

Creating your first Virtual Earth Web Page

By Dr. Neil Roodyn

This article will help you understand how to get started using version 2 of the Virtual Earth Map Control. The map control used in Virtual Earth is a JScript control and a cascading style sheet. Together these can be used to present a great user experience for online map content.

By the end of this article we will have created a web page that displays a map control and allows for some user input as show in figure 1.

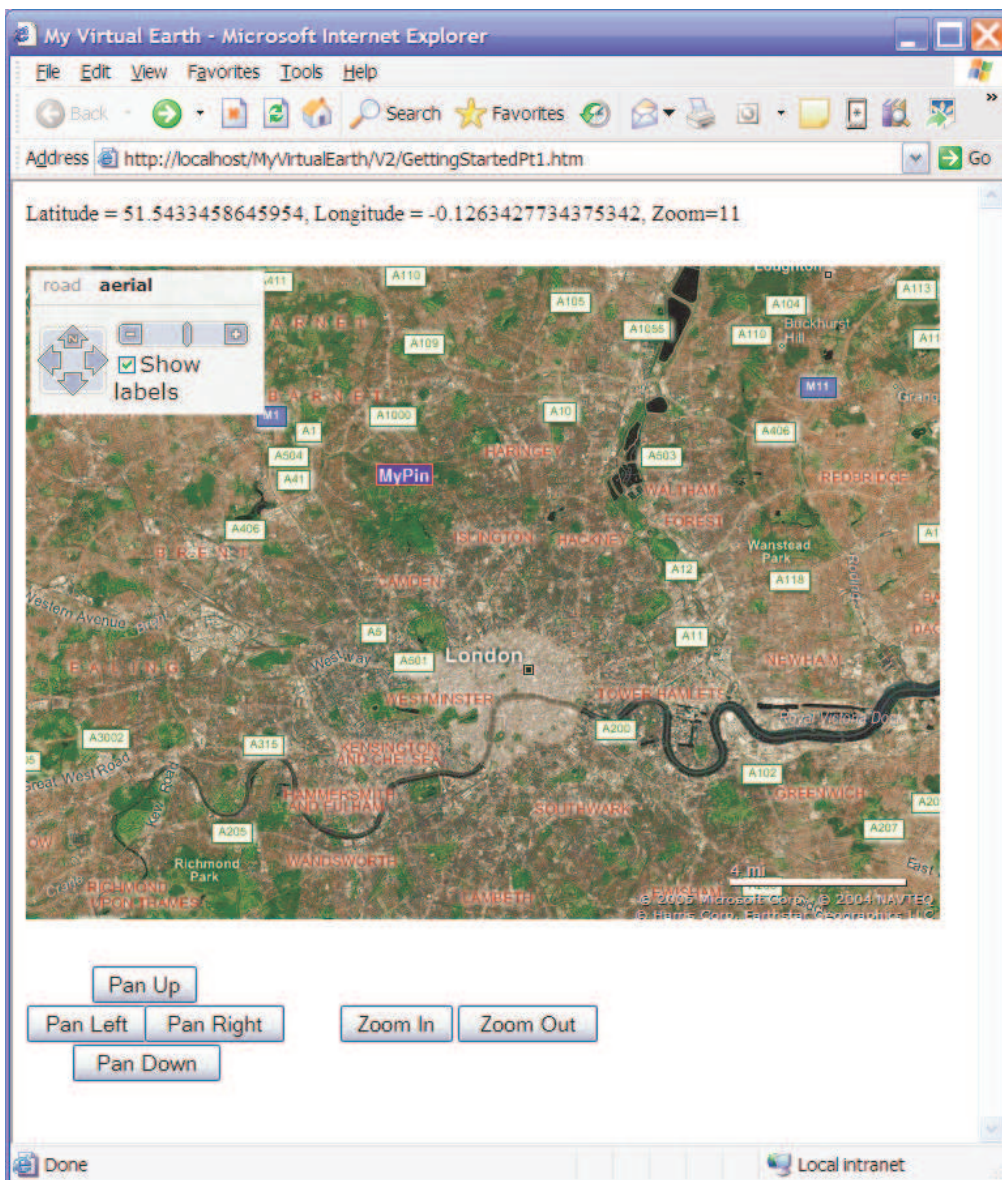


Figure 1

Using the MapControl

The Virtual Earth Map Control script can be found at <http://dev.virtualearth.net/standard/v2/MapControl.js>

You should link directly to this on your site to ensure you always use the most update version of the control.

Along with the Jscript for the control you will also need to include some styles that the control uses. These can be found in the cascading style sheet located in the same path <http://dev.virtualearth.net/standard/v2/MapControl.css>

Creating a Map Control instance

To create a new instance of a Virtual Earth Map Control you will need to write a small method in JScript on your page. This will create an instance of the MapControl, position it on the page and set up the initial content of the control.

The MapControl constructor has the following prototype:

```
Msn.VE.MapControl._constructor(map, params);
```

map

The map object you are creating.

params

An object containing the parameters required to initialize the map.

The params object parameter contains the following members

latitude	Specifies the latitude of the center point of the map.
longitude	Specifies the longitude of the center point of the map.
zoomlevel	Specifies the initial zoom level for the map.
mapStyle	Specifies the map style. Set this value using the MapStyle enumeration.
showScaleBar	Determines if the map scale is displayed on the map. Boolean.
showDashboard	Determines if the map controls (navigation and zoom controls) are displayed on the map. Boolean.
dashboardSize	Specifies the relative size of the dashboard. Set this value using either the DashboardSize enumeration.
dashboardX	Specifies the <i>x</i> position of the upper-left corner of the dashboard control, relative to the upper-left corner of the map object.
dashboardY	Specifies the <i>y</i> position of the upper-left corner of the dashboard control, relative to the upper-left corner of the map object.

A Boolean value that specifies whether the map control will support obliqueEnabled Bird's Eye images. If set to **true**, you must also specify the *obliqueUrl* parameter.

obliqueUrl Specifies the URL to the page for Bird's Eye images.

Example:

A simple web page with a Virtual Earth map control can then be created as shown in Listing 1.

```
<html>
<head>
  <title>My Virtual Earth</title>
  <link href="http://dev.virtualearth.net/standard/v2/MapControl.css"
        type="text/css" rel="stylesheet" />
  <script src="http://dev.virtualearth.net/standard/v2/MapControl.js">
  </script>
  <script>
    var map = null;

    function OnPageLoad()
    {
      var params = new Object();
      params.latitude = 32.69;
      params.longitude = -117.13;
      params.zoomlevel = 12;
      params.mapstyle = Msn.VE.MapStyle.Road;
      params.showScaleBar = true;
      params.showDashboard = true;
      params.dashboardSize = Msn.VE.DashboardSize.Normal;
      params.dashboardX = 5;
      params.dashboardY = 5;

      map =
        new Msn.VE.MapControl(
          document.getElementById("myMap"),
          params);

      map.Init();
    }
  </script>
</head>
<body onload="OnPageLoad()">
  <div id="myMap"
        style="WIDTH: 600px; HEIGHT: 400px; OVERFLOW:hidden">
  </div>
</body>
</html>
```

Listing 1.

This should allow you to create a page that looks similar to that shown in figure 2 below. The control provides a number of features for 'free'. You should be able to:

- move the map around by dragging it

- zoom in and out with the mouse wheel
- zoom in by double clicking on a location
- change the map style from road to aerial using the dashboard
- move around the map using the compass control on the dashboard
- zoom in and out of the map using the dashboard

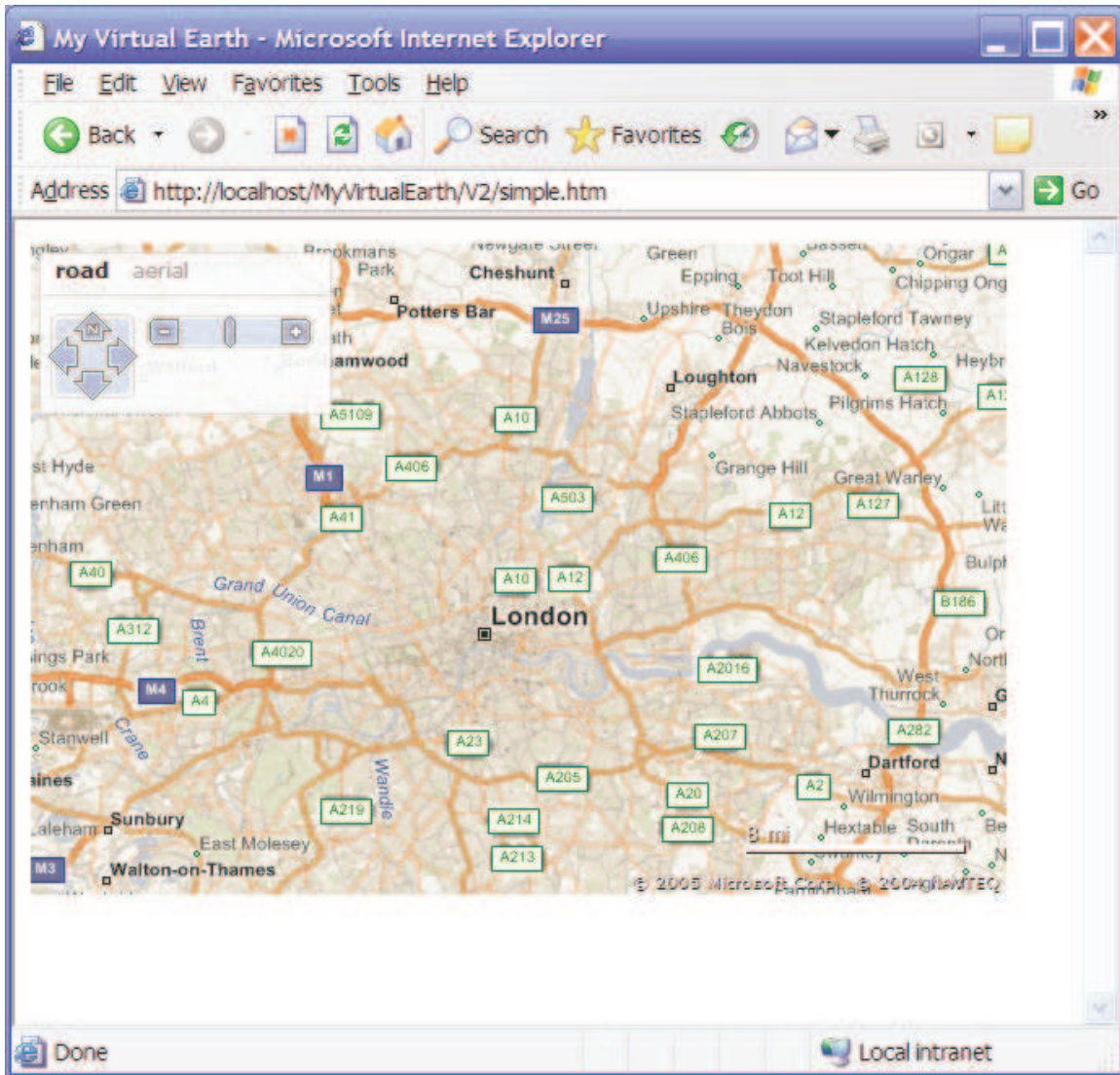


Figure 2

Receiving Events from the Map Control

As the map displayed on the control is changed the map control raises events that provide information about the map currently displayed in the control.

The events you can receive from the control are:

`onchangeview` Occurs whenever the map view changes.

<code>onclick</code>	Occurs when the user clicks on the map.
<code>Oncontextmenu</code>	Occurs when the user right-clicks on the map.
<code>onendcontinuouspan</code>	Occurs when a continuous pan of the map ends.
<code>onendzoom</code>	Occurs when the map zoom ends.
<code>onerror</code>	Occurs when there is a map control error.
<code>onmapstylechange</code>	Occurs when the map style changes.
<code>onmouseup</code>	Occurs when the user releases a mouse click on the map.
<code>onobliquechange</code>	Occurs only when the Bird's Eye image scene ID is changed. This event only fires if the map is currently displaying a Bird's Eye image and that image is changed.
<code>onobliqueenter</code>	Occurs when switching to Bird's Eye imagery from another map style.
<code>onobliqueleave</code>	Occurs when switching from Bird's Eye imagery to another map style.
<code>onresize</code>	Occurs when the map is resized.
<code>onstartcontinuouspan</code>	Occurs when a continuous pan of the map begins.
<code>onstartzoom</code>	Occurs when the map zoom begins.

All of the event functions are passed a single parameter. This event parameter is defined in the Jscript as:

```
function ni(e,gt,fi)
{
    this.view=e;
    this.oblique=gt;
    this.error=fi;
}
```

While it would have been nice if Microsoft had not obfuscated the code we can see that the `MapEvent` object contains a `MapView` object, an `Oblique` data object and error information.

For this first exercise we will focus on the `MapView` object.

The `MapView` class contains the following members:

mapStyle	The map style of the current map view.
zoomLevel	The zoom level of the current map view.
center	The center point (pixel) of the current map view.
latlong	The center point (LatLng) of the current map view.
pixelRect	The bounding rectangle (pixel) of the current map view.
latlongRect	The bounding rectangle (latlong) of the current map view.
sceneId	The ID of the oblique scene in the current map view. Returns null if MapStyle is not 'o'.
sceneOrientation	The orientation of the oblique scene in the current map view. Returns null if MapStyle is not 'o'.

The first events we will examine are the panning events. These are raised each time the map starts or stops panning, or scrolling. As you might expect the onstartcontinuouspan event is raised when the map starts scrolling and the onendcontinuouspan event is raised when the map control finishes scrolling the map.

To the simple page we created in the previous step we can add some code to handle the onendcontinuouspan event and display information as to the new centre point of the map.

The code shown in Listing 2 demonstrates this.

```

<html>
<head>
  <title>My Virtual Earth</title>
  <link
href="http://dev.virtualearth.net/standard/v2/MapControl.css"
  type="text/css" rel="stylesheet" />
<script src="http://dev.virtualearth.net/standard/v2/MapControl.js">
</script>

  <script>
var map = null;

function OnPageLoad()
{
  var params = new Object();
  params.latitude = 51.567;
  params.longitude = -0.026;
  params.zoomlevel = 10;
  params.mapstyle = Msn.VE.MapStyle.Road;
  params.showScaleBar = true;
}

```

```

        params.showDashboard = true;
        params.dashboardSize = Msn.VE.DashboardSize.Normal;
        params.dashboardX = 5;
        params.dashboardY = 5;

        map =
            new Msn.VE.MapControl(
                document.getElementById("myMap"),
                params);

        map.Init();

        map.AttachEvent("onendcontinuouspan",
            function(e)
            {
                document.getElementById("info").innerHTML =
                    'Latitude = ' +
                    e.view.latlong.latitude +
                    ', Longitude = '
                    + e.view.latlong.longitude +
                    '), Zoom=' +
                    e.view.zoomLevel;
            } ) ;
    }
</script>
</head>
<body onLoad="OnPageLoad()">
    <div id="info" style="HEIGHT: 50px; font-size:10pt">
</div>

    <div id="myMap"
        style="WIDTH: 600px; HEIGHT: 400px; OVERFLOW:hidden">
    </div>
</body>
</html>

```

Listing 2

We can do the same with the onendzoom event by adding a function to handle that event, Listing 3.

```

<html>
<head>
    <title>My Virtual Earth</title>

<link href="http://dev.virtualearth.net/standard/v2/MapControl.css"
    type="text/css" rel="stylesheet" />
<script src="http://dev.virtualearth.net/standard/v2/MapControl.js">
</script>

    <script>
        var map = null;

        function OnPageLoad()
        {

```

```

var params = new Object();
params.latitude = 51.567;
params.longitude = -0.026;
params.zoomlevel = 10;
params.mapstyle = Msn.VE.MapStyle.Road;
params.showScaleBar = true;
params.showDashboard = true;
params.dashboardSize = Msn.VE.DashboardSize.Normal;
params.dashboardX = 5;
params.dashboardY = 5;

map =
    new Msn.VE.MapControl(
        document.getElementById("myMap"),
        params);

map.Init();

map.AttachEvent("onendcontinuouspan",
    function(e)
    {
        document.getElementById("info").innerHTML =
            'Latitude = ' +
            e.view.latlong.latitude +
            ', Longitude = '
            + e.view.latlong.longitude +
            '), Zoom=' +
            e.view.zoomLevel;
    } ) ;

map.AttachEvent("onendzoom",
    function(e)
    {
        document.getElementById("info").innerHTML =
            'Latitude = ' +
            e.view.latlong.latitude +
            ', Longitude = '
            + e.view.latlong.longitude +
            '), Zoom=' +
            e.view.zoomLevel;
    } ) ;

}
</script>
</head>
<body onLoad="OnPageLoad()">
    <div id="info" style="HEIGHT: 50px; font-size:10pt">
</div>

    <div id="myMap"
    style="WIDTH: 600px; HEIGHT: 400px; OVERFLOW:hidden">
</div>
</body>
</html>

```

Listing 3

The page should now appear as shown in Figure 3.

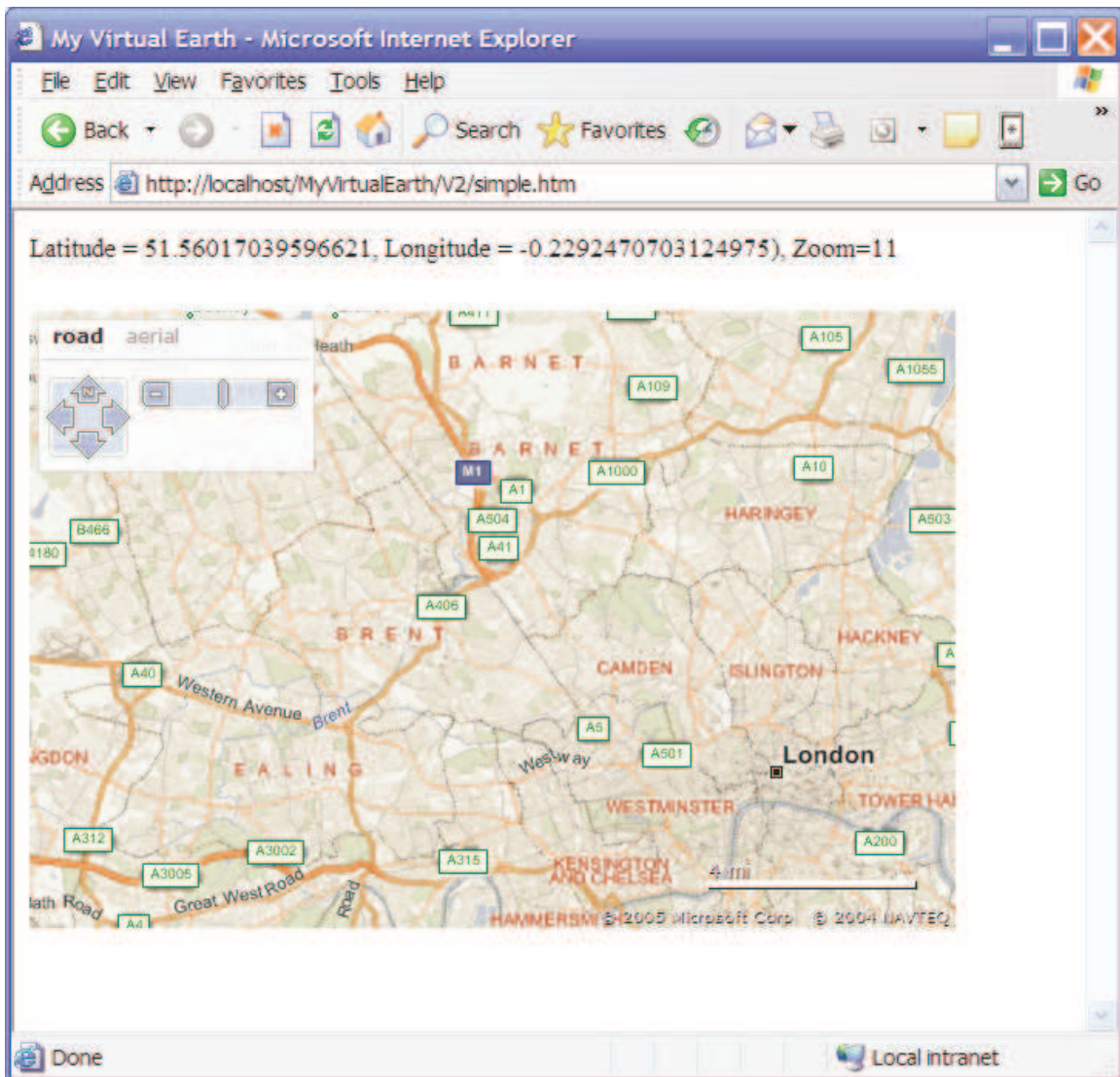


Figure 3

Refactoring out the Duplicate Code

In our Jscript we now have duplicate code in the onendcontinuouspan and onendzoom event handlers. It is a good practice to replace duplicate code with a single function that is called from both places. The AttachEvent method allows you to use a method name as well as putting the code inline.

The code can therefore be changed to look like Listing 4.

```
<html>
<head>
  <title>My Virtual Earth</title>

<link href="http://local.live.com/css/MapControl.css"
  type="text/css" rel="stylesheet" />
```

```

<script src="http://local.live.com/MapControl.ashx">
</script>

<script>
var map = null;

function UpdateInfo(e)
{
    document.getElementById("info").innerHTML =
        'Latitude = ' +
        e.view.latlong.latitude +
        ', Longitude = '
        + e.view.latlong.longitude +
        ', Zoom=' +
        e.view.zoomLevel;
}

function OnPageLoad()
{
    var params = new Object();
    params.latitude = 51.567;
    params.longitude = -0.026;
    params.zoomlevel = 10;
    params.mapstyle = Msn.VE.MapStyle.Road;
    params.showScaleBar = true;
    params.showDashboard = true;
    params.dashboardSize = Msn.VE.DashboardSize.Normal;
    params.dashboardX = 5;
    params.dashboardY = 5;

    map =
        new Msn.VE.MapControl (
            document.getElementById("myMap"),
            params);

    map.Init();

    map.AttachEvent("onendcontinuouspan",
        UpdateInfo);

    map.AttachEvent("onendzoom",
        UpdateInfo) ;

}
</script>
</head>
<body onLoad="OnPageLoad()">
<div id="info" style="HEIGHT: 50px; font-size:10pt">
</div>

<div id="myMap"
style="WIDTH: 600px; HEIGHT: 400px; OVERFLOW:hidden">
</div>
</body>
</html>

```

Listing 4: refactored out the duplicated code

Changing the Map Style

Previously, in the first version of the Virtual Earth map control you had to add code to change the map style between aerial, road and hybrid. The new MapControl does most of the hard work for us. You may still want to change the map style from your code and the mechanism for doing this remains the same, using the SetMapStyle function.

The available map styles are:

- aerial: an aerial satellite image of the map display
- road: a street map of the area in the control.
- hybrid: a combination of aerial and vector. The aerial image is overlaid
- oblique: oblique or birds eye imagery

Once the map control is displayed it is possible to change the map style by using the SetMapStyle function on the Map Control:

```
SetMapStyle (mapStyle)
```

The function takes one parameter to indicate the style. Similar to setting the initial style the parameter is the first single character of the style; 'a', 'r', 'h' or 'o'. Alternatively (and preferably) you can use the MapStyle enumeration
MapStyle { Road, Aerial, Hybrid, Oblique }

Adding a Pushpin to the Map

The ability to add pins to the map allows us to indicate particular locations on the map control. Pins work by overlaying information on the map control. The prototype of the AddPushpin method looks like this:

```
AddPushpin (id, lat, lon, width, height, className, innerHtml, zIndex)
```

id: the identifier of the pin. This should be unique for the pin on the map control.

lat: the latitude of the location to place the pushpin

lon: the longitude of the location to place the pushpin

width: the width of the pin

height: the height of the pin

The width and height are used to calculate the offset of the pin so that the center of the pin is at the latitude and longitude specified.

HINT: If you want to have the bottom right corner of the pin at the latitude and longitude then you could double these values.

Classname: name of a style class for the pin. Without this the pin will not be displayed. This can be described in a CSS file or inline as per the example below.

innerHTML: the text to go in the Pushpin

zIndex: the order in which the pins are drawn on the map.

In the following example (Listing 5) the onclick event is used to add a pushpin to the map where the user clicked.

```
<html>
<head>
  <title>My Virtual Earth</title>
<STYLE TYPE="text/css" MEDIA=screen>
<!--
.pin
{
width:44px;height:17px;
font-family:Arial,sans-serif;
font-weight:bold;font-size:8pt;
color:White;overflow:hidden;
cursor:pointer;text-decoration:none;
text-align:center;background:#0000FF;
border:1px solid #FF0000;
z-index:5}
-->
</STYLE>

<link href="http://local.live.com/css/MapControl.css"
      type="text/css" rel="stylesheet" />
<script src="http://local.live.com/MapControl.ashx">
</script>

  <script>
var map = null;

function UpdateInfo(e)
{
    document.getElementById("info").innerHTML =
        'Latitude = ' +
        e.view.latlong.latitude +
        ', Longitude = '
        + e.view.latlong.longitude +
        ', Zoom=' +
        e.view.zoomLevel;
}

function MouseClick(e)
{
    map.AddPushpin('pin',
        e.view.latlong.latitude,
        e.view.latlong.longitude,
        88,
```

```

        34,
        'pin',
        'MyPin',
        1);
    }

    function OnPageLoad()
    {
        var params = new Object();
        params.latitude = 51.567;
        params.longitude = -0.026;
        params.zoomlevel = 10;
        params.mapstyle = Msn.VE.MapStyle.Road;
        params.showScaleBar = true;
        params.showDashboard = true;
        params.dashboardSize = Msn.VE.DashboardSize.Normal;
        params.dashboardX = 5;
        params.dashboardY = 5;

        map =
            new Msn.VE.MapControl (
                document.getElementById("myMap"),
                params);

        map.Init();

        map.AttachEvent("onendcontinuouspan",
            UpdateInfo);

        map.AttachEvent("onendzoom",
            UpdateInfo) ;

        map.AttachEvent("onclick",
            MouseClick);
    }
</script>
</head>
<body onLoad="OnPageLoad()">
    <div id="info" style="HEIGHT: 50px; font-size:10pt">
</div>

    <div id="myMap"
    style="WIDTH: 600px; HEIGHT: 400px; OVERFLOW:hidden">
</div>
</body>
</html>

```

Listing 5

Viewing this page in a browser will now allow push pins to be added by clicking on the map. There are several issues with this:

- Each time the map is dragged another push pin gets added.
- Double clicking on the map to zoom in no longer works, because a pin is added first that receives the double click event.

- It is possible to add multiple pins with the same identifier.

A solution would be not to add push pins using the onclick event. For now let's stick with this and each time we add a pin remove the previous pin.

To remove a pin use the RemovePushpin function.

```
RemovePushpin(id);
```

This function takes a single parameter which identifies the pin to remove.

Removing a pushpin will remove any pushpins that share the same identifier.

The MouseClick function shown in Listing 5 can be changed to remove the previous pin as shown in Listing 6.

```
map.onMouseClick=function(e)
{
    map.RemovePushpin('pin');
    map.AddPushpin('pin',
    e.latitude,
    e.longitude,
    88,
    34,
    'pin',
    'MyPin');
}
```

Listing 6

We should now have a page that will contain a single push pin indicating the last place on the map that was clicked, Figure 5.

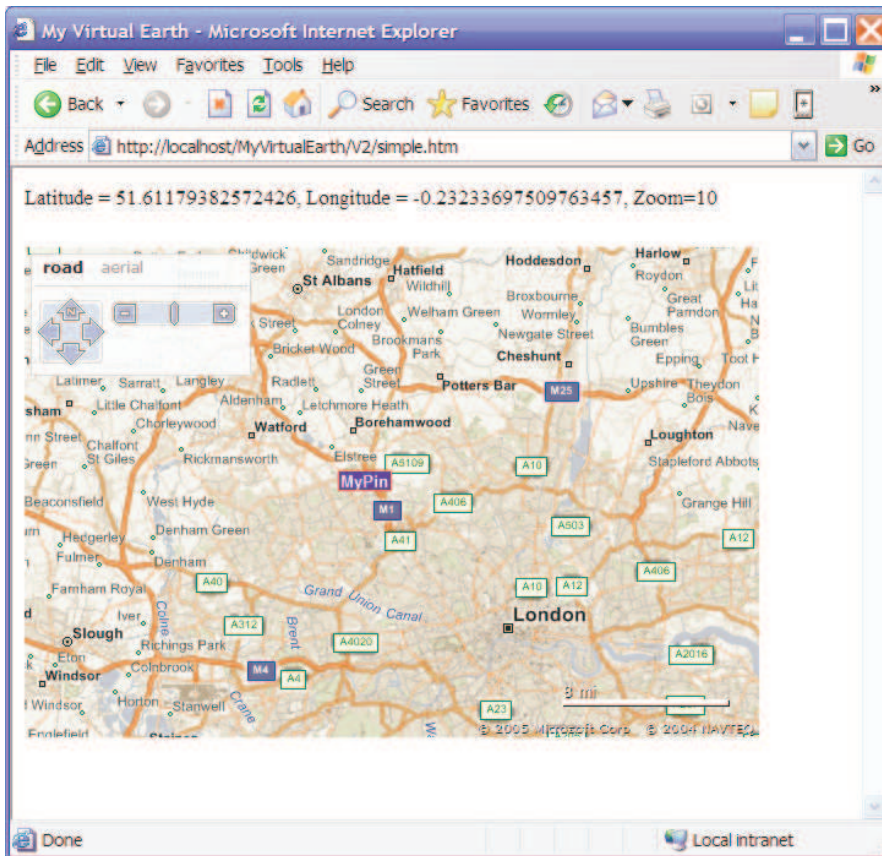


Figure 5

Adding Navigation Controls

The map control has some useful built in navigation features but it is sometimes useful to provide extra controls on the web page to allow the user to navigate around the map. In this final section of this article you will learn how to add buttons to the web page that can control the contents of the map control.

Panning

We will start by adding buttons to pan around the map. In the body element of the HTML page buttons can be added with simple HTML code:

```
<input type="button" value="Pan Up"
onclick="DoPanUp () "
ID="PanUpButton" NAME="PanUpButton"
style="position:absolute;left:60px;top:500" />
```

```
<input type="button" value="Pan Left"
onclick="DoPanLeft () "
ID="PanLeftButton" NAME="PanLeftButton"
style="position:absolute;left:10px;top:530" />
```

```

<input type="button" value="Pan Right"
onclick="DoPanRight()"
ID="PanRightButton" NAME="PanRightButton"
style="position:absolute;left:100px;top:530"/>

<input type="button" value="Pan Down"
onclick="DoPanDown()"
ID="PanDownButton" NAME="PanDownButton"
style="position:absolute;left:45px;top:560"/>

```

The code to action on the button clicks can then be added to the script section in the page. Using the PanMap method on the map control. PanMap takes 2 parameters, x and y. These indicate by how much to pan the map view in the x and y directions.

```

function DoPanUp()
{
    map.PanMap(0, -100);
}
function DoPanDown()
{
    map.PanMap(0, 100);
}
function DoPanLeft()
{
    map.PanMap(-100, 0);
}
function DoPanRight()
{
    map.PanMap(100, 0);
}

```

If you load this page in a browser and click the buttons you can see the map jump around. This is not a great user experience. It would be nicer to show the map scrolling smoothly in each of the directions. This can be achieved using the ContinuousPan function of the map control. ContinuousPan takes a third parameter along with the x and y. This third parameter indicates how many times to repeat the pan. In this way a number of small operations can be put together to provide the appearance of the map scrolling.

```

function DoPanUp()
{
    map.ContinuousPan(0, -10, 20);
}
function DoPanDown()
{
    map.ContinuousPan(0, 10, 20);
}

function DoPanLeft()
{
    map.ContinuousPan(-10, 0, 20);
}
function DoPanRight()
{
    map.ContinuousPan(10, 0, 20);
}

```

Zooming

Next we will add 2 buttons in the HTML; to zoom in and to zoom out.

```
<input type="button" value="Zoom In" onclick="DoZoomIn()"
ID="ZoomInButton" NAME="ZoomInButton"
style="position:absolute;left:250px;top:530"/>
```

```
<input type="button" value="Zoom Out" onclick="DoZoomOut()"
ID="ZoomOutButton" NAME="ZoomOutButton"
style="position:absolute;left:340px;top:530"/>
```

The accompanying script code can use the ZoomIn and ZoomOut functions of the map control. Each functional call will simply increase or decrease the zoom level by 1.

```
function DoZoomIn()
{
    map.ZoomIn();
}

function DoZoomOut()
{
    map.ZoomOut();
}
```

Conclusion

If you have followed along with this article you should now have a page that looks similar to that shown in Figure 1. The complete code listing is provided in Listing 7 below.

Using the Virtual Map Control is relatively simple and it provides a very compelling user experience for mapping and location identification.

```
<html>
<head>
    <title>My Virtual Earth</title>
<STYLE TYPE="text/css" MEDIA=screen>
<!--
.pin
{
width:44px;height:17px;
font-family:Arial,sans-serif;
font-weight:bold;font-size:8pt;
color:White;overflow:hidden;
cursor:pointer;text-decoration:none;
text-align:center;background:#0000FF;
border:1px solid #FF0000;
z-index:5}
-->
</STYLE>
```

```

<link href="http://local.live.com/css/MapControl.css"
      type="text/css" rel="stylesheet" />
<script src="http://local.live.com/MapControl.ashx">
</script>

<script>
var map = null;

function UpdateInfo(e)
{
    document.getElementById("info").innerHTML =
        'Latitude = ' +
        e.view.latlong.latitude +
        ', Longitude = '
        + e.view.latlong.longitude +
        ', Zoom=' +
        e.view.zoomLevel;
}

function MouseClick(e)
{
    map.RemovePushpin('pin');
    map.AddPushpin('pin',
        e.view.latlong.latitude,
        e.view.latlong.longitude,
        88,
        34,
        'pin',
        'MyPin',
        1);
}

function OnPageLoad()
{
    var params = new Object();
    params.latitude = 51.567;
    params.longitude = -0.026;
    params.zoomlevel = 10;
    params.mapstyle = Msn.VE.MapStyle.Road;
    params.showScaleBar = true;
    params.showDashboard = true;
    params.dashboardSize = Msn.VE.DashboardSize.Normal;
    params.dashboardX = 5;
    params.dashboardY = 5;

    map =
        new Msn.VE.MapControl (
            document.getElementById("myMap"),
            params);

    map.Init();

    map.AttachEvent("onendcontinuouspan",
        UpdateInfo);

    map.AttachEvent("onendzoom",

```

```

        UpdateInfo) ;

        map.AttachEvent("onclick",
            MouseClick);
    }

    function DoPanUp()
    {
        map.ContinuousPan(0, -10, 20);
    }
    function DoPanDown()
    {
        map.ContinuousPan(0, 10, 20);
    }

    function DoPanLeft()
    {
        map.ContinuousPan(-10, 0, 20);
    }
    function DoPanRight()
    {
        map.ContinuousPan(10, 0, 20);
    }

    function DoZoomIn()
    {
        map.ZoomIn();
    }

    function DoZoomOut()
    {
        map.ZoomOut();
    }

</script>
</head>
<body onLoad="OnPageLoad()" >
    <div id="info" style="HEIGHT: 50px; font-size:10pt">
</div>

    <div id="myMap"
    style="WIDTH: 600px; HEIGHT: 400px; OVERFLOW:hidden">
</div>

    <input type="button" value="Pan Up"
    onclick="DoPanUp()"
    ID="PanUpButton" NAME="PanUpButton"
    style="position:absolute;left:60px;top:500"/>

    <input type="button" value="Pan Left"
    onclick="DoPanLeft()"
    ID="PanLeftButton" NAME="PanLeftButton"
    style="position:absolute;left:10px;top:530"/>

    <input type="button" value="Pan Right"
    onclick="DoPanRight()"

```

```
ID="PanRightButton" NAME="PanRightButton"
style="position:absolute;left:100px;top:530"/>

<input type="button" value="Pan Down"
onclick="DoPanDown()"
ID="PanDownButton" NAME="PanDownButton"
style="position:absolute;left:45px;top:560"/>

<input type="button" value="Zoom In" onclick="DoZoomIn()"
ID="ZoomInButton" NAME="ZoomInButton"
style="position:absolute;left:250px;top:530"/>

<input type="button" value="Zoom Out" onclick="DoZoomOut()"
ID="ZoomOutButton" NAME="ZoomOutButton"
style="position:absolute;left:340px;top:530"/>

</body>
</html>
```

Listing 7