, though in some cases environmental factors may lead to the delay being greater. This is most likely to be an issue in areas highly populated with Wi-Fi devices.

Barcodes2d – Using 2d Barcodes to Trigger Mediascape Actions

2d barcodes are similar to the kind of barcodes that you may find on a tin of beans in a supermarket, except that they contain data arranged over two dimensions rather than just the one. This means that a lot more data can be stored in 2d barcode than a traditional 1d version, for instance an entire URL can be embedded easily. The Barcodes2d object allows Mediascapes to include this exciting new technology as part of the experience. Most Windows Mobile 5 or above devices that contain a built-in camera should be compatible with this object.



Example of 2d Barcode



Reading a 2d barcode using an HP iPAQ 6815

Characteristics

Interaction Model

In order to read a 2d barcode as part of a Mediascape, the user will need to activate the code reader. This may be done by pressing a button, tapping the screen, or via another interaction that the designer of the Mediascape has deemed appropriate. The screen of the device will then switch over to a video feed from the camera. The user must aim the camera at the barcode so that it fills enough of the screen for the system to be able to read the code. Once the code has been successfully read, the Mediascape action associated with that code is activated, such as playing some audio or showing an image etc. The system also has the notion of *pop-up text* - this is a string of text that will appear on the code reading screen during the time that the camera is over a code. This is often used as a way of telling the user what will happen if they were to activate the code.

Range

The exact range depends on the physical size of the code, the amount of data that is and the optics of the camera. For a code of around 3 inches square, containing the text 'this is a test' a iPAQ 6815 will need to be around 15-20 cm away.

Latency

The complete process from deciding to read a code to the associated action occurring is a few seconds for a practiced user, and rather longer for a first time user. In general it probably best to avoid designing a system where many codes must be read in quick succession.

Hardware

Provided that your Mediascape device has a built in camera and runs Windows Mobile 5 or above, no extra hardware is required.

Click the link below to view the compatibility chart that lists which mobile devices are compatible with the 2dBarcodes system.

[Mobile Device Compatibility Chart](#)

The 2d Barcodes themselves can be generated and downloaded in a couple of different ways, see *Generating 2d Barcodes* below for details.

Using 2dBarcodes in Mscape Maker

To add to 2d Barcode capability to your Mediascape, you should add the Barcodes2d object.

- Start Mscape Maker
- Right-click on 'Sensors' in the list of Mediascape objects.
- Select 'Add Barcodes 2d'

The Barcodes2d object has two events, *OnCodeRead* and *OnCodeLost* which are triggered when a barcode is read, and a few seconds after a barcode cannot be read any more respectively. The Timeout property on the Barcodes2d object is used to set the exact length of time after a code cannot be seen until the *OnCodeLost* event is triggered. These events are useful when you do not know at the time you are building the Mediascape which barcodes will be encountered during the running of your Mediascape. Both of these events have parameters 'URI' and 'PopupText' which contains the URI and pop-up text embedded in the 2d barcode respectively.

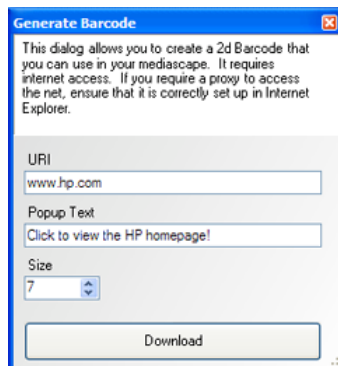
Generating 2d Barcodes

2d Barcodes can be read either from the printed page, or even directly off a computer screen. To generate your own custom 2d barcodes, you can use one of the services that we provide.

Using Mscape Maker (recommended)

Mscape Maker has the capability to generate codes once the 2dBarcodes object has been added.

- Right-click on Barcodes2d on the left-hand panel
- Select Generate Barcode



The generate barcode screen in Mscape Maker

Note that this function connects to our barcode generation system on the internet, so net access is required. If you access the internet through a proxy, ensure that it is set up correctly in Internet Explorer.

- Enter the URI text you require. Note that you can actually use free-form text in any format; you are not limited to URI formats such as <http://www.hp.com>.
- Enter the Popup text you require (if any). This text will appear over the video feed from the camera when the 2d barcode is read. If you intend to immediately display content such as video, images, or web pages when this code is read, the popup text will not be visible. In this case you should leave the Popup text field blank.
- The size field controls the size of the output image. The default of 7 will produce an image about 160x160 pixels, though the exact size depends on the amount of text in the URI and popup text fields.

In general, it is best to have the minimum possible amount of data in the barcode, as more text will result in a denser barcode which is in turn more difficult for the barcode reader software to process.

Using the *ActivePrint* Website

It is possible to generate codes using the [ActivePrint Code Generation Page](#).

- Go to <http://www.activeprint.org/codes.php>
- Enter your popup text into the appropriate field (if popup text is required)
- Choose 'Web' as the link type. Do not use SMS Dial Contact or Data, as these all insert extra hidden data into the code which will not be correctly processed by Mscape.
- Enter your URI. If you are using plain text as opposed to an actual URI, a message will pop about the data type being invalid. You can safely ignore this.
- Choose **Data Matrix** as the Code type. This is very important, as QR codes cannot be read by the Barcodes2d object in Mscape.
- The size field controls the size of the output image. The default of 7 will produce an image about 160x160 pixels, though the exact size depends on the amount of text in the URI and popup text fields.
- Hit Submit to generate the code.
- You can save the code to your computer by right-clicking the image and selecting 'Save Target as' or 'Save Image as'.

Example

This example will show the webpage embedded in a barcode if the URI starts with *http://* and will play an audio if the URI is "playASound".

Place the following code into the *Buttons* object's *OnCenter* event.

```
Barcodes2D.Start();
```

This enables the user to start the barcode reading process by pressing the central cursor button on their device.

This code should go in the Barcodes2d' *OnCodeRead* event, and requires that the WebPages object (right-click on Media and select Add Web Pages) and an audio called myAudio are in the Mediascape.

```

Barcodes2D.Stop();
if (URI.StartsWith("http://"))
{
    WebPages.LoadUrl(URI);
}
else if (URI == "playASound")
{
    myAudio.Play();
}

```

Once a code has been read the barcode reading process is stopped. The user will need to press the central cursor button again to restart it.

Using Barcode2d Object in Mscape Maker

If you do know in advance the contents of the barcodes that will be used in your Mediascape you may find it useful to use the Barcode2d object. This object is a child of Barcodes2d, and essentially allows you to associate an *OnEncountered* and *OnLost* event with a known Bluetooth name.

- To add a Barcode2d, ensure your Mediascape contains a Barcodes2d object. If not, follow the instructions under *Barcodes2d Object* to add it.
- Right-click Barcodes2d and choose 'Add Barcode 2D Device'
- In order to match the new Barcodes2d object with a real 2d barcode in the field, you must type the Uri embedded in the barcode into the Uri property of the Barcodes2d object using the properties box. Note that the URI is case-sensitive.

Unlike most other sensors, the Barcodes2d object **must** be explicitly set to read mode before any barcodes can be read.

To do this, use the *Barcodes2D.Start()*; function. This function is commonly associated with a button on the device, which is achieved using one of the events on the Mediascape's *Buttons* object such as *OnCenter*.

By default, the code reader will stay active once a code has been read. If this is not the behavior you require, you could use the *Barcodes2D.Stop()*; function in the *Barcodes2D.OnCodeRead* event. This will stop the code reader as soon as any code has been read.

Setting Up Mscape Player to use Barcodes2d

Barcodes2d should not require any setup on Mscape player. If your device has a compatible camera, and the *Barcodes2D.Start()*; call has been made in your script then the video feed should appear. If your device does not have a compatible camera, then you may get an error message. No COM PORT setup is required for Barcodes2d.

Fix for Rotated or Inverted Video Feeds

Some devices will display the video feed reversed or rotated. If this happens to you then you can either wait for version 2.2 experimental release, or if you want it fixed right now you can apply the patch below.

- Download the file by right-clicking and selecting 'Save File As' or 'Save Target As'
- Connect your mobile device to your PC and copy the file to \Program Files\mscape\extensions

- Start mscap player, load a mediascape that uses barcodes2d.
- There'll be a 'rotate' button at the bottom right of the screen, click this to get into the settings screen. You can set the rotation of the video feed as well as the vertical flip - these settings will persist between sessions.

Using Bluetooth Names to Trigger Mediascape Actions

The Bluetooth Devices sensor allows any electronic device equipped with Bluetooth to become part of a Mediascape. For example, most consumer mobile phones now have bluetooth capabilities, as well as many laptop PCs and the majority of PDAs. The system is based upon the bluetooth name of the device, which in most cases can be set by the owner of the device. The names of nearby bluetooth-equipped devices can be read by the mobile device running the Mediascape and actions such as media playback can be triggered in response. Exactly which names become triggers, and the action that is taken is entirely up to the Mediascape authors. It is possible to write a Mediascape that will work with any bluetooth device names; you do not necessarily need to know all of the possible names in advance.

Bluetooth Devices only allows a Mediascape to respond to the detection of bluetooth devices, it does not allow any direct networking, messaging, or file transfer between devices.

Characteristics

Interaction Model

- When running in a Mediascape, no further user interaction is required. The user can walk around and media will be triggered when appropriate Bluetooth devices come into or out of range.
- The Bluetooth radio signal propagates in an unidirectional fashion, meaning that the user of a Mediascape does not need to point their device towards a target device to be able to trigger actions.
- A mobile device running a Mediascape using Bluetooth Devices can be seen by other Mediascape devices - e.g. a device can detect other devices and be detected at the same time.

Range

Class 2 Bluetooth devices (such as most mobile phones or PDAs) will have an operating range of around 10 meters, though the exact range will vary between devices and in different locations.

Latency

The major drawback with Bluetooth as a sensing device is that the latency between a target device being in range and your device being able to detect it is quite high. In our tests using an HP rx5935, the time between a target device having its Bluetooth activated and that device becoming accessible was between **7-15 seconds**. This time will vary between devices and will also depend on the number of other Bluetooth devices nearby. We have found that once a bluetooth device has been read once, the time it takes to re-read the same device again is a few seconds shorter. It is also possible to trigger interactions when a Bluetooth device has been 'lost' (*BluetoothDevices.OnDeviceLost* or *BluetoothDevice.OnLost*). This event is fired when a device can no longer be read. In our experiments we have determined that it can take around 45 seconds for a device to register as having been lost.

Notes

It is possible to maintain a connection to other Bluetooth devices such as a Bluetooth GPS while this system is running without any ill effects.

Hardware

A major advantage of using Bluetooth in a Mediascape is that in most cases no extra hardware is required. For the Mediascape playback device, any Bluetooth-enabled Windows Mobile device is sufficient.

The link below will take you to a page that lists which mobile devices are compatible with the Bluetooth Devices system.

[Mobile Device Compatibility Chart](#)

Any Bluetooth device will work as a trigger device, such as a Bluetooth mobile phone, laptop etc. Ensure that the Bluetooth radio is turned on, and use the settings on the device itself to set the *bluetooth name* (or in some cases *device name*) to a name that you will later use in your Mediascape.

For example, if you plan on placing a mobile phone near an entrance to a building you might want to name the device 'Entrance'. This name will be used as the ID property in Mscape Maker.

In the past we have occasionally used use *Blip Systems* 'blip beacon' devices, which are simply a plug in device that broadcasts a fixed bluetooth name. These plug directly into a wall, so are useful for marking a fixed indoor area.



<http://www.blipsystem.com>

April 2008 - Unfortunately it appears that blip systems no longer manufactures this device. What you may want to try in its place is a cheap mobile phone with bluetooth, permanently plugged into the mains via its charger.

Using Bluetooth Devices in Mscape Maker

To add the Bluetooth Devices object to your Mediascape:

- Start Mscape maker
- Right-click on 'Sensors' in the list of Mediascape objects.
- Select 'Add Bluetooth Devices'

The Bluetooth Devices object has two events, *OnDeviceEncountered* and *OnDeviceLost* which are triggered whenever any Bluetooth devices is encountered or lost respectively. These events are useful when you do not know at the time you are building the Mediascape which Bluetooth names will be encountered during the running of your Mediascape. Both of these events have a parameter 'ID' which contains the name of the Bluetooth device that has been read or lost.

Example

This example will play a sound when a nearby device is encountered if the the bluetooth name contains the text 'HP', and shows an image if the name contains the word 'Phone'.

This code should go in the Bluetooth Devices' *OnDeviceEncountered* event, and requires that an audio called myAudio and an image called myImage is in the Mediascape.

```

if (ID.IndexOf("HP") != -1)
{
    myAudio.Play();
}
if (ID.IndexOf("Phone") != -1)
{
    myImage.Show();
}

```

Note that this example is by default case-sensitive. If you want Bluetooth names including 'Hp' 'hP' and 'hp' to play audio, then the first line should be:
if (ID.ToLower().IndexOf("hp") != -1)

Using Experimental Bluetooth Device Object in Mscape Maker

If you do know in advance which Bluetooth names will be used, for example if you are setting up a fixed installation, you may find it useful to use the Experimental Bluetooth Device object. This object is a child of Bluetooth Devices, and essentially allows you to associate an *OnEncountered* and *OnLost* event with a known Bluetooth name.

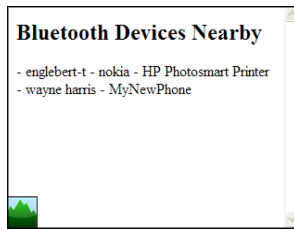
- To add a Experimental Bluetooth Device, ensure your Mediascape contains a Bluetooth Devices object. If not, follow the instructions under *Bluetooth Devices Object* to add it.
- Right-click Bluetooth Devices and choose 'Add Bluetooth Device'
- To set the bluetooth name of this object, enter it in the ID field of the properties box.
- The *OnEncountered* and *OnLost* events will be triggered when a Bluetooth name matching this ID is encountered or lost.

Setting Up Mscape Player to Use Hardware Required for the Bluetooth Devices Object

Bluetooth Devices should not require any setup on Mscape player. If Bluetooth is turned off on your mobile device, Mscape player will turn it on if a Mediascape including the Experimental Bluetooth Device is loaded. This feature may potentially not work on all devices, so if you have problems try manually turning on Bluetooth before loading Mscape player.

Sample Mediascape

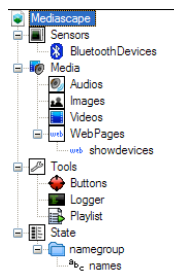
This simple Mediascape lists the names of the bluetooth devices that are nearby using an HTML page, and keeps it updated as devices are discovered or lost. It may also come in handy as a tool for testing the latency and range of your Bluetooth devices.



Screenshot of sample Mediascape

Click the link below to download the mediascape msz file. Double-click the icon when downloaded and it will automatically be installed to Mscape library. Select the Mediascape inside Mscape library and click the 'Edit' button on the right to open it in Mscape Maker.

How It Works



Contents of the sample Mediascape

In addition to the objects which exist in a default Mediascape, this sample contains the Bluetooth Devices and WebPages objects giving access to bluetooth and the web browser respectively. Also there's a 'names' variable inside the state section, whose value contains a list of the currently-found names (separated by - symbols and spaces). This is contained within a StateGroup (a collection of variables) called 'namegroup'. This is so we can use the handy WebPages.ShowWithStateGroup method - see later. A simple HTML page was created using the standard Mediascape JavaScript library that shows the value of the names variable. See the document [How do I show the user information using an HTML page?](#) For more details.

When the Bluetooth Devices object triggers a OnDeviceEncountered event, the list is checked to see if the new bluetooth name (held in the event's ID parameter) already exists, if not the new name is added to the 'names' variable.

```
if (names.Value.IndexOf(ID) == -1)
{
names.Value += " - " + ID;
}
showdevices.ShowWithStateGroup(namegroup);
```

When the OnDeviceLost event is triggered, the name is removed from it.

```
names.Value = names.Value.Replace(" - " + ID, "");
showdevices.ShowWithStateGroup(namegroup);
```

In the case of both events, the 'showdevices' web page is shown, passing in the current value of 'names' using the 'namegroup' group in the state section.

IRBeacons – Using Infra-Red Beacons to Trigger Mediascape Actions

IR Beacons allow your Mediascape device to pick up Infra-Red signals from IR Beacons and convert those into Mediascape actions such as the playback of media. Many mobile devices have infra-red capability meaning that in those cases no extra hardware is required for the receipt of IR messages. However the beacons themselves will need to be purchased (see hardware section below).

Characteristics

Interaction Model

Infra-red requires line-of-sight between transmitter and receiver, meaning that in many cases reading a IR beacon is a deliberate action on the part of the user. If this is not the intended use, it can be possible to fill a room with an IR beam by setting the beacon's beam to full power and placing the beacon high in the room. Provided the mobile device is held away from the body and the IR receiver is not covered the signal can be received without user input.

Range

The range of an IR Beacon depends on the setup of the beacon itself, and the type of IR receiver built into the mobile device. In general the range is between 2 and 15 meters, with the average about 10 meters.

Latency

Again, this depends on the settings of the IR beacon itself. We usually recommend that the IR beacons are set to pulse every second, which results in the latency between being in range of a beacon and the Mediascape being notified is around a second.

Hardware

As previously mentioned, the Mediascape playback device does not need any extra hardware provided that an IR receiver is built into the device.

The link below will take you to a page that lists which mobile devices are compatible with the IRBeacons system.

[Mobile Device Compatibility Chart](#)

For the IR Beacons themselves, we use [Lesswire's IrDA beacons](#).



an old style IR Beacon from Lesswire

Unfortunately, Lesswire no longer manufacture an all-in-one solution as seen above, where batteries and a case are included. The newer version of the product is simply the board and the buyer themselves will need to add a case and the battery.

We have written a guide to setting up Lesswire's beacons to ensure compatibility with Mscape.

[Programming the Lesswire IrDA Beacons](#)

Using IR Beacons in Mscape Maker

To add the IR Beacons object to your Mediascape:

- Start Mscape Maker
- Right-click on 'Sensors' in the list of Mediascape objects.
- Select 'Add IR Beacons'

The IRBeacons object has a single event *OnTagRead*. This event is triggered whenever a pulse from any nearby IR Beacons is read. There is a parameter *id* to the event which carries the name of the beacon. In general it is not recommended to use the *OnTagRead* event, but instead to use the *OnEncountered* and *OnLost* events of the IRBeacon objects.

Using the IRBeacon Objects in Mscape Maker

The IRBeacon object is a child of IRBeacons and essentially allows you to associate an *OnEncountered* and *OnLost* event with a known IR Beacon name.

- To add a IRBeacon, ensure your Mediascape contains a IRBeacons object. If not, follow the instructions under *IRBeacons Object* to add it.
- Right-click IRBeacons and choose 'Add IR Beacon'
- To set the name of the beacon, enter it in the ID field of the properties box.
- The *OnEncountered* and *OnLost* events will be triggered when a IR beacon with a name matching this ID is encountered or lost.

Setting Up Mscape Player to Use IRBeacons.

The first thing to do is to disable *Receive all incoming beams*. This is very important as IR Beacons will not work if this option is active.

- On the mobile device, go to Start / Settings / Connections / Beam
- Uncheck the box labeled *Receive all incoming beams*

In general, any Windows Mobile device that has an IR port should be compatible. However, before IRBeacons will work correctly you need to tell Mscape player which COM PORT the IR port is on.

- Load Mscape player. If the Load Mediascape screen is showing hit the back arrow to go the main menu.
- In the main menu, select *Sensor setup*
- A list of all of the currently-available sensors that require COM PORTs is shown.
- Sensors that are not set up will have an [!] icon shown next to them.
- To set up IRBeacons, click its entry on the list.

The exact name and number of the port to use will vary between devices, but generally for IRBeacons you should choose one labeled IR, IrCOMM or similar, but *not* one labeled IrDA.

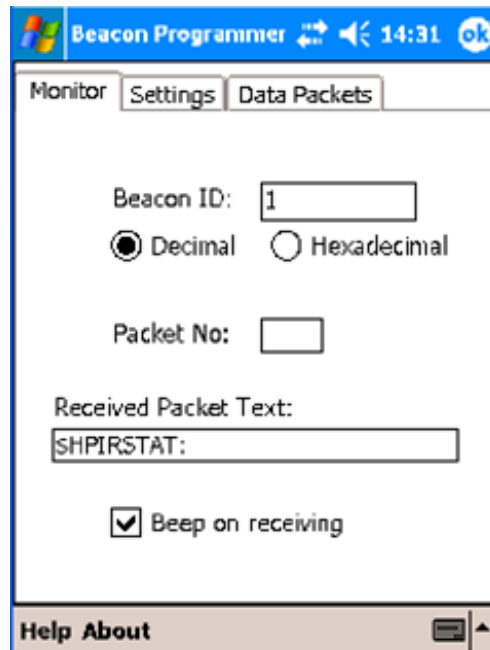
Programming the Lesswire IrDA Beacons

<http://www.lesswire.com/> - company that manufactures the IrDA? beacons

The Lesswire beacons must be correctly configured for mscape player to be able to read them. For example, the IR Beacon must be set to transmit a name in a specific format. This document takes you through the process of performing this configuration.

- The beacon programming program must be on the iPAQ. Contact Lesswire for the CAB package.

Start the Beacon Programmer



To see what the name of the beacon is currently set to

- Go to the *Monitor* tab
- Turn Beep on receiving ON (Check box). Should hear beeps when iPAQ points at beacon - if not check beacon is ON.
- The current beacon name should display in the Received Packet Text

Setting Up the Beacon

To ensure that the beacon will work with the mscape software you need to set up a series of basic settings.

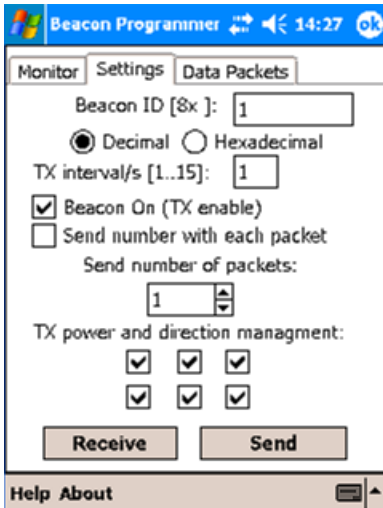
- Go to the *Settings* Tab.

Use the following settings:

- Beacon ID should be Decimal and 1 (not 0)
- TX intervals should be 1. This is the number of seconds between each flash of the beacon.
- Beacon On (TX enable) check box should be ON (checked). If this is disabled the Beacon's green light will not flash and it'll appear to be dead.
- Send number with each packet should be OFF (unchecked)
- Send number of packets should be 1 - this is important!

You can control the strength and direction of the beacon by checking / unchecking the six boxes under 'TX power and direction management'.

- These should be set up required for your purposes.
- For limiting the width of the area covered by the IR beam set the center squares on only. One box set is sufficient for close range, or two boxes for a longer distance (approximately 6-10 meters).



This image shows the correct settings for the settings tab.

Once you have set up the settings as required you need to send the data to the beacon. *Hold the device to the beacon and press 'Send' If the Send succeeds the beacons Green and Red lights should flash quickly.

Sometimes the data sent to the beacon can get corrupted so its worth reading back the data to check it's set up correctly.

- To check the settings on this tab are correct hold the device press 'Receive' and verify if the setup is correct.

To Program The Name

The name of the IR beacon is the name that will be used inside of mscapemaker to identify this IR beacon.

Go to the *Data Packets* tab

- Set Packet number to 1 for programming a beacon name (Should not be 0)
- Set Packet Text to the required name.

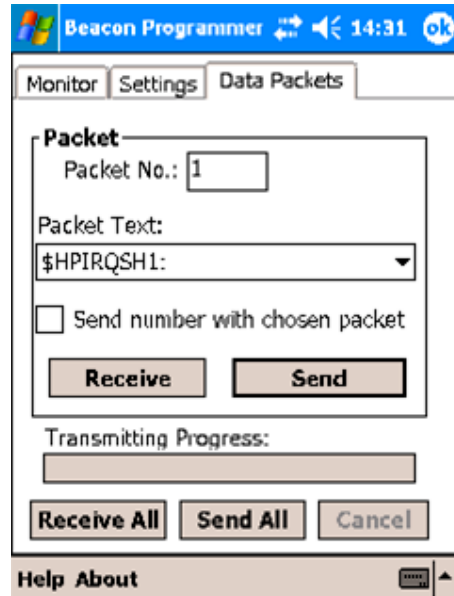
A name MUST start with \$ have 8 characters and end with a colon eg \$HPIRCH02:

- Uncheck box 'Send number with chosen packet'

Next you need to *send* the updated information to the beacon.

- Hold the iPAQ close to the beacon approx 5-10cm away
- *Tip* Lay beacon on its back so that you can see the lights flashing when you point the iPAQ at it.
- Press 'Send' - Both the red and green lights on the beacon should flash quickly
- Check that the right name was set by looking at the name on the Received Packet Text in Monitor tab
- If it is not correct then repeat the previous steps.

- Beacon On (TX enable) check box should be ON (checked)
- Send number with each packet should be OFF (unchecked)
- Send number of packets should be 1
- TX power and direction management boxes should be checked according to the set up required. For limiting the direction of the beam just set the center squares on, one box is sufficient for close range & two boxes for a longer distance (approximately 6 meters).
- Press Send and check the name again either from the Monitor tab or press Receive in the Settings tab to get the current settings.
- Repeat the 'Program the name' process and checking all the setting are correct until it works!



Turning the IR Beacon On

- You need to open up the IR beacon to turn it on. * *Tip* : The catches are about 2cm from the top and botton. An iPAQ stylus can be used to prize the case open by wedging them in at these points.
- Once the beacon is open there should be a pair of red and black wires attached at one end to the battery and the other end to a green connector. The green connector needs to be plugged into the two pins at the left hand side of the beacon. The RED wire must be at the bottom (plugged into the + point) and the balck at the top (- point). Once this is plugged in the Green and Red lights should start flashing. If not check the cables on the wire have not come loose and it is firmly plugged in. *Replace the front.
- If the battery is charged run it off battery.
- To charge the battery or run the beacon off the mains plug the power supply in at the top.


When the IR beacon is on you should see a Green and Red light flash occasionally in the top right of the dark face.

Mobile Device Sensor Compatibility

This page shows which sensor types are compatible with various Windows Mobile devices.

 Fully Working
  Not Available
  /  etc. partially working, see notes below table

| Device Name | Bluetooth | 2d Barcodes | RFID Tags | IR Beacons | RFBeacons (Obsolete) |
|-------------------|-----------|---------------------|-----------|------------|----------------------|
| HTC SP5pm | | (use flip vertical) | | | |
| HTC Touch HD | | (use flip vertical) | | | |
| HTC Tytn II | | (use flip vertical) | | | |
| Samsung Omnia | | (use flip vertical) | | | |
| Pocket Loox | | (use flip vertical) | | | |
| Vodafone 1520 | | (use flip vertical) | | | |
| HTC Touch Cruise | | | | | |
| HP iPAQ 910 | | (use flip vertical) | | | |
| HP iPAQ 6815 | | | | | |
| HP iPAQ 2490 | | | | | |
| HP iPAQ 5935 | | | | | |
| HP iPAQ 6915/6965 | | | | | |
| HP iPAQ 210 | | | | | |
| HP iPAQ 110 | | | | | |
| HP iPAQ 610 | | *1 | | | |

 - 2d barcodes work in conjunction with an [HP Photosmart Mobile Camera](#)

 - May be possible with a custom lead - this device uses a newer serial port connector

*1 The iPAQ 610 displays the video feed cut into two and vertically stretched due to an issue with it's camera driver.

If you have tried out the system on a device not listed in the above table please let us know by posting a message in the [mscape experimental forum](#) and we will update the table above.

Using Sound in Mediascapes

Sound is one of the most important media types in a Mediascape, it is the one that most people make use of, it can be the most atmospheric and can engage with people without distracting them.

When you record a sound and save it on your computer, it is stored there as an audio file. Using the Mscape maker you can import one of these audio files and when it is brought into the Mscape maker it becomes an **audio object**.

[Which audio file formats should I use?](#)

Using Fades

When you use the standard Play and Stop functions the audio will start and stop abruptly, as if you'd pressed the start and stop buttons on a music player. In some situations you may want the sound to start and stop more gently for a smoother effect. The way to achieve this effect is to use the fade functions available in the audio object in Mscape.

Creating A Fade

In Mscape, audio object have three fade actions available for use.

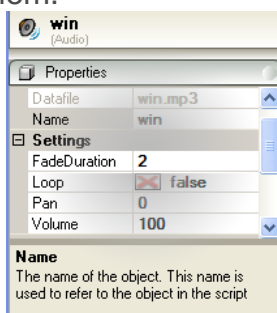
| | |
|-------------------------|---|
| <i>FadeIn</i> | Starts the sound playing while gently fading it in |
| <i>FadeOutAndStop</i> | Gently fades the sound out, and stops the audio once it is done |
| <i>_FadeOutAndPause</i> | Gently fades the sound out, and pauses once it is done. When the sound is played using <i>Play</i> or <i>FadeIn</i> , the sound will continue from the point at which it was paused |

To use these actions, drag-and-drop an audio onto a region, and use the wizard interface to select them from the list. Try using *FadeIn* for *OnEnter* and *FadeOut* for *OnExit*.

Alternatively you can type the name of the audio to use into the *OnEnter* event in the script window, hit the (period) key and select the appropriate action from the list. For example, if you have an audio called *pirate* in your Mediascape, select a region and type *pirate.FadeIn()*; into the *OnEnter* event.

Changing the Length of the Fade

The length of the fade is a property of the audio itself. Newly-imported audio objects will have the fade set to two seconds, but you can change this by typing your new value (in seconds) into the *FadeDuration* field on the properties box while the audio is selected. Note that this property only applies to the selected audio - other audio files will keep the default time of two seconds unless you specifically change them.

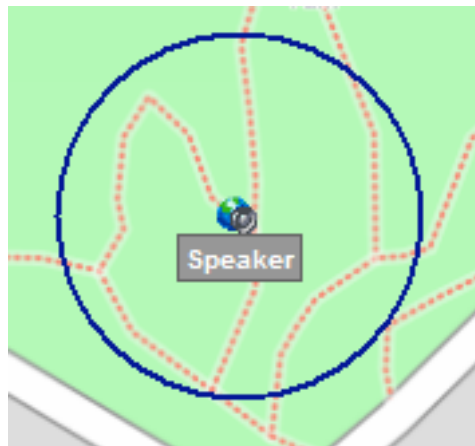


FadeDuration property to change the length of the audio fade

Speakers - Giving Sound A Place

A speaker is an object in Mscap that allows you to locate a sound to a specific point on earth. What makes this different to a region with an associated sound is that the sound will get louder the closer the user is to that point.

A speaker is represented by a point surrounded by a circle - it is as though there were a loudspeaker playing right in the middle of the circle. The edge of the circle represents the point at which you can no longer hear the sound coming from the speaker.



A speaker object

Creating A Speaker

- A speaker is created by clicking the **speaker** button below the map.
- To move the speaker around the map picks it up by clicking on the central icon.
- To change the range of the speaker, click and drag the blue border.

A newly-created speaker will not do anything by itself; you must associate an audio object with it first.

To associate an audio object with the speaker, drag and drop the object onto the speaker icon at the centre of the speaker region. If you have done this correctly a message will pop up to inform you that the audio is now associated.

Audio File Formats and Settings for Mediascapers

We generally recommend the use of compressed audio formats (mp3, ogg) over the uncompressed WAV format. The difference in file size (around 11x smaller) is very large when compared to the minor drop in quality.

The only case when we recommend WAV is when your mediascape requires many (over 5) audios to be played at the same time. The average mobile device is only capable of decoding around 5 mp3/ogg files at once before running out of CPU, whereas around 10-15 WAV files are possible.

MP3 Compression Settings

These are our recommended settings for mp3 audio in mediascapes. Remember that your user will be listening to your audio out in the real world which is full of background noise, so you can get away with a lower quality than would be acceptable for home listening.

- **Constant Bitrate**
- **22050hz** sample rate
 - **All audio in mediascapes is played back at 22050hz**, so increasing this or beyond will make no difference at all except for making your mediascape larger!
- For **music and background audio**:
 - **Stereo** is preferable if the source material is stereo.
 - **96kps** rate
- For **speech**
 - **Mono** is preferable unless the speech is mixed with background music
 - **64kps** bitrate

WAV Settings

As stated above, we do not generally recommend WAV audio for use in mediascapes. If you do use it however, use the following settings

- **22050hz** sample rate
- **16bit**
- **Stereo or Mono** depending on your source material

OGG Settings

Note that Windows Media Player does not have support for OGG playback built in by default. To add it, go [here](#) and download 'Ogg Codecs for windows'

We recommend the following settings for OGG encoding:

- **22050hz** sample rate
- **Stereo or Mono** depending on your source material
- **0.2** quality or close. Music may require a higher setting for a good level of clarity.

Using Images in Mediascapes

Images are handled in a similar way to audios when creating a Mediascapes.

Creating a Mediascape Using Mscape Maker explains how to import images and show them when the user enters a real-world area.

Image Size and Format

Mscape Mediascapes use images in the following formats

| | |
|-------------------|---|
| jpg*/*jpeg | This format is excellent for photographic imagery is generally recommended |
| gif | Line drawings tend to come out looking best in this format |
| bmp | This format is uncompressed, so the file sizes are large. It's not recommended to use BMP |

The Mscape Maker and the Mscape player will ensure that the image will always be fitted to the size of the screen and shown in full; however it is best to try and match your images to the size of the screen of your mobile device. The usual screen size is **320 by 240** pixels for screens in a landscape or **240 by 320** for a portrait.



Do not directly import large images into your Mediascape. Mscape will helpfully fit images to the size of the screen, but large images (such as those straight out of a digital camera) may be slow to load, or in some cases may not load at all on your mobile device even if the image appears without issue when using Mscape tester. We recommend using a program (such as the excellent free Photoshop alternative [Paint.NET](#)) to scale down your images to the size of the mobile device screen (usually 320x240) before importation into Mscape.

Other Considerations

Legible Text

If you are making your Mediascape available to others, for example through the mscapers.com web site, then they may be using them on hardware that is somewhat different to yours. In particular the screen could be smaller. So if you are using images that incorporate text make sure that the text is very legible as it may get shrunk down. Alternatively consider using HTML for doing the text as this will have less text shrinking issues (see the section 'Using HTML in Mediascapes')

Screen Format

Also bear in mind that the dimensions of the screen may be different. In particular it may be a square screen which will mean that the image will get shrunk. By default it will stay the same shape and there will be black space around it. You can change this by using the properties of the Images object.

Using Videos in Mediascapes

Video Size and Format

Mscape supports the Flash video format (.swf). As with images it is recommended that you resize your video to the desired screen size (normally **320 by 240** in landscapes). With video this is the first step in the crucial task of compressing your video file, which is essential for the best possible playback quality. Remember that the device used to run an Mscape is likely to be much less powerful than the computer you work on so you need to compress files in a way best suited to the device.

Other Considerations

Testing Playback Quality

Don't rely on the tester to determine the quality of video playback. It's good practice to test your Mediascapes on a suitable device and this is especially true when embedding video. Also consider the location where video will be shown and consider the issue of screen glare - try and use locations where the user can find shade and properly enjoy your video content.

Interactive Flash Movies

If you wish to add a Flash 'movie' that has interactive elements then you should add it as a [FlashMovie](#) **not** as a video.

Related Articles

[How to Create Video Files for Mscape using Adobe Flash CS3 Professional](#)

How to Use Adobe Flash in a Mediascape

This document introduces the use of Adobe Flash in the Mscape system.

For quick reference and troubleshooting you may want to try the [Flash FAQ](#).

What is Flash?

Flash is a tool and file format for creating interactive, media-rich applications for the internet.

You can use it in a simple drag-and-drop manner, or you can get more involved and write scripts (programming code) to create more complex things. With the Flash tool you create a file called a 'Flash movie'. This can be imported into your Mediascape.

Here are some reasons you might want to use Flash.

- You can put a lot of interesting interactions and UI capabilities into your Mediascape that are not possible with 'pure' Mscape code.
- You can leverage all of the great examples, tutorials, and Flash authoring communities on the web to really help you along.
- You know a bit of Flash already, and want to put your skills to work in an Mscape environment

Creating Flash Movie Files

To create Flash movie files you need a Flash authoring environment such as [Adobe Flash CS3](#), which can be purchased from Adobe directly online from <http://www.adobe.com>, or via most software retailers.

There is a [30-day trial version](#) available for free download if you want to see if Adobe Flash is right for you and your Mscape prior to purchase.

It is possible to use older versions of the Flash authoring tool, such as Macromedia Flash MX, but we recommend getting the latest up-to-date version as many of the examples on this site cannot be opened with older versions.

While it's possible to import, display and run previously-built flash movies, for best results we recommend that you design them specifically to fit your Mediascape. The PDA runs an older version of Flash (Flash 7), and the PDA itself is quite a different platform to author flash content for due to the screen size, interaction mechanism, and, by desktop PC standards, a very slow processor.

See [Creating a Flash Movie for Mscape](#) for a step-by-step guide on how to make Flash movies that are suitable for use in a Mediascape.

Adding Flash Movies to the Mediascape

In order to add Flash Movies to your Mediascape, you first need to add the *FlashMovies* object.

- With your Mediascape open in Mscape Maker, right-click on the 'Media' object in the top-left panel. Choose 'Add Flash Movies' from the menu.
- A new button will appear on the menu bar labeled *Flash Movies*
- Click this, and an open file dialog will appear allowing you to choose the Flash movie.

Imported movies will then appear below the 'Flash Movies' object.

Note that your flash movie will not play automatically when your Mediascape is started, you need to explicitly start it using the Flash movie's *Play* command. To do this, simply drag-and-drop the movie into the *Mediascape* (top of the left panel) object's *OnLoaded* event, type . (period) and select Play from the pop-up list.

Creating Flash Movies that Interact with the Mediascape

It is possible to communicate information between a Flash file and a Mediascape. This is similar to what is possible with HTML, but whereas HTML is limited to text and numbers, Flash offers more interesting possibilities in terms of graphics and interactivity. You could create games and puzzles that the user interacts with as part of the Mediascape.

[How do I control the Flash movie from within my Mediascape?](#)

[How do I control my Mediascape from the Flash movie?](#)

Further Help

The [Mscape Flash FAQ](#) contains questions and answers for many of the most frequently-asked questions, so this should be your first port of call for more help.

Creating a Flash Movie for Mediascapes

Adobe Flash gives you as the author a lot of different options for your flash movie. You can target different Flash Player versions, use different screen resolutions & framerates, different versions of [ActionScript](#) and more.

The aim of this guide is to help you find the best settings to use for your Flash Movie for use in a mediascape. Note that I am assuming that you are using Flash CS3, earlier versions may have slightly different options available.

Setting Up

- Start up Flash CS3
- Select File / New to create a new document
- Choose 'Flash File ActionScript 2.0'.

mscape player runs on **Windows Mobile** which currently only supports **Flash 7**. This version of flash supports **ActionScript 2.0** only, so this option should always be used. If you want to include any third-party ActionScript code, make sure it doesn't use ActionScript 3.0, or it will not work in your mscape.

- A new blank Flash movie will appear in the interface. Next up we need to set the screen size and frame rate options.

Most Windows Mobile devices are **320x240** in portrait mode, and **240x320** in landscape mode, so these are the recommended sizes to use. For frame rate, remember that the PDA is many times slower than your PC, so try and keep the frame rate as low as possible. We tend to use the default of **12 frames per second**.

- On the main menu, go to Modify / Document. Set width to 320, and height to 240. Set Frame rate to 12.
- Next we are going to change the publish settings to ensure the Flash movie is compatible with the Windows Mobile version of Flash, Flash 7.
- On the main menu, go to File / Publish Settings.
- Click the *Flash* tab.
- Select **Flash 7** from the drop-down list by the text 'version'. Double-check that [ActionScript](#) 2.0 is selected further down the page. The other default options are fine, though if you're going to use audio in flash you may want to experiment with higher quality *audio stream* and *audio event* settings. 96kpbs stereo for these can give excellent results for music and speech.

Speed Considerations

Flash movies will run considerably slower on your mobile device than they do on your desktop computer. It is a very good idea to regularly try out your flash movie on your mobile device during the development process to ensure that it runs at a reasonable speed.

Here are a few tips that can help you keep your movie fast and responsive.

- When using animation, try and confine that animations to small areas of the screen.
- Avoid transparency wherever possible, especially for larger objects.

- When using images, ensure the resolution of the image is as low as possible. For example, if you want a photo in the background, do not import a large image, and then scale it down inside Flash. Instead, reduce the resolution of the photo to the actual size you wish it to be on screen using a tool like *PhotoShop*, and then import it into Flash.

Use the lowest frame-rate that is acceptable for your project.

How Do I Control the Flash Movie from Within My Mediascape?

Flash movies allow mediascapes to have a graphically-rich animated UI. In order for this UI to be truly there are two parts to this. Creating the Flash movie to include in the mediascape, and then calling functions in the mediascape to control the Flash movie.

Create a Flash Movie

See [Creating a Flash Movie](#) for mediascape for a step-by-step guide on how to make Flash movies that are suitable for use in a mediascape.

Include the Standard Mediascape Library

Once you have created your flash movie, in order to allow the Flash movie and your mediascape to communicate with one another, you need to add the mediascape [ActionScript](#) library.

Click the first frame of your movie, and bring up the Action editor (F9). Add the following:

```
#include "mediascape2.as"
```

Make sure that the file 'mediascape2.as' is available in the same directory as your Flash file so that it can be included.

Write [ActionScript](#) Functions

For each action that you want the flash movie to be able to perform, you'll need to write an [ActionScript](#) method.

For example, you may have a Flash Movie that shows a welcome page on frame 1, and an animated sequence of a monster on frame 7. When your user walks into a region, you want the animation to begin. To achieve this, you'll need to write an [ActionScript](#) function that jumps to frame 7 - this function should go under the `#include "mediascape2.as"` line that you've already added.

```
function StartAnimation(){
    gotoAndPlay(7);
}
```

Calling the Functions From Your Mediascape

From the mediascape you need to call this function you have defined above.

Firstly though you need to actually get the Flash movie running so that it will appear on the screen of the mobile device and react when you call the functions.

In the script for the 'OnLoad' event for the mediascape (the script that is run when the mediascape first starts up) you need to add the line:


```
myflash.Play();
```

(Where myflash is the name of your flash movie)

You can type this in or you can drag the Flash file from the left-hand panel and drop it in the script window then type a dot to get the possible completions.

Now you need to call the Flash function. You do this in response to some event. For example you could create a region and select it to show the script for the 'OnEnter' event in the script window (bottom-right panel).

Once again type this line in or drag and drop your Flash movie from the left-hand panel to the script window and it will put the name there. If you now type a full-stop (a period) after the name a menu will pop-up showing the possible completions. Select 'RunActionScript' (you have to double click) and you will have the following line in the script:

```
myflash.RunActionScript(name);
```

The 'name' bit needs to be changed into the name of the function that you want to call. Like this:

```
myflash.RunActionScript("StartAnimation")
```

Now, when this particular mediascape event is triggered, this bit of code will run the function in the Flash movie.

Passing Data From the Mediascape to the Flash Movie

This is a similar process to that above, the main difference being that the function in the Flash movie needs to accept arguments and the function in the mediascape that calls it must pass it those arguments.

For example if we had a function in the Flash movie that displayed a score and it was declared like this:

```
function ShowScore(sc){
    scoreDisplay = Number(sc); // ensure that sc is read as a number (see 'important note' below)
}
```

Then, in the mediascape we would need to call this function and pass it the argument like this:

```
myflash.RunActionScriptWithParams("ShowScore", 150);
```

Note that we are using *RunActionScriptWithParams* instead of *RunActionScript* and that we use the name of the function and the parameter that we are passing. The more parameters the function accepts the more we put in the statement to call it from the mediascape.

Important Note

All pieces of data passed from the mediascape to flash are passed as strings, e.g. pieces of text. This will usually cause no problems as flash should intelligently convert those strings into numbers if and when required. However in some cases you may get unexpected results.

```
function AddScore(sc)
{
    currentScore = sc + 10; // this may NOT work, do not use!
}
```

In the above example, if currentScore is a 100 and the function AddScore is called with sc set to 10, currentScore may be set to 10010 rather than 110. This is because Flash may decide that you want to concatenate the text strings "100" and "10" together, giving "10010", rather than adding them together numerically.

To ensure it will always work, surround uses of any numeric parameters to functions called directly by the mediascape with the Number() function.

```
function AddScore(sc)
{
    currentScore = Number(sc) + 10; // This is the correct way
}
```

Sending Multiple Pieces of Data

You are not restricted to sending a single piece of data with each call to [RunActionScriptWithParams?](#). It's very simple to send more than one - if you define an ActionScript function with multiple parameters, you can simply add extra parameters to the corresponding call to [RunActionScript?](#)

For example:

```
function MultipleParameters(a,b,c,d,e)
{
    _total = Number(a) + Number(b) + Number(c) + Number(d);
    _playerName = e;
}
```

To call this function in mscape..

```
myflash.RunActionScriptWithParams("MultipleParameters", 1,2,3,4,"bob");
```

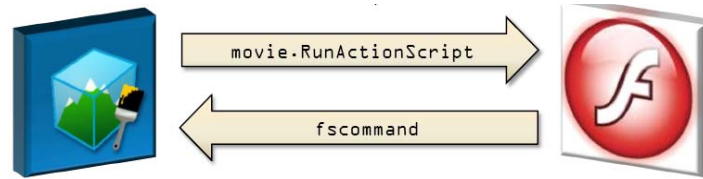
Controlling the Mediascape From Flash

It is also possible to control the mediascape from within the flash movie. For example, you may want to have buttons in your flash movie that affect the mediascape - perhaps playing an audio or loading a new flash movie.

How Do I Control the Mediascape From the Flash Movie?

You should read the document [How to Control Flash from the Mediascape](#) before following this section

It is possible to send notifications and data back to the Mediascape from within a Flash movie. For example, you may want to notify the mediascape when a particular Flash button on screen has been pressed, and then perform an action such as playing a sound in response.



Sending Button Presses from Flash to Mscape

- Download the example FLA file `PressExample.fla` and the Mediascape Flash Actionscript Library `mediascape.as` to the same folder on your computer.
- Open the FLA file in Flash.
- Click the large 'Press' button
- In the Actions Window (Window / Development Panels / Actions Panel), ensure the following code is inserted:

```
on (press)
{
    fsccommand("press");
}
```

This code uses the `fsccommand` method to send a message to the mediascape containing the word "press". The mediascape will receive a `OnFSCommand` event on the Flash movie object carrying with it the data "press". The mediascape author can then choose to respond to this event by looking at the data and playing a sound in response.

- Publish the swf file (File / Publish)
-

Next we need to create the mediascape.

- Open Mediascape Editor
- Import an audio file by clicking the 'Audio' button on the toolbar.
- Add a Flash Player (right-click *Media* and select *Add Flash Movies*)
- A button labelled *Flash* will appear in the main toolbar. Click this button to import the swf that you have just published.
- Click on the newly-added flash object in the mediascape window (left-hand panel)

The script window (bottom-right panel) will list the event `OnFSCommand` on your Flash movie object. This event will be triggered every time there is a `fsccommand()` method in the flash movie.

- To respond to the `fsccommand` from flash by playing a sound add the following code

```
if (Command == "press")
{
    myAudio.Play();
}
```

(where myAudio is the name of the audio you just imported)

You can insert as many calls to *fscommand()* in your Flash actionscript as you want, for example you can have many buttons that send different commands. All of these commands will end up in the same place in the mediascape - the *OnFSCommand* event. In order to differentiate between the different commands in the mediascape you should add *if* blocks to your mediascape *OnFSCommand* code for each possible command. For example:

```
if (Command == "Press")
{
    myAudio.Play();
}
if (Command == "Pause")
{
    myAudio.Pause();
}
if (Command == "Exit")
{
    goodbyeImage.Show();
}
```

Sending Data from Flash to Mscape

The previous section introduced the sending of notifications from mscape to flash - when a button is pressed etc. In some cases however, you may wish to send data - for example the current score, or the X and Y position of an enemy.

Sending a Single String

Sending a single piece of string data is straightforward.

```
fscommand("playerName", _currentPlayerName);
```

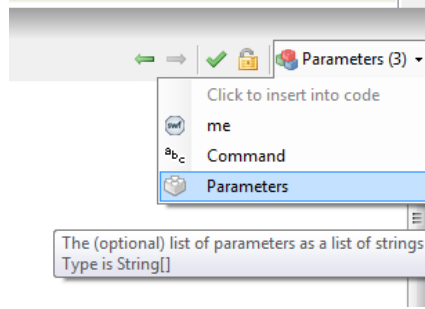
Here, we're sending back the name of the current player to mscape (held in the *_currentPlayerName* variable in actionscript).

To pick this up in flash, in the *OnFSCommand* event for the flash movie we'd write:

```
if (Command == "playerName")
{
    Logger.Log("The player's name is " + Parameters[0] );
}
```

First up, we're checking that it was the *playerName* command that was sent. Then we're writing out to the log window a message. In order to get the name of the player out we're going to use the object called *Parameters* which is sent along with the *OnFSCommand* event.

To see the *Parameters* object, click on the 'Parameters' button on the events window (on mscape 2.5 only, for mscape 2.1 click the right-most button on the script window toolbar) and all of the parameters for the *OnFSCommand* event will be shown.

Viewing Parameters to the `OnFSCommand?` Event

The Parameters object is a list of strings, one for each parameter sent from the flash movie. In this case there is only one - the name of the player - so we use the syntax `Parameters[0]` to get it out.

Sending a Single Number

```
fscommand("currentScore", _score);
```

In this example, the command 'currentScore' is sent, along with the value of the variable called `_score`.

The corresponding code in mscapexml may look like this.

```
if (Command == "currentScore")
{
    score.Value = Int32.Parse(Parameters[0]);
}
```

The `Int32.Parse` section requires a little explanation.

All data sent from flash to mscapexml is converted to strings (text). The scripting language in mscapexml is quite strict, as it is based on C# - so you cannot assign a string value to a number value. In the example above, `score` is a `Number` object (in the 'State' section), so you need to convert the string value into a number before you can make the assignment. The `Int32.Parse` function converts a string into an integer (like 13 or 5). If your number is floating-point - e.g. 12.3 or -1.0324 - you should use `Double.Parse` instead.

Sending Multiple Pieces of Data

Sending multiple pieces of data from flash to mscapexml is slightly more involved, but still perfectly possible.

The `fscommand` method in ActionScript has only two parameters, `Command` and `Parameters`. This means you cannot do the trick where you send extra parameters by just adding them to the list - e.g. `fscommand("sendLocation", _x, _y)` will not work.

Instead we need to construct a single string that contains all the parameters we want separated by commas, and pass that into the second parameter to `fscommand`.

Here's an example in [ActionScript](#).

```
var playerName = "Bob";
var x = 1003;
var y = 500;
fscommand("playerMoved", playerName + "," + x + "," + y);
```

HOW DO I CONTROL THE MEDIASCAPE FROM THE FLASH MOVIE?

Note that as commas are used as the separator, your strings should **not** contain any commas or mscage will split them into separate parameters. Be particularly careful with any input text written by your users!

To pick up the parameters in the OnFSCCommand event in your mscage you'd do the obvious thing and use the indexes (1,2) of the new parameters in the Parameters array.

```
if (Command == "playerMoved")
{
    name.Value = Parameters[0]; // gets the playerName variable out
    playerRegion.X = Double.Parse(Parameters[1]); // gets the X out - see earlier note on receiving
numbers
    playerRegion.Y = Double.Parse(Parameters[2]); // gets the Y out
}
```

Frequently – Asked Questions for Flash

What Version of the Flash Authoring Tool Should I Use?

We recommend Adobe Flash CS3 Professional, this is available from <http://www.adobe.com/products/flash/>. Older versions, e.g. Macromedia Flash MX can be used, but you may not be able to open many of the example FLA files found on this site.

What Version of Flash Should I Output For? (Flash Lite / Flash 7 / 8 / 9 etc)

Flash 7 is very important that that you set your movie to target this, and not any more recent versions. To set it, go to the File / Publish Settings screen, then the Flash tab, and choose 'Flash 7' next to the text version. Don't use any of the 'Flash Lite' versions. This may change when Mscape expands to include Windows Mobile devices without touchscreens, but for now Flash 7 only is supported.

What ActionScript Version Should I Use?

Flash 7 (see above) only supports a maximum of **Actionscript 2.0**, so this should always be used. If you want to include any third-party [ActionScript](#) code, make sure it doesn't use Actionscript 3.0, or it will not work in your Mscape.

What Frame Rate and Screen Size Should I Use?

Most Windows Mobile devices are **320x240** in portrait mode, and **240x320** in landscape mode, so these are the recommended sizes to use. For frame rate, remember that the PDA is many times slower than your PC, so try and keep the frame rate as low as possible. We tend to use the default of **12 frames per second**. To set the screen size and frame rate, go to Modify / Document via the main menu.

My Flash Movie worked fine in Mscape tester, but doesn't play back properly on the PDA

Probably what has happened is that you have published your Flash movie for Flash Player 8 or 9. This will run fine on your desktop PC, but not on the PDA as that only runs Flash 7. Also make sure that Flash Player 7 is installed on your mobile device - Flash Lite will **not** work. If Mscape player gives you an error when you open an Mscape that uses Flash ensure that Flash Lite is not installed, and install Flash Player 7 instead.

How Do I Create Video for Use in Mscape?

Mscape uses Flash to play videos as well as interactive Flash content. If you have Adobe Flash CS3 Professional, the following page steps you through the sequence to output movies that are compatible with Mscape. [How to Create Videos Using Flash CS3 Professional](#).

How Do I Control the Playback of My Movie From Mscape?

See the following help page for details [How to Control a Flash Movie from a Mediascape](#).

How Do I Tell the Mscape that a Button Has Been Pressed in My Flash Movie?

See the following help page for details [How to Control the Mediascape from a Flash Movie.](#)

Where Can I Get Help with Flash Authoring?

There are many places on the web where you can find tutorials, sample code, and forums. One of our favorites is [FlashKit](#). In addition to tutorials and samples, this site also has a large database of free sounds, graphics, and animations that you can include in your Flash movies.

How Do I Loop a Flash Movie?

The Flash communication system built into Mscape prevents movies looping by default. To work around this, you should put a gotoAndPlay(1); action in the last frame of your movie using the [ActionScript](#) window (F9).

LoadMovie / LoadVars / XML Doesn't Work Properly Within Flash

There is a difference between the way Flash handles URLs to local files when run as part of a Mediascape. In short, you cannot use relative URLs to local resources in Flash movies running as part of an Mscape. See the following document for details on how to work around this issue - [UsingLoadMovie](#).

I've Stopped my Flash Movie, but Some Sounds are Still Playing?

Audios that are played looped in Flash don't stop when you run the Stop function within Mscape. See the following document for details on how to work around this issue - [LoopedAudios](#).

Slide-Shows

How Do I Use Slide-Shows in My Mediascape?

What Is An Mscape Slide-Show?

A slide-show is a collection of audio, images and video that are played one after another (or even together) in a particular sequence.

In that way it is more complex than just an image or an audio.

Images and audio files can be imported into the Mediascape, but a slide show has to be created in the Mscape Maker.

Think of it as telling the user a short story with media.

Why Would I Want a Slide-Show in My Mediascape?

Mediascapes are about media and places; you can associate an image with a place or an audio with a place. But there may be times in your Mediascape where you want to show the user more than just one image or audio. You may want the user to rest a while and see a collection of information relating to the place that they are in.

For example, if you have a collection of old images and sounds relating to a historic building on a larger site then you could assemble them into a slide-show and play that slide show when the user arrives at that particular building.

How Do I Create a Slide-Show?

Gather Your Media

You need a collection of different types of media that you want to show and possibly extra pieces of media that you want to include to bring them all together, for example; a musical backing track or a spoken narrative.

Enable Slide-Shows

To do anything with slide-shows in Mscape Maker you first have to enable their use.

Right click on the media object in the left hand panel.

Select 'Add Slideshows' from the menu

The 'Slideshows' object will appear in the list of media objects.

Create a Slide-Show

Right-click the 'Slideshows' object and select 'Add Slideshow'.

A new slide-show object will appear below the 'Slideshows' object. You can edit its name if you want to.

Now double click this slide-show and a window will appear where you can add things to the slide-show.

There are two parts to it. In the top part you can drag an audio file to play in the background, and in the larger, lower box you can drag the media items that you want to show. When you drag an image or video into the lower box it will always be placed at the top of the list.

Note: You can only drag media items here from the left-hand panel of Mscape maker. You cannot drag media here from the folders on your computer or desktop. First you must import them into the Mediascape and then drag them from the left-hand panel to the slide-show window.

Each item in the lower box has a time associated with it, you build the slide-show by editing these times to control the order and duration that each item is shown for. The order of the list depends upon the timing information for each item. As you change them you will see the list shift about to keep them all in the right order.

Note: If you have an item that you want to show more than once in the slide-show you simply drag it into the box as many times as you need it and adjust the timing controls accordingly.

Associate the Slide-Show with a Region

Below the map in Mscape maker are buttons for creating different shaped regions. Click on one (for example; 'create circle') and you will see a circular region appear in the centre of the main screen. You can move it around and resize it or just leave it where it is.

Drag the slide-show object you have just created, from the left-hand bar and drop it onto the region. This will make a wizard (a dialogue) appear that will ask you how you want the slide-show object to react to the enter event and the exit event. The usual choices are to play on enter and to stop on exit. You will be asked to confirm the choices you made, and that is it. You have a region that has a slide-show associated with it and it will start on entry to the region and stop on exit.

Testing the Slide-Show

As a slide show is more complex than just including an image or audio in your Mediascape you may want to test it to confirm that it all works. To do this try running your Mediascape on the Mscape tester by clicking the 'Tester' button on the top bar of the window.

WebPages

Using HTML (web page markup) in a Mediascape

[What is HTML?](#)

[Why would I want to use HTML pages in my Mediascapes?](#)

[How do I get a single HTML page into my Mediascape?](#)

[How do I get complex or multiple HTML pages into my Mediascape?](#)

[How do I show the user information using an HTML page?](#)

[How do I get information from the user with an HTML page?](#)

What is HTML?

HTML is a language that is used for adding format and layout to the text of web pages. An HTML page is basically just a web page; it consists of tags that you put around parts of the text to give it certain layout properties. You can use it to enclose bits of text and say things like ‘this text is a heading’, ‘this text should be written in bold’, etc. Mediascapes can include simple web pages in the much the same way that they can include audio and images.

Using HTML

To work with HTML pages you will need to have some familiarity with HTML code. Ideally you will have had basic experience in creating a simple web page by directly editing the HTML code of the page. You can create web pages with drag and drop applications such as DreamWeaver[?], but creating pages in a drag and drop manner does not give you an insight into the inner workings and structure of the code of an HTML page.

Learning HTML

HTML is not as complex as a programming language but as you add more and more formatting to information it can get difficult to see what is going on. Having said that you can learn the basics of HTML very quickly and with those basics you can already do useful stuff.

Other Things You Can Do

Web pages can be more than just HTML layout, they can include more complex technologies, below are few of them:

JavaScript - These are bits of programming code that do things when the page is displayed.

Style sheets – These are descriptions that control how the elements in a web page are displayed, what size and style of font, colors, alignment etc.

XML – it is possible to do complex things with web pages that can update their content from XML files in the Mediascape. When a Mediascape stores variables it does so in a simple XML format.

Using any of these technologies is complex, idiosyncrasies mean that the way that they are presented on the screen of the mobile device could differ from how they appear in the browser on your desktop computer or on the simulated screen of the Mscape tester. You can’t do any damage if you give them a try, but the development process could be frustrating if you do not already have experience in the area.

Why Would I Want to Use HTML Pages in my Mediascapes?

HTML pages provide a simple method of presenting information on the screen. If you wanted to show the user some text or numbers you could build an image with this information in and show it to the user at the appropriate time. Presenting information as HTML pages takes up less file space than images and there are some situations where images would not be able to be used, for example information that changes during the course of the Mediascape. The key times you would want to use HTML pages are:

- **Dynamic information:** Using HTML pages allows you to show the user information that is changing; things like their score in a game.
- **Repetitive information:** You can also use HTML pages as templates, if you have a hundred pages of fixed information you could produce a hundred images to load into the Mediascape or you could define one HTML page that you use and fill with the appropriate content a hundred times.

Getting user interaction: As well as showing the user information, an HTML page can also be used to interact with the user and feed the results of this interaction back into the Mediascape.

How Do I Get a Single HTML Page Into my Mediascape?

Single Web Pages and Complex Web Pages

When dealing with web pages we have to make a distinction between single and complex web pages. A single web page is a simple web page contained in a single file. It is a standalone file that does not involve any other pages or files.

A complex web page does rely on other files, either it contains links to other web pages ('click here for more information on rabbits') or it includes other files such as images ('Here is a picture of my rabbit'). It is more than one file that are related in some way.

Importing Single Web Pages

As a single web page is one file, it can be imported in almost the same way as importing audios or images.

Enable Web Pages

To do anything with web pages in Mscape maker you first have to enable web pages. Right click on the media object in the left hand panel
Select 'Add web pages' from the menu

Import Your Web Page

The 'web pages' object will appear in the list of media objects and the web pages button will appear in the import part of the top bar. To import a single web page click on the button in the top bar or right click on the object in the left hand panel then select the web page you want to import.

You can check your web pages by clicking with the right mouse button on your web page in the left hand panel and choosing 'preview' from the menu. Alternatively, you can include the web page into a region and then test your Mediascape with the Mscape tester to see if the web page displays correctly on the simulated mobile device screen.

If you make changes to your HTML page the Mscape maker will not know about them, you must remember to re-import the changed version into your Mediascape.

How Do I Get Complex or Multiple HTML Pages Into my Mediascape?

Single Web Pages and Complex Web Pages

When dealing with web pages we have to make a distinction between single and complex web pages. A single web page is a simple web page contained in a single file. It is a standalone file that does not involve any other pages or files.

A complex web page does rely on other files, either it contains links to other web pages ('click here for more information on rabbits') or it includes other files such as images ('Here is a picture of my rabbit'). It is more than one file that are related in some way.

Importing Complex Web Pages

If you try to import a complex web page in the manner described above, it will be imported but it will not bring all the other files and pages with it. So any links will not work and any images will not be shown.

Develop Your Web Pages

The best approach is to develop your complex web pages in the content folder of your Mediascape (see below). Here you can create folders and add extra pages and images. The one thing you have to make sure is that all the starting point pages (that you want to call up directly in the Mediascape) are in the content folder and not in sub-folders within that.

To find out where the content folder is, make sure that you have saved your Mediascape and then, in the 'tools' menu, choose 'open content folder'.

Import Your Starting Point Pages

When you import one of the starting point pages (in the same way as you'd import a single web page, above) it will appear in the web pages object but all the links and images will be functioning correctly.

How Do I Show the User Information Using an HTML Page?

here are two parts to this; you need to make an HTML page to show the information and then you need to send information to it from the Mediascape.

Showing the Information in an HTML Page

The HTML page is a conventional HTML page with two additions.

- A small fragment of script to show the information on the page so that the user can see it.
- A large script that gets the information passed from the Mediascape. You just copy and paste this in so you don't need to even look at it.

Here is a simple HTML page without the additions:

```
<html>
<head>
<title>My page</title>
</head>
<body>
Hello World!
</body>
</html>
```

If you are not too sure about all the terms then you need to learn some basic HTML before going further.

In the 'body' of the page we want to add the information that we sent to the page from the Mediascape. In the Mediascape we will shortly be identifying this information by calling it 'score' so the bit we want to add here looks like this

```
<script type="text/javascript">
  document.write(srchData.score);
</script>
```

We can put it in the file after the 'Hello World!' text, like this:

```
<html>
<head>
<title>My page</title>
</head>
<body>
Hello World!
<script type="text/javascript">
  document.write(srchData.score);
</script>
</body>
</html>
```


HOW DO I SHOW THE USER INFORMATION USING AN HTML PAGE?

Now, to make it all work, we need to paste in a large script to get the information. This goes in the 'head' part of the page. Once it is there we don't do anything else with it. The final HTML will look like this:

```
<html>
<head>
<title>My page</title>
  <script type="text/javascript">
    function decodeSearchString() {
      var nameValue = new Array();
      var searchStr = unescape(location.search.substring(1));
      if (searchStr) {
        var formElement = searchStr.split("&");
        var tmpArray = new Array();
        for (k = 0; k < formElement.length; k++) {
          tmpArray = formElement[k].split("=");
          nameValue[tmpArray[0]] = tmpArray[1];
        }
      }
      return nameValue
    }
    var srchData = decodeSearchString();
  </script>
</head>
<body>
Hello World!
<script type="text/javascript">
document.write(srchData.score);
</script>
</body>
</html>
```

In order to work the HTML page needs to be imported into the Mediascape. See [How do I get a single HTML page into my Mediascape?](#) for how to do this.

Note: If you want more than one bit of information to be shown in the HTML page then simply add more of these small code fragments with the appropriate variable names in place of the name 'score'. If you want to show lots of information but you don't know in advance what they are going to be called then you can use this page, this approach is also useful for debugging:

Sending Information From the Mediascape

In the Mediascape you need to set up a variable to hold the information you want to send to the HTML page. Imagine we want to show the users score in some Mediascape game.

Creating Variables

In the left-hand panel on the Mscape maker right-click on the 'State' object and select 'Add Group'. Right click the group and select 'Add Number'.

Note: You can change the name of the group and the name of the variable simply by clicking on the names and editing them.

When the number is selected you can see its properties in the bottom-left panel of the Mscape maker. Here you can change the initial value that the variable has.

Note: If you want more than one bit of information to be sent to the HTML page then simply add more variables to this group and they don't have to be numbers they can be texts or logical True/False values.

Sending Variables to the HTML Page

Now you want to say 'hand this variable to the HTML page'. You do this in response to some event. For example you could create a region and select it to show the script for the 'OnEnter' event in the script window (bottom-right panel).

Drag and drop your HTML page from the left-hand panel into the script window and it will put the name there. If you now type a full-stop (a period) after the name a menu will pop-up showing the possible completions. Select 'ShowWithStateGroup' (you have to double click) and you will have the following line in the script:

```
myhtml.ShowWithStateGroup(extraData);
```

The extraData bit needs to be changed into the name of the variable group that you created, if it is called 'Group01' then you need this line of script:

```
myhtml.ShowWithStateGroup(Group01);
```

When the event is triggered, this bit of code will display the HTML page on the mobile device and hand it the variable to be displayed.

How Do I Get Information From the User With an HTML Page?

Html is one of the ways of showing information to a user and getting them to respond. All devices that can run the Mscape player can also show HTML unlike Adobe Flash. One of the benefits of HTML is that you can display dynamic information such as a score without having to create images for all possibilities. You can also get the user to tap on links and buttons in the HTML and then respond to that from inside your Mediascape. This how to is about finding out when a user has tapped on the Html and responding to it, there is lots more information on other aspects of using HTML in Mediascapes here !!![Insert link to html in docs!!!](#)

Let's begin by [starting the Mscape maker and creating a new Mediascape.](#)

For this how to you will also need a way of editing very simple html documents (also called web pages). I recommend Windows Notepad, you should be able to get to this from the start menu by clicking on Start-> All Programs -> Accessories -> Notepad. You can of course use your favorite web page maker like Dreamweaver or Front Page. The extra code added automatically by Microsoft word when editing html tends to cause problems and so is best avoided.

Open Notepad and copy and paste the following text into Notepad.

```
<html>
<body>
Hello, this is a web page
<BR>
<a href="page2.html">Go to second page</a>
</body>
</html>
```

Now save this document as page1.html into somewhere you can find and then import it into your new Mediascape. For more on importing media please see [How to import media into a Mediascape](#). So this is a very simple page it will let the user go to a different page but will not let them trigger events in your Mediascape. Now we are going to add a new page that will give the user the chance to interact with the Mediascape, again open Notepad, past this in and save it out as page2.html then import page2.html. It does not matter if you do not understand what is happening just yet it will be explained later on.

```
<html>
<head/>
<body>
This is page 2
<BR>
<a href="page1.html">Go to First page</a>
<BR>
<a href="mediascape://Play">Play a sound</a>
<BR>
<a href="mediascape://Stop">Stop a sound</a>
</body>
</html>
```

Now we can go back to the Mscapemaker. This is a good time to import a piece of audio that we will control with the second web page that you have just added. To start this off we want the first page to show up when the user starts the Mediascape, to do this we click on the *Mediascape* object at the very top of the left hand panel in the maker. You should now see two events in the events panel near the bottom of the maker window. Drag the object called *page1* into the *OnLoaded* event, type "." and select *Show*. This now means that when the Mediascape is loaded the html page called *page1* will be shown to the user.

If you look at the html in page one you will see that it has a link to page 2. This is a normal kind of link that you get on many web pages, it will cause page 2 to be shown. Now if you look at page 2 then you will see that there is a normal kind of link to page 1 and also two extra links. These extra links have a slightly different kind of address in the *href*, the address start "Mediascape://". If the user clicks on a link who's *href* starts "Mediascape://" it will not go to a new page, instead the information will go to your Mediascape as an event.

Now I will explain how to use these events and then once you have tried it in the tester this should all make sense. Click on *page2* in the left hand panel of the maker. You will see three events in the events panel near the bottom of the maker *OnLoaded*, *OnLinkClicked* and *OnFormPosted*. It is the middle one *OnLinkClicked* that we will use in this how to.

!!!Image of three event tabs!!!

When a special link (one that has "Mediascape://") is clicked on then this event will trigger. If you select the *OnLinkClicked* tab and then click on the little icon on the right of all the tabs

!!!Image of parameters button!!!

Then you will see a new little panel called the *Parameters* panel in the bottom right of the maker window. This will have two entries in it *Me* and *link*.

!!!Image of parameters panel!!!

These parameters are extra bits of information that we can use in this event. For this we are going to use the *link* parameter. This link parameter is the bit of text after the "Mediascape://" in the href. So if the user clicks on the link that says

```
<a href="mediascape://Play">Play a sound</a>
```

then the *link* parameter will be the single word *Play*. It is best to avoid spaces and special characters for the *link* text as this can make it work differently between the tester and the player.

In order to play the audio we imported at the beginning using the special links, the Mediascape needs to match the right link with the right action. To do this we compare the *link* parameter with the words "Play" and "Stop" as those are the words after "Mediascape://" in the html. Here is an example of doing that for a piece of audio called *audio1*.

```
if(link == "Play")
{
    audio1.Play();
}
```

```
if(link == "Stop")  
{  
    audio1.Stop();  
}
```

This Mediascape is now ready to be tested in the tester. If you click on the tester button on the makers tool bar then you should see the tester come up in a new window and *page1* will show up in the left hand panel that is meant to represent the device screen. If you click on the link in that page then you will see *page2*, here there is the link back to the first page and the two special links that we added. If you click play then you should hear the music start to play and the stop link should stop the audio.

Congratulations you have now learnt how to add links between html pages and your Mediascape! If you are having trouble why not look at the example that is now posted here: !!!Post example here!!!

How to Use Hardware Buttons

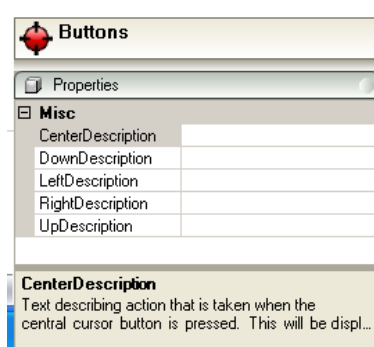
Almost all devices that can play Mediascapes have buttons. You can use these buttons when building your Mediascape so then when a user actually goes out and plays it then they can control stuff in the Mediascape. For example if you want the user to be able to mark a place then you can give them a button for that and they can press it whenever they want to mark that space.

Let's begin by [starting the Mscape maker and creating a new Mediascape](#).

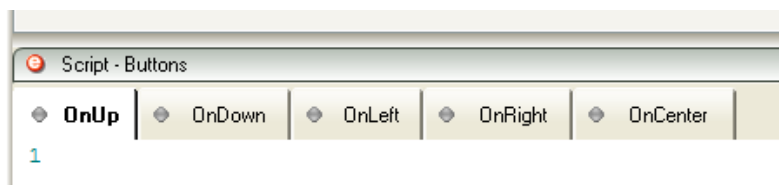
The now if you look in the *Tools* section in the left hand panel you will see an entry called *Buttons*. This is the *Buttons* object that I will now describe how to use.



If you click on it then you will see its properties in the bottom left of your maker window, we will come back to these at the end of the how to.



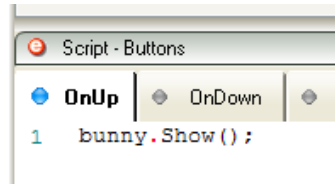
You will also see five new events in the script panel at the bottom of the maker window. This is the important bit of the buttons object. There are five events *OnUp*, *OnDown*, *OnLeft*, *OnRight* and *OnCenter*, these correspond to the four directions and the middle of your device's joystick or directional pad. These are the buttons that we can control and what the user can press to trigger things in the Mediascape.



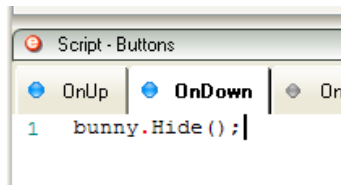
Before we can go any further we need something for the buttons to do. For this we are going to import some images, please follow the steps in [How To Import Media](#) to add a few image files, try to make sure they are not much bigger than the resolution of your mobile device, otherwise they will be quite slow to load. Now we can use buttons to show images.

HOW TO USE HARDWARE BUTTONS

Click on the *Buttons* object and look at the script panel near the bottom of the maker that we looked at earlier. Let's make an image show when the user press up on their device. To do this we drag one of the image objects that you imported earlier into the event window and drop it. We then type a "." and a list of things that we can do with the image will show up, we will select *show*. The event window should now look a bit like this, only with your own image name.



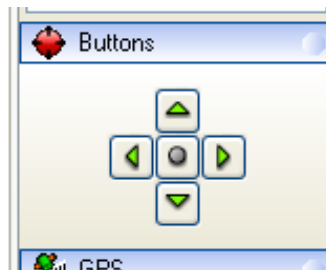
Now let's hide the same image when the user pushes down. To do this we click on the *OnDown* tab in the events windows and drag the same image object into that event. Then we type a "." and select *hide*.



Before we finish there is one more thing to do, if you remember there are properties on the button object as well. These are labels to help the user know what each button does. They will show up on the player under the *Button Help* entry in the player's main menu. You can write whatever you want into these but remember that the player's screen is only quite small and not much text will fit. To test this lets run it in the tester. To do this click on the tester button on the tool bar at the top of the maker.



In the tester you can pretend to press buttons by using the little buttons on the right of the tester. Try pressing the up and down arrows.



Congratulations you have now built a Mediascape that uses buttons! Other things that people often use buttons for is playing and pausing a piece of audio, showing the map display, or showing some kind of help page for the Mediascape the user is currently in. These are all shown in the example Mediascape that I have uploaded here: But I'm sure you can think of better uses for buttons!

[Learn how to use buttons to implement the controls Pause, Resume and Repeat the last instruction.](#)

Pause, Resume and Repeat the Last Instructions

Pause and Resume

We have found that people appreciate the ability to pause and restart a Mediascape. This is so that they can control interruptions, talk to friends and pace themselves. The other frequently used control that people like to use is the ability to repeat the last thing that was played. This is particularly important on missions or games which require you to go to a certain place or perform some action. If people miss the instruction or are interrupted and want to resume then they want to be able to replay it.

All of these capabilities are easy to implement if you use the Playlist in your Mediascape. To pause simply call `Playlist.Pause()`; in a script window. To resume call `Playlist.Resume()`; The most common way to add these controls is to map them to hardware buttons [How To: Use hardware buttons in your Mediascape](#) It is also a good idea to have a message on the screen to say that the Mediascape is Paused and which button to press to resume it. Assuming you have an image called Paused in your Mediascape then adding the following code to one of the cursor buttons will allow you to Pause and Resume your Mediascape.

```
if (Playlist.Status == PlaylistStatus.Paused)
{
    paused.Hide();
    Playlist.Resume();
}
else
{
    Playlist.Pause();
    paused.Show();
}
```

Repeat Last Instruction

It is easiest to make this a function of pressing one of the hardware buttons. You simply need to add the following two lines of code to the script of one of the cursor buttons.

```
Playlist.Stop();
Playlist.Resume();
```

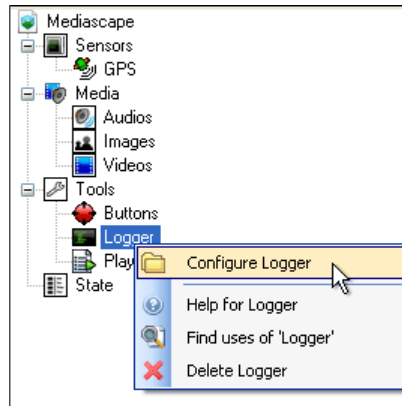
Stopping and resuming the playlist will play the last thing that was in the buffer. If nothing is currently playing it will simply play the last thing, if something is playing it will stop it and start it again from the beginning.

How to Use the Logger

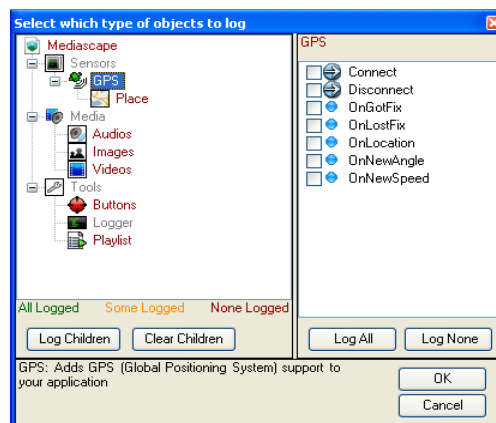
The Logger is used to output data, collected during the running of an Mscape, to a text file. This will generally be used to debug an Mscape or to collect information about a user's experience, for example [recording a trace of the route they followed](#).

Setting Up The Logger

Right click on the Logger object and select *Configure Logger*



The left-hand-side of the dialog box that opens will list all the objects you have added to your Mediascape (e.g. maps, audios etc.). When you click on an object all the actions that can be logged for that object will be displayed in the right-hand-column.



If you wish to log a particular event place a click in the relevant box in the right-hand column. For example, if you were concerned about the reliability of GPS coverage you might want to tick 'OnGotFix' and 'OnLostFix'. You would also be likely to want to know where this event happened so you would also want to [record a trace](#).

Short-cut buttons are provided to speed up this selection process: In the left-hand column 'Log Children' will place a tick in all the events for the selected object and 'Clear Children' will remove them. The 'Log All' and 'Log None' in the right hand column have the same effect.

Accessing The Log

log.txt - When you run your Mediascape the log will be saved as 'log.txt' in a '_Logger' subfolder within your Mediascapes content folder. Each line of the text file is comma delimited and will start with the time a particular event was recorded. The next two elements will show which object triggered the log. A letter will then follow ('E' for 'event' or 'A' for 'action'), then the event name and finally any data associated with that event (e.g. coordinate data).

The output window - Since it is often used for debugging purposes the log is displayed in the output window of the Mscape tester. This can be especially useful if you are writing to the log programmatically.

Related Articles

[Recording a trace of the route they followed](#)
[Using the Logger in code](#)

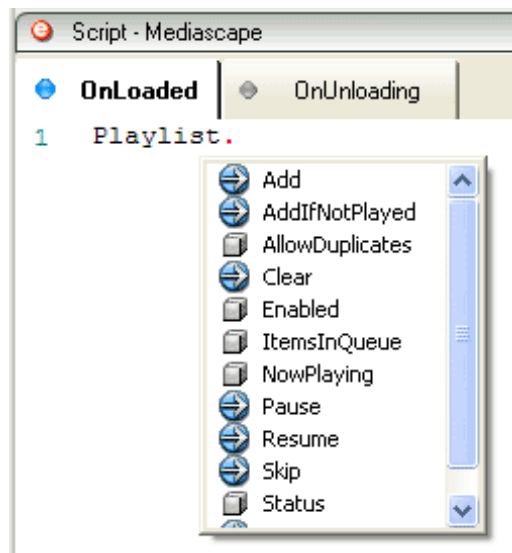
How to Use the Playlist

The playlist is a very useful tool that allows you to queue media objects so that they get played in the order that they are added to the playlist. It appears under the Tools menu. (To add a playlist right click on Tools).

The playlist has several important properties and actions.

1. The Allow Duplicates property determines whether you can add the same audio to the playlist or not. If it is set to false then if an audio is already playing or on the playlist it will not add it again.

2. The playlist actions are shown when you add the playlist object to the script window



Particularly Useful Actions Are

AddIfNotPlayed – this ensures that you only ever play a media item once. It is really useful if you want to prevent the same thing being played over and over every time you re-enter a region. GPS can often jump in and out of a region even if the user is stationary so it is a good idea to use AddIfNotPlayed to avoid a frustrating experience.

Clear – clearing the playlist will remove all items in the queue but will not stop the currently playing item. It is most often used on Exit from a region so that the media retains its relevance to a location. Please read the following example to learn how to use the Playlist effectively. [EffectiveUseOfPlaylist.](#)

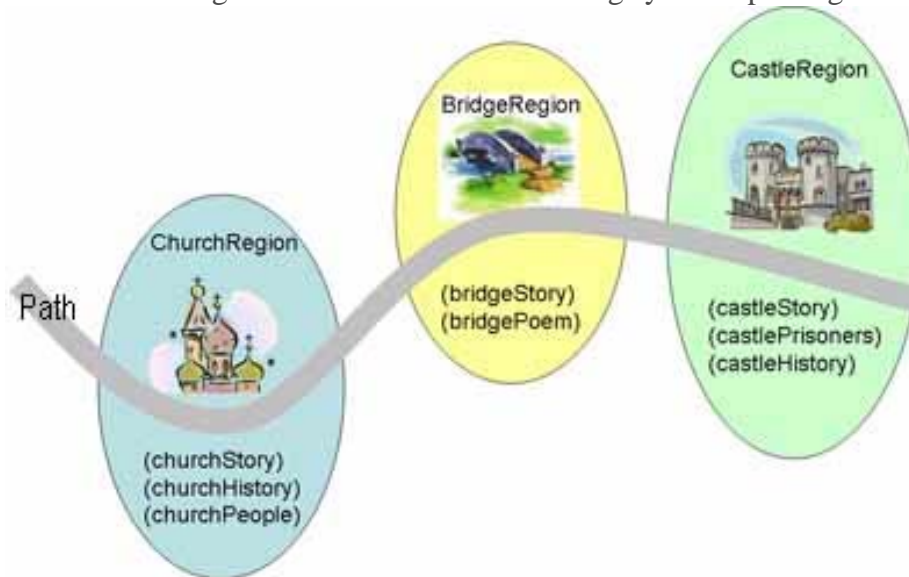
Pause, Resume and Repeat last instruction. These are most commonly mapped to hardware buttons to give the user more control over their experience. Please read the following example to learn how to implement Pause, Resume and Repeat the last instruction. [PauseResumeSkip.](#)

Remember not to put a looped piece of audio on the playlist as it will never finish and so anything queued after it will never play

Effective Use of a Playlist

Playlists should be used for car based journeys or walks so that you can layer information and retain its relevance to a particular location.

Imagine an audio walk or drive along a road with stories about things you are passing.



The figure shows a walk/drive design. There are basically three main regions related to a church, a bridge and a castle. As you walk by the church you will hear the story of the church. Then if you stay in the Church Region you can hear the history of the church and the people involved with the church. If you leave the region before hearing about the church history or its people then you will not hear them unless you return to that area. As you walk onwards down the road you will enter the bridge region. If you are going so fast that you are still listening to the church story while going through the bridge region then you will get to the castle without ever hearing about the bridge. While you are by the castle you will hear about the castle. However if you took your time (or in a driving scenario if you were in slow traffic) you would hear all of the stories about the bridge while you could still see the bridge and then you would hear all about the castle.

How To Implement This Experience

Create your maplib of the walk area and import it into your new Mediascape [Maps in Mediascapes](#)

Add the three regions to the map – ChurchRegion, BridgeRegion and CastleRegion.

Record the audio for the various stories – audio file names are shown in parentheses eg (churchStory).

Import the audio files into your Mediascape. [How to: Import Media](#)

Select ChurchRegion. On the OnEnter script type the following

```
Playlist.AddIfNotPlayed(churchStory);
Playlist.AddIfNotPlayed(churchHistory);
Playlist.AddIfNotPlayed(churchPeople);
```

Click the OnExit tab of the ChurchRegion script. Type the following in the script window.

```
Playlist.Clear();
```

Select the BridgeRegion. On the OnEnter script type the following

```
Playlist.AddIfNotPlayed(bridgeStory);  
Playlist.AddIfNotPlayed(bridgePoem);
```

Click the OnExit tab of the BridgeRegion script. Type the following in the script window.

```
Playlist.Clear();
```

Select the CastleRegion. On the OnEnter script type the following

```
Playlist.AddIfNotPlayed(castleStory);  
Playlist.AddIfNotPlayed(castlePrisoners);  
Playlist.AddIfNotPlayed(castleHistory);
```

Click the OnExit tab of the CastleRegion script. Type the following in the script window.

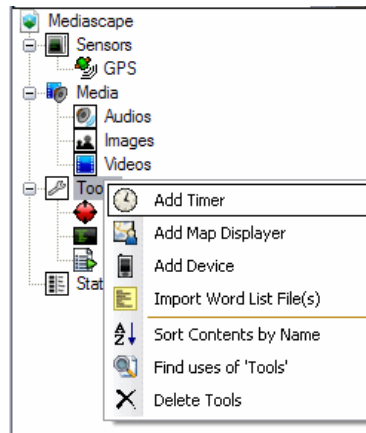
```
Playlist.Clear();
```

That is it.

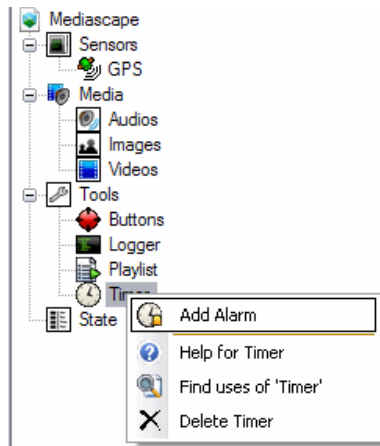
How to Use a Timer

Imagine making a Mediascape in which you want something to happen after a period of time. For example, you might want to end a game after the user has been playing for ten minutes, or show an image five seconds after the user has entered a region. It is easy to do this in Mscape by using the *Timer* object. This brief tutorial will show you how.

Let's begin by [starting the Mscape maker and creating a new Mediascape](#). The next step is to right click on the *Tools* section and select *Add Timer*



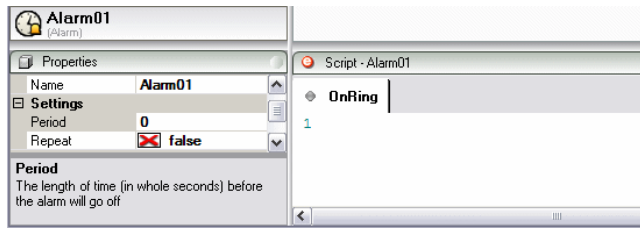
Now right click on the new *Timer* object and select *Add Alarm*



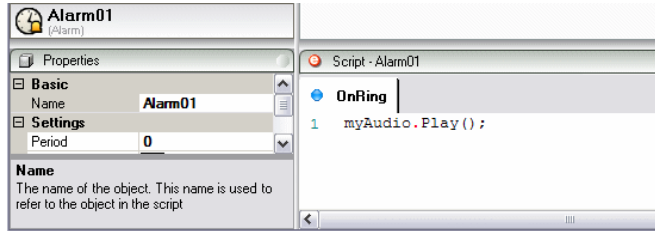
This adds a new *alarm* object to the Mediascape called *Alarm01*.

An alarm object has the following important characteristics:

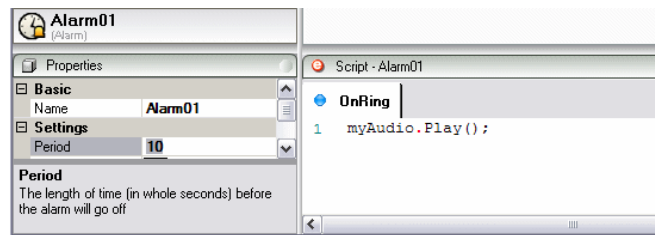
- An event, *OnRing* which will fire when the timer is ready
- A property, *Period*, which specifies the number of (whole) seconds before the timer will fire its *OnRing* event
- Methods to *Start* and *Stop* the timer
- Let's give the timer something to do when it fires. First, we need some media. Follow the guide to [importing media](#) to add an audio object and name it *myAudio*. Now click on the *Alarm01* object and look at the lower part of the maker where you will see that the script window currently points at the alarm's *OnRing* event:



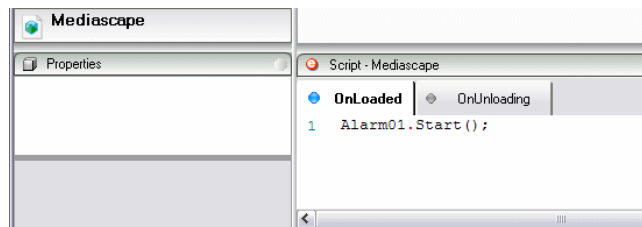
Drag the *myAudio* object into this window and drop it. Then type "." and select *Play* from the drop down list. Alternatively, click inside the empty script window and type *myAudio.Play()*; Either way, you should now see this:



Next we want to set the period for the alarm. To the left of the script window, you should see *Alarm01's* properties. Click on the cell next to *Period* and type *10*:



Now, all that is left is to set the alarm. Click on the *Mediascape* object. You should be able to see the *OnLoaded* event in the script window. Drag and drop the *Alarm01* object into this window, type "." and select *Start* from the drop down list. Alternatively type *Alarm01.Start()*; directly into the script window.



That's it! Now we have a simple Mediascape consisting of a timer with a single alarm that will fire 10 seconds after the Mediascape is loaded and play a short audio. If you have been following along in the Mscape maker, try your Mediascape now in the tester.

Try right-clicking on the *Alarm01* object and selecting *Help* to find out what else you can do with a timer alarm. Experiment with starting an alarm in the *onEnter* event in a *region*, or define multiple alarms... See what it can do, and have fun!

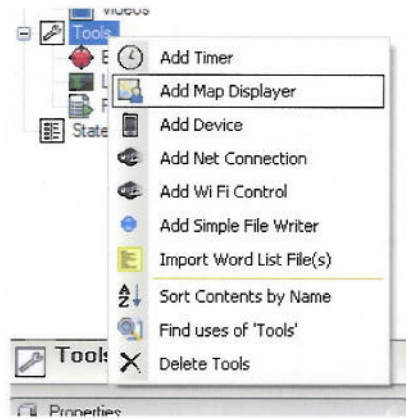
How to Show the User's Position on Screen

Sometimes you may want to show the user's position on screen as part of your Mediascape. In Mscape it is possible to show the user on a map, and to display and hide the map as and when required.

This is done using the *MapDisplayer* object.

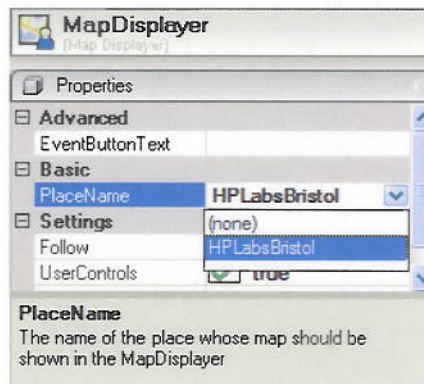
- Open Mscape maker, and create a new Mediascape.
- Import a map, or create a new map using the map service
- Right-click on *Tools* on the left hand panel, and select *Add Map Displayer*

Map Service will not be available after March 31, 2010



You need to tell the Map Displayer which map it should display. It is possible to show the user a different map to that used for designing the Mediascape.

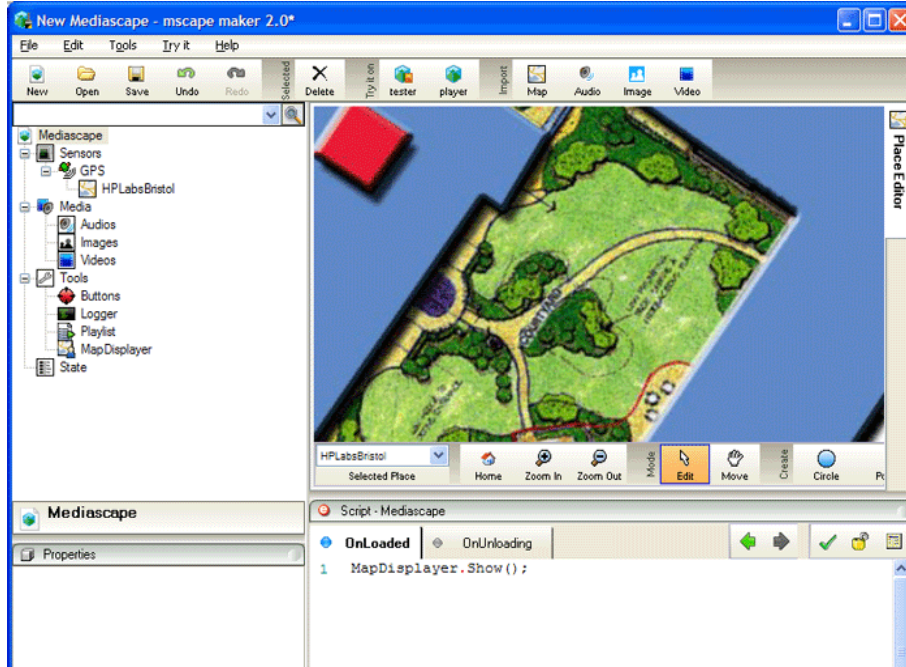
- Ensure that MapDisplayer is selected in the left-hand panel.
- In the properties box, click the drop-down box next to *PlaceName* and select your map



You do need to tell Mscape when you want the MapDisplayer to be shown on the screen, as it will not be shown by default. If you want the map to be shown when the Mediascape starts, click on *Mediascape* at the top of the list on the left, and select the *OnLoaded* event in the script window.

- Drag-and-drop the MapDisplayer object into the *OnLoaded* window.
- Type a . (period) character to bring up the list of actions and events available on the MapDisplayer? .
- Choose *Show*

HOW TO SHOW THE USER'S POSITION ON SCREEN

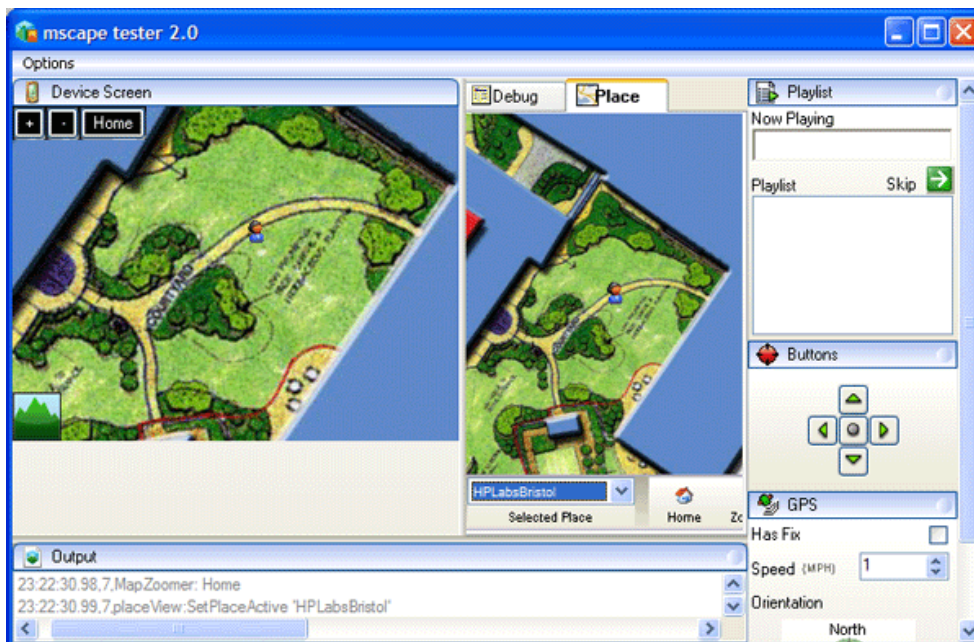


To hide the MapDisplayler, drag-and-drop the MapDisplayler object to the script window, type . and select Hide

- Click *Try it on.. Tester* and the Mediascape will open in Mscapemaker.

You'll see that the MapDisplayler is now visible in the Mobile Device screen. If you click on the map in central panel, you'll see the user's position is shown on both the central panel and the MapDisplayler.

The MapDisplayler will show a zoomed-in version of the map by default; if the user gets near the edge of the screen, the view will shift to keep them on the screen.

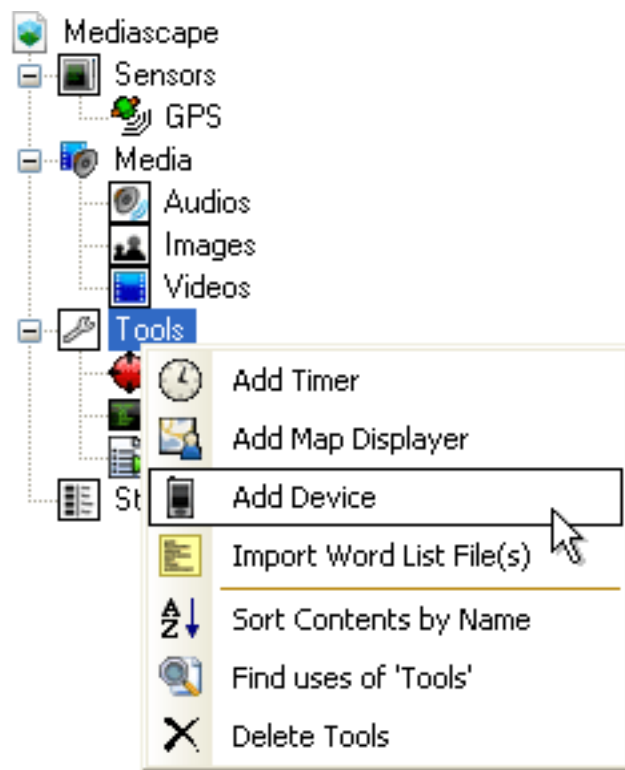


Device

This tool gives access to information about the type and state of a device your Mediascape is running on. In the first instance it could be used to deliver media more appropriate to a particular device, perhaps because of limited screen resolution. It can also report battery levels and memory usage, giving you the opportunity to take appropriate action: perhaps displaying a warning message, or automatically saving data.

Using Device

Note that in a new Mediascape the Device tool is not available by default. If you need to access device information you first need to add it: right-click on 'Tools' and click 'Add Device':



The properties and actions available under Device are mostly self-explanatory and are reasonably well-documented in the Mediascape toolkit help. As well as the OnPowerLow and OnPowerCritical events it is possible to access the battery level using the BatteryRemaining property. This could be tested at a given event, or regularly using a repeating timer. The same applies to memory properties.

Note that MemoryLoad is a percentage where 100 means all available memory is in use and 0 that none is in use.

How to Use the Word List

The Word List tool allows you to check whether a particular word is present in an external file which contains a list of words split onto separate lines.

The Text File

Obviously to use this tool the first thing you will need is a text file containing a list of words. Each new line in this file will be treated as a separate word so might look something like this:

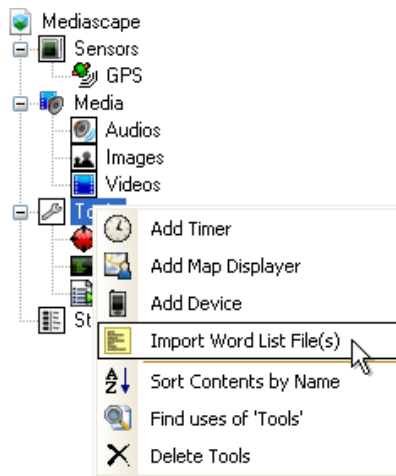
Fee
Fi
Foo
Fum

Each line need not contain a single word: it could contain a pair of words separated by a space or a complete sentence; however the text you test against this will have to match exactly in order for it to be found by the Word List tool. Note that this means a word in your list that includes a trailing space would not be found if you searched for the word without the space: i.e. "fee" is not the same as "fee ".

To import the wordlist into the Mediascape toolkit it will need to have a .txt extension.

Using The Word List

Right click on Tools and select 'Import Word List File(s)'. Note that multiple files can be imported at once:



When you want to query a word list in your Mediascape you must first load it using the Load action:

```
MyWordList.Load();
```

You might choose to do this when your Mediascape first loads or at a given event. Until this load action is carried out all searches will fail and the length of the list will be zero. Once loaded you can check the length of the list with the 'ListLength' property. To test if a word is present in the list use the WordExists action in a condition:

```

if (MyWordList.WordExists("fee")) {
    // do this if the word is in the list
}
else {
    // otherwise do this
}

```

This will check if the word "fee" (without "inverted commas") exists in the loaded text file. Note that searches are case sensitive and, as mentioned above; trailing spaces could cause the search to fail.

Loading New Lists

It is possible to work with a single Word List object and use this to test against multiple files by changing the 'Datafile' property and reloading the file. This might be preferable to using multiple Word List objects if your text files are large and could therefore impact on memory usage. As an example let's say you have two text files you need to check for the presence of the word "slug". You could check against both files using a single Word List object as follows:

```

MyTrueFalseVariable.Value = false;
MyWordList.Datafile = "TextFile1.txt";
MyWordList.Load();
if (MyWordList.WordExists("slug")) {
    MyTrueFalseVariable.Value = true;
}
MyWordList.Datafile = "TextFile2.txt";
MyWordList.Load();
if (MyWordList.WordExists("slug")) {
    MyTrueFalseVariable.Value = true;
}

```

If the word "slug" is found in either 'TextFile1.txt' or 'TextFile2.txt' 'MyTrueFalseVariable' will be set to true (this was used as an example, alternatively an event could be triggered). Note that the second 'Load()' is essential for this to work, otherwise the second search would still be carried out on 'TextFile1.txt'.

This approach could be extended further by using a loop to check for the presence of a word in a large number of text files (i.e. using an array of file names). Also, with a properly formed search term, it might even be possible to check against the contents of a log file...

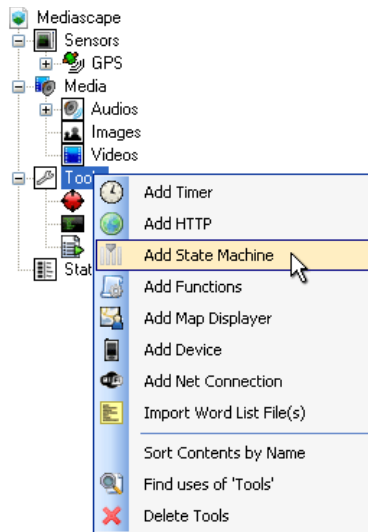
Note that when assigning the file name using the 'Datafile' property you are able to load text files with extensions other than '.txt'.

StateMachine

The StateMachine is used to execute a block of code at specific points in your Mediascape. Whilst it would be possible to duplicate its effects, for example by putting code on button presses or hotspots, or by using a function, the StateMachine avoids potential duplication of code and also provides additional functionality designed specifically for the task.

Adding A StateMachine

Right click on Tools and select StateMachine:



The StateMachine has no editable properties.

Adding And Using States

Once you have added the StateMachine you then need to add an appropriate number of 'Mstates'.

These correspond to each significant 'state' in your Mediascape and have associated 'OnEnter' and 'OnLeave' events. To add an Mstate right click on StateMachine and select 'Add Mstate'.

Let's examine what you might want to add an Mstate for. A first obvious choice might be an 'introduction' state in which you set the default values of several variables used later in your Mediascape, play a specific audio track and show a relevant image. You may also want to disable your map regions until the introduction has finished and the Mediascape begun. On your 'introduction' state's OnEnter event you might add the following code:

```
score.Value = 0;
numberOfMonsters.Value = 1;
introScreen.Show();
SpookyAudio1.Play();
map.Enabled = false;
```

Now when the player finishes the game and starts again, you simply need to switch to the introduction state (see below) to reset the variables and play the introduction. This immediately offers an advantage over the Mediascape OnLoaded event since that only gets executed once, whilst you can call your introduction state as and when required.

You could then add further states as appropriate. These might correspond to game levels, floor levels in a virtual building, and maybe different periods in history; basically whatever is relevant to your Mediascape.

State Events

Each state has an OnEnter and OnLeave event to which code specific to that state can be added. This could do all manner of things:

- set variables
- play/stop audio, video etc.
- activate or deactivate map regions
- switch to a different map altogether
- test existing variables (e.g. check the user has accrued enough experience points to go to the next level and send them back to the previous level/state if not)
- etc...

The StateMachine OnEnter Event

Note that the StateMachine itself has an OnEnter event. This is triggered each time a state change happens so would be used for code that should be executed every time, regardless of which state is being entered. This can be used to avoid having to update duplicated code on every Mstate event.

Switching States

Obviously once you have set up your Mstates you need to be able to switch between them. The simplest way to achieve this is by typing the name of the Mstate you want to trigger followed by a full-stop followed by the *Enter()* action:

```
myIntroState.Enter();
```

Note that it is also possible to change state by invoking the StateMachine's *Enter()* action, though in this case you must specify the name of the required state between the brackets:

```
StateMachine.Enter(myIntroState);
```

When you switch to a new state the previous state's OnLeave event will be triggered as well as the new state's OnEnter event.

Checking The Current State

There are times you may need to check what the currently active state is, or if a specific state is currently active. For example if a region is active in all states you may wish to change the behavior of that region depending on the current state. In this case you would check the value of the StateMachine's `ActiveState`. In this example a text variable is set to a different value depending on the current state:

```
if (StateMachine.ActiveState == "level1") {
    currentLevel.Value = "level1";
}
else if (StateMachine.ActiveState == "level2") {
    currentLevel.Value = "level2";
}
```

Alternatively you might only want to check whether a specific state is active in which case you would check against that state's `Active` boolean property. In this example an audio file will play if the 'ending' Mstate is currently active:

```
if(ending.Active) {
    scarySound.Play();
}
```

Note that since it is a boolean property (i.e. true/false), the condition can be abbreviated from **`if(ending.Active == true)`** to **`if(ending.Active)`**.

The ability to check the currently active state is an additional advantage that the StateMachine has over using [functions](#).

How to Tell if You Are Online

Introduction

The NetConnection object is designed to provide useful information about the state of the device's network connection. You can use it to tell if you are currently connected to the internet and what your IP address is.

To create the NetConnection object right click on the Tools entry in the left hand pane of the Mscapemaker. Then select "Add Net Connection". This object has two events that you can use to do things in the Mscapemaker. It also has a number of properties that allow you to set options and read the current state of the network, the properties are in the bottom left of the maker.

Setting Options On The NetConnection Object

Destination Property

This object checks to see if you are online by going to a URL and asking for a little bit of the page at that URL to see if it gets a reply. It does not download the full page but it does need a valid page at that address. Which page it goes to and how often it does this are both set using properties. To change the properties of this object select NetConnection in the left hand panel in the maker. The properties will then be visible in the bottom left of the maker. The URL that is checked is set in the Destination property, you can set it to any page that you expect to be able to reach and think will be reliable. If you are using a private wireless network then make sure that you can get to this page from your private network.

PeriodBetweenTests Property

A tiny part of this page will be requested frequently. You can set how often by changing the "PeriodBetweenTests" property. This sets the number of seconds between the beginning of each request. Each request might take some time so you should make sure that the gap between checks is longer than the expected time for each check. Also as this test will use some data, if you expect your users to be paying for their data then you should set this property to make sure you do not use too much of their data. Each request should only be a few hundred bytes but if you check very often then this can quickly add up. If two successive tests fail then the object will now consider itself no longer connected.

Timeout Property

Each time that a request is made it will take some time for the tiny bit of the web page to come back to the player. The timeout property sets how long in seconds it will wait before considering that page unreachable and the device no longer connected. If the value of this property is too high then it will take longer for the Mscapemaker player to realize that it is no longer connected, if it's too low then you might actually be connected but still think that you are not.

Getting The Status Of The Network Connection

The NetConnection object has two properties that may only be read while the Mediascape is running:

ConnectedToDestination Property

This property will be true if you are currently connected to the network.

FirstIPAddress Property

This property will return the first IP address allocated to the current device that does not have the form 127.0.0.X (local loopback), 0.X.X.X (invalid) or 169.X.X.X (reserved). On most devices this will be the IP address used to contact that device.

Responding To Network Events

The NetConnection object has two events that signal changes to the network status:

GotConnectionToDestination Event

This event will be triggered when you were offline and then go online. This will trigger the first time that you got a successful reply from your destination.

LostConnectionToDestination Event

This will be triggered if two successive requests for the little bit of the specified web page failed.

Using Functions

Functions are useful when you find yourself copying and pasting identical pieces of event code into multiple regions (or other resources). This works fine until you realize you've made a mistake in that code, because then you need to go through each and every copy you've made to fix it.

The *Functions* object in Mscape allows you to write the functionality you need once, and then simply call it from each of the places in your Mediascape where it is needed. This way if you want to change the way your piece of functionality works, perhaps to play a different sound, or to award more points, you need only make the change in a single place.

A Simple Example

Imagine you have a game which is about collecting coins, where each coin is represented by a region. You walk into the region, a 'coin' sound is played and your score is updated. Now imagine that there are 100 of these regions - copying and pasting the code for this each time you want to make a change would be a real chore.

Let's see what we can do with functions.

- Create a new Mediascape
- Import a map
- Draw yourself a bunch of regions.
- Download [this sound effect](#) (right-click, Save As), and then import it into Mscape maker - it should appear in your maker as 'sndCoin'.
- Add a variable to hold the score. Right-click 'State' and choose 'Add Number' - call the new number 'score'.
- Right-click on Tools and Select 'Add Functions'
- Right-click on the newly-added Functions object and select 'Add Function'
- Call the new function 'fnCoin'.
- Select fnCoin and type the following into the 'Function' event - this is the event that will be triggered when the function is run.

```
sndCoin.Play();
score.Value++;
Logger.Log("Your score is now " + score.Value);
```

- Now we need to call the function from each of your regions.
- Select the first region and type..

```
fnCoin.Run();
```

- Right-click that region in the left-hand panel and choose 'Copy Behavior'.
- Right-click each other region in turn and select 'Paste Behavior'

Run this in the tester to try it out.

USING FUNCTIONS

You might be thinking *ok, that's fine, but I still had to copy and paste behavior to each region?* That's true but what if you decide later on that 10 points should be awarded for each coin pickup?

Previously you'd have had to go through every single region and change the code - or change it once and repeat the copy-paste process we did before. When using functions we can select `fnCoin` and change the code once to read.

```
sndCoin.Play();
score.Value += 10; // <-- change here.
Logger.Log("Your score is now " + score.Value);
Handy huh?
```

Functions and Parameters

There are two ways of running a function - `fnCoin.Run()` and `fnCoin.RunWithParams(..)`.

In the previous example we used `.Run()` - and the reason for this is that every single region behaved in exactly the same way. However, if we wanted to make some coins worth more than others we'd need to use the `.RunWithParams()`.

`RunWithParams()` runs the function but additionally passes in a list of optional *parameters*.

Parameters are simply a way of passing a piece of data from the place where the function is called (e.g. the region) into the function's code (e.g. where the points are awarded) Here's what we'd do for the coins example.

```
In region 1 (small money).
fnCoin.RunWithParams(5);
In region 2 (big money).
fnCoin.RunWithParams(100);
In fnCoin.
sndCoin.Play();
```

```
// Take the first parameter, and convert to a Double (a number).
```

```
// Remember that computers start counting at 0 - so parameter 0 is the first one, 1 is the second etc.
```

```
score.Value += Parameters.GetNumberAt(0);
Logger.Log("Your score is now " + score.Value);
```

Multiple Parameters

You can actually pass in any number of parameters into the `.RunWithParams` call - each parameter should be divided by a comma, e.g. `RunWithParams? (10, "a", "b");`

You'd then retrieve these in the function itself using `GetNumberAt? (0)`, `GetStringAt? (1)`, `GetStringAt? (2)`. You need to match the `.GetXAt` call to the type of the parameter - e.g. parameter 0 is a number, so we use `.GetNumberAt`, while parameter 1 and 2 are Strings so we use `.GetStringAt`. If you get this wrong you'll see an error in the output window in Mscage tester like 'The parameter at 2 is not a string, it's a Double' (Double is what the scripting language calls a number). Parameters will return 0 or an empty string if something goes wrong like this.

Make sure that every time you call MyFunction you pass in the same number of parameters, and in the same order. If not you'll find that your function doesn't behave as you expect.

Passing In Mediascape Objects And Media Items

It's actually possible to pass anything in as a parameter - for example you can pass in a media object such as an Audio. This is handy if your regions all have identical behavior *except* they play different audios.

E.g. *In region A's OnEnter event*

```
MyRegionFunction.RunWithParams(audio1)
```

In region B's OnEnter event

```
MyRegionFunction.RunWithParams(audio2)
```

MyRegionFunction Definition

```
// Do things that happen on every region entry..
```

```
Audio a = (Audio)Parameters.GetObjectAt(0);
a.Play();
```

The fiddly part here is that you need to *cast* the object that's returned from the GetObjectAt method into the type of object you require. This is done by pre-pending the name of the type of object in round brackets to the GetObjectAt? call, e.g (Audio)Parameters.GetObjectAt(x) and assigning it to a new local variable (Audio a = ...)

This technique works for any object type in Mscape, e.g. Image, Video, CircleRegion , PolygonRegion , simply replace the word Audio with that of the type you wish to use.

One thing to note is that the popup list of properties and actions available on an object does not show when you type the Using this method, so it's best to check the names of the actions or properties you want to run by selecting an object of that type (e.g. click on an audio) and looking in the properties, or type the name an object and hit '.'

Here's an example where the radius of a CircleRegion is used to assign points scored by a user - bigger regions score more points. You'll need to add a Number variable called 'score' for this.

E.g. *In region A's OnEnter event*

```
MyRegionFunction.RunWithParams(me) // me is the region that was entered
```

MyRegionFunction Definition

```
CircleRegion r = (CircleRegion)Parameters.GetObjectAt(0);
score.Value += (int)r.Radius / 10 + 1; // round off the radius value to an integer, and divide by 10
Logger.Log("Score is " + score.Value);
```

How to Communicate with the Outside World

How to tell if you are online

How to get information from the outside world

How to tell the world something

How to get images, audio and flash from a server

How to Get Information from the Outside World

Introduction

Most mscares use a person's own private sensors (such as GPS and RFID) and history (i.e. where they have been) to deliver the right thing in the right place and moment. You can now also use other sources of information to do this, such as the local weather, or someone else's position. You could even store information online and change it every day without having to change the Mscare. This help page will guide you through hooking up your Mscare to a network. I.e. getting information from the internet and using it within a Mscare. You can also download content such as images but that will be covered in a separate help page.

In this example we will build a very simple Mscare that finds the name of the nearest town or city.

HttpRequester Object

This is the object that is used to download things from the internet. You can make as many of these as you need. It is best to make one object for each type of request you are going to make. So for example you might make one to get the local traffic information and another to get the location of other players. For this example we will just use one Http Requester object. To make a Http Requester object right click on tools and add the *Http* object. Then right click on that object and select add Http Requester. You will now have an object that can fetch things from the Internet.

Making a simple request

The Http Requester object has a number of methods, most of them will cause it to query a web server and then return the requested data at some point. For this example we will use the most simple of those methods with is Get. For this example we will wait until the GPS has a fix and then use its first location to send a request for the nearest town or city name.

To do this, start the Mscare maker and then add the Http Requester object by adding Http to the tools section and then a Http Requester. Rename the Http Requester object to NameGetter. We will also need a TrueFalse called madeRequest, this can be added by right clicking on the State object in the left hand pane of the maker, make sure that MadeRequest starts false. All the other objects that we need are already there.

We now need to wait until there is a valid GPS location that we can use to look up the place name. On the GPS'sOnLocation Event add the following code:

```
if(MadeRequest.Value == false)
{
    MadeRequest.Value = true;

    NameGetter.Get("http://ws.geonames.org/findNearbyPlaceName?lat=" + GPS.Latitude + "&lng=" + GPS.Longitude);

    // You will * NOT* have the result by now.
}
```

This will only be run once, since once we get our first OnLocation event then we change the TrueFalse to true and it will not run again. The line NameGetter.get..... is the line that causes the player to go off and make the request.

The actual request is made up out of two things, the URL of the web service that we want to talk to and some extra information about where we are. This is done by adding some text together to make a single longer bit of text that forms the final URL that goes out to the server. This request might take some time to complete, it could take several seconds. The rest of the Mscrape will keep running while this happens. So you should not assume in any line of script after this that you have automatically got the result. We will deal with the result in the next section of this example.

Dealing with the response from a request

Once we have made the initial request, the system will go off and talk to the outside world and wait for it to respond. Once it does respond one of two events will be triggered, either OnResponse or OnResponseFailed. If the system cannot recover the data that was requested then the OnResponseFailed event will be triggered. If the request succeeded then the OnResponse event will be triggered. For this example we will assume that the request works.

The OnResponse event has two parameters, these are extra bits of information that can only be used during that event handler. One of these parameters is called IncomingData. This parameter is an object with all the information that came back from the request. To read the data that has come back from the request we call methods on the IncommingData object. In this example we are after the place name for this location.

A good place to start in understanding how to make sense of the data you have received and how to use it, is to look at the data that you get back in a web browser. For this example I will use a known location and just type the URL into a web browser to see what the data looks like. I will use this URL:

<http://ws.geonames.org/findNearbyPlaceName?lat=51.45&lng=-2.6>

The response from the server looks like this:

```
<geonames>
  <geoname>
    <name>Bristol</name>
    <lat>51.45</lat>
    <lng>-2.5833333</lng>
    <geonameId>2654675</geonameId>
    <countryCode>GB</countryCode>
    <countryName>United Kingdom</countryName>
    <fcl>P</fcl>
    <fcode>PPL</fcode>
    <distance>1.1549</distance>
  </geoname>
</geonames>
```

From this we only actually want the country name which, in this case, is "United Kingdom". The IncommingData parameter has the full result but there are a number of ways of getting just the bit we want.

The system can read three different types of data XML, JSON and URL encoded form data. They are all treated the same in the rest of this example unless it's specifically mentioned that they are different.

Using IncomingData.FindTextNamed

The IncomingData parameter from the OnResponse event has a method called FindTextNamed this allows you to specify what you want and the method will make its best guess at your request and return it to you as a text. The response has to have some data that exactly matches the name you have supplied but capitalization does not matter. The exact way the data is represented is not fixed for example if you use the line:

```
IncomingData.FindTextNamed("countryName");
```

Then it will match all of the follow situations, if more than one is found it will use the first one:

```
<anything name="countryName" value="United Kingdom"/>
```

```
<anything name="countryName">
  United Kingdom
</anything>
```

```
<countryName>
  United Kingdom
</countryName>
<anything countryName="United Kingdom"/>
```

If the data you want to read is relatively simple and you are sure it will not change in order then this is probably the easiest way to read and make sense of the incoming data. But it is not very processor efficiecent and therefore if you want to read lots of data from a very big response this might not be the best option.

There are also methods for getting numbers and TrueFalse from the Incoming data in the same way. It will do the same level of matching but instead of returning some text it will return the correct type of information.

Using IncomingData.ConvertDataTo

As well as reading a single item at a time as we did in the section above, it is also possible to read the entire response in one go and convert it into a group or list of state objects. For example, create a Group in the State section called *geoname* or *geonames* and add to that group a text object called *name*, another called *countryName* and finally a number called *distance*. Note that the group can be called anything you like but the state objects must match the terms used in the HTTP response. We can then call:

```
IncomingData.ConvertDataTo(geoname);
```

And all the values will be read in one go.

Sometimes, HTTP responses will contain lists of data rather than, say, a single geoname. Look at the URL <http://ws.geonames.org/postalCodeSearch?postalcode=9011&maxRows=10> for example, and

notice that there are ten elements starting with the tag `code`. Using a [StateList](#), we can read items from all ten elements in a single call. Create a [StateList](#) object called `code` (note that the name does matter here - it must match the name of the repeated element in the response) and add a text object called `postal code`. Call

```
IncomingData.ConvertDataTo(code);
```

and notice that the [StateList](#) now contains ten items each with a different postcode.

You can find out more about making a list of state objects here: [StateList](#). Note that the player is making a best-guess effort to match the structure of the [StateList](#) in the incoming data. Be careful - it is easy to get the structure slightly wrong and miss data. Also note that converting incoming data again will append any new values to the [StateList](#). If you want to replace existing values, then simply clear the [StateList](#) first. Finally, it is worth noting that if the original data is url encoded form data then converting to a [StateList](#) will not work.

Using IncomingData.FindTextByPath

This method uses technology called XPath to find the right thing within a response from a server. It is a very powerful but also rather complicated way to find what you want. If you think that the above two methods will not be sufficient for your requirements then you should do some independent reading about XPath and try that as a technique.

There are a few changes to normal XPath in the Mscape system:

- Default namespaces are completely ignored.
- Other namespaces are not yet supported.
- You can do XPath on JSON data but you need to start any full paths with `"/json/"`.
- If you want to navigate with a json list you will need to add another step in the path, each node within the list will be called the same name as the name of the list.
- You can do XPath on form data but any full paths will have to start `"/form/"`.

Conclusion

Using the techniques described above you should now be able to make a simple request to a server and then make sense of the reply. You should be able to choose an approach to making sense of the data that is returned. I have deliberately not discussed what you could do with this data as that is up to you!

How to Tell the World Something

Introduction

There are many reasons why you may want to tell the outside world about what is going on within your Mediascape. For example if you are building a multiplayer game then you will need each player to inform a central system or all the other players of their location and activities. As another example if you have a Mediascape that is dependent on local weather then you need to tell the server where you are in order to get the correct weather information.

This example assumes that you have already read and understood at least the first half of the other example [How to get information from the outside world](#).

Using the URL to tell a server about yourself

This topic is briefly covered in this example: [How to get information](#). It is quite common to use extra information in the URL of the request to a server to express some conditions on the request. For example if you search on a search engine and look at the URL then you will find the your original search term in the URL.

To add this kind of extra information to a request you can simply add different bits of text and numbers together. For example if you have a game where each player has a Name text object and Score, Name, Latitude and Longitude number objects. If you wanted to send all this information in a query string to a server called <http://www.mymediascapegame.org> then you could use a line like this to add them together.

```
HttpRequester01.Get("http://www.mymediascapegame.org/?name=" + Name.Value + "&score=" + Score.Value + "&Lat=" + Latitude.Value + "&lng=" Longitude.Value);
```

This will add all the bits of text and numbers together. Using this technique you can add almost anything to a URL.

Posting data to a server

Another way of sending data to a server is to post the data. This is the same technique that is used when you fill in a form on the web and click submit. Sometimes you can even use this technique to convince a web server that it has just been posted to by a web page.

Doing a Post is very similar to doing a Get that is explained in detail here [How to get information from the outside world](#). However to start a post you use a different action on the HttpRequester object. You can post data from a [StateList](#) or from a StateGroup. To post a [StateList](#) you should use a line like this:

```
HttpRequester01.PostStateList("http://myserver.com/myMediascapeGame/",StateList01,SendFormat.xml);
```

This will post the full contents of the [StateList](#) called StateList01 to the web site myserver.com it will do so by converting the statelist into XML. There are three formats that can be used to convert lists and groups into something that can be sent, XML, Json and URL encoded form data. As an author you can specify the format by changing the send format in the PostStateList or PostGroup? methods.

Once the post is sent then the script will continue to run while the system transmits the data and waits for a reply. Once the Mscape player has received the reply the OnReceived event from the HttpRequester object that made the post will trigger.

If you do not want to post your data as XML, Json or form data, or if you have a very specific formatting requirement then you can also post the exact text that you want to be transmitted. This is done by calling the *PostText* method and supplying the url to post to and the payload.

Put and Delete

As well as Get which was discussed in another article and Post which is discussed in this one there are two other supported actions that you can ask of a server. You can Put data on a server, this is very similar to the Post methods. You can also call Delete on a server this is very similar to a Get method but the server will respond differently. If you do not understand Put or Delete then do not worry they are far less common than Get and Post. But if there is a need to use them you can.

How to Get Images, Audio and Flash from a Server

Introduction

In the previous two articles we talked about sending and receiving information, but not content. We sent and received numbers, words and TrueFalse's now we talk about downloading content, images, audio and flash files. We can only download and play files of a compatible type, we cannot download and play something that Mscap does not already support.

Getting the Content

The process of requesting content is exactly the same as the process of requesting anything else. You can retrieve content by using a Get, Post, Put or Delete. Once the OnResponse event has triggered you can use the InComingData parameter to save the downloaded file to an existing object of the same type. So for example if you download an image file then you can save it to an existing image object and then use it just like any other image object. If you get the type wrong or you try to save the downloaded file to a file currently playing (audio or flash) then the HTTP operation will succeed but you will not be able to save the file. To force a download to a file already playing then simply stop (the audio) or unload (the flash movie) the relevant script object before saving the file. If you try to save the file to an image object that is currently showing then you will need to hide and then show it again to reflect the change.

The new file will remain for the length of the Mediascape session or until it's saved over by a new download. However if the user loads a different Mediascape or quits the player then the new file will be lost and the original file will always be returned.

To save a downloaded file call the IncomingData.SaveDataTo method and pass in the media object that you want to replace. Once this is done then you can use the media object normally.

Variables

What Is A Variable

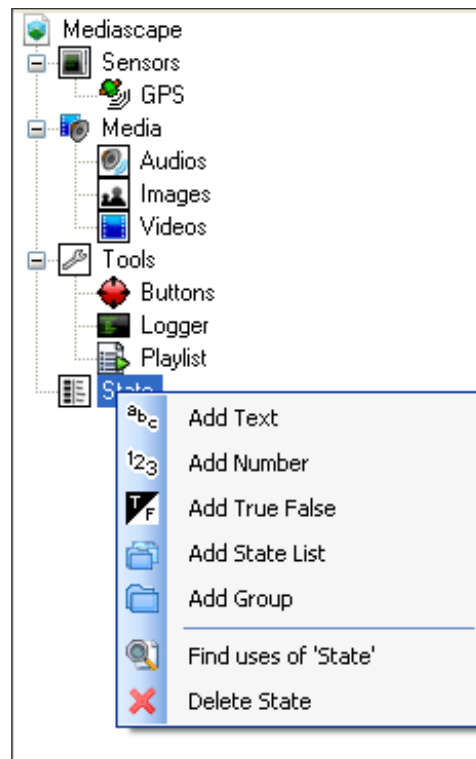
There are times when you will need to access information in your Mscape, for instance to make decisions on what happens when a particular event occurs. Some objects already contain information (known as properties) that you can access and use in this way. For example a Circle Region's *EnteredCount* tells you how many times a user has entered the region; so you could use this value to change which audio track is played to them the next time they enter it.

Sometimes however you will need to have access to information that is not stored in one of the existing objects, or you might want to store the state of an object at a particular time. For example you might want to know how many times the user went back to a particular region before they solved a puzzle. If they keep going back to the region after solving the puzzle the *EnteredCount* will keep going up, so you need to store this value separately.

To store this information Mscape needs to be specifically told it has to remember it and you do this by assigning the information to a variable. Think of a variable as a container for the information. Now, if you were stuck on a desert island and went on a foraging expedition you wouldn't be able to magically produce containers as and when you needed them - you would need to plan in advance and take them with you. The same is true of variables in Mscape. You need to decide what information you need to store and define the necessary variables in advance.

Variable Types

To define a variable right-click on the State button and a menu appears:



The question you might now ask yourself is "why so much choice"? Why not just one type of variable? To put it simply, computers have different types of containers depending on what you want to store in them. On your foraging expedition you wouldn't try and gather water in a basket, you would use a bottle. If you found some fruit it would go in the basket. The same concept applies when storing variables in Mscape. When you know the information you want to store is a number (e.g. coordinates, *EnteredCounts* etc.) you would create a Number variable. If it's a piece of text (e.g. a region's name) it would be stored in a Text variable and so on.

The reasons for this are twofold. The first is that the computer isn't good at knowing what kind of information you're giving it to store or how to handle it, so needs to be told. By telling it a variable is a number it then knows that, for example, if you add it to another number it should give you the sum.

The second is to ensure efficient use of the computer's physical memory. If you know a variable will only ever have two values you could store this as text ('on' or 'off', 'true' or 'false' etc.), or as a number ('0' or '1'). This is such a common requirement however that there is a specific type of variable - a boolean, or 'True False' in Mscape - designed to store it in the most efficient way possible. It also avoids the possibility for error, for example if you mistyped 'true' as 'treu'.

If you try and store information in the wrong kind of variable one of two things might happen: You will either get an error or the information will be converted to a different variable type. The latter is most likely to happen when you assign a number to a Text variable. This could become a problem if you then tried to add the variable to another number. Rather than giving you the sum of the two numbers the computer would treat them as text and combine them. For example where you might expect '2 + 1' to give a result of '3' the computer would give '21'.

This is a fairly simplistic overview of the more common variable types. [State Lists](#) are used to store lists of items (usually referred to as arrays) and the Group is an Mscape tool that allows you to save and load the values of variables to and from an external file.

Defining A Variable

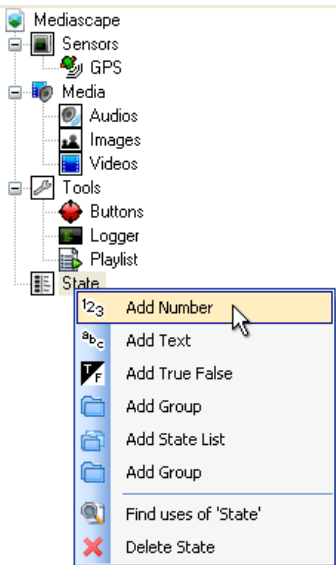
Once you have decided what type of variable you want to create right-click on State and select the type you want to add. With the variable selected, look at the properties below. It is a good idea to set the Name to something that meaningfully describes what the variable stores; and you may also find it useful to include what type of variable it is. You can also use this to set the starting Value of the variable.

Number Variables

This is a variable used to store a number.

Creating A Number Variable

Right click on State and select 'Add Number'.



In the properties it's a good idea to change the name to something meaningful (i.e. something that describes what the variable will store like 'playerAge', or 'currentX'). The default value is 0, but this can be changed in the properties.

Setting The Value With Code

Setting the value of a number variable in code is fairly straightforward:

```
myNumberVariable.Value = 42;
```

You start with the name of your number variable, add a full stop - at which point a pop-up should appear - add 'Value', then an equals sign. You then add the number you want to store and finish with a semi colon. You can also store decimal numbers:

```
myNumberVariable.Value = 1.61803;
```

And negative numbers:

```
myNumberVariable.Value = -17.77;
```

Accessing The Value Of A Number Variable

Simply reference the *Value* property of the variable. For example to add two number variables together and assign the result to a third number variable:

```
theResultNumber.Value = myFirstNumber.Value + mySecondNumber.Value;
```

A Note On Number Types

Programmers will be aware that storing numbers on a computer is more complex than it may seem. Mscap stores numbers as type *double*. In real terms this is unlikely to be significant for most people; though does help to explain why GPS coordinates have so many digits after the decimal point.

Related Articles

[Using the Math Library](#)

[Using Random Numbers](#)

True/False

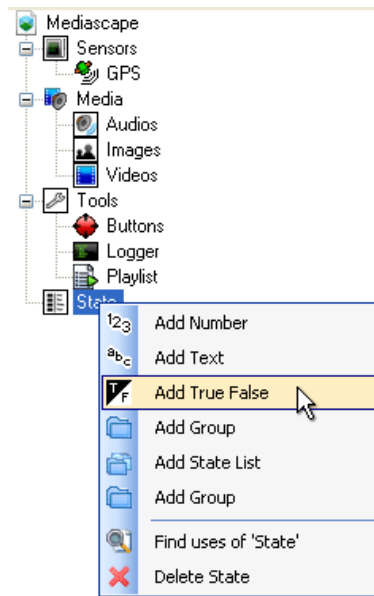
This is a specialized variable that can be thought of as a switch. It has only two states: true or false (on or off). It is of course possible to store equivalent information in either a text or number variable, but it is recommended that when appropriate a true/false variable is used for the following reasons:

- it is more efficient in terms of memory usage
- it is less prone to error (see below)
- it allows shortcuts to be used when testing its value

Note that this type of variable is more commonly referred to as a Boolean.

Creating A True/False Variable

Right click on State and select 'Add True False'.



In the properties change the name to something meaningful (i.e. something that describes what the variable will store like 'introComplete', or 'gameOver'). You can also set a default value for the variable.

Setting The Value With Code

Setting the value of a true/false variable in code is straightforward. In fact there are only two possible options:

```
myTrueFalseVariable.Value = true;
```

```
myTrueFalseVariable.Value = false;
```

You start with the name of your text variable, add a full stop - at which point a pop-up should appear - add 'Value', then an equals sign. You then set the variable to either *true* or *false*. Note the absence of

TRUE / FALSE

inverted commas. The value being stored in Mscape is **not** text, but one of two states understood by Mscape.

If you happen to make a typo in the state name Mscape will display an error message when you save or test your Mscape. If you had chosen to use a text variable to store a true/false state you would receive no such warning in the case of a typo and your Mscape might fail with no indication of where the problem lies. Note also that *true* and *false* appear blue in the code editor since they are recognized *keywords*.

Accessing The Value Of A Text Variable

Simply reference the Value property of the variable. The most likely reason to want to access a true/false variable is to test its value with a condition. In this case there are some handy shortcuts.

The usual long-hand methods of testing the state of true/false variables are as follows:

To test if the value is *true*:

```
if (myTrueFalseVariable.Value == true) {  
    // do this...  
}
```

To test if the value is *false*:

```
if (myTrueFalseVariable.Value == false) {  
    // do this...  
}
```

However to test whether a true/false variable is *true* you may omit the "==" true" from the condition:

```
if (myTrueFalseVariable.Value) {  
    // do this...  
}
```

There is also a shortcut method to test if a value is false by using the NOT operator (an exclamation mark: !), placing it in front of the variable reference as follows:

```
if (!myTrueFalseVariable.Value) {  
    // do this...  
}
```

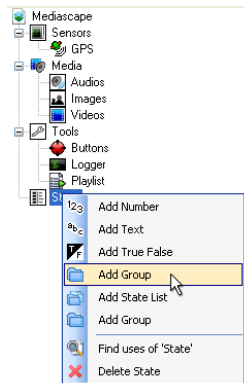
Effectively we're saying "if NOT myTrueFalseVariable" (i.e. it is *false*) then do the following.

Groups

A groups is not actually a variable but a tool used to organize variables and to store them in an external text file. This can be useful if you wish to save information for later use outside of a Mediascape, but also to allow a Mediascape to store its state when closed and then continue from when it left off when it is restarted.

Creating A Group

Right click on State and select 'Add Group'.

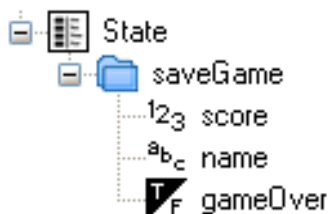


In the properties you can change the group name, set whether it 'AutoSaves' (see below) and set a filename that will be used for the external text file.

Adding Variables To A Group

On its own a group doesn't actually do anything. You need to first add variables to it. In this respect it works similarly to a *Folder*: you can click and drag existing variables into a group or right-click on the group and create new variables directly inside it. As such it might be a useful tool to organize large numbers of variables, but its true power lies in its ability to save and load the value of all variables stored within it.

In the example below a group called 'saveGame' has been added, which contains three variables: 'score' (number), 'name' (text) and 'gameOver' (true/false).



When saved the values of the three variables will be stored in an external text file and can be restored later when the Mediascape is resumed; even if the device it is running on has been turned off in the meantime.

Saving And Loading A Group Manually

To save and load the state of variables within a group at a given event - for example a button click - you use the associated `Save()` and `Load()` actions:

```
myGroup.Save();
myGroup.Load();
```

You would use these if you wanted to make the choice to save and reload the group variables data optional. So you might add a save button to give the user the ability to save at any given point and then a load button to return to that point; or alternatively give the user the option to load the previously stored position when the Mediascape first opens.

Note that if no data has been saved - or the associated text file has been deleted - when the `Load()` action is called it will fail silently (i.e. no error message will be displayed and the Mediascape will continue to run normally). If you wish to let the user know that there is no saved data you could test the value of an appropriate variable. In the example group above we could test whether the name variable has been set with the following condition:

```
if (name.Value != "") {
  // do this
}
else {
  // let them know there's no saved date
}
```

Caution should be taken to ensure that the tested variable will be updated in a saved file. In this case we would need to ensure that the name could not be left blank. Alternatively a dedicated true/false variable could be added and set to true when the file is saved.

Autosave

If you want the group variables to be automatically saved when the Mediascape exits and then automatically reloaded when it is restarted you should set the Autosave property to `true`. This saves the need to use the `Save()` and `Load()` actions; though does mean the user has no choice in the matter.

If you would rather make the choice to load on start-up optional you could instead leave Autosave set to `false` and place the `Save()` action on the Mediascape OnUnloading event.

Files and Filenames

By default saved data is stored in a file called 'Store01.xml' which is saved in the Mediascapes ! *Content folder, within a _store subfolder. XML is a common format designed to be human-readable and can be opened in a text editor such as notepad. Each time the _Save() action is invoked the contents of this file are updated. Needless to say, if the file is deleted from the device all associated save data will be lost.*

You can change the default filename directly within the group's properties; however it is also possible to set it using code:

GROUPS

```
myGroup.Filename = "myNewFilename";
```

Note that the filename is treated as *text*, so must be placed within "inverted commas". The '.xml' extension is added automatically by Mscape so does not need to be included.

Changing the filename has two important effects:

1. When the save action is called after changing the filename, if no file of that name exists a new save file will be created. If a file of that name does exist its contents will be replaced.
2. When the load action is called after changing the filename it will load data from a file with a corresponding name. If no such file exists no data will be loaded.

So why would you want to do this? Since setting a new unique filename creates a new save file you could use this approach to save a snapshot of the state of the group variables at any given point whilst the Mediascape is running, simply by adding a date time stamp to the filename; though if this is the only reason for using a group you may find the [Logger](#) a more appropriate tool.

Alternatively you could offer the user several save 'slots' with fixed names and then give the option to choose which one to save to and load from. So from your 'save' button you would have a menu of save-slots (for example: save01, save02, save03 etc.) and set the filename appropriately for each one before calling the *Save()* action. For example:

```
myGroup.Filename = "save01";  
myGroup.Save();
```

In this case to reload the variable data from this group we would first have to set the filename to match before calling *Load()*:

```
myGroup.Filename = "save01";  
myGroup.Load();
```

Important Points To Consider

If you intend to create a Mediascape that can be stopped and later restarted using data from group variables you need to think very carefully about what information is relied upon in your Mediascape. Whilst all the variables in your group will have been saved and restored, other values will be reset to their original state.

A perfect example is the EnteredCount of a region. When your Mediascape is restarted this will be reset to 0. If you rely on this value to determine what content is played OnEnter then the user may see content that is either repeated or not appropriate to the stage they've reached in the experience. In this case a separate variable should be added to your group and increased each time the user enters the region. You would then test against this variable - which will have been saved and reloaded with the group - to determine what content to play to the user. Needless to say saving the state of an Mscape could significantly increase the complexity of the project.

Known Limitations

Currently State Lists do not save correctly when placed in groups: only the last item in the list is saved.

StateList – Dynamic Lists of Items

StateList is an object that helps you work with lists of items, where the numbers of items are not known at the time you design your Mediascape. An item is defined as a discrete piece of data that comprises of a collection of variables such as Numbers, Strings, or TrueFalse objects.

The StateList is used in two main ways:

- In your Mediascape you can use an StateList to keep track of a dynamic number of author-defined items. For example, in a game you may have a variable number of enemies. You need to keep track of the X, Y and type of each enemy and need to be able to add and remove enemies as they are encountered or defeated. See below for an example.
- Using the WebServices system, you can use StateLists to help you download data from a web service to be used in your Mediascape. If you are expecting the data to contain a list of items, for example you're expecting to receive a list of all the pubs in a given area, you should add an StateList to the payload. Then, inside the [StateList](#) you should add variables that match the names and types of the pieces of data you require. For instance, for a pub you may add variables called 'Address', 'Name', and 'PriceOfPint'. See WebServices for more details.

Using StateList to Keep Track of Dynamic Objects

In this example, you are building a game with a variable number of enemies. For each enemy, you need to remember the X & Y positions on the map, and also the type of enemy.

- Create an StateList (right-click on State, and choose 'Add StateList')
- Rename it to 'Enemies'
- Right-click Enemies, and select 'Add Number'
- Rename the new number to 'X'
- Repeat this to create another number called 'Y'
- Right-click Enemies, and select 'Add Text'
- Rename the new text to 'Type'

You've now created the definition of what constitutes an enemy. But right now there'll be no enemies in the list. Let's add two.

- Click on Mediascape, and select the event OnLoaded

Type (you can omit the parts after // these are there to say what's happening)

```

Enemies.Add(); // adds a new enemy to the list
// The newly-added item will now be selected.
// Changing X, Y, Type will change the variables
// for the currently-selected Enemy.
X.Value = 10;
Y.Value = 10;
Type.Value = "BigBeastie";

```

```
Enemies.Add();  
X.Value = 20;  
Y.Value = 20;  
Type.Value = "LittleBeastie";
```

The above code will create two enemies. If you wanted to move the second enemy ('littlebeastie') to the right by 10 meters, you would type the following...

```
Enemies.Index = 1;  
// Select the second item - 0 would be the first.  
// X, Y, Type will now be the values for that enemy.  
X.Value += 10; // add 10
```

To iterate through all of the enemies and move them down 10 meters, you'd type:

```
for (int i = 0; i < Enemies.Length; i++)  
{  
    Enemies.Index = i;  
    Y.Value -= 10;  
}
```

Note that the above code will work for any number of enemies.

If the player had managed to defeat the first enemy, you'll probably want to remove it from the list.

```
Enemies.RemoveAt(0); // remember 0 is the first item, not 1
```

Note that when you remove an item, the index of the subsequent items will be updated, so the Enemy that was at index 1 will now be at index 0.

To add a new enemy at a later point, you simply repeat what we did to start off with...

```
Enemies.Add();  
X.Value = 30;  
Y.Value = 35;  
Type.Value = "MediumBeastie";
```

Articles

These cover all aspects of creating and working with Mscapes; from design concepts, adding media and interaction, to programming and troubleshooting.

Design

- [Design a Game](#)
- [Reduce the file size of your Mediascape](#)
- [Move an anchored Mediascape to a different location](#)
- [Make a portable game](#)
- [Modify an existing Mediascape](#)
- [Convert an anchored Mediascape to portable](#)
- [Using Mscape maker with an older *Mobile Bristol* type Mediascape](#)
- [Create a new Mediascape](#)
- [Mediascapes in the Field](#)
- [Designing a successful Mediascape \(pdf\)](#)

Media

- [Display some text on the screen](#)
- [Show several images in sequence to the user at a particular point](#)
- [Free Online Sources for Media for your Mediascape](#)
- [Import media into a Mediascape](#)
- [Import Audio into a Mediascape](#)

Maps and GPS

- [Creating a Mediascape using a blank map](#)
- [Recording your GPS route and showing it on screen](#)
- [Having trouble with GPS?](#)
- [Using Maps](#)
- [Setting up GPS on Mscape player](#)
- [Simulate GPS on Mscape player](#)

Interaction

- [Use the user's history when triggering media](#)
- [Different methods of user interaction with button presses](#)

Programming

- [Introduction to Programming](#)
- [an 'introduction' - Conditions](#)
- [Introducing the Mscape scripting language](#)
- [Using Random Numbers](#)
- [Using Functions](#)
- [Use the Math Library - Sin, Cos, Square Root etc](#)
- [Using the Logger in code](#)
- [Reserved words](#)

Flash

- [Create a located quiz game](#)
- [Allow the user to enter text](#)
- [Recommended video settings for Flash CS3](#)
- [Using looped Audio in Flash](#)
- [Using LoadMovie \(Accessing files external to Flash\)](#)

Testing and Troubleshooting

- [Simulate GPS on Mscape player](#)
- [Debug Your Mediascape](#) (perhaps split to debug in tester/debug in device)
- [Resolving Installation Problems](#)
- [Installing the player using Mscape Library](#)
- [Installing the player without Mscape library \(e.g. from a non-Windows machine\)](#)
- [Install additional languages](#)

Design a Game

General Guidelines for making a Mediascape game

Introduction

Mediascape games can be conceptually simple like “tag” or “hide and seek” or they can be complex quests such as the kind illustrated in the video Roku’s reward. Whatever the game, the process of Mediascape game design is to translate the game idea into a form which the current medium supports and one that will work within the current constraints of the toolkit.

All games have a hierarchy of rule systems: there are the rules of the game itself, the local rules and the wider rules of the culture and society in which the game will be played. If we take the game of soccer as an example

- The rules of the game are specified in the “rule book” and taught to each player. Game props such as goal posts, corner flags and chalk lines are used to help enforce the rules.
- The local rules take into account the local circumstances such as the size of the area you have to play in whether there enough players for a full team of 11, how long should the game last etc. These local rules are usually enforced by a referee.
- The wider rules take into account local laws and social rules. For example in a non professional game are ball games allowed in this area? In the professional game of soccer there is a governing body FIFA who enforce the integrity of the game across the world.

The point of highlighting this rule hierarchy is to show that much of game play depends on human and social mechanics not just the in-built game mechanics. And so whilst the current Mscape platform does not currently support capabilities for devices to talk to one another or for devices to talk to a game server, it is still possible to design compelling and social games that rely on human and social protocols rather than system protocols.

To take a very simple example you could easily turn the stamp the mole Mediascape game into a children’s party game just by telling the children where they need to place the mole holes and see who can do it quickest. You could also add obstacles or make them further apart to make it more challenging.

To design and create a Mediascape game follow these seven steps.

- 1. Identify the core game mechanic that you want to use and implement and test that before you design the detail of the game.**

Whenever you design a game there will be at least one main game mechanism that underpins it. For example hitting a ball with a racquet, moving a piece on a board or shooting at targets with a weapon. Mediascape games will typically involve movement around an outdoor space and maybe some interaction with the screen.

Mechanics based on GPS and movement.

You should think of your GPS as being like a balloon attached to a long piece of elastic which you hold. As you walk the balloon will more or less track where you are, though it will very rarely be directly overhead and it will also be prone to gusts of wind blowing it about. If you want a game to be based on your physical movement you should let it account for the fact that the GPS will not move as quickly as you can – just like the balloon it will jump and then follow depending if you turn or go in a straight line. If you are basing the game on movement and relying on GPS there are many things to bear in mind.

- GPS needs open spaces to work reliably – and built up areas will interfere with the readings
- Even in an open space GPS can drift. It is dangerous to rely on region sizes of less than ten meters if you need to associate regions with specific landmarks.
- GPS catches up quite slowly. If you run fast it can take a while for GPS to catch up with you.

So if you want to use the simple mechanic of moving between regions to trigger events build and test a prototype first. (Stamp the mole and Doubloons are good Mediascapes to try out as they use this mechanic).

Mechanics based on screen interaction.

There are several different ways to create screen interaction. [How to let the user press buttons](#)

You can create simple buttons by adding hotspots to images. They are reliable and effective. However if you want to display dynamic information like your score, number of things collected etc then you will need to use HTML. Be warned that if you use HTML buttons the action the user needs to make is a quick tap rather than a press. This is because HTML is designed to ignore presses of the screen when the point at which you press on the screen is more than a few pixels away from the point at which you release the press. (Danger UXB uses HTML for screen interaction in the mini games – you should try it out to get an idea of this issue).

Flash interaction.

If you are going to use flash as the front end interface to your game you should try out some simple tests to decide how much of the game logic you want to implement in flash and how much should be done in the maker. [How to use Flash](#)

2. Sketch out the game as a storyboard

Once you are happy with your core mechanic and you have tested that it works you should sketch out the rest of the game design. On paper sketch the series of stages that the game will go through and thus the number of different images that you need to create. You should also think about the overall structure – if the game is complex make sure that the design can break down into discrete chunks or game segments. Map out the overall design interaction – how people will move from one game segment to the next.

Rich narrative games. If it is a plot and narrative based game you should write the key elements down. You should also try to break up the narrative into plots and sub plots. These may be levels of the game or different missions. For example in the Tower of London game there were a series of missions and each mission could be designed separately. You should then write the scripts for the main dialogue. It is helpful to write the scripts in a similar style to that of a stage play. Try to keep

each piece of dialogue to 30 seconds or less in duration. If you have to stay in the same place for too long people get self conscious. For each piece of dialogue identify the character who is speaking and how their piece of dialogue gets triggered. It may get triggered automatically after the end of the previous piece of dialogue or it may get triggered by walking into a new or pressing a button. Specify any conditional logic that might be associated with the piece. For an example of a script that we used for the design of the Tower of London game please see this file. * Polar_bear_script_v3.pdf: Polar Bear Script

3. Record temporary audio and image files

It is a good idea to record all of your scripts in “scratch” form so that you can see how well they play out in the Mediascape. You should also create rough images for any of the dialog screens. You should test out this scratch content in both the tester and by walking outside. If you are creating an anchored game then ideally you would want to test it in the real setting. However if you are creating the game a long way away from its real setting you should still test the game in an outside area. To move regions from one area to another follow this link [How to Move an Anchored Mediascape](#)

4. Develop, test and iterate.

If it is a large game you should develop it in stages and test it as you go. Continue to implement the Mediascape in the maker, test it in the tester and copy it to your PDA to test it outside.

5. Create final audio, image and video assets.

Once you are happy with the overall flow of the dialog you should create your polished audio, images and video. This can simple mean re-recording your scripts with your friends and editing them to cut out any background noise, mistakes, umms and ahhs. In Mediascapes such as Hidden Danger UXB this involved bringing actors into a recording studio to record the dialog and introduction. A graphics artist created the images.

6. Add final content to the Mediascape

The process for swapping out the scratch content for the final content is very easy. Simply right click on the Mediascape object and replace the file, this will delete the old file from your current directory and replace it with your new file. It is best if the old and new files have slightly different names for example your old file might be called introductionScratch.mp3 and your new file would be called introduction.mp3.

7. Upload the finished game to the website

Once all the final content is in place you should be finished. Upload your new game to the website.

How to Reduce the Size of Your Mediascape

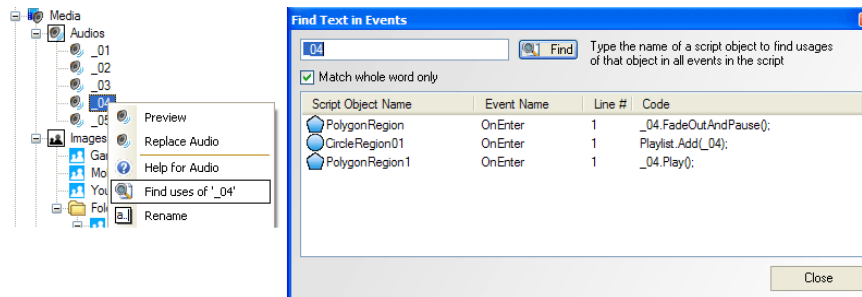
How to: Reduce the Size of Your Mediascape

If you've created a few Mediascapes using Mscape maker, and played around with adding lots of different kinds of media, you may find that your output file (created using the Save to Web function) is quite large. All of those pieces of content can quickly add up to make a fairly hefty download time for anyone who may want to download your Mediascape from www.mscape.com. Large Mediascapes will also take up more space on the mobile device, reducing the number of Mediascapes your user can carry with them at once. This tutorial will discuss various ways of making your Mediascape smaller.

Removing Unused Content Files

It's very common to find that you have quite a few unused pieces of media left over in your finished Mediascape, for example scratch files (non-final content created quickly to test parts of your 'scape) and content that didn't make the final cut. When your Mediascape is compressed for uploading to the website, all content files listed in Mscape maker - including those that are not actually used - are added to the bundle. Note that files left over in the content directory that are not listed in the Mediascape will NOT be added to the bundle (except for those in the WebPages folder - see below). This means it's worth deleting any content files from within Mscape maker that are not used - simply right-click them in the list on the left and select delete. Make sure you have backups of all the files as deleting content files cannot be undone.

If you are not sure if a content file has been used, you should make use of Mscape maker's 'Find uses of..' function. Right-click on a media item, and select 'Find uses of..'. This will search all of the event script code, and list every time you have used the content file. If no results are returned, then you may be able to delete the content file.



Warning: In version 2.0 of Mscape maker 'Find uses of' will not list uses of content files in Slideshows or Speakers. Double-check that your content files are not used in these before deleting them!

Mediascapes that use WebPages have a special folder in the content directory called 'WebPages'. All files inside this folder will be copied to your output file - so it's worth looking inside here and removing any files or folders that are not used. To view this folder, go to the Tools menu, select 'Open Content Folder', and open the '!WebPages' folder.

Reducing the Size of Your Content Files

Once you have removed unused content files the next job is to reduce the size of the content you are using.

Image Files

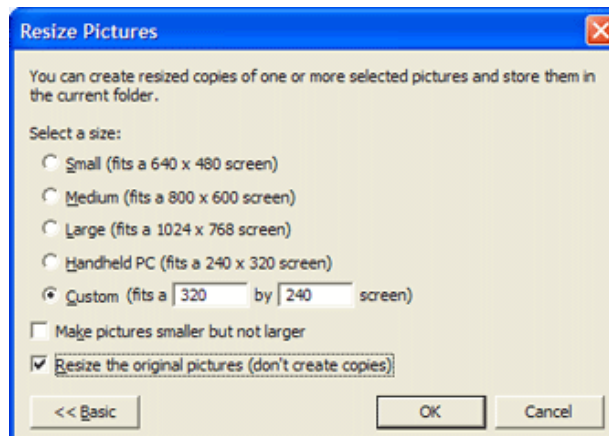
The simplest way of reducing the size of your image files is to ensure that they fit the screen size and orientation of the mobile device. The image system in Mscape is able to shrink large images to fit the screen, but it is better to make sure they are the right size in the first place. This'll make for a smaller download, and a faster and more responsive Mediascape.

A useful tool for resizing images is Microsoft's Free Image Resize Powertoy. You can download this from: [here](#)

Once you've installed this you can resize your images by right-clicking on the image(s) in Windows Explorer and selecting 'Resize Images'. In Mscape maker, go to Tools / Open Content Folder to open the content folder.



Click the 'Advanced >>' button and set the options as shown below. It's important to set the check the 'don't create copies' checkbox so that the resized images have the same name - this means the Mediascape will immediately use the new version. Ensure that you've got backups of your images before you reduce them in size this way!



Choose the appropriate size for your mobile device screen. Standard QVGA screens (such as that on the HP rx5935 Travel Companion) are [240x320 in portrait mode, and 320x240 in landscape mode](#).

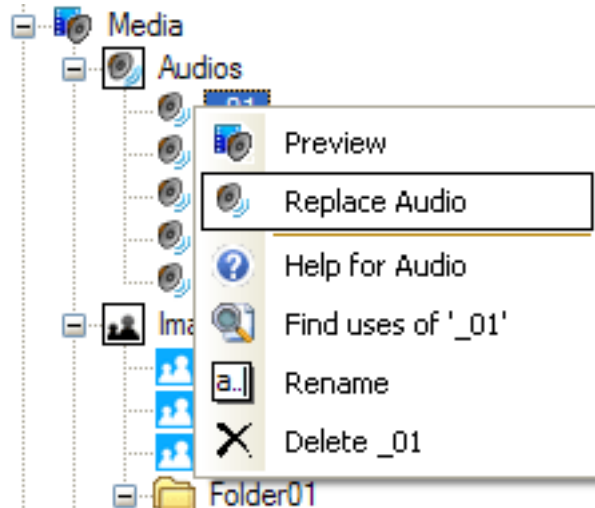
Audio Files

In many Mediascapes Audio files often form the bulk of the overall file size. This means its particularly important that these files are shrunk down as much as possible.

You can use a free audio editing program such as Audacity (<http://audacity.sourceforge.net/>) to edit and trim your audio files. It's worth removing silences from the beginning and end of your audio files, and shortening very long sounds if the full length is never needed.

If you are using an uncompressed format such as WAV for your audio, you should convert them to a compressed format - either mp3 or ogg.

If you have converted an audio into a new format the file extension may have changed (e.g. audio3.wav would become audio3.mp3). In order for the Mediascape to find the newly named version you will need to replace the audio files in Mscape maker with the new versions. To do this, right click on the content file in Mscape maker and choose 'Replace..'. This will pop up a Open File dialog allowing you to select the newly converted file.

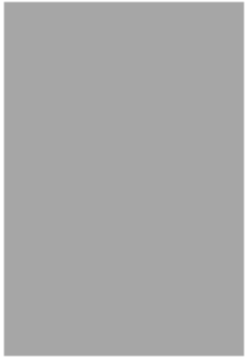


Video Files

The same rules for reducing file size for audio also apply to video. Ensure that your video matches the screen size and orientation of your mobile device, and trim the beginning and end of the movie. You can do this using your favorite video editor.

See the document [CreatingVideoFiles](#) for information on the recommended video format and compression settings.

Portrait or Landscape Format



Portrait



Landscape

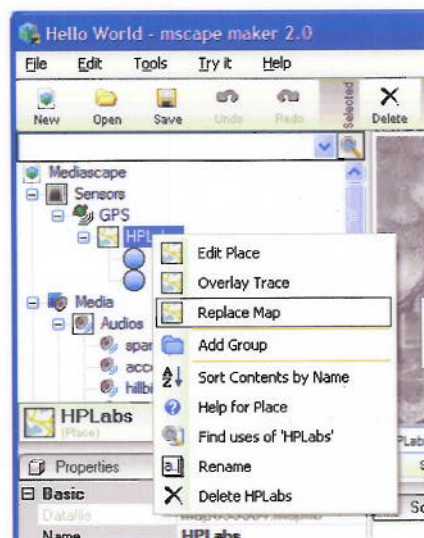
Portrait format is upright, like a conventional portrait painting.

Landscape format is 'on its side', like a landscape painting.

How to Move an Anchored Mediascape to a Different Location

It is simple to move an anchored Mediascape to a different location. You will need to have or create a maplib of your new location. Follow these steps.

- Open the anchored Mediascape that you want to move in the maker.
- Right click on the place object that you want to change and choose Replace Map.



Map Service will not be available after March 31, 2010

If you have already created a maplib choose From File and select the maplib for your new destination.

If you have not got a maplib of your location and you are in the United States choose From Map Service.

If you are outside the US and need to create a new maplib follow this link to find out how. [Create your own map](#)

In the dialog box that asks "Do you want to move the located objects so they fit onto the new map" select Move them.

The regions should then appear in your new map and if you need to you can move them. Now save this Mediascape with a different name, copy it to your iPAQ and go out and play it.

There is another description of this task on the page [Making an anchored Mediascape into a portable Mediascape](#)

How to Make a Portable Game

Portable games are Mediascapes that can be played anywhere. The usual mechanism is to get the user to place the regions needed in the game. This tutorial will take you through a step by step account of how the game "Stamp the Mole" was created. It is a firsthand account so that you can get an idea of the thought process used and the steps that are taken.

Making the Stamp the Mole Game

Coming up with the idea.

I wanted to make a simple game to act as a tutorial. I also wanted to make it portable so that it can be played anywhere. I thought of different ideas that used the simple game mechanic of moving between different regions. The first idea I had was to do a physical version of "low to high" where you had to remember the positions of the lowest to the highest numbers flashed on the screen by walking to their positions. However this seemed a bit too complex for a simple first game. I then thought of the fairground mole bashing game where moles pop up from holes and you hit them with a rubber mallet.

I thought it might translate well to a physical game as you can walk to a region and stamp on a virtual mole. So that is what I am going to do.

Designing the game.

I thought I would just have 3 regions as it is a simple game, I don't want it to take too long to set up and you want to be able to move quite quickly. I thought of having cute mole graphics – but I am not a graphics person and it would take too long so I will stick with something very basic using the map displayer and the ability to move a pin around on it. So I will have a very simple pin that says "THE MOLE IS HERE" which will move from region to region and you have to get to that region and stamp. I have decided the goal of the game is to stamp on 10 moles to win otherwise you lose and the moles will dig up your garden. I will have a timer so that if you get to the mole in time it is a success, if not you fail. Each time you succeed the timer will decrease by 2 seconds. My initial thoughts are to try it with a time of 30 seconds to get to the mole decreasing by 2 seconds so that by the end you only have 10 seconds to get to the mole. I will have to test this out and maybe change the timings as a result of testing. So now I think I have my overall game design worked out I will start coding!
In the maker.

I have opened up the Mscape maker. I know this is going to be a portable game so I don't need a map with real world co-ordinates. But I do need a map to put regions on. So I do the following
Make a blank image map.

Select Tools – Create Map From Image In the dialog box that say Choose the source of your co-ordinates. I choose Don't bother with co-ordinates I then import an image which is just a completely white jpeg (320 * 320 pixels big). This is because I want the pin to show up on a white screen so I don't want a fussy background picture.

I then save this as a maplib called gamearea.maplib. (I happen to have a directory called maplibs that I put all of my Mscaple maplibs.).

Import the map into the maker.

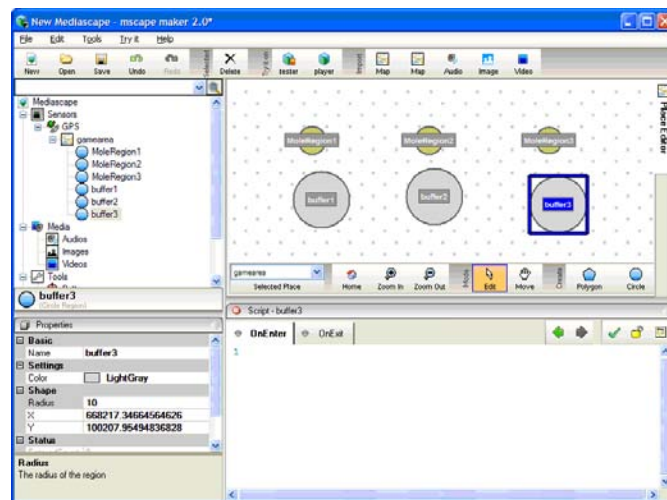
Now I want to use the maplib I just made so I do the following Select the Map icon (top toolbar of the maker) – On the dialog box that says How do you want to import the map? I choose From file. I then navigate to where I saved the maplib and choose gamearea.maplib.

Add the three mole regions.

Now I want to add the three regions that will be where the moles pop up. I click on the Circle icon and then change the properties of the circle to be Name : MoleRegion1 Radius : 5 – the radius is in meters so this will give me 10 meter diameter circles. I add two more circles and call them MoleRegion2 and MoleRegion3 and set their radius to 5 as well. I choose circles because they are the easiest to set their co-ordinates to move them to my current GPS location, they have a center point X,Y position and a radius.

Add three buffer regions.

Now I need 3 buffer regions to help me when I ask the user to place the mole regions. I don't want the user to place the regions too close together or on top of each other so I am going to add some logic which tests whether the user is in a buffer region and if they are clear of a buffer region then I will let them add a mole region. I add three circles and change their properties so that they are a light grey color with name buffer1, buffer2, buffer2 each with a radius of 10. My screen now looks like this (the map is zoomed in so that you can see the regions)



Make the fixed images

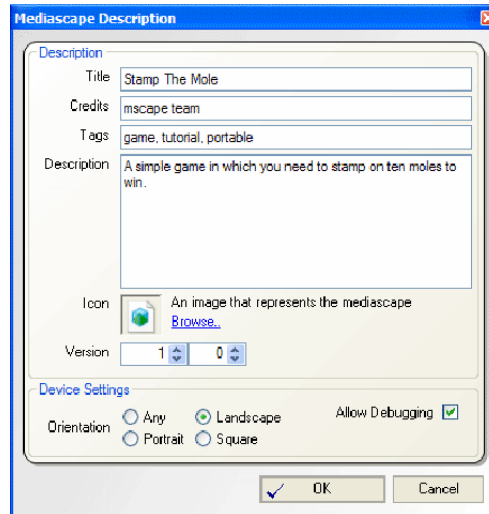
Now I need to make a number of screens for the user interface - to place the mole holes, prompt the user, introduction, win and lose etc. I make these screens as 320*240 pixel jpeg images – as I have decided to make this game [landscape](#). (If it was a portrait game I would have made them 240*320 pixels big).

I was tempted to get carried away with some nice graphics – but I resisted doing that now as I know it is important to test the game. When it works it is really easy to replace the images with more exciting ones.

Now I import these images into the Mediascape. I click on the Image icon. Navigate to the stamp the mole folder and select all of the images. They now appear in the object list in my Mediascape.

Save Mediascape

I have done a fair amount so it is now time to save my Mediascape. I click on File - Save as. A Mediascape Description dialog box pops up.



I set the orientation to Landscape and make sure that Allow Debugging is checked. This is so that when I go out to test it if anything is wrong I can get to a debug screen and look at the values of variables. I save the Mediascape in the stamp the mole folder. Now at this stage the urge to have a cute picture of a mole is so strong that I go and ask my friend Ben to draw one for me – one for the Mediascape icon image and one with a mole about to be stamped on and a win and lose image.

Once he has drawn it I will scan it in (or take a photo of it) and make the image size 320 * 240 so that it does not exceed the memory size of my IPAQ.

Add “welcome – tap screen to continue” when the Mediascape first loads.

When the Mediascape starts up I want the welcome screen to show and for the user to tap the screen to continue. I click on the Mediascape object. In the script OnLoaded event I add the following code `welcome.Show();` I do this by dragging the welcome object over to the script window. This saved me typing in the name. (I had to be careful not to actually select the welcome object as this would change the script window – it’s a select and drag without clicking on the object). Then after the name I put the decimal point and this brought up a list of auto complete actions and I chose Show. Now as I want the whole screen to be live so that the user can tap anywhere I add a hotspot and leave the size at their defaults as this will cover the whole screen. I right click on the welcome image object and select add hotspot.

Add GPS code for detecting a fix.

In the script window associated with the hotspot I write the following code

```
if (GPS.HasFix == true)
    placemolehole.Show();
else
    img_GPSfix_personal.Show();
```

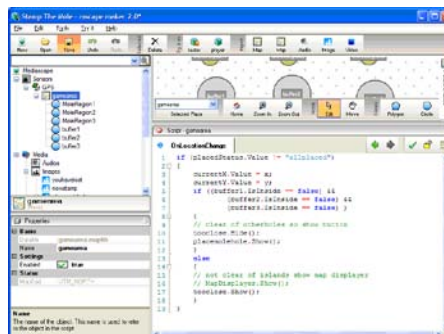
As the game will not work unless the GPS is working well then whenever the GPS does not have a fix I want to put up the "Waiting for GPS" screen. I select the GPS object. There are a number of tabs. In the OnGotFix tab I add `img_GPSfix_personal.Hide();` In the OnLostFix tab I add `img_GPSfix_personal.Show();`

Add the code to place the regions

Now I need to add the code which prompts the user to place the regions. I have done this before in the Doubloons game so I open that up in another window so that I can copy the code.

First I add a variable to track how many holes I have placed. I scroll down the object list to the State object. I right click on State and add number. I change the name to HolesPlaced and leave the value as 0. I also add two more number state variables (Right click on State – Add number) I make their names `currentX` and `currentY`. I then select the gamearea object and enter the following code in the script window

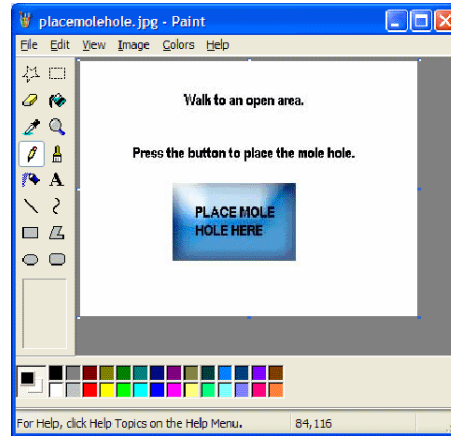
```
if (placedStatus.Value != "allplaced")
{
    currentX.Value = x;
    currentY.Value = y;
    if ((buffer1.IsInside == false) &&
        (buffer2.IsInside == false) &&
        (buffer3.IsInside == false) )
    {
        // clear of otherholes so show button
        tooclose.Hide();
        placemolehole.Show();
    }
    else
    {
        // not clear of islands show map displayer
        // MapDisplayer.Show();
        tooclose.Show();
    }
}
}
```



Every time the GPS location changes this event will get called. If you have not yet placed all the holes it will copy the GPS reading x and y into our variables x and y. It also tests if you are inside a buffer region or not and if not it puts up the placemolehole dialog box.

Add the Place region code.

The placemolehole image has a button on it. We need to know the pixel co-ordinates of the top left point of the button and the image width to define a hot spot on the iPAQ screen. To do this I open up the image in the Paint program. As you move the cursor around the image the pixel co-ordinates show up in the bottom of the window. Make a note of the co-ordinates of the top left edge, the top right edge and the bottom left edge.



For this image they are Top Left (84,116) Bottom Left (84,192) Top Right (208,116). This makes the button area 124 pixels wide (208 minus 84) and 76 pixels high (192 minus 116).

Right click on the placemolehole image object and choose Add Hotspot. Select the Hotspot and fill in the parameters. Height 124, Width 76, X 84 and Y 116 Add the following code

```
switch (placedStatus.Value)
{
case "none" :
    // set the first mole hole and its buffer
    MoleRegion1.X = currentX.Value;
    MoleRegion1.Y = currentY.Value;
    buffer1.X = currentX.Value;
    buffer1.Y = currentY.Value;
    placedStatus.Value = "one";
    break;
case "one" :
    // set the first mole hole and its buffer
    MoleRegion2.X = currentX.Value;
    MoleRegion2.Y = currentY.Value;
    buffer2.X = currentX.Value;
    buffer2.Y = currentY.Value;
    placedStatus.Value = "two";
    break;
case "two" :
```

```
// set the first mole hole and its buffer
MoleRegion3.X = currentX.Value;
MoleRegion3.Y = currentY.Value;
buffer3.X = currentX.Value;
buffer3.Y = currentY.Value;
placedStatus.Value = "three";
break;
}
```

This code basically tests to see what stage of hole placing you are at and then setting the regions and buffer x and y to your current position. We then move the stage to the next hole. After I added the code I did a check for syntax errors by clicking on the green tick. I then tested the Mediascape in the tester. I clicked around on the screen to simulate GPS position and I pressed the button. The regions move to where my position is and it appears to work as I intended. Now need to add in more logic to start the game after all regions have been placed.

In the mean time Ben has finished the mole pictures. They are great! So I have imported them as well.

Add the Map Displayer and Pin.

The map displayer automatically displays your current position on the screen. Pin objects have a property to display on the map displayer. I will use this to programmatically display and hide a pin with a label "THE MOLE IS HERE" which will be set to the center point of the hole regions.

I right click on the Tools object and choose add Map Displayer. I set the Place Name property of the mapdisplayer to be gamearea.

I right click on the map in the maker and choose Add Pin. I change the name to THE MOLE IS HERE.

Add timers.

I add two timers. One for showing a Watch for the moles screen and the other for time to find the mole. I right click on Tools and choose add Timer. Then I right click on Timer and add Alarm. I set the name to watchForMoles and the period to 10. I right click on Timer again and add Alarm. I set the name to molesOut and the period to 30. The periods are in seconds so this should allow 10 seconds to show the prompt and then 30 seconds to get to the mole.

Now I go back to the OnTapped code of the Hotspot02 in the placemolehole object. After the placedStatus.Value = "allplaced"; line I add the following two lines of code Mole.Show(); watchForMoles.Start();

This will show the mole picture and start the 10 second timer.

Random function to set the mole pin

I now need to work out how best to implement my design for randomly choosing the hole that the mole should pop up from. Basically I need to know which hole I am at currently, and then randomly choose one of the others. I know how to generate a random number by declaring a local Random variable r which if you call next with a number generates a random number in that range so the following code Random r = new Random(); r.Next(10); generates a number between 0 and 9. I add the following code to the OnRing function of the WatchForMoles Timer.

```

// add logic to decide which region to set the mole pin
MapDisplayer.Show();
Random r = new Random();
//r.Next(2) generates a random number between zero and 1
switch (currentHole.Value){
    case "MoleRegion1" :
        if (r.Next(2) == 0) {
            currentHole.Value = "MoleRegion2";
            THE_MOLE_IS_HERE.X = MoleRegion2.X;
            THE_MOLE_IS_HERE.Y = MoleRegion2.Y;
        }
        else {
            currentHole.Value = "MoleRegion3";
            THE_MOLE_IS_HERE.X = MoleRegion3.X;
            THE_MOLE_IS_HERE.Y = MoleRegion3.Y;
        }
        break;
    case "MoleRegion2" :
        if (r.Next(2) == 0) {
            currentHole.Value = "MoleRegion1";
            THE_MOLE_IS_HERE.X = MoleRegion1.X;
            THE_MOLE_IS_HERE.Y = MoleRegion1.Y;
        }
        else {
            currentHole.Value = "MoleRegion3";
            THE_MOLE_IS_HERE.X = MoleRegion3.X;
            THE_MOLE_IS_HERE.Y = MoleRegion3.Y;
        }
        break;
    case "MoleRegion3" :
        if (r.Next(2) == 0) {
            currentHole.Value = "MoleRegion1";
            THE_MOLE_IS_HERE.X = MoleRegion1.X;
            THE_MOLE_IS_HERE.Y = MoleRegion1.Y;
        }
        else {
            currentHole.Value = "MoleRegion2";
            THE_MOLE_IS_HERE.X = MoleRegion2.X;
            THE_MOLE_IS_HERE.Y = MoleRegion2.Y;
        }
        break;
}

```

```

THE_MOLE_IS_HERE.ShowOnMapDisplayer = true;
MapDisplayer.Show();
molesOut.Start();

```

That sets the pin to a random hole and if the map displayer is shown will display that pin because the ShowOnMapDisplayer property is true. It then displays the MapDisplayer and starts the timer to get to the mole.

Now it is time to save and test on the tester. Seems to work – I think I will need to add some sound effects and audio later. Now we shall add code to the molesOut timer to implement what happens if you don't get to the mole on time. I add the code `YouLose.Show();` to the `OnRing` Event. This will be the end of the game – I will probably need to add an audio that says – oh dear you did not get to the mole on time so you have lost and your garden will not be overrun with moles. I shall test the game with it like this. The alternative implementation I could try is to let you always have 10 tries and if you don't get all ten then you lose. With this implementation the first time you fail the game ends.

Add logic for stamping the mole.

If you do get to the mole on time we want to show the `STAMP` screen and then decrement the timer and go back to watch for moles. In the `OnEnter` event of `MoleRegion1` I add the following code

```
if (currentHole.Value == "MoleRegion1" && placedStatus.Value == "allplaced")
{
    MoleFoot.Show();
    molesOut.Stop();
    molesOut.Period = molesOut.Period - 2.0;
    watchForMoles.Start();
}
```

I copy the code and paste it into `MoleRegion2` and `MoleRegion3` and change the test to check the `currentHole.Value` to the appropriate region. Another quick test on the tester shows this is basically working.

Now I need to add a variable to keep a count of the number of stamps so that you end the game with ten stamps.

I right click on `State` add a `Number` and change its name to `molesStamped` and leave the value as 0.

Now I modify the code in `MoleRegion1` to say

```
if (currentHole.Value == "MoleRegion1" && placedStatus.Value == "allplaced")
{
    MoleFoot.Show();
    molesOut.Stop();
    molesStamped.Value = molesStamped.Value + 1;
    if (molesStamped.Value == 10) {
        // You have won
        YouWin.Show();
    }
    else {
        molesOut.Period = molesOut.Period - 2.0;
        watchForMoles.Start();
    }
}
```

And edit `MoleRegion2` and `MoleRegion3` as well.

Now I try it on the tester. Hurrah it works. Time for my first outside test. So I will copy the whole directory over to my SD card, pop it into my iPAQ and go outside to test.

Not too bad as a first test – even though I lost! What I found was the region size is probably about right. The zoom level of the map displayer was not in enough and the MOLE IS HERE pin was a bit ambiguous as to which hole was active. I will modify the Mediascape so that there are three pins – one for each hole you placed – and I will programmatically make them visible or invisible rather than move the one pin.

Aesthetically I need to make some of the images have a landscape orientation so that they look nicer. I also need a hole has been successfully placed image to show before the prompt to walk and place another.

I shall also try to make some audio prompts and sound effects. I have a digital recorder so I can easily record my own audio files. Add 3 pins – change the logic. I add three pins to the Mediascape and name them Mole_at_Hole_1, Mole_at_Hole_2, Mole_at_Hole_3. I now need to change the logic of the watch for moles so that instead of moving one pin I make the right pin visible. I change the logic to this

```
// add logic to decide which region to set the mole pin
Random r = new Random();
//r.Next(2) generates a random number between zero and 1
switch (currentHole.Value){
    case "MoleRegion1" :
        Mole_at_Hole_1.ShowOnMapDisplayer = false;
        if (r.Next(2) == 0) {
            currentHole.Value = "MoleRegion2";
            Mole_at_Hole_2.ShowOnMapDisplayer = true;
        }
        else {
            currentHole.Value = "MoleRegion3";
            Mole_at_Hole_3.ShowOnMapDisplayer = true;
        }
        break;
    case "MoleRegion2" :
        Mole_at_Hole_2.ShowOnMapDisplayer = false;
        if (r.Next(2) == 0) {
            currentHole.Value = "MoleRegion1";
            Mole_at_Hole_2.ShowOnMapDisplayer = true;
        }
        else {
            currentHole.Value = "MoleRegion3";
            Mole_at_Hole_3.ShowOnMapDisplayer = true;
        }
        break;
    case "MoleRegion3" :
        Mole_at_Hole_3.ShowOnMapDisplayer = false;
        if (r.Next(2) == 0) {
            currentHole.Value = "MoleRegion2";
            Mole_at_Hole_2.ShowOnMapDisplayer = true;
        }
        else {
```



```

    currentHole.Value = "MoleRegion1";
    Mole_at_Hole_1.ShowOnMapDisplayer = true;
}
break;
}

```

```

MapDisplayer.Show();
molesOut.Start();

```

I no longer need the MOLE IS HERE pin so I want to delete it. But first I want to check it is not used in the Mediascape. I right click on the MOLE IS HERE pin and choose Find uses of "THE MOLE IS HERE". A dialog box pops up with an empty list so I know it is not used anywhere and it is safe to delete.

I delete it and check syntax (green tick) . All good so I test it on the tester. In the test I realize I have not yet added the code to move the pins to where you place the holes. I edit the code in Hotspot02 of the placethemole image. I add the two lines to set the X,Y of each pin in each of the region placements. The amended code now looks like this

```

Mole_at_Hole_1.ShowOnMapDisplayer = false;
Mole_at_Hole_2.ShowOnMapDisplayer = false;
Mole_at_Hole_3.ShowOnMapDisplayer = false;
// add logic to decide which region to set the mole pin
Random r = new Random();
//r.Next(2) generates a random number between zero and 1
switch (currentHole.Value){
    case "MoleRegion1" : {
        if (r.Next(2) == 0) {
            currentHole.Value = "MoleRegion2";
            Mole_at_Hole_2.ShowOnMapDisplayer = true;
        }
        else {
            currentHole.Value = "MoleRegion3";
            Mole_at_Hole_3.ShowOnMapDisplayer = true;
        }
    }
    break;
    case "MoleRegion2" :{
        if (r.Next(2) == 0) {
            currentHole.Value = "MoleRegion1";
            Mole_at_Hole_1.ShowOnMapDisplayer = true;
        }
        else {
            currentHole.Value = "MoleRegion3";
            Mole_at_Hole_3.ShowOnMapDisplayer = true;
        }
    }
    break;
    case "MoleRegion3" :{
        if (r.Next(2) == 0) {

```

```

    currentHole.Value = "MoleRegion2";
    Mole_at_Hole_2.ShowOnMapDisplayer = true;
}
else {
    currentHole.Value = "MoleRegion1";
    Mole_at_Hole_1.ShowOnMapDisplayer = true;
}
}
break;
}

```

MapDisplayer.Show();

molesOut.Start();

I also set the zoom level of the map displayer to 10.

Time for another test. Happy with that now to record some audio.

I record all my audio on my dat recorder, copy the files onto my PC and edit them with Audacity. I save them as mp3 files and now need to import them into my Mediascape.

Add audio to my Mediascape.

On the OnLoaded event of my Mediascape I add welcome1.Play(); to play my audio welcome when the Mediascape starts up.

I change the code in the Hotspot01 of the welcome screen so that instead of showing the placemolehole image I play the placehole audio.

I cut the placemolehole.Show(); code and replace it with placehole.play(). Then in the placehole audio OnStarted event I paste placemolehole.Show(); This means that when the audio starts playing the image will show. In the OnTapped event of Hotspot02 I add moleplacedok.Play(); to cases none and one. This means that when you tap the screen and place a hole you will hear a success message.

I also add moleplacedok.Play(); in the case two part of the code so that when you have placed them all you hear that you have placed them all.

I add placeanother.Play(); to the OnFinished event of the moleplacedok audio.

I add watchscreen.Play(); to the OnFinished event of the gothimaudio so that as soon as that audio finishes you are told to watch the screen.

I add molealert.Play(); just before I display the mapdisplayer in the OnRing event of watchForMoles timer.

I add youwin1.Play(); to the win condition code in the OnEnter event of each of the mole regions.

I add gothim.Play(); to the else condition code in the OnEnter event of each of the mole regions.

I add watchscreen.Play(); to the OnFinished event of the gothim audio.

I test things on the tester and when I am happy I try it again on the iPAQ.

It is mostly good – I get an odd behavior after I lose that I then still get the mole. I realize I have not put a test in the region code to see if the game has ended. So I add a true/false variable – right click on State. And call it endgame. I set it to true in the region code and in the OnRing event of molesOut (when you lose). The region code now looks like this

```

if (currentHole.Value == "MoleRegion3"
    && placedStatus.Value == "allplaced"
    && endgame.Value == false)
{
    MoleFoot.Show();
    molesOut.Stop();
    molesStamped.Value = molesStamped.Value + 1;
    if (molesStamped.Value == 10) {
        // You have won
        youwin1.Play();
        YouWin.Show();
        endgame.Value = true;
    }
    else {
        gothim.Play();
        molesOut.Period = molesOut.Period - 2.0;
        watchForMoles.Start();
    }
}

```

A couple more tests outside and then I am finished!

I then upload the Mediascape to the Mscapes website and attach these notes to the description.

How to Modify and Reuse Bits of a Mediascape

Looking at other people's Mediascapes is the best way to learn about how a Mediascape is put together. Luckily for you, our plucky Mscape user, Mediascapes use an open format where anyone can look at, learn from, and (license notwithstanding) reuse code and media in their own projects!

Reusing Media Files

When you download a Mediascape from mscapers.com you are actually downloading a whole bunch of files zipped up into a bundle with a .msz (Mediascape Zipped) extension.

When you download a msz file, Mscape library will helpfully extract this file into its constituent parts and save it into your library folder (by default called 'My Mediascapes' in your 'My Documents' folder).

You can therefore poke about with the innards of a 'scape really easily by using Windows Explorer to navigate around this folder to see what pieces of media are used. For example, a Mediascape called 'earthlings!' will consist of a msl (Mediascape Scripting Language) file that contains all of the code, and a folder called something like earthlings_Content that contains all the audio, images, videos, web pages, maps, and flash movies used.

Provided that the license on the Mediascape you downloaded allows it, you can use as a learning tool to inspire or inform your own media creation activities, or even directly use these media atoms in your own work.

If you have Mscape maker open already, you can add media from other Mediascapes in your library by simply clicking one of the import buttons, navigating to a content folder within the My Mediascapes folder and picking and choosing whatever media takes your fancy.

Reusing Bits of Code

When you play the latest, greatest Mediascape creation you may find yourself wondering exactly how they managed to achieve some particularly clever effects. Again, you are in luck as the open format that Mediascape use means you can look directly at the code so you can learn exactly how it was done.

For the masochistic or XML fetishist it is possible to simply open the .msl file directly in a text editor, but this isn't really recommended (but we won't stop you..). Instead try opening the Mediascape in Mscape maker (File / Open, navigate to the My Mediascapes folder in My Documents and select the Mediascape you are interested in.

Poke around a bit, click on various objects in the script view and read through the code in the events window. If the designer of the Mediascape was particularly helpful you may even find they have commented the code (shown in green in the events window) describing exactly what they have done and how it works.

Making an Anchored Mediascape into a Portable Mediascape

There are many Mediascapes available on the Mscape web site that are fixed to be used in a certain area. If you want to try them out you could go to that area, but if this is far away then it is possible to download the Mediascape and adapt it to work in your own area. You could try out a guided tour of Venice in the park behind your apartment for example.

To do this you need to do the following:

Prepare a map file of the locality where you want to try the Mediascape out. The park behind your apartment for example. Remember it has to be an image of a map with associated co-ordinates, it is not enough just to use an image of a map with no idea of where it is, (see the section on 'Building your own Mediascape; maps and media' for more about this).

Download the anchored Mediascape you want to shift, and open it up with the Mscape maker (see the sections Getting and using your first Mediascape and Mediascapes on the Mscape website for more on this).

In the left-hand panel of the window you will see the structure of the Mediascape with all the objects. Under the 'Sensors' group there is a group 'GPS' and under this there is the map object.

If you right click on this map object one of the menu options is 'Replace map'. Select this to start up a dialogue and choose 'From File' to open a new map file.

Select the map of your locality and then the dialogue will ask what you want to do with the position of the objects. You should choose 'move them' as you want to shift them to your locality.

The Mediascape on display will then change to show the new map with all the regions from the old Mediascape on it. They are all kept together but the placement is done automatically based on the boundaries of the two maps in question.

Unfortunately there is as yet no way to select all the regions and shift them together to adjust the Mediascape.

There is also a ['How To' for this task, with screen dumps.](#)

Using Mscape Maker with an Older Mobile Bristol Type Mediascape

Converting Mobile Bristol Mediascapes

As Mediascapes are a relatively new medium Hewlett Packard are continually striving to improve the tools and facilities for creating and playing them. The drawback with this is that as the Mscape maker and the Mediascapes that it produces become more sophisticated the older Mediascapes made with earlier versions of the Mscape maker no longer work correctly.

If you are a frequent user of computers for other things you may be familiar with this situation with such things as importing word processor documents into newer versions of the word processor.

Fortunately, it is possible to convert the older Mediascapes so that they become new Mediascapes. The new Mscape maker has been designed to do as much of the conversion for you, however it cannot do everything and so in some cases there are certain items that will need your input in order to be correctly converted.

The conversion process

When you import an older Mobile Bristol Mediascape into the new Mscape maker much of the conversion happens automatically. However, not all of it does and there are three areas where you may have to intervene to ensure that the conversion process has been successful.

To import a Mediascape you do the following

Firstly, the conversion process is as thorough and as fail safe as it can be, but, as with any computer based activity, it is a good idea to make a back up copy before you start doing anything, just in case.

From within the Mscape maker select File and choose the Convert MB menu option.

Choose the Mediascape you want to import and convert. A warning message will appear alerting you to the conversion.

If you go ahead with it then the conversion is carried out and the new version of the Mediascape is created and saved with the extension '.msl'.

When the conversion is finished a report box will appear indicating the success of the conversion.

The Mediascape is now converted and is ready for further editing.

You may have to make adjustments by hand in the following situations:

Rewriting of event scripts

The automatic conversion process does its best to convert the event scripts. This conversion is difficult and thus what takes place is a best guess at conversion. The original script is enclosed in

comments along with the best guess conversion and then the user must amend them. Basically in the comments it says; 'This is what you had originally, and this is roughly what I think it should be now'.

For example where an event script in a Mobile Bristol Mediascape may have been this:

```
<if cond="inRegion(regions$frog) and variables$frog=0">
<then>
<playMedia media="media$frog" volume="100" loop="false" position="0" fade="0" delay="0"/>
</then>
<else>
<DisplayHTMLImage source="frog.jpg" tag="ImageTag" background="88aaff"/>
</else>
</if>
```

After conversion it will appear as this:

```
/*
*****
The handler code in Mobile Bristol format looked like this:
*****
<if cond="inRegion(regions$frog) and variables$frog=0">
<then>
<playMedia media="media$frog" volume="100" loop="false" position="0" fade="0" delay="0"/>
</then>
<else>
<DisplayHTMLImage source="frog.jpg" tag="ImageTag" background="88aaff"/>
</else>
</if>
*****
```

For the current version, try this instead:

```
*****
if (frog_reg.IsInside && frog_var.Value == 0)
{
frog_audio.Volume = 100; // May not be needed. Check the values of the properties on this object.
frog_audio.Loop = false; // May not be needed. Check the values of the properties on this object.
frog_audio.FadeDuration = 0; // May not be needed. Check the values of the properties on this object.
frog_audio.Play();
}
else
{
frog_image.Show();
}
*/
```

Such scripts can either be sorted out 'by hand' and re-written using the new style of scripting with reference to the 'phrase book' below.

Alternatively, you can remove the entire script and generate the new scripts by doing drag and drop operations within the Mscape maker.

What happens during the conversion?

For completeness it may be useful to know some of the steps that take place during the conversion although it is not vital that you understand this in order to carry out conversions. The Mscape maker creates a MapLib file for the Mediascape and links it into a place object, adding any regions and speakers in the original Mediascape. A content folder is created and all the referenced content files are moved there. If required, a web root folder is created and all HTML pages are moved there.

The original .mbw, .mbp and .mbs files that made up the Mediascape are moved to an Archive sub-folder. Any deprecated objects (such as notifications and functions) are copied to a .del file in the Archive sub-directory.

MSL

The Mobile Bristol scripting language was based on something called XML, this meant that all parts of it were enclosed in 'pointy brackets', the new scripting language is called MSL, which stands for 'Mediascape Scripting Language. MSL structures Mediascapes into four sections:

Sensors, which are used to hold information about sensors used to trigger contextual events, such as GPS.

Media, used to hold the audio, images, videos etc, that are used in the Mediascape.

Tools, to hold useful functions, for example the Playlist and the Logger.

State, used to hold variables.

MSL is based on a language called C sharp (see below) so there are curly brackets, but a lot less of them than with Mobile Bristol scripts.

Here is a snippet of Mobile Bristol script:

```
<if cond="variables$count = 1">
  <then>
    <playMedia media="media$frogs" volume="100"/>
  </then>
  <else>
  </else>
</if>
```

and here is the same snippet converted to MSL script:

```
if (count.Value == 1)
{
  frogs.Volume = 100;
  frogs.Play();
}
```


C# has strong influences from languages like C++, Java and Delphi, so if you have some familiarity with any of those you may recognize elements in C# and thus the MSL used in Mediascapes. The syntax of MSL also has a few similarities with the syntax of Flash ActionScript and JavaScript so again some familiarity with these may come as an advantage when dealing with MSL.

Things to note with MSL

There are less terms and not every term in the script is enclosed in brackets, so it is a lot easier to write.

You don't have to state where things like media resources are coming from so you can say 'frogs' instead of having to say 'media\$frogs'.

Actions and attributes of objects like media resources are referred to in terms of the object that they relate to. So to play 'frogs' with a certain volume in Mobile Bristol scripting you used to say:

```
<playMedia media="media$frogs" volume="100"/>
```

In effect; 'I want to play something, this is what I want to play and this is how loud I want to play it'. With MSL you now say:

```
frogs.Volume = 100;  
frogs.Play();
```

Here you are saying; 'Set the frog's volume to this' then 'Play the frogs'.

MSL 'Phrase book'

If you are used to writing scripts in Mobile Bristol then this section shows the terms you will be used to and how they should be done within MSL. 'Manual page' refers to the Mobile Bristol Manual which you may already be familiar with if you are programming in the Mobile Bristol environment.

Using actions

| | |
|-------------|---|
| Name | playMedia |
| Manual page | 71 |
| Example | <pre><playMedia media="media\$harp" volume="100" loop="false" position="0" fade="0" delay="0"/></pre> |
| MSL rewrite | <pre>harp.Volume = 100; harp.Loop = false; harp.FadeDuration = 0; harp.Play();</pre> |
| Comments | <i>Delay</i> and <i>position</i> parameters are not supported |

| | |
|-------------|--|
| Name | stopMedia |
| Manual page | 72 |
| Example | <stopMedia media="media\$sharp" fade="0" delay="0"/> |
| MSL rewrite | harp.FadeDuration = 0; harp.Stop(); |
| Comments | <i>Delay</i> parameters is not supported |

| | |
|-------------|--|
| Name | setMediaVolume |
| Manual page | 72 |
| Example | <setMediaVolume media="media\$sharp" volume="50" fade="0" delay="0"/> |
| MSL rewrite | harp.Volume=50; |
| Comments | <i>fade</i> and <i>delay</i> parameters are not supported for volume changes |

| | |
|-------------|--|
| Name | stopAll |
| Manual page | 72 |
| Example | <stopAll fade="0" delay="0"/> |
| MSL rewrite | Audios.StopAll(); |
| Comments | Audios is the container for audio objects in MSL |

| | |
|-------------|--|
| Name | setVolumeAll |
| Manual page | 73 |
| Example | <setVolumeAll volume="75" fade="0" delay="0"/> |
| MSL rewrite | n/a |
| Comments | It is not possible to set an overall volume for all audios |

Source actions

Sources in Mobile Bristol map onto Speaker objects in MSL.

| | |
|-------------|---|
| Name | playSource |
| Manual page | 74 |
| Example | <playSource source="sources\$source1" volume="100" loop="false" position="0" fade="0" delay="0"/> |
| MSL | source1.AutoPlay = true; |

| | |
|----------|--|
| rewrite | |
| Comments | MSL speakers do not have <i>play</i> and <i>stop</i> methods but a similar effect can be achieved by toggling the boolean property <i>AutoPlay</i> . All other intended effects, such as setting a volume on a source, must now be applied to the speaker's associated audio object. |

| | |
|-------------|---|
| Name | stopSource |
| Manual page | 74 |
| Example | <code><stopSource source="sources\$source1" fade="0" delay="0"/></code> |
| MSL rewrite | <code>source1.AutoPlay = false;</code> |
| Comments | MSL speakers do not have <i>play</i> and <i>stop</i> methods though a similar effect can be achieved by toggling the boolean property <i>AutoPlay</i> . |

| | |
|-------------|---|
| Name | setSourceVolume |
| Manual page | |
| Example | <code><setSourceVolume media="sources\$source1" volume="100" fade="0" delay="0"/></code> |
| MSL rewrite | n/a |
| Comments | It is not possible to directly set the volume of a speaker, but it can be set for the associated audio object |

Image actions

Image display in Mobile Bristol was rather sparsely supported with a single action. This maps indirectly onto one of the rich new image capabilities in mscape.

| | |
|-------------|---|
| Name | DisplayHTMLImage |
| Manual page | 80 |
| Example | <code><DisplayHTMLImage source="coins.jpg" tag="ImageTag" background="88aaff"/></code> |
| MSL rewrite | <code>coinsImage.Show();</code> |
| Comments | It is not possible to display an image directly from file in mscape. Import the image file into an <i>Image</i> object and call its <i>Show</i> method. |

URL actions

URL actions in Mobile Bristol allowed local audio files to be played without needing a corresponding media object. This capability is not supported in mscape.

| | |
|-------------|---|
| Name | playUrl |
| Manual page | 75 |
| Example | <code><playUrl url="files\mySound.mp3" alias="mysound" volume="100" loop="false" position="0" fade="0" delay="0"/></code> |
| MSL rewrite | n/a |
| Comments | Not supported in mscape - import audio file into the mediascape as an audio object |

| | |
|-------------|--|
| Name | stopUrl |
| Manual page | 76 |
| Example | <code><stopUrl alias="mysound" fade="0" delay="0"/></code> |
| MSL rewrite | n/a |
| Comments | Not supported in mscape - import audio file into the mediascape as an audio object |

| | |
|-------------|--|
| Name | setUrlVolume |
| Manual page | 77 |
| Example | <code><setUrlVolume alias="mysound" volume="75" fade="0" delay="0"/></code> |
| MSL rewrite | n/a |
| Comments | Not supported in mscape - import audio file into the mediascape as an audio object |

Debug actions

It was possible to *trace* a text message to file in Mobile Bristol. In mscape this capability is provided by the *Logger* object.

| | |
|-------------|--|
| Name | trace |
| Manual page | 77 |
| Example | <code><trace text="Hello World"/></code> |
| MSL rewrite | <code>Logger.Log("Hello World");</code> |

| | |
|----------|--------------------------|
| Comments | Use <i>Logger</i> object |
|----------|--------------------------|

| | |
|-------------|---|
| Name | setTraceLevel |
| Manual page | 77 |
| Example | <code><setTraceLevel level="5"/></code> |
| MSL rewrite | n/a |
| Comments | Not supported in mscpae |

Logic actions

Mobile Bristol only supported one form of flow control - conditional expressions using the *if-then-else* pattern.

| | |
|-------------|--|
| Name | if |
| Manual page | 78 |
| Example | <pre><if cond="variables\$count = 1"> <then> <playMedia media="media\$frogs" volume="100"/> </then> <else> </else> </if></pre> |
| MSL rewrite | <pre>if (count.Value == 1) { frogs.Play(); }</pre> |
| Comments | Unlike in Mobile Bristol, empty else blocks are not required in mscape |

| | |
|-------------|---|
| Name | set |
| Manual page | 78 |
| Example | <code><set var="variables\$var1" value="123"/></code> |
| MSL rewrite | <code>var1.Value = 123;</code> |

| | |
|----------|--|
| Comments | In mscpae, values are assigned to variables using their <i>Value</i> property. |
|----------|--|

Flash actions

| | |
|-------------|---|
| Name | sendFlashMessage |
| Manual page | 79 |
| Example | sendFlashMessage('FUNC','updateScore',200) |
| MSL rewrite | flashMovie1.RunActionScriptWithParams("updateScore",200) |
| Comments | The use of Flash has changed considerably in mscscape. See dedicated help pages for detail. |

HTML actions

| | |
|-------------|---|
| Name | NavigateHTMLPage |
| Manual page | 80 |
| Example | <NavigateHTMLPage source=""files\newpage.html"/> |
| MSL rewrite | WebPages.LoadUrl('files\newpage.html'); |
| Comments | Relative URLs in mscscape are relative to the content folder, rather than the script folder as in Mobile Bristol. |

Event actions

In Mobile Bristol it was possible to simulate the effect of a change in the user's location. This is no longer possible.

| | |
|-------------|---|
| Name | raise |
| Manual page | |
| Example | <raise event="events\$mbLocationChangedEvent" x="11" y="22"/> |
| MSL rewrite | n/a |
| Comments | Not supported in mscscape |

Functions

Functions in Mobile Bristol are used to return a value that can then be included in an expression. For example, the function `inRegion` might be used as follows:

```
<if cond="inRegion(regions$myRegion)">
  <then>

    <trace("In region")/>
  </then>
<else>
</else>
</if>
```

In mscope, many, though not all, of these functions are now implemented as read-only properties on the appropriate object. So, the previous example might be written:

```
if (myRegion.IsInside)
{
  Logger.Log("In region");
}
```

Standard translations for the MB functions are listed below:

Region functions

| | |
|-------------|--|
| Name | enteredCount |
| Manual page | 83 |
| Example | <code>enteredCount(regions\$myRegion)</code> |
| MSL rewrite | <code>myRegion.EnteredCount</code> |
| Comments | |

| | |
|-------------|--|
| Name | inRegion |
| Manual page | 83 |
| Example | <code>inRegion(regions\$myRegion)</code> |
| MSL rewrite | <code>myRegion.IsInside</code> |
| Comments | |

Media functions

| | |
|-------------|--|
| Name | getMediaPosition |
| Manual page | 81 |
| Example | getMediaPosition(media\$story) > 2000) |
| MSL rewrite | story.Played |
| Comments | There are no functions to discover how much of an audio has been played in mscape. Very often, this function was used in Mobile Bristol to restart an audio from the position it was last played to. In mscape, this intention can be achieved directly through the <i>Pause</i> and <i>Play</i> methods on audio objects. |

| | |
|-------------|--|
| Name | getMediaMaxPosition |
| Manual page | 80 |
| Example | getMediaMaxPosition(media\$story) > 2000) |
| MSL rewrite | n/a |
| Comments | There are no functions to discover how much of a media has been played in mscape |

| | |
|-------------|---|
| Name | isMediaOn |
| Manual page | 81 |
| Example | isMediaOn(media\$story) |
| MSL rewrite | (story.Playing story.Paused) |
| Comments | The Mobile Bristol notion of an audio being <i>on</i> is represented by one of two conditions in mscape |

| | |
|-------------|-----------------------------|
| Name | isMediaPaused |
| Manual page | |
| Example | isMediaPaused(media\$story) |
| MSL rewrite | story.Paused |
| Comments | |

| | |
|-------------|------------------------------|
| Name | isMediaPlaying |
| Manual page | 81 |
| Example | isMediaPlaying(media\$story) |
| MSL rewrite | story.Playing |
| Comments | |

| | |
|-------------|-----------------------------|
| Name | isMediaFading |
| Manual page | 81 |
| Example | isMediaFading(media\$story) |
| MSL rewrite | story.Fading |
| Comments | |

URL functions

Because it is not possible to play audio files directly in mscape, none of the associated URL functions are supported.

| | |
|-------------|--------------------------|
| Name | getUrlMaxPosition |
| Manual page | 82 |
| Example | getUrlMaxPosition(alias) |
| MSL rewrite | n/a |
| Comments | Not supported in mscape |

| | |
|-------------|-------------------------|
| Name | getUrlPosition |
| Manual page | 82 |
| Example | getUrlPosition(alias) |
| MSL rewrite | n/a |
| Comments | Not supported in mscape |

| | |
|------|----------------|
| Name | isUrlOn |
|------|----------------|

| | |
|-------------|------------------------|
| Manual page | 82 |
| Example | isUrlOn(alias) |
| MSL rewrite | n/a |
| Comments | Not supported in mscap |

| | |
|-------------|------------------------|
| Name | isUrlPaused |
| Manual page | |
| Example | isUrlPaused(alias) |
| MSL rewrite | n/a |
| Comments | Not supported in mscap |

| | |
|-------------|------------------------|
| Name | isUrlPlaying(|
| Manual page | |
| Example | isUrlPlaying((alias) |
| MSL rewrite | n/a |
| Comments | Not supported in mscap |

| | |
|-------------|------------------------|
| Name | isUrlFading |
| Manual page | |
| Example | isUrlFading(alias) |
| MSL rewrite | n/a |
| Comments | Not supported in mscap |

Source functions

It is not possible in mscap to express functions on a speaker related to its associated audio object. These calls will need to be rewritten as accessing the associated audio object's properties.

| | |
|-------------|------------------------------|
| Name | getSourceMaxPosition |
| Manual page | 82 |
| Example | getSourceMaxPosition(source) |

| | |
|-------------|--------------------------|
| MSL rewrite | n/a |
| Comments | Not supported in mscapex |

| | |
|-------------|---------------------------|
| Name | getSourcePosition |
| Manual page | 82 |
| Example | getSourcePosition(source) |
| MSL rewrite | n/a |
| Comments | Not supported in mscapex |

| | |
|-------------|--------------------------|
| Name | isSourceOn |
| Manual page | 82 |
| Example | isSourceOn(source) |
| MSL rewrite | n/a |
| Comments | Not supported in mscapex |

| | |
|-------------|--------------------------|
| Name | isSourcePaused |
| Manual page | 82 |
| Example | isSourcePaused(source) |
| MSL rewrite | n/a |
| Comments | Not supported in mscapex |

| | |
|-------------|--------------------------|
| Name | isSourcePlaying |
| Manual page | 82 |
| Example | isSourcePlaying(source) |
| MSL rewrite | n/a |
| Comments | Not supported in mscapex |

| | |
|-------------|-----------------------|
| Name | isSourceFading |
| Manual page | 82 |

| | |
|-------------|-------------------------|
| Example | isSourceFading(source) |
| MSL rewrite | n/a |
| Comments | Not supported in mscape |

Location functions

Mobile Bristol had an implicit model of the user's unique location which could be accessed by the functions below. In mscape, the user's location is only known during a *Place* object's *OnLocationChanged* event. If the last location is needed later, the author should save the current location in state variables during the event handler, eg:

```
myX.Value = x;
myY.Value = y;
```

| | |
|-------------|-------------------------|
| Name | getUserLocationX |
| Manual page | 83 |
| Example | getUserLocationX() |
| MSL rewrite | n/a |
| Comments | Not supported |
| Name | getUserLocationY |
| Manual page | 83 |
| Example | getUserLocationY() |
| MSL rewrite | n/a |
| Comments | Not supported |

Random function

Allows a random number to be generated in the interval from *min* to *max* inclusive. In mscape, this capability is provided by the library object *Random* which must be created before use.

| | |
|-------------|--------------------------|
| Name | rand |
| Manual page | 83 |
| Example | rand(0, 2) |
| MSL rewrite | (new Random()).Next(0,3) |

| | |
|----------|--|
| Comments | Note that the number generated by the mscscape Random object will be <i>less than</i> the second parameter rather than <i>less-than-or-equal-to</i> as in Mobile Bristol |
|----------|--|

Changes if you use Flash movies

If your mediascape includes a Flash movies then you will have to do the following.

Make changes to the original Flash file and re-publish as a new Flash movie

Include the Flash movie into the new mediascape

Rewrite parts of the Flash ActionScript and mediascape scripts if required

Make changes to the original Flash file and re-publish as a new Flash movie

Copy the mediascape.as file into the same directory as the Flash file. (You can find the 'mediascape.as' file in the help file under; 'User Interfaces - Adobe Flash'. It is linked to under the heading 'Allowing Communication with the Mediascape'.)

Change the include line in the Flash ActionScript from:

```
#include "mb.as"
```

so that it includes this new include file, like this:

```
#include "mediascape.as"
```

Publish a new Flash movie from this file.

If your Flash file has ActionScript that passes data to the mediascape then you will need to change this before publishing the movie. See below for more on this.

Include the Flash movie into the new mediascape

Add the new Flash movie object by right clicking on the media object in the left hand bar of the mscscape maker. Then give the file name of the Flash movie when prompted for it.

Rewrite parts of your scripts if required

If you use scripting on either the Flash movie side or the mediascape side to communicate then you will need to change this as follows:

If you have used ActionScript to send data from the Flash movie to the mediascape then:

In the ActionScript you will have statements to send data like this:

```
sendXML("one", "two");
```

In order to work in MSL these should be changed to look like this:

```
fscommand("one", "two");
```

When you do this from within the ActionScript of your Flash movie you have to have a corresponding bit of script in the mediascape to listen out for the arrival of data being communicated in this way. This too has changed to that the old mediascape script event called:

```
mbFlashEvent
```

should be replaced by the new event:

```
OnFsCommand
```

If you have used scripting in the mediascape to send data from the mediascape to the Flash movie then:

In the mediascape script you will have calls like this:

```
sendFlashMessage("FUNC", "updateGPS", "fixed");
```

Where FUNC declares that you are calling a function, the 'updateGPS' is the name of the function and 'fixed' is the parameter that you are passing to this function. In the new MSL script in the mediascape you will need to replace such calls with:

```
mymovie.RunActionScriptWithParams("updateGPS", "fixed");
```

Or

```
mymovie.RunActionScript("updateGPS");
```

Here 'mymovie' is the name of the flash movie object. In order to do this you will need to know the name that you gave to the Flash movie object when you imported it (as above), this is usually the name of the Flash movie file, although you can change what it is called within the mediascape if you want to.

The actual script that forms the function being called ('updateGPS' in this case) does not need to be changed at all.

Possible changes due to HTML-page problems

Find out if you have you got HTML problems

If you have used HTML in your mediascape then there is a chance that you will have problems with the conversion. To find out if you have problems you need to view your HTML homepage (on the mobile device or in the mscape tester) and check the following:

If the page included images or Flash files confirm that they are there in the new converted version.

If the page had links to other HTML pages confirm that these links work by clicking on them.

If there are problems with either of these checks then it means that the conversion process was not able to copy all the parts of your HTML to the new mediascape and you will have to copy these parts yourself.

Copying missing parts of HTML

The missing files are files that are in the Mobile Bristol mediascape but that have not been copied to the new mscape mediascape. You have to find them in your Mobile Bristol folders and copy them to your mscape folders making sure that they are in the same place relative to the HTML home page.

FIND THE DIRECTORY WHERE THE FILES ARE

When you created your Mobile Bristol mediascape you were asked to say where you wanted to save it to. The standard place is on the desktop in 'My Documents' in a folder called 'My Mobile Bristol Projects', but you might have specified a different location. You were also asked to give the mediascape a name. This name was then used for the folder that contained all the assets for the mediascape. So if your mediascape is called 'mymediascape' and you used the standard location then you should look for a folder called:

'My Documents/My Mobile Bristol Projects/mymediascape'.

FIND THE FILES TO BE COPIED

The missing files should be in this folder or in a sub-folder (a folder within this folder). If they are in a sub-folder then you will need to copy the whole sub-folder.

If you can't remember all the files, then have a look at your HTML home page. It will have been copied unchanged to the new mscape mediascape but you can still see the Mobile Bristol version in your mediascape directory it will have the name that you gave it when you created it. If you open it up to look at the links to external assets (such as images, Flash movies or other pages) it will show you what files you need to copy over and where they are in relation to the HTML home page.

COPY THEM TO THE NEW DIRECTORY

When you converted your Mobile Bristol mediascape and saved it you gave it a name (for example 'mynewmediscape'). The folder that you saved it too is where you want to copy the missing files to. In this folder there will be a file called 'mynewmediscape.msl' and a folder for the content called 'mynewmediscape_content'. In this content folder is a folder called 'Web Pages' containing the copied HTML home page. This is your starting point for copying the files.

MAKE SURE THEY STAY IN THE SAME RELATIVE POSITIONS

In the Mobile Bristol mediascape folders the missing files all had a certain relationship with the HTML home page in terms of their location. You need to copy them over so that they maintain this relationship.

Thus, if the file was in the same folder as the home page then it needs to be copied over so that it is in the same folder as the copy of the home page in the 'Web Pages' folder described above.

Imagine if the file is in a folder called 'cat-pages' in the same folder as your HTML home page, then in the folder that your HTML home page was copied to you need to create a new folder called 'cat-pages' and copy the file in question into it.

You can confirm that you have copied all the pages to the correct places by performing the check described at the beginning, but this time perform it more extensively by visiting all the files that the home page links to.

If you had an HTML home page in a sub-directory...

When you were creating your Mobile Bristol mediascape the usual place to put your HTML home page is in the 'mymediascape' folder as described earlier. However, you could have put it in a sub-folder. If this is the case the conversion process will have copied this sub-folder and all of its contents, and the only reason that it won't be working as expected is if one of the HTML pages links to a file that is outside the sub-folder. The process for copying the files is the same, the only difference being the location of the HTML home page.

Having problems?

If you are struggling with this operation then it may be worth contacting someone in your organisation who is more familiar with moving files around, for example the person that set your computer up.

Background information

This information is not vital but it may be useful if you need to deal with any other issues that may arise.

With Mobile Bristol you could specify one special HTML home page as a start page, and this page could then link to other pages if required.

With mscape mediascapes HTML pages are better integrated into the overall model. You can have many HTML pages and they are treated the same way that other media types are treated. Just as audios and images are grouped together in the left hand column of the mscape maker so too is there a place where HTML pages are grouped together.

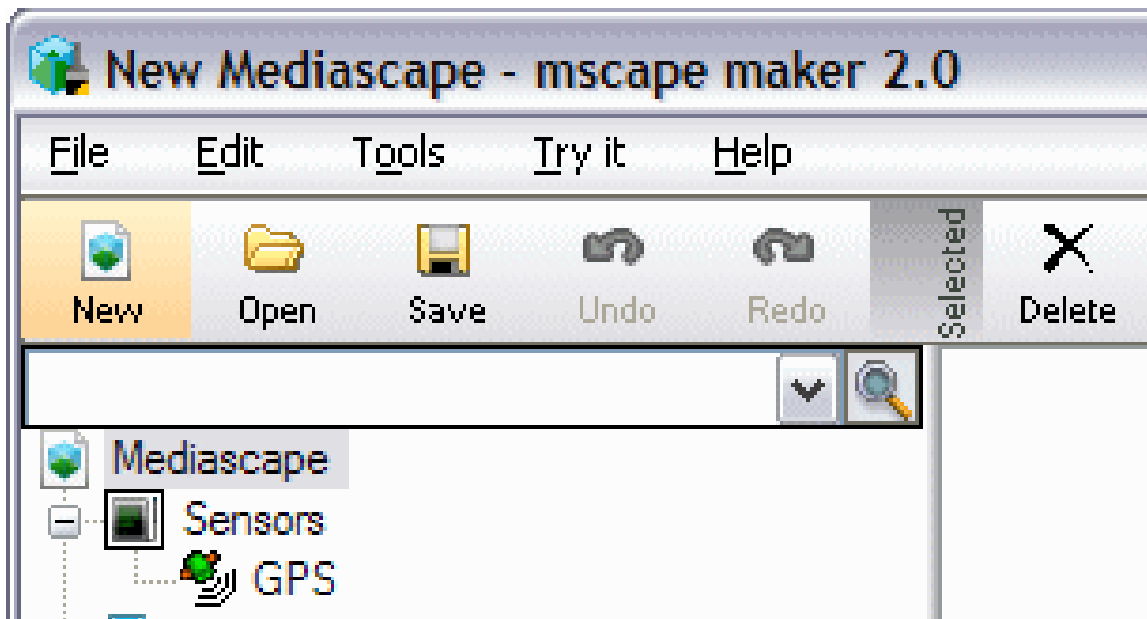
When the conversion process sorts through a Mobile Bristol mediascape it converts this special case HTML page into one HTML page object in the mscape mediascape. It also adds a 'show' instruction in the 'onLoaded' event in mediascape so that the result exhibits the same behaviour as the Mobile Bristol mediascape, but now it is mimicked in the new mscape environment.

How to Create a New Mediascape

Start the mscape maker by clicking on the *Start* bar and selecting it from the mscape group :



Click on the *New* toolbar button in the top-left corner of the mscape maker



Mediascapes in the Field

Low Batteries

Mediascapes involve using technology in the outdoors environment. The only power available is battery power to make sure that everything is well charged up before you set off. Remember that charging batteries up is not something that you can do ten minutes before you leave, you have to have it all charged well in advanced.

Other useful tips regarding batteries are of course to take some spare batteries if you have them, and again make sure that they too are fully charged.

If you are demonstrating the mediascape to a group of people then you can get round battery problems by having some headphone splitters (a component that plugs into the headphone socket on the device and has two headphone sockets on it, thus allowing you to plug two sets of headphones into a device). In this way if you have failing batteries you can get some of the audience to go round in groups of two while sharing one device.

Bad Weather

Any event that takes place outside is subject to the weather. The worst factor is rain. Take precautions to make sure that equipment cannot be damaged by rain. House it in waterproof rucksacks or other carrying bags.

Hot weather can also cause problems. Screens can be difficult to read in bright sunlight and wearing headphones while carrying equipment around in the summer heat can be very wearing. Set up your 'centre of operations' in the shade if possible and avoid doing things in the couple of hours after mid-day when temperatures are at their hottest.

Headphone Distraction

If you, or others under your supervision, are using a mediascape that requires people to wear headphones then bear in mind that listening on these can interfere with your ability to hear such things as traffic and bicycle bells.

Make sure people are warned about the environment and make sure the environment is not hazardous in any way to people who may be unable to hear all the sounds around them.

It is possible to minimise the effects by opting for headphones that cover the ears rather than using the 'in-ear', bud-like ear phones which cut out more sound from the environment.

At the time of writing this document Carl Kruger, a state senator in New York is pushing for a fine for pedestrians wearing headphones while crossing the road after two of his constituents had recently been killed in traffic. In one case bystanders even shouted warnings which the victim could not hear.

Distracting Screens

Having visual media as part of your mediascape means that the person using it is expected to spend some of their time watching the screen. This can happen when they are stationary or when they are moving, depending on the design of the mediascape itself.

Be aware of the distracting effect that screens can have and make sure that people are not expected to walk while looking at the screen in areas that are physically dangerous, for example close to cliffs, rivers or roads.

Be especially aware if you are dealing with large groups of excitable children where an added danger is having them run into one another!

Distracting Video

The drawbacks with headphones and those with screens are combined when video is included in the mediascape. Be aware of both sets of issues.

Physical Dangers

If people are distracted they are not just a danger to themselves in certain situations, they can also be a danger to the physical environment around them. People lost in the world of the mediascape may inadvertently trample through flower beds if they are being chased by virtual dragons.

So think carefully about where the mediascape is being sited, where will the busy parts of the mediascape be located? What about the fast parts where people tend to run? Are they in open spaces or next to something fragile?

Cables

Walking around a space while carrying digital equipment also involves a fair number of wires. If the space includes spiky vegetation or architectural features such as fence decorations then there is a chance that dangling cables could be snagged, leading to injury or damage to equipment.

Be aware of the environment you are using the mediascape in and make sure that cables are as well tucked in as possible and that people know of the issues.

Security

The concentration of technologies involved in using a mediascape means that there are issues surrounding having it stolen. The risks are higher when operating in busy city environments or in school environments where there are pupils from many classes in the area.

Make sure that you have good procedures for keeping an eye on your pool of equipment and for checking equipment in and out. One of the drawbacks with pocket-sized, portable gadgetry is that it is too easy to pop it in your pocket.

Suitability for the Audience

A mediascape is like any other piece of pre-recorded media, it can be hard hitting for certain people, especially children so think about who your audience is going to be and think carefully about graphic sound effects or colourful language.

In particular have special advice for adults with children with regard to behaviour and content of the mediascape.

Giving Instructions

If your mediascape is going to be used by other people apart from your immediate circle of friends then you will probably need to give instructions out. If you are going to be giving out mobile devices to only a few people then you can probably rely on spoken instructions. If you suspect that there may be many people using it, or it is complex enough to require quite a bit of explanation then it may be better to prepare a printed sheet of paper to

be copied and handed out with the mobile devices. These sheets could be laminated if you are planning on going ahead even if the weather is a bit bad.

An alternative approach that you might want to consider is to have the initial instruction on a sheet of paper and then the rest of the instructions in spoken form as part of the mediascape itself.

Loaning out Mobile Devices

Remember that you are probably operating in a public space and that even basic player kits are quite expensive bits of equipment. Make sure you have a robust process for lending out the player kits. Think about how you will cope with a rush of people wanting one. What happens if you do not have enough? How will you ensure that unscrupulous participants don't just walk off with the equipment? What happens if someone breaks something?

Giving Support to Pairs and Groups

When you set up a mediascape your audience will not just be single people. It may also include pairs of people, whole families or larger groups of friends. Make sure that you have systems in place to support them and make their group experience a good one.

You might want to think about having headphone splitters and extra headphone sets so that multiple people can use one mobile device.

Protocols of the Environment

When you deploy a mediascape the place that you choose to deploy it may have certain social and behavioural protocols attached to it. Think of how your behaviour is different in a public library for example. Or think of how you should behave in a stately home, or a park etc.

In particular think carefully about gaming elements and the inclusion of time based challenges in the mediascape where that mediascape is to be deployed in contexts that are not compatible with lots of running around.

There are a number of issues that you should bear in mind to ensure that the process of experiencing a mediascape runs smoothly and is as enjoyable as possible.

If you are just showing a mediascape to your friends, if you are organising a larger public mediascape event or if you are in the process of designing your own mediascape then there are points in the list that follow that are applicable.

How to Display Some Text on the Screen

There are several solutions to this:

Create an image with the text on it

Importing and displaying images is easy in mscapemaker. If you can create an image with your text on it then displaying it is simple.

This simplicity has to be balanced against the fact that you might have problems making sure that the text is legible as some mobile devices have very small screens. Also making any changes to the text becomes a time consuming process.

See [Using images in a mediascape](#)

Display the text with HTML or Flash

Both of these can be used to create and display text. Both offer the ability to alter the text as the mediascape is playing but HTML is probably the better option of the two as it is more text oriented.

See [How to use HTML \(web page markup\) in a mediascape](#)

See [How to use Adobe Flash in a mediascape](#)

How to Show Several Images in Sequence to the User at a Particular Point

Sometimes you may want to show more than just one image to the user. There are several ways of doing this:

Using a slideshow

A slideshow is a sequence of images and/or videos, a bit like a Microsoft Powerpoint presentation. They are simple to put together in the mscapemaker and you can add an audio as a background to them.

See [How to use slide-shows](#)

Using HTML or Flash

Both of these can be used to put together a sequence of images. They are more complex to create and they depend on using other software but they could be quicker if you already have experience with them.

They also offer the ability to do more complex things with sequences of images such as allowing the user to control the timing of the sequence etc.

See [How to use HTML \(web page markup\) in a mediascape](#)

See [How to use Adobe Flash in a mediascape](#)

Using hotspots in images

A hotspot is a rectangular part of an image that the user can click on. Importing an image and adding one large hotspot is pretty easy.

It would be possible to set up a hotspot that loaded a second image when the user clicked on it, and then to repeat this to create a chain of images that would load the next one when clicked.

This would be quite fiddly to set up and it would be difficult to adjust the order if required, but it could be easier than using HTML or Flash if you are unfamiliar with them.

See Using hotspots in an image

Free Online Sources of Media

To help get you going on your mediascape you'll need some audio, images, or videos. You can just grab things off the web, but you should check the license terms of anything you use - for instance just taking an image off Google Images will almost certainly be breaking someone's copyright terms.

However, there are quite a few places on the web that offers up media files that are both free to download and free of copyright restrictions (this is particularly true if you are using the content for non-commercial projects).

Some of our favorites are listed here..

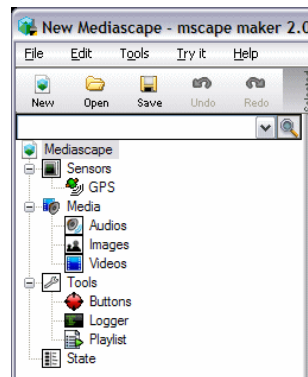
- [The FreeSound Project](#) - A great site for **audio**, many hundreds of thousands of sounds, both from real-world and electronic sources.
- [SoundSnap.com](#) - Another great source, this is particularly good for sound effects.
- [FlashKit.com](#) - A site that is aimed at users of Adobe Flash, but the media available here can be used for mediascapes too. They have a great selection of **background music** that you can use to add atmosphere to your mediascape as well as a selection of flash-based animations and **sound FX**. Note: You need to be careful with the licensing terms of the media on this site, authors often specify their own license restrictions in the description of the media file.
- [Flickr](#) - An old favorite. There are many free images on here, but again, check the terms before you use an image.
- [creativecommons.org](#) - A portal for all things [creative-commons](#).

How to Import Media Into a Mediascape

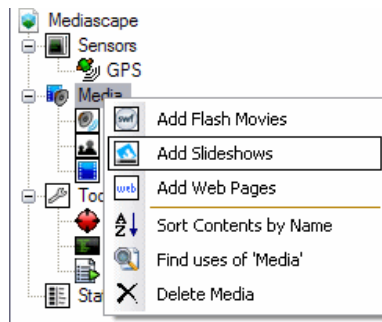
The mscape platform allows a number of different media types to be used in mediascapes - audio, images, video, Flash movies, HTML pages and slideshows. Before a piece of media can be used, however, it must be imported into the mscape maker. This tutorial will show you how to do that.

Let's begin by [starting the mscape maker and creating a new mediascape](#).

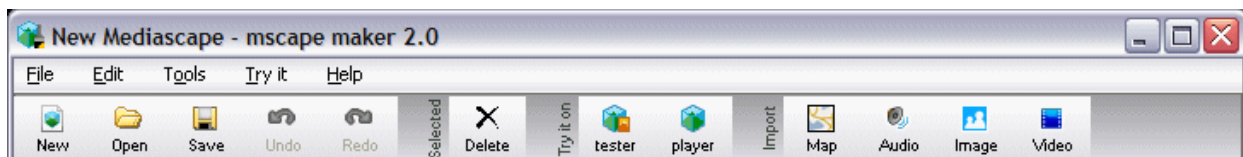
Notice the panel on the left hand side of the mscape maker. You will see a section labeled Media containing three subsections: Audios, Images and Videos:



This structure reflects the default mediascape template that is loaded when the `New` button is clicked. But what if I wanted to use a Flash movie or a slideshow? Easy, just right-click on the `Media` section and select the appropriate menu entry:

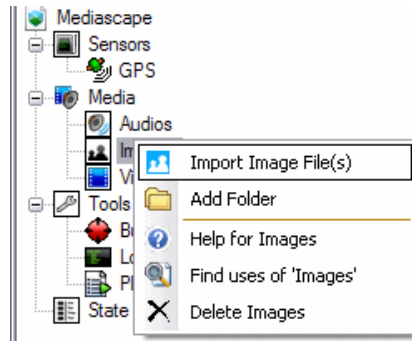


However, for this tutorial, let's import an image. There are two ways to do this: First look at the toolbar at the top of the mscape maker:

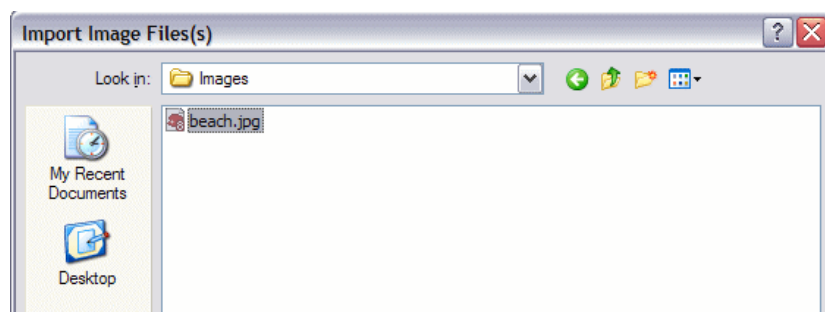


And click on the `Image` button in the `Import` section.

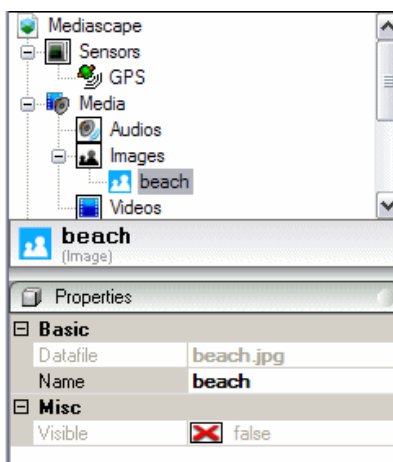
Alternatively, right-click on the `Images` section and select `Import Image File(s)`:



Whichever method you choose, the effect will be to bring up a file dialog that will allow you to find and select the image file:



Select the image that you wish to import and click Open. Now you should be able to see a new Image object in the Images section with the same name as the file (without the extension). If a script object with that name already exists, the new image object will be given a unique name such as beach01 or beach02.



This screenshot also shows the properties of the new image object in the panel immediately below the list of script objects. This allows to check that you have imported the right file and change some of the image's properties.

That's it. You are now ready to use the image object that you have just imported in the mediascape. Follow the same procedure to import any of the other media types, and have fun.

How to Import an Audio File

Click on 'Audio' in the 'Import' part of the top button bar, (alternatively you can right-click the 'audios' object in the left-hand panel and select 'import audio file' from the menu). Use the prompt to locate the audio file and click on the 'open' button.

The audio file becomes an audio object and appears in the left-hand panel under the 'audios' object.

It has the same name as the file, however you can edit this if you want to change it to something else. Simply click once on the name to select it, and then after a short pause click on it again to edit it.

If you want to see pictures showing how you import media then click have a look at [this page](#).

File names

When you start bringing resources into a mediascape you have to be a bit careful with their names. The mediascapes in mscape maker are based on a scripting language, like a programming language, if you start bringing resources in that have names that sound like bits of script then the mediascape gets confused and may not work.

For example importing an audio file called 'try.wav' would cause problems since 'try' is a word that means something in the scripting language. Here is a list of the names that you should avoid: [Terms to avoid](#)

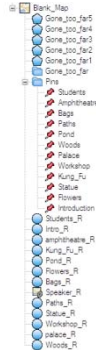
(In the future we will be adapting the mscape maker so that it stops you doing this before it gets

How to Make a Mediascape Using a Blank Map

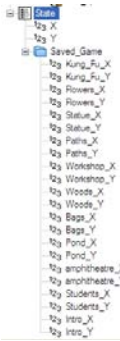
These are the rough steps required when making a Mediascape like the Gulbenkian Gardens, where you do not have access to a map and do not wish to use one of the wizards.

In the maker:

- 1) Create blank map (see instructions below)
- 2) Create all the regions



- 3) Make a new folder in the State section (Saved_Game) and create number variables to store the x and y coords for all the regions. Also create two variables outside of the saved folder to temporarily store the current X and Y values. e.g



- 4) For the Map on the onLocationChange event add the following code so that whenever your location changes the new coordinates are saved into your X and Y variables so that you can use this information to set the new locations of the regions.

```
Script - Blank_Map
OnLocationChange
1 X.Value = x;
2 Y.Value = y;
```

- 5) Create an image (placer) with hotspot for each region



HOW TO MAKE A MEDIASCAPE USING A BLANK MAP

6) For each hotspot, in the onTapped event add the code to move the relevant region and store the new coordinates in the variables. It is also useful to make a small sound play when the hotspot is tapped so that you know it was pressed OK. e.g

```
Script - Hotspot01
OnTapped |
1
2 Intro_X.Value = X.Value;
3 Intro_Y.Value = Y.Value;
4
5 Intro.X = Intro_X.Value;
6 Intro.Y = Intro_Y.Value;
7
8 Pin1.X = Intro_X.Value;
9 Pin1.Y = Intro_Y.Value;
10
11 audYes.Play();
```

7) Add code to another hotspot to save the new values of the variables

```
Script - Hotspot12
OnTapped |
1 Saved_Game.Save();
2
3 audYes.Play();
```

8) Create another image (start) with a hotspot which when tapped loads the values from the variables in the saved store into the values for all the regions. E.g

```
Script - start
OnTapped |
1 Saved_Game.Load();
2
3 Intro_X = Intro_X.Value;
4 Intro_Y = Intro_Y.Value;
5 Kung_Fu.X = Kung_Fu.X.Value;
6 Kung_Fu.Y = Kung_Fu.Y.Value;
7 Flowers.X = Flowers.X.Value;
8 Flowers.Y = Flowers.Y.Value;
9 Statue.X = Statue.X.Value;
10 Statue.Y = Statue.Y.Value;
11 Paths.X = Paths.X.Value;
12 Paths.Y = Paths.Y.Value;
13 Workshop.X = Workshop.X.Value;
14 Workshop.Y = Workshop.Y.Value;
15 Woods.X = Woods.X.Value;
16 Woods.Y = Woods.Y.Value;
17 Bags.X = Bags.X.Value;
18 Bags.Y = Bags.Y.Value;
19 Pond.X = Pond.X.Value;
20 Pond.Y = Pond.Y.Value;
21 amphitheatre.X = amphitheatre.X.Value;
22 amphitheatre.Y = amphitheatre.Y.Value;
23 Students.X = Students.X.Value;
24 Students.Y = Students.Y.Value;
25
26 Pin1.X = Intro_X.Value;
27 Pin1.Y = Intro_Y.Value;
28 Pin2.X = Kung_Fu.X.Value;
29 Pin2.Y = Kung_Fu.Y.Value;
30 Pin3.X = Flowers.X.Value;
31 Pin3.Y = Flowers.Y.Value;
32 Pin4.X = Statue.X.Value;
33 Pin4.Y = Statue.Y.Value;
34 Pin5.X = Paths.X.Value;
35 Pin5.Y = Paths.Y.Value;
36 Pin6.X = Workshop.X.Value;
37 Pin6.Y = Workshop.Y.Value;
38 Pin7.X = Woods.X.Value;
39 Pin7.Y = Woods.Y.Value;
40 Pin8.X = Bags.X.Value;
41 Pin8.Y = Bags.Y.Value;
42 Pin9.X = Pond.X.Value;
43 Pin9.Y = Pond.Y.Value;
44 Pin10.X = amphitheatre.X.Value;
45 Pin10.Y = amphitheatre.Y.Value;
46 Pin11.X = Students.X.Value;
47 Pin11.Y = Students.Y.Value;
48
49 MapDisplay.Show();
```

9) Add code so that the start and placer images can be shown by pressing the hardware buttons (or you could make an additional image with hotspots which is shown when the Mediascape is first loaded or when the button on the *MapDisplay* is pressed, which links to these two images)

Outside:

- 10) Load the Mediascape and bring up the image with the region hotspots
- 11) Walk to where you wish to place a region and tap the relevant hotspot, then repeat until you are happy with the positions of all your regions.
- 12) Save the variables by pressing the save hotspot
- 13) Exit the Mediascape and re-load it, this time press the start hotspot so that the saved locations are loaded making the regions move to where you placed them, you can then experience the Mediascape as normal.

In the Maker:

14) If you wish to make it so that the regions are fixed to their new positions so that you don't have to load the saved data every time you wish to play the Mediascape you will have to transfer the saved data from the device onto your PC. (Content folder for your Mediascape -> _stored -> Store00.xml)You can open the XML file in a simple text editor. Set the X and Y in the Properties box for each of the regions to the values that you saved in the xml file.

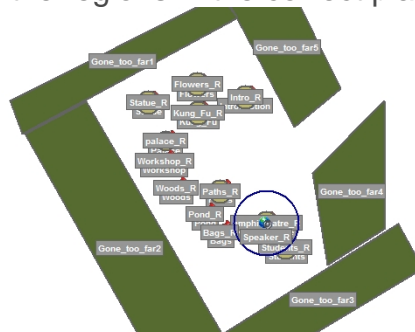
```
<?xml version="1.0" encoding="utf-8" ?>
<NotifyArrayList>
  <Number Value="361786.98154344875" Name="Kung_Fu_X" />
  <Number Value="178234.73551764747" Name="Kung_Fu_Y" />
  <Number Value="361803.96262064279" Name="Flowers_X" />
  <Number Value="178189.22623073938" Name="Flowers_Y" />
  <Number Value="0" Name="Statue_X" />
  <Number Value="0" Name="Statue_Y" />
  <Number Value="0" Name="Paths_X" />
  <Number Value="0" Name="Paths_Y" />
  <Number Value="0" Name="Workshop_X" />
  <Number Value="0" Name="Workshop_Y" />
  <Number Value="0" Name="Woods_X" />
  <Number Value="0" Name="Woods_Y" />
  <Number Value="0" Name="Bags_X" />
  <Number Value="0" Name="Bags_Y" />
  <Number Value="0" Name="Pond_X" />
  <Number Value="0" Name="Pond_Y" />
  <Number Value="361784.94381417549" Name="amphitheatre_X" />
  <Number Value="178149.83013162192" Name="amphitheatre_Y" />
  <Number Value="0" Name="Students_X" />
  <Number Value="0" Name="Students_Y" />
  <Number Value="361710.22707450716" Name="Intro_X" />
  <Number Value="178215.03746808058" Name="Intro_Y" />
</NotifyArrayList>
```

15) If you want to see where the regions are in the *MapDisplayer* add a Pin to the centre of each of the regions, and set *ShowOnMapDisplayer* to be true in each Pins Properties box. At the same time that the regions positions are moved and saved, you should make that same value the position for the Pin. This will also help to re-centre the map.

How to make a blank map:

- 1) Outside - Using a GPS device stand where you wish the centre of your map area to be and note the latitude and longitude.
- 2) At computer - Using a coordinate transform program e.g <http://www.gps2cad.com/coordtrans/coordconvert.aspx> convert the latitude and longitude into UTM data grid coordinates.
- 3) Imagine a square large enough to cover the area for you mediascape where the centre of the square is the UTM coordinate you recorded earlier. Calculate the coordinates for the corners of your square from the centre value.
- 4) Create a blank square image in a paint program.
- 5) Using the create map from image tool to create a map using the blank image and the coordinates you calculated.

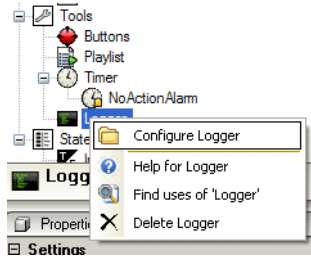
Here is the final blank map with all the regions in the correct places viewed in the maker



How to Track Your GPS and Replay Your Route

As you walk around testing your anchored mediascape it is really useful to be able to record your route so that you can see how well your GPS behaves and whether there are any problematic areas.

First you need to open your anchored mediascape in the mscapemake. Then you need to configure the Logger object to log your GPS on Moved events. Right click on the Logger object and select Configure Logger.



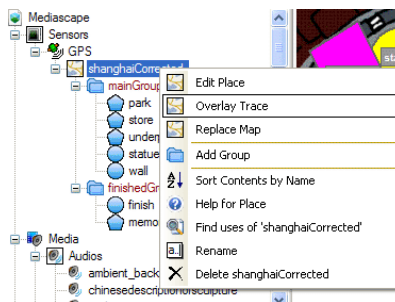
In the dialog box which pops up select the map object and tick the OnLocationChange property. Select OK to close the dialog box.



Now copy this mediascape onto your device to go out and trace your route. Active-synch your device and choose File – Save to PDA from the mscapemake. When you load and run this mediascape it will output a trace of your GPS readings to a file called Log in a folder called _Logger inside the content folder of your mediascape. (The output file name is specified in a property of the Logger).

To see your output trace. First you need to copy the log file onto your pc. Find the Log.txt file on your device or SD card. You can either active synch your device and browse to it or put the SD card directly into your pc and use file explorer.

Then open up your mediascape in the mscapemake. Right click on the map object and choose Overlay Trace.



You then need to browse to the location that you copied the log file to. The GPS trace will appear on the map as a series of dots and red lines. To clear the trace Right click on the map object and there will be a Clear Trace option.

Having Troubles with Your GPS

The mountain icon in the left of your screen shows the status of your GPS. It will change colour depending on the state of your GPS



It can take several minutes for your iPAQ to see enough satellites to get a good GPS fix. It helps if you stand in an open space with a clear view of the sky. Once you get a GPS fix it will may take while for the readings to settle down they may be quite erratic to begin with.

My Device Stays on the Grey Mountain

If You Are Outside

- Make sure that you are in an open space with a clear view of the sky. If you have a lot of buildings or trees around you then these will restrict the devices ability to see GPS satellites and it may take longer to get a location fix.
- If your device also has a satellite navigation program on it like TomTom then try starting this up to see if it also has problems getting a GPS fix. If it gets a connection exit the program and try mscap again.
- It may just be a very bad time of day with not many satellites orbiting over head. So please try again at a different time of day.

If You Are Inside

- GPS does not work indoors. Sometimes you can get a GPS location fix if your device is by a window but you will not be able to play mediascapes inside.

My Device Is Showing a Red Mountain

- If the device is showing the red mountain then either mscap player is not set up correctly to communicate with the GPS, or the GPS itself is not working.
- The document [Setting Up GPS](#) in mscap player guides you through the process of correctly setting up your GPS device.

If You Have An RX59-Series Travel Companion

Some rx59-series Travel Companion devices use by default GPS settings designed for vehicles rather than pedestrians. This causes some difficulties when these devices are used by pedestrians, as the GPS position tends to stay put until you have moved around 20 meters at which point the position will jump to the correct place. The mscap player installation should have automatically set your device to use pedestrian mode. Note that setting the GPS to pedestrian mode does NOT have any effect on the built-in TomTom navigation program - it will still perform just as well.

Using Maps in Mediascapes

Mscape Maps Include Co-ordinates

A mediascape consists of media objects spread about on a map. A map for a mediascape is no ordinary map. It is based on an image of a map, but that is not all; when you use the mediascape with your mobile device, it knows where it is in terms of GPS co-ordinates, but it needs to know where it is on your map. Thus you need to have co-ordinates associated with your map. In this way the device can say; 'Right, I'm at GPS co-ordinates XYZ, so where's that on the mediascape map? Right, I'm in the region with the dog barking sound so I had better play it...'

This special sort of map is the map that needs to be used when you create a mediascape with the mscape maker. There are different ways of getting one.

(Sometimes, this type of map with co-ordinates may be referred to as a 'maplib')

*Map Service will not
be available after
March 31, 2010*

Using The Mscape Map Service

Click on the map icon in the top bar of mscape maker to import a map, you will be asked if you want to import a map from the mscape map service or from a file. Creating a map file to import is dealt with in the sub-section below.

The mscape map service is a customised map making service. It is easy to use, the controls are similar to many mapping/satellite-photography services already online. You can switch between maps and satellite photographs, you navigate across a map to the area you require, and then click a button to import the map of that area into mscape maker. The co-ordinates are automatically incorporated into the map.

This service is continually being improved and we are working to overcome any drawbacks that you may experience in speed, scope or image quality.

Creating Your Own Map

Mscape maker also allows you to create your own 'home made' mscape map, you will need an image of a map and then you have to do a bit of work to find the co-ordinates and associate them with the map.

Getting a Map Image

To start with you need an image of a map. You could use a scan of an old map, or of a sketch that some children have made. Trace an existing map either on a computer on with tracing paper. Or you could request a copy of a map from a more conventional mapping company.

In order for it to be used in the mscape tools, the image of the map has to be in one of these formats; BMP, JPG, GIF, (The BMP format is a common Microsoft format and JPG and GIF are formats often used for images on web pages, you can usually tell what an images format is by the letters after the dot in the file name. Note that JPG is also called JPEG).

Once you have the image of your map you have to get co-ordinates for it and then use mscape maker to associate the two things. There are two approaches to gathering the co-ordinates to give to mscape maker.

Either you can get them from a second party and then key them in 'by hand', or you capture the co-ordinates by going outside to the area of the map with your mobile device and using a special tool to gather the co-ordinates as you walk around and the co-ordinates can then be transferred to your computer ready for use in the mscape maker.

Gathering The Co-ordinates From A Second Party

Co-ordinate Systems

It is not much good just having some co-ordinates. The information; 'The treasure is buried at 44 32' is useless if you do not know what co-ordinate system is being used. Is this latitude and longitude? Is it number of paces from the old oak tree on Skull Island?

You need to have both the co-ordinates and the information about the co-ordinate system being used. The co-ordinate system will depend on which county you are mapping. Different countries have different systems and there are some international systems as well. The key items of information about the co-ordinate system are: The country, the co-ordinate system being used, and the zone of that co-ordinate system. (some of these co-ordinate systems need to know which zone you are talking about, but not all of them). Once you have all that information you can give the co-ordinates and know that they will be correctly interpreted.

There are several ways of getting co-ordinates to attach to your map.

Contact Whoever Provided The Map

If you got your map from an official source, or someone who is knowledgeable about maps then they are the best point of contact for finding out about what the co-ordinates are.

In order to pin down the map precisely you need to have the co-ordinates of the four corners of the map (in actual fact three corners is enough but having four will help to cross check the information for mistakes).

Once again do not forget to get the details of the co-ordinate system being used from them as well as the actual co-ordinates.

Get Them From www.streetmap.co.uk

This site is UK specific.

It lets you click around on a huge map and zoom in and out until you find the bit of landscape you are interested in.

When you click anywhere on the map it draws an arrow pointing to where you clicked and the co-ordinates of that point appear in the text area below the map. Unfortunately it also puts the square you clicked in at the centre of the screen so things do get a bit confusing and you have to re-center the map by clicking in one of the other squares to make it the centre. It is not too complex and after a bit of clicking and watching you will quickly get the hang of it.

You have to click in all four corners to get the co-ordinates for whatever map you are planning on using. Write them down or cut and paste them into a text file (keeping track of which is which).

The other thing you always need to know is which country and co-ordinate system it is. There may be similar services for other countries, let us know if you find one particularly useful.

Capturing The Co-ordinates Outside With The Map-Aligner

When you use the map-aligner you will be taking the map outside on a mobile device and you will click on the map to say, 'This bit of the map here is the place where I am now standing...'

The mscape map-aligner is part of the mscape toolkit and is installed on the mobile device when you install the mscape player included in the toolkit. To capture the co-ordinates for your map image you must also transfer the map to the mobile device as well.

When you are outside you run the map-aligner and use it to open your map image. The map-aligner will then guide you through the process of capturing co-ordinates until there are enough points for it to have a good idea about how the two relate.

The advantage of this method is that you don't have to spend too much time worrying about co-ordinate systems and you don't have to write down and check lots of numbers. The disadvantage is that you have to do a fair bit of wandering around outside in the exact location that you are designing your mediascape for.

Here's how you can find the map-aligner on your mobile device:

- Tap Start / Programs / File Explorer
- Navigate to My Device / Program Files / mscape
- Map-aligner is there in the list.

Entering The Co-ordinates Into Mscape Maker

Once you have the co-ordinates from either of these two methods you can combine them with the map image using the mscape maker. You go to the 'Tools' menu and select 'Create map from image'. You are then led through a dialogue that prompts you to import the image and then provide the co-ordinates either by importing the file you made with your mobile device or by typing them in by hand.

Setting Up GPS on Mscap Player

In order for mscap player to use your GPS, it must be correctly configured so that it knows where the device is. On many devices with built-in GPS such as the iPAQ rx5900 or 610 series models, the program should find the GPS automatically. In this case mscap player will boot straight to the *load mediascape* screen.

For other devices you may need to set up the GPS manually.

If you're using an external GPS make sure that you have connected it to your device before starting mscap player.

If you are using a globalsat bluetooth GPS receiver, see the [Bluetooth GPS Set-Up Guide](#) provided by globalsat for instructions on pairing the GPS with your mobile device. For other BT GPS devices, see the manufacturers' instructions

The following screen will appear when mscap player is started if it cannot automatically determine the location of the GPS device.

Here you can choose the *COM port* that your GPS is connected to. Unfortunately different devices use totally different port numbers and names so there are no hard and fast rules as to which one to choose. Generally if you are using a bluetooth GPS then one or more of the COM ports should be named appropriately. If you choose a port and it turns out to be wrong, mscap player will inform you and offer you an opportunity to select a different port. Once you have found the correct port mscap player should switch to the *Load Mediascape* screen.

The port information is saved so next time mscap player is started, it will use the saved settings.

How to Simulate GPS on Mscape Player

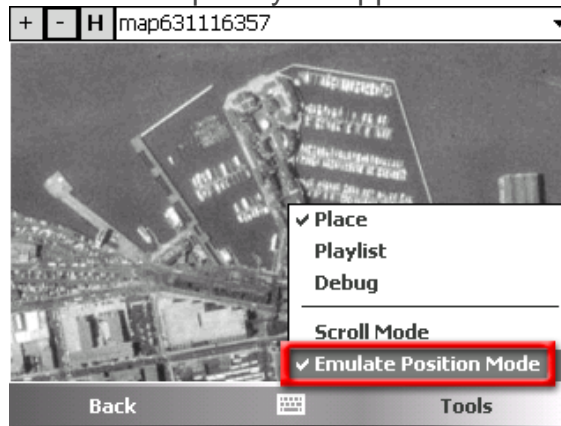
Sometimes you may want to simulate GPS on your mobile device when a GPS fix is not available, for example when you are inside, or when you don't have a GPS device.

You must first enable debugging tools in the mediascape that you wish to debug. See the document [EnablingDebuggingInMscapePlayer](#) for details on how to enable this feature

Simulating GPS Points

Once you are in Designer Tools mode, ensure that the 'Place' view is active by selecting 'Place' from the Tools menu. This will show you the map that your mediascape is using. If there is more than one map, you can switch by using the drop-down box at the top of the screen. You can zoom in and out using the + and - buttons, and recenter the map using the H (short for 'Home') button..

Next, select 'Emulate Position Mode' from the Tools menu. This means that when you tap on the map, a GPS position will be simulated at the point you tapped.



Tap on the map and a small character will appear on the screen. Tap around the screen to simulate positions.



If you want to see what would be on the users screen, for example to see what image is currently showing, go back to the mediascape by pressing the Back button.

Simulating the OnGotFix Event

Some mediascapes such as [Doubloons](#) or [Stamp the Mole](#) may not 'start' until the GPS gets a fix.

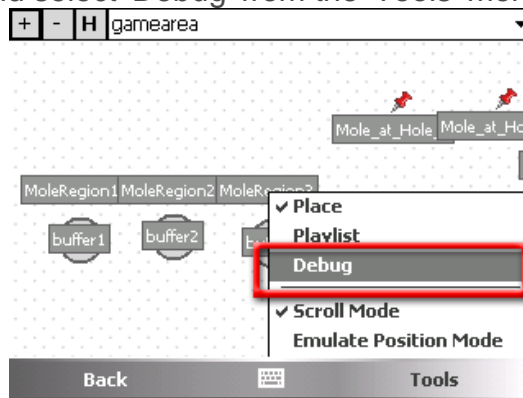


Waiting for GPS fix ...

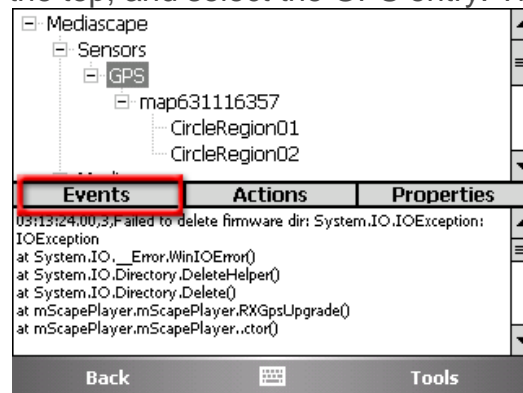


Simulating positions as described above will NOT trigger the OnGotFix event, so this must be done manually.

Go into Designer Tools mode, and select 'Debug' from the 'Tools' menu.



Scroll the upper panel right up to the top, and select the GPS entry. Then tap on 'Events'



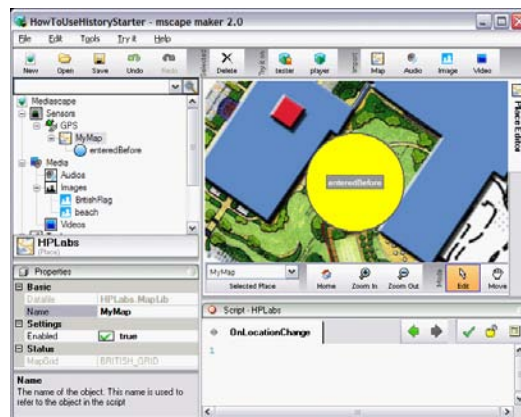
Select 'OnGotFix' from the list and click the 'Invoke!' button.

Go back to the mediascape by pressing the 'Back' button

How to Use the User's History When Triggering Media

One of the ways to enrich a user's experience of a mediascape is to make the behaviour of the mediascape dependent on what the user has already done. For example, imagine playing one piece of audio when the user enters a region for the first time and a different piece of audio when the user returns to that region. In the mscap platform, such behaviour can be simply enabled through a combination of basic scripting and built-in properties. This tutorial will show you how.

Let's begin by [creating a new mediascape](#), [importing some media](#), [adding a map](#) and creating a region. Or, to save time, just download this mediascape and open it in the mscap maker. You should see something like this:



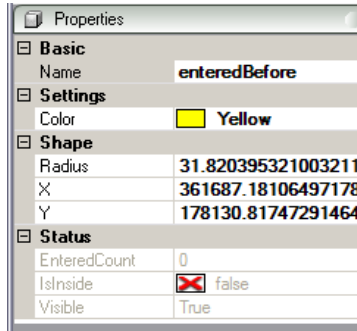
As you can see, this simple mediascape has a single region, named *enteredBefore*, and two images - *BritishFlag* and *beach*. We will return to audio shortly.

Now, click on the *enteredBefore* region in the Place Editor panel and type the following code into the *OnEnter* event handler in the Script Window:

```
if (enteredBefore.EnteredCount == 1)
{
    beach.Show();
}
else {
    BritishFlag.Show();
}
```

Try running this mediascape in the tester. First, click inside the *enteredBefore* region - you should see a picture of a beach appear in the top left panel. Now click outside the region and back in again. This time you should see two islanders and the British flag. What do you think would happen if you left the region and entered it again? Try it and see!

This bit of script works by testing the value of a built-in property when the region is entered. Go back to the mscap maker, select the *enteredBefore* region and look at the property panel in the lower left corner. Note that you may need to resize or scroll this panel to see all the properties.



Many of these properties will be self-explanatory. We want to focus on the first property in the *Status* section - *EnteredCount*. This will be grayed out and you will not be able to change its value, unlike some other properties. That's because *EnteredCount* is a *run-time* property whose value is filled in automatically by the player while the mediascape is being used. You have probably worked out by now that this particular property counts the number of times the user has entered that region since loading the mediascape. Hence the logic we added to the region's *OnEnter* event handler.

So that's the basics done. We can test whether the user has been to this region before and display different images accordingly. You might also see how you could use a similar test to see whether the user has already visited a *different* region, or to show different images on each of the first five visits to a region.

As you might expect, there are other properties that you can test, both for regions and for other script objects. Let's illustrate this by returning to think about audio. We could use the same piece of logic just illustrated to play different pieces of audio when the user enters the region for the first time and on subsequent occasions. However, we might also want to ensure that the user hears all of the first piece of audio before she hears the second audio, even if she leaves and re-enters the region before that first audio has finished playing. We can satisfy those two requirements by slightly changing the event handler script to read:

```
if (ukintro.Played == false)
{
    ukintro.Play();
}
else
{
    uka3.Play();
}
```

The crucial change is that now we test the *Played* property of the *ukintro* audio rather than the *EnteredCount* of the *enteredBefore* region. You will find the new script in the *OnEnter* event for the new region *heardBefore*. Try running the mediascape in the tester and hear what happens as you move in and out of the region. By the way, for those of you wondering about the possibility of playing the *ukintro* audio multiple times if you re-enter the region before it has finished, don't worry. Calling the *Play* method on an audio that is already playing has no effect.

So that's it for this tutorial. Use the right-click Help menu on other script objects to find out what other properties you can use in this way. You might also want to think how user-defined state variables can be used in such tests ... but that is a topic for another article.

How to Let the User Press Buttons to Control the Mediascape

When you desing a mediascape you sometimes want to get information from the user. In its simplest form you just want the user to tap on a button to indicate something to the mediascape. There are several approaches to this.

Use an image with a hotspot

A hotspot is a rectangular part of an image that the user can click on. Importing an image and adding one large hotspot is pretty easy.

It is possible to add several hotspots but getting them in the right place is very much a 'do-it-yourself' activity and requires creating an image with rectangular buttons on and then measuring the positions and dimensions of those buttons so that you can key this information in.

See [Using hotspots in an image](#)

Using HTML or Flash

Both of these can be used to create buttons that the user can press. The fact that the button has been pressed can then be communicated back to the mediascape and appropriate actions taken.

See [How to use HTML \(web page markup\) in a mediascape](#)

See [How to use Adobe Flash in a mediascape](#)

Use the real hardware buttons on the mobile device

Many mobile devices have hardware buttons. It is quite easy to make your mediascape react to these buttons.

See [How To: Use Hardware Buttons](#)

Introduction to Programming

Article by Ben Calder (aka [blindfish](#)) - April 2009

Introduction

For those with little or no previous experience the idea of using programming in a mediascape may seem a little daunting; however complex interactions can be achieved using surprisingly little code. It's also worth pointing out that, for example, the simple process of playing an audio clip is done using programming code. Having a better understanding of how this all works will help you get the most out of the mediascape toolkit.

Structure

Programming languages work very much like written languages - you have to watch out for things like spelling and grammar (in programming speak: 'syntax'), but just as important is the overall structure of your document. When writing an essay you would normally start by setting out the structure in an 'essay plan', before writing anything down. It's a good idea to follow this approach when programming: try and sketch out what you want to achieve before worrying about the details of what code to use and the exact syntax. Of course before you can do this you do need to have some idea of what programming tools are available to you; this article aims to provide just such an overview.

A Brief Note On Objects

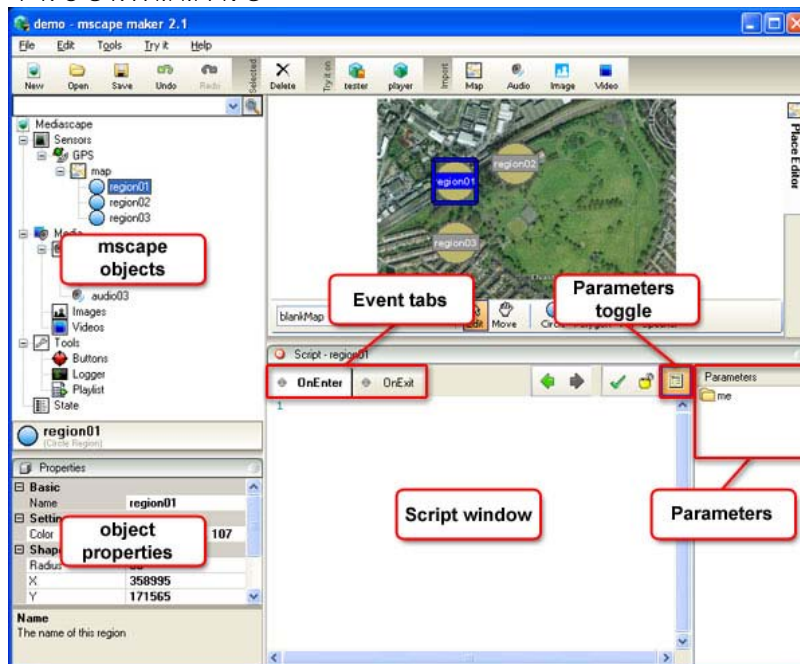
Throughout the article there will be frequent mention of 'objects'. Whilst this has a specific meaning in the context of programming languages, for the purposes of mscape development it is enough to know that an 'object' is simply a thing (for want of a better word). Any media you add, any sensors, tools or even state variables are 'objects'. As you will see objects have properties, and usually actions, associated with them and it is by manipulating these properties and using these actions that you create a mediascape experience.

Events, Parameters, Properties, and Actions

Events

Programming in the mediascape toolkit is 'event based'. This means any code you write will be triggered by a particular event. All mediascape objects have associated event tabs displayed in the Script window. So for example the top level Mediascape object has an 'Onloaded' and 'OnUnloading' event; an audio file has an 'OnStarted' and 'OnFinished' event; a region has an 'OnEnter' and 'OnExit' event and so on. Select the appropriate tab and add your code in the script window and it will be run when the event occurs.

The event names should be fairly self-explanatory; so whenever you add an object to your mediascape it's a good idea to look at the associated event tabs to get a better understanding of the actions that can be taken with it. Events will be the starting point of any interaction and therefore any programming you do.



Parameters, Properties and Actions

Parameters

Each event has associated 'parameters' which can be viewed by opening the 'Parameters Window' (click the toggle button to open this). Some parameters are specific to a type of event - for instance the GPS 'OnLocation' event has an x and y parameter which can be used to access the updated x and y coordinates of the user.

















However, for the most part the only parameter available is 'me', which is basically a short cut to accessing the object that triggered the event. To test this out type 'me' into the script window and follow this with a full stop. A pop-up window will appear with a list of words. Now delete 'me.' and try typing the name of the object the event belongs to followed by a full stop. So if you originally typed 'me.' into the OnEnter event of a region called 'myRegion' try typing 'myRegion' followed by a full stop. You should see the same pop-up window with the same list of words.

Finally go to an event that does not belong to 'myRegion', let's say the OnStarted event of an Audio track. Type 'myRegion' followed by a full stop - you'll again see the same pop-up. Now try 'me' followed by a full stop - this time the list will be different. So what is this pop-up list about?


Properties and Actions


As well as associated events, each object you add to your mediascape may have associated properties or actions. To make things easy for you the mscscape toolkit lists those available for a particular object whenever you type its name (or 'me'), followed by a full stop, in the script window. You can then use the mouse or cursor to select from the list, or start typing the name of a property or action. You'll notice that when you select from the list another pop-up will appear with a description of the currently selected item. This tells you whether it is a property or an action and gives you further information and an example of it in use. If you hit return (or double-click) the property or action code will automatically be added into the script window using the correct syntax.

The complete list of properties and actions for an Audio object:

-  Datafile
-  FadeDuration
-  FadeIn
-  FadeOutAndPause
-  FadeOutAndStop
-  Fading
-  Loop
-  Name
-  Pan
-  Pause
-  Paused
-  Play
-  Played
-  Playing
-  Stop
-  Volume

The icons to the left represent the type:


Property


Action

The difference between a Property and an Action should be fairly obvious. Properties describe the object: for example the x and y coordinates of a circle region, its radius and colour. When you select a circle region in the list of Mscap objects you'll see the object's properties in the panel below. You'll notice that the properties listed here correspond to those in the pop up. You'll also see that some properties are displayed in black whilst others are greyed out. Greyed out properties are read-only (i.e. you can access their values but are unable to change them), whilst those in black can be changed either in the panel or by using code.



Actions are things you can do with an object: for example you can play, stop or pause an audio track. Notice that when you add an action from the pop-up it is always followed by curved brackets: (). When looking through your code you can use this convention to distinguish between object properties and actions. You should hopefully have realised that some events will only ever be triggered if a

certain action is carried out first. For instance the OnFinished event of an audio will only ever happen if the audio is first played.

One final point to notice here is the distinction between using 'me' and an object name. 'me' is convenient if you're accessing the object that triggered an event, but you'll often want to start an action on another unrelated object, in which case you need to use that object's full name. A good example is playing an audio when someone enters a region: in the region's OnEnter event you type the name of the audio followed by the 'Play' action: myAudio.Play();.

Variable and Property Types

Properties belong to specific objects, but there are times when you might want to store a value independently of an object. This is where [variables](#) come into play. One important factor you will notice on reading the linked page is the issue of the type of variable you use. As with variables, properties come in different types and it's important to know what type you're dealing with: if you need to store it in a variable you need to use a matching type and type is also important when comparing properties and variables.

Reading Parameters, Properties, and Variables

Using the Logger

To demonstrate how to access the values stored in event parameters, properties and variables we'll be using the Logger tool. This can be used to display a message in the output window of the tester by using its 'Log' action. To test it out, in a new mediascape click on the Mediascape object and make sure the OnLoaded event tab is selected. In the script window type 'Logger' followed by a full stop, then select the 'Log' action (you can either double-click it or type 'L' followed by Return). In the script window you should now see:

```
Logger.Log("message");
```

Run your mediascape in the tester and, in the Output panel at the bottom, you'll see something like:

```
11:35:10,37,1,LOG: message
```

The numbers before the first comma represent the time of the event (don't worry about the other numbers). 'LOG' simply tells you that this event is a log generated by the logger. 'message' is what we wrote to the log. To confirm this close the tester and change the text in inverted commas to 'Hello world!':

```
Logger.Log("Hello world!");
```

When you run the tester the message displayed after 'LOG' is now 'Hello world!'. We'll go into more detail on actions later but for the moment you just need to know that anything you put between the round brackets of the Log action will be displayed in the logger. Any text message you want to display should be surrounded by "inverted commas" and, because of the way the log action works, there must always be at least one item of text in the message you send.

Being able to display plain text messages in the logger is already fairly useful. When using the tester you can, for example, confirm that the system has registered an event taking place. Try adding a log message to the OnEnter event of a region and see when this gets displayed in the tester. The logger

becomes even more useful for when you use it to display the values of variables, properties and event parameters.

Event Parameters

The reason you're most likely to use event parameters is to access information from sensors. In your mediascape click on the GPS sensor and select the '!OnLocation' event tab and if necessary toggle open the Parameters. You'll see that, as well as 'me', there are 'x' and 'y' parameters.

Accessing these is actually very straightforward: whenever you want to use the value of an event parameter in your code you simply type in the name of the parameter. Note that it's important to match the name, including any capitalisation, exactly otherwise you will get an error.

Let's put this into practice. In the GPS OnLocation event type the following:

```
Logger.Log("x coordinate: " + x);
```

Run the tester and click anywhere on the map. The message displayed shows 'x coordinate: ' followed by the current value of x. Each time you click on the map - representing a change in location - you will see a new message with an updated value for x.

Notice that we've introduced a new concept in this log: by using the plus (+) sign we can add the value of the parameter to the message so it all gets displayed together on one line. It's important to remember the plus sign: without it you will get an error. (In programming speak the plus sign is one of many 'operators'.) Notice also that I included a space after the colon inside the inverted commas. The code will work without it, but if you try removing it and run the tester you'll see why it's worth including.

Let's take this addition concept a little further. To display both the x and y coordinates in the log we could use the Logger twice as follows:

```
Logger.Log("x coordinate: " + x);  
Logger.Log("y coordinate: " + y);
```

However you can add as much as you like to a message so long as you remember to include inverted commas around any text and a plus sign between each element of the message. So we could display both x and y values on a single line using:

```
Logger.Log("x: " + x + " y: " + y);
```

This gives us a label 'x: ' followed by the value of x, followed by a label ' y: ', finally followed by the value of y. Again the spaces around the labels inside the inverted commas aren't essential but make it much easier to read the resulting log message. Notice how your code is automatically colour-coded depending on what it represents: anything between inverted commas (i.e. text) is blue, plus signs are red and the parameters stay black. This feature is useful for spotting errors in your code: for example if you missed out a closing inverted comma in the above code the following plus sign would be blue (i.e. part of the text) instead of red.

Properties (and variables)

Accessing properties isn't as straightforward as accessing event parameters but it's still easy enough. Let's say we want to access the Angle reported by the GPS sensor. This gives you the angle between

the last two recorded locations. As such it only provides accurate information when the user is in motion - not when they're standing still - but you might still find it useful.

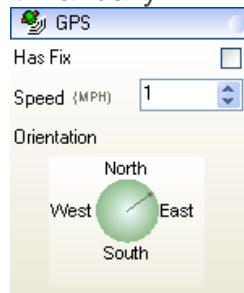
It is possible to access this value on the GPS OnNewAngle event as it is one of the event parameters, but what if you want to access it in a region's OnEnter event? This is one limitation of event parameters: they can only be accessed directly by your code when it is written in a specific event tab. Fortunately Angle is also a property of the GPS object so we can get at it through that.

First we type the name - matching spelling and any capitalisation exactly - of the object the property belongs to, in this case 'GPS', followed by a full stop, then followed by the name of the property, in this case 'Angle'. As previously mentioned, the moment you type a full stop after the name of an object a pop-up box will display all available properties and actions for that object. (If a pop-up box doesn't appear you'll know you haven't matched the name of the object exactly: check spelling and capitalisation!) At this point you can simply double-click the property you want or start typing it till it is selected and hit Return to auto-complete. As with object names you **must** match spelling and capitalisation exactly.

So to log the angle you can use:

```
Logger.Log("angle: " + GPS.Angle);
```

If you placed this code on the the OnEnter event tab of a region the logger would display the angle each time you enter the region. Note that in the tester the value of the angle is controlled by the 'Orientation' control under the GPS section. By default this is set to 0, so the angle reported in the tester will always be 0 unless you change it manually.



The mscap Tester GPS widget, including Orientation control: click and drag around this to update the angle.

The ObjectName.Property syntax works for all object properties: simply type the name of the object followed by the property you wish to access. The same approach also applies to any variables you create. The value of the variable is accessed through its property 'Value'. For example to log the value of a Number variable called 'currentX' you would use the following code:

```
Logger.Log("currentX: " + currentX.Value);
```

Setting Properties and Variables

In the previous section we saw how to access object properties - for example the radius of a CircleRegion or Volume of an audio clip - using the object's name followed by a dot followed by the property name:

```
Object.Property  
myCircleRegion.Radius
```

myAudioClip.Volume

Also remember that to access the value of a state variable you need to reference its 'Value' property:

myNumberVariable.Value

So far we've only looked at reading values and displaying them with the logger. This can be useful when *debugging* a mediascape because it allows you to check the value of variables at a given event (we'll see another use for reading variables in a later section) but it doesn't actually **do** anything. So let's see how to set properties and how this can be used to add interactivity to a mediascape.

Limitations

Once you have identified the property you want to change giving it a new value is fairly easy, once you have established two important points:

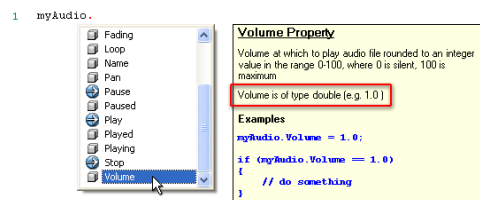
That the property can in fact be changed

It's important to understand that not all properties can be changed using code. You'll notice that the greyed-out (i.e. read-only) properties tend to correspond to equivalent event parameters. For example since GPS Speed and Angle are based on the user's movements it doesn't make sense to be able to change them using code, so the Mscape Toolkit simply doesn't let you change them.

The type of property you're dealing with

The value you give (in programming speak: 'assign') to a property should match the 'type' of the property. Like [variables](#) a property is in effect a 'container' and the value stored in it must match the type of container. If you try and assign a value that doesn't match the type you will either get an error or potentially unexpected results.

In practice the most common types of property you're likely to deal with are: Text (otherwise known as 'strings'), Numbers (there are actually several sub-types of number) and Booleans (things that can be either true or false). However in some cases there are more specialised types which must be assigned in a specific way. When you select an item from the property pop-up dialog of an object the second pop-up will include the type of that property and an example - e.g. "Angle is of type int (e.g. 1)".



pop-up for Audio Volume property with the type highlighted. 'double' is a number type.

The following table can be used to determine what type of value can be assigned:

| Type | Value |
|--------|--|
| string | Text. Always placed between "inverted commas". |
| int | Number - in this case a positive or negative, round number (no decimal places) |
| double | Number - a positive or negative number that can include decimal places |
| bool | 'true' or 'false' |

Setting A Property

Once you have established the type of property, assigning a value to it is fairly easy. You simply reference the property as you would when reading it, follow this with the *assignment operator* (a single equals sign: '=') followed by the value you want to assign. To see an example of this put into practice try adding the following code to a region's OnEnter event:

```
me.Radius = 30;
```

This line of code is known as a 'Statement' - we'll come back to this in a moment. First run the tester and click inside the region, thereby triggering the OnEnter event. Assuming the radius was not already set to 30 you'll see the size of the region change when you click inside it. Remember that 'me' references the object triggering the event - in this case the region being entered. You could also have written the name of the region followed by the rest of the code. Alternatively you could change the size of another region by referencing it by name. So if you have two regions ('region01' and 'region02') and want to change the size of 'region02' when someone enters 'region01' (for instance making it smaller and therefore more difficult to find) you would add the following code to region01's OnEnter event:

```
region02.Radius = 30;
```

Note that up till now we've been using the auto-complete feature provided by the mscape toolkit, which can be useful both as a shortcut and in order to avoid syntax errors. In this case we've added some extra code manually. You might have noticed that every line of code (or statement) added so far has ended with a semi-colon ';'. It is essential to include this semicolon otherwise you will get an error message. To understand why try changing your code to the following:

```
me.Radius
=
30
;
```

Run the tester and you'll see you get exactly the same result. This is because the toolkit ignores line-breaks in your code. Writing a statement on a single line isn't required and is done for clarity and convenience, so you have to tell the toolkit when it has reached the end of a statement by adding a semi-colon. Along with general typos, forgetting the semi-colon is one of the most common mistakes made by beginners.

Uses and Examples Of Property Assignment

There are many uses of property assignment. Some, like the radius example above, have a direct effect on your mediascape, others will change the behaviour of an action when it is performed. The following table contains some examples of property assignment with an explanation of their effects.

| Code | Type | Result |
|---------------------------|------------|---|
| myAudio.Loop = true; | True-False | the Audio file 'myAudio' will loop when played |
| myAudio.Volume = 50; | Number | sets the volume for 'myAudio' to 50 (half the maximum of 100) |
| myAudio.FadeDuration = 5; | Number | determines the length of a fade when a fade-in or fade-out action is carried out on 'myAudio' |

| | | |
|--|------------|---|
| <code>myCircleRegion.X = 358995.0;</code> | Number | sets the X position of 'myCircleRegion' - i.e. moves the region to a different location. |
| <code>myRegionGroup.Enabled = false;</code> | True-False | 'myRegionGroup' is a map group containing regions. Setting Enabled to false means none of the region events in the group can be triggered. |
| <code>myRegionGroup.Enabled = true;</code> | True-False | reverses the previous operation, meaning region events within the group can now be triggered. |
| <code>Logger.Filename = "myfile.txt";</code> | Text | This sets the file name of the log to 'myfile.txt'. |

With so many objects and associated properties it's obviously not practical to list every possibility. Much of the challenge of programming is working out which property to change to achieve the desired effect. It's also worth pointing out that in some cases it might be simpler to set a property directly in the property panel. For example if you always want a particular audio to loop you can simply set the Loop property to 'True' using the drop-down. However, when you want a property to change at a particular event you will need to do this with code. For example you might reduce the volume of a looping background Audio track when another Audio track is triggered when the user enters a region.

Setting Variables

Setting a variable value isn't any different from setting a property. In fact all you have to remember is that to set a variable you need to change the variable's 'Value' property and, as before, match the type of the variable. The following table gives some examples of variable assignment:

| Code | Type | Notes |
|--|------------|--|
| <code>myTextVariable.Value = "Hello world";</code> | Text | straightforward text assignment |
| <code>myTextVariable.Value = "true";</code> | Text | this is not the same as 'true' when assigned to a True/False variable |
| <code>myTextVariable.Value = "123";</code> | Text | this is not the same as assigning 123 to a number variable: it will now be treated as text |
| <code>myNumberVariable.Value = 123;</code> | Number | in this case we're storing a number and can perform number operations (see below) |
| <code>myNumberVariable.Value = 3.14159265;</code> | Number | number variables can store decimal numbers |
| <code>myNumberVariable.Value = -24.5;</code> | Number | number variables can store negative numbers |
| <code>myBoolean.Value = true;</code> | True/False | note the absence of inverted commas around <i>true</i> . 'true' is a special word recognised by the mscape toolkit |
| <code>myBoolean.Value = false</code> | True/False | again note that there are no inverted commas around <i>false</i> |

Assigning a Property To A Variable

Setting variables to arbitrary values has its uses, but they also serve another important function. You don't always have access to event parameters outside of the event, and you may need to store the

state of a property at a particular point - this is when you would store the value in a variable. A couple of examples might help here:

1. Because they are event parameters, and have no associated properties*, the x and y values accessible from the map 'OnLocationChange' event cannot be accessed from outside that event. If you wanted to check the user's current location on a button press - for example in order to calculate the distance between them and a particular region - it wouldn't be possible.

* note that in the latest Beta release 'Northing' and 'Easting' properties are now available.

2. You want to record how many times a user entered a region before they solve a puzzle. The region EnteredCount property will continue to increase each time they enter the region after the puzzle is solved, so you need to store it at the point they succeed.

As with all assignments so far this isn't difficult to achieve, so long as you remember to match the variable type to the property or parameter type. It's likely that most of the properties you will want to store are numbers and the occasional booleans. Setting the value of a variable to match that of a property or event parameter simply requires you to *assign* the property to the variable:

```
myVariable.Value = ObjectName.PropertyName;
myVariable.Value = ParameterName;
```

So to record the 'EnteredCount' of a region in a Number variable called 'region01EnteredCount' you would use the following code:

```
region01EnteredCount.Value = Region01.EnteredCount;
To store the x and y parameters of the map OnLocationChange in Number variables 'currentX' and
currentY' you would add the following to the OnLocationChange event:
```

```
currentX.Value = x;
currentY.Value = y;
```

Now each time the OnLocationChange event is triggered (usually around once per second) the value stored in the variables will be updated. Because State variables are accessible from any event within your mediascape you can access the current map coordinates at any point using the currentX and currentY variables.

More Complex Variable Assignments

The examples of property and variable assignment in the previous section had one notable limitation: they didn't allow you to manipulate the existing value of a property or variable. What if you want to decrease an audio track volume by ten each time a region is entered? What if you didn't know the coordinates you wanted to move a region to, only by how far you wanted to shift it? What if you wanted to change the filename of your log to ensure previous log files weren't overwritten? As it happens you've already seen how to achieve this.

Operators

When you logged the x coordinate in the previous section you used the + operator to display the value of x alongside a text label:

```
Logger.Log("x coordinate: " + x);
```

This approach can also be used with variables and properties. We'll start by duplicating the result of the above code using a text variable 'myText'. In the GPS OnLocation event type the following two lines:

```
myText.Value = "x coordinate: " + x;
Logger.Log(myText.Value);
```

When you run the tester and click on the map you'll get the same result as before - a log message with a label and the value of the current x coordinate. The important thing to realise here is that we have combined the label and coordinate into a single variable: 'myText' and the variable is updated each time the event is triggered.

Looking at this example you might see that we've actually broken the rule about always assigning properties or parameters of the same type. It's time to explain why this works. One important feature of Text variables is that they can be forgiving when you try and assign non-text values to them using an operator. As long as one part of the operation **is** text, the other part will automatically be converted to text. So in this example the value of x is converted from a number to text. Since all we're doing with the number is displaying it in the log that's not too much of a problem.

Operators and Numbers

When working with numbers the effect of operators changes and the number of operators available increases substantially. In this case the good old plus sign + has the expected result. In the Mediascape OnLoaded event type the following:

```
myNumber.Value = 6 + 2;
Logger.Log("result: " + myNumber.Value);
```

In this case what gets stored in the number variable and displayed by the Logger is the result of the sum of 6+2. i.e. 8. It shouldn't come as too much of a surprise that all the basic arithmetic operators are available, and have the expected result, as shown in the following example:

```
myNumber.Value = 6 + 2;
Logger.Log("6+2: " + myNumber.Value);
myNumber.Value = 6 - 2;
Logger.Log("6-2: " + myNumber.Value);
myNumber.Value = 6 * 2;
Logger.Log("6*2: " + myNumber.Value);
myNumber.Value = 6 / 2;
Logger.Log("6/2: " + myNumber.Value);
```

Note that each assignment sets a new value for 'myNumber' ignoring the previous value. Effectively we're using the logger as a calculator; and we're not limited to calculations with just two numbers: we can add, multiply, subtract or divide as many numbers as we like. Try the following:

```
myNumber.Value = 6+2*4-2/2;
Logger.Log("result: " + myNumber.Value);
```

What do you get as the result in the tester? Now try it on a calculator. Do you get the same result? The answer is almost certainly 'no'. This example isn't designed to put you off calculations with multiple values, it just highlights a potential problem with these types of calculations: the result will depend on the order in which the operations are carried out. A calculator takes the result of an operation and applies the next operation to the result; so:

```
6+2 = 8
8*4 = 32
32-2 = 30
30/2 = 15
```

Programming languages deal with these calculations a little differently. There is a set order of precedence for operations, with multiplication and division coming first, then adding and subtraction. So:

```
2*4 = 8
2/2 = 1
6+8-1 = 13
```

Another way of writing this is:

```
6 + (2*4) - (2/2) = 13
```

Brackets (or 'parentheses') can be used when you want the next operation to be carried out on the result of the calculation inside the brackets. So to get the same result as a calculator in the tester you would need to do the following:

```
myNumber.Value = (((6+2)*4)-2)/2;
Logger.Log("result: " + myNumber.Value);
```

Basically when in doubt about operator precedence, or if you're getting an unexpected result, it's a good idea to use parentheses. Just remember that for every opening bracket you'll need a closing bracket.

Operators and Number variables

In practice you're unlikely to be carrying out calculations with arbitrary numbers, you'll want to work with existing variables, properties and parameters. This is simply a matter of referencing a number variable, property or parameter in your calculation. For example:

```
// using the x parameter in the OnLocationChange event:
myNumber.Value = x + 10;
// using the radius of a CircleRegion:
myNumber = myCircleRegion.Radius - 10;
// using another number variable:
myNumber = playerScore.Value / 10;
// using two existing number variables:
```

```
myNumber = playerScore.Value * playerLevel.Value;
```

In the last example above you'll see that a calculation can be made up solely of number variables, properties or in fact parameters.

Division by zero

It's worth adding a warning at this point about one particular calculation you should avoid at all costs. Try adding the following to the Mediascape OnLoaded event:

```
myNumber.Value = 1/0;
Logger.Log("one divided by zero = " + myNumber.Value);
```

You should see the following error message:

Whilst some mathematicians may be happy with the idea of dividing by zero and giving the answer as 'infinity', computers don't like it one bit; to the point that programming languages will usually report division by zero as an error. So, if you ever need to divide a number by the value of a variable, **you should ensure that the variable is not zero or your mscape will crash.**

Recursion

Finally let's answer the questions we posed at the beginning. How do we manipulate the existing value of a variable, property or parameter so that we can, for example decrease the volume of an audio clip, or change the position of a region? Look at the following code:

```
Logger.Log("value before: " + myNumber.Value);
myNumber.Value = myNumber.Value + 1;
Logger.Log("value after: " + myNumber.Value);
```

At first glance this doesn't look like it should work. How can something be equal to itself plus one? What you have to remember is that the equals sign is an *assignment operator* and does **not** represent equality. We're not saying that myNumber.Value is equal to myNumber.Value plus one; we're saying we want to **set** its value to be equal to itself plus one. This little trick is known as recursion and is incredibly useful. To test it let's use a button input. Select the OnCenter event of the Buttons object and add the following code:

```
myNumber.Value = myNumber.Value + 1;
Logger.Log("result: " + myNumber.Value);
```

Run the tester and click the center button in the Buttons panel on the right. Keep clicking and see what happens. Each time you click it the value of myNumber is increased by one. Now try replacing the operation with each of the following, testing each one in turn:

```
myNumber.Value = myNumber.Value - 1;
myNumber.Value = myNumber.Value * 2;
myNumber.Value = myNumber.Value / 2;
myNumber.Value = myNumber.Value + myNumber.Value;
```

You'll see that each time you click the button the operation is applied to the current value of myNumber. These recursive operations are important enough to have their own assignment operators. Instead of an equals sign and repeating the variable reference you use the type of

operation (plus, minus, multiplication, division) followed by an equals sign, followed by the value you want to apply. So:

```
myNumber.Value = myNumber.Value + 1;  
// can be written as:  
myNumber.Value += 1;
```

```
myNumber.Value = myNumber.Value - 1;  
// can be written as:  
myNumber.Value -= 1;
```

```
myNumber.Value = myNumber.Value * 2;  
// can be written as:  
myNumber.Value *= 2;
```

```
myNumber.Value = myNumber.Value / 2;  
// can be written as:  
myNumber.Value /= 2;
```

```
myNumber.Value = myNumber.Value + myNumber.Value;  
// can be written as:  
myNumber.Value += myNumber.Value;  
// Note that this is equivalent to:  
myNumber.Value *= 2;
```

Adding and subtracting 1 to the value of a number is used so often in programming that there are separate shortcuts for this also:

```
// To add one to the value:  
myNumber.Value ++;  
// To subtract one from the value:  
myNumber.Value --;
```

This technique is incredibly powerful and has a range of applications. In your mspace you might use it to add to a player's score, or even to move the position of a region. For example try adding the following to the Button OnRight event:

```
region01.X +=10;
```

Mediascape Scripting Language: An Introduction

What is an event script?

An event script is a little piece of programming that is associated with an event and that is followed when that event happens. When an event happens the script is triggered and reacts to that particular event.

Think of it as being like hospital staff. When an event happens like 'baby born' that event triggers a number of people into action who are specially skilled to deal with that event.

In a mediascape the events are not quite as dramatic and are dictated by the nature of mediascape, so that events are such things as 'the user has entered a region', 'the user has pressed button A on the device while in region 23', etc.

If you are familiar with Flash...

Some people involved in multimedia may already have familiarity with a number of different packages and systems. If you have some familiarity with Flash and ActionScript (the programming language used in Flash) then this could be useful in terms of understanding the behaviour of event scripts in the mscape maker.

In Flash, much can be achieved through the act of dragging and dropping assets onto the stage and coupling various behaviours to them. However once you move into the world of ActionScript many other things become possible, it becomes easier to do things that took time with the drag-and-drop interaction and it becomes possible to do things that you couldn't do before.

The situation is similar with the mscape maker, you can do quite a bit with the drag-and-drop interface but you can do more with the scripting abilities.

Writing event scripts is programming

The way that you write event scripts in the mscape maker is based on parts of a programming language called C# (pronounced 'sea sharp'). If you are a programmer and you have used C# you will see things that you are familiar with. If you have programming experience in languages like C++, JavaScript or Flash ActionScript then some aspects of it will be familiar.

However, if you are not a programmer then do not despair, you can pick it up fairly quickly and you can achieve quite advanced things without having to do lots of programming.

Things you can do with event scripts

Event scripts are a useful way of doing more complex things with what you have got. You can make your mediascape do things like this:

Link two sounds to a region, play one sound the first time someone enters a region and then only play the second sound whenever they re-enter that region.

Have a game where there is a region containing sound and images of a lion, and the user can wander around and then select a point in the space they are in by clicking a button on the screen to say 'put the lion region at the point where I am standing'.

Play a sound that asks the user a questions and they can answer by clicking on 'yes' or 'no' buttons on the screen. The mediascape can then adjust itself depending on their answer.

Have a go of some of the mediascapes available on the mscape website to experience more of the possibilities.

The basics of event scripts

Event scripts are about doing things in response to events. So the basic model is that you can say 'when this particular event occurs I want you to do this action'. Simple enough, but what does that look like when it is translated in to the language of event scripts. Let's build up the structure bit by bit.

I want to say 'if this particular event has happened then do this thing and do this thing as well'. To express this 'if-then' in script I would write the following:

```
if (this particular event has happened)
{
    Do this thing;
    Do this thing as well;
}
```

- NOTE
- The list of things to do is enclosed with 'pointy brackets' (or 'braces' or whatever you want to call them).
- Each thing to be done instruction is ended with a semi-colon.
- There is **NO** semi colon after the 'if (this particular event has happened)'. You can think of the semi-colon as behaving like a full-stop in writing.
- The condition of the 'if' part is enclosed in brackets.

If we wanted to play a fanfare of trumpets the fourth time the user entered a region around an old ruined castle then we would want a script that looked like this:

```
if (it's the fourth time that the user has entered the region be the casle gate)
{
    Play the trumpet fanfare;
}
```

Of course this combination of script and chatty sentences won't work. To make it work we actually would write:

```
if (castle_gate.EnteredCount == 4)
{
    fanfare.Play();
}
```



```
fanfare.Volume = 100;
```

Where 'castle_gate' is the region we have defined in the right place on the map and 'fanfare' is the audio file of trumpets that we have imported into the mediascape.

- NOTE
- When we set something we use one equals sign, so we use '=' when setting the volume. (In effect 'make this equal to').
- When we are doing an 'if' and want to test if two things are equal we use two equals signs; '=='. (In effect 'is this equal to').

The underlying model in scripts in the mscape maker is that you create things and these things have actions associated with them. You can call up these actions by addressing the object and telling it the action you want it to do.

For a good comparison imagine being in the house of the future where everything is intelligent and can react to your voice. You could not just shout 'switch on' when you came into a room, you would have to say what you wanted to switch on 'Lights; switch on!' or 'TV; switch on!'. Other things would have other actions that you could call upon; 'Door; open' or 'Curtains; close'. Certain objects would only have certain actions, so you couldn't say 'Curtains; switch on' for example. And imagine if things were so advanced that CDs could play themselves, so if you wanted to listen to the Scissor Sisters latest you wouldn't talk to the CD player you'd just say 'ScissorSisters; play!'

Now you can see similar things in the portion of script above. We have two things; we have the region around the castle; 'castle_gate' and the audio object 'fanfare' which is a recording of a trumpet fanfare. When we want to find out about the number of times the user has entered the region we ask for that property of the region; 'castle_gate.EnteredCount'. When we want to do things to the audio we address it directly by name and then tell it what we want of it, so 'fanfare.Volume = 100' sets its volume to 100 and; 'fanfare.Play()' makes it play.

Play has brackets and Volume doesn't because Volume is a property of the fanfare whereas Play is an action that fanfare can carry out. Properties are like nouns and actions are like verbs.

Finding out more

Help pages

In the help pages on the mscape maker and the help pages on the website there is more technical documentation about the possibilities of the different script objects. For example you could find out more about all the things you can do with audio objects; you can do far more than just play them.

Example mediascapes

If you have downloaded a mediascape from the mscape website you can run it on your mobile device, alternatively you can open it up in the mscape maker and examine how the designer has put it together. One of the things you can look at is the event script window in the bottom right corner of the window. This shows the event (written in the tab) and the script that is followed when the event happens. You can click on different objects in the mediascape to look at the script associated with

them. As well as the objects on the map in the main part of the screen all the things in the left hand bar are objects and they too can have scripts. Even the top level object of the mediascape has two events that can have script in them.

Introduction to the Mscape Scripting Language

What Is An Event Script?

An event script is a little piece of programming that is associated with an event and that is followed when that event happens. When an event happens the script is triggered and reacts to that particular event.

Think of it as being like hospital staff. When an event happens like 'baby born' that event triggers a number of people into action who are specially skilled to deal with that event.

In a mediascape the events are not quite as dramatic and are dictated by the nature of mediascape, so that events are such things as 'the user has entered a region', 'the user has pressed button A on the device while in region 23', etc.

If You Are Familiar With Flash...

Some people involved in multimedia may already have familiarity with a number of different packages and systems. If you have some familiarity with Flash and ActionScript (the programming language used in Flash) then this could be useful in terms of understanding the behaviour of event scripts in the mscape maker.

In Flash, much can be achieved through the act of dragging and dropping assets onto the stage and coupling various behaviours to them. However once you move into the world of ActionScript many other things become possible, it becomes easier to do things that took time with the drag-and-drop interaction and it becomes possible to do things that you couldn't do before.

The situation is similar with the mscape maker, you can do quite a bit with the drag-and-drop interface but you can do more with the scripting abilities.

Writing Event Scripts Is Programming

The way that you write event scripts in the mscape maker is based on parts of a programming language called C# (pronounced 'sea sharp'). If you are a programmer and you have used C# you will see things that you are familiar with. If you have programming experience in languages like C++, JavaScript or Flash ActionScript then some aspects of it will be familiar.

However, if you are not a programmer then do not despair, you can pick it up fairly quickly and you can achieve quite advanced things without having to do lots of programming.

Things You Can Do With Event Scripts

Event scripts are a useful way of doing more complex things with what you have got. You can make your mediascape do things like this:

Link two sounds to a region, play one sound the first time someone enters a region and then only play the second sound whenever they re-enter that region.

Have a game where there is a region containing sound and images of a lion, and the user can wander around and then select a point in the space they are in by clicking a button on the screen to say 'put the lion region at the point where I am standing'.

Play a sound that asks the user a questions and they can answer by clicking on 'yes' or 'no' buttons on the screen. The mediascape can then adjust itself depending on their answer.

Have a go of some of the mediascapes available on the mscape website to experience more of the possibilities.

The Basics Of Event Scripts

Event scripts are about doing things in response to events. So the basic model is that you can say 'when this particular event occurs I want you to do this action'. Simple enough, but what does that look like when it is translated in to the language of event scripts. Let's build up the structure bit by bit.

I want to say 'if this particular event has happened then do this thing and do this thing as well'. To express this 'if-then' in script I would write the following:

```
if (this particular event has happened)
{
  Do this thing;
  Do this thing as well;
}
```

- NOTE
- The list of things to do is enclosed with 'pointy brackets' (or 'braces' or whatever you want to call them).
- Each thing to be done instruction is ended with a semi-colon.
- There is **NO** semi colon after the 'if (this particular event has happened)'. You can think of the semi-colon as behaving like a full-stop in writing.
- The condition of the 'if' part is enclosed in brackets.

If we wanted to play a fanfare of trumpets the fourth time the user entered a region around an old ruined castle then we would want a script that looked like this:

```
if (it's the fourth time that the user has entered the region be the
casle gate)
{
  Play the trumpet fanfare;
}
```

Of course this combination of script and chatty sentences won't work. To make it work we actually would write:

```
if (castle_gate.EnteredCount == 4)
{
  fanfare.Play();
  fanfare.Volume = 100;
}
```

Where 'castle_gate' is the region we have defined in the right place on the map and 'fanfare' is the audio file of trumpets that we have imported into the mediascape.

- NOTE
- When we set something we use one equals sign, so we use '=' when setting the volume. (In effect 'make this equal to').
- When we are doing an 'if' and want to test if two things are equal we use two equals signs; '=='. (In effect 'is this equal to').

The underlying model in scripts in the mscapemaker is that you create things and these things have actions associated with them. You can call up these actions by addressing the object and telling it the action you want it to do.

For a good comparison imagine being in the house of the future where everything is intelligent and can react to your voice. You could not just shout 'switch on' when you came into a room, you would have to say what you wanted to switch on 'Lights; switch on!' or 'TV; switch on!'.

Other things would have other actions that you could call upon; 'Door; open' or 'Curtains; close'. Certain objects would only have certain actions, so you couldn't say 'Curtains; switch on' for example. And imagine if things were so advanced that CDs could play themselves, so if you wanted to listen to the Scissor Sisters latest you wouldn't talk to the CD player you'd just say 'ScissorSisters; play!'

Now you can see similar things in the portion of script above. We have two things; we have the region around the castle; 'castle_gate' and the audio object 'fanfare' which is a recording of a trumpet fanfare. When we want to find out about the number of times the user has entered the region we ask for that property of the region; 'castle_gate.EnteredCount'. When we want to do things to the audio we address it directly by name and then tell it what we want of it, so 'fanfare.Volume = 100' sets its volume to 100 and; 'fanfare.Play()' makes it play.

Play has brackets and Volume doesn't because Volume is a property of the fanfare whereas Play is an action that fanfare can carry out. Properties are like nouns and actions are like verbs.

Finding Out More

Help pages

In the help pages on the mscapemaker and the help pages on the website there is more technical documentation about the possibilities of the different script objects. For example you could find out more about all the things you can do with audio objects; you can do far more than just play them.

Example Mediascapes

If you have downloaded a mediascape from the mscapemaker website you can run it on your mobile device, alternatively you can open it up in the mscapemaker and examine how the designer has put it together. One of the things you can look at is the event script window in the bottom right corner of the window. This shows the event (written in the tab) and the script that is followed when the event happens. You can click on different objects in the mediascape to look at the script associated with them. As well as the objects on the map in the main part of the screen all the things in the left hand bar are objects and they too can have scripts. Even the top level object of the mediascape has two events that can have script in them.

Using Random Numbers

Sometimes you may want to add some uncertainty to your mediascape - for example when a region is entered you may want to play a random audio from a list.

Generating a random number is easy..

```
// Used for generating random numbers
Random rnd = new Random();

//Pick a random number between 0 and 4 (not including 4)
int randNum = rnd.Next(0,4);
```

There are many ways you can use this random number - if you are an experienced programmer you can probably think of dozens of ways. I'll outline a couple of handy ones here.

Showing a Random Image from a Folder

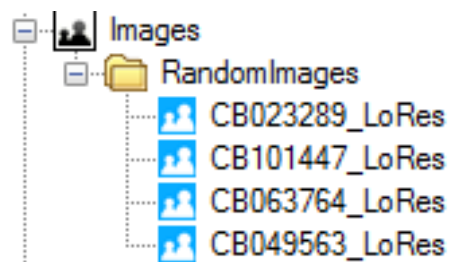


Image showing a folder with a series of images in it

You can use a random number to show a random image from a folder. To try this out, add an Folder inside the Images object called RandomImages and add a series of images to it (any number is fine).

Type the following code into the Buttons object's OnUp event (you can omit the parts after the //)

```
// create the random number generator
Random random = new Random();
// make a random number between 0 and the number of images in the folder.
int index = random.Next(0, RandomImages.Items.Count);
// get out the image at that position. Note that ALL objects with children have a hidden property called
// Items containing the child objects - in this case we're getting the index'th image inside the RandomImages property.
Image r = (Image)RandomImages.Items[index];

// show the image
r.Show();
```

You can use this technique for any media type - just replace the word Image on line 3 with Audio or WebPage or whatever.

Using Functions

Functions are useful when you find yourself copying and pasting identical pieces of event code into multiple regions (or other resources). This works fine until you realise you've made a mistake in that code, because then you need to go through each and every copy you've made to fix it.

The *Functions* object in mscape allows you to write the functionality you need once, and then simply call it from each of the places in your mediascape where it is needed. This way if you want to change the way your piece of functionality works, perhaps to play a different sound, or to award more points, you need only make the change in a single place.

A Simple Example

Imagine you have a game which is about collecting coins, where each coin is represented by a region. You walk into the region, a 'coin' sound is played and your score is updated. Now imagine that there are 100 of these regions - copying and pasting the code for this each time you want to make a change would be a real chore.

Lets see what we can do with functions..

- Create a new mediascape
- Import a map
- Draw yourself a bunch of regions.
- Download this sound effect (right-click, Save As), and then import it into mscape maker - it should appear in your maker as 'sndCoin'.
- Add a variable to hold the score. Right-click 'State' and choose 'Add Number' - call the new number 'score'.
- Right-click on Tools and Select 'Add Functions'
- Right-click on the newly-added Functions object and select 'Add Function'
- Call the new function 'fnCoin'.
- Select fnCoin and type the following into the 'Function' event - this is the event that will be triggered when the function is run.

```
sndCoin.Play();
score.Value++;
Logger.Log("Your score is now " + score.Value);
```

- Now we need to call the function from each of your regions.
- Select the first region and type..

```
fnCoin.Run();
```

- Right-click that region in the left-hand panel and choose 'Copy Behaviour'.
- Right-click each other region in turn and select 'Paste Behaviour'

Run this in the tester to try it out.

You might be thinking *ok, that's fine, but I still had to copy and paste a behaviour to each region?* That's true but what if you decide later on that 10 points should be awarded for each coin pickup?

Previously you'd have had to go through every single region and change the code - or change it once and repeat the copy-paste process we did before. When using functions we can select `fnCoin` and change the code once to read..

```
sndCoin.Play();
score.Value += 10; // <-- change here..
Logger.Log("Your score is now " + score.Value);
```

Handy huh?

Functions and Parameters

There are two ways of running a function - `fnCoin.Run()` and `fnCoin.RunWithParams(..)`.

In the previous example we used `.Run()` - and the reason for this is that every single region behaved in exactly the same way. However, if we wanted to make some coins worth more than others we'd need to use the `.RunWithParams()`.

`.RunWithParams()` runs the function but additionally passes in a list of optional *parameters*. Parameters are simply a way of passing a piece of data from the place where the function is called (e.g. the region) into the function's code (e.g. where the points are awarded)

Here's what we'd do for the coins example..

In region 1 (small money)..

```
fnCoin.RunWithParams(5);
```

In region 2 (big money)..

```
fnCoin.RunWithParams(100);
```

In `fnCoin`..


```
sndCoin.Play();
```

```
// Take the first parameter, and convert to a Double (a number).
// Remember that computers start counting at 0 - so parameter 0 is the first one, 1 is the second etc.
score.Value += Parameters.GetNumberAt(0);
Logger.Log("Your score is now " + score.Value);
```

Multiple Parameters

You can actually pass in any number of parameters into the `.RunWithParams` call - each parameter should be divided by a comma, e.g. `RunWithParams (10, "a", "b");`

You'd then retrieve these in the function itself using `GetNumberAt (0)`, `GetStringAt (1)`, `GetStringAt (2)`. You need to match the `.GetXAt` call to the type of the parameter - e.g. parameter 0 is a number, so we use `.GetNumberAt`, while parameter 1 and 2 are Strings so we use `.GetStringAt`. If you get this wrong you'll see an error in the output window in mscapetester like 'The parameter at 2 is not a string, it's a Double' (Double is what the scripting language calls a number). Parameters will return 0 or an empty string if something goes wrong like this.

 Make sure that every time you call `MyFunction` you pass in the same number of parameters, and in the same order. If not you'll find that your function doesn't behave as you expect.

Passing in Mediascape Objects and Media Items

It's actually possible to pass anything in as a parameter - for example you can pass in a media object such as an Audio. This is handy if your regions all have identical behaviour *except* they play different audios.

E.g. *In region A's OnEnter event*

```
MyRegionFunction.RunWithParams(audio1)
```

In region B's OnEnter event

```
MyRegionFunction.RunWithParams(audio2)
```

MyRegionFunction Definition

```
// Do things that happen on every region entry..
```

```
Audio a = (Audio)Parameters.GetObjectAt(0);
a.Play();
```

The fiddly part here is that you need to *cast* the object that's returned from the `GetObjectAt` method into the type of object you require. This is done by pre-pending the name of the type of object in round brackets to the `GetObjectAt` call, e.g. `(Audio)Parameters.GetObjectAt(x)` and assigning it to a new local variable (`Audio a = ...`)

This technique works for any object type in mscapetester, e.g. `Image`, `Video`, `CircleRegion`, `PolygonRegion`, simply replace the word `Audio` with that of the type you wish to use.

One thing to note is that the popup list of properties and actions available on an object does not show when you type the `.` using this method, so it's best to check the names of the actions or properties you want to run by selecting an object of that type (e.g. click on an audio) and looking in the properties, or type the name an object and hit `'`

Here's an example where the radius of a `CircleRegion` is used to assign points scored by a user - bigger regions score more points. You'll need to add a `Number` variable called 'score' for this.

E.g. *In region A's OnEnter event*

```
MyRegionFunction.RunWithParams(me) // me is the region that was entered
```

MyRegionFunction Definition

```
CircleRegion r = (CircleRegion)Parameters.GetObjectAt(0);  
score.Value += (int)r.Radius / 10 + 1; // round off the radius value to an integer, and divide by 10  
Logger.Log("Score is " + score.Value);
```

Using the Maths Library

Mscape has a series of built-in functions for those times when you really need to do a little bit of trigonometry - for example you might want to find the distance between two points, or the angle between two lines.

I'm not going to teach you the maths for doing these on this page (Google for the Pythagoras theorem or basic trigonometry) - but instead I'll list the mathematical functions you can access as part of your mscape script.

Here's a simple example of using one of these functions..

```
Logger.Log (" The square root of 9 is " + Math.Sqrt(9) );
```

Note that all trigonometrical functions use angles in radians, not degrees.

| | |
|------------------|---|
| Math.Abs(n) | Returns the absolute value of a specified number |
| Math.Acos(n) | Returns the angle whose cosine is the specified number |
| Math.Asin(n) | Returns the angle whose sine is the specified number |
| Math.Atan(n) | Returns the angle whose tangent is the specified number |
| Math.Atan2(x, y) | Returns the angle whose tangent is the quotient of two specified numbers. <i>This function will give you the angle of a line from 0,0 to x,y in radians, measured from the x-axis</i> |
| Math.Cos(n) | Returns the cosine of the specified angle |
| Math.Cosh(n) | Returns the hyperbolic cosine of the specified angle |
| Math.Cosh(n) | Returns e raised to the specified power |
| Math.Floor(n) | Returns the largest integer less than or equal to the specified number |
| Math.Log(n) | Returns the logarithm of a specified number |
| Math.Log10(n) | Returns the base 10 logarithm of a specified number |
| Math.Max(n, m) | Returns the larger of two specified numbers |
| Math.Min(n, m) | Returns the smaller of two numbers |
| Math.Pow(x, y) | Returns a specified number raised to the specified power |
| Math.Round(n) | Rounds a value to the nearest integer or specified number of decimal places |
| Math.Sign(n) | Returns a value indicating the sign of a number (-1 negative, 1 positive) |
| Math.Sin(n) | Returns the sine of the specified angle |
| Math.Sinh(n) | Returns the hyperbolic sine of the specified angle |
| Math.Sqrt(n) | Returns the square root of a specified number |
| Math.Tan(n) | Returns the tangent of the specified angle |
| Math.Tanh(n) | Returns the hyperbolic tangent of the specified angle |

Using the Logger in Code

In the Mscape Tester you can write to the Output window at any point using the following code:

```
Logger.Log("your message");
```

Where anything between the brackets will be displayed in the output window.

One very useful application of this is to show the current value of State variables (Text, Numbers etc.) when a particular event is triggered, which can really help when debugging an mscape. This is not quite as simple as you might expect however, as simply trying to directly log some State variables will cause an error (in programming speak: the Logger will only accept a string). The most reliable approach is to combine the State variable with some text; and is also recommended as you then have a label next to the value displayed in the log.

So if you wanted to log a State Number called *myNumber* you would do so as follows:

```
Logger.Log("myNumber: " + myNumber.Value);
```

We use `Logger.Log` as before, open the bracket and add some text, between "inverted commas", to act as a label - it makes sense to use the name of the State Number. Notice the space added before the closing inverted comma: without this there wouldn't be a space between the label and the number. The plus sign means we want to add something to the text, in this case the value of `myNumber` which we access as normal using `'myNumber.Value'`. We then close the bracket and end the line with a semi-colon. The value of `'myNumber'` will then be displayed in the Output window when the event is triggered.

Note that normally anything displayed with `Logger.Log` is **not** saved in the Logger file when the mediascape is run on a device (or in the Tester). Assuming you have configured this to store data (for example a [trace](#)) you can write to the logger programmatically using `Logger.WriteHeader` and `Logger.WriteFooter` (or in Mscape 2.5 Beta using `Logger.LogToFile`).

It's also possible to force the Mscape Player to save a log at a particular point, simply by setting a new unique file name. There are several ways of achieving this, for example by using a Number variable as a counter, appending it to the file name and incrementing it at each save; or by appending the current date and time. So, for example, to set a file name using the date and time you would use the following code:

```
Logger.FileName = "myFileName_" + DateTime.Now.ToString("dd-MM-yy_HH-mm");
```

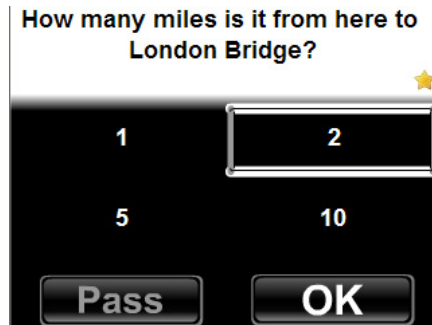
Where *myFileName_* is some arbitrary text and *DateTime.Now...* generates the current date followed by the time on the device. This code could be added to any Event, for example a button press or when the user leaves a particular region.

Terms to Avoid

| | | | |
|----------|-----------|------------|-----------|
| abstract | event | new | struct |
| as | explicit | null | switch |
| base | extern | object | this |
| bool | false | operator | throw |
| break | finally | out | true |
| byte | fixed | override | try |
| case | float | params | typeof |
| catch | for | private | uint |
| char | foreach | protected | ulong |
| checked | goto | public | unchecked |
| class | if | readonly | unsafe |
| const | implicit | ref | ushort |
| continue | in | return | using |
| decimal | int | sbyte | virtual |
| default | interface | sealed | volatile |
| delegate | internal | short | void |
| do | is | sizeof | while |
| double | lock | stackalloc | |
| else | long | static | |
| enum | namespace | string | |

Creating a Located Quiz Game

Located quiz games are mediascapes where the user is asked multiple-choice questions at various points around the real world. The answers to these questions can usually be found around the area where the question pops up.



You may have this question pop up near to a road sign that contains the correct answer. You can have a lot of fun coming up with interesting questions for your players, for example you may wish to have some questions that can only be answered by asking local people, or that require some maths or other mental challenges.

These kinds of games can work really well as a social game for a mediascape event as teams can share a single device, and send members off in various directions to find the answer. The mscapers team have run successful games similar to this in London for the Girl Geek Mediascape event, and for the Quality Me Time Event in Les Arcs, France. Both of these mediascapes are available to download from the mscapers.com site.

In order to make it easy for you to create your own located quiz game, I have torn the guts out of the Les Arcs game and generalized it.

This version will no longer automatically switch from the status screen to the map screen after 15 seconds. This was causing issues for people who were mixing videos, images, etc with the Flash QuizGame interface.

How it works

The interface is all handled by a Flash movie, which loads the list of questions from an external text file. The format of this text file is explained below. In the mediascape, you create regions where you want each question to pop up, you match the name of the region with the name of the question, and you send a message to the Flash movie telling it to show the question that matches the name of the region.

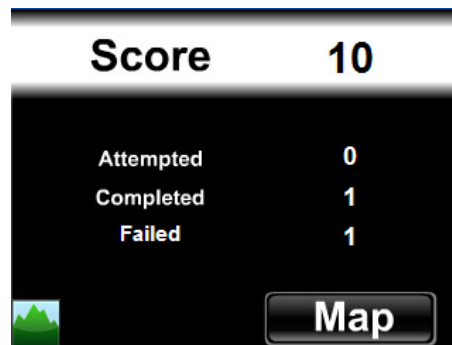
Each question is text only, and has four possible answers, only one of which is correct. When a region is entered, the appropriate question is shown, and the map is disabled. This means that while you are answering a question, no more questions will pop up until you have finished. The player can choose to answer the question, or pass. If they pass, they can re-try the question by walking back into that question region.

CREATING A LOCATED QUIZ GAME

The system will save the state of the game within the Flash movie, so you can reset the iPAQ, close mscscape player, and return to the game at a later date if required. To use this feature, choose 'Continue' from the title screen. To start a new, blank game, choose 'New Game'.



You can test out all of your questions in order without needing a mediascape by clicking the small 'test' button at the bottom right of the title screen.



After each question, the user will be taken to a status screen which shows them their score, and the number of questions they've tried, answered correctly, and answered wrongly.

This screen will time out to the map, showing the player where they are on screen.



Template Mediascape

I've made an example mediascape that should be used as the basis for your quiz game. There's a bit of logic in there to set up the communication between the flash movie and the mediascape, so it's better to start from this than to try and rebuild it all from scratch.

You'll want to swap out the map that's included for one of your local area - make sure you leave the name of the map as the default 'map' or some of the logic won't work.

Note that it's not worth trying to play this template mediascape in the real world - there are only two questions, and the answers may not even be correct! If you're in the area, try out the girl geek game or the les arcs game (see above for links).

Writing your own Questions

First up is to walk your area and come up with a series of questions. Personally I've found the best way to do this is to walk around with a PDA with a voice recorder, and to record the questions and the four possible answers in this as you go. Try and make sure that the questions are quite short, as the screen will only fit three lines of text. Each of the four answers should be pretty short too, you've got two lines worth of text to play with.

Once you have compiled your list of questions, you add them into the game by editing a text file. This file is called 'questions.txt' and is inside the 'WebPages' folder in the template mediascapes content directory. If you have the template mediascape open in mscape maker the quickest way to get to the file is to select Open Content Folder from the Tools menu, then navigate inside the 'WebPages' folder.

Open it in a text editor and it'll look something like this:

```
{
  "london":
  {
    "level":1,
    "question":"How many miles is it from here to London Bridge?",
    "answers":{"a":"1","b":"2","c":"5","d":"10"},
    "correct":"b"
  },
  "bridge":
  {
    "level":1,
    "question":"What is this bridge called?",
    "answers":{"a":"Waterloo","b":"Westminster","c":"Dave","d":"Cindy"},
    "correct":"a"
  }
};
```

I'll explain each part in turn. Note that it's VERY important that the layout of this file is preserved, don't go deleting brackets or double quotes or other formatting.

"london": - 'london' is the name of the question. Later you'll need to match the name of the region to this, so make sure you don't use any spaces or symbols, or start its name with a number.

"level":1 - this indicates the difficulty of the question, this example has difficulty level 1, where 1 is easy, 2 is medium, 3 is hard. The difficulty will be shown on screen as a series of stars, and also affects the score gained for getting the question right (easy = 10 points, medium = 20 points, hard = 30 points). It's up to you to decide which questions are given which difficulty levels.

"question": "How many.." - this is the text of the question itself. Try and keep it as concise as possible

"answers": {"a": "1", "b": "2" ... } This part gives the four possible answers (a,b,c,d). Again, keep these as short as you can, text and numbers are both acceptable - but don't forget the double quotes "".

"correct": "b" - This tells the system which is the correct answer. The example here gives 'b' as the right answer.

The bridge section defines a second question, it should be pretty self-evident what's going on here.

In order to add more questions, simply copy-and-paste one of the questions **before** the final }. Ensure that you place a comma , before the name of your new question. Things will go very wrong if you forget or mess up the formatting in some other way, so be careful!

Testing out your Questions

It's a good idea to test your questions out as you go along, to make sure it's all working and you haven't accidentally broken the formatting. To do this, start up the mediascape in mscapetester, simulate the GPS fix, and then hit the 'Test' button on the Flash interface. This mode will cycle through all of your questions in turn and doesn't require you to do any GPS simulation etc.

Triggering Questions from Regions

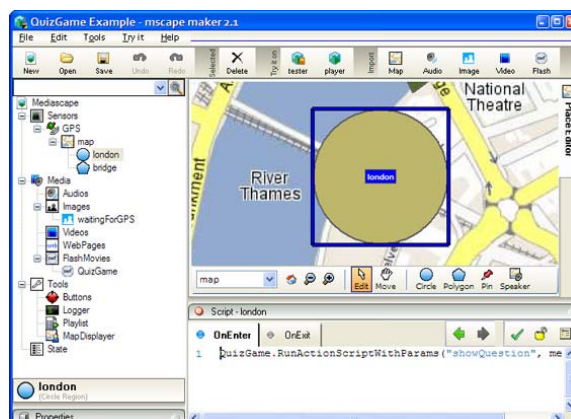
The final step is to make regions trigger the questions in the correct place. Luckily this is pretty straightforward.

Create yourself a region, polygon or circles are both fine, and give it a name which exactly matches the name of your question (e.g. london for the first example above).

Then you need to make the region tell Flash that it should show a question that matches the name of the region.

Copy-and-paste the line below into the 'OnEnter' event of the Region.

```
QuizGame.RunActionScriptWithParams("showQuestion", me.Name);
```



Note that this code is the same for all regions, regardless of what the name of the region is. The me.Name part gets out the name of the region that has been entered, and sends it off to the Flash movie.

Once you've done this, fire up mscapetester, simulate the "OnGotFix" event of the GPS by checking the 'Has Fix' box in the GPS gadget on the right and get playing! The system will only let you answer each question once, so make sure you choose 'New Game' from the title menu of the game.

Troubleshooting

If you get a screen where the question and all the answers appear as 'undefined' then you've either not matched the name of question to the name of the region (it is case sensitive) or there is a mistake in your questions.txt file such as an extra bracket or missing comma. Check it really carefully!

Advanced: Further Customization

The graphics and sound for the question game are all embedded in the Flash movie, so it is not possible to change them without rebuilding the 'QuizGame.swf' component and re-importing it into the mediascape.

If you'd like to change the graphics or sound then you should download the FLA and accompanying source code from the link below, and edit it using Flash CS3. Note that you'll need to be fairly confident with using Flash to do this without breaking the game.

[FLA and accompanying source code for the quiz game Flash component](#)

Important: You will need the Adobe Flash CS3 editor - older versions will not be able to open the FLA file

If you're interested in seeing how the game works, the bulk of the actionscript is in a set of external files.

| | |
|-----------------------|--|
| mediascape2.as | This file enables the communication between the Flash movie and the mediascape |
| mediascapeCommands.as | This file contains the actionscript functions that are directly invoked from the mediascape ("showQuestion" etc) |
| Debug.as | Code for running the question test mode |
| JSON.as | Library functions for parsing the question.txt file |
| QuestionEngine.as | Code for running the question game itself. This file has the bulk of the code |

How to Allow the User to Enter Text Such as Their Name

As part of your mediascape you may wish to have the player enter in text data, for instance if you want to refer to the player by name they'll need to type their name in. If you have tried to achieve this you will have had problems as mscap player blocks the standard Windows Mobile on-screen keyboard meaning that there is currently no built-in way of entering text.


This tutorial presents an alternative on-screen keyboard implementation that uses Adobe Flash, and allows you to download sample files so you should be able to get going pretty quickly.

This implementation was based on an [open source on-screen keyboard](#) . The original version was designed for kiosk systems using a large screen so it has been customized it to fit a standard Windows Mobile device in portrait mode (240x320).

I have created a demo Flash system that uses the new component for you, as well as a demo mediascape that uses it.



The on-screen keyboard

 This sample is designed for mediascapes that already have an Adobe Flash interface, and as such is probably most useful for those with Adobe Flash authoring experience.

Contents of the download

| | |
|------------------------|---|
| onscreenkeyboard2 fla | The FLA file that builds the onscreen keyboard component. You only need to open this if you want to customize the look-and-feel of the keyboard (see the OnScreenKeyboard MovieClip within this FLA) |
| onscreenkeyboard2 swf | The swf for the keyboard component. This must be included in your mediascape |
| UseInFlashDemoHost fla | A demo Flash app that hosts the keyboard. This is a simple flash movie that demonstrates how to pop up the keyboard at a particular point in the movie, and get the return value. Click the S shape to pop up the keyboard. |
| UseInFlashDemoHost swf | The swf for the hosting app demo. This is used in the demo mediascape. |
| KeyboardTester as | Source file for keyboard. You shouldn't need to change this. |
| StandardKey as | Source file for keyboard. This can be used to customize the text that appears on the space shift del ok buttons. |
| OnScreenKeyboard as | Source file for keyboard. You shouldn't need to change this. |

HOW TO ALLOW THE USER TO ENTER TEXT SUCH AS THEIR NAME

Inside the UseInFlashDemoHost flash file there are two important parts of the code..

When the S is clicked..

```
on(press)
{
    _global.TextEntry_Title = "Enter your name";
    mcTextEntry.loadMovie(_global.swfDirectory + "onscreenkeyboard2.swf");
}
```

mcTextEntry is a movieclip defined on the stage that is used to define the location of the keyboard (0,0). _global.TextEntry is used to set the title that appears at the top of the keyboard screen ('Enter your name').

When the user clicks OK, the function TextEntry _Complete is run on the main timeline (at _root) Here's the definition..

```
function TextEntry_Complete(text)
{
    unloadMovie(mcTextEntry); // get rid of the keyboard
    trace(text); // print the text entered out to the log
}
```

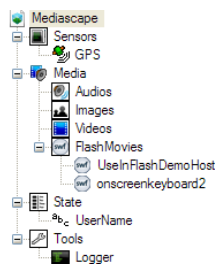
Example Mediascape

The example mediascape included in the package is called keyboardTest and simply plays the UseInFlashDemoHost flash movie. Note that the onscreenkeyboard2 swf file is also included - this is essential.

The way that the mediascape knows what text was entered is via the OnFSCCommand event. Important: It's actually the UseInFlashDemoHost swf that will fire the OnFSCCommand NOT the onscreenkeyboard2 swf.

Here's the contents of the OnFSCCommand event for UseInFlashDemoHost..

```
if (Command == "TextEntry")
{
    Logger.Log("User entered " + Parameters[0]);
    UserName.Value = Parameters[0]; // save the text that was typed into the variable called UserName.
}
```



contents of the example mediascape

If you run this on your mobile device, then clicking the S will open the on-screen keyboard. Type your message and click OK to hide the keyboard. The text you have types will be output to the log file. To view the log on mscape player

- Enable 'Designer Tools' from the main menu
- Click back to return to the mediascape
- Click the paintbrush icon
- Click View / Debug
- The lower panel shows the log, scroll it to the bottom to view the latest entry.

On mscape tester the log is visible at the bottom of the screen.

Using the on-screen keyboard for mediascapes without an existing Adobe Flash interface

The example mediascape demonstrates how to use the on-screen keyboard when you already have an existing Adobe Flash interface. If this is not the case, here's how to build a mediascape that uses the component..

- Add onscreenkeyboard2.swf to your project (right-click Media, select *Add Flash Movies*, then click *Flash* on the main toolbar).
- Add a Text variable called UserName to your project. (right-click State, select *Add Text*)
- At the point you want the keyboard to pop up, e.g. entering a region type:

```
onscreenkeyboard2.Play();
```

- To save the text they've typed in to the state variable, select onscreenkeyboard2, select the OnFSCommand event and type in:

```
if (Command == "TextEntry")
{
    Logger.Log("User entered " + Parameters[0]);
    UserName.Value = Parameters[0];
}
```

How to Create Video Files for Mscapex Using Adobe Flash CS3 Professional

- Create a new Flash Movie (File / New)
- Choose Flash File (Actionscript 2.0) from the New Document list.
- Set movie size to 320x240, and match the framerate to the framerate of your video (Modify / Document)
- Change publish settings to target Flash Player 7 (IMPORTANT!) - File / Publish Settings / Flash / Version.
- Go to File / Import Video
- Select Video 'on my computer' (I chose an AVI file)
- Click Next
- Select 'Embed Video in SWF and play in timeline'
- Leave the 'Embedding Page' as default settings (Embedded video, integrated, embed the entire video etc). Click Next
- I chose the Flash 7 - Low Quality setting (but Flash 7 - High Quality should also work)
- Click the 'Crop and Resize' tab, and check 'Resize Video' ensuring that you set it to 320x240.
- Hit Next, hit Finish
- Select File / Publish
- Go to mscapex maker, import the resulting swf

If you discover that the audio in the resulting video file goes out of sync, then make sure you have matched the frame rate of your Flash movie to that of the video you imported

In Flash, Looped Audios Don't Stop When the Movie is Stopped

If you have an looped audio attached to a frame in Flash, then the Stop function will stop the movie itself, but the audio will continue playing. This issue turns out that to be a problem within flash itself rather than anything mscap related. You can reproduce the problem by running a swf in Adobe's own Flash Player App (the one that runs when you double-click a swf file if you have Flash CS3 etc installed. Run the movie, and then uncheck 'Play' (essentially calling Stop) on the Control menu - the images will stop, but the looping audio continues.

Workaround

- You need to make the flash movie controllable from mscap
- follow the instructions in the help for [Controlling Flash from the Mediascape](#)
- Create yourself a frame on the main timeline which stops the looping audio, by clicking on the frame then in the properties box, set Sound to the sound to stop, and choose 'Stop' from the Sync dropdown.

Then in the first frame of the movie type this into the actions window:

```
function StopLooping()
{
    GotoAndStop(<number of your frame that stops the loop>);
}
```

Finally, in your mediascape run the function above (after you've called *MyFlashMovie.Stop*) by using

```
MyFlashMovie.RunActionScript("StopLooping");
```

Using the LoadMovie ActionScript Command

Large complex Flash movies can take a long time to load, so many designers split their projects into many smaller movies and load the parts on demand using the *loadMovie* [ActionScript](#) command. Splitting a large movie into parts also makes it easier to have several designers working at once, as each designer can work on a separate file.

For more information on the *loadMovie* command and its variants, see the [ActionScript](#) dictionary entry [here](#).

Mediascape-Specific Issues

When used in a mediascape, flash movies lose the ability to load other flash movies using the *loadMovie* command when used in conjunction with a relative path name.

E.g. a movie called *A.swf* in the folder `"/storage card/MyScape_Content"` cannot load a movie *B.swf* in the same folder using `loadMovieNum("B.swf", 1)`.

The way to get around this problem is to use the full path of the swf file on the device.

For example:

```
loadMovieNum("file:///storage card/MyScape_Content/B.swf", 1);
```

To unload the movie again, you should call

```
unloadMovieNum(1);
```

To prevent you needing to always know the full path of the movie on the device, you can use the following piece of code, kindly donated by Tom Milner on the mscape forums.

```
var fUrl = _root._url;
var delim = "\\";
if(fUrl.lastIndexOf(delim)===-1) delim = "/";
var swfName = fUrl.substring(fUrl.lastIndexOf(delim)+1, fUrl.lastIndexOf(".swf"));
_global.swfDirectory = fUrl.substring(0, fUrl.lastIndexOf(delim, fUrl.lastIndexOf(swfName)))+delim;
```

Copy the above to the actions window on the first frame of the movie, and you can then replace the above call to *loadMovieNum* with the much easier to manage:

```
loadMovieNum(_global.swfDirectory + "B.swf", 1);
```

Note that this issue will actually affect any attempt to use local file resources from your mediascape, such as the *loadVars* command, or when trying to load a local XML file. You can work around the issue using the same technique as shown here.

ActionScript

ActionScript is the scripting or programming language used in Adobe's Flash system. You can use it to control all aspects of a Flash movies behaviour and appearance.

You can create Flash movies in a drag-and-drop manner without ever writing any ActionScript by hand. Or you can do teh whole thing with ActionScript withou doing any drag-and-drop style creation. Even drawing shapes can be done with ActionScript and drawing commands.

The usual situation is somewhere in between. Those things that are easily done by drag-and-drop are best done that way and other more complex stuff can be done with bits of ActionScript.

ActionScript has similarities to JavaScript, so if you are familiar with that then ActionScript will not be too hard.

How to Debug Your Mediascape

When testing out your mediascape using mscapetester or mscapetester player, you may find out that certain parts are not working as you expected. This is particularly common in games as these often contain fairly complex pieces of script to define the rules of the game.

There are two different classes of error:

- **Script Syntax Errors** - this is when there are mistakes in the text of your script, for example missing out closing brackets, or forgetting to put a semicolon ; at the end of each line. This class of error can be fixed by ensuring you follow the correct syntax rules - see the document [Event scripts: an introduction](#) for more information.
- **Logic Errors** - Logic errors are when your script follows the syntax rules correctly, but does not perform the intended function. This document demonstrates some of the tools that can be used to help you fix this type of error.

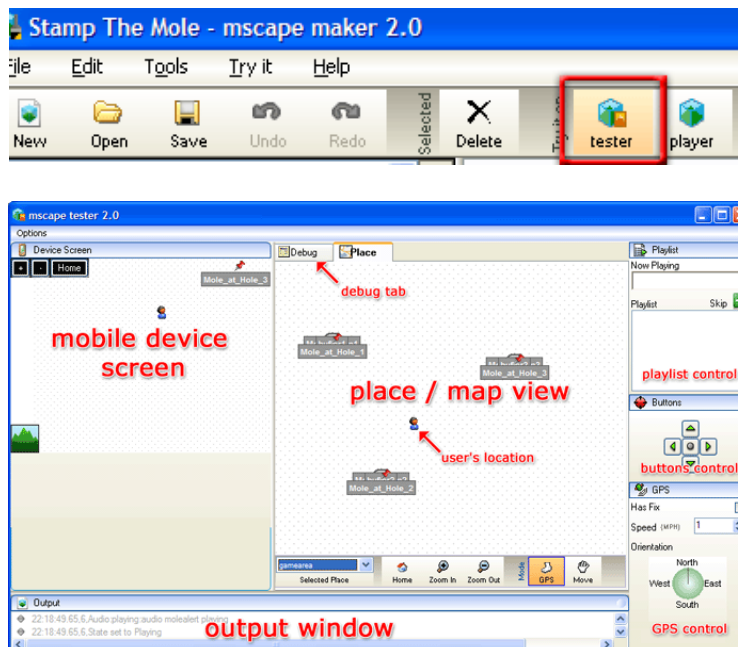
Note that the debug tools do not automatically fix problems for you, I'm afraid this is still up to you as the designer! What they do allow, however, is for you to thoroughly test your mediascape to find out what the problem is.

Using Debug Tools In Mscapetester

mscapetester has the richest set of debug tools. If you are in front of your desktop machine rather than out in the world, it's recommended to use mscapetester to debug logic errors in your mediascape. Of course, it is essential to also test on the mobile device as well!

In this example, we're going to use the ['Stamp the Mole' Mediascape](#). If you want to follow along, download the mediascape and open it in mscapetester.

- Click *Tester* to open the mediascape in mscapetester.



Some of the features you can use for debugging your mediascape are:

- Simulate GPS positions
 - You can also simulate the OnGotFix? event on the GPS object
- Checking the Value of Variables
 - You can also view the value of any property on any object
- Play Audio, Show Images
 - You can run any action on any object in your mediascape
- Seeing what's playing on the playlist
 - You can also skip the currently-playing audio

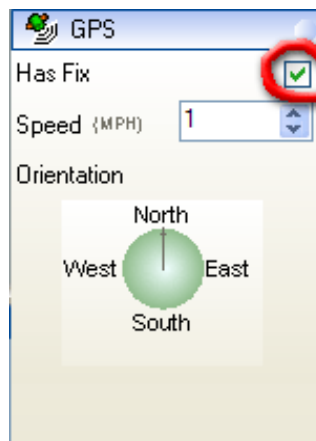
Simulating GPS Positions

This is one of the most basic functions that mscapetester provides.

- Ensure that the Place tab is visible, and that *GPS* mode is selected.
- You can then simulate GPS positions by clicking on the map - A small person icon represents the current location of the user.



- Some mediascapes may not begin until the GPS has a fix. You can simulate a GPS getting a fix by checking the *Has Fix* checkbox in the GPS control on the right-hand side of mscapetester.

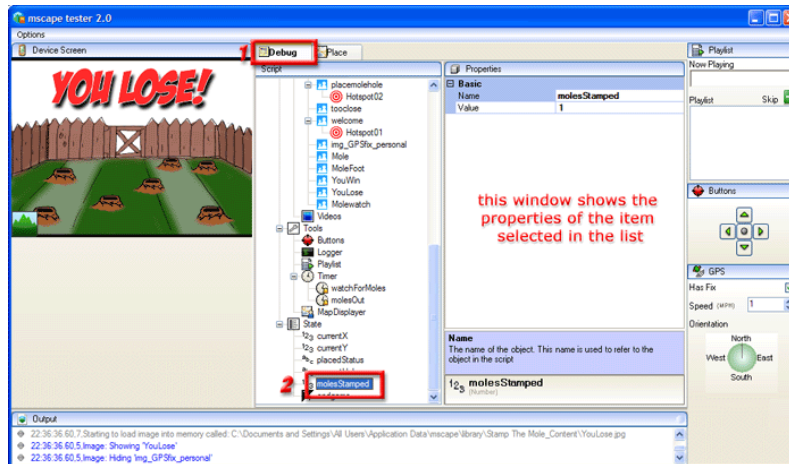


Checking the Value of Variables

Seeing the values of your variables is a very useful tool to help you debug your mediascape.

- Click the Debug tab
- Scroll the left panel down until you get to the section called *State*
- Click on the *molesStomped* variable

The right part of the window will show the properties of the selected item - the important one is the *Value* property.

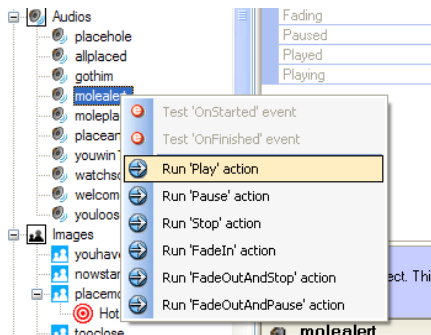


If you find at any point that the variable does not have the value you expected, you can change the value by typing a new value into the box.

Playing Audio, Showing Images

Using the Debug tab (see above) you can run any action on any object in mscscape tester. For example, you can play, pause, and stop any audio in your mediascape.

To do this, scroll to the *Audios* item, and right-click on an audio item. This will bring up a list of all the events and actions available for that object; select one from the list and mscscape tester will run it for you.



This same idea works for all objects including images, flash movies, videos, slideshows.

Using Debug Tools In Mscscape Player

It is possible to access many of the same debugging tools that are available in mscscape tester in mscscape player. However, you must first enable debugging tools in the mediascape that you wish to debug.

See the document [EnablingDebuggingInMscscapePlayer](#) for details on how to enable debugging for your mediascape

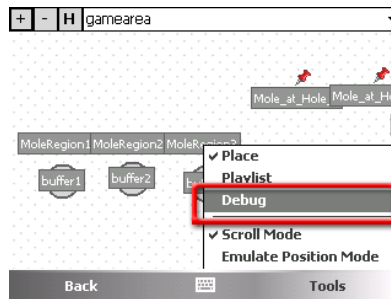
Once in Designer Tools, there are many different tasks you can perform

- Checking the Value of Variables
 - You can also view the value of any property on any object
- Play Audio, Show Images
 - You can run any action on any object in your mediascape
- Simulate GPS positions without a GPS Fix
 - You can also simulate the OnGotFix? event on the GPS object

Checking The Values Of Variables

It's possible to view the value of any of the variables you have defined in the State section of your mediascape. This is very useful for checking that variables have the values you expect.

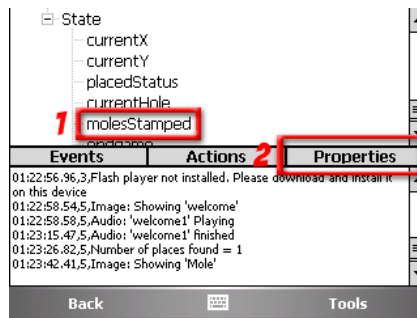
From the Designer Tools screen, tap the 'Tools' menu and select 'Debug'.



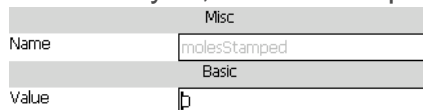
This will bring up the debug screen. The bottom half of the screen shows messages that are output from the mediascape, while the top half is the same list of mediascape objects that appears on the left of the mscape maker.

In this example, we're going to look at the value of the 'molesStamped' variable in the ['Stamp the Mole' Mediascape](#). If you want to follow along, download the mediascape, open it in mscape maker and make sure you enable the 'Allow Debugging' option as shown above before running the mediascape on your mobile device.

Scroll the upper window down until you pass the 'State' item. Select 'molesStamped' and press 'Properties'.

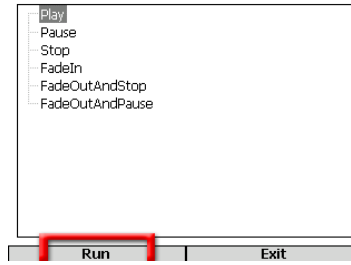


The screen will show the value of the variable. This technique works for any properties on any of the objects in the mediascape (e.g. an audio's Played, Volume etc properties).

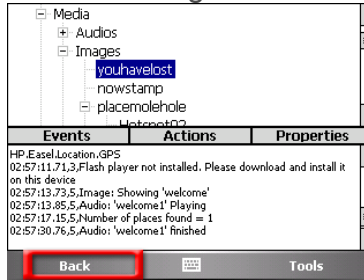


Playing Audio, Showing Images, and Running Other Actions

Follow the instructions above to display the Debug screen. To play an audio, scroll the upper window to the 'audios' section, tap on an audio, and click 'Actions'. You will see a list of actions that can be performed on the audio. Click 'Play', then click 'Run', and the audio will play.



To display an image, use the technique above to select an image file, and choose 'Show' from the list of actions. To see the image you'll need to exit Designer Tools mode by pressing the Back button.



This method can be used to run any action on any object in the mediascape.

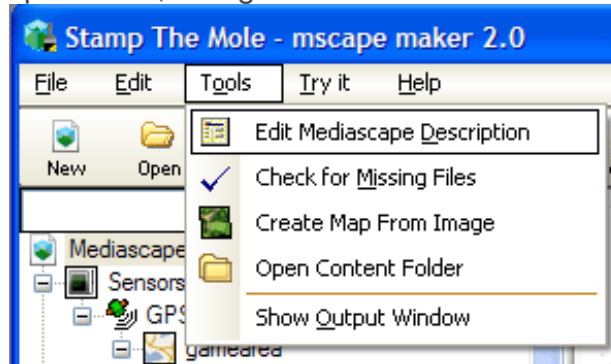
Simulate GPS Positions Without a GPS Fix

Sometimes you may want to simulate GPS on your mobile device when a GPS fix is not available, for example when you are inside, or when you don't have a GPS device. See the document [HowToSimulateGPSOnMscapePlayer](#) for details on how to perform this function.

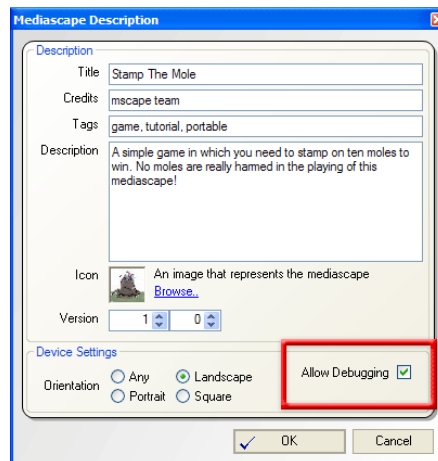
Enabling Debugging Tools in Mscape Player

It is possible to access many of the same debugging tools that are available in mscape tester in mscape tester. However, you must first enable debugging tools in the mediascape that you wish to debug.

Load your mediascape in mscape maker, and go to *Tools / Edit Mediascape Description*.

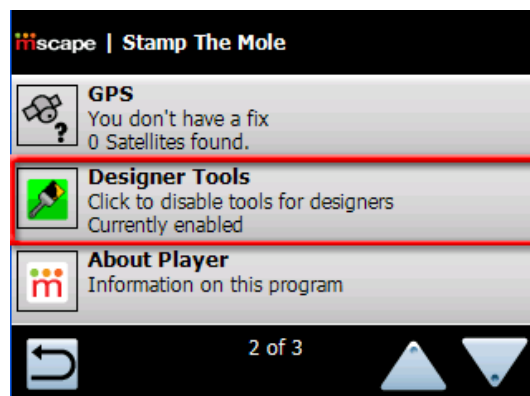


Ensure that 'Allow Debugging' is checked.



Copy the changed mediascape to the mobile device, load your mediascape and press the menu button to enter the menu system

Scroll down to the 'Designer Tools' option, and click it, ensuring that the text reads 'Currently Enabled'.



Click the back button to go back to the mediascape. A new 'paintbrush' icon will appear - click this to enter Designer Tools mode.



Once in Designer Tools, there are many different tasks you can perform

- Checking the Value of Variables
- You can also view the value of any property on any object
- Play Audio, Show Images
 - You can run any action on any object in your mediascape
- Simulate GPS positions without a GPS Fix
 - You can also simulate the *OnGotFix* event on the GPS object

See the document [HowToDebugYourMediascape](#) for more detail on the debugging tools.

Resolving Installation Problems

If you have problems installing mscape player on your PDA there is a manual alternative to the automatic installer.

Follow the steps below to achieve this.

- Connect your device to the machine via ActiveSync. Right-click the green ActiveSync icon in the system tray next to the clock, and choose Explore to bring up a Windows Explorer window.
- Open another Windows Explorer window and navigate to C:\Program Files\mscape\cabs (If you have installed mscape to a different directory this path may be different)
- Copy the file mscapeplayer.ARMV4.CAB from the cabs folder to your device
- If your device is Windows Mobile 5 or above, copy the file NETCFv2.wm.armv4i.cab across as well.
- If your device is Pocket PC 2003 Second Edition or above, copy the file NETCFv2.ppc.armv4.cab over.
- If fp7_ppc_en.cab is there (this is the Adobe Flash installer) copy that across too. If it is not there, follow the instructions you get from the library from File menu / Install Mscape Player / Install Flash
- On your device, go to Start Menu / Programs / File Explorer
- Navigate to where you copied the cab files to (the default is My Documents)
- Tap on the NETCF cab to install the .NET framework.
- When this finishes you may need to restart the ipaq
- Tap on mscapeplayer.ARMV4.cab to install mscape player
- Tap on fp7_ppc_en.cab to install flash

Get a Mediascape onto Your Mobile Device Using Mscape Library

Downloading

When you download a mediascape from the web site you will be asked if you want to save it to your computer or open it from the current location. The easiest thing to do is to choose to open it from the current location. The mscape library manager starts up and the mediascape is automatically loaded into your library.

(If you do choose to download it to your computer you can choose where to save it to and once it has downloaded you can double click it to start the mscape library manager.)

The mscape library

The mscape library is a tool that you use to manage your collections of mediascapes and to shift them back and forth between your desktop computer and your mobile device. If you own an iPod you can think of the mscape library as being a sort of iTunes for mediascapes!

The main part of the screen shows the list of mediascapes that you have on your desktop computer. If you click on one, the right-hand panel shows an overview of what it is about.

In the left-hand panel is a list of the different places that you can put mediascapes. Some of them are on your computer and some of them are on your mobile device (when it is connected to your computer). When you click them the central panel shows the list of mediascapes that are in that location.

Mediascapes on your mobile device

If you have a mobile device connected when you are running the mscape library you will see the different folders available for mediascape storage listed under *Mobile Device* on the left-hand panel.

Different mobile devices may offer different options here, but usually there will be folders called *Storage Card / SD Card* (if one is present), and *My Documents*. It doesn't make much difference where you put your mediascapes, the mscape player plays mediascapes in all these locations. The only consideration you might make is that there is usually more space on the storage card than on the other internal locations.

Copying mediascapes


To copy mediascapes from from the desktop machine to the mobile device.

- Ensure that *Library* is selected under *Desktop Computer* in the left-hand panel.

- Select the mediascapes you want to move by clicking the tick boxes next to them. It is possible to copy multiple mediascapes in one go, simply check the boxes for those you wish to copy.
- Click the 'Copy To' button at the bottom right of the mediascape list. The list of destination folders on the mobile device will be shown - select the appropriate one. If there is an SD card in your device we recommend that you choose that.
- mscape library will copy the selected mediascapes over to the device.

You can also copy mediascapes from your device to your desktop machine.

- Click the mobile device's folder that contains the mediascapes to copy (e.g. Storage Card). This is listed under *Mobile Device* on the left-hand panel. The list will change to show mediascapes in that folder on the mobile device.
- Select the mediascapes you want to move by clicking the tick boxes next to them. It is possible to copy multiple mediascapes in one go, simply check the boxes for those you wish to copy.

 If your desktop machine has an SD Card reader, it's possible to copy mediascapes directly to it. This is often much faster than copying via the mobile device itself. Insert the SD Card into your machine, and it will appear underneath *Desktop Machine* in the left-hand panel. Now when you click the 'Copy To' button, the SD Card will be listed as a destination to copy to.

When you are done

When you have copied the mediascape over you can view the contents of the location you copied it to, to confirm that it is there and then you can disconnect your mobile device.

Get a Mediascape onto Your Mobile Device, the Traditional Method

Download a mediascape

Download the mediascape called; 'Doubloons'. This will save the mediascape on your computer as a zip file. Choose the directory you want to save the file in. (You could create a My-Mediascapes directory and save all your mediascapes in that).

Open the zip file. Click the Open button on the download complete window. This will open up the zip file in a WinZip window.

Select Extract. Choose the directory you want the expanded mediascape to be in. (Keep it in your My-Mediascapes directory).

Transfer the mediascape to your mobile device

Now you want to copy the folder to your mobile device. There are several ways you can do this. The easiest is using the mscape library

ActiveSync

You connect your mobile device to your computer as described earlier in this section, and wait until the ActiveSync icon (a circle containing two arrows) in the lower right corner of your screen turns green.

Right click on it and select 'explore'. This shows the folders on your mobile device.

Drag and drop the mediascape files you have just downloaded (a file with the extension .msl and a corresponding content folder) into one of the folders on your mobile device (for example 'ipaq file store' or the SD card if your device has one).

SD card


If both your mobile device and your computer can read SD cards you can copy your mediascapes onto the card and transfer it that way.

Open up the File Explorer on your computer and go to your mediascapes directory. You should see a folder in which there is a file with the extension .msl and a corresponding content folder. Select these files and copy them to your SD card.

When this is complete transfer the SD card to your mobile device.

Installing Additional Languages to Mscape Player

Mscape Player 2.5 and above now supports non-english languages.

 At present multi-language support is only available in mscape player - mscape library, maker, and tester are English only.

In order to install an additional language pack, follow the instructions below.

- Run mscape library
- **Connect your mobile device.** If mscape player 2.5 or above is not already installed, click the 'Install mscape player' button that appears under mobile device, or select 'Install mscape player' from the File menu.
- **Ensure your machine is connected to the internet.**
- From the main menu of mscape library select Options / Download Language Pack.
- Tick the languages that you are interested in, and click the Download button.



- When the download completes, click Close.
- From the main menu of mscape library select Options / Install Language Pack.
- In the window that appears tick the languages that you wish to install to your device and click the Install button.
- When this completes, click Close.

To install the languages on additional devices, simply repeat the 'Install Language Pack' step above. You do not need to re-download the packs each time.

- To access the additional languages in mscape player select 'Language Settings' from the main menu, then tap the language you wish to use.

