

# Silverlight 4

## فهرست مطالب

فصل ۲۲ – برنامه نویسی گرافیکی در Silverlight	۴۸۶
مقدمه	۴۸۶
معرفی اشکال گرافیکی مهیا در Silverlight	۴۸۶
ترسیم اشکال هندسی غیر استاندارد در Silverlight	۴۸۸
تغییر شکل دادن (Transform) اشیاء در Silverlight	۴۹۰
تبدیل قالب‌های گرافیکی دیگر به XAML	۴۹۳

چاپ عمومی غیر رایگان این مطالب بدون مجوز کتبی از طرف نویسنده به هر نحوی غیرمجاز است.  
انتشار این مطالب بر روی اینترنت و یا استفاده از آن به صورت مستقیم و یا غیر مستقیم در نشریات الکترونیکی با ذکر مأخذ بلا مانع است.

## فصل ۲۲ - برنامه نویسی گرافیکی در Silverlight


### مقدمه







تاکنون در طی فصل‌های قبلی کتاب با یک سری از اشیاء ساده گرافیکی مانند مستطیل، بیضی و غیره آشنا شدیم؛ اما بررسی تفصیلی آن‌ها صورت نگرفت. گرافیک در Silverlight از نوع برداری بوده و برای نمایش در برنامه‌های Web بسیار مناسب است؛ زیرا به سادگی می‌توانند خود را با حفظ کیفیت اصلی، با اندازه‌های متفاوت مرورگر وفق دهند. همچنین کاربران با نقایص بینایی را نیز باید در نظر داشت؛ اینگونه کاربرها تمایل دارند اندازه‌ی اشیاء، متون و طرح بندی برنامه را جهت استفاده از آن‌ها بزرگتر نمایند و اینجا است که گرافیک برداری، دسترسی پذیری برنامه‌های شما را بهبود خواهد بخشید.

### معرفی اشکال گرافیکی مهیا در Silverlight

در طی یک جدول ساده و کاربردی قصد داریم با اشکال گرافیکی مهیا در Silverlight آشنا شویم. تمامی این اشکال از نوع UIElement می‌باشند (برای مثال رخدادهای صفحه کلید و Mouse به صورت پیش فرض برای آن‌ها مهیا است).

#### جدول ۱- اشکال گرافیکی مهیا در Silverlight

توضیحات	شیء گرافیکی
مختصات نقاط ابتدا و انتهای یک خط که توسط خواص $X1$ ، $Y1$ ، $X2$ و $Y2$ معرفی می‌شوند مطلق نبوده و نسبی هستند (بر اساس محل قرارگیری پنل طرح بندی تعیین موقعیت خواهند شد)	<p>Line</p>  <pre>&lt;Canvas x:Name="myCanvas" Height="20" Width="50"&gt;   &lt;Line Stroke="Black" X1="10" Y1="10" X2="30" Y2="30" /&gt; &lt;/Canvas&gt;</pre>
همانطور که ملاحظه می‌نمائید امکان تشکیل یک مستطیل با گوشه‌های گرد نیز وجود دارد. سادگی این مورد را با امکانات HTML سنتی برای تهیه‌ی گوشه‌های گرد یک پنل مقایسه نمائید.	<p>Rectangle</p>

	 <pre>&lt;Rectangle Stroke="Black" Width="104" Height="64" Canvas.Left="8" Canvas.Top="8"/&gt;</pre>  <pre>&lt;Rectangle Stroke="Black" Width="104" Height="64" Canvas.Left="8" Canvas.Top="8" RadiusX="10" RadiusY="10"/&gt;</pre>
<p>از خاصیت Stroke در کلیه اشیاء گرافیکی Silverlight جهت تعیین رنگ حاشیه‌ی خارجی شکل استفاده می‌شود.</p>	<p>Ellipse</p>  <pre>&lt;Ellipse Stroke="Black" Width="104" Height="64" Canvas.Left="8" Canvas.Top="8"/&gt;</pre>
<p>توسط Polyline به سادگی می‌توان سری به هم پیوسته‌ی یک سری خطوط را ترسیم کرد. هر جفت اعدادی که در آن ملاحظه می‌کنید بیانگر مختصات یک نقطه هستند که با فاصله از هم جدا شده‌اند:</p> <p>"[X-Coordinate],[Y-Coordinate]"</p> <p>یکی دیگر از خواص جالب این شیء، StrokeDashArray می‌باشد که برای ترسیم dashed lines کاربرد دارد. اعداد این آرایه مضربی از StrokeThickness را جهت ترسیم یک خط تیره و سپس ترسیم فاصله‌ی پس از آن، ارائه خواهند داد.</p>	<p>Polyline</p>  <pre>&lt;Polyline Stroke="Black" Points="10,50 20,40 23,44 25,49 40,12 46,50 51,42 55,50" /&gt;</pre>  <pre>&lt;StackPanel&gt; &lt;Polyline Stroke="Blue" StrokeThickness="14" Points= "10,30 60,0 90,40 120,10 350,10" Margin="10" /&gt;  &lt;Polyline Stroke="Blue" StrokeThickness="14" StrokeDashArray="1 2" Margin="10" Points="10,30 60,0 90,40 120,10 350,10" /&gt; &lt;/StackPanel&gt;</pre>
<p>تفاوت Polygon با Polyline تنها در بسته بودن شکل نهایی آن است.</p>	<p>Polygon</p>  <pre>&lt;Polygon Stroke="Black" Points="10,40 20,10 60,10 70,40 10,40" /&gt;</pre>

کلیه اشکال گرافیکی فوق دارای خاصیت Stretch نیز هستند که در این مورد در طی فصل معرفی کار با تصاویر و فایل‌های چند رسانه‌ای توضیحات لازم ارائه شد.

## ترسیم اشکال هندسی غیر استاندارد در Silverlight

هر چند در ابتدا شاید نتوان بین اشکال هندسی و گرافیکی که پیشتر معرفی شدند تفاوتی قائل شد اما باید در نظر گرفت که اشیاء هندسی که در ادامه معرفی می‌گردند، از نوع UIElement نیستند. اما مزیت آن‌ها ایجاد اشیایی است که جزو اشکال گرافیکی استاندارد Silverlight نیستند. علاوه بر آن امکانات Clipping (مشخص سازی ناحیه‌ی قابل رویت از یک شیء دیگر که در فصل معرفی بکارگیری تصاویر و امکانات چند رسانه‌ای ذکر گردید) و hit-testing را نیز مهیا می‌کنند. اشکال هندسی ساده شامل LineGeometry، RectangleGeometry و EllipseGeometry می‌شوند. این اشکال به تنهایی قابل استفاده نبوده و باید توسط دربرگیرنده‌هایی مانند شیء Path ارائه شوند. برای مثال یک LineGeometry را در حالت ساده به صورت زیر می‌توان تعریف کرد:

### XAML

```
<Path Stroke="Black" StrokeThickness="1" >
  <Path.Data>
    <LineGeometry StartPoint="8,8" EndPoint="72,72" />
  </Path.Data>
</Path>
```

شیء Path برای ایجاد اشکال پیچیده‌ی هندسی به این مجموعه اضافه شده است. برای مثال شکل هندسی کلید زیر را در نظر بگیرید:



شکل ۱- ایجاد یک کلید به کمک اشیاء هندسی

کدهای XAML متناظر با این شیء کلید به شرح بعد هستند:

### XAML

```
<Path Stroke="Navy" StrokeThickness="8" Fill="Navy">
  <Path.Data>
    <GeometryGroup FillRule="Evenodd">
      <EllipseGeometry Center="20,40" RadiusX="15" RadiusY="15" />
      <LineGeometry StartPoint="20,40" EndPoint="70,40" />
      <LineGeometry StartPoint="66,38" EndPoint="66,55" />
      <LineGeometry StartPoint="55,38" EndPoint="55,55" />
    </GeometryGroup>
  </Path.Data>
</Path>
```

```
<EllipseGeometry Center="14,40" RadiusX="8" RadiusY="8" />
</GeometryGroup>
</Path.Data>
</Path>
```

کاربرد اشکال هندسی ساده‌ی LineGeometry و EllipseGeometry را در این مثال می‌توان مشاهده نمود. از GeometryGroup برای ترکیب موجودیت‌های منقطع و ایجاد یک شکل جدید استفاده می‌شود. توسط خاصیت FillRule آن مشخص می‌شود که نواحی متداخل چگونه باید یکدیگر را پوشش داده و ترکیب شوند. دو مقدار مجاز آن EvenOdd و Nonzero می‌باشند. در حالت EvenOdd برای اینکه تشخیص داده شود یک نقطه داخل ناحیه‌ی پوششی است، یک خط فرضی از آن به طرف خارج و بی نهایت ترسیم می‌شود. اگر تعداد تقاطع‌های این خط فرضی یک عدد فرد باشد، نقطه درون ناحیه‌ی پوششی قرار دارد و برعکس. در حالت Nonzero نیز همان خط فرضی ترسیم می‌شود اما در این حالت اگر خطوطی آن‌را از جهت چپ به راست قطع کنند یک عدد به شمارشگر اضافه شده و اگر از راست به چپ آن‌را قطع کنند یک عدد از این شمارشگر که از صفر شروع خواهد شد کسر می‌گردد. اگر نتیجه‌ی حاصل صفر شد، به این معنا است که نقطه خارج از ناحیه قرار دارد و برعکس.



شکل ۲- استفاده از PathGeometry برای ترسیم خطوط و منحنی‌های پیچیده

علاوه بر این امکانات، توسط اشیاء PathGeometry می‌توان خطوط و منحنی‌های پیچیده‌ای را ترسیم کرد. هر PathGeometry از یک سری شیء PathFigure که بیانگر قسمتی از آن می‌باشند، تشکیل خواهد شد. کدهای XAML شکل فوق در ادامه ذکر شده‌اند. در هر PathFigure می‌توان از اشیایی مانند LineSegment برای ترسیم یک خط، از PolyLineSegment برای ترسیم یک سری از خطوط، از ArcSegment جهت ترسیم یک منحنی بین دو نقطه، از BezierSegment برای ترسیم منحنی Bezier بین دو نقطه، از PolyBezierSegment برای ترسیم منحنی‌های cubic Bezier، از QuadraticBezierSegment جهت ترسیم منحنی‌های quadratic Bezier، و از PolyQuadraticBezierSegment برای ترسیم یک سری از منحنی‌های quadratic Bezier، استفاده کرد.

#### XAML

```
<Canvas
Width="100" Height="100" Background="Gray">
```

```

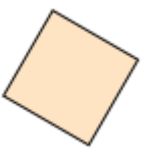
<Path Stroke="Red" StrokeThickness="2">
  <Path.Data>
    <PathGeometry>
      <PathGeometry.Figures>
        <PathFigure StartPoint="5,5">
          <PathFigure.Segments>
            <ArcSegment Size="10,10" RotationAngle="30"
              Point="20,10"
              IsLargeArc="False"
              SweepDirection="Clockwise" />
            <BezierSegment Point1="40,0"
              Point2="60,60" Point3="75,90"/>
            <LineSegment Point="80,15" />
            <PolyLineSegment Points="50,90 3,7" />
            <QuadraticBezierSegment Point1="90,90" Point2="70,60"/>
          </PathFigure.Segments>
        </PathFigure>
      </PathGeometry.Figures>
    </PathGeometry>
  </Path.Data>
</Path>
</Canvas>


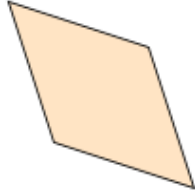
```

## تغییر شکل دادن (Transform) اشیاء در Silverlight


با استفاده از المان Transform در Silverlight می‌توان هر نوع UIElement را تغییر شکل داد. ارزش واقعی آن‌ها را در طی فصل پویا نمایی بیشتر درک خواهیم کرد. در ادامه در طی جدول بعد مروری خواهیم داشت بر انواع امکاناتی که المان Transform در اختیار ما قرار می‌دهد.

جدول ۲- انواع حالات تغییر شکل در Silverlight

توضیحات	حالات مختلف تغییر شکل
<p>RotateTransform سبب چرخش یک شیء در جهت عقربه‌های ساعت می‌شود. این چرخش در حالت پیش فرض حول نقطه‌ی صفر و صفر است که توسط خواص CenterX و CenterY مشخص شده است. اگر علاقمند بودید که جهت چرخش برخلاف عقربه‌های ساعت باشد، یک مقدار منفی را در خاصیت Angle وارد نمایید (از ۳۶۰ تا منفی ۳۶۰ درجه).</p>	<p>RotateTransform</p>  <pre> &lt;Rectangle Width="50" Height="50"   Fill="Bisque"   Stroke="Black"&gt; </pre>

	<pre> &lt;Rectangle.RenderTransform&gt;   &lt;TransformGroup&gt;     &lt;RotateTransform       CenterX="0"       CenterY="0"       Angle="30"/&gt;     &lt;/TransformGroup&gt;   &lt;/Rectangle.RenderTransform&gt; &lt;/Rectangle&gt; </pre>
<p>ScaleTransform سبب تغییر ابعاد یک شیء در جهات X و Y می‌گردد. برای مثال در سناریوهایی که نیاز به عملیات Zoom وجود داشته باشد بسیار مفید خواهد بود (با توجه به اینکه به هر نوع UI Element ایی قابل اعمال است).</p> <p>در اینجا نیز می‌توان خواص CenterX و CenterY را مشخص نمود که حالت پیش فرض آن صفر و صفر است (نقطه‌ی بالا سمت چپ شیء).</p>	<p><b>ScaleTransform</b></p>  <pre> &lt;Rectangle Width="30" Height="30"   Fill="Bisque"   Stroke="Black"&gt;   &lt;Rectangle.RenderTransform&gt;     &lt;TransformGroup&gt;       &lt;ScaleTransform         ScaleX="2.5"         ScaleY="1.5"/&gt;     &lt;/TransformGroup&gt;   &lt;/Rectangle.RenderTransform&gt; &lt;/Rectangle&gt; </pre>
<p>SkewTransform سبب مورب سازی یک شیء حول نقطه‌ی مشخص شده با خواص CenterY و CenterX به اندازه‌ی زوایای AngleX و AngleY می‌گردد.</p>	<p><b>SkewTransform</b></p>  <pre> &lt;Rectangle Width="75" Height="75"   Fill="Bisque"   Stroke="Black"&gt;   &lt;Rectangle.RenderTransform&gt;     &lt;TransformGroup&gt;       &lt;SkewTransform         CenterX="10"         CenterY="10"         AngleX="18"         AngleY="18"/&gt;     &lt;/TransformGroup&gt;   &lt;/Rectangle.RenderTransform&gt; &lt;/Rectangle&gt; </pre>
<p>TranslateTransform سبب انتقال مکان یک شیء از نقطه‌ای به نقطه‌ی دیگر می‌گردد.</p>	<p><b>TranslateTransform</b></p> <pre> &lt;Rectangle Width="50" Height="50"   Fill="Bisque"   Stroke="Black"&gt;   &lt;Rectangle.RenderTransform&gt;     &lt;TransformGroup&gt;       &lt;TranslateTransform         X="25" Y="25"/&gt;     &lt;/TransformGroup&gt;   &lt;/Rectangle.RenderTransform&gt; &lt;/Rectangle&gt; </pre>
<p>MatrixTransform بر اساس فرمول ذیل عمل می‌کند:</p> $X1 = X * M11 + Y * M21 + OffsetX$ $Y1 = X * M12 + Y * M22 + OffsetY$	<p><b>MatrixTransform</b></p>



<p>که پارامترهای آن به شکل بعد مقدار دهی خواهند شد :</p> <pre>&lt;MatrixTransform   Matrix="M11, M12, M21, M22, OffsetX, OffsetY"/&gt;</pre>	 <pre>&lt;Rectangle Width="130"   Height="100"     Stroke="Orange"     StrokeThickness="15"&gt;   &lt;Rectangle.RenderTransform&gt;     &lt;MatrixTransform       Matrix="1,1.5,1.25,1.25,1.5,1" /&gt;     &lt;/Rectangle.RenderTransform&gt;   &lt;/Rectangle&gt;</pre>
--	---



شکل ۳- اعمال گروهی از تغییر شکل‌ها به یک کنترل TextBox معمولی

اگر به مثال‌های قبل دقت کرده باشید، از المان TransformGroup در آن‌ها به کرات استفاده گردید. به کمک TransformGroup می‌توان چندین تغییر شکل پی در پی را به یک شیء اعمال نمود. به علاوه در این مثال‌ها تنها از یک شیء مستطیل استفاده گردید اما همانطور که ذکر شد، اعمال این تغییر شکل‌ها به کلیه عناصر UI مجاز است.

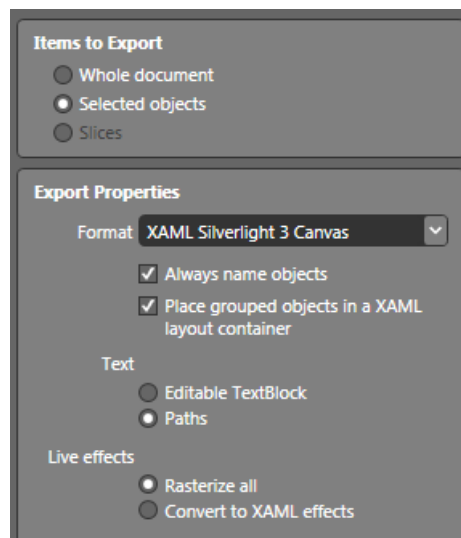
لطفاً به کدهای XAML مثال بعد در این زمینه دقت بفرمائید. نتیجه‌ی حاصل در شکل ۳ نمایش داده شده است. هرچند اعمال این تغییر فرم‌ها به کلیه عناصر UI مجاز است اما لطفاً زیاده روی نکنید و این فقط یک مثال بود!

#### XAML

```
<TextBox x:Name="myTextBox" Height="30" Width="90">
  <TextBox.RenderTransform>
    <TransformGroup>
      <RotateTransform Angle="15" />
      <SkewTransform AngleX="10" AngleY="10" />
      <ScaleTransform ScaleX="2" ScaleY="2" />
      <TranslateTransform X="10" Y="10" />
    </TransformGroup>
  </TextBox.RenderTransform>
</TextBox>
```

## تبدیل قالب‌های گرافیکی دیگر به XAML

در عمل اکثر تصاویر گرافیکی زیبای نمایش داده شده در برنامه‌های Silverlight به کمک دست و در کنار هم قرار دادن اشیاء مختلف گرافیکی تهیه نمی‌شوند. برای این منظور یکی از ابزارهای مورد استفاده، برنامه‌ی Microsoft Expression Design است که قابلیت گشودن فایل‌های برداری محصولات شرکت Adobe را نیز دارا است (برای مثال فایل‌های ai). در این برنامه از منوی File، گزینه‌ی Export آن را انتخاب کرده و در صفحه‌ی باز شده، XAML سازگار با Silverlight را انتخاب کنید (شکل ۴).



شکل ۴- امکان تهیه‌ی خروجی XAML از برنامه‌ی Microsoft Expression Design.

علاوه بر برنامه‌ی تجاری Microsoft Expression Design، تعدادی تبدیل‌کننده‌ی رایگان نیز در این زمینه مهیا هستند. برای مثال در آدرس‌های ذیل می‌توانید تبدیل‌کننده‌های رایگان فایل‌های Adobe Illustrator و XAML به SWF را دریافت نمایید:

<http://www.mikeswanson.com/xamlexport/>

<http://www.mikeswanson.com/swf2xaml/>

همچنین لیست مشروح دیگری را در آدرس بعد مشاهده خواهید نمود (برای مثال تبدیل فایل‌های Visio به XAML و غیره):

<http://blogs.msdn.com/b/mswanson/archive/2006/02/26/wpftoolsandcontrols.aspx>