Autodesk®
Scaleform®

Scale9Grid 사용자 가이드

이 문서는 크기 조절이 가능한 창, 패널 및 버튼을 생성하기 위한 Scaleform 내 Scale9Grid 기능을 사용하는 데 관한 세부사항을 설명한다.

저자: Maxim Shemanarev, Michael Antonov

버전: 1. 01

최종 편집: 2008 년 3 월 21 일

Autodesk®
GAMEWARE

Copyright Notice

Autodesk® Scaleform® 4.4

Disclaimer

연락처:

문서	Scale9Grid 사용자 가이드
주소	Autodesk Scaleform Corporation 6305 Ivy Lane, Suite 310 Greenbelt, MD 20770, USA
홈페이지	http://www.scaleform.com/
이메일	info@scaleform.com
직통전화	(301) 446-3200
팩스	(301) 446-3199

목차

1. 소개: 플래시와 Scaleform 에서의 Scale9Grid

Scale9Grid 는 영화 클립을 위한 컴포넌트 스타일을 정의하기 위해 Flash®에서 사용된다. Scale9Grid 는 그래픽 및 디자인 요소에 있어 일반적으로 실시한 선형 스케일링 대신에 제공된 화면 자산을 사용하기 위해 효과적으로 리사이즈를 하는 영화 클립 기호 및 사용자 인터페이스 컴포넌트를 개발자가 만들 수 있도록 한다.

플래시 용어에서 Scale9Grid 는 또한 9 슬라이스 스케일링이라고도 알려져 있다. 9 슬라이스의 사용이 가능할 때 영화 클립은 개념적으로 그리드와 같은 오버레이가 있는 9 개의 구역으로 나뉘며 9 개의 영역 각각은 독립적인 크기 조절이 이뤄진다. 영화 클립의 시각적 통합을 유지하기 위해 모서리 크기를 조절하지 않는 반면 이미지의 나머지 영역을 스케일링한다 (늘리는 경우와는 반대로).

플래시 스튜디오에서 9 슬라이스 스케일링은 영화 클립의 "기호 속성" 대화창에 있는 체크 박스를 통해 가능하다. 이에 더해 MovieClip.scale9Grid 와 Button.scale9Grid 속성 또한 ActionScript 에 노출되어 조각별 스케일링에 대한 프로그램 제어가 가능하다. 플래시에서의 Scale9Grid 사용에 대한 보다 자세한 사항은 플래시 스튜디오 문서를 참고한다.

어도비 플래시 플레이어는 Scale9Grid 를 많은 실용적인 분야에서 사용하는 데 있어 상대적으로 사용을 어렵게 만드는 제한 사항을 갖는다. 예를 들어 표준 플래시 플레이어는 이미지 슬라이스를 지원하지 않으며 자유로운 회전 또는 기타 임의 변환에 있어 Scale9Grid 의 처리를 지원하지 않는다. Scaleform 는 Scale9Grid 를 더욱 일관적인 방법으로 지원함과 동시에 플래시와도 양호한 호환성을 유지한다. 어도비 플래시 및 Scaleform 모두의 Scale9Grid 특성이 아래의 표에 나와 있다.

Scale9Grid 특성	**어도비 플래시 플레이어**	**Scaleform**
변환	X/Y 스케일만	임의 변환
비트맵	경계 상자를 기준으로 평균 스케일링	Scale9Grid 에 따른 자동 타일
그래디언트 및 이미지 채우기	경계 상자를 기준으로 평균 스케일링	경계 상자를 기준으로 평균 스케일링
하위 기호 (인접한 영화 클립, 버튼, 그래픽)	지원하지 않음	지원함
인접 Scale9Grids	지원하지 않음	지원하지 않음
닫힌 기호 변환	지원함	지원함
텍스트	일반적으로 스케일링	일반적으로 스케일링

Scaleform 는 플래시와의 역 호환성을 보호하는데 이는 플래시에서 작동하는 Scale9Grid 의 기능이 Scaleform 에서 동일한 특성을 가짐을 의미한다. 그러나 플래시에서 지원하지 않는 특성은 다르게, 그러나 더욱 실용적인 방법으로 작동할 수 있다. 예를 들어 하위 기호의 지원 (인접한 영화 클립)은 인터랙티브하게 리사이즈하는 창의 생성에 있어 필수적인 요소이다. 두 플레이어 구현 사이의 차이점을 이 문서 전반에서 다룬다.

Scale9Grid 의 바로 사용할 예제를 개발자에게 제공하기 위해 이 문서는 다수의 샘플 FLA/SWF 파일을 참고한다.

2. 변환

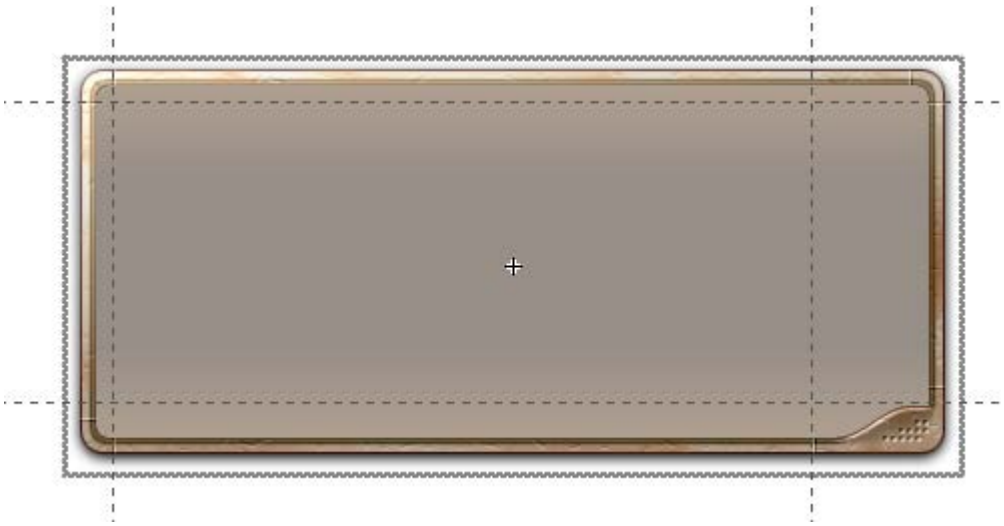
플래시와는 달리 Scaleform 는 Scale9Grid 가 적용된 영화 클립의 임의 변환을 지원한다. 반면 플래시에서 참조된 영화 클립의 회전이나 왜곡이 불가능하다. 그러나 스프라이트에서 결과로 생성되는 영화 클립을 포함한 후 닫힌 스프라이트의 회전이나 왜곡을 사용할 수 있다. 닫힌 스프라이트의 변환은 플래시와 Scaleform 모두에서 동일하게 작동한다. 높은 수준의 기호를 변환하는 것은 동일한 Scale9Grid 로 다르게 보이는 도형을 그리는 것을 가능하게 하며 이 예는 다음에 나온다.

3. 비트맵 처리

비록 그래디언트와 비트맵 도형 채우기 스타일이 플래시와 Scaleform 에서 동일하게 변환되나 별도의 스탠드 얼론 비트맵은 다르게 처리된다. 플래시에서 스탠드 얼론 비트맵이 9 슬라이드로 스케일링 된 영화 클립에 위치하는 경우 플레이어는 비트맵이 사용자에 의해 명시적으로 9 조각으로 분리되지 않은 경우라면 Scale9Grid 의 모든 설정을 무시한다. 반면 Scaleform 플레이어는 자동으로 비트맵을 자르므로 Scale9Grid 로 바르게 스케일링된다. 이러한 속성은 개발을 상당히 단순하게 만들며 아티스트가 수동으로 이미지를 자르지 않고도 리사이즈가 가능한 버튼 및 창을 생성할 수 있게 된다. 이에 더해 자동 슬라이스는 스티칭 컨텐츠가 EdgeAA 로 렌더링 된 경우 안티 앨리어싱 심을 제거한다. 자동 슬라이스 속성을 다음 예에서 더욱 자세히 다룬다.

아티스트가 아래의 이미지와 유사한 윈도우 배경을 나타내는 비트맵을 만들었다고 가정하자.

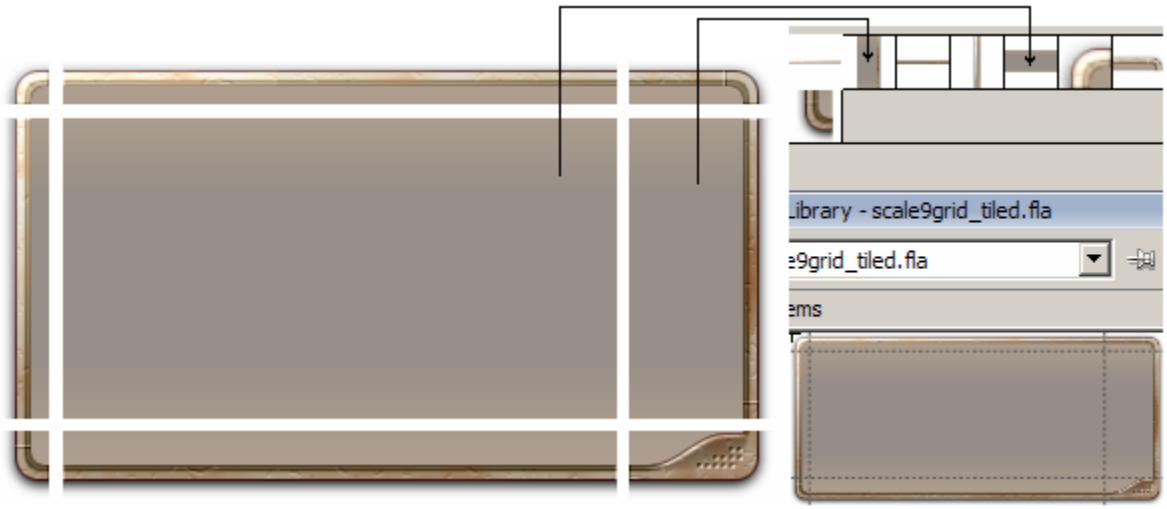
그 후 아티스트가 Scale9Grid 를 적용하여 모서리의 모양을 스케일링 할 때 보호할 수 있다. 이를 설정하기 위한 가장 당연한 방법은 다음과 같이 영화 클립을 생성하고 Scale9Grid 를 이에 적용하는 것이다.



그러나 예상한 바와 같이 플래시에서 작동하지 않는다. 결과로 나온 영화 클립을 스케일링하면 Scale9Grid 가 적용되지 않은 것처럼 나타난다.

Scaleform 에선 모서리의 모양을 보호하면서 예상했던 결과를 보여준다.

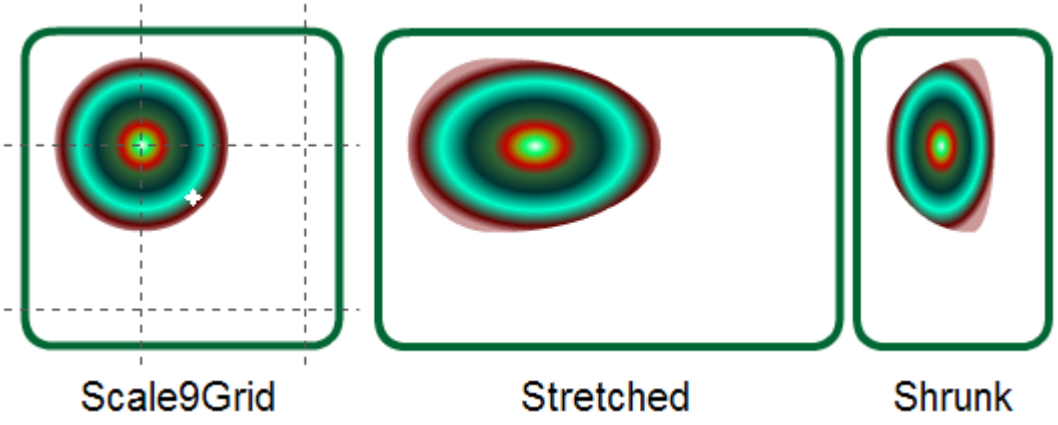
어도비 플래시는 경계 상자를 기준으로 이미지에 대한 평균 스케일을 계산한다. 이는 9 슬라이스 스케일링이 비트맵과 작동하는 것이 여전히 가능하나 Scale9Grid 와 정확히 대응하는 9 개의 독립된 이미지를 필요로 함을 의미한다.



Scaleform 는 플래시 파일을 재생할 때 이러한 작업을 자동으로 실시한다.

그러나 이미지를 임의로 회전하거나 왜곡시키는 경우 Scaleform 는 플래시와 동일하게 처리한다. 이렇게 "수동으로 붙인" 창의 예제는 scale9grid_tiled.fla 와 scale9grid_tiled.swf 를 통해 표시된다.

비트맵 및 그레디언트 채우기 스타일로 된 도형은 플래시와 Scaleform 에서 동일하게 나타난다. 그레디언트와 비트맵 채우기 모두 Scale9Grid 변환을 보존할 수 없다는 것을 확인한다. 결국 변환은 벡터 도형 외곽선에만 적용되며 채우기 스타일 컨텐츠에선 그렇지 않다는 것을 확인한다. 예를 들어,

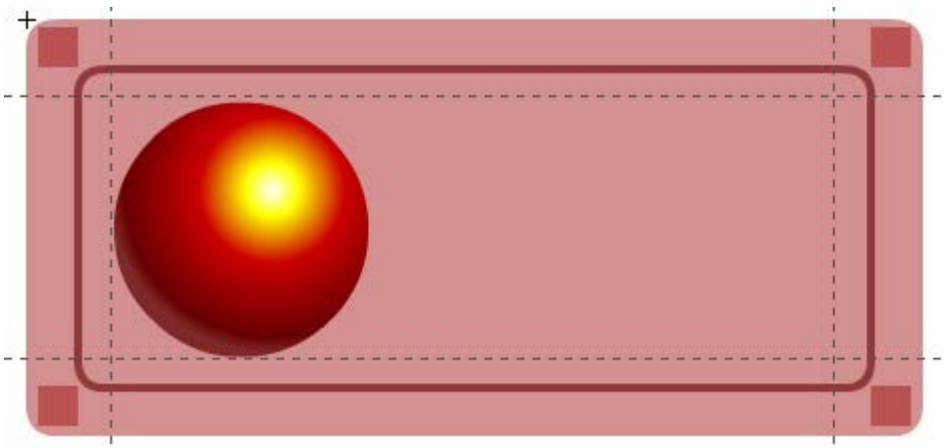


4. 인터랙티브 윈도우

여기에선 인터랙티브하게 리사이즈가 가능한 창에 관한 예를 제공하고 사용자 입력과 작업하는 데 필요한 ActionScript 소스 코드를 설명한다.

4.1 플래시 호환 예

다음의 예는 인터랙티브하게 리사이즈가 가능한 창을 플래시 호환 버전으로 만드는 방법을 설명한다. 이 예는 scale9grid_window1.fla 와 scale9grid_window1.swf 샘플 파일에 따른다. 플래시 스튜디오로 다음과 같은 영화 클립을 만들었다고 가정하자.



요지는 구석에 있는 작은 사각형을 드래그해서 표면 위 다른 부분을 드래그해서 이동하는 방법을 통해 창을 리사이즈하는 것이다. 여기에서 큰 문제는 플래시와 ActionScript 가 다른 도형과 영화 클립 내 레이어를 구분하는 리트 테스트에 대한 메커니즘을 제공하지 않는다는 점이다. 이에 더해 9 슬라이드 스케일링은 플래시에서 인접한 영화 클립에 대해선 작동하지 않는다. 히트 테스트 도형에 대한 해결책으로 프로그래머는 명시적으로 좌표계를 점검해야 한다. 아래의 예는 주어진 영화 클립과 관련한 ActionScript 프로그램을 나타낸다.

```
import flash.geom.Rectangle;

var bounds:Object = this.getBounds(this);
for (var i in bounds) trace(i+" --> "+bounds[i]);
this.XMin = bounds.xMin;
this.YMin = bounds.yMin;
this.XMax = bounds.xMax;
```

```
this.YMax = bounds.yMax;
this.MinW = 120;
this.MinH = 100;
this.OldX = this._x;
this.OldY = this._y;
this.OldW = this._width;
this.OldH = this._height;
this.OldMouseX = 0;
this.OldMouseY = 0;
this.XMode = 0;
this.YMode = 0;


this.onPress = function()
{
      this.OldX = this._x;
      this.OldY = this._y;
      this.OldW = this._width;
      this.OldH = this._height;
      this.OldMouseX = _root._xmouse;
      this.OldMouseY = _root._ymouse;
      this.XMode = 0;
      this.YMode = 0;

      var kx = (this.XMax - this.XMin) / this._width;
      var ky = (this.YMax - this.YMin) / this._height;
      var x1 = this.XMin + 6  * kx;   //좌측
      var y1 = this.YMin + 4  * ky;   //상단
      var x2 = this.XMax - 26 * kx;   //우측
      var y2 = this.YMax - 25 * ky;   //하단
      var w  = 20 * kx;
      var h  = 20 * ky;
      var xms = this._xmouse;
      var yms = this._ymouse;

      if (xms >= x1 && xms <= x1+w)
      {
            if (yms >= y1 && yms <= y1+h)
            {
                  this.XMode = -1;
                  this.YMode = -1;
            }
            else
            if (yms >= y2 && yms <= y2+h)
            {
                  this.XMode = -1;
                  this.YMode =  1;
            }
      }
      else
      if (xms >= x2 && xms <= x2+w)
      {
            if (yms >= y1 && yms <= y1+h)
            {
                  this.XMode =  1;
                  this.YMode = -1;
            }
```

```
            else
            if (yms >= y2 && yms <= y2+h)
            {
                  this.XMode =  1;
                  this.YMode =  1;
            }
      }

      if (XMode == 0 && YMode == 0)
      {
            this.startDrag();
      }
}


this.onRelease = function()
{
      this.XMode = 0;
      this.YMode = 0;
      this.stopDrag();
}

this.onReleaseOutside = function()
{
      this.XMode = 0;
      this.YMode = 0;
      this.stopDrag();
}

this.onMouseMove = function()
{
      var dx = _root._xmouse - OldMouseX;
      var dy = _root._ymouse - OldMouseY;

      if (this.XMode == -1)
      {
            this._x     = this.OldX + dx;
            this._width = this.OldW - dx;
            if (this._width < this.MinW || _root._xmouse > this.OldX + this.OldW)
            {
                  this._x     = this.OldX + this.OldW - this.MinW;
                  this._width = this.MinW;
            }
      }
      if (this.XMode == 1)
      {
            this._width = this.OldW + dx;
            if (this._width < this.MinW || _root._xmouse < this.OldX)
                this._width = this.MinW;
      }
      if (this.YMode == -1)
      {
            this._y      = this.OldY + dy;
            this._height = this.OldH - dy;
            if (this._height < this.MinH || _root._ymouse > this.OldY + this.OldH)
            {
                  this._y      = this.OldY + this.OldH - this.MinH;
                  this._height = this.MinH;
```

```
            }
        }
        if (this.YMode == 1)
        {
            this._height = this.OldH + dy;
            if (this._height < this.MinH || _root._ymouse < this.OldY)
                this._height = this.MinH;
        }
}
```

위와 같이 로직은 오히려 복잡하다. 영화 클립 및 기타 변수의 초기 경계 상자를 가질 필요가 잇다. 움직임을 제어하는 중요한 값은 XMode 와 YMode 이다. -1 의 모드값은 각각 왼쪽 또는 창 상단을 드래그하고 있음을 의미한다. 1 은 오른쪽 또는 하단을 드래그하고 있음을 의미한다.

주요 히트 테스트 로직이 onPress() 함수에 포함되어 있다. 우선 스케일링 계수인 kx 와 ky 를 계산해야 하는데 그 이유는 마우스 좌표계는 Scale9Grid 변환 로직을 자동으로 보호하지 않기 때문이다.

```
var kx = (this.XMax - this.XMin) / this._width;
var ky = (this.YMax - this.YMin) / this._height;
```

그 후 다음과 같은 히트 테스트 사각형 (이 경우에선 정사각형)을 계산한다.

좌측 상단: x1, y1, x1+w, y1+h

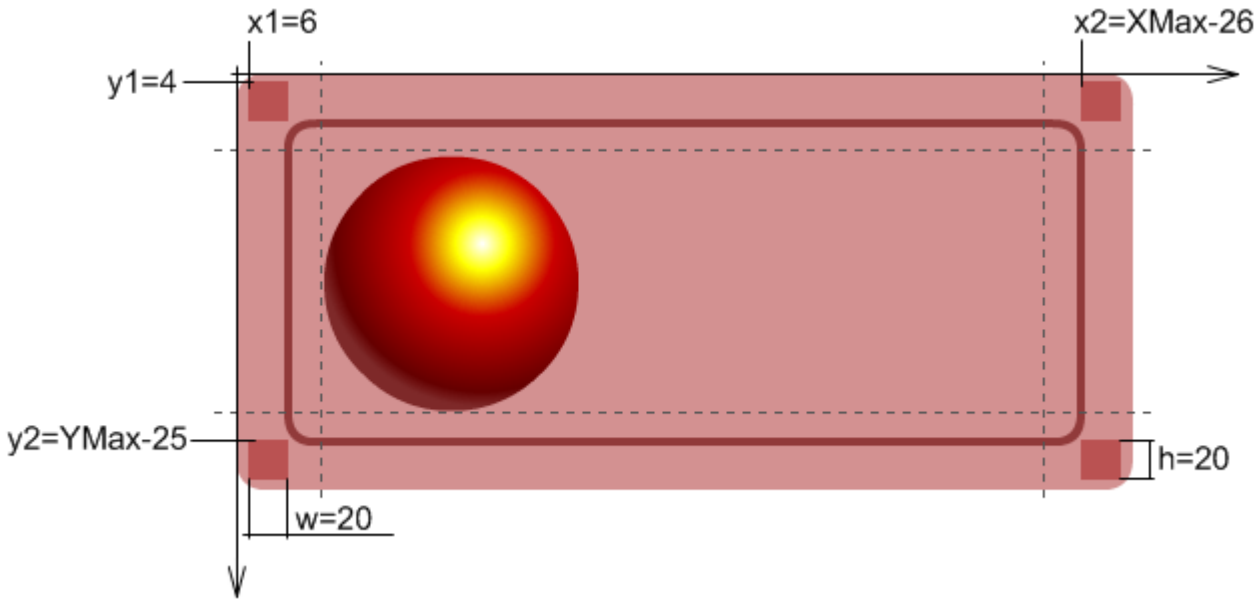
우측 상단: x2, y1, x2+w, y1+h

좌측 하단: x1, y2, x1+w, y2+h

우측 하단: x2, y2, x2+w, y2+h

```
var x1 = this.XMin + 6  * kx;   // 좌측
var y1 = this.YMin + 4  * ky;   // 상단
var x2 = this.XMax - 26 * kx;   // 우측
var y2 = this.YMax - 25 * ky;   // 하단
var w  = 20 * kx;
var h  = 20 * ky;
```

상수인 6, 4, 26, 25, 20 을 확인한다. 실제로는 영화 클립 내에서 상대 도형에 정확히 대응해야 하는 좌표계이다. 사실 구석에 있는 사각형은 정사각형일 필요가 없고 제거가 가능하다. 이는 이 방법에 있어선 큰 단점이다. 모양을 바꿀 때 이에 따른 ActionScript 코드를 수정해야 한다. 다음의 그림은 사용한 상수의 의미를 설명한다.



다음 단계에서 좌표계를 프로그램적으로 확인하고 각각의 리사이즈 모드를 할당한다.

```
var xms = this._xmouse;
var yms = this._ymouse;
if (xms >= x1 && xms <= x1+w)
{
      ... and so on
}
```

물론 모서리에 있는 도형이 복잡할수록 더 복잡한 로직을 필요로 한다. 예를 들어 다음과 같은 모서리는 두 개의 사각형에 대한 점검이 필요하다.

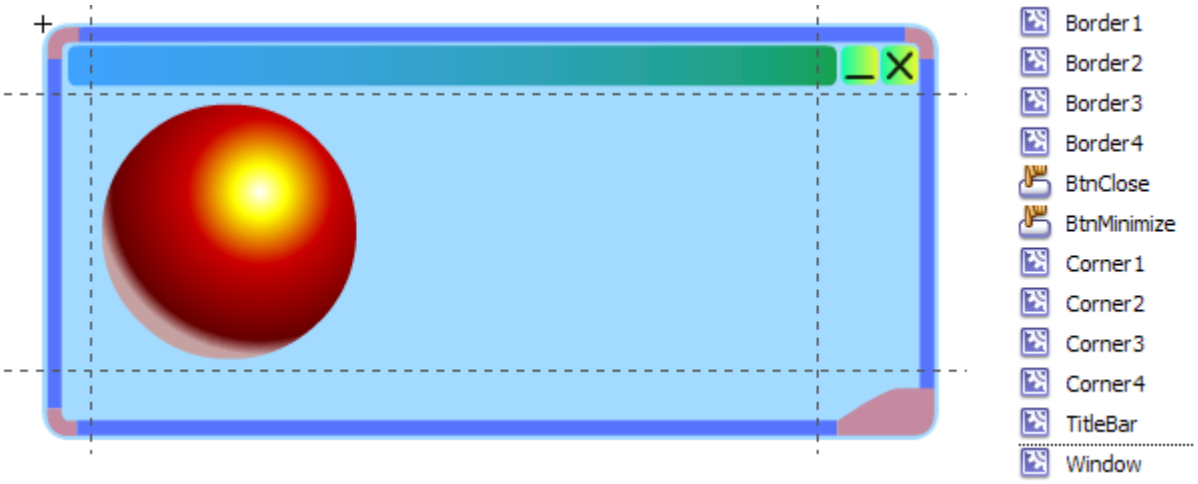
둥근 모서리 또는 불규칙 도형의 점검은 매우 복잡하다. 이에 더해 이러한 예는 리사이즈 경계를 처리하지 않는다.

ActionScript 의 나머지 로직은 창 좌표계 및 크기의 변화를 최소 폭과 높이 제한을 준수하면서 변화시킨다.

이미지 배경이 있는 창을 플래시와 완벽히 호환시키기 위해선 수동 접합이 필요하다는 점을 확인한다.

4.2 고급 Scaleform 예

Scaleform 에선 작동하나 플래시에서 작동하지 않는 하나의 예는 더욱 고급이다. 이는 Scaleform 가 인접한 영화 클립을 제대로 변환하는 것에 따르며 이는 임의의 불규칙 도형에 대해 마우스 히트 테스트 기능을 사용할 수 있음을 의미한다. 해당 내용이 scale9grid_window2.fla 와 scale9grid_window2.swf 파일에 들어 있다. 이러한 예에 사용되는 그래픽은 더욱 복잡하며 하드 코딩된 좌표계를 사용하지 않고 쉽게 재사용이 가능한 일반화된 ActionScript 코드를 기준으로 하는 더욱 향상된 기능을 제공한다.



샘플 플래시 파일은 별도의 영화 클립과 Border 1 에서 4 (파란색), Corner 1 에서 4 (분홍색)과 같은 버튼, TitleBar 등으로 구성되어 있다. Scaleform 가 인접 영화를 Scale9Grid 에 따라 올바로 정렬시키기 때문에 영화 클립에 함수를 할당하면 된다. ActionScript 는 단순하고 분명해진다. OnResize() 함수는 이전의 예와 동일하다.

```
import flash.geom.Rectangle;
//trace(this.scale9Grid);
this.MinW = 100;
this.MinH = 100;
this.XMode = 0;
this.YMode = 0;
this.OldX = this._x;
this.OldY = this._y;
this.OldW = this._width;
this.OldH = this._height;
```

```
this.OldMouseX = 0;
this.OldMouseY = 0;

function AssignResizeFunction(mc:MovieClip, xMode:Number, yMode:Number)
{
      mc.onPress          = function() { this._parent.StartResize(xMode, yMode);
      mc.onRelease        = function() { this._parent.StopResize(); }
      mc.onReleaseOutside = function() { this._parent.StopResize();    }
      mc.onMouseMove      = function() { this._parent.OnResize(); }
}

AssignResizeFunction(this.Border1,  0, -1);
AssignResizeFunction(this.Border2,  1,  0);
AssignResizeFunction(this.Border3,  0,  1);
AssignResizeFunction(this.Border4, -1,  0);
AssignResizeFunction(this.Corner1, -1, -1);
AssignResizeFunction(this.Corner2,  1, -1);
AssignResizeFunction(this.Corner3,  1,  1);
AssignResizeFunction(this.Corner4, -1,  1);
this.TitleBar.onPress          = function() { this._parent.startDrag(); }
this.TitleBar.onRelease        = function() { this._parent.stopDrag(); }
this.TitleBar.onReleaseOutside = function() { this._parent.stopDrag(); }

function StartResize(xMode:Number, yMode:Number)
{
      this.XMode = xMode;
      this.YMode = yMode;
      this.OldX = this._x;
      this.OldY = this._y;
      this.OldW = this._width;
      this.OldH = this._height;
      this.OldMouseX = _root._xmouse;
      this.OldMouseY = _root._ymouse;
}

function StopResize()
{
      this.XMode = 0;
      this.YMode = 0;
}
function OnResize()
{
      var dx = _root._xmouse - OldMouseX;
      var dy = _root._ymouse - OldMouseY;

      if (this.XMode == -1)
      {
            this._x     = this.OldX + dx;
            this._width = this.OldW - dx;
            if (this._width < this.MinW || _root._xmouse > this.OldX + this.OldW)
            {
                  this._x     = this.OldX + this.OldW - this.MinW;
                  this._width = this.MinW;
            }
      }
      if (this.XMode == 1)
      {
            this._width = this.OldW + dx;
```

```
            if (this._width < this.MinW || _root._xmouse < this.OldX)
                this._width = this.MinW;
      }
      if (this.YMode == -1)
      {
            this._y      = this.OldY + dy;
            this._height = this.OldH - dy;
            if (this._height < this.MinH || _root._ymouse > this.OldY + this.OldH)
            {
                  this._y      = this.OldY + this.OldH - this.MinH;
                  this._height = this.MinH;
            }
      }
      if (this.YMode == 1)
      {
            this._height = this.OldH + dy;
            if (this._height < this.MinH || _root._ymouse < this.OldY)
                this._height = this.MinH;
      }
}
```

이 방법은 두 가지 단점이 있다. 우선 어도비 플래시 플레이어와의 호환이 불가능하다. 두 번째 문제는 내부에 많은 영화 클립을 갖고 있어서 추가적인 draw primitive 와 여분의 메모리를 사용하게 된다. 이는 단순하고 일반적인 솔루션에 대해 여러분들이 지불하는 대가이다. 만약 추가적인 draw primitive 가 중요한 경우 두 가지 방법을 결합하고 경계와 모서리를 윈도우 프레임이 나타내는 하나의 도형으로 대체하는 것이 가능하다. 추가적인 로직을 ActionScript 코드에 추가할 필요가 있을 때도 있지만 처음의 예보다는 덜 복잡하고 덜 "취약"해야 한다. 주요 장점은 불규칙 경계 및 모서리로 된 영화 클립의 히트 테스트를 위해 임의의 도형을 사용할 수 있다는 점이다. 도형 및 이동 tween 또한 작동한다.

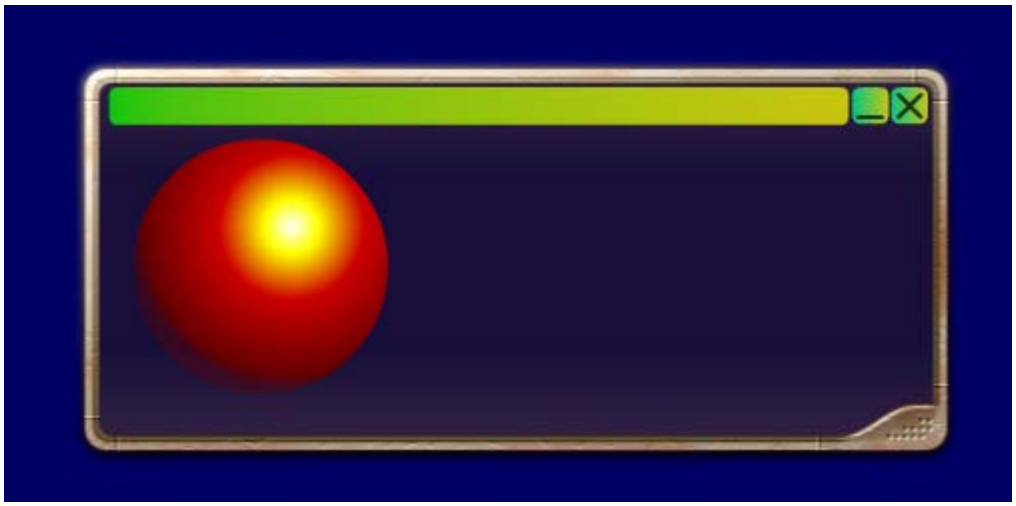
플래시 스튜디오가 Scale9Grid 로 SWF 파일을 생성할 때 잠재적인 버그를 갖는다는 사실을 언급하는 게 중요하다. 이를 scale9grid_window2.fla 로 재생산할 수 있다. 만약 "TitleBar" 레이어를 선택하고 창 가운데 작은 사각형을 그리면 Scale9Grid 를 부정확해진다. 플래시 스튜디오가 인접한 영화 클립과 기타 그래픽이 동일한 레이어에 위치하는 것을 용인하지 않는다. 몇몇 경우 이 사각형을 제거한 후 올바른 Scale9Grid 의 복구에 실패하기도 한다. "Save and Compact" 작업은 다음을 돕는다. 문제가 지속되는 경우 ActionScript 에서 Scale9Grid 를 복구하는 게 가능하다. 맞는 동안 단순히 추적한다.

```
trace(this.scale9Grid);
```

버그가 나타나고 제거할 수 없다면 다음의 코드를 사용해서 복구할 수 있다.

```
this.scale9Grid = new Rectangle(24, 35, 363, 138);
```

비트맵 기반 배경의 예에선 scale9grid_window3.fla 와 scale9grid_windoe3.swf 파일을 참고한다. 이 파일에서 모서리 경계에 대한 영화 파일을 투명하게 만들었는데 그 이유는 그 목적이 단지 히트 테스트 기능을 지원하기 대문이다. 단순히 배경 이미지 내에 있는 모서리의 도형을 복제한다.



이 문서 초반에서 다룬 바와 같이 Scaleform 에서 배경으로서 단일 이미지를 사용하는 반면 플래시에선 수동으로 이를 자르고 붙여야 한다. 이에 더해 모서리 및 경계 영화 클립 기술은 어도비 플래시 플레이어에서 작동하지 않는데 그 이유는 히트 테스트 변환을 제대로 보호하지 않기 때문이다. 비슷하게 Scale9Grid 클립의 회전 및 왜곡이 표준 플래시 플레이어에선 제대로 작동하지 않는다. 그러나 Scaleform 와 함께 가능한 단순하고 효과적으로 리사이즈가 가능한 창을 만들기 위해 노력하는 동시에 scale9grid 가 변환을 제대로 지원함을 확인한다.