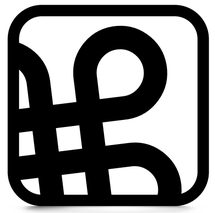


Macoun

Bézier-Pfade: Theorie und Praxis

Martin Winter

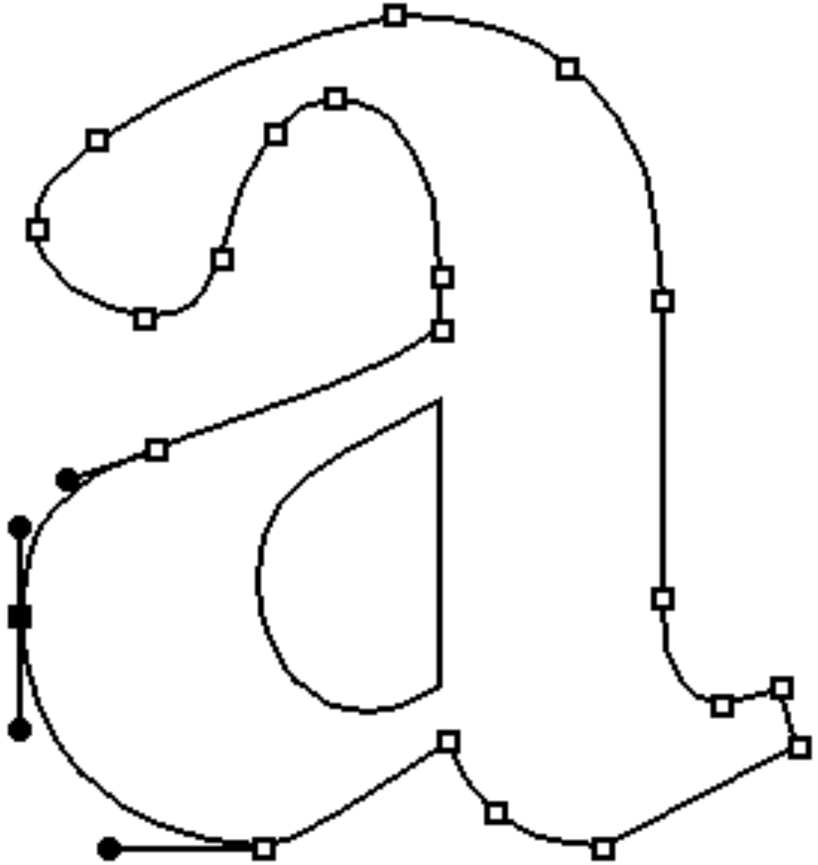
Ablauf

- Was sind Bézierkurven?
- Wozu Bézierkurven?
- Bézierkurven anwenden
- Tips & Tricks
- Praxisbeispiel

Was sind Bézierkurven?

Was sind Bézierkurven?

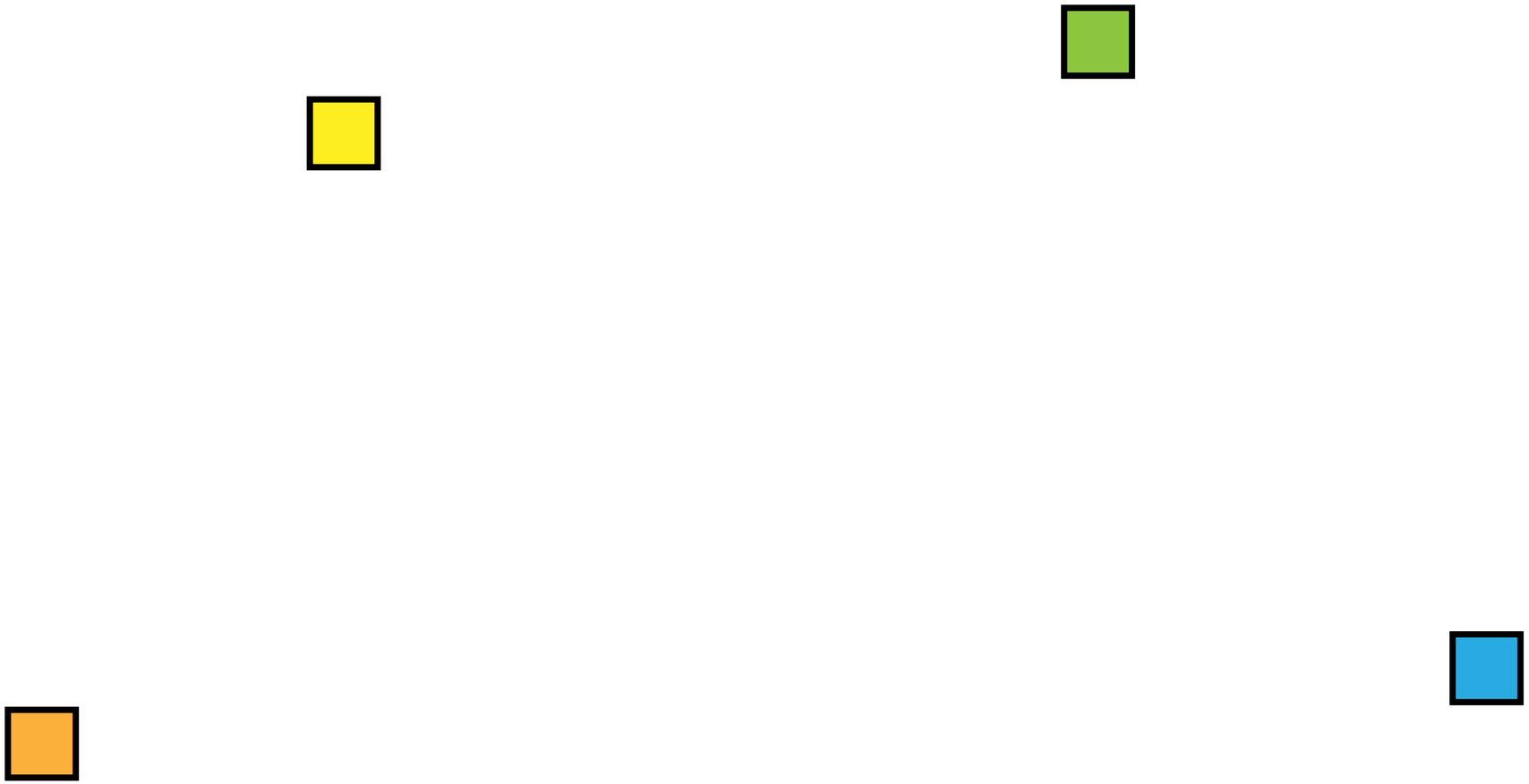
- »Vektorgrafik«
- parametrische Kurven: $0 \ldots t \ldots 1$
- Kontrollpunkte: $p_0 \ldots p_n$
- konvexe Hülle
- Grad n
- rekursiv konstruierbar (de Casteljau)



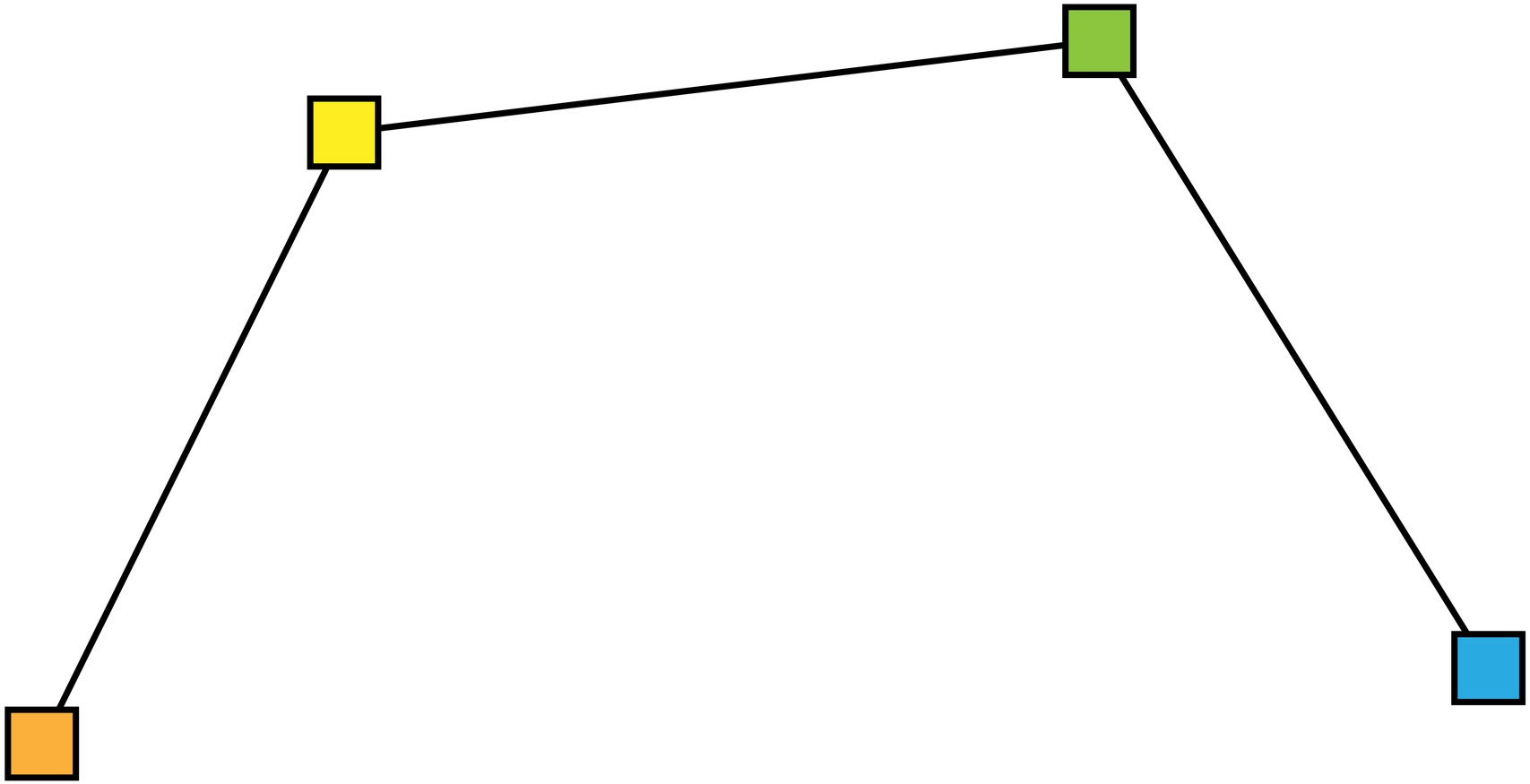
Kurven, Splines, Pfade

- Kurve
- Spline/Pfad
- Subpfad
- Füllregeln
  - Non-Zero Winding Rule
  - Even-Odd Winding Rule

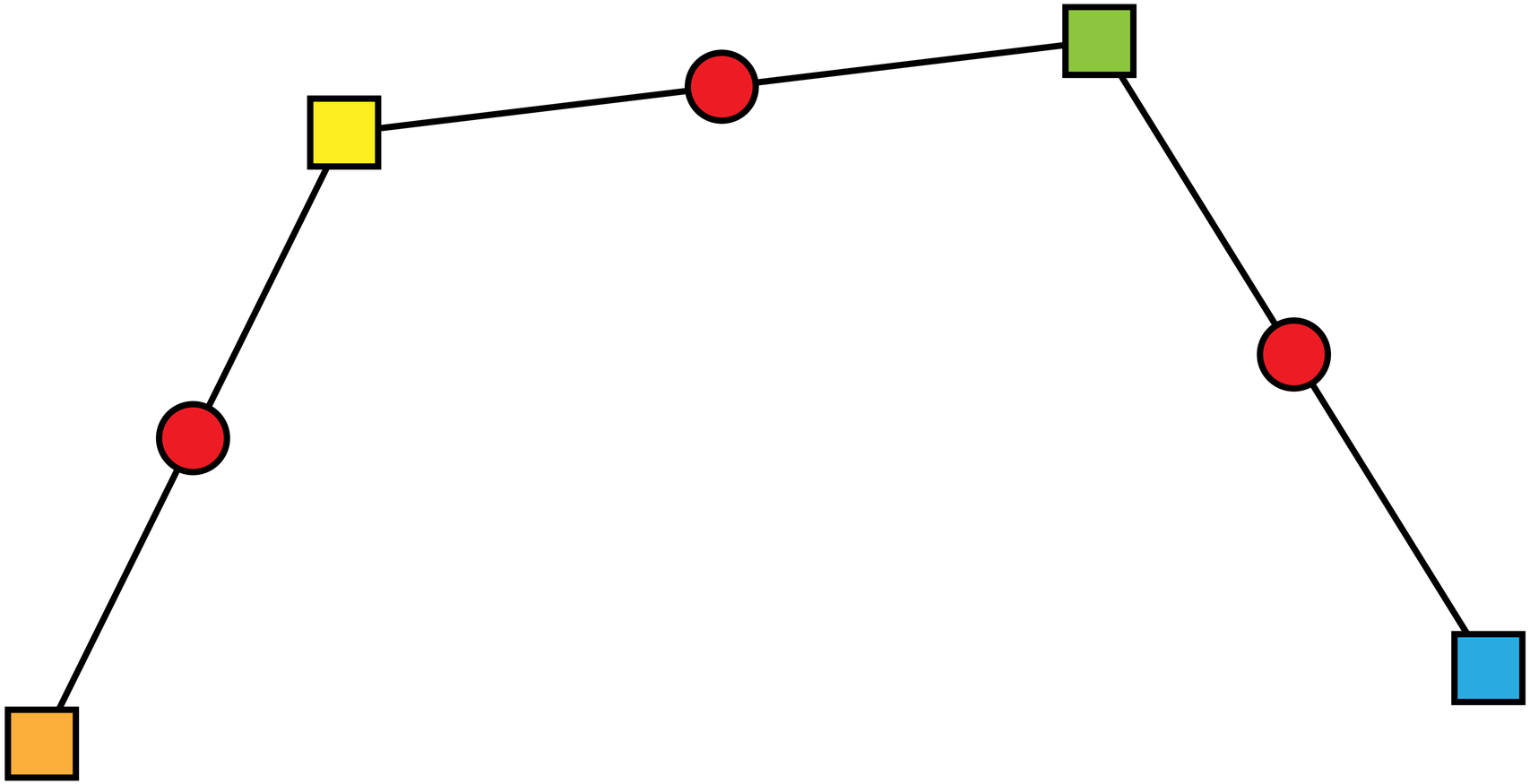
De-Casteljau-Algorithmus



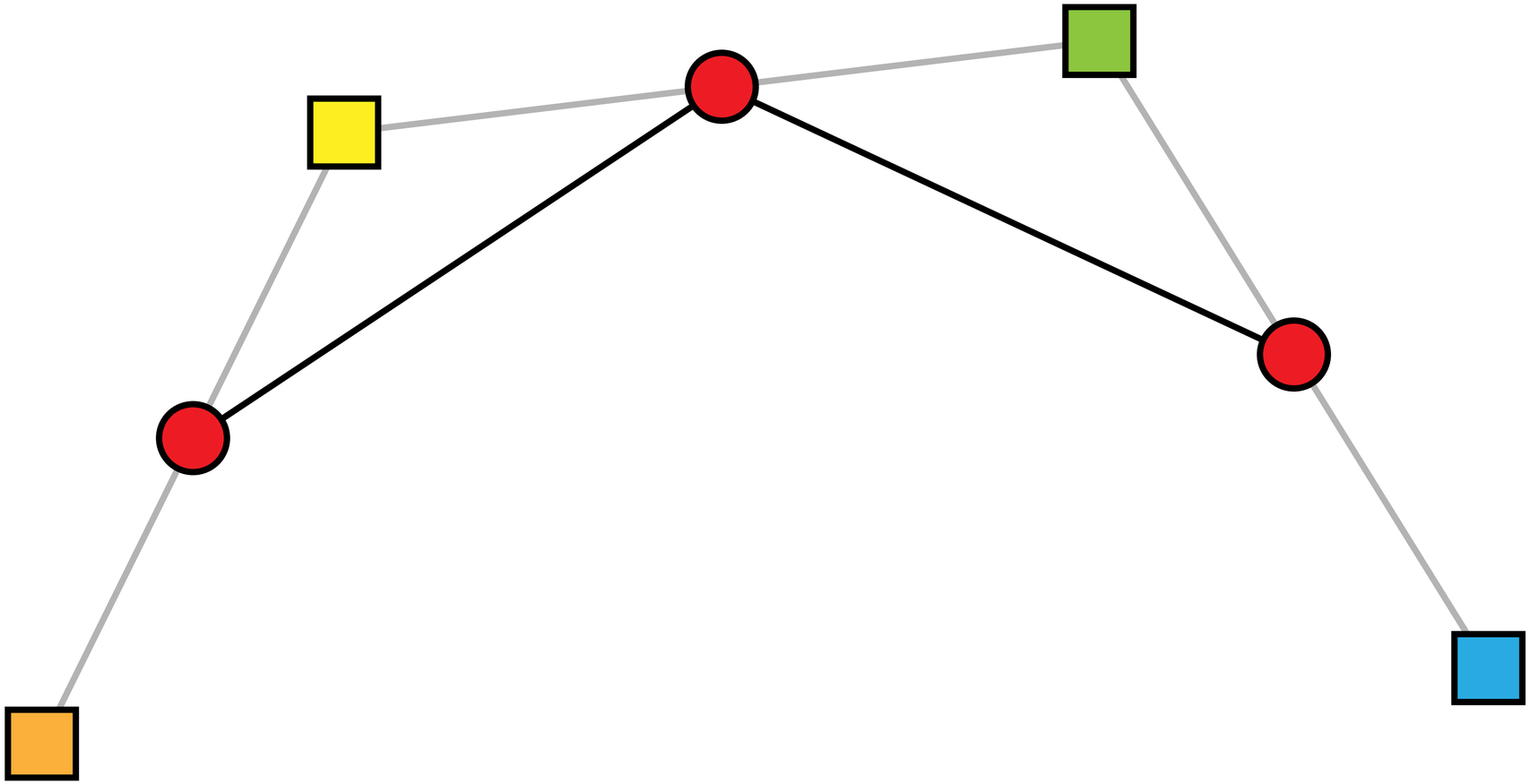
De-Casteljau-Algorithmus



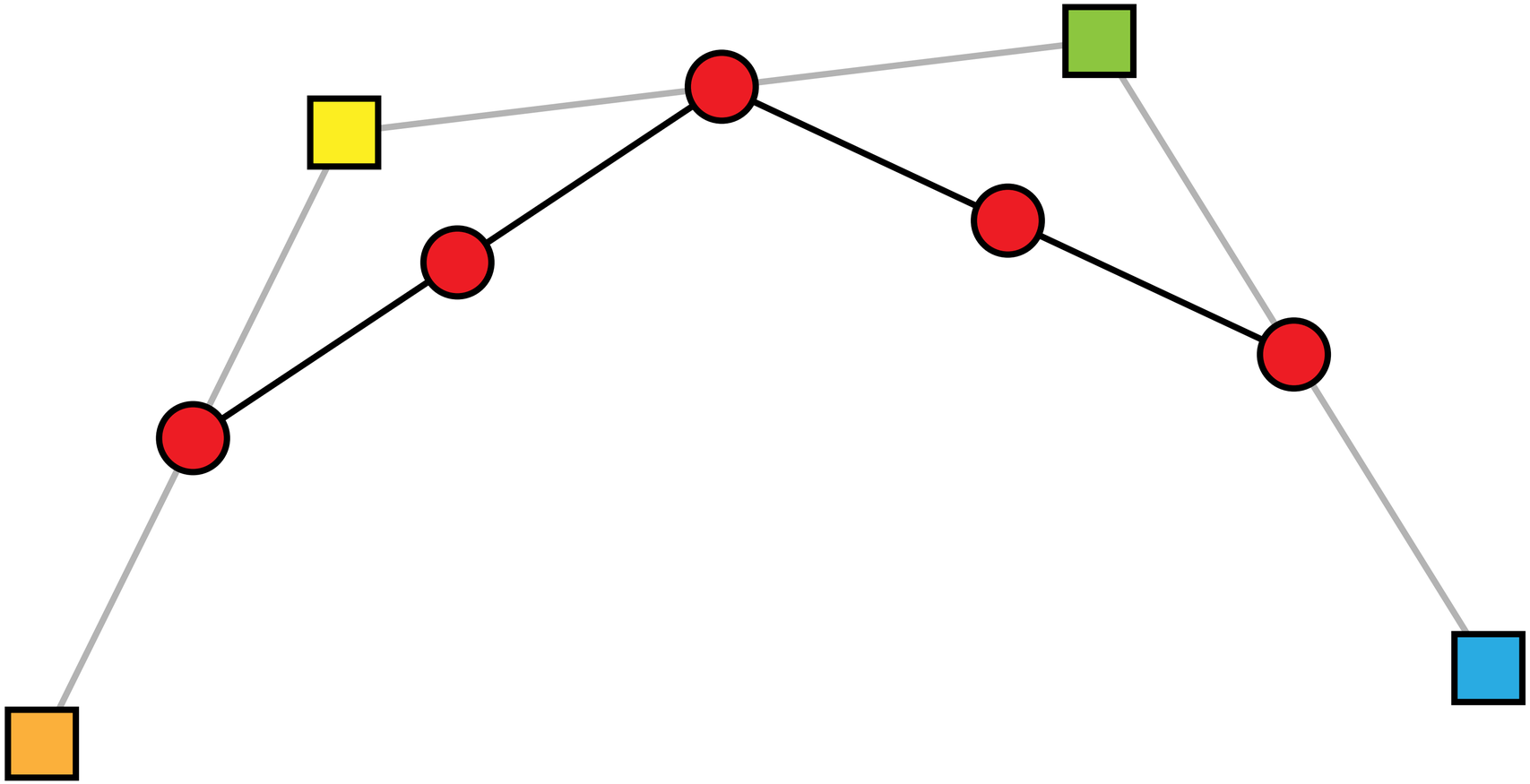
De-Casteljau-Algorithmus



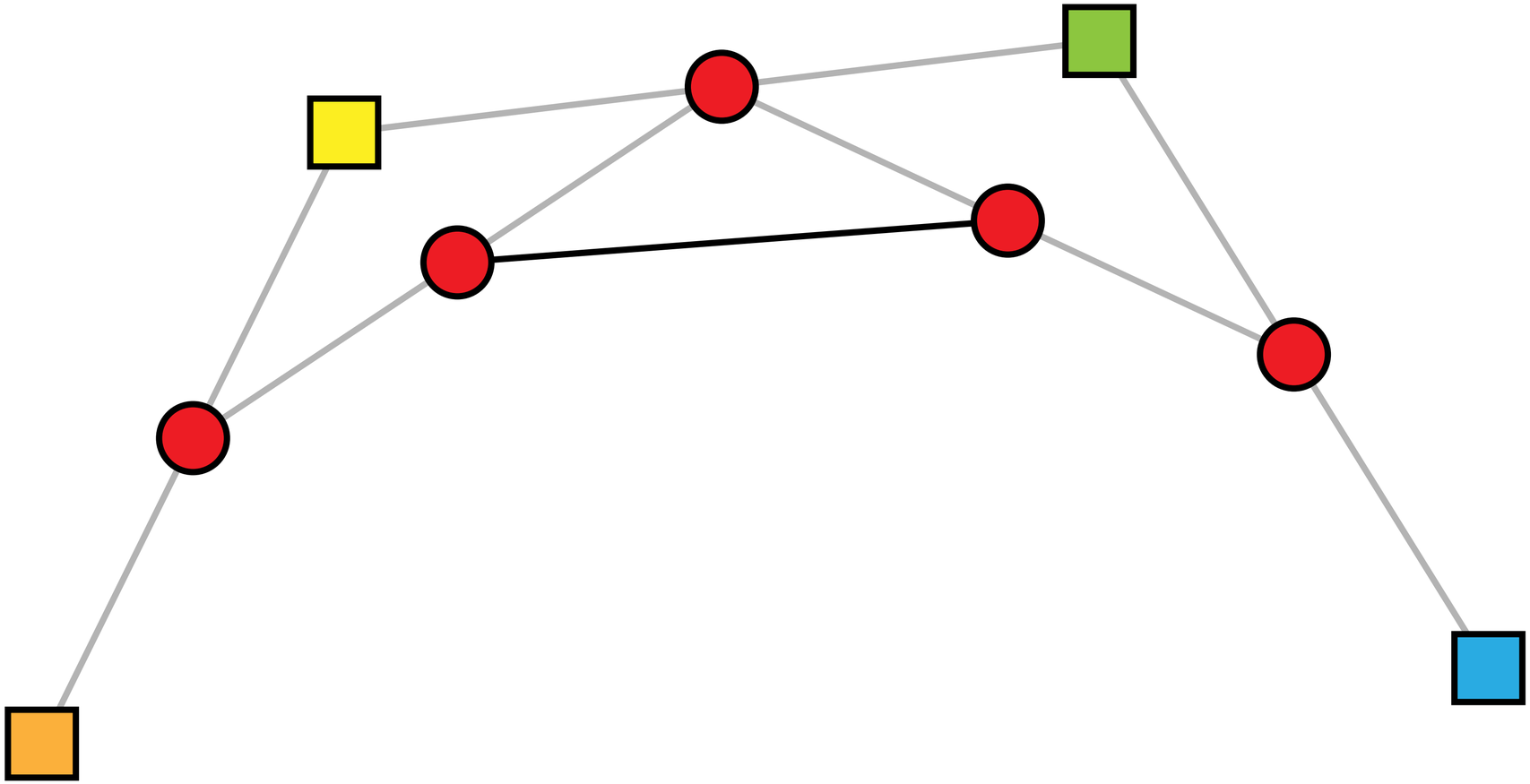
De-Casteljau-Algorithmus



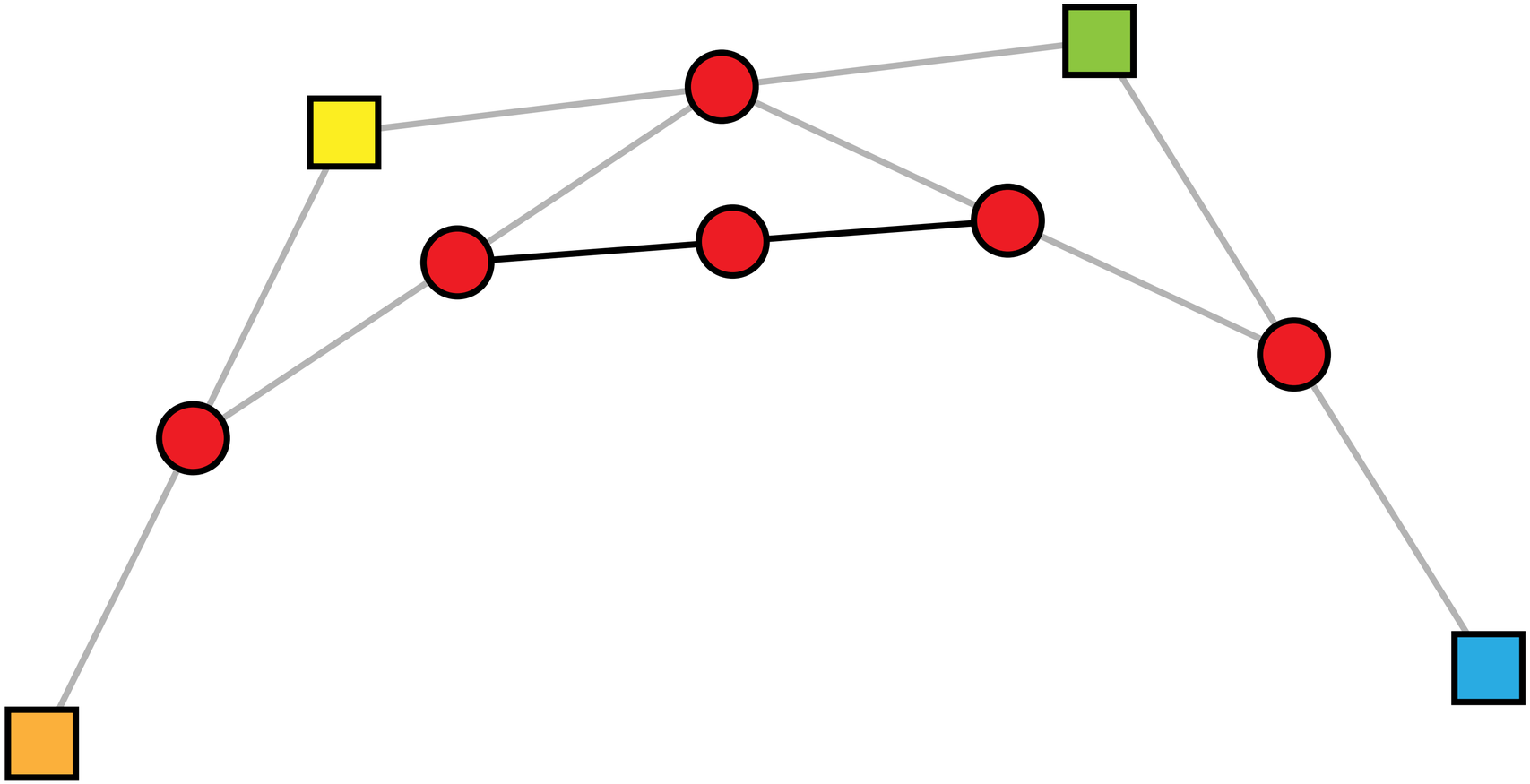
De-Casteljau-Algorithmus



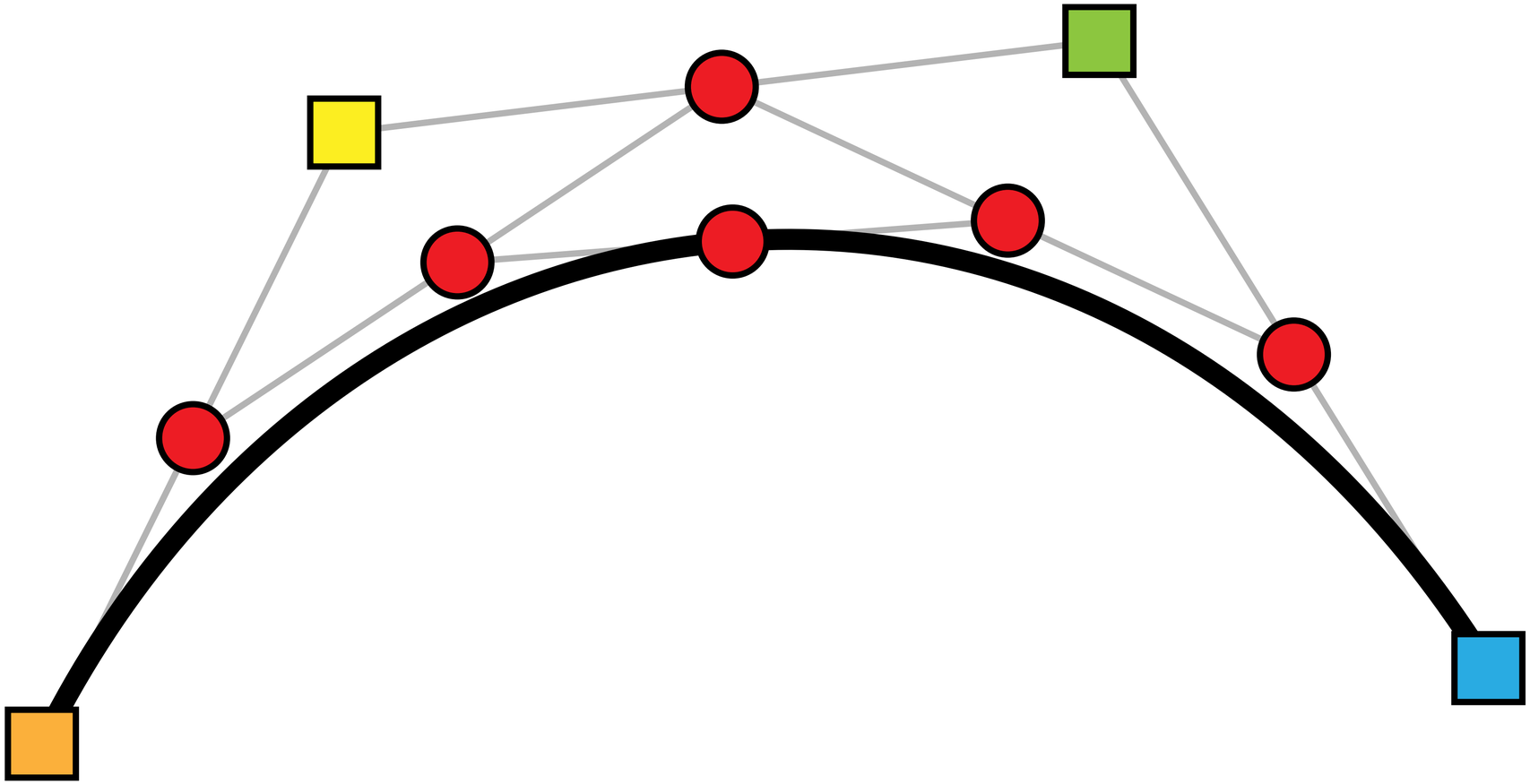
De-Casteljau-Algorithmus



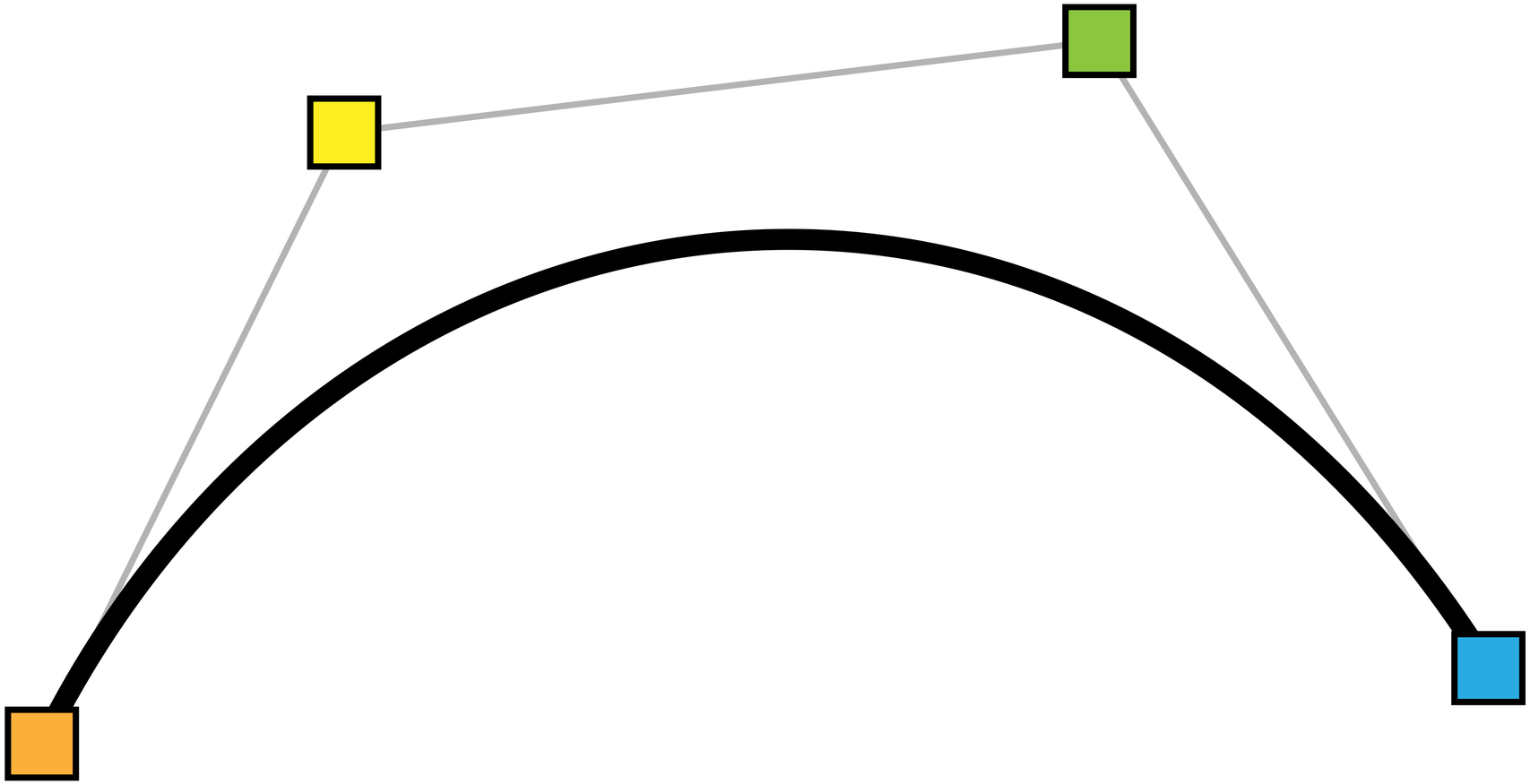
De-Casteljau-Algorithmus



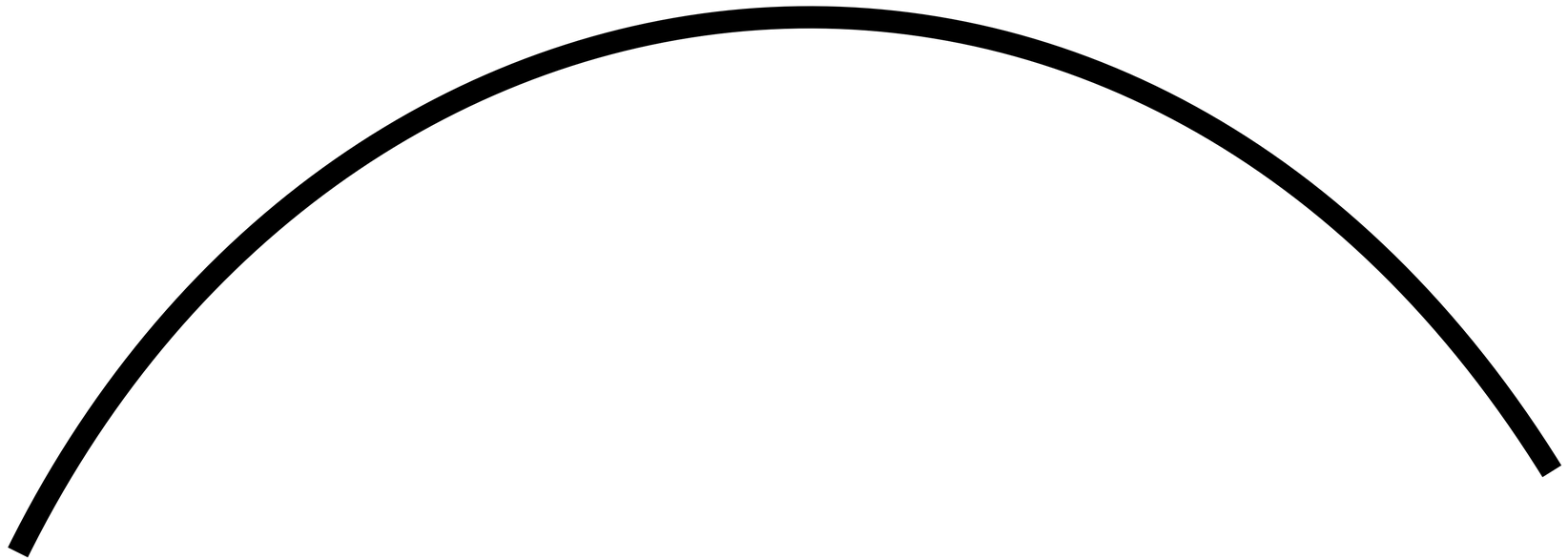
De-Casteljau-Algorithmus



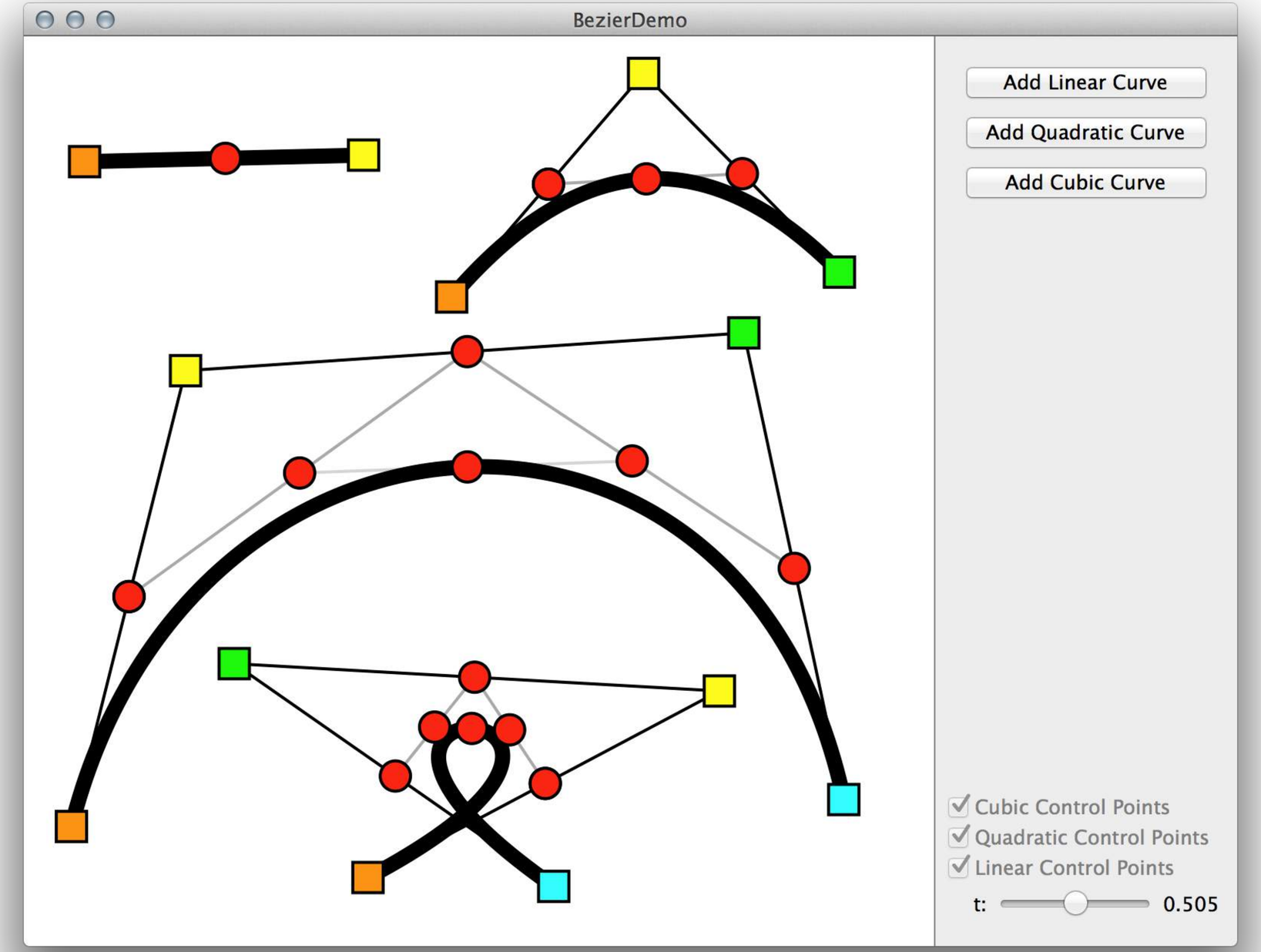
De-Casteljau-Algorithmus



De-Casteljau-Algorithmus



Demo


BezierDemo
Add Linear Curve
Add Quadratic Curve
Add Cubic Curve
Cubic Control Points
Quadratic Control Points
Linear Control Points
t:
0.505

De-Casteljau-Algorithmus

- https://github.com/martinwinter/bezierdemo/
- https://www.jasondavies.com/animated-bezier/
- erlaubt außerdem Splitten von Pfaden

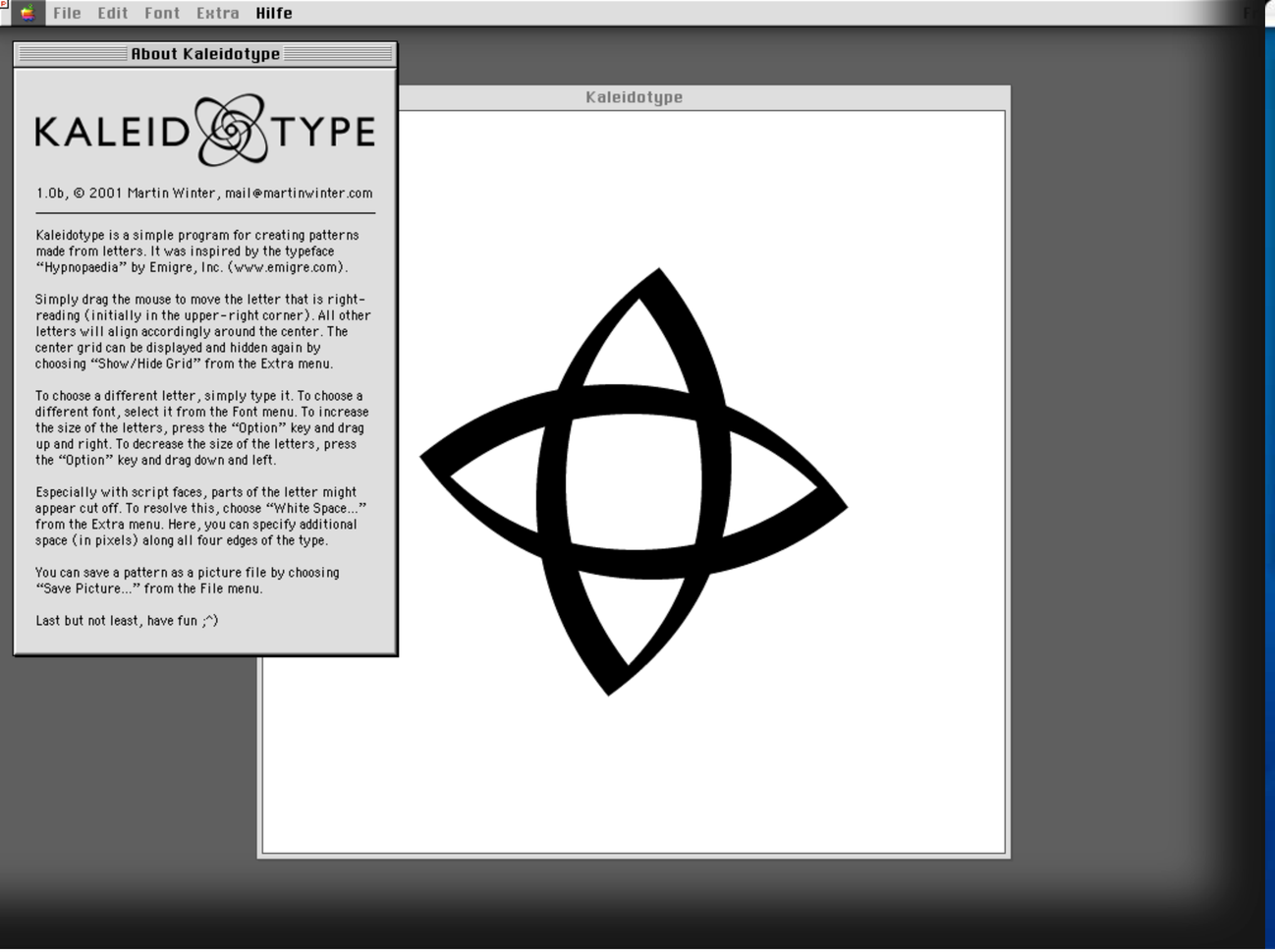
Wozu Bézierkurven?

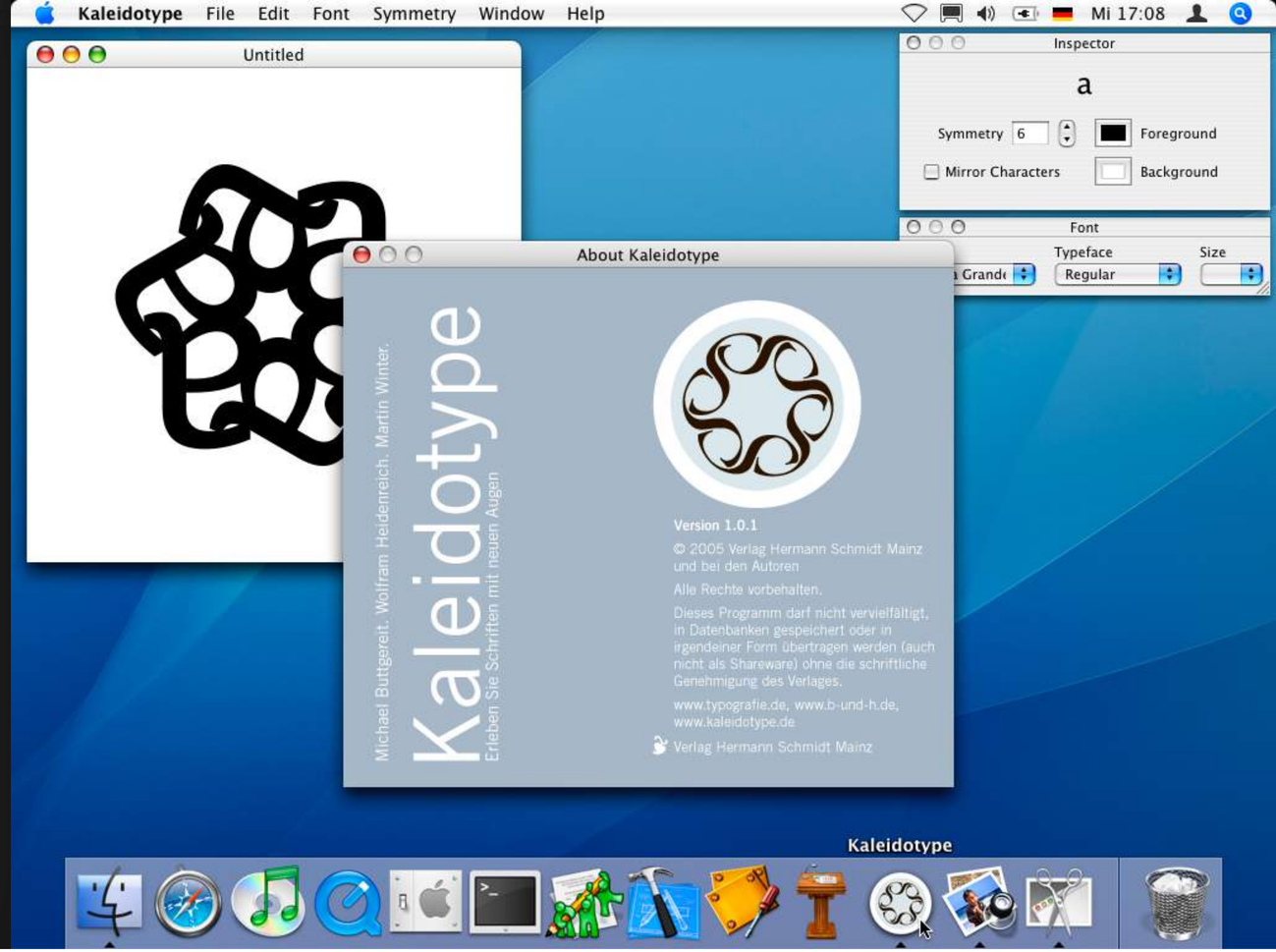
Wozu Bézierkurven?

- französische Automobilindustrie, 1960er Jahre
- Paul de Casteljau, Citroën
- Pierre Bézier, Renault

Wozu Bézierkurven?

- auflösungsunabhängig
  - Schrift, skalierbare Icons
- weiche Übergänge
  - Animationskurven, Visualisierungen des Static Analyzers
- effizient
  - geringer Speicherbedarf
  - einfach und schnell zu zeichnen





Pixel vs. Bézierkurven

Bézierpfade anwenden

Bézierpfade anwenden

Bézierpfade anwenden

- vieles gleich, einiges unterschiedlich
  - `-[NSBezierPath elementAtIndex:]`
  - `-[UIBezierPath addQuadCurveToPoint:controlPoint:]`
- aktueller Punkt
- aktueller Subpath

```
NSBezierPath *path =
    [NSBezierPath bezierPath];
[path moveToPoint:point];
[path curveToPoint:point2
     controlPoint1:cp1
     controlPoint2:cp2];
[path closePath];
```

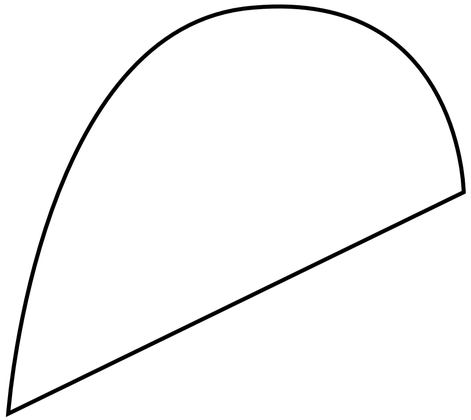
```
UIBezierPath *path =
    [UIBezierPath bezierPath];
[path moveToPoint:point];
[path addCurveToPoint:point2
        controlPoint1:cp1
        controlPoint2:cp2];
[path closePath];
```

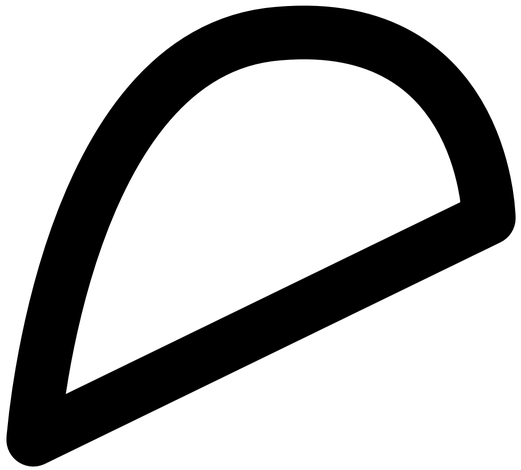
```
CGMutablePathRef path =
    CGPathCreateMutable();
CGPathMoveToPoint(path, NULL,
    point.x, point.y);
CGPathAddCurveToPoint(path, NULL,
    cp1.x, cp1.y, cp2.x, cp2.y,
    point2.x, point2.y);
CGPathCloseSubpath(path);
. . .
CGPathRelease(path);
```

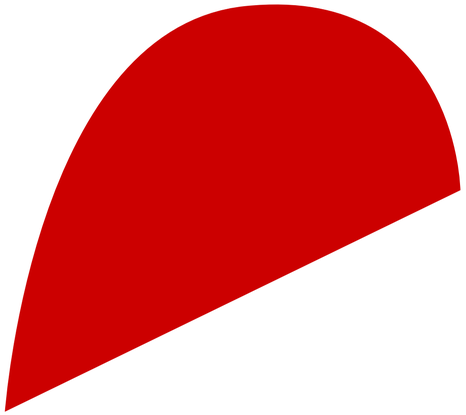
```
CGContextBeginPath(context);
CGContextMoveToPoint(context,
    point.x, point.y);
CGContextAddCurveToPoint(context,
    cp1.x, cp1.y, cp2.x, cp2.y,
    point2.x, point2.y);
CGContextClosePath(context);
```

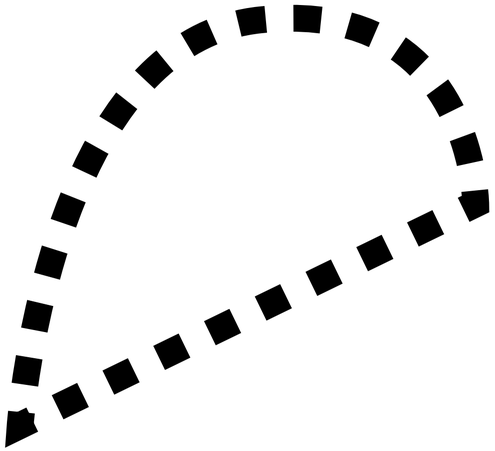
Bézierpfade anwenden

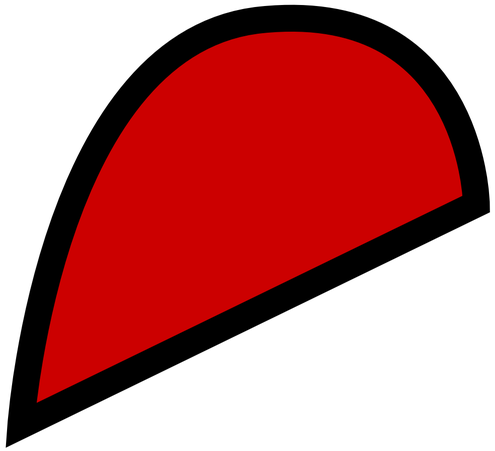
- Pfade an sich haben keine Attribute
  - Pfad = Bewegung des Stifts
- können auf viele Arten gezeichnet bzw. verwendet werden
  - Aussehen = welcher Stift
- Linienstärke, Linienfarbe, Füllfarbe
- Linienenden, Ecken, Strichelung
- Beschneidungspfad für Bilder und andere Pfade











```objc
[[NSColor redColor] setFill];
[[NSColor whiteColor] setStroke];

// Use path multiple times.
[path fill];
[path stroke];
```

```objc
[[UIColor redColor] setFill];
[[UIColor whiteColor] setStroke];

// Use path multiple times.
[path fill];
[path stroke];
```

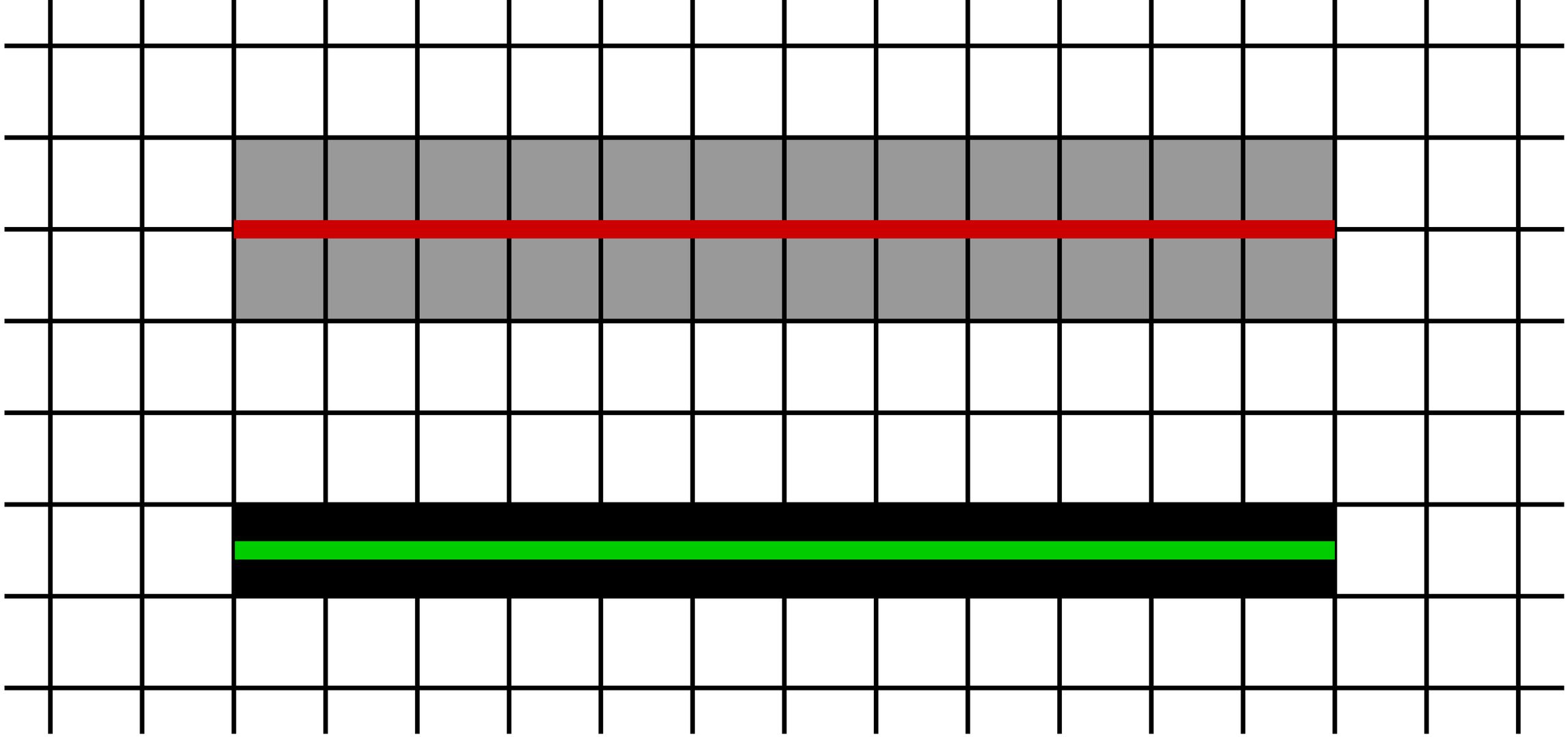
```objc
CGContextSetFillColorWithColor(context, cgRedColor);
CGContextSetStrokeColorWithColor(context, cgWhiteColor);

// Current path is reset by each operation.
CGContextAddPath(context, path);
CGContextFillPath(context, path);
CGContextAddPath(context, path);
CGContextStrokePath(context, path);

// Convenience function for fill + stroke.
CGContextAddPath(context, path);
CGContextDrawPath(context, path, kCGPathFillStroke);
```

Demo

Bézierpfade anwenden

- https://github.com/martinwinter/bezierdemo2/
- ausführliche Erklärungen und Beispiele:
  Apple Quartz 2D Programming Guide > Paths

Tips & Tricks



http://orangejuiceliberationfront.com/
are-your-rectangles-blurry-pale-and-have-rounded-corners/

Hit Testing

```
// Detects on path and in enclosed area
// Always uses non-zero winding rule
[path containsPoint:point]
```

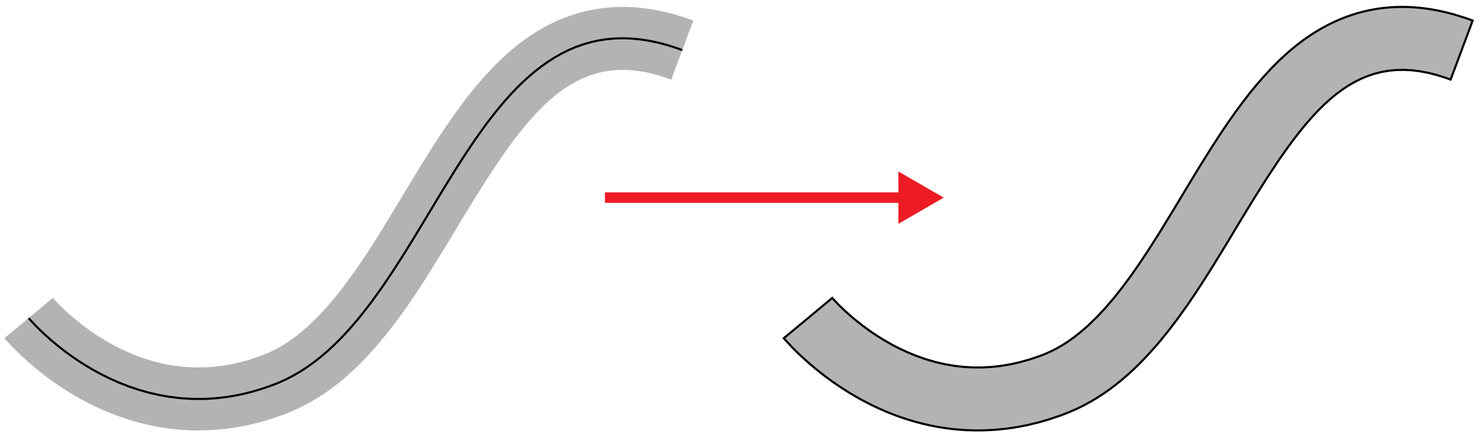
```
[path containsPoint:point]
```

```
// Choose winding rule
CGPathContainsPoint(
    path,
    transform,
    point,
    eoFill)
```

```
// Choose not just winding rule but
// drawing mode (stroke/fill)
CGContextPathContainsPoint(
    context,
    point,
    drawingMode)
```

Hit Testing

```
// Convert linear path into a shape
CGPathCreateCopyByStrokingPath(
    path, transform,
    lineWidth, lineCap, lineJoin,
    miterLimit)
```



Affine Transformationen

- einfach Kontrollpunkte transformieren
- viele Funktionen enthalten Transformations-Parameter
- häufig angewendet: vertikale Spiegelung
  - oft besser, den gesamten Kontext zu transformieren

```
CGContextTranslateCTM(context, 0.0, viewHeight);
CGContextScaleCTM(context, 1.0f, -1.0f);
```

Pfade für Schriftzeichen

- OS X: `-[NSBezierPath appendBezierPathWithGlyph:inFont:]`
  - Glyphen mit `NSLayoutManager` erzeugen
- plattformübergreifend: Core Text

Pfade für Schriftzeichen

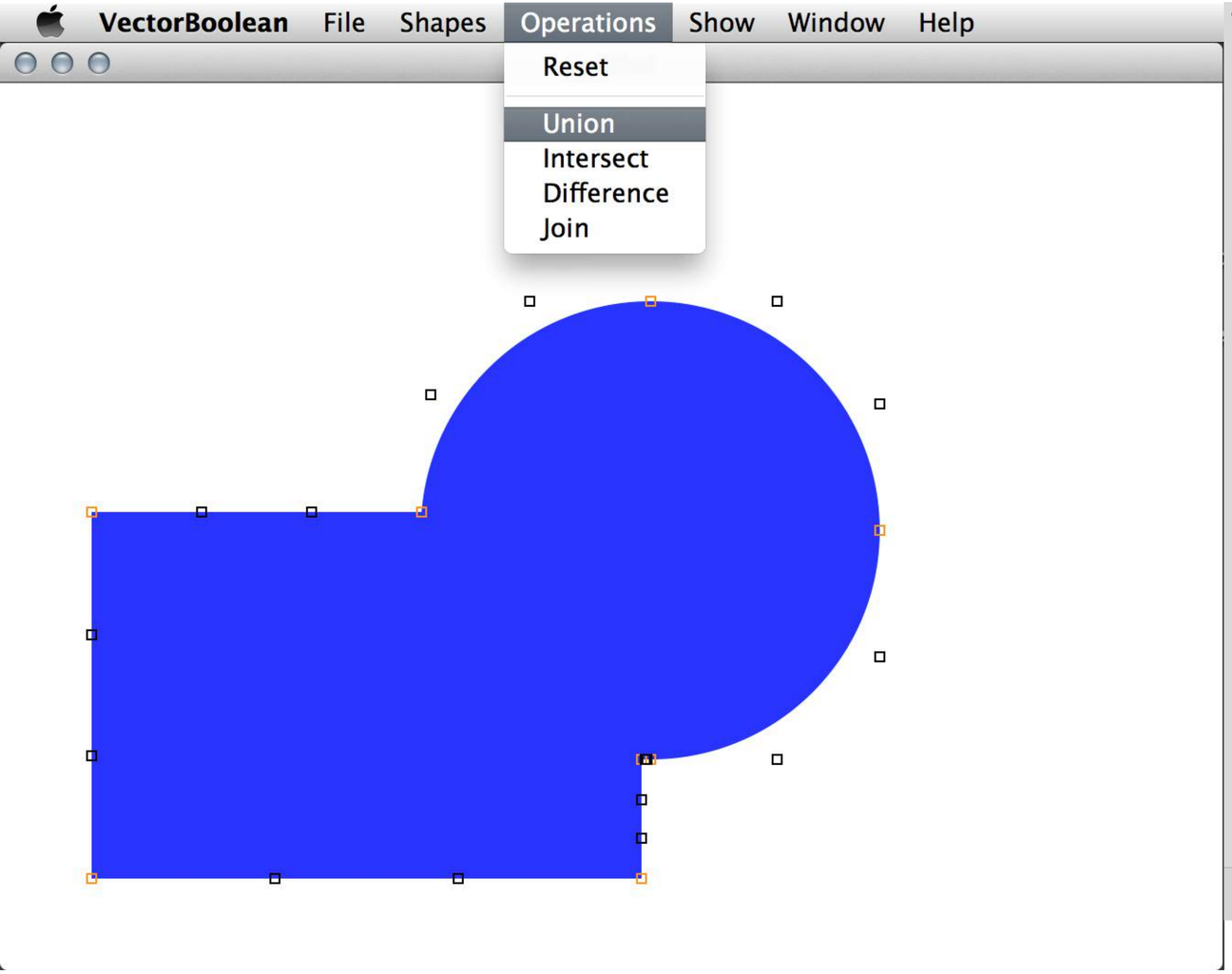
```
CTTypesetterRef typesetter =
    CTTypesetterCreateWithAttributedString(attributedString);

CTLineRef line  = CTTypesetterCreateLine(typesetter, range);
CFArrayRef runs = CTLineGetGlyphRuns(line);
CTRunRef run    = CFArrayGetValueAtIndex(runs, 0);

CGGlyph buffer[range.length];
CTRunGetGlyphs(run, range, buffer);
CGGlyph glyph    = buffer[0];
CGPathRef CGPath = CTFontCreatePathForGlyph(font, glyph, NULL);

CFRelease(typesetter)
. . .
```

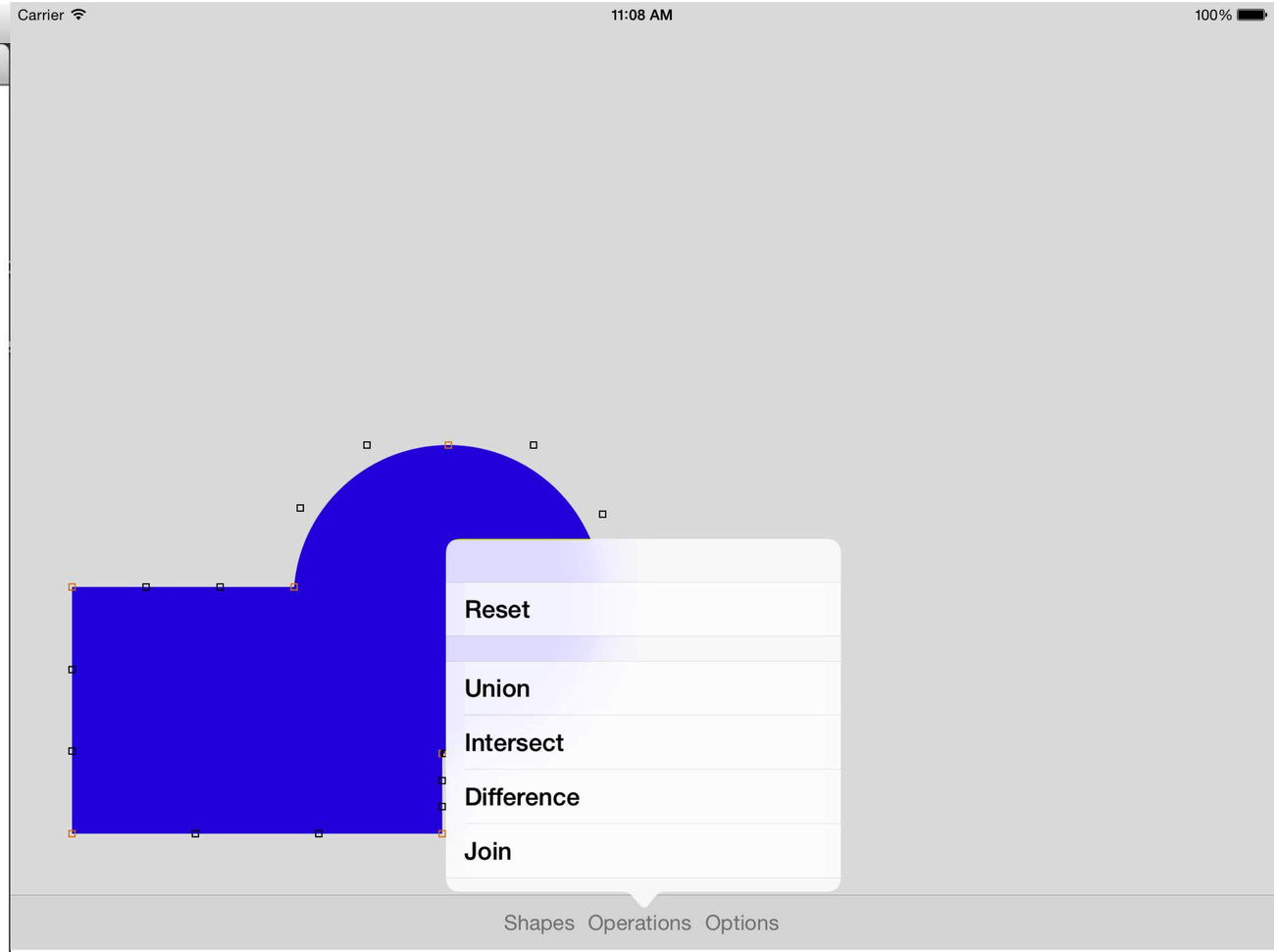
Demo (Kaleidotype)

Boolesche Operationen

- keine APIs in den Frameworks
- http://bitbucket.org/andyfinnell/vectorboolean/
  - eigener Fork mit Unterstützung für CGPath: http://bitbucket.org/martinwinter/vectorbooleancg/
  - Original inzwischen verbessert

Boolesche Operationen





Demo

Pfadelemente untersuchen

- OS X: `-[NSBezierPath elementAtIndex:associatedPoints:]`
- plattformübergreifend: `CGPathApplierFunction`
- erlaubt keine Modifikation
  - ggf. neuen Pfad erzeugen
- Beispiel für Serialisierung von CGPath: https://gist.github.com/martinwinter/9626774

```
void MWCGPathApplierFunction(void *info, const CGPathElement *element)
{
    // Info can be anything, in this case an array to collect information
    // about the path elements.
    NSMutableArray *array = (NSMutableArray *)info;

    CGPathElementType type = element->type;
    CGPoint *points = element->points;

    switch (type)
    {
        case kCGPathElementMoveToPoint:
            // Do whatever is appropriate for Move elements.
            break;

        case kCGPathElementAddLineToPoint:
            // Do whatever is appropriate for AddLineTo elements.
            break;
. . .
```

```
// Call applier function to extract information into the array.
NSMutableArray *array = [NSMutableArray array];
CGPathApply(path, array, MWCGPathApplierFunction);

// Use previously extracted information to recreate path.
CGMutablePathRef path = CGPathCreateMutable();
for (NSUInteger index = 0; index < [array count]; index++)
{
   NSDictionary *dictionary = array[index];
   CGPathElementType type = [(NSNumber *)dictionary[@"type"] intValue];
   CGPoint *points = (CGPoint *)[(NSData *)dictionary[@"points"] bytes];

   switch (type)
   {
       case kCGPathElementMoveToPoint:
           CGPathMoveToPoint(path, NULL, points[0].x, points[0].y);
           break;
. . .
```

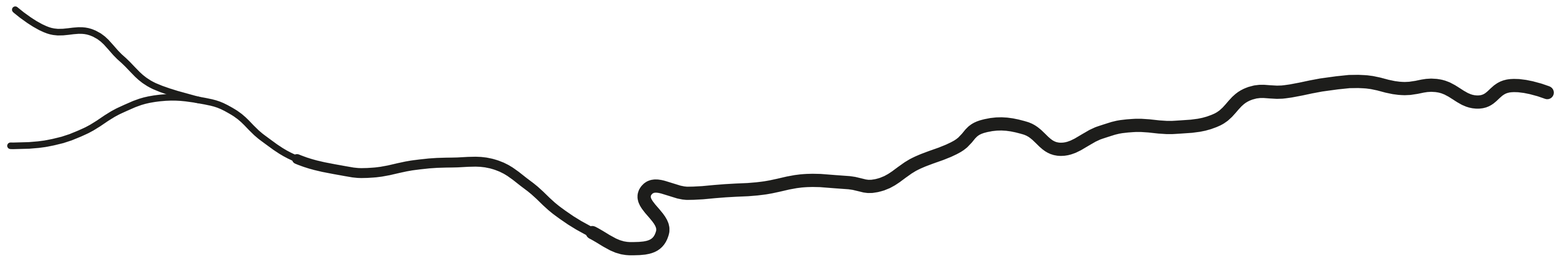
Kreisbögen

- vielfältige APIs in allen drei Frameworks: `...Arc...`
- mathematischer Hintergrund für manuelles Zeichnen
- Abstand der Kontrollpunkte von der Mittelachse:
  - `kappa = 0.5522847498 = 4 / 3 * (sqrt(2) - 1)`
- http://whizkidtech.redprince.net/bezier/circle/

Praxisbeispiel

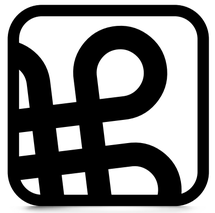
Praxisbeispiel

- iPad-App für strategische Unternehmensberatung
- Statistik-Diagramme
- Visualisierung organisationeller Metriken als Insel
  - u. a. dynamisch erzeugte Flüsse mit realistischer Physik



Fragen?

Vielen Dank!



Macoun