POKROČILÁ 2D VEKTOROVÁ GRAFIKA

Afinní transformace

Kombinování primitiv

Vektorové formáty

AFINNÍ TRANSFORMACE

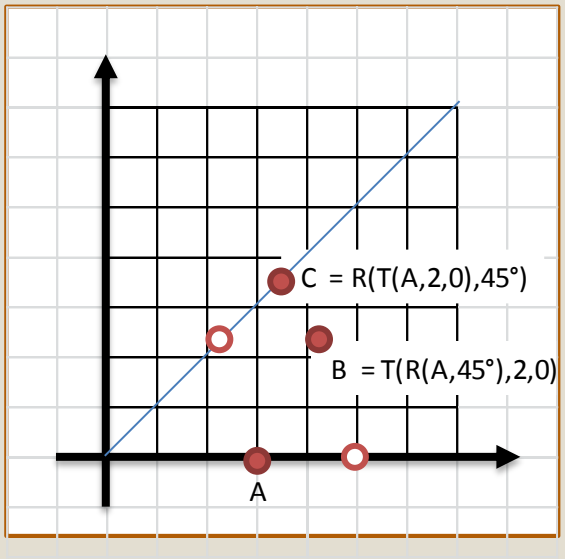
- Vytvoření mnoha modelů jednoduché pro speciální případ, ale složité obecně
  - např. kružnice se středem v počátku vs. jinde
  - šipka orientovaná dle souřadných os vs. obecně
  - text vodorovný vs. svislý (nebo šikmý)
- Řešení:
  - nastavit pro grafický kontext popis, jak chceme transformovat to, co budeme na něj kreslit
  - modelovat jednoduše, jak nám to vyhovuje
    - grafická knihovna automaticky model transformuje do výsledné pozice dle našeho „popisu“

AFINNÍ TRANSFORMACE

- Transformace:
  - Afinní: rovnoběžky zůstávají rovnoběžkami
  - Projektivní: rovnoběžky se mohou stát různoběžkami
  - Obecné: přímka se může stát křivkou

AFINNÍ TRANSFORMACE

- Popis výsledné transformace složen z elementárních transformací:
  - Posunutí (translace)
  - Rotace okolo počátku souřadného systému
  - Změna měřítka (scaling)
    - může být v každé ose jiná
- Pořadí aplikace elementárních transformací je klíčové
  - Různé grafické knihovny, nebo dokonce jejich různé části, mohou vyžadovat různou specifikaci



AFINNÍ TRANSFORMACE

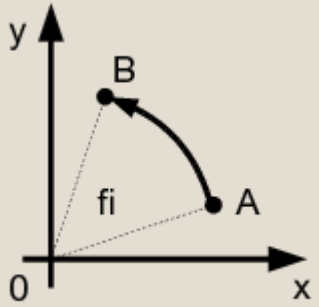
- Posunutí (translace) bodu o vektor $\vec{v}$

$$B_x = A_x + v_x$$
$$B_y = A_y + v_y$$

- Rotace o úhel φ

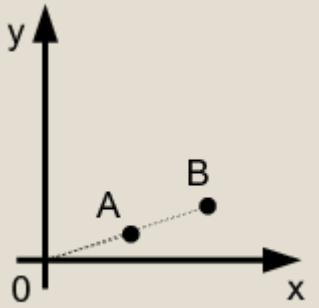
$$B_x = A_x \cdot \cos\varphi - A_x \cdot \sin\varphi$$
$$B_y = A_x \cdot \sin\varphi + A_y \cdot \cos\varphi$$

  - Pomůcka: když $\varphi = 0$, nic se nesmí stát



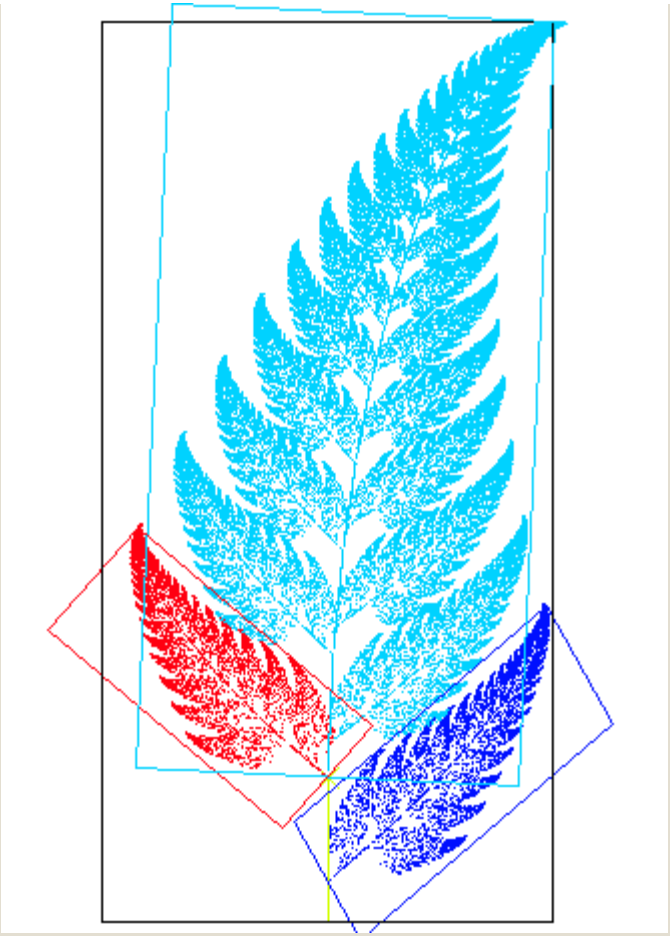
- Změna měřítka

$$B_x = A_x \cdot S_x$$
$$B_y = A_y \cdot S_y$$



AFINNÍ TRANSFORMACE

- Instancování
  - Model definován na jednom místě
  - Použit na více místech
    - Může být transformovaný
    - Ale v paměti je jen jednou

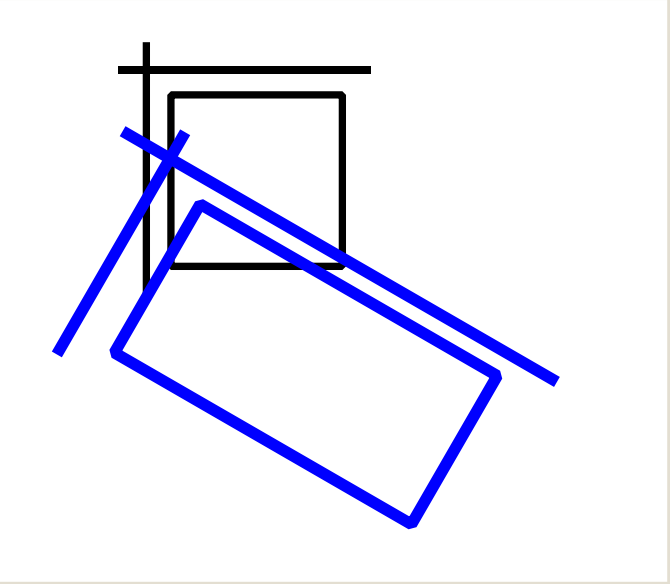


AFINNÍ TRANSFORMACE V JAVĚ

- Třída `AffineTransform`
  - Metody translate, rotate, scale, …
  - Instance předána grafickému kontextu
- Základní transformace lze provádět rovnou přes grafický kontext

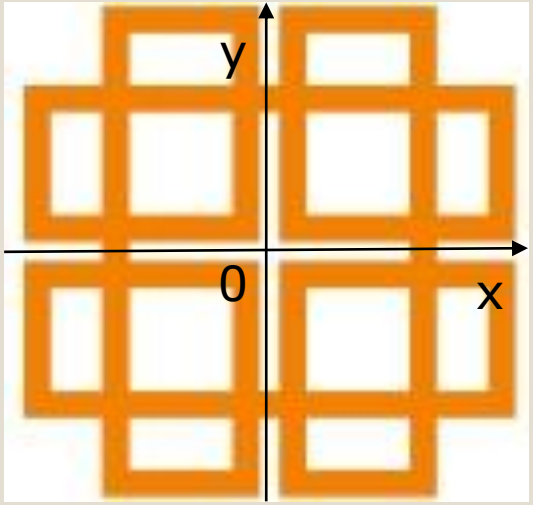
AFINNÍ TRANSFORMACE V JAVĚ

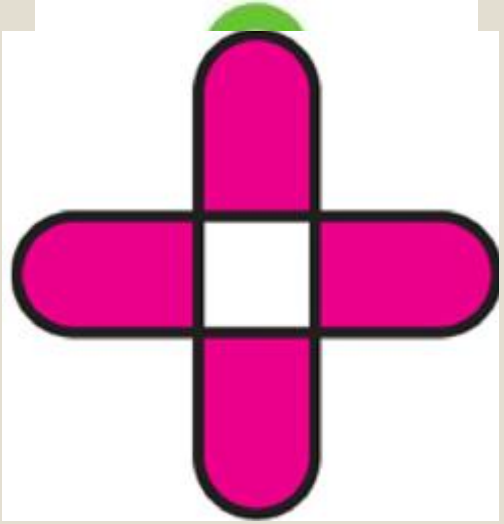
```
//Uložení aktuální transformace
AffineTransform saveTransform = g2d.getTransform();

//“Nulová“ transformace
AffineTransform identity = new AffineTransform();
g2d.setTransform(identity);

//kreslení „normálně“

g2d.rotate(Math.toRadians(30.0));
g2d.scale(2, 1);
g2d.translate(10, 20);

//kreslení do transformovaného systému

//obnovení původní transformace
g2d.setTransform(saveTransform);
```



MODELOVÁNÍ OBJEKTŮ

- Jak vykreslit tyto složitější objekty?





- Jak poznat, že uživatel na objekt kliknul?

MODELOVÁNÍ OBJEKTŮ

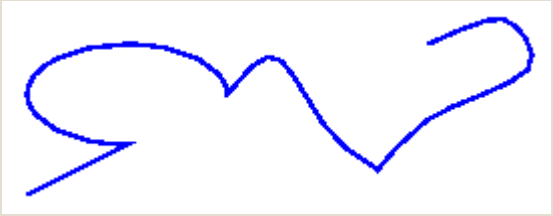
- Myšlenka: objekt nevzniká postupným vykreslováním primitiv, ale definován předem
- Objekt určen svým obrysem nebo vnitřkem
- Popis objektu uložen v paměti počítače
  - Lze různě modifikovat dle požadavků aplikace
  - Lze testovat na kolizi s jiným objektem
    - např. hit-testy
  - Lze vytvořit v jiné aplikaci, uložit na disk a později v naší aplikaci načíst a použít

MODELOVÁNÍ OBJEKTŮ

- Objekt modelován složením primitiv
- Grafické knihovny mohou podporovat
  - Různě složitá primitiva
    - bod, lomená čára, parametrická křivka, implicitní funkce
  - Různé způsoby skládání
    - pouhé sloučení neprotínajících se primitiv, operace sloučení, rozdílu a průniku protínajících se objektů
  - Různou definici vnitřku objektu
    - Objekt určen vnějším tvarem, uvnitř data chybí

CESTA - PATH

- Trajektorie pera opisující obrys objektu
- Cesta složena z elementárních primitiv
- Může být otevřená nebo uzavřená
  - Některé knihovny umožňují automatické uzavření
- Definice cesty
  - Zahájení cesty (BeginPath)
  - Přidání primitiv do cesty
  - Ukončení cesty (EndPath)



- Některé knihovny mají cestu výhradně pro kreslení do grafického kontextu
  - Např. GDI, HTML5

CESTA - PATH

- Nejčastěji podporovaná primitiva:
  - Bod
  - Úsečka
  - Obdélník
  - Oblouk
    - včetně speciálního případu elipsy a kružnice
  - Parametrické křivky
- Cesta může být definována po částech, tj. „primitivem“ může být jiná cesta

CESTA - PATH

- Grafické knihovny mohou rovněž podporovat přidání do cesty obrysů písmen textu
  - Vhodné pro převod písma do křivek
    - Snadná distribuce

- Cesta je také používána pro definování křivky, podle které má být zobrazen text
  - Zobrazení si většinou musíme udělat sami

CESTA – PATH V JAVĚ

- Jmenný prostor `java.awt.geom`
- Podporovaná primitiva
  - Body – `Point2D`
  - Úsečky – `Line2D`
  - Základní tvary – `Rectangle2D`, `RoundRectangle2D`, `Ellipse2D`, `Arc2D`, `Dimension2D`
  - Kvadratické a kubické křivky – `QuadCurve2D`, `CubicCurve2D`
  - Obecné tvary – `Path2D`

CESTA – PATH V JAVĚ

- Primitiva implementovaná ve dvou verzích:
  - Jednoduchá (`float`) a dvojnásobná (`double`) přesnost
  - Např. `Point2D bod = new Point2D.Double();`
- Třídy primitiv obsahují vedle metod pro jejich definici také metody pro manipulaci s nimi
  - Např. vzdálenost dvou bodů, průsečík přímek, …

CESTA – PATH V JAVĚ

- Cesta = třída `GeneralPath`

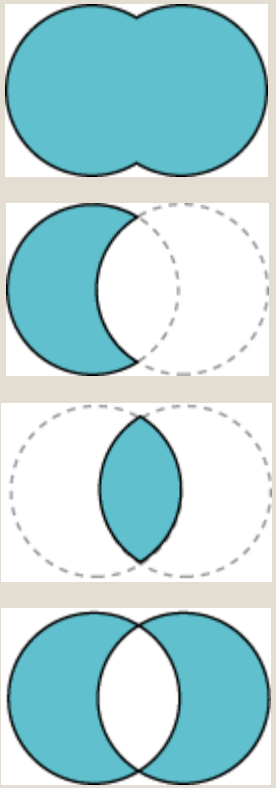
```
int[] x = { -20, 0, 20, 0};
int[] y = { 20, 10, 20, -20};

GeneralPath p = new GeneralPath();
p.moveTo(x[0], y[0]);
for (int i = 1; i < x.length; i++) {
  p.lineTo(x[i], y[i]);
}
p.append(ovalShape);
p.closePath();

g2.translate(300, 100)
g2.draw(p);
```

REGION

- Obrysem regionu je uzavřená cesta
- Rozlišování uzavřená cesta vs. region je pouze myšlenkové:
  - Uzavřená cesta = neuvažujeme vnitřek
  - Region = uvažujeme vnitřek, obrys není zajímavý
- Region určen kombinováním primitiv
  - Množinové (booleovské) operace

REGION

- Obvyklé booleovské operace:
  - Sloučení (union)
  - Průnik (intersection)
  - Rozdíl (subtraction)
  - Doplněk (XOR)
    - v podstatě sloučení – průnik
- Problémy, které mohou nastat:
  - Vyšší výpočetní náročnost
  - Numerická nepředvídatelnost
    - Vše závisí na rasterizéru



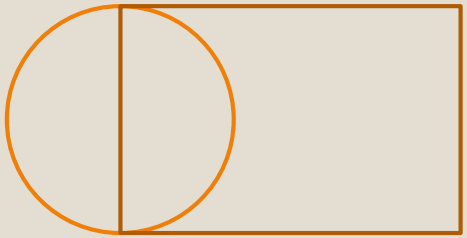
REGION

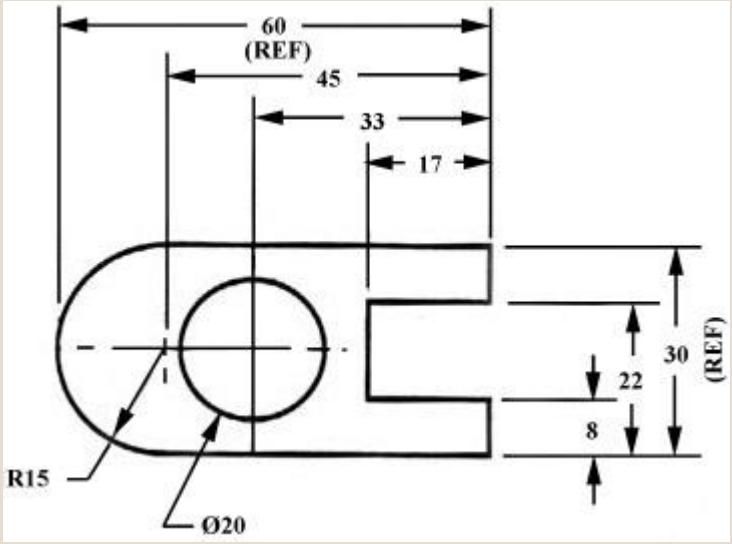
- Podporovaná primitiva:
  - Obdélník
  - Elipsa a kruh
  - Polorovina
  - …
- „Primitivem“ může být rovněž jiný region

REGION

- Regiony používány zejména pro:
  - Constructive area geometry (CAG)
  - Ořezávání (clipping)

Union (+)

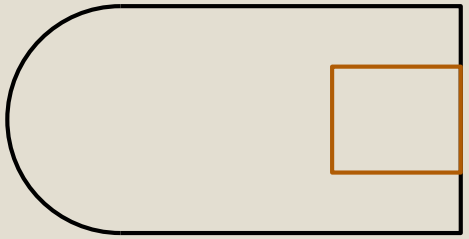


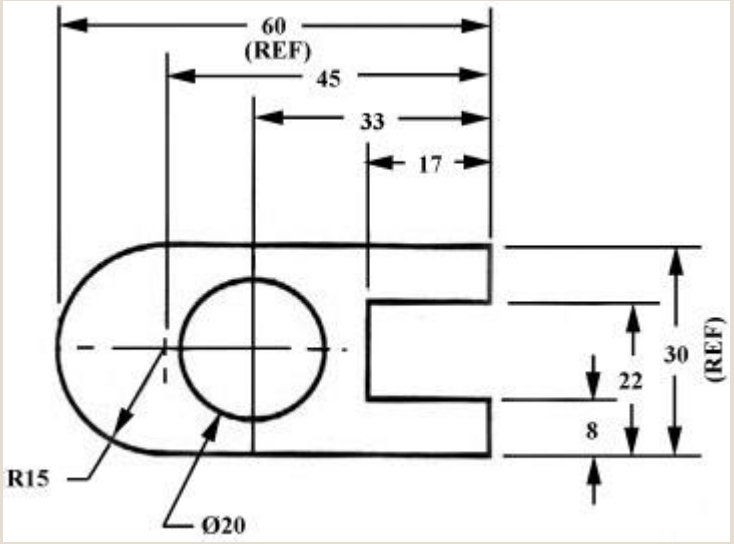


REGION

- Regiony používány zejména pro:
  - Constructive area geometry (CAG)
  - Ořezávání (clipping)

Subtraction (-)

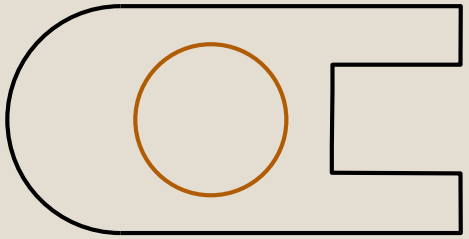


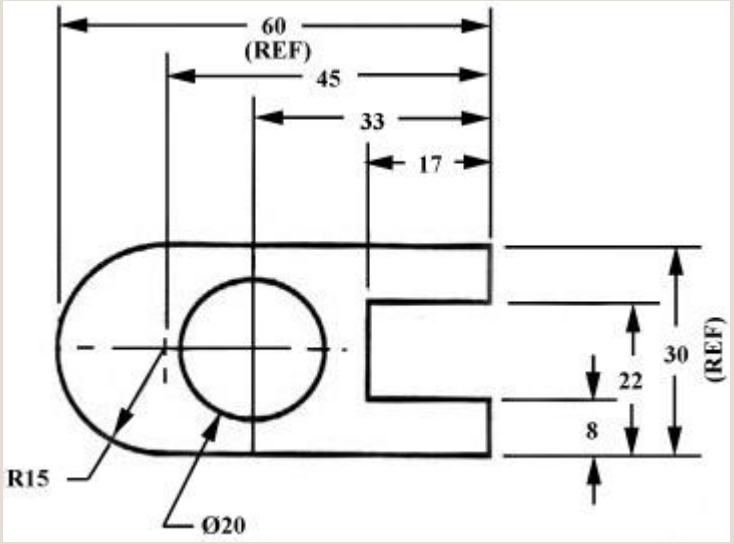


REGION

- Regiony používány zejména pro:
  - Constructive area geometry (CAG)
  - Ořezávání (clipping)
    - viz později

Subtraction (-)



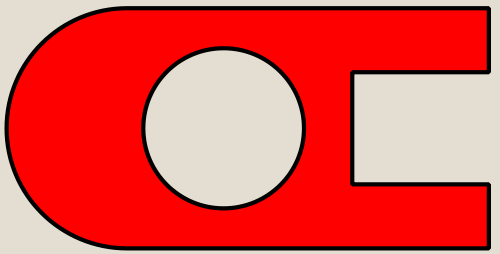


REGION V JAVĚ

- Region = třída `Area` (`java.awt.geom`)
- Konstruuje se nad nějakým počátečním tvarem (instance třídy implementující `Shape`)
  - Primitiva totožná s těmi z cesty, tj. Rectangle2D, Ellipse2D, …
- Výsledný region (objekt) lze vykreslit jako
  - vyplněný (`fill`) – plocha vnitřku regionu
  - nevyplněný (`draw`) – tvar tažením kreslícího nástroje po obrysu regionu

REGION V JAVĚ

```
Ellipse2D part1 = new Ellipse2D.Double();
part1.setFrame(0, 0, 30*mmToPx, 30*mmToPx);

Rectangle2D part2 = new Rectangle2D.Double();
part2.setRect(15*mmToPx, 0, 45*mmToPx, 30*mmToPx);

Area obj1 = new Area(part1);
obj1.add(new Area(part2));

…

Area obj2 = new Area(part3);
obj2.add(new Area(part4));

Area objFinal = new Area(obj1);
objFinal.subtract(obj2);

g2.draw(objFinal);
```



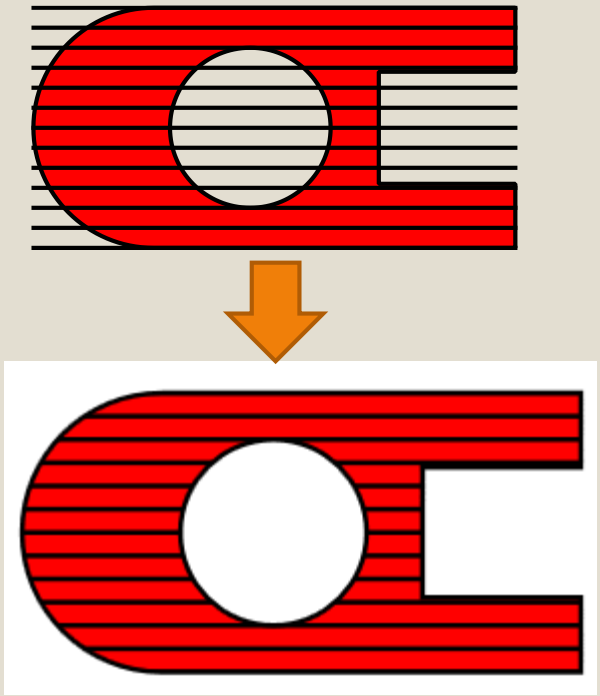
OŘEZÁVÁNÍ

- Ořezávání (clipping) odstraní veškerou grafiku mimo zvolenou ořezovou oblast
  - Oblast se specifikuje grafickému kontextu
- Ořezová oblast = region
- Význam:
  - Zjednodušení vykreslování složitých tvarů
  - Skládání vektorových obrazů do jednoho
- Může být výpočetně náročné
  - Efektivní jen pro speciální tvary ořezových oblastí

OŘEZÁVÁNÍ V JAVĚ

- Nastavení ořezové oblasti: metoda `setClip`

```
g2.setClip(objFinal);
g2.draw(pattern);
```



```
GeneralPath pattern = new GeneralPath();

double maxX = 60*mmToPx, maxY = 30*mmToPx;
double stepY = 2.5*mmToPx, curY = 0.0;
while (curY <= maxY) {
   Line2D line = new Line2D.Double();
   line.setLine(0, curY, maxX, curY);
   pattern.append(line, false);
   curY += stepY;
}
```

FORMÁTY VEKTOROVÉ GRAFIKY

- Vektorovou grafiku lze obvykle
  - Uložit na disk v nějakém vektorovém formátu
  - Načíst z nějakého vektorového formátu
  - Přehrát (zobrazit) na grafickém kontextu
    - Původní nebo odlišné měřítko
- Nejčastěji používané vektorové formáty:
  - Binární formáty
    - Windows Metafile (WMF), Enhanced Metafile (EMF), EMF+
    - Binary Application Markup Language (BAML)
    - Portable Document Format (PDF)
    - OpenDocument Graphics (ODG)
    - Adobe Illustrator Artwork (AI)
    - Adobe Flash (SWF)
    - Encapsulated PostScript (EPS)
    - CorelDraw (CDR)

FORMÁTY VEKTOROVÉ GRAFIKY

- Nejčastěji používané vektorové formáty:
  - Textové formáty
    - Encapsulated PostScript (EPS)
    - Extensible Application Markup Language (XAML)
    - Scalable Vector Graphics (SVG)

FORMÁTY VEKTOROVÉ GRAFIKY

Formát	+	-	Knihovny
AI	Profesionální úroveň pro DTP	Podpora jen v „profi“ DTP aplikacích	???
CDR		Pouze CorelDraw	Nejsou
ODG		Nepříliš rozšířen mimo OpenOffice	???
EPS, PDF	Univerzální	Není podpora v „běžných“ aplikacích	???
SWF	Animace, možnosti interakce, multimédia	Téměř nerozšířen	
BAML	Binární XAML	Na rozdíl od XAML není plošně používán	WPF

FORMÁTY VEKTOROVÉ GRAFIKY

Formát	+	-	Knihovny
DXF	3D, textový Autodesk specifikace	Komplexní	???
DWG	2D i 3D	Pouze Autodesk, nativní binární formát	OpenDWG hack
XAML	2D i 3D, Textový formát	Není rozšířen mezi aplikacemi	WPF

FORMÁTY VEKTOROVÉ GRAFIKY

Formát	+	-	Knihovny
WMF	Rozšířen	Zastaralý formát. Vhodný jen pro kliparty.	GDI, MFC, wxWidgets, …
EMF	Rozšířen (Windows)	Jeho verzi EMF+ programy neumí často dobře zpracovat	GDI, MFC, wxWidgets, GDI+, WinForms, WPF
SVG	Textový formát	Mnoho aplikací ho špatně interpretuje	Qt, *Batik (Java)*

FORMÁTY VEKTOROVÉ GRAFIKY

- Formáty lze mezi sebou konvertovat
  - Může vést ke ztrátě informace
- Např. InkScape

ULOŽENÍ VEKTOROVÉ GRAFIKY

- Obvyklý postup je:
  - Vytvořit pro zvolený podporovaný vektorový formát grafický kontext kompatibilní s kontextem konkrétního zařízení (pro něj je obsah určen)
  - Kreslit na vytvořený grafický kontext normálně
  - Uvolnit vytvořený grafický kontext (uzavření souboru)
- Nejjednodušším postupem pro textové formáty často může být:
  - Vlastní zápis do souboru dle specifikace

GDI+ (C++) A EMF

```
myMetafile = new Metafile(L"MyDiskFile.emf", hdc);

myGraphics = new Graphics(myMetafile);

myPen = new Pen(Color(255, 0, 0, 200));
myGraphics->DrawLine(myPen, 0, 0, 60, 40);
delete myPen;

delete myGraphics;
delete myMetafile; //uzavře soubor
```

BATIK A SVG

- Apache™ Batik SVG Toolkit
  - http://xmlgraphics.apache.org/batik/
  - Nadstavba na grafickou knihovnu AWT
  - Nejprve je nutné vytvořit hlavičku SVG dokumentu

```
import org.apache.batik.dom.GenericDOMImplementation;
import org.apache.batik.svggen.SVGGraphics2D;
import org.w3c.dom.DOMImplementation;
import org.w3c.dom.Document;

DOMImplementation domImpl =
      GenericDOMImplementation.getDOMImplementation();

// Create an instance of org.w3c.dom.Document.
String svgNS = "http://www.w3.org/2000/svg";
Document document = domImpl.createDocument(svgNS, "svg", null);
```

BATIK A SVG

- Poté vytvořit grafický kontext a kreslit do něj
- Konečně dokument uložit na disk

```
SVGGraphics2D svgGenerator = new SVGGraphics2D(document);
Graphics2D g2 = (Graphics2D)svgGenerator;

//kreslení

//uložení s povolením CSS stylů
try {
   svgGenerator.stream(new FileWriter("mujobrazek.svg"), true);
} catch (Exception e) {
   //Ošetření výjimky
}
```

WPF (C#) A XAML

- WPF je nativně vektorová knihovna
- Dva odlišné přístupy:
  - Shapes – primárně určeny pro výstup na obrazovku
  - Drawing Objects (Geometry) – obecnější, pokud se mají zobrazit na obrazovku, je nutné jejich „grafický kontext“ převést na Shape

WPF (C#) A XAML

```
//Vytvoření grafického kontextu
DrawingImage geometryImage = new DrawingImage();

//Kreslení (skládáním) na grafický kontext

//Převod na vizuální podobu
Image anImage = new Image();
anImage.Source = geometryImage;

Canvas myCanvas = new Canvas();
myCanvas.Children.Add(anImage);

using (var xmlWriter = new XmlTextWriter("s.xaml", Encoding.UTF8))
{
  xmlWriter.Formatting = System.Xml.Formatting.Indented;
  XamlWriter.Save(myCanvas, xmlWriter);
}
```

WPF (C#) A XAML

```
var part1 = new EllipseGeometry(…);
var part2 = new RectangleGeometry(…);
var obj1 = new CombinedGeometry(
      GeometryCombineMode.Union, part1, part2);
…
var objFinal = new CombinedGeometry(
      GeometryCombineMode.Exclude, obj1, obj2);

//Vytvoření šrafované výplně
var brush = new DrawingBrush();
…
//Vykreslení objektu
geometryImage.Drawing = new GeometryDrawing(
      brush, new new Pen(Brushes.Black, 2), objFinal);
```

SVG A XAML

D:\Education\UPG\Prednasky\Pr03_AdvVec\vystup.svg

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE svg PUBLIC '-//W3C//DTD SVG 1.0//EN'
          'http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/
<svg xmlns:xlink="http://www.w3.org/1999/xlink" style="fi
><!--Generated by the Batik Graphics2D SVG Generator--><de
  /><g
  ><defs id="defs1"
    ><clipPath clipPathUnits="userSpaceOnUse" id="clipPath
      ><path d="M102.0472 18.8976 C122.921 18.8976 139.842
      /></clipPath
    ></defs
    ><g style="fill:red; stroke-width:2; stroke:red;" tran
    ><path style="stroke:none;" d="M102.0472 18.8976 C122.
      /><path d="M102.0472 18.8976 C122.921 18.8976 139.84
      /><path d="M0 0 L226.7717 0 M0 9.4488 L226.7717 9.44
    /></g
  ></g
></svg
>
```

D:\Education\UPG\Prednasky\Pr03_AdvVec\vykres.xaml

```
<Canvas Name="myCanvas" xmlns="http://schemas.microsoft.com/winf
  <Image Stretch="None" HorizontalAlignment="Left">
    <Image.Source>
      <DrawingImage>
        <DrawingImage.Drawing>
          <GeometryDrawing>
            <GeometryDrawing.Brush>
              <DrawingBrush ViewportUnits="Absolute" Viewport="0
                <DrawingBrush.Drawing>
                  <DrawingGroup>
                    <DrawingGroup.Children>
                      <GeometryDrawing Brush="#FFFF0000" Pen="{x
                        <GeometryDrawing.Geometry>
                          <RectangleGeometry Rect="0,0,3.779527
                        </GeometryDrawing.Geometry>
                      </GeometryDrawing>
                      <GeometryDrawing Brush="#FF000000" Pen="{x
                        <GeometryDrawing.Geometry>
                          <RectangleGeometry Rect="0,4.72440944
                        </GeometryDrawing.Geometry>
                      </GeometryDrawing>
                      <GeometryDrawing Brush="#FFFF0000" Pen="{x
                        <GeometryDrawing.Geometry>
                          <RectangleGeometry Rect="0,12.2834645
                        </GeometryDrawing.Geometry>
                      </GeometryDrawing>
                    </DrawingGroup.Children>
                  </DrawingGroup>
                </DrawingBrush.Drawing>
```

NAČTENÍ VEKTOROVÉ GRAFIKY

- Načtení a zobrazení vektorové grafiky je v různých grafických knihovnách odlišné
- EMF v GDI, MFC, aj. nadstavby nad GDI
  - Načtení souboru s grafikou (`GetEnhMetaFile`)
  - Vykonání uložených grafických příkazů nad grafickým kontextem (`PlayEnhMetaFile`)
    - Nebo provádění po částech (`PlayEnhMetaFileRecord`)
- EMF(+) v GDI+, WinForms

```
Metafile metafile = new Metafile("prvni.emf");
graphics.DrawImage(metafile, 0, 0, 200, 200);
```

NAČTENÍ VEKTOROVÉ GRAFIKY

- SVG a Batik
  - Speciální komponenta `JSVGCanvas` pro zobrazení na obrazovce (v nějakém GUI)
  - API pro převod SVG na rastrový obrázek, který lze již zobrazit na příslušném grafickém kontextu

```
JFrame f = new JFrame();
f.setSize(640, 480);

JSVGCanvas canvas = new JSVGCanvas();
canvas.setURI(new File("d:\\vystup.svg").toURI().toString());
f.getContentPane().add(canvas);

f.setVisible(true);
```

NAČTENÍ VEKTOROVÉ GRAFIKY

- XAML a WPF
  - Třída XamlReader a rovnou se napojí

SVG A XAML

- Co mají společného
  - Postaveno nad XML
  - Relativně hodně rozšířené
  - Vektorová grafika
  - Možnost animované grafiky
- SVG
  - W3C standard (od 1999)
  - Není zaštítěno žádnou velkou komerční firmou
  - Primárně určeno pro webovou grafiku (HTML5)
  - Interpretováno

SVG A XAML

- XAML
  - Microsoft (uvedeno spolu s WPF v roce 2006)
  - Primárně určeno pro .NET aplikace
    - MS Windows, Windows Mobile, …
  - Lze použít webově
    - SilverLight
  - Součástí mohou být jiné prvky než jen grafika
    - UI, napojení na databáze, data binding, …
  - Elementy realizovány třídou na pozadí
    - BAML = binární forma, není interpretováno

SVG A XAML

- SVG:

```
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 105 95">
  <path fill="#7B4" d="M106,13c-21,9-31,4-40-2l-10,35c9,6,20"/>
  <path fill="#49c" d="M39,83c-9-6-18-10-39-2l10-35c21-9,31-4"/>
  <path fill="#E63" d="M51,42c-5-4-11-7-19-7c-6,0-12,1-20"/>
  <path fill="#FD5" d="M55,52c9,6,18,10,39,2l-10,35c-21,8-30"/>
</svg>
```

- XAML:

```
<Canvas
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation">
  <Path Data="M106,13c-21,9-31,4-40-2l-10,35c9,6,20" Fill="#7B4"/>
  <Path Data="M39,83c-9-6-18-10-39-2l10-35c21-9,31-4" Fill="#49c"/>
  <Path Data="M51,42c-5-4-11-7-19-7c-6,0-12,1-20" Fill="#E63"/>
  <Path Data="M55,52c9,6,18,10,39,2l-10,35c-21,8-30" Fill="#FD5"/>
</Canvas>
```

SVG A XAML

- Hlavička souborů .SVG a .XAML
  - `<?xml version="1.0" encoding="UTF-8"?>`
  - Není nezbytně nutná
    - Irelevantní, pokud vektorovou grafiku vkládám do jiného webového dokumentu
- Kořenový (root) element SVG

```
<svg xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink" version="1.1"
width="800px" height="800px" viewBox="0 0 800 800"
>
        <Title>Název obrázku</Title>
        <!-- Zde je obsah -->
</svg>
```

SVG A XAML

- Kořenový (root) element XAML

```
<Page
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
       width="800px" height="800px" Title="Název obrázku" >
       <Canvas>
              <!-- Zde je obsah -->
       </Canvas>
</Page>
```

SVG A XAML

- Alternativa 1
  - Funkční, ale není doporučeno

```
<Canvas
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
       width="800px" height="800px" >
       <!-- Zde je obsah -->
</Canvas>
```

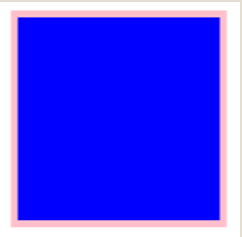
- Alternativa 2
  - Určeno pouze při vkládání grafiky do dokumentu s nadefinovanými jmennými prostory (schemas)

```
<Canvas width="800px" height="800px" >
       <!-- Zde je obsah -->
</Canvas>
```

SVG A XAML

- Obsahem je seznam grafických elementů
  - viz např. http://www.w3schools.com/svg/
  - http://msdn.microsoft.com/en-us/library/ms744948.aspx
- Obdélník

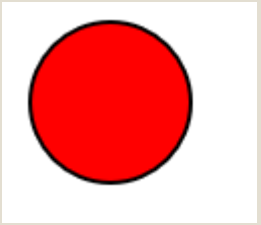
```
<rect x="50" y="20" width="150" height="150" fill="blue"
stroke="pink" stroke-width="5" />

<Rectangle Canvas.Left="50" Canvas.Top="20" Width="150"
Height="150" Fill="Blue" Stroke="Pink" StrokeThickness="5"/>
```



SVG A XAML

- Kružnice a elipsa

```
<circle cx="50" cy="20" r="40" fill="red" stroke="black" />
<ellipse cx="50" cy="20" rx="40" ry="50" fill="red" />
```



```
<Ellipse Canvas.Left="60" Canvas.Top="10" Width="40"
Height="40" Fill="Red" Stroke="Black" />
```

- Úsečka

```
<line x1="0" y1="0" x2="200" y2="200" stroke="black" stroke-width="2"/>

<Line X1="0" Y1="0" X2="200" Y2="200" Stroke="Black" StrokeThickness="2"/>
```

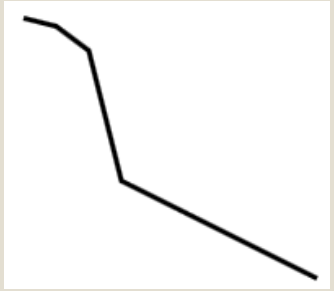
SVG A XAML

- Polygon

```
<polygon points="200,10 250,190 160,210" fill="green"/>

<Polygon Points="200,10 250,190 160,210" Fill="Green"/>
```

- Lomená čára

```
 <polyline points="20,20 40,25 60,40 80,120 120,140 200,180"
        fill="none" stroke="black" stroke-width="3" />

<Polyline Points="20,20 40,25 60,40 80,120 120,140 200,180"
        Stroke="black" StrokeThickness="3" />
```



SVG A XAML

- Text

```
<text x="0" y="15" fill="red" font-size="36" font-
family="Segoe UI Light">Hello SVG</text>

<TextBlock Canvas.Left="0" Canvas.Top="15" Foreground="Red"
FontSize="36" FontFamily="Segoe UI Light">Hello from
XAML</TextBlock>
```

SVG A XAML

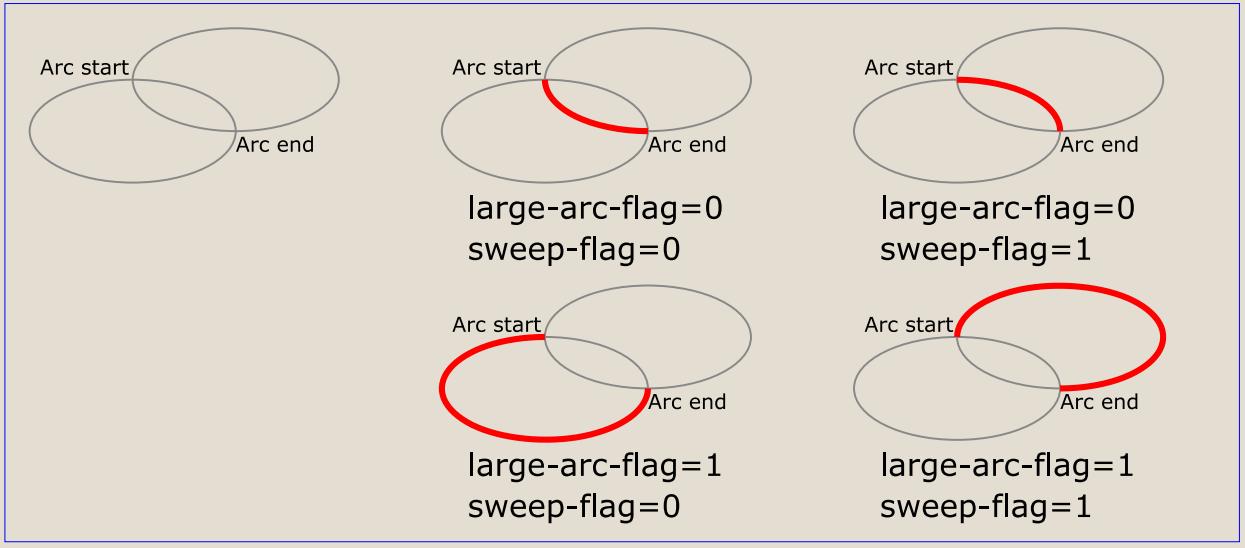
- Cesta SVG:
  - `<path d="popis cesty"/>`
- Cesta XAML:
  - `<Path Data="popis cesty">`
- Popis cesty zapsán posloupností příkazů a jejich parametrů
  - Parametry oddělovány mezerou nebo čárkou

SVG A XAML

- Příkazy:
  - M x y – posun na bod (x, y)
  - L x y – čára z aktuální pozice do bodu (x, y)
  - H x – horizontální čára do bodu (x, aktuální y)
  - V y – vertikální čára do bodu (aktuální x, y)
  - Q x2 y2 x3 y3 – kvadratická Bezierova křivka
  - C x2 y2 x3 y3 x4 y4 – kubická Bezierova křivka
  - S x3 y3 x4 y4 – kubická Bezierova křivka hladce navázána na předchozí křivku
  - Z – automatické uzavření cesty

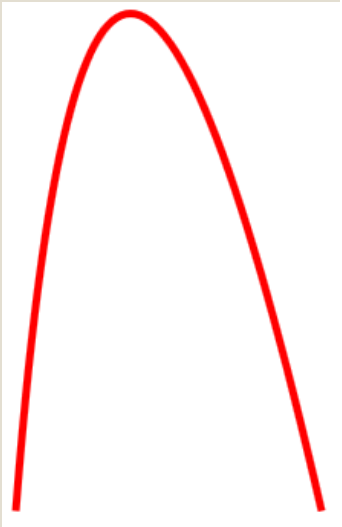
SVG A XAML

- A rx ry angle large-arc-flag sweep-flag x y – eliptický oblouk z aktuálního bodu do bodu (x, y)
  - elipsa má velikost poloos rx, ry a je otočena o úhel angle
  - aktuální bod a cílový bod leží na elipse
  - mohou existovat dvě elipsy splňující tyto parametry
    - poslední dva parametry dodefinují, co vlastně chci



SVG A XAML

- Příklad:
  - `<path d="M 100 350 Q 150 -300 300 350" stroke="red"  stroke-width="5" fill="none"/>`
  - `<Path Data="M 100 350 Q 150 -300 300 350" Stroke="White"  StrokeThickness="5"/>`



SVG A XAML

- XAML má alternativní definici cesty
  - Přehlednější
  - Snadnější úprava
  - Delší zápis (více Bytů)

```
<Path Stroke="Black" StrokeThickness="1"
  Data="M 10,100 L 100,100 100,50 Z M 10,10 100,10 100,40 Z" />
```

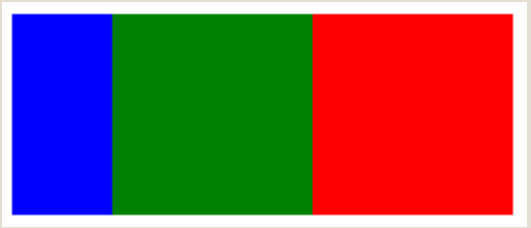
```
<Path Stroke="Black" StrokeThickness="1">
  <Path.Data>
    <PathGeometry>
      <PathGeometry.Figures>
        <PathFigureCollection>
          <PathFigure IsClosed="True" StartPoint="10,100">
            <PathFigure.Segments>
              <PathSegmentCollection>
                <LineSegment Point="100,100" />
                <LineSegment Point="100,50" />
              </PathSegmentCollection>
            </PathFigure.Segments>
          </PathFigure>
          <PathFigure IsClosed="True" StartPoint="10,10">
            <PathFigure.Segments>
              <PathSegmentCollection>
                <LineSegment Point="100,10" />
                <LineSegment Point="100,40" />
              </PathSegmentCollection>
            </PathFigure.Segments>
          </PathFigure>
        </PathFigureCollection>
      </PathGeometry.Figures>
    </PathGeometry>
  </Path.Data>
</Path>
```

SVG A XAML

- XAML podporuje vložení dalších prvků včetně rastrového obrázku nebo vektorové grafiky specifikované jako DrawingGeometry
- Seskupování grafických prvků
  - Snadnější manipulace
  - SVG: vnořený element svg nebo g
    - Element g je preferován
  - XAML: vnořený Canvas
    - Případně další možnosti (např. StackPanel, …)

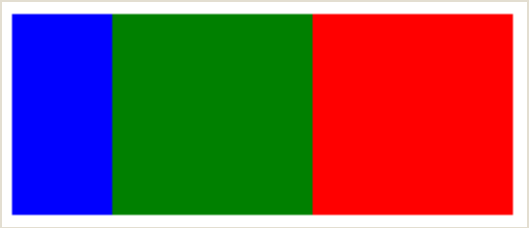
SVG A XAML

```
<rect x="10" y="10" height="100" width="100" stroke="none" fill="blue"/>
<svg x="50">
  <rect x="10" y="10" height="100" width="100" stroke="none" fill="green"/>
  <rect x="110" y="10" height="100" width="100" stroke="none" fill="red"/>
</svg>

<rect x="10" y="10" height="100" width="100" stroke="none" fill="blue"/>
<g transform="translate(50,0)">
  <rect x="10" y="10" height="100" width="100" stroke="none" fill="green"/>
  <rect x="110" y="10" height="100" width="100" stroke="none" fill="red"/>
</g>
```



SVG A XAML

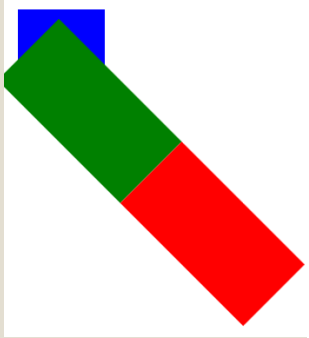
```
<Rectangle Canvas.Left="10" Canvas.Top="10" Height="100" Width="100" Fill="Blue"/>
<Canvas>
  <Canvas.RenderTransform>
    <TranslateTransform X="50"/>
  </Canvas.RenderTransform>

  <Rectangle Canvas.Left="10" Canvas.Top="10" Height="100" Width="100" Fill="Green"/>
  <Rectangle Canvas.Left="110" Canvas.Top="10" Height="100" Width="100" Fill="Red"/>
</Canvas>
```



SVG A XAML

- Afinní transformace SVG:
  - Atribut `transform="`*`posloupnost funkcí`*`"`
    - Funkce oddělované čárkou
  - Možné transformační funkce:
    - translate(dx,dy) – posun o dx, dy
    - rotate(uhel) – rotace o zadaný úhel (ve stupních)
    - scale(sx,sy) – změna měřítka

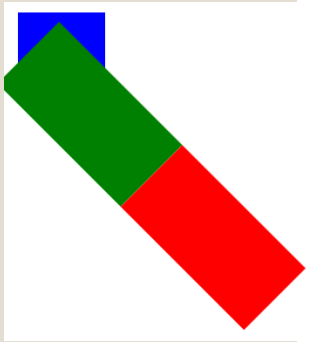
```
<g transform="translate(50,0),rotate(45),scale(2,1)">
```



SVG A XAML

- Afinní transformace XAML:
  - Element `Canvas.RenderTransform`
    - Obsahuje buď jeden transformační element nebo celou skupinu transformačních elementů (`TransformGroup`)
    - Možné transformační elementy: `TranslateTransform`, `RotateTransform`, `ScaleTransform`

```
<Canvas>
  <Canvas.RenderTransform>
     <TransformGroup>
       <ScaleTransform ScaleX="2" ScaleY="1"/>
        <RotateTransform Angle="45"/>
        <TranslateTransform X="50"/>
      </TransformGroup>
  </Canvas.RenderTransform>
```



KONEC

- Příště: Interaktivní vektorová grafika