FingerEyes-Xr for HTML5 Tutorials - 06

그래픽 레이어 활용하기

2015년 3월 2일, 1차 배포

Good Solution
Good Partner

geoservice

㈜지오서비스

그래픽 레이어 활용하기

그래픽 레이어는 다양한 그래픽 요소를 추가하여 지도 상에 정보를 표현할 수 있는 기능을 제공하는 레이어입니다. 그래픽 레이어를 이용해 표현할 수 있는 그래픽 요소에는 포인트(Point), 폴리라인(Polyline), 폴리곤(Polygon), 사각형(Rectangle), 타원(Ellipse), 텍스트(Text)가 있습니다. 이 그래픽 요소 중 포인트는 마커(Marke) 개념으로써 다양한 심벌(Symbol)로 표현될 수 있는데, 사각형이나 원과 같은 도형이나 이미지로 표현될 수 있습니다. 이렇게 표현된 그래픽 요소에 대해서 코드를 통해 추가가 가능하고 마우스를 통해 편집이 가능합니다. 물론 마우스를 통한 추가 역시 가능합니다.

이 글은 이러한 그래픽 레이어에 대한 설명을 담고 있습니다. FingerEyes-Xr은 Flex 버전과 HTML5 버전이 존재하며 이 글은 HTML5에 대해 글입니다. FingerEyes-Xr for HTML5에 대한 소스 코드는 GitHub에서 다운로드 받을 수 있으며 URL은 https://github.com/FingerEyes-Xr/src 입니다. 아래의 화면은 FingerEyes-Xr for HTML5에 대한 GitHub 웹 화면입니다.

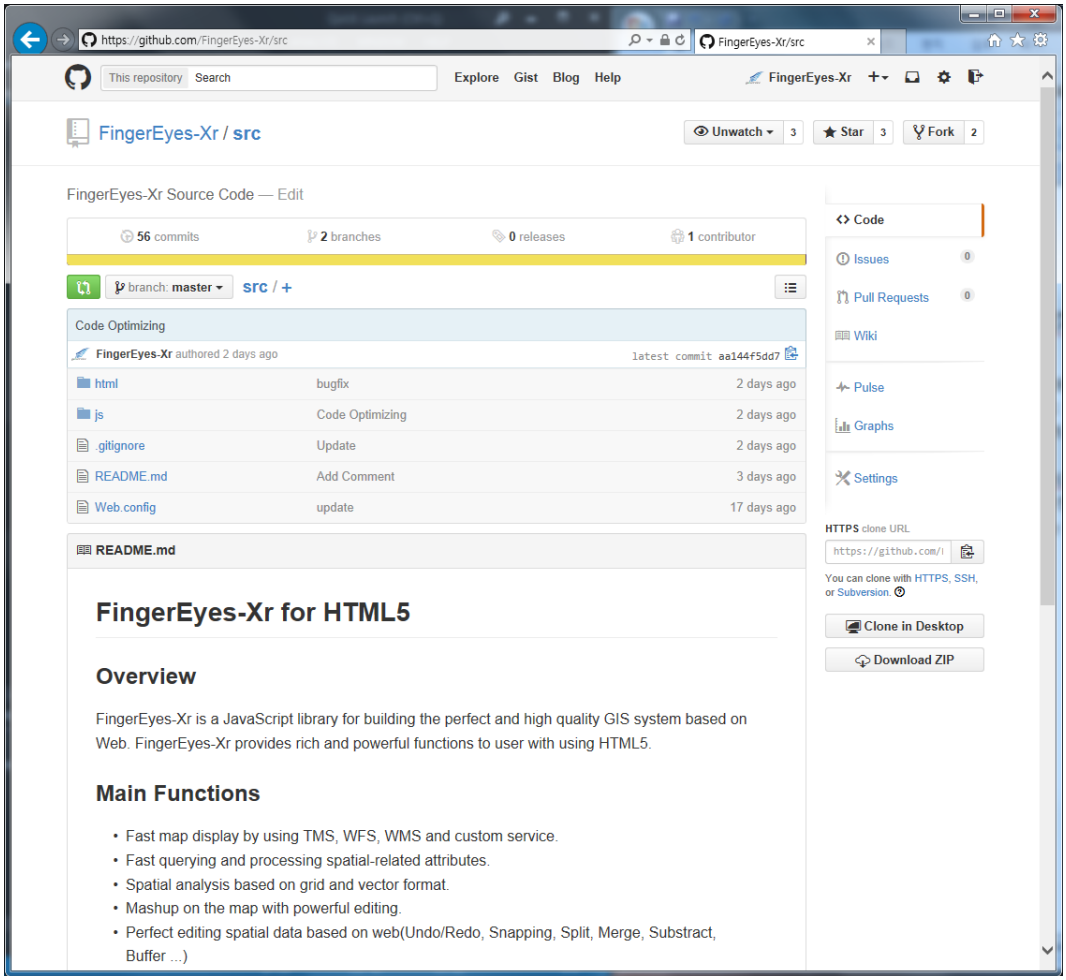

그림 1. FingerEyes-Xr for HTML에 대한 GitHub

필자는 이 글에 대한 예제 코드를 VisualStudio 2012를 이용하여 작성 하였습니다. 독자 여러분들이 어떤 툴을 사용하든 문제는 없겠으나 VisualStudio 2012에서 제공하는 JavaScript 디버깅과 기본적으로 제공하

는 웹서버(IIS)를 통해 웹 기반의 프로그래밍을 편리하고 효율적으로 진행할 수 있었기에 선택하였습니다. 참고로 웹 기반의 프로그래밍은 웹서버를 이용하여 URL 형태로 접근하지 않으면 올바르게 실행되지 않으므로 반드시 웹서버를 통해 URL 형태로 접근하시기 바랍니다.

이제 본격적으로 그래픽 레이어를 FingerEyes-Xr에서 사용해 보는 코드를 작성해 보겠습니다. 먼저 map.html 파일을 웹서버에서 접근할 수 있는 곳에 생성합니다. 필요하다면 map.html이 아닌 다른 파일명으로 생성 하여도 진행에 문제는 없습니다. 다만, URL을 통해 웹브라우저에서 실행할 수 있는 경로에 생성한다는 것이 중요합니다. 일반 html을 생성하고 다음처럼 입력합니다.

```
01 <html xmlns="http://www.w3.org/1999/xhtml">
02 <head>
03    <title>FingerEyes-Xr : ShapeMapLayer</title>
04 </head>
05 <body>
06
07 </body>
08 </html>
```

코드 1. 초기 map.html

위의 코드에서 3번 라인 밑에 다음 코드를 추가합니다.

```
01 <script src="http://www.geoservice.co.kr/z/Xr.min.1.0.js"></script>
02
03 <script type="text/javascript">
04
05 </script>
```

코드 2. 기본 Script

1번 코드에서 CDN(Content Delivery Network)을 사용하여 FingerEyes-Xr의 JavaScript 라이브러리에 대한 소스 코드를 추가 하고 있습니다. 그리고 4번의 빈 공백 부분에 앞으로 추가할 JavaScript 코드가 입력될 것입니다.

UI를 구성하기 위해 <body> 부분을 다음처럼 변경합니다.

```
01 <body>
02    <div id="mainLayout">
03       <div id="mapDiv"></div>
04       <div id="title">
05          FingerEyes-Xr for HTML5 :
06          <font color="#349bd6">Graphic Edit Example</font>
07          <br />
08          <input type="button" value="ZoomIn" onclick="onZoomInMap();">
09          <input type="button" value="ZoomOut" onclick="onZoomOutMap();">
10          <input type="button" value="MapView" onclick="onView();">
11          <input type="button" value="Edit" onclick="onEdit();">
12          <input type="button" value="New Point" onclick="onAddPoint();" />
13          <input type="button" value="New Polyline" onclick="onAddPolyline();"/>
14          <input type="button" value="New Polygon" onclick="onAddPolygon();" />
15          <input type="button" value="New Text" onclick="onAddText();" />
16          <input type="button" value="New Rect" onclick="onAddRect();" />
17          <input type="button" value="New Ellipse" onclick="onAddEllipse();" />
```

```
18              <input type="button" value="Undo" onclick="onUndo();" id="btnUndo" />
19              <input type="button" value="Redo" onclick="onRedo();" id="btnRedo" />
20          </div>
21      </div>
22  </body>
```

코드 3. 기본 UI 구성

특히 Id가 mapDiv인 DIV 요소에 맵(Map)이 표시될 것입니다. 일단 여기까지 작성하고 실행해 보면 다음과 같은 화면이 나타나는 것을 볼 수 있습니다.

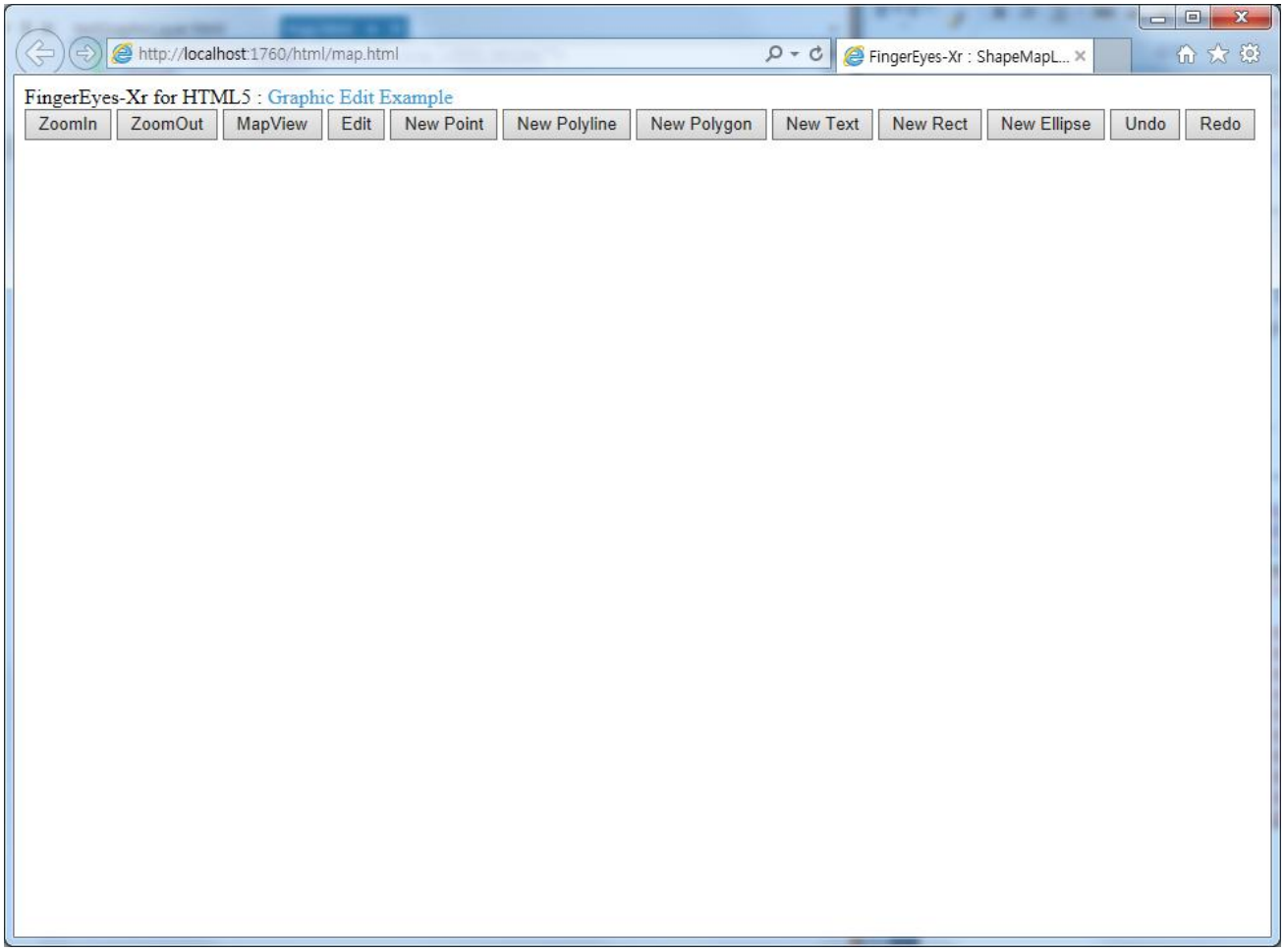

그림 2. 기본 UI 구성에 대한 실행 화면

Button이 12개가 있는데, **각각의 버튼은 아래 표와 같은 기능을 수행합니다.**

버 튼	기 능 설 명
ZoomIn	지도를 확대합니다.
ZoomOut	지도를 축소합니다.
MapView	지도 조작(확대, 축소, 이동) 모드
Edit	그래픽 요소 편집 모드.
New Point	편집 모드에서 새로운 포인트를 추가합니다.
New Polyline	편집 모드에서 새로운 폴리라인을 추가합니다.
New Polygon	편집 모드에서 새로운 폴리곤을 추가합니다.

New Text	편집 모드에서 새로운 텍스트를 추가합니다.
New Rect	편집 모드에서 새로운 사각형을 추가합니다.
New Ellipse	편집 모드에서 새로운 타원을 추가합니다.
Undo	편집 기능에 대한 되돌리기 기능
Redo	편집 기능에 대한 재실행 기능

표 1. 버튼 기능 설명

이제 만든 페이지에 스타일을 적용해 보도록 하겠습니다. <head> 바로 밑에 아래의 스타일 코드를 추가합니다.

```
<style>
    body
    {
        margin:0px;
        padding:0px;
    }

    #mainLayout
    {
        width:100%;
        height:100%;
        border:none;
    }

    #mapDiv {
        top:0px;
        left:0px;
        position:relative;
        width:100%;
        height:100%;
        border:none;
        overflow:hidden;
    }

    #title {
        top:12px;
        left:12px;
        padding: 12px;
        position:absolute;
        background:rgba(0,0,0,0.7);
        border:none;
        overflow:auto;
        border-radius: 12px;
        font-size: 24px;
        color: #ffffff;
        font-family: "Arial";
    }
</style>
```

코드 4. UI에 대한 스타일 적용

스타일이 적용된 페이지는 다음과 같습니다.

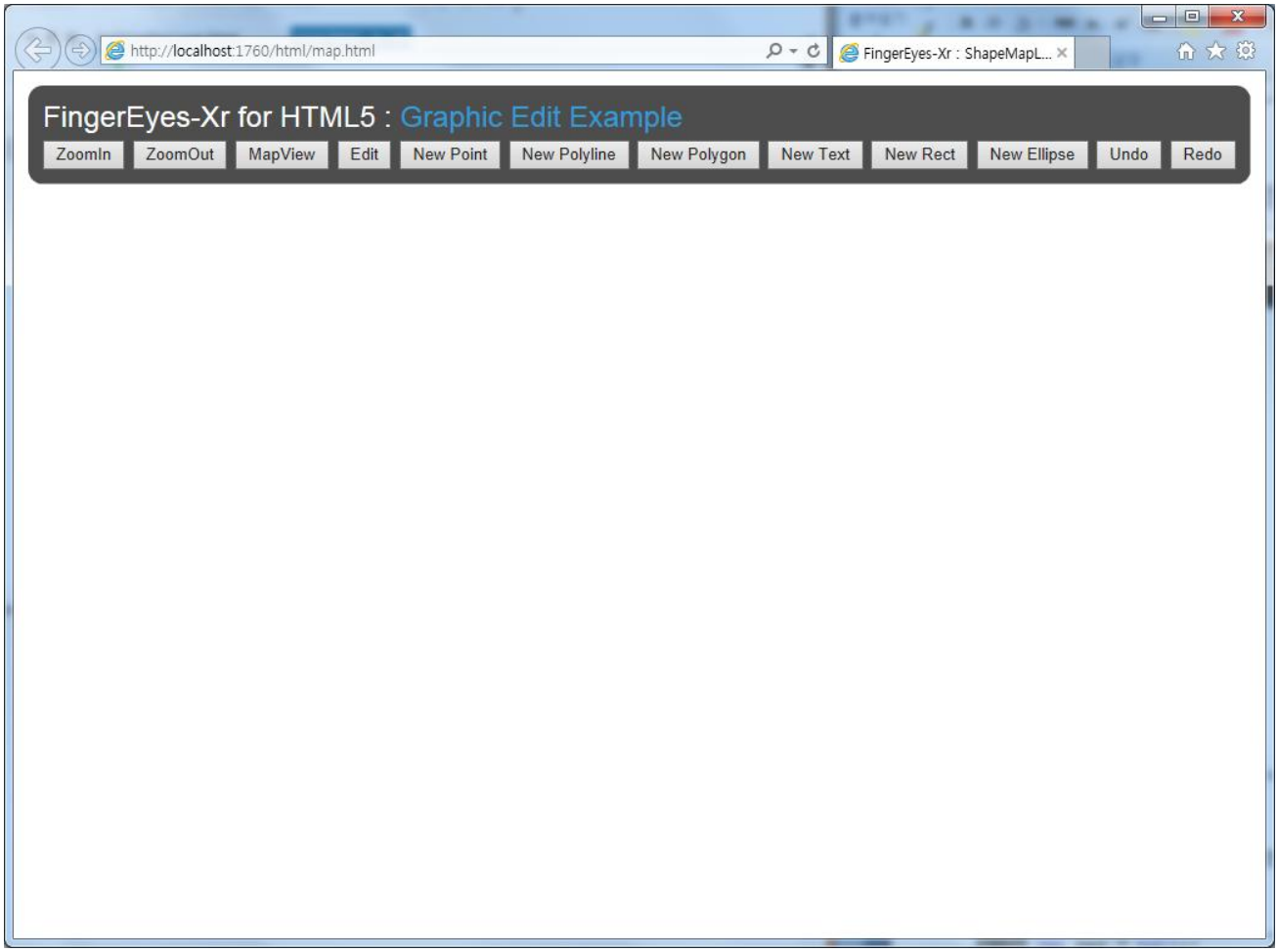

그림 3. 스타일이 적용된 UI

이제 UI와 스타일 적용이 완료되었으므로, 이제 코드를 작성해 보겠습니다. **코드 2. 기본 Script**에서 4번 줄에 다음 코드를 추가합니다.

```
01  var map = null;
```

코드 5. DIV와 연결될 지도 객체 정의

이 map 변수는 지도 객체에 대한 참조로 사용됩니다. 이제 <body>에 대한 onload 이벤트를 load() 함수로 지정합니다.

```
01  <body onload="load()">
```

코드 6. body 요소의 onload 이벤트 지정

그리고 load 함수를 다음처럼 추가합니다.

```
01  function load() {
02      map = new Xr.Map("mapDiv", {});
03
04
05
06  }
```

코드 7. body의 onload 이벤트 함수

2번 코드는 id가 mapDiv인 DIV를 이용하여 지도 객체를 생성하고 이를 앞서 정의해 둔 map 객체에 저장하고 있습니다.

그래픽 레이어를 추가하기에 앞서 참조가 되는 항공영상 레이어와 수치지도 레이어를 추가하겠습니다. 마우스를 이용해 그래픽 요소를 그릴 때 이러한 레이어를 참고할 수 있으며, 특히 수치지도 레이어를 대상으로 스냅핑 기능을 이용할 수 있습니다. 보다 자세한 내용은 해당 기능에 대한 코드를 작성하면서 다시 설명 드리겠습니다. 아래의 코드처럼 항공영상 레이어와 수치지도 레이어를 추가하는 코드를 load 함수 구현부 끝에 추가합니다.

```
01  var lyr = new Xr.layers.TileMapLayer("basemap",
02      {
03          proxy: "http://222.237.78.208:8080/Xr",
04          url: "http://222.237.78.208:8080/yp_tiles/a",
05          ext: "jpg"
06      });
07
08  var shpLyr = new Xr.layers.ShapeMapLayer("jibun",
09      { url: "http://222.237.78.208:8080/Xr?layerName=JIBUN" });
10
11  shpLyr.visibility().visibleByScale(true);
12  shpLyr.visibility().fromScale(0);
13  shpLyr.visibility().toScale(1001);
14
15  var theme = shpLyr.theme();
16  var pen = theme.penSymbol();
17  var brush = theme.brushSymbol();
18
19  pen.color('#ffff00');
20  pen.width(2);
21
22  brush.color('#ff0000');
23  brush.opacity(0.0);
```

코드 8. 항공영상지도를 위한 타일맵 레이어와 수치지도 레이어 생성

위 코드에 대한 자세한 내용은 각 레이어를 설명하는 다른 튜토리얼을 참고하시기 바랍니다. 다음은 실제 그래픽 레이어에 대한 코드입니다. 아래의 코드를 load 함수 구현부의 끝에 추가합니다.

```
01  var graphicLyr = new Xr.layers.GraphicLayer("grp");
```

코드 9. 그래픽 레이어의 생성

그래픽 레이어는 GraphicLayer 클래스를 통해 생성할 수 있습니다. 이 클래스의 생성자는 하나의 인자를 갖습니다. 이 인자는 레이어의 이름으로 고유한 식별자입니다. 이 레이어의 이름으로 레이어 관리자로부터 레이어를 찾아 참조할 수 있습니다. 위의 코드는 이렇게 그래픽 레이어를 생성해서 graphicLyr라는 변수에 저장해 놓습니다.

이렇게 총 3개의 레이어 객체를 생성했으며 아래의 코드를 통해 이 레이어 객체를 레이어 관리자에 추가합니다. 이 코드를 load 함수 구현부의 끝에 추가합니다.

```
01  var lm = map.layers();
02
03  lm.add(lyr);
04  lm.add(shpLyr);
05  lm.add(graphicLyr);
```

코드 10. 레이어 들의 추가

이제 실행을 하면 앞서 추가한 레이어 들로 구성된 지도가 화면에 표시되어야 할 것입니다. 그러나 한가지가 더 필요합니다. 즉, 해당 레이어 들이 모두 추가된 후에 화면 상에 표시될 지도의 좌표와 축척을 지정해야 합니다. 아래의 코드는 이처럼 해당 레이어 들의 추가가 완료 되었을 때 호출되는 이벤트를 통해 화면에 표시될 지도와 좌표와 축척값을 지정하는 코드입니다. 이 코드를 load 함수의 구현부 끝에 추가합니다.

```
01  map.onLayersAllReady(function () {
02      var mbr = shpLyr.MBR();
03      var cm = map.coordMapper();
04
05      cm.zoomByMBR(mbr);
06      cm.zoomByMapScale(766);
07
08      map.update();
09  });
```

코드 11. 레이어 들의 추가가 완료되었을 때 지도의 중심 좌표와 축척을 지정하는 코드

위의 코드를 살펴보면 2번 코드를 통해 앞서 추가한 수치지도 레이어에 대한 참조를 갖는 변수인 shpLyr에서 해당 레이어의 전체 MBR을 가져 오고 있습니다. 3번 코드는 CoordMapper 객체를 얻어와 cm이라는 변수에 저장하고 있는데, CoordMapper는 지도의 이동, 회전, 확대, 축소 기능을 제공하고 지도 좌표와 화면 좌표 간의 변환 기능을 제공합니다. 그리고 5번 코드는 2번 코드에서 얻은 MBR을 통해 화면에 표시될 영역을 지정하고 있습니다. 그리고 6번 코드는 축척을 1 : 766으로 지정하기 위해 이 축척의 분모에 해당하는 값인 766를 인자로 하여 CoordMapper의 zoomByMapScale 함수를 호출하고 있습니다. 그리고 8번 코드는 실제 지도를 그리라는 함수를 호출하고 있습니다.

실행하면 아래의 화면처럼 항공영상과 수치지도가 표시되고 있는 것을 볼 수 있습니다. 또한 그래픽 레이어는 아직 그래픽 요소를 추가한 것이 없으므로 아무것도 표시되고 있지 않습니다.

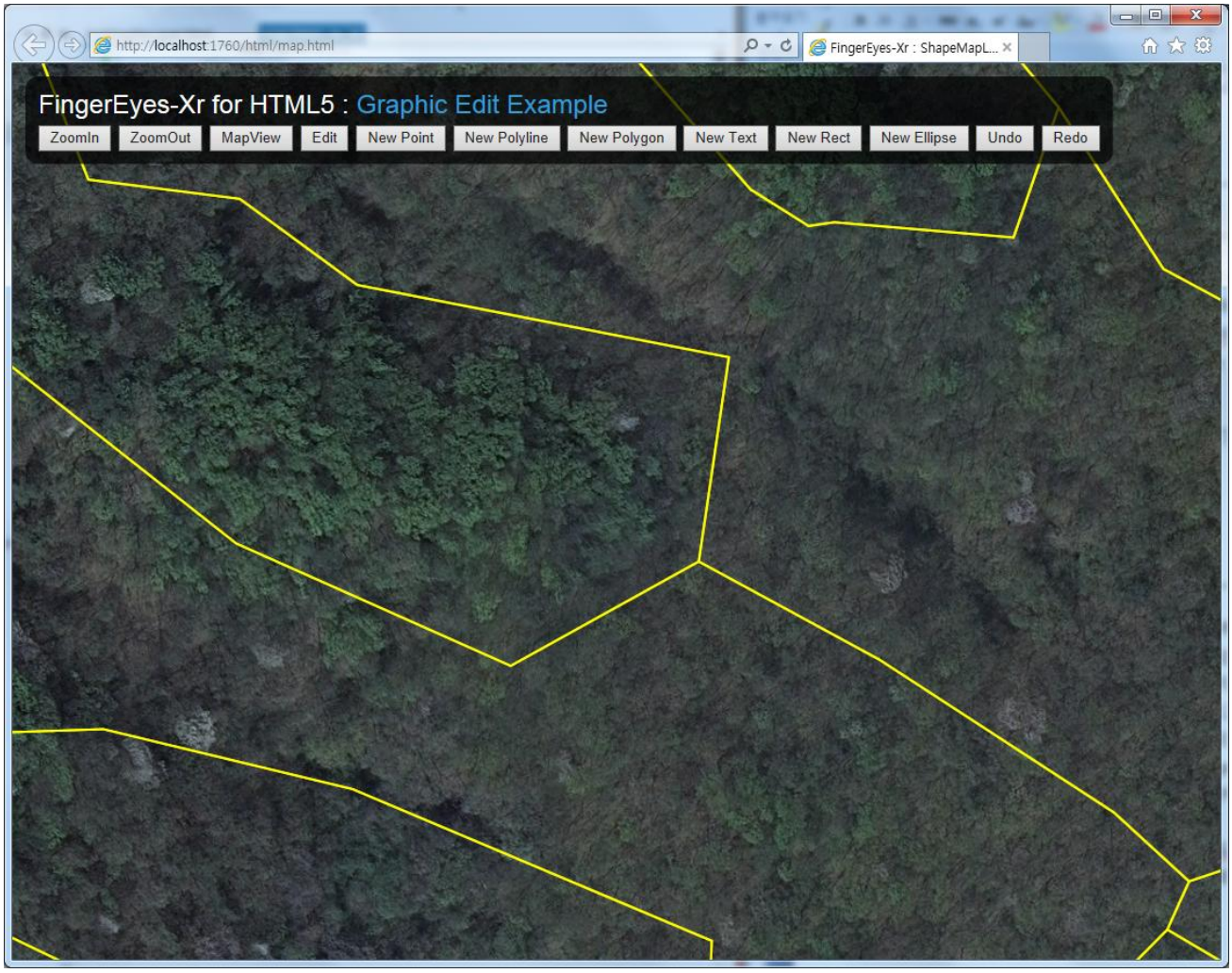

그림 4. 항공영상을 위한 타일맵 레이어와 수치지도 레이어 그리고 그래픽 레이어 표시 화면

이제 각 버튼들에 대한 코드를 작성해 보겠습니다. 먼저 지도를 확대하는 ZoomIn 버튼과 ZoomOut 버튼에 대한 클릭 이벤트 함수들은 다음과 같습니다.

```
01  function onZoomInMap() {
02      var cm = map.coordMapper();
03      cm.zoomByMapScale(cm.mapScale() * 0.5);
04      map.update();
05  }
06
07  function onZoomOutMap() {
08      var cm = map.coordMapper();
09      cm.zoomByMapScale(cm.mapScale() * 2);
10      map.update();
11  }
```

코드 12. 확대, 축소 버튼에 대한 클릭 이벤트

위의 코드에 대한 기본적인 내용은, 먼저 지도 확대를 위한 CoordMapper 객체의 zoomByMapScale 함수를 통해 지도를 확대하고 축소할 수 있다는 것입니다. 현재 지도 축척값에 0.5를 곱하여 이 함수에 지정하면 지도가 확대될 것이고 2를 곱하여 이 함수에 지정하면 지도가 축소될 것입니다. 현재의 지도 축척을 얻기 위해서는 CoordMapper 객체의 mapScale 함수를 통해 얻을 수 있습니다. 지도의 축척값을 변경

한 후에는 update 함수를 호출하여 지도 그리기를 실행합니다.

다음은 MapView 버튼과 Edit 버튼에 대한 클릭 이벤트 함수입니다.

```
01  function onEdit() {
02      map.userMode(Xr.UserModeEnum.EDIT);
03  }
04
05  function onView() {
06      map.userMode(Xr.UserModeEnum.VIEW);
07  }
```

코드 13. MapView와 Edit 버튼의 클릭 에빈트

Edit 버튼이 클릭되면 현재 마우스에 대한 사용자 모드를 Xr.UserModeEnum.EDIT로 설정합니다. 이 모드로 설정되면 마우스 조작은 그래픽 요소에 대한 편집으로 작동합니다. 그래픽 요소를 새롭게 추가하는 등의 편집을 실행하기에 앞서 반드시 이처럼 사용자 모드를 편집 모드로 전환해야 합니다. 그리고 MapView 버튼은 편집 모드에서 다시 지도 조작 모드로 전환하는 기능을 수행합니다. 이를 위해서 userMode라는 함수를 통해 Xr.UserModeEnum.VIEW 인자 값으로 사용자 모드를 전환하고 있습니다.

이제 New Point 버튼에 대한 클릭 이벤트에 대한 함수를 정의해 보겠습니다. 이 함수의 정의에 앞서 새롭게 추가되는 그래픽 요소에 대한 ID 값 지정을 위해 전역 변수로 newId 값을 정의해 놓겠습니다. 아래의 코드를 전역 위치에 추가합니다.

```
01  var newId = 0;
```

코드 14. 새로운 그래픽 요소를 추가할 때 사용되는 ID 값 변수

그리고 New Point 버튼에 대한 클릭 이벤트 함수의 코드를 아래처럼 추가합니다.

```
01  function onAddPoint() {
02      if (!map.edit().newPoint(newId++)) {
03          alert("Not Edit Mode ");
04      }
05  }
```

코드 15. New Point 버튼의 클릭 이벤트

위의 코드를 자세히 설명하면, 지도를 참조하는 map 객체의 edit() 함수를 통해 편집 관리자(EditManager) 객체를 얻습니다. 그리고 이 객체의 newPoint 함수를 호출하여 새로운 포인트 그래픽 요소를 추가할 수 있는 상태로 전환합니다. 이 상태로 전환되면 마우스 클릭 등을 통해 새로운 포인트가 추가될 수 있습니다. 이 newPoint 함수는 1개의 인자를 갖는데, 이 인자는 새롭게 추가될 그래픽 요소의 ID 값을 지정하는 것입니다. 앞서 전역 변수로 newId 값을 정의해 놓았고 이 값을 증가시켜 ID 값으로 사용하고 있습니다. newPoint와 같은 함수는 현재 편집 모드가 아닐 경우이거나 그래픽 요소의 ID 값이 이미 사용 중일 때 false를 반환합니다. 이처럼 새로운 그래픽 요소를 추가하는 newPoint와 같은 함수들은 모두 동일한 개념을 갖고 있습니다.

이제 실행하고 편집 버튼을 누른 후 새로운 포인트를 추가하고 싶겠으나, 매우 중요한 한가지가 빠져 있

습니다. 바로 편집에 있어서 편집 대상이 되는 그래픽 레이어를 지정해 놓지 않았습니다. 편집 대상이 되는 그래픽 레이어를 지정하기 위해서 **코드 10. 레이어 들의 추가**에서 5번 코드 밑에 다음 코드를 추가합니다.

```
01  var lm = map.layers();
02
03  lm.add(lyr);
04  lm.add(shpLyr);
05  lm.add(graphicLyr);
06
07  map.edit().targetGraphicLayer(graphicLyr);
```

코드 16. 편집 대상 레이어의 지정

이제 실행하고 Edit 버튼을 눌러 편집 모드로 전환 한 뒤 New Point 버튼을 실행하여 지도 화면 상에 표인트를 추가하고자 하는 위치에 마우스를 클릭하면 작은 사각형 모양의 포인트가 표시되는 것을 볼 수 있습니다. 아래의 화면은 하나의 새로운 포인트 요소를 추가해 본 화면입니다.

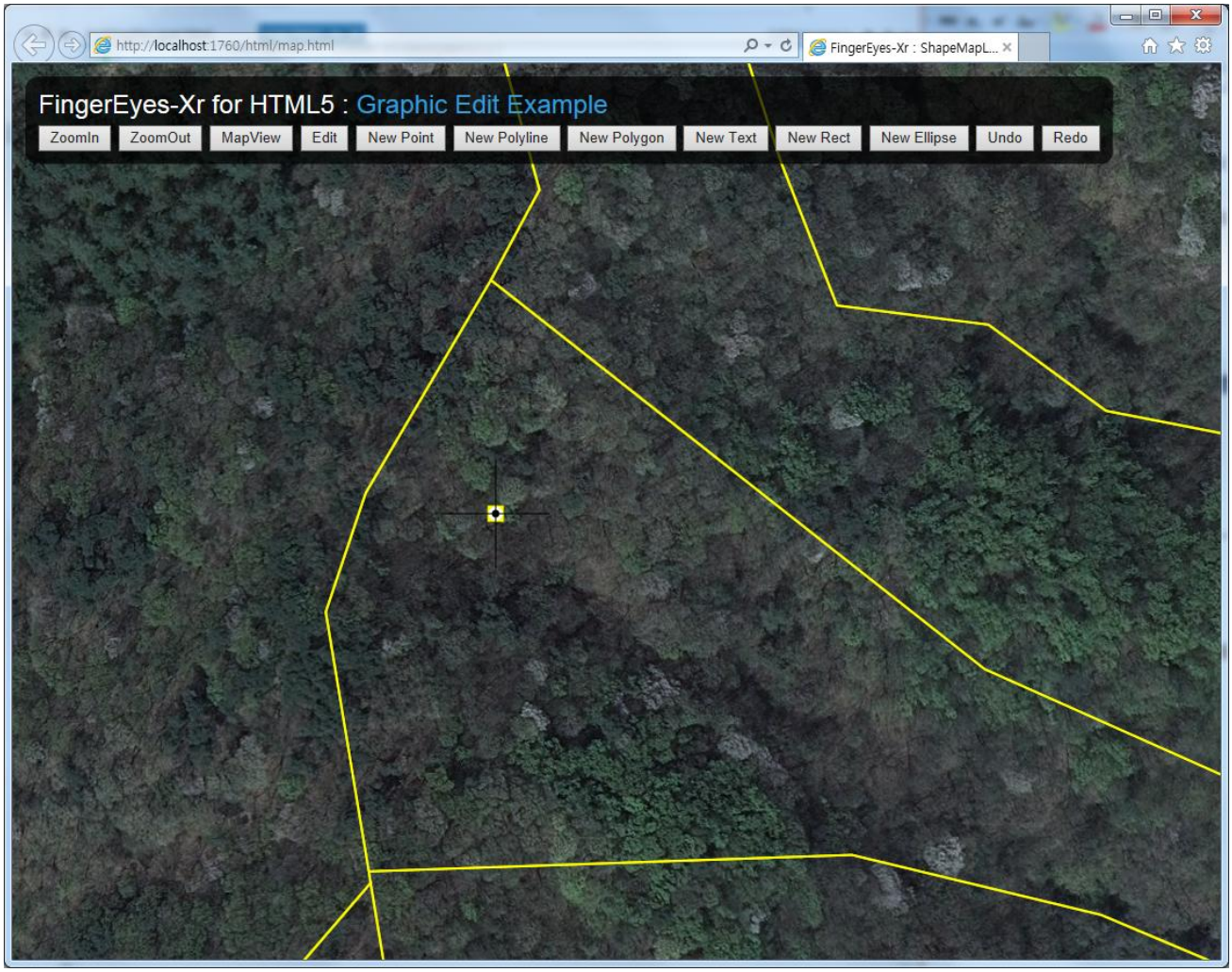

그림 5. 새로운 포인트 추가

다음은 New Polyline 버튼에 대한 클릭 이벤트 함수입니다.

```
01  function onAddPolyline() {
02      if (!map.edit().newPolyline(newId++)) {
03          alert("Not Edit Mode ");
04      }
05  }
```

코드 17. 새로운 폴리라인 그래픽 요소 추가

New Point 버튼의 코드와 달라진 부분은 newPoint 함수 대신에 newPolyline 함수가 사용되었다는 것 뿐입니다. 이처럼 New Polygon, New Text, New Rect, New Ellipse 버튼에 대한 함수도 유사하며 각 버튼에 대한 클릭 이벤트 함수는 다음과 같습니다.

```
01  function onAddPolygon() {
02      if (!map.edit().newPolygon(newId++)) {
03          alert("Not Edit Mode ");
04      }
05  }
06
07  function onAddText() {
08      if (!map.edit().newText(newId++)) {
09          alert("Not Edit Mode ");
10      }
11  }
12
13  function onAddRect() {
14      if (!map.edit().newRectangle(newId++)) {
15          alert("Not Edit Mode ");
16      }
17  }
18
19  function onAddEllipse() {
20      if (!map.edit().newEllipse(newId++)) {
21          alert("Not Edit Mode ");
22      }
23  }
```

코드 18. 새로운 폴리곤, 텍스트, 사각형, 타원 그래픽 요소 추가

이제 실행하여 다양한 그래픽 요소를 마우스를 통해 생성하고, 이미 생성된 그래픽 요소를 선택하여 재편집할 수 있습니다. 아래의 화면은 다양한 그래픽 요소를 추가하고 편집한 화면 중 하나입니다.

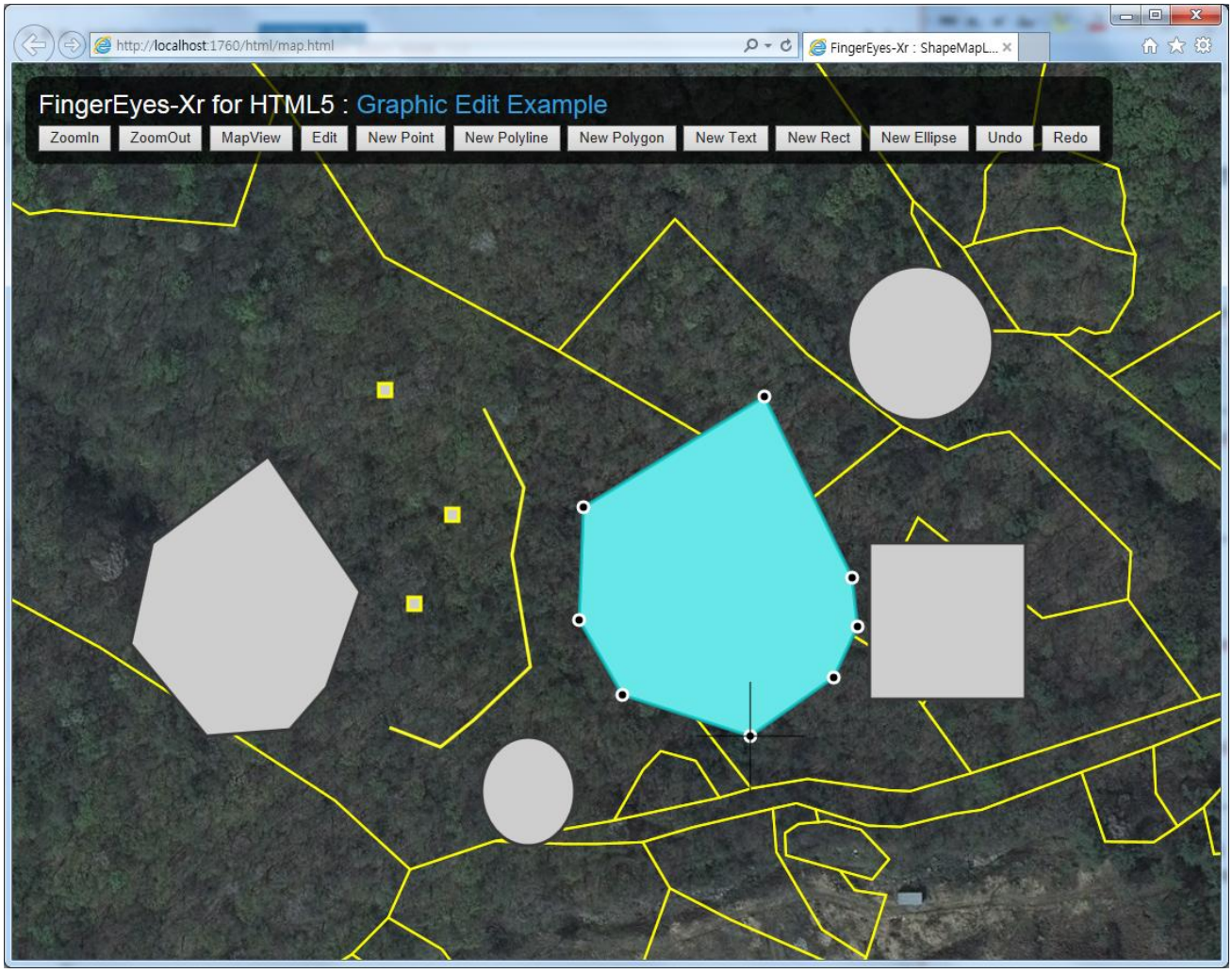

그림 6. 다양한 그래픽 요소 추가 및 편집

특히, 폴리라인과 폴리곤에 있어서 새로운 정점(Vertex)를 추가하거나 정점을 삭제 하는 기능이 있습니다. 이 두 도형에 대해 새로운 정점을 추가하기 위해서는 먼저 선택하고자 하는 도형을 선택한 상태에서 새로운 정점을 추가하고자 하는 위치에서 **Ctrl 키를 누르고 클릭**하면 됩니다. 그리고 기존의 정점을 삭제하기 위해서는 해당 도형을 선택하고 삭제하고자 하는 **정점의 위치에서 마우스 버튼을 누르고 Del 키**를 입력 하면 됩니다.

이제 Undo와 Redo 버튼에 대한 클릭 이벤트 함수에 대해 살펴볼 차례입니다. Undo와 Redo 버튼은 편집에 대해 되돌리기 기능과 다시실행에 대한 기능입니다. 이 버튼은 이러한 되돌리기와 다시실행이 가능한 상태에서만 활성화되어야 하고 그렇지 않은 경우에는 비활성화 된 상태여야 합니다. 그리고 도형의 편집 시에 되돌리기와 다시실행이 가능한 상태일 경우 이 버튼들은 활성화되어야 합니다. 편집 상태에 대한 이러한 UI의 반응을 위해 아래의 코드를 load 함수 구현부의 마지막 부분에 추가합니다.

```
01  btnUndo.disabled = true;
02  btnRedo.disabled = true;
03
04  map.addEventListener(Xr.Events.UndoStateChanged, onUndoStateChanged);
05  map.addEventListener(Xr.Events.RedoStateChanged, onRedoStateChanged);
```

코드 19. Undo와 Redo에 대한 상태 정보 및 이벤트

위의 코드 중, 1번과 2번은 각각 Undo 버튼과 Redo 버튼을 비활성화 시켜 놓는 것입니다. load 함수가 실행될 시점에서는 되돌리기와 재실행 될 내용이 없으므로 비활성화되어 있는 것이 자연스럽습니다. 그리고 4번과 5번 코드는 각각 되돌리기와 다시 실행 될 내용에 대한 상태가 변경될 때마다 실행되는 콜백(Callback) 함수를 지정하는 것입니다. 이 2개의 콜백함수는 아래와 같습니다.

```
01  function onUndoStateChanged(e) {
02      btnUndo.disabled = e.disabled;
03  }
04
05  function onRedoStateChanged(e) {
06      btnRedo.disabled = e.disabled;
07  }
```

코드 20. Undo와 Redo에 대한 상태 정보 변경 이벤트

이러한 되돌리기와 다시 실행 될 내용에 대한 상태가 변경될 때 마다 호출되는 콜백 함수의 인자는 disabled라는 프로퍼티를 제공하고 있습니다. 이 프로퍼티를 각 UI의 disabled에 반영해 주면 자연스러운 UI 연동이 이루어 집니다.

이제 Undo 버튼과 Redo 버튼에 대한 이벤트 함수의 코드를 살펴보겠습니다. 이 두 함수는 다음과 같습니다.

```
01  function onUndo() {
02      map.edit().history().undo();
03  }
04
05  function onRedo() {
06      map.edit().history().redo();
07  }
```

코드 21. Undo와 Redo 버튼에 대한 클릭 이벤트

이제 실행해 보고 그래픽 요소를 추가하고 추가된 도형을 선택하여 재편집해 보시기 바랍니다. 그리고 Undo와 Redo 버튼을 클릭하여 되돌리기와 다시실행 기능을 실행해 보시기 바랍니다.

편집에 대한 마지막 기능으로써 스냅핑(Snapping) 기능에 대해 살펴보겠습니다. 스냅핑 기능은 편집 시 정점(Vertex)를 지정할 때 참조가 되는 다른 레이어를 구성하는 정점이나 선분(Segment 또는 Edge)에 정확히 들러 붙게 하는 기능입니다. 이러한 스냅핑 기능을 위한 코드는 아래와 같으며 이 코드를 **코드16. 편집 대상 레이어의 지정**의 코드 부분에서 마지막에 추가합니다.

```
01  map.edit().snap().add(shpLyr);
02  map.edit().snap().add(graphicLyr);
03  map.edit().snap().vertexSnapMode(true);
04  map.edit().snap().edgeSnapMode(true);
```

코드 22. 스냅핑 기능 설정

스냅핑은 그 대상을 여러 개로 설정할 수 있습니다. 위의 코드에서도 2개의 레이어에 스냅핑 대상을 잡

고 있습니다. 특히 2번은 graphicLayer를 편집 대상 레이어로 했을 뿐만 아니라 스냅핑 대상으로도 잡고 있습니다. 3번 코드는 스냅핑을 정점(Vertex)으로 할 것인지의 여부를 지정하는 것이며, 4번 코드는 선분(Segment 또는 Edge)를 대상으로 할 것인지의 여부를 지정하는 것입니다.

이제 편집 기능을 실행 해 보면 수치지도를 구성하는 좌표로 마우스를 가까이 이동하면 빨간색 십자 표시와 함께 정점의 위치가 붙는 것을 볼 수 있습니다. 수치지도를 구성하는 선분에 마우스를 가까이 이동하면 노란색 십자 표시가 나타나며 역시 편집을 위한 커서 위치가 스냅핑 된 위치로 붙게 됩니다.

그림 7. 스냅핑 기능에 대한 빨간색 십자 표시

지금까지 그래픽 레이어에서 편집 기능을 살펴보았습니다. 이러한 그래픽 레이어의 편집 기능은 공간 데이터의 편집 기능에도 그대로 활용되는 기본 기능입니다.

지금까지 작성한 전체 소스코드는 아래와 같습니다.

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <style>
        body
        {
```

```
            margin:0px;
            padding:0px;
        }

        #mainLayout
        {
            width:100%;
            height:100%;
            border:none;
        }

        #mapDiv {
            top:0px;
            left:0px;
            position:relative;
            width:100%;
            height:100%;
            border:none;
            overflow:hidden;
        }

        #title {
            top:12px;
            left:12px;
            padding: 12px;
            position:absolute;
            background:rgba(0,0,0,0.7);
            border:none;
            overflow:auto;
            border-radius: 12px;
            font-size: 24px;
            color: #ffffff;
            font-family: "Arial";
        }
    </style>

    <title>FingerEyes-Xr : ShapeMapLayer</title>

    <script src="http://www.geoservice.co.kr/z/Xr.min.1.0.js"></script>
    <script type="text/javascript">

        var map = null;

        function load() {
            map = new Xr.Map("mapDiv", {});

            var lyr = new Xr.layers.TileMapLayer("basemap",
                {
                    proxy: "http://222.237.78.208:8080/Xr",
                    url: "http://222.237.78.208:8080/yp_tiles/a",
                    ext: "jpg"
                });

            var shpLyr = new Xr.layers.ShapeMapLayer("jibun",
                { url: "http://222.237.78.208:8080/Xr?layerName=JIBUN" });

            shpLyr.visibility().visibleByScale(true);
            shpLyr.visibility().fromScale(0);
            shpLyr.visibility().toScale(1001);

            var theme = shpLyr.theme();
            var pen = theme.penSymbol();
            var brush = theme.brushSymbol();
```

```
            pen.color('#ffff00');
            pen.width(2);

            brush.color('#ff0000');
            brush.opacity(0.0);

            var graphicLyr = new Xr.layers.GraphicLayer("grp");
            var lm = map.layers();

            lm.add(lyr);
            lm.add(shpLyr);
            lm.add(graphicLyr);

            map.edit().targetGraphicLayer(graphicLyr);

            map.edit().snap().add(shpLyr);
            map.edit().snap().add(graphicLyr);
            map.edit().snap().vertexSnapMode(true);
            map.edit().snap().edgeSnapMode(true);

            map.onLayersAllReady(function () {
                var mbr = shpLyr.MBR();
                var cm = map.coordMapper();

                cm.zoomByMBR(mbr);
                cm.zoomByMapScale(766);

                map.update();
            });

            btnUndo.disabled = true;
            btnRedo.disabled = true;

            map.addEventListener(Xr.Events.UndoStateChanged, onUndoStateChanged);
            map.addEventListener(Xr.Events.RedoStateChanged, onRedoStateChanged);
        }

        function onZoomInMap() {
            var cm = map.coordMapper();
            cm.zoomByMapScale(cm.mapScale() * 0.5);
            map.update();
        }

        function onZoomOutMap() {
            var cm = map.coordMapper();
            cm.zoomByMapScale(cm.mapScale() * 2);
            map.update();
        }

        function onEdit() {
            map.userMode(Xr.UserModeEnum.EDIT);
        }

        function onView() {
            map.userMode(Xr.UserModeEnum.VIEW);
        }

        var newId = 0;

        function onAddPoint() {
            if (!map.edit().newPoint(newId++)) {
                alert("Not Edit Mode ");
            }
        }
```

```
        function onAddPolyline() {
            if (!map.edit().newPolyline(newId++)) {
                alert("Not Edit Mode ");
            }
        }

        function onAddPolygon() {
            if (!map.edit().newPolygon(newId++)) {
                alert("Not Edit Mode ");
            }
        }

        function onAddText() {
            if (!map.edit().newText(newId++)) {
                alert("Not Edit Mode ");
            }
        }

        function onAddRect() {
            if (!map.edit().newRectangle(newId++)) {
                alert("Not Edit Mode ");
            }
        }

        function onAddEllipse() {
            if (!map.edit().newEllipse(newId++)) {
                alert("Not Edit Mode ");
            }
        }

        function onUndoStateChanged(e) {
            btnUndo.disabled = e.disabled;
        }

        function onRedoStateChanged(e) {
            btnRedo.disabled = e.disabled;
        }

        function onUndo() {
            map.edit().history().undo();
        }

        function onRedo() {
            map.edit().history().redo();
        }

    </script>

</head>

<body onload="load()">
    <div id="mainLayout">
        <div id="mapDiv"></div>
        <div id="title">
            FingerEyes-Xr for HTML5 :
            <font color="#349bd6">Graphic Edit Example</font>
            <br />
            <input type="button" value="ZoomIn" onclick="onZoomInMap();">
            <input type="button" value="ZoomOut" onclick="onZoomOutMap();">
            <input type="button" value="MapView" onclick="onView();">
            <input type="button" value="Edit" onclick="onEdit();">
            <input type="button" value="New Point" onclick="onAddPoint();" />
            <input type="button" value="New Polyline" onclick="onAddPolyline();"/>
```

```
            <input type="button" value="New Polygon" onclick="onAddPolygon();" />
            <input type="button" value="New Text" onclick="onAddText();" />
            <input type="button" value="New Rect" onclick="onAddRect();" />
            <input type="button" value="New Ellipse" onclick="onAddEllipse();" />
            <input type="button" value="Undo" onclick="onUndo();" id="btnUndo" />
            <input type="button" value="Redo" onclick="onRedo();" id="btnRedo" />
        </div>
    </div>
</body>

</html>
```