As I presented in my previous post, the bulk addition allows us to add tons of shapes with a incredible speed.

Through this article, we'll see how this bulk addition works, and how in our case, we can optimize this part.

The facts:

In a recent analysis, Redfin's developers *(Redfin is an important real estate website)* have chosen to switch from VE to GMap for performance issues.

After discussing with them, we could learn that they don't use the traditional GMarker but instead they developed their own GOverlay-based element.

So, in a formal way and as a curious developer, I try to see what we can do to improve performance with VE and compare the result.

All of this is made to mail Microsoft about it and make some suggestions; it will be made in a couple of days.

How we can improve?

As a developer, we cannot change the compression of the script files, tiles imagery or some other elements used in VE.

I will send some mails to MS in the weeks to come to express this need in performance.

In this PoC case, it is about the analysis of the existing script, we will only improve the part concerning the Bulk Addition even if this solution is not perfect for every use (it corresponds to the expressed needs).

Under the hood?

Let analyze how the bulk add works. Firstly we used VEMap.AddShape() with a VEShape array as parameter so we use the bulk addition possibility.

```
VEMap.prototype.AddShape = function(b) {
    var a = this.m_velayermanager.VE_LayerManager.GetCollectionByIndex(0);
    a._mapGuid = this.GUID;
    a.AddShape(b)
};
```

We can see what is done through this code:

- The first line in the method get the base layer through the VELayerManager
- Then it sets the reference of the map by using the GUID.
- The shape addition is done in the VEShapeLayer class by calling VEShapeLayer.AddShape() with the same parameter (the VEShape array).

Through this class, is this method :

```
VEShapeLayer.prototype.AddShape = function(a, b) {
    if (a && typeof a.length != "undefined")
        this.AddShapes(a, b);
    else {
        VEValidator.ValidateObject(a, "_veshape", VEShape, "VEShape");

        if (a._shplayer != null || a.Primitives[0] == null)
            throw new VEException("AddShape",
                                  "err_invalidargument",
                                  L_ShpExist_text);

        a._shplayer = this;
        this.UpdateEntityAnnotation(a);
        this.AddAnnotation(a, b);
        if (a.GetVisibility())
            a.Show()
    }
};
```

The interesting part in this if that the parameters are checked and the type is checked to. In fact, the first line tests if it's an array which is our case.

We have to go deeper and see what's in VEShapeLayer.AddShapes().

```
VEShapeLayer.prototype.AddShapes = function(c) {
    VEValidator.ValidateObjectArray(c, "_veshapeArray",
                                    VEShape, "VEShape Array");
    var h = c.length, g = GetVEMapInstance(this._mapGuid);

    if (g && g.GetMapMode() == Msn.VE.MapActionMode.Mode3D)
        for (var a = 0; a < h; ++a)
            this.AddShape(c[a]);
    else {
        if (this._index == 0 && this.GetShapeCount() == 0)
            this.Show();

        var d = [], f = this.Annotations.length;

        for (var a = 0; a < f; a++)
            this.Annotations[a].ClearDomElements();

        if (!Msn.VE.Environment.IsIE50())
            for (var a = 0; a < f; a++) {
                var b = this.Annotations[a];
                if (b._isDrawn) {
                    if (b.IconUrl == null)
                        b.IconUrl = Msn.VE.API.Constants.iconurl;
                    d.push(b.ToHtml())
                }
            }

        for (var a = 0; a < h; ++a) {
            var b = c[a];
            b._shplayer = this;
            this.UpdateEntityAnnotation(b);
            this.AddAnnotation(b);

            if (b.IconUrl == null)
                b.IconUrl = Msn.VE.API.Constants.iconurl;

            d.push(b.ToHtml());
            b._isDrawn = true
        }

        var e = $ID(this.GetId());

        if (e)
            if (Msn.VE.Environment.IsIE50())
                e.innerHTML += d.join("");
        else
            e.innerHTML = d.join("")
    }

    this.Cluster()
};
```

After the tests on the parameter and after getting every information from the VEShape object (like title, description, IconUrl).

If we take a closer look, we can see that the rendering of this objects is made through a simple method called VEShape.ToHtml().

Time to Modify

That's precisely this method that we'll modify, we gonna modify to add a single DIV instead of many HTML elements as we don't need any of this in this PoC focused on the speed of addition.

Here is the code that we have to add to our page's script, it will change how the VE script works:

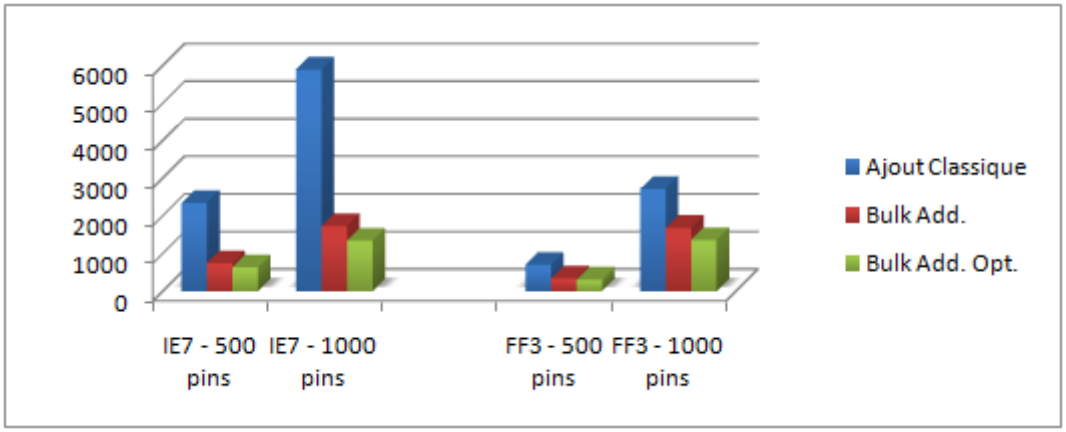
```
VEShape.prototype.ToHtml = function() {
  var a = [], b = GetVEMapInstance(this._shplayer._mapGuid);

  if (b && b.vemapcontrol) {
    var c = b.vemapcontrol.GetPushpinMapPixel(
      new Msn.VE.LatLong(this.Latitude, this.Longitude),
      b.vemapcontrol.GetZoomLevel()
    );

    var idPin = (this.Primitives[0].type == VEShapeType.Pushpin ?
                 this.Primitives[0].iid :
                 Msn.Drawing.GetLabelUID(this.Primitives[0].iid));

    a.push('<div id=');
    a.push(idPin);

    //STYLE
    a.push(' style="z-index:');
    a.push(this.GetZIndex());
    a.push(";width:25px;height:25px;background-image:url(");
    a.push(Msn.VE.API.Constants.iconurl);
    a.push(");position:absolute;left:");
    a.push(c.x - 25 / 2);
    a.push("px;top:");
    a.push(c.y - 25 / 2);

    //EVENTS
    a.push('px;" onclick="alert(\'');
    a.push(idPin.toString()); //
    a.push('\');" onmouseover="VEShowVEShapeERO(\'');
    a.push(this.Primitives[0].iid);
    a.push('\',\'');
    a.push(this._shplayer._mapGuid);
    a.push('\');" onmouseout="VEHideVEShapeERO(false);">');

    a.push("</div>");
  }
  if (debugView == true) {
    alert(a.join(""));
    debugView = false;
  }

  return a.join("")
};
```

Results:

Here is a simple table that exposes the results in milliseconds:

	Classical Add	Bulk Addition	Bulk Addition Opt.
IE7 - 500 pins	2350 ms	751 ms	640 ms
IE7 - 1000 pins	5900 ms	1740 ms	1350 ms
FF3 - 500 pins	710 ms	345 ms	310 ms
FF3 - 1000 pins	2725 ms	1690 ms	1360 ms

The optimized Bulked addition is faster than the two other operations.



The improvements:

- From the classical import:
  - IE : 300% better
  - FF: 110% better
- From the bulk addition:
  - IE : 20-25% better
  - FF: 15-20%

For information, 10k elements load in 55s with standard Bulk Add, 48s with this optimized bulk addition.

Important Notes:

By overriding this method that render the VEShape, you must keep in mind that it's not officially supported by Microsoft and so it can stop working in the next version.

So, it's really a PoC to prove that in some cases we can go deeper with the API and improve the performance to earn some precious milliseconds.

Adding to the fact that I did not test it with other VEShape object (as Polygon or Polyline) and some bad effects might happen.

Oh, and of course, you should keep in mind that Microsoft recommends not to add more than 500 shapes on the Map Control for stability and responsiveness issues. If you need to add more than 500 elements, you should implement clustering solution.

Conclusion:

This little "what's under this" into VE API allows us to observe some improving points and will be transformed in some suggestions and mails for MS.

I think that we can definitely improve the pin's customization possibility as I think on this point, GMap API is better (we can easily manage pin's shadow, offsets…).

I would like to thank Aurélien Verla which is a real JavaScript guru. He rapidly gives me a hint to improve this addition by using a string array and add it to the DOM in one shot.
This trick is already used in the API so it proves that Aurélien was right by saying this and it also justifies the API quality.