

Creating your first Virtual Earth Web Page

By Dr. Neil Roodyn

This article will help you understand how to get started using the Virtual Earth Map Control. The map control used in Virtual Earth is a JScript control that presents a great user experience for map content.

This article can be downloaded in PDF [here](#).

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.

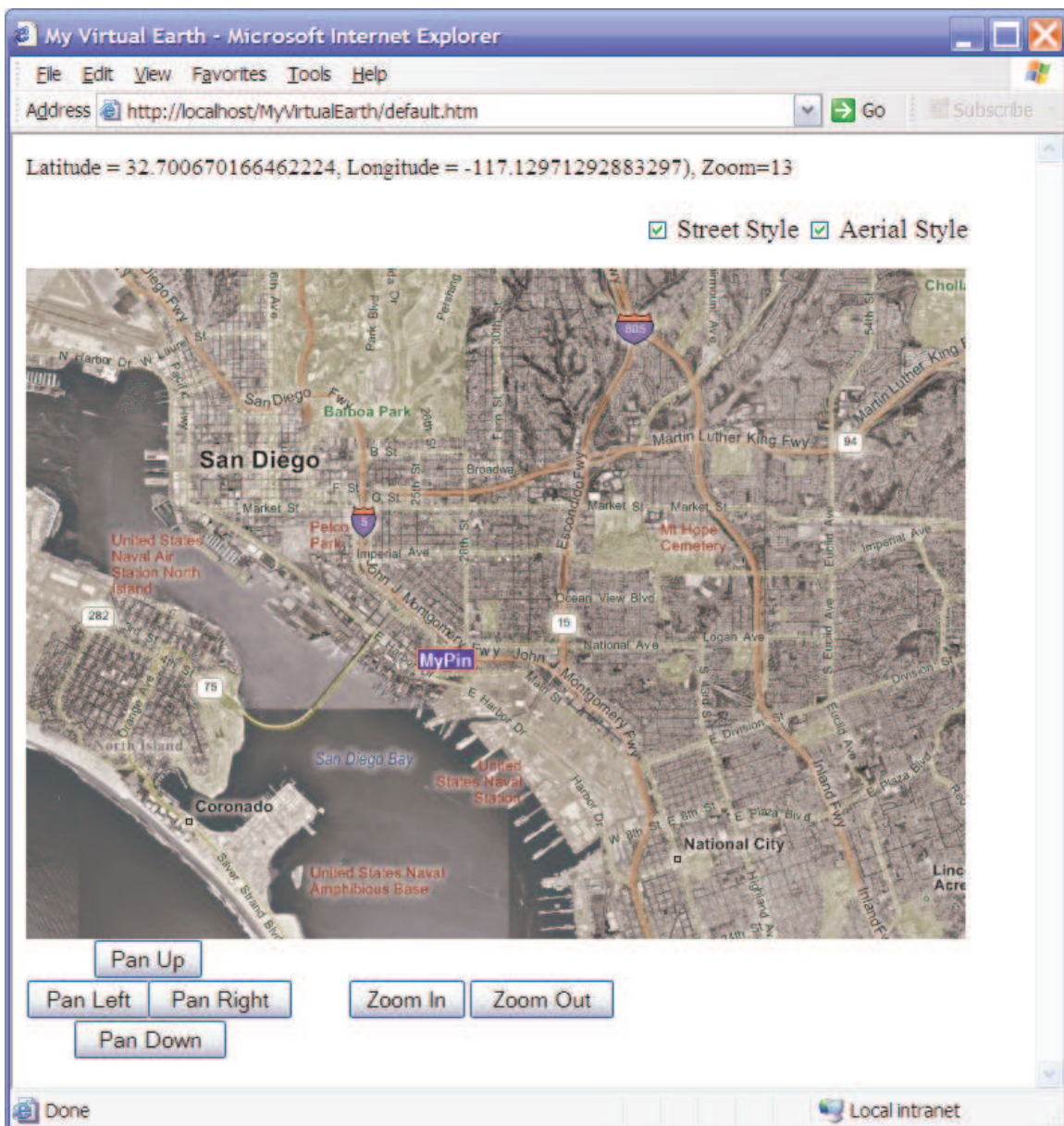


Figure 1

Using the MapControl

The Virtual Earth Map Control script can be found at

<http://virtualearth.msn.com/js/MapControl.js>

You can link directly to this on your site but that will cause issues with security. By default most browsers will not allow Jscript to run from another site than the one being browsed. Users would have to confirm that they wanted to run the Jscript from the VirtualEarth site. This doesn't create a great user experience.

The simple solution is to copy the MapControl.js file to your site. Once you have a copy of the JS file in your domain then the map control can be accessed.

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 VE_MapControl constructor has the following prototype:

```
VE_MapControl(Latitude, Longitude, Zoom, MapStyle, PositionType, Left, Top, Width, Height);
```

Latitude – the latitude of the centre point of the map to display in the control

Longitude – the longitude of the centre point of the map to display in the control

Zoom – the scale to zoom in of the map displayed. A number from 2 to 18. Where 2 is zoomed as far out as the control will allow and 18 is zoomed in as close as the control will allow.

MapStyle – the style to use when displaying the map. There are currently 3 style options; aerial, road, and hybrid. To set a style use the first letter (lowercase) of the style.

a – aerial: Displays an aerial satellite image of the map display

r – road: Displays a vector street map of the area in the control.

h – hybrid: Displays a combination of aerial and vector. The aerial image is overlaid with vector street and location information.

PositionType – the way in which the control should be positioned on the page. The two main options here are “relative” and “absolute”.

Left – the position of the left of the control on the page

Top – the position of the top of the control on the page

Width – the width of the control.

Height – the height of the control.

Example:

```
map = new VE_MapControl(32.69, -117.13, 12, 'r', "absolute", 400, 10, 400, 300);
```

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>
  <script src="MapControl.js"></script>
  <script>
    var map = null;

    function OnPageLoad()
    {
      map = new VE_MapControl(32.69, -117.13, 12,
        'r', "absolute", 10, 10, 700, 500);
      document.body.appendChild(map.element);
    }
  </script>
</head>

<body onLoad="OnPageLoad()" >
</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

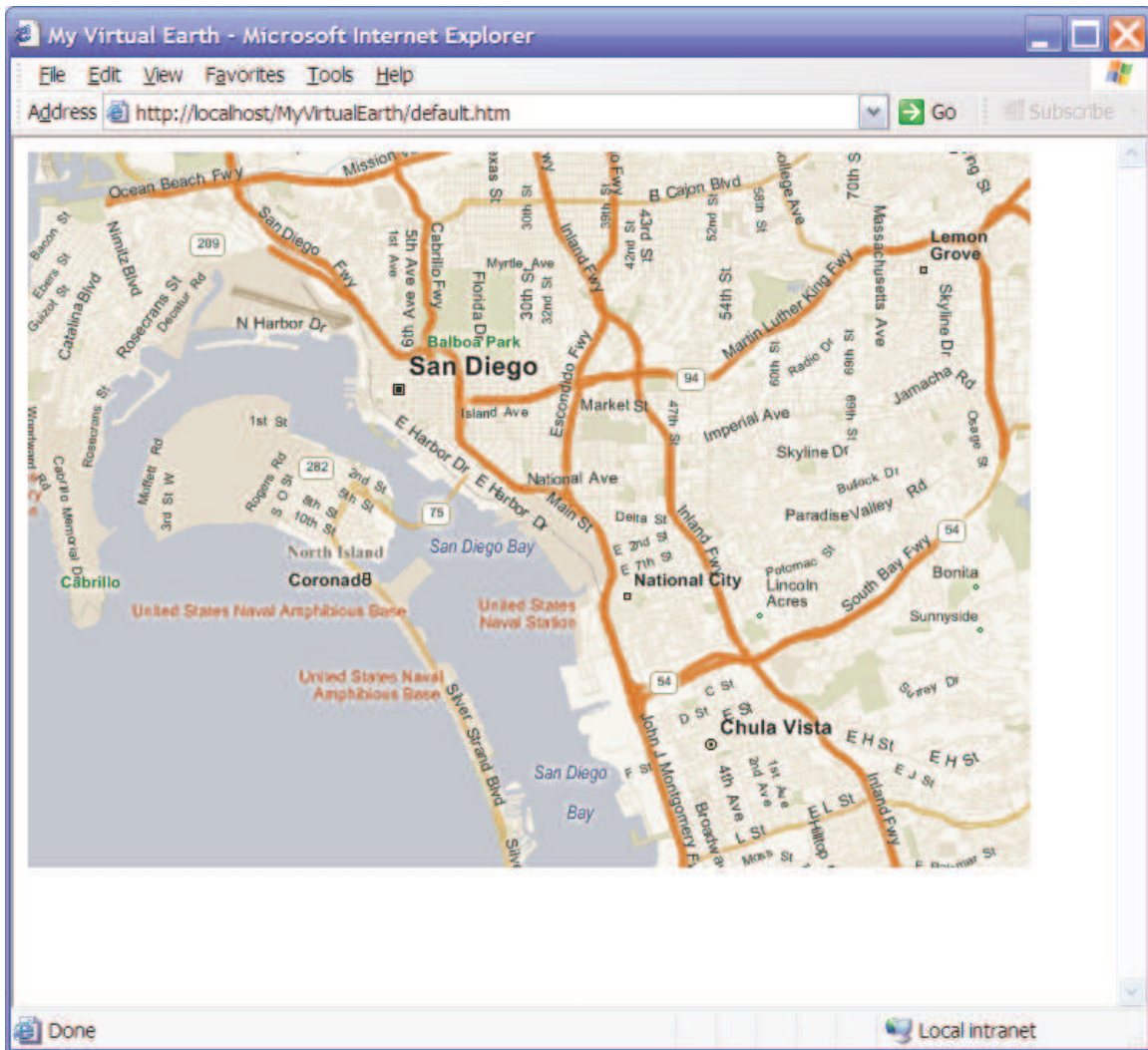


Figure 2

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:

```
onStartContinuousPan  
onEndContinuousPan  
onStartZoom  
onEndZoom  
onMouseClicked  
onMouseDown  
onMouseUp
```

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

```
function VE_MapEvent(srcMapControl, latitude, longitude, zoomLevel)
```

```

{
    this.srcMapControl=srcMapControl;
    this.latitude=latitude;
    this.longitude=longitude;
    this.zoomLevel=zoomLevel;
}

```

The latitude and longitude indicates the centre point of the map. The zoomlevel provides the amount zoom applied to the map displayed.

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>
    <script src="MapControl.js"></script>
    <script>
        var map = null;

        function OnPageLoad()
        {
            map = new VE_MapControl(32.69, -117.13, 12,
                'r', "absolute", 10, 100, 700, 500);
            document.body.appendChild(map.element);

            map.onEndContinuousPan=function(e)
            {
                document.getElementById("info").innerHTML =
                    'Latitude = ' +
                    e.latitude +
                    ', Longitude = '
                    + e.longitude +
                    '), Zoom=' +
                    e.zoomLevel;
            }
        }
    </script>
</head>

<body onLoad="OnPageLoad()">
<div id="info" style="font-size:10pt">
</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>
  <script src="MapControl.js"></script>
  <script>
    var map = null;

    function OnPageLoad()
    {
      map = new VE_MapControl(32.69, -117.13, 12,
        'r', "absolute", 10, 100, 700, 500);
      document.body.appendChild(map.element);

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

      map.onEndZoom=function(e)
      {
        document.getElementById("info").innerHTML =
          'Latitude = ' +
          e.latitude +
          ', Longitude = ' +
          e.longitude +
          '), Zoom=' +
          e.zoomLevel;
      }
    }
  </script>
</head>

<body onLoad="OnPageLoad()">
<div id="info" style="font-size:10pt">
</div>

</body>

</html>
```

Listing 3

The page should now appear as shown in Figure 3.

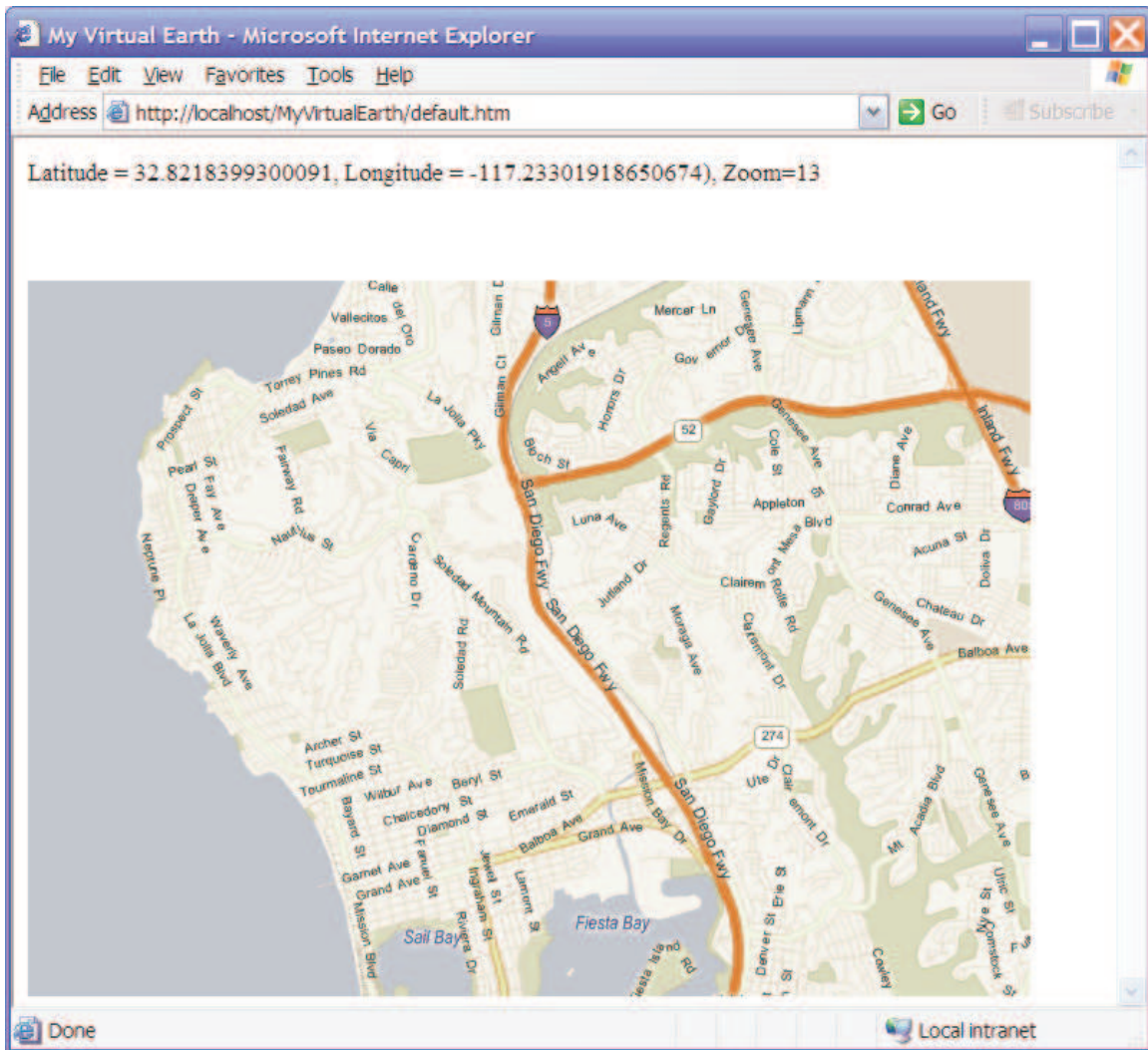


Figure 3

Changing the Map Style

Previously, in the section describing the creation of the Map Control, the 3 different map styles were explained.

The 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

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' or 'h'.

On the page we are creating let's add two checkboxes to allow the user to change the map style.

```
<html>
<head>
  <title>My Virtual Earth</title>
  <script src="MapControl.js"></script>
  <script>
    var map = null;

    function OnPageLoad()
    {

      map = new VE_MapControl(32.69, -117.13, 12,
        'r', "absolute", 10, 100, 700, 500);
      document.body.appendChild(map.element);

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

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

    function ChangeMapStyle()
    {
      var Aerial = document.getElementById("AerialStyleCheck");
      var Vector = document.getElementById("VectorStyleCheck");
      var s = 'r';
      if (Aerial.checked && Vector.checked)
      {
        s = 'h';
      }
      else if (Aerial.checked)
      {
        s = 'a';
      }
      map.SetMapStyle(s);
    }

  </script>
</head>

<body onLoad="OnPageLoad()">
<div id="info" style="font-size:10pt">
</div>
<div id="MapStyle">
```

```
        style="POSITION:absolute;LEFT:470px;TOP:60px">
            <INPUT id="VectorStyleCheck"
            type="checkbox"
            name="VectorStyleCheck"
            onclick="ChangeMapStyle()"
            checked="checked"
            >
            Street Style
            <INPUT id="AerialStyleCheck"
            type="checkbox"
            name="AerialStyleCheck"
            onclick="ChangeMapStyle()"
            >
            Aerial Style
    </div>
</body>
</html>
```

Listing 4.

Viewing this page in a browser now allows the user to see any of the 3 styles. Figure 4 shows a hybrid view of the map in the control.

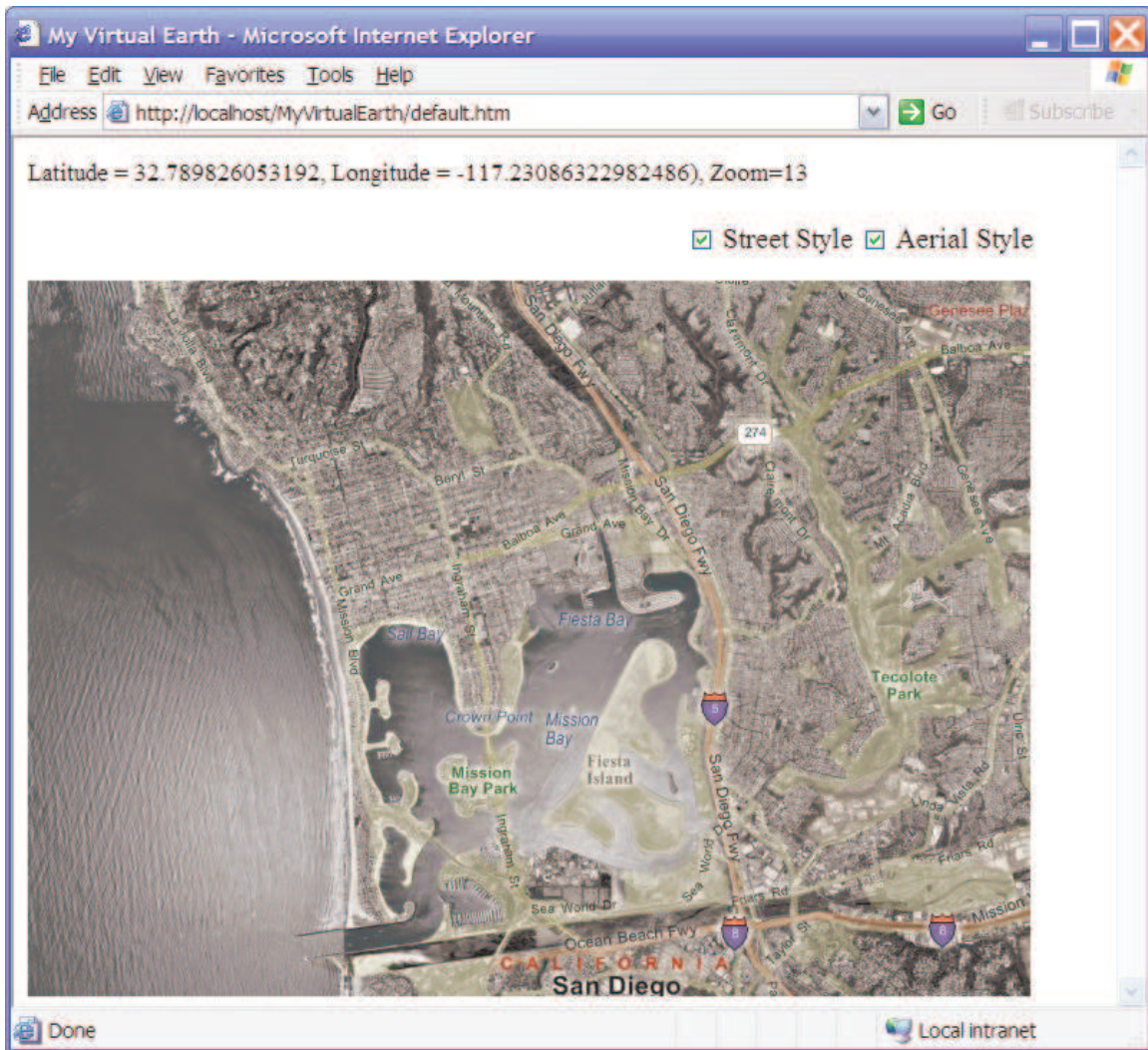


Figure 4

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)
```

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

In the following example (Listing 5) the onMouseClick 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>

  <script src="MapControl.js"></script>
  <script>
  var map = null;

  function OnPageLoad()
  {
    map = new VE_MapControl(32.69, -117.13, 12,
      'r', "absolute", 10, 100, 700, 500);

    document.body.appendChild(map.element);

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

    map.onEndZoom=function(e)
    {
      document.getElementById("info").innerHTML =
```

```

        'Latitude = ' +
        e.latitude +
        ', Longitude = ' +
        e.longitude +
        '), Zoom=' +
        e.zoomLevel;
    }

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

function ChangeMapStyle()
{
    var Aerial = document.getElementById("AerialStyleCheck");
    var Vector = document.getElementById("VectorStyleCheck");
    var s = 'r';
    if (Aerial.checked && Vector.checked)
    {
        s = 'h';
    }
    else if (Aerial.checked)
    {
        s = 'a';
    }
    map.SetMapStyle(s);
}

</script>
</head>

<body onLoad="OnPageLoad()">
<div id="info" style="font-size:10pt">
</div>
<div id="MapStyle"
    style="POSITION:absolute;LEFT:470px;TOP:60px">
    <INPUT id="VectorStyleCheck"
        type="checkbox"
        name="VectorStyleCheck"
        onclick="ChangeMapStyle()"
        checked="checked"
    >
    Street Style
    <INPUT id="AerialStyleCheck"
        type="checkbox"
        name="AerialStyleCheck"
        onclick="ChangeMapStyle()"
    >
    Aerial Style
</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 `onMouseClicked` 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 `onMouseClicked` function shown in Listing 5 can be changed to remove the previous pin as shown in Listing 6.

```
map.onMouseClicked=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.

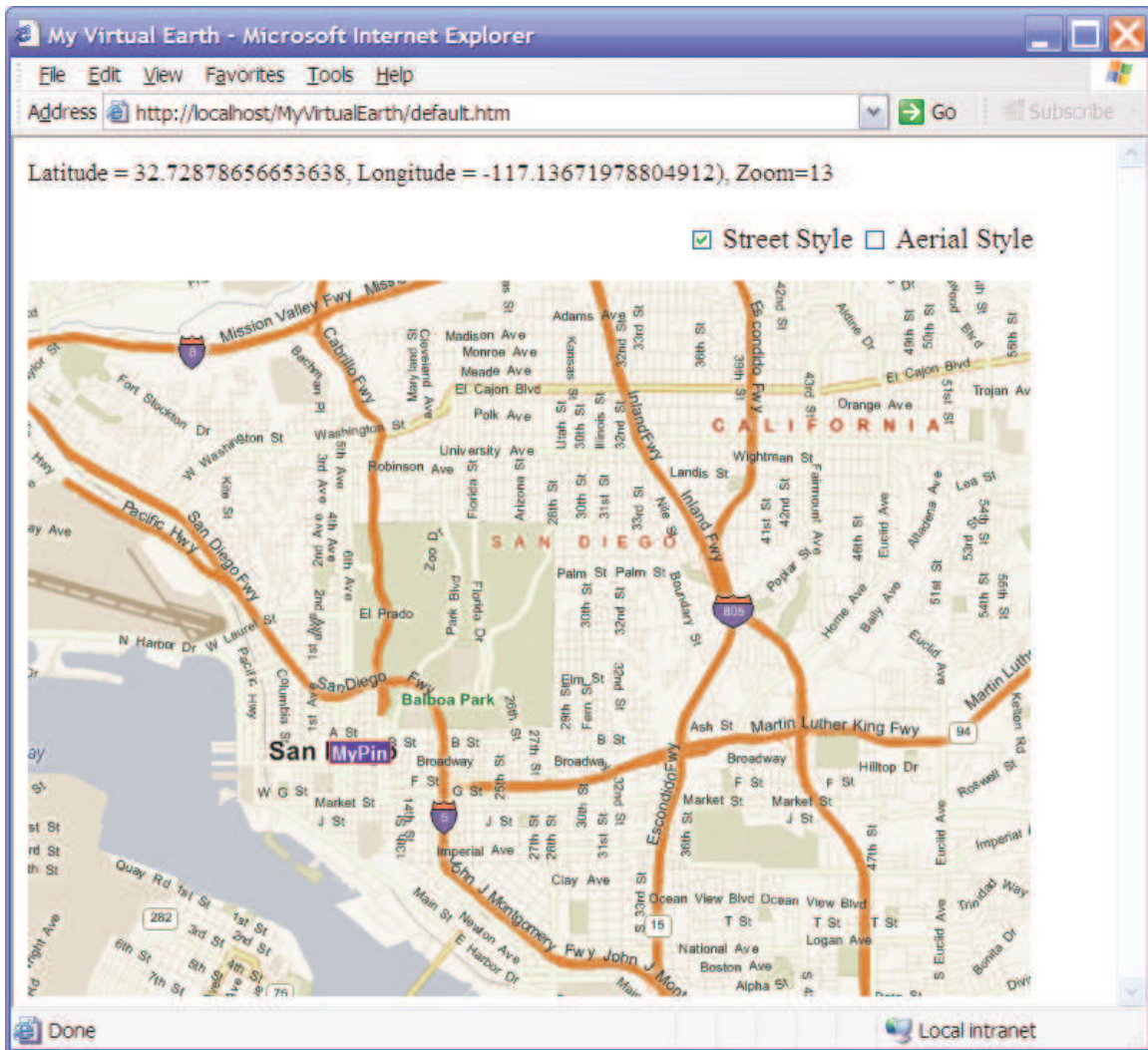


Figure 5

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:600"/>

<input type="button" value="Pan Left"
onclick="DoPanLeft()"
ID="PanLeftButton" NAME="PanLeftButton"
style="position:absolute;left:10px;top:630"/>

<input type="button" value="Pan Right"
onclick="DoPanRight()"
ID="PanRightButton" NAME="PanRightButton"
style="position:absolute;left:100px;top:630"/>

<input type="button" value="Pan Down"
onclick="DoPanDown()"
ID="PanDownButton" NAME="PanDownButton"
style="position:absolute;left:45px;top:660"/>

```

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:630"/>
```

```
<input type="button" value="Zoom Out" onclick="DoZoomOut()"
ID="ZoomOutButton" NAME="ZoomOutButton"
style="position:absolute;left:340px;top:630"/>
```

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>

<script src="MapControl.js"></script>
<script>
var map = null;

function OnPageLoad()
{
    map = new VE_MapControl(32.69, -117.13, 12,
        'r', "absolute", 10, 100, 700, 500);
    document.body.appendChild(map.element);

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

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

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

function ChangeMapStyle()
{
    var Aerial = document.getElementById("AerialStyleCheck");
    var Vector = document.getElementById("VectorStyleCheck");
    var s = 'r';
    if (Aerial.checked && Vector.checked)
    {
        s = 'h';
    }
    else if (Aerial.checked)
    {
        s = 'a';
    }
}

```

```

        }
        map.SetMapStyle(s);
    }

    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="font-size:10pt">
</div>
<div id="MapStyle"
    style="POSITION:absolute;LEFT:470px;TOP:60px">
    <INPUT id="VectorStyleCheck"
        type="checkbox"
        name="VectorStyleCheck"
        onclick="ChangeMapStyle()"
        checked="checked"
    >
    Street Style
    <INPUT id="AerialStyleCheck"
        type="checkbox"
        name="AerialStyleCheck"
        onclick="ChangeMapStyle()"
    >
    Aerial Style
</div>

<input type="button" value="Pan Up"
onclick="DoPanUp()"
ID="PanUpButton" NAME="PanUpButton"
style="position:absolute;left:60px;top:600"/>

<input type="button" value="Pan Left"
onclick="DoPanLeft()"
ID="PanLeftButton" NAME="PanLeftButton"

```

```
style="position:absolute;left:10px;top:630"/>
<input type="button" value="Pan Right"
onclick="DoPanRight()"
ID="PanRightButton" NAME="PanRightButton"
style="position:absolute;left:100px;top:630"/>
<input type="button" value="Pan Down"
onclick="DoPanDown()"
ID="PanDownButton" NAME="PanDownButton"
style="position:absolute;left:45px;top:660"/>
<input type="button" value="Zoom In" onclick="DoZoomIn()"
ID="ZoomInButton" NAME="ZoomInButton"
style="position:absolute;left:250px;top:630"/>
<input type="button" value="Zoom Out" onclick="DoZoomOut()"
ID="ZoomOutButton" NAME="ZoomOutButton"
style="position:absolute;left:340px;top:630"/>

</body>
</html>
```

Listing 7