Bézier

2013.3.30
Cocoa勉強会 関西

大森智史

@oogon / satoshi.oomori

- スライド、サンプルは基本的にすべて公開します。

- 後ほどFacebookページにて

あんた、誰？

- と、いうわけで自己紹介。

- 大森智史といいます。
- Objective-Cで遊んでます。
- Cocoa勉強会関西は最初からいます。

- 2/23に本が出ました。

- まさかの4000円台

- と、いうことで本題に

今日のお話

Bézier

Bézier

- こんなやつですな

ところで

- iOS 6 になって、ものすごく気になるUIパーツができました。

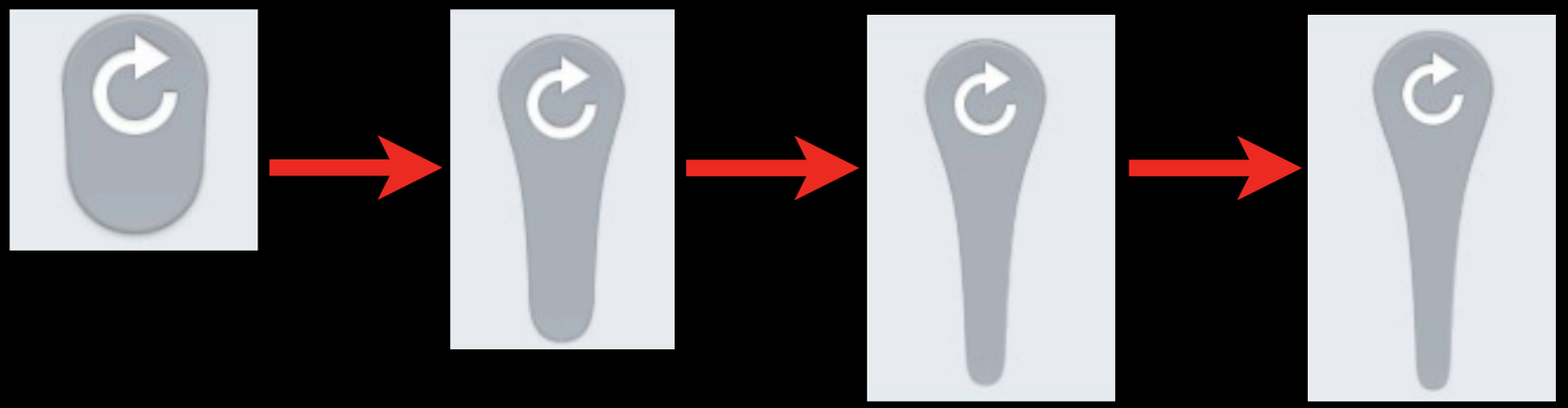
それは

- これです。

- びよーん。

- どうやってるのかなあ。

- じっくりと見てみることにします。



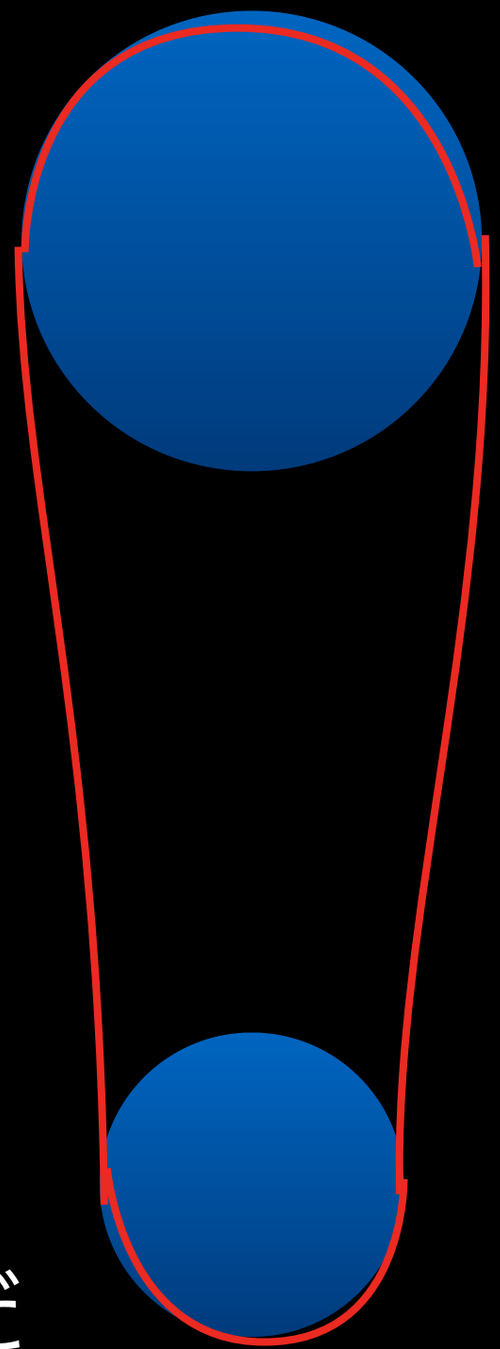
- 最初は楕円

- 下が段々小さくなっていく、上の円と下の円の接続は曲線

- 下がさらに小さく、上も少し小さく

- さらに続く

- つまり
- 上半円、下半円、それをつなぐ曲線
- 塗り
- これでいい訳だ



- 円、曲線を書くには...

Bézier

- OS Xには、NSBezierPath、

- iOSには、UIBezierPathがある！

- 実際にやってみましょう。
- //下の半円

```
UIBezierPath *currentPath = [UIBezierPath
bezierPath];
```

```
[currentPath addArcWithCenter:
    CGPointMake(200,currentPosition.y+200)
        radius:100-(currentPosition.y/5)
    startAngle:radians(0)
      endAngle:radians(180)
     clockwise:YES];
```

- 度→ラジアンはマクロを作っておくと便利です。

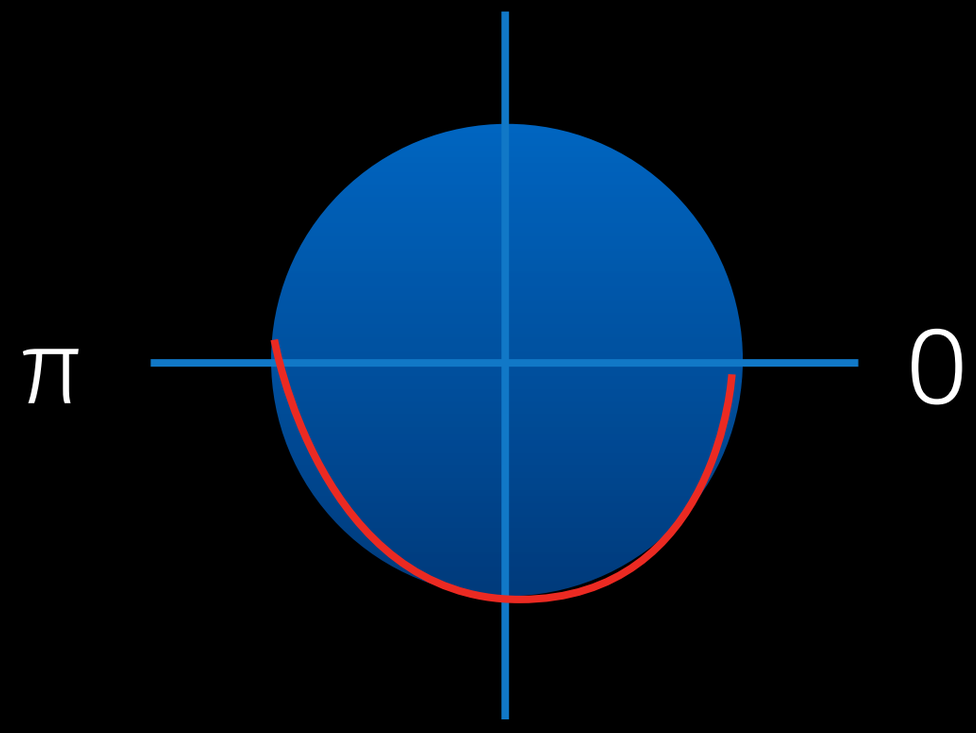
```
//PIを3.14...と定義
#define PI 3.14159265358979323846

//関数定義degrees*PI/180がradians
static inline double radians(double
degrees) { return degrees * PI / 180; }
```

- ベジェパスを作る

```
[UIBezierPath bezierPath];
```

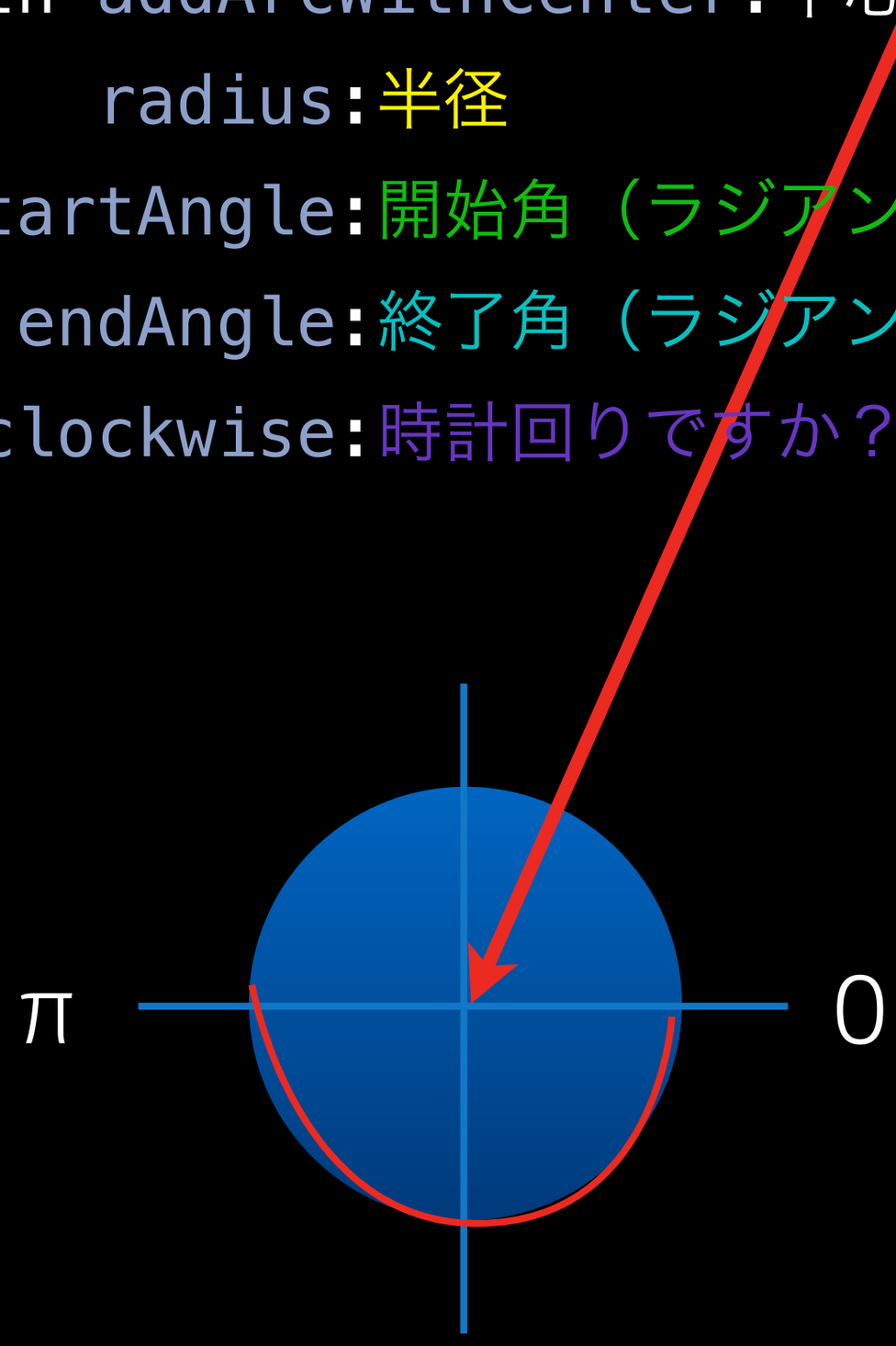
- 弧を描く

```
[currentPath addArcWithCenter:中心点
                       radius:半径
                   startAngle:開始角（ラジアン）
                     endAngle:終了角（ラジアン）
                    clockwise:時計回りですか？
];
```



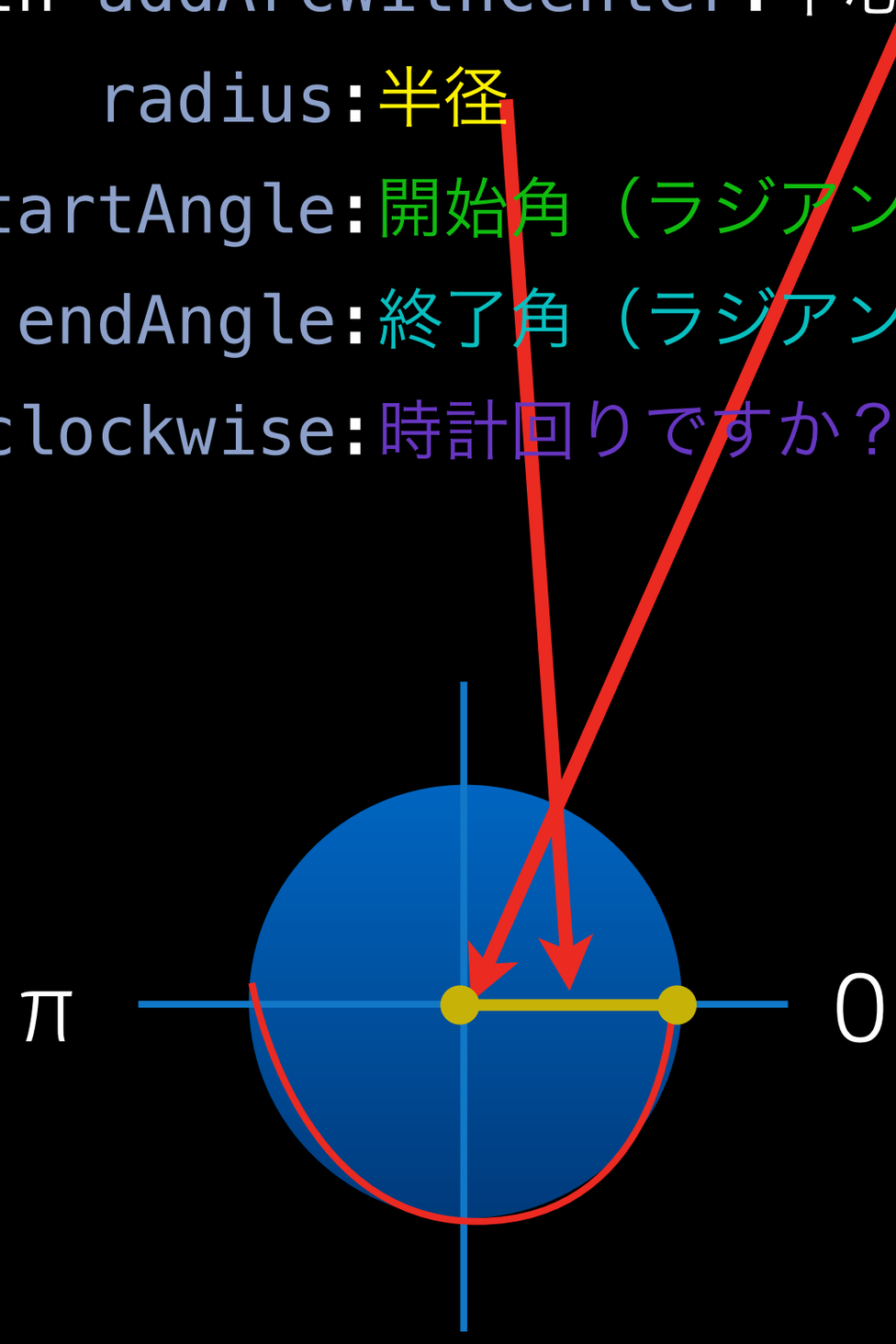
- 弧を描く

```
[currentPath addArcWithCenter:中心点
                       radius:半径
                   startAngle:開始角（ラジアン）
                     endAngle:終了角（ラジアン）
                    clockwise:時計回りですか？
];
```



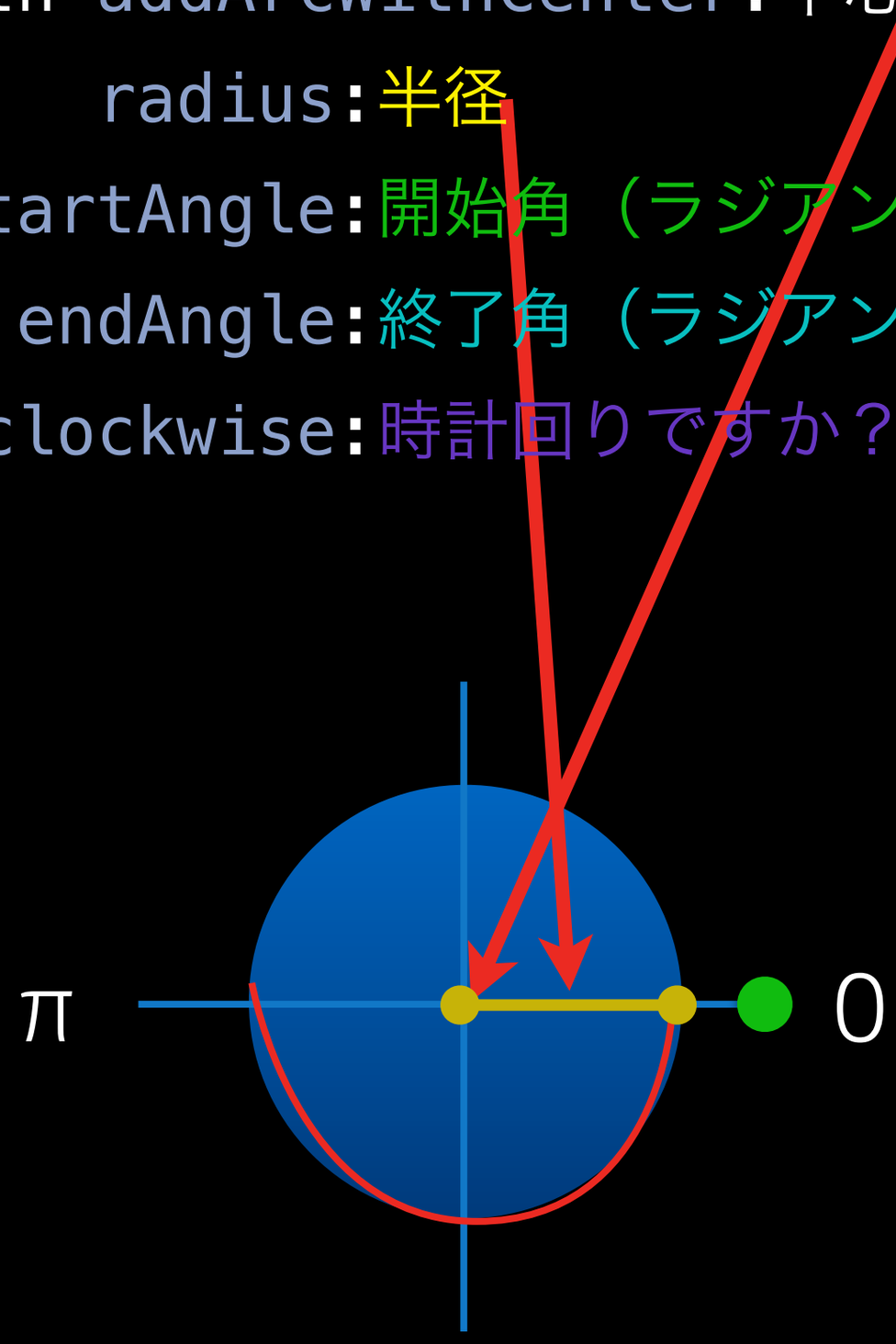
- 弧を描く

```
[currentPath addArcWithCenter:中心点
                       radius:半径
                   startAngle:開始角（ラジアン）
                     endAngle:終了角（ラジアン）
                    clockwise:時計回りですか？
];
```



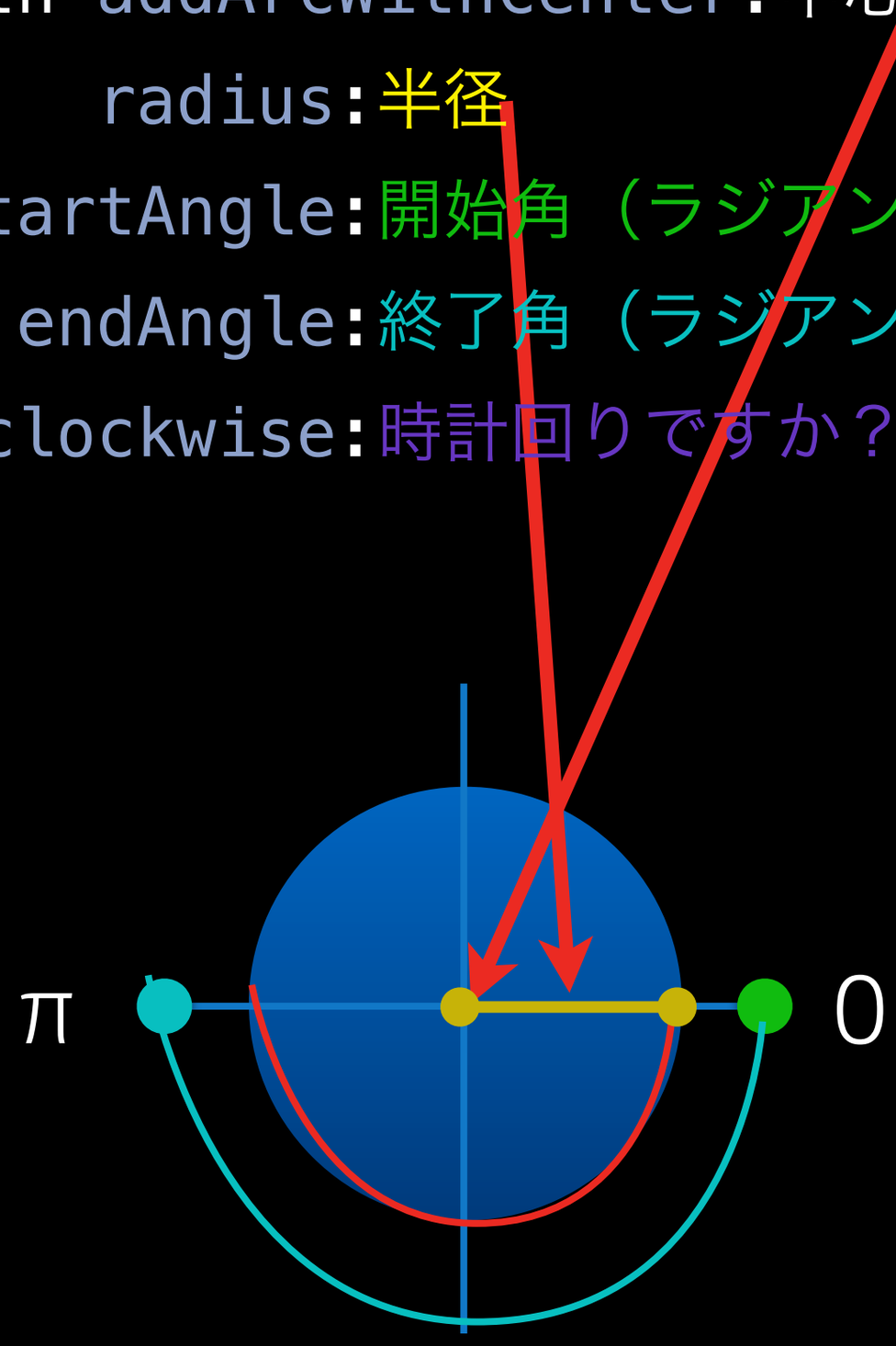
- 弧を描く

```
[currentPath addArcWithCenter:中心点
                       radius:半径
                   startAngle:開始角（ラジアン）
                     endAngle:終了角（ラジアン）
                    clockwise:時計回りですか？
];
```



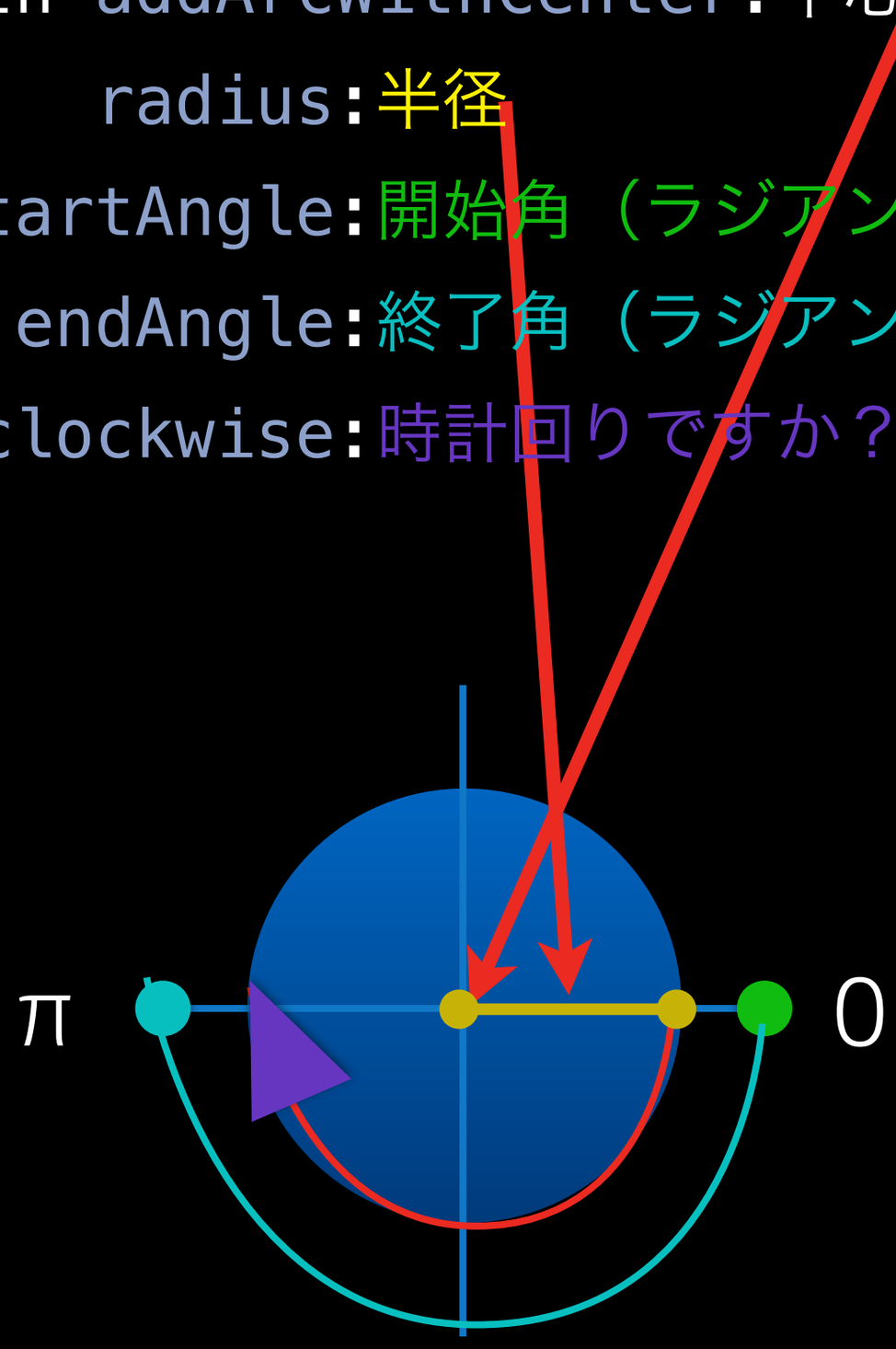
- 弧を描く

```
[currentPath addArcWithCenter:中心点
                       radius:半径
                   startAngle:開始角（ラジアン）
                     endAngle:終了角（ラジアン）
                    clockwise:時計回りですか？
];
```

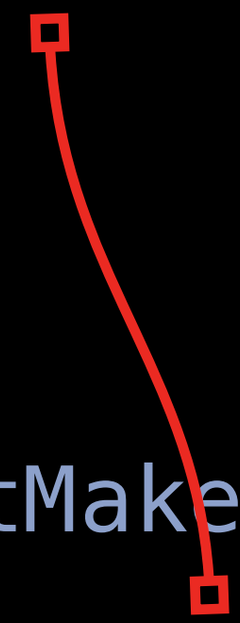


- 弧を描く

```
[currentPath addArcWithCenter:中心点
                       radius:半径
                   startAngle:開始角（ラジアン）
                     endAngle:終了角（ラジアン）
                    clockwise:時計回りですか？
];
```



- 次に曲線



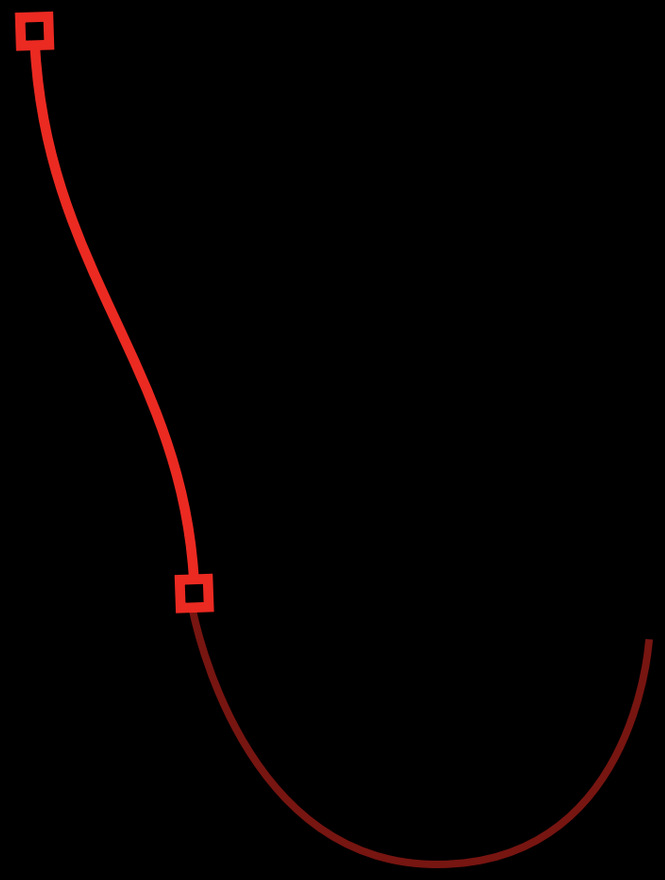
```
//左側のつなぎ
[currentPath
addCurveToPoint:CGPointMake(100,200)

controlPoint1:CGPointMake(100+
(currentPosition.y/
5),currentPosition.y+200-
(currentPosition.y/3))

controlPoint2:CGPointMake(100,200+
(currentPosition.y/5))];
```

- 曲線を描く

```
[currentPath addCurveToPoint:終了点
  controlPoint1:コントロールポイント1
  controlPoint2:コントロールポイント2
];
```

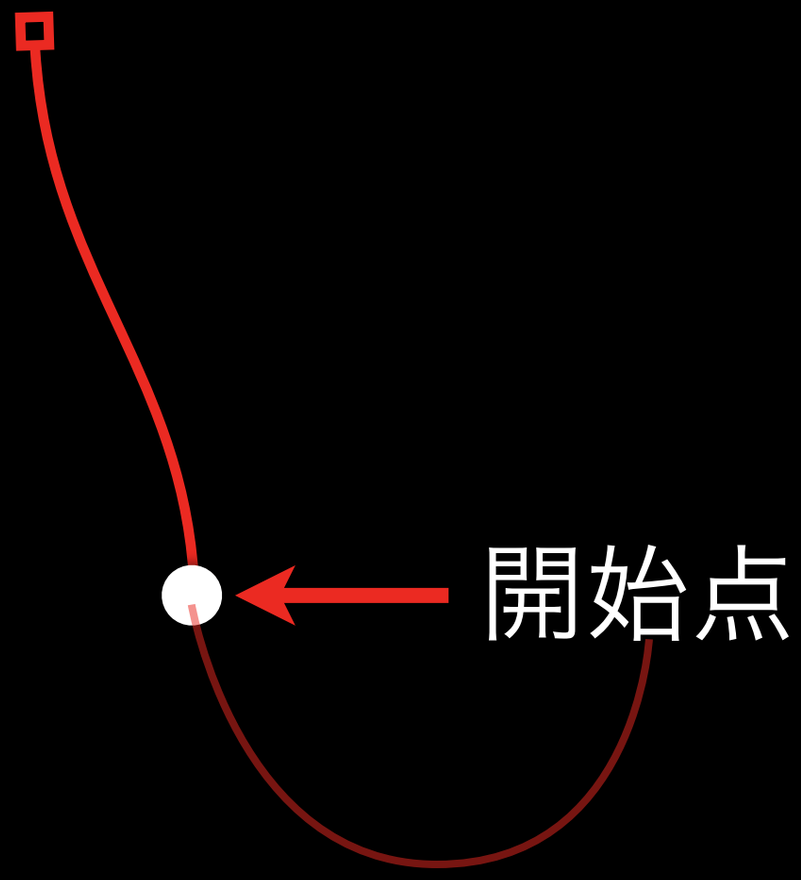
- 曲線を描く

```
[currentPath addCurveToPoint:終了点
  controlPoint1:コントロールポイント1
  controlPoint2:コントロールポイント2
];
```



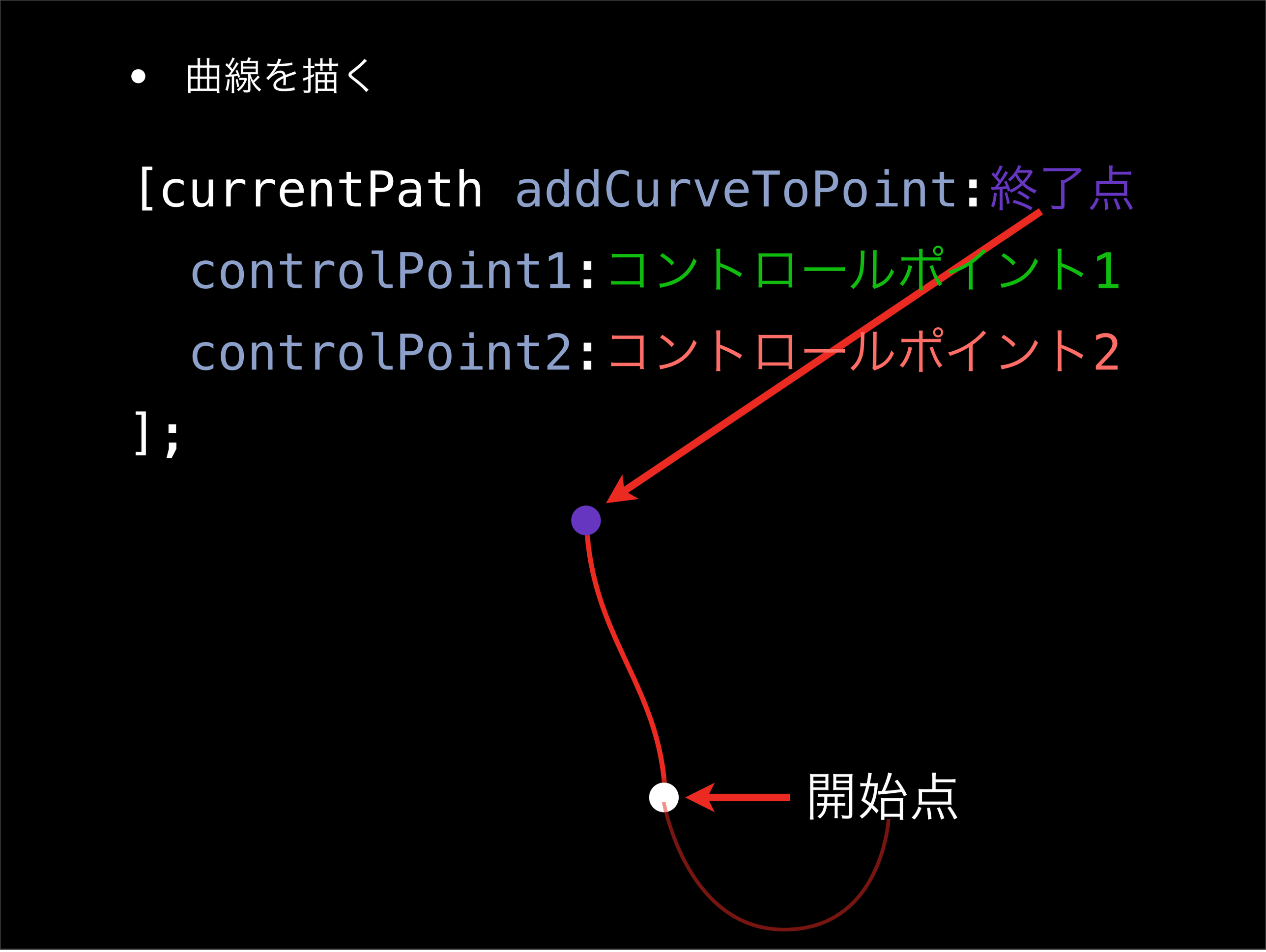
- 曲線を描く

```
[currentPath addCurveToPoint:終了点
  controlPoint1:コントロールポイント1
  controlPoint2:コントロールポイント2
];
```



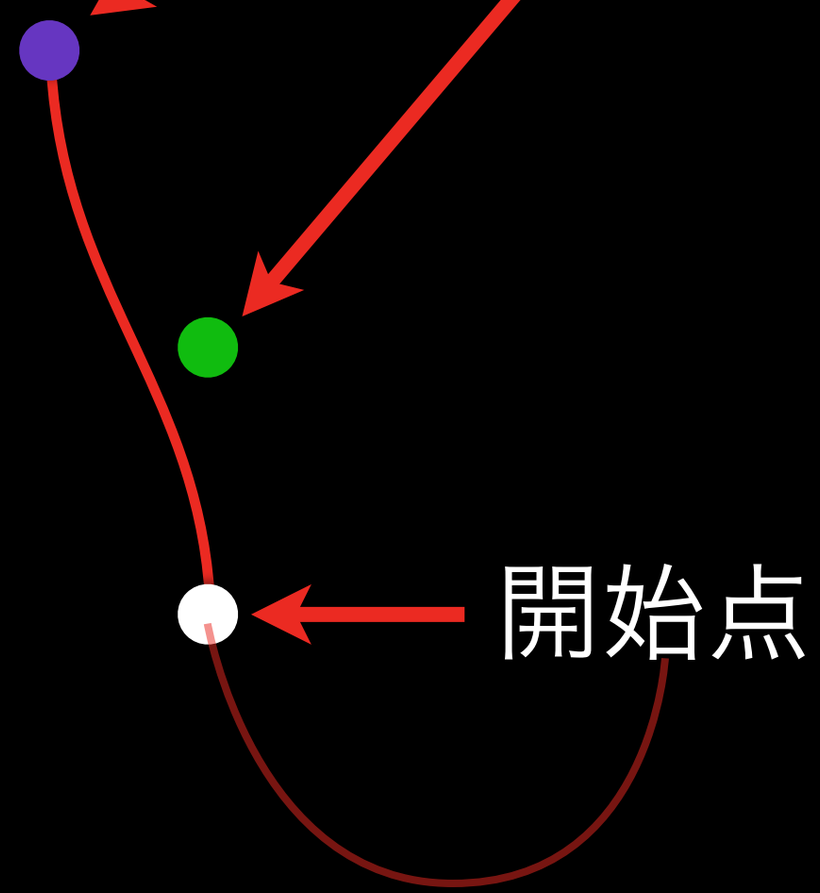
- 曲線を描く

```
[currentPath addCurveToPoint:終了点
  controlPoint1:コントロールポイント1
  controlPoint2:コントロールポイント2
];
```



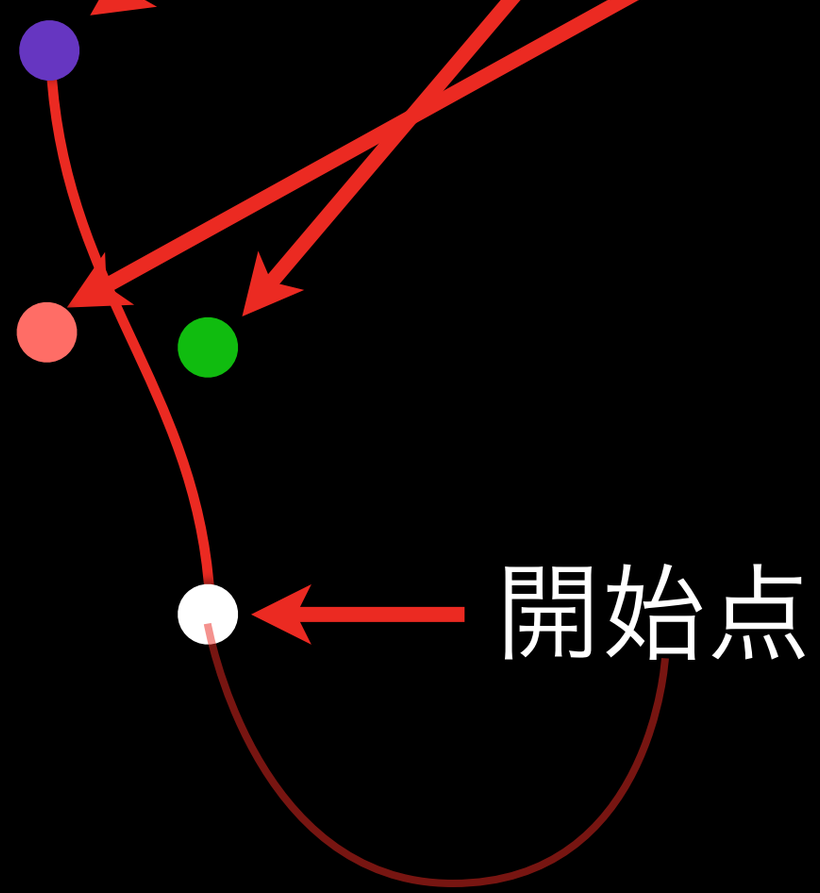
- 曲線を描く

```
[currentPath addCurveToPoint:終了点
  controlPoint1:コントロールポイント1
  controlPoint2:コントロールポイント2
];
```



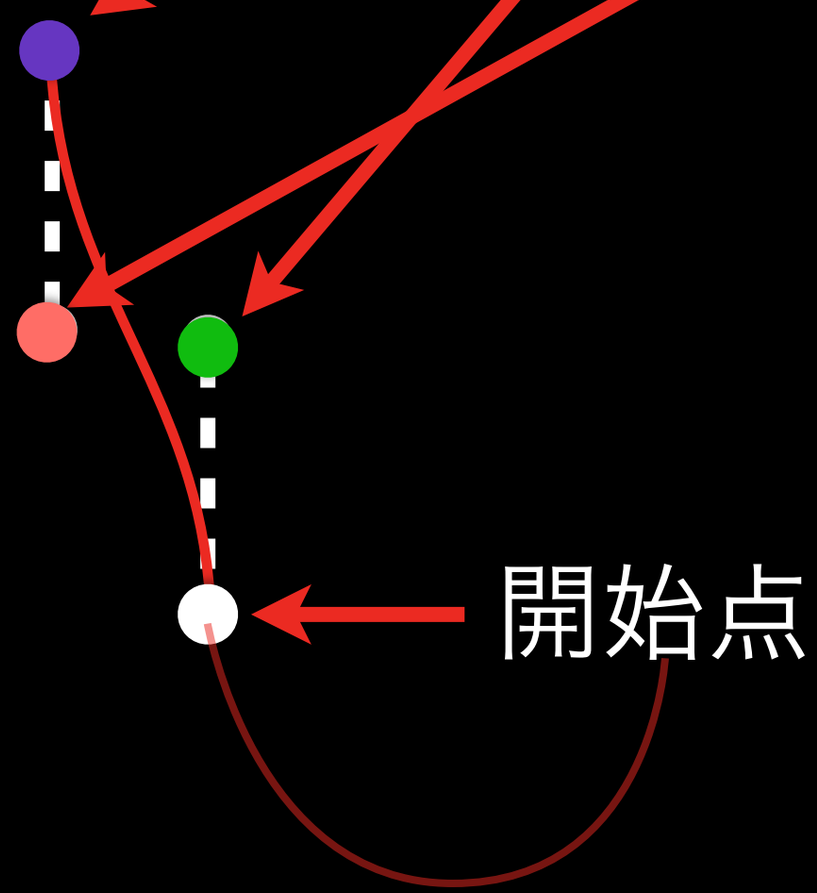
- 曲線を描く

```
[currentPath addCurveToPoint:終了点
  controlPoint1:コントロールポイント1
  controlPoint2:コントロールポイント2
];
```

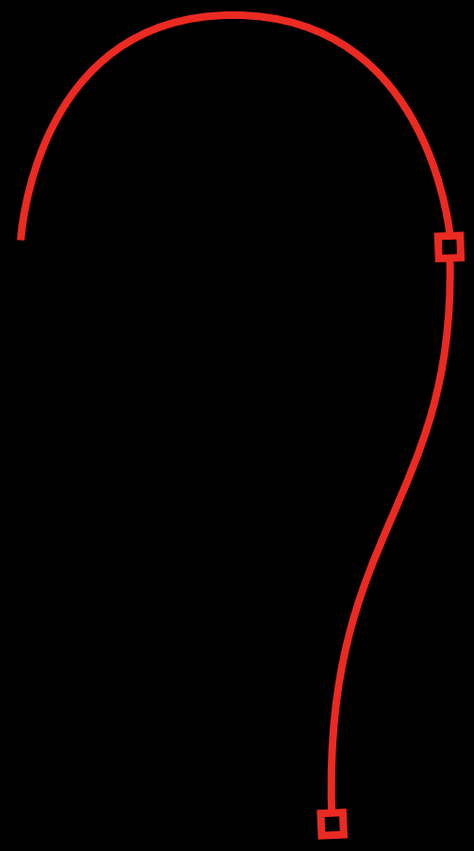


- 曲線を描く

```
[currentPath addCurveToPoint:終了点
  controlPoint1:コントロールポイント1
  controlPoint2:コントロールポイント2
];
```



- 同様に上半弧、右側曲線を描きます。



- パスを閉じて

```
[currentPath closePath];
```

- カラーをセットして

```
[[UIColor lightGrayColor] setFill];
```

- 塗ります。

```
[currentPath fill];
```

- これで、上下の移動距離に応じて弧の大きさを変えて、曲線の長さを変えて描画します。

- DEMO

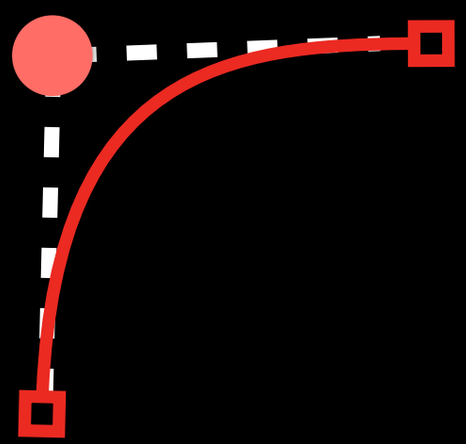
- Bezier

- これだけではなんなので...

Bezierについての
まめ知識

知っとくと便利

- Bézierについてのお話



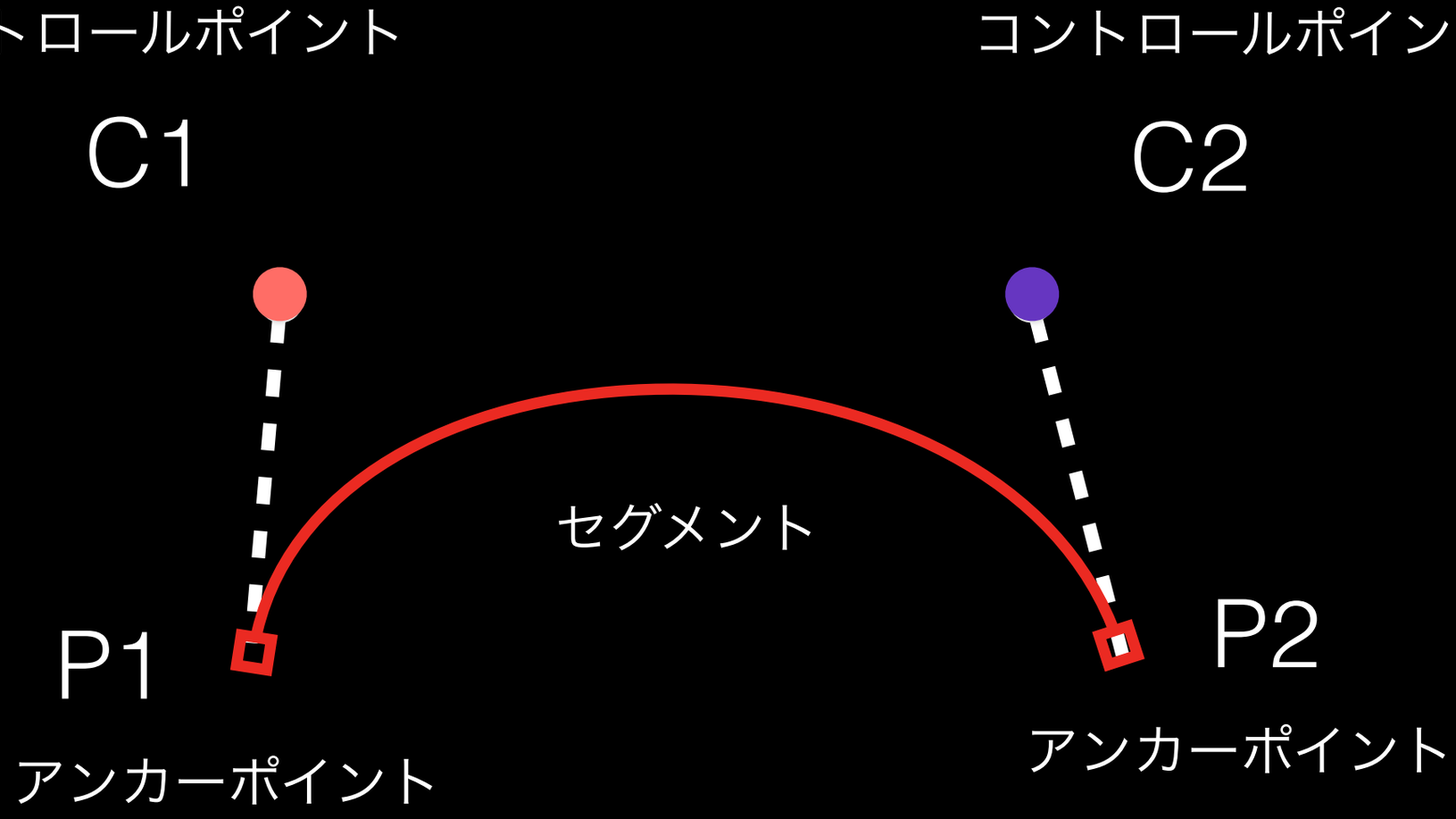
2次ベジェ

角丸などに使われます。

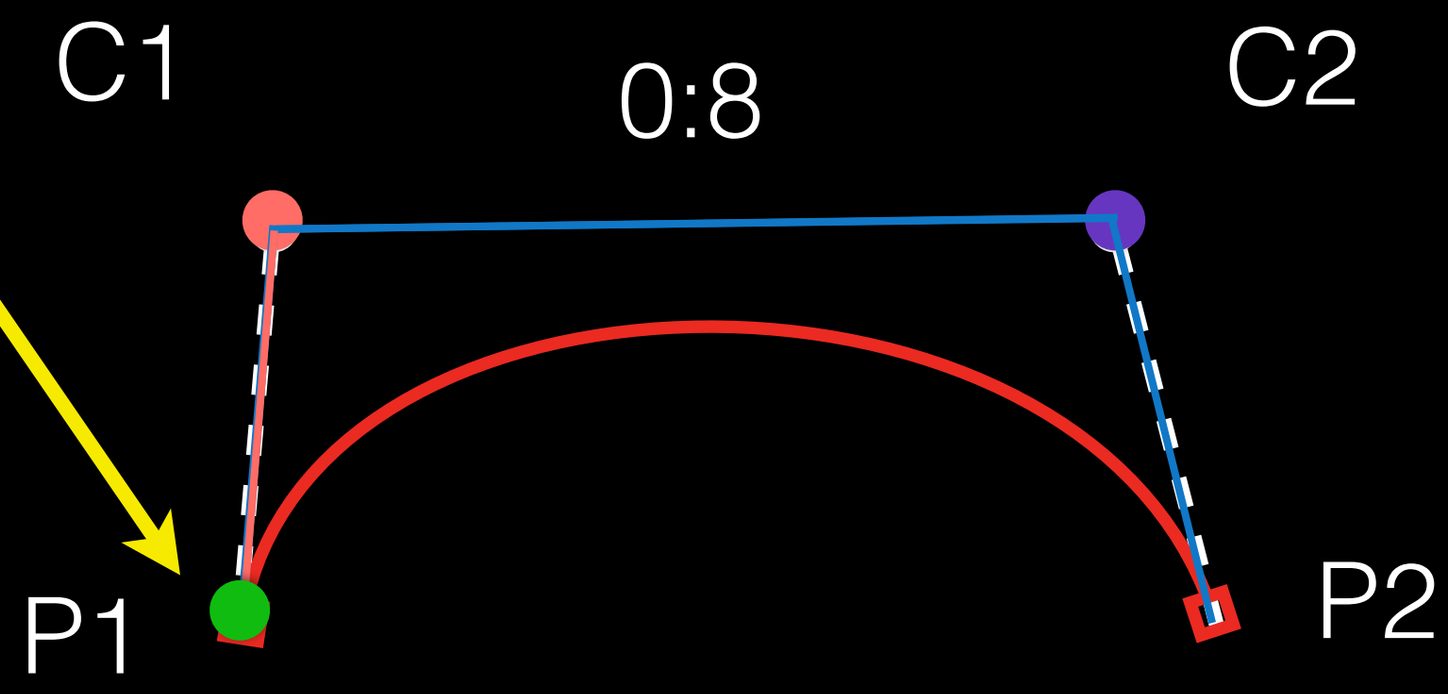
3次ベジェ

曲線に使われます。

- Bézierについてのお話

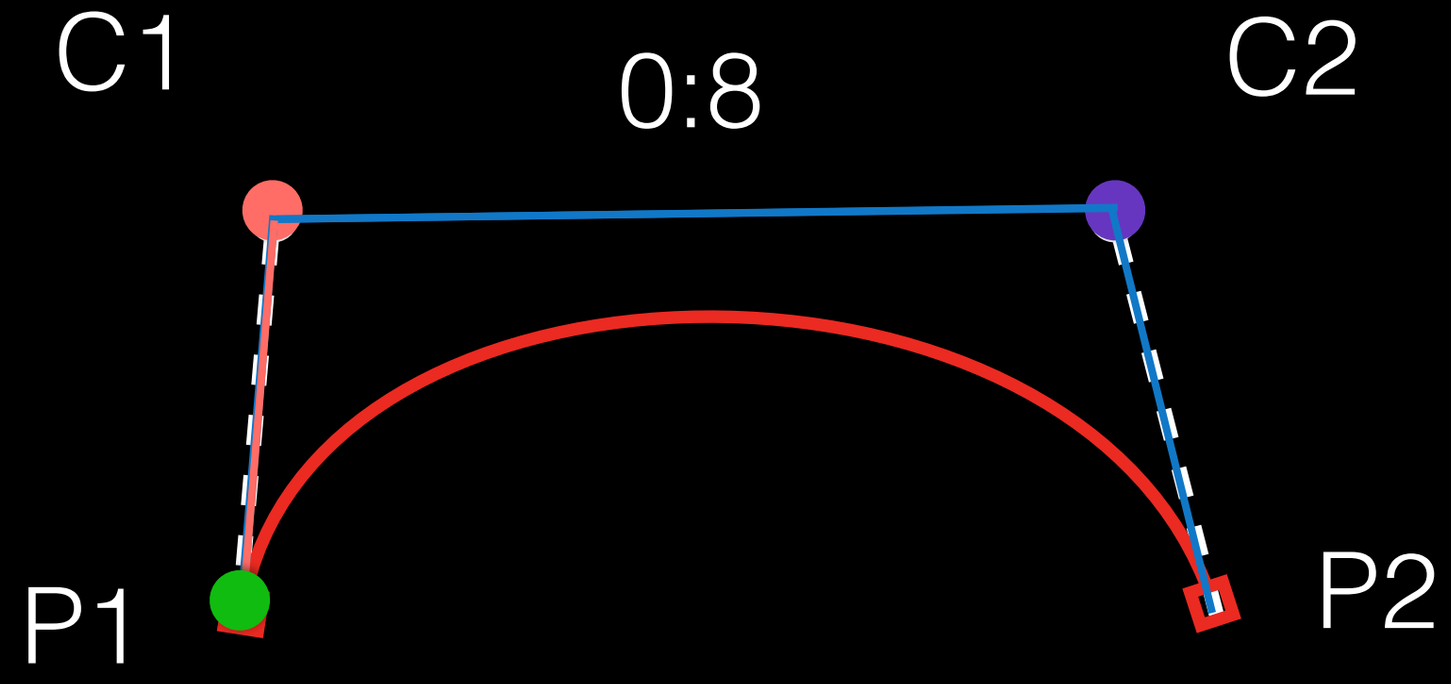


- Bézierについてのお話



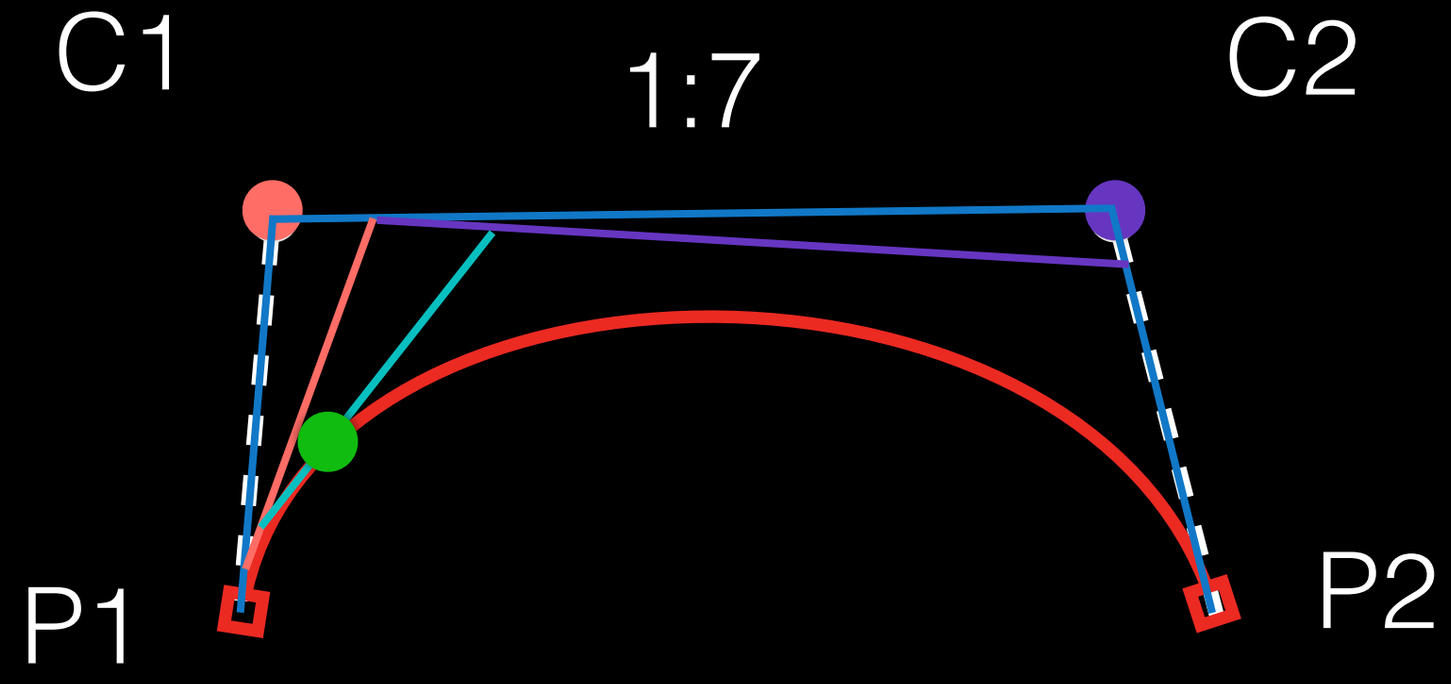
参考：「中学生でもわかるベジェ曲線」
http://ruiueyama.tumblr.com/post/11197882224

- Bézierについてのお話



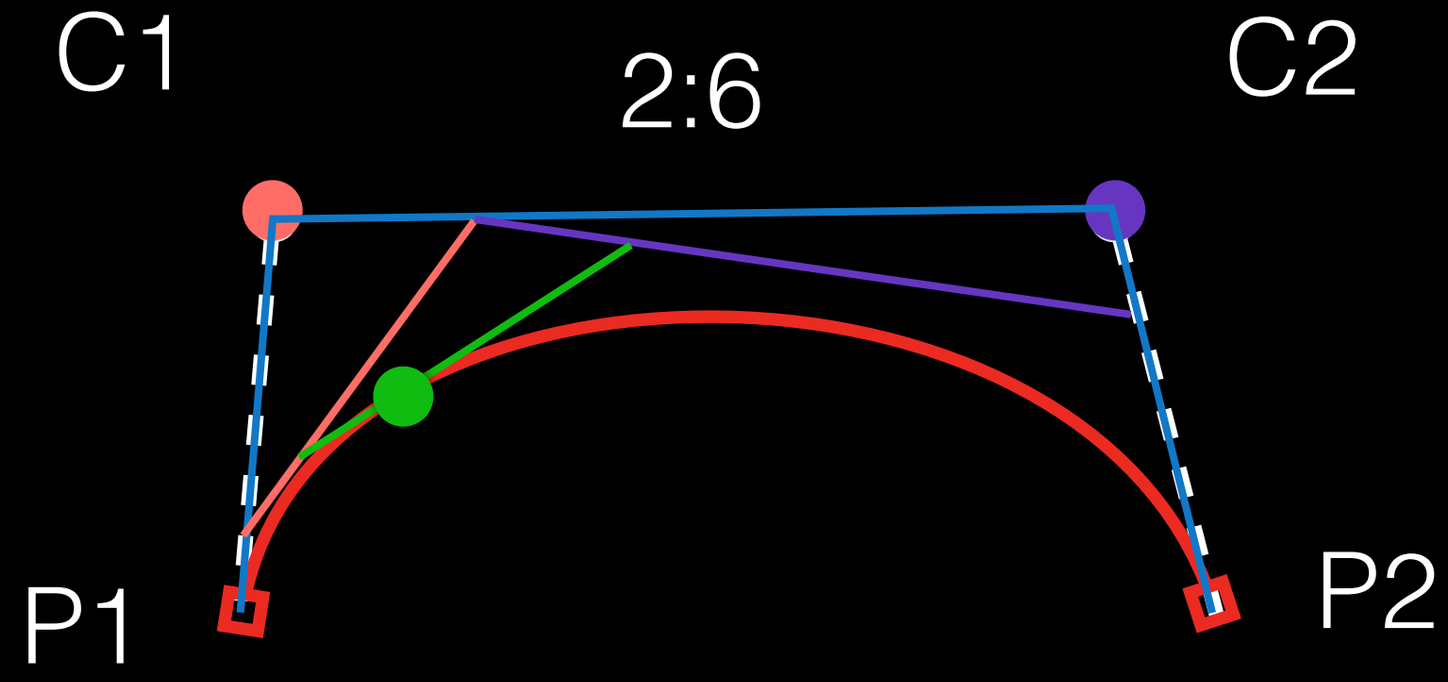
参考：「中学生でもわかるベジェ曲線」
http://ruiueyama.tumblr.com/post/11197882224

- Bézierについてのお話



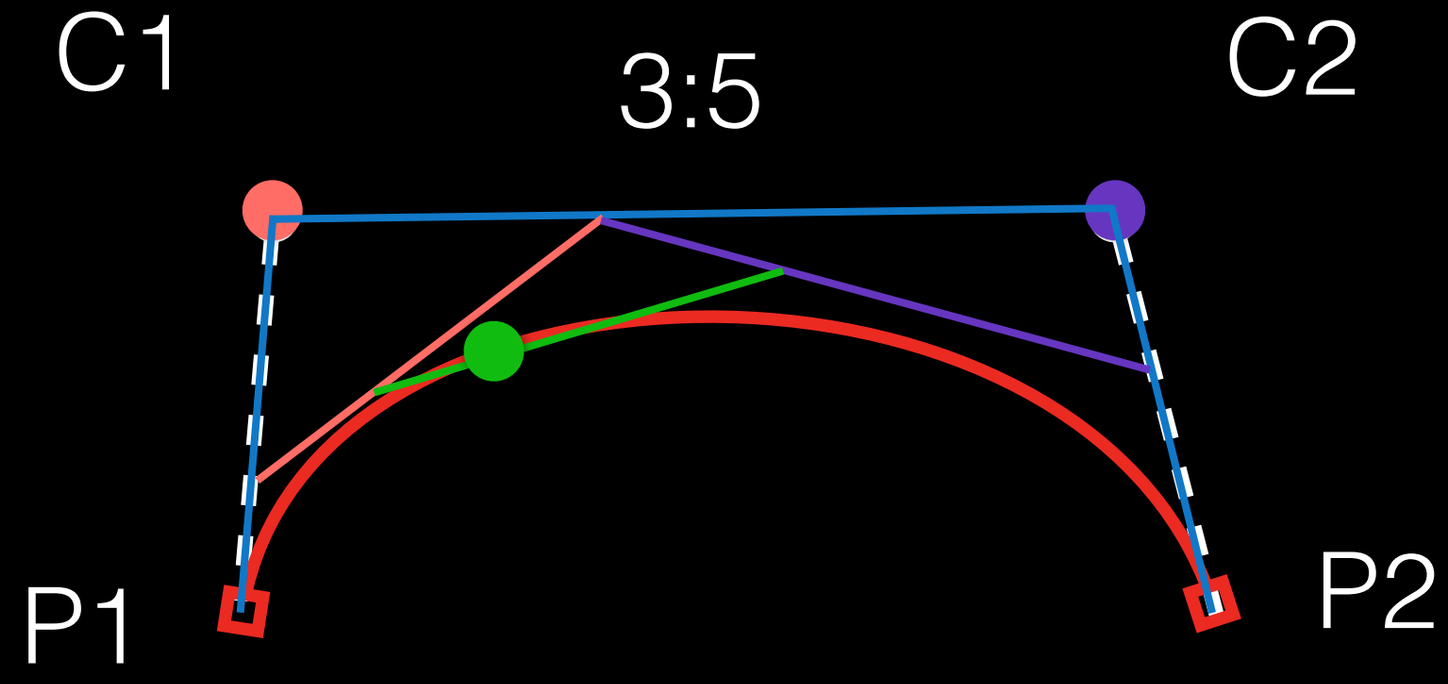
参考：「中学生でもわかるベジェ曲線」
http://ruiueyama.tumblr.com/post/11197882224

- Bézierについてのお話



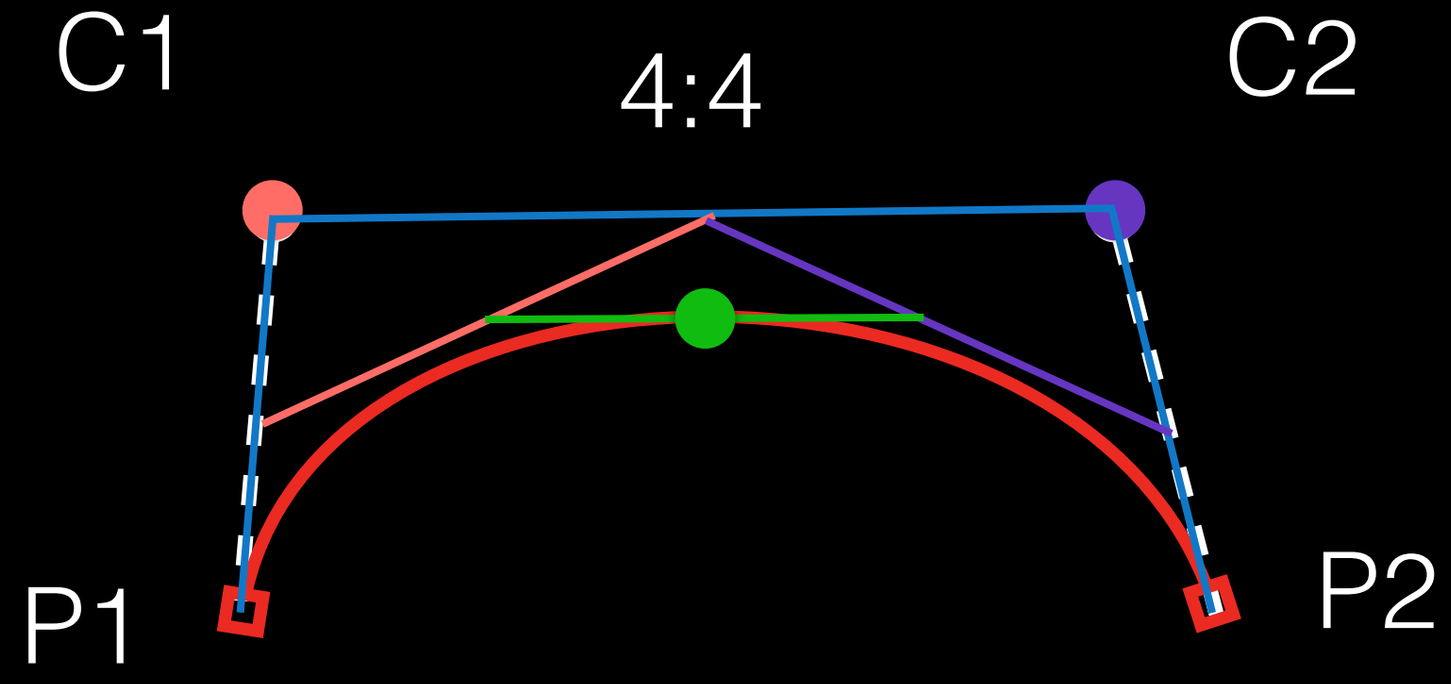
参考：「中学生でもわかるベジェ曲線」
http://ruiueyama.tumblr.com/post/11197882224

- Bézierについてのお話



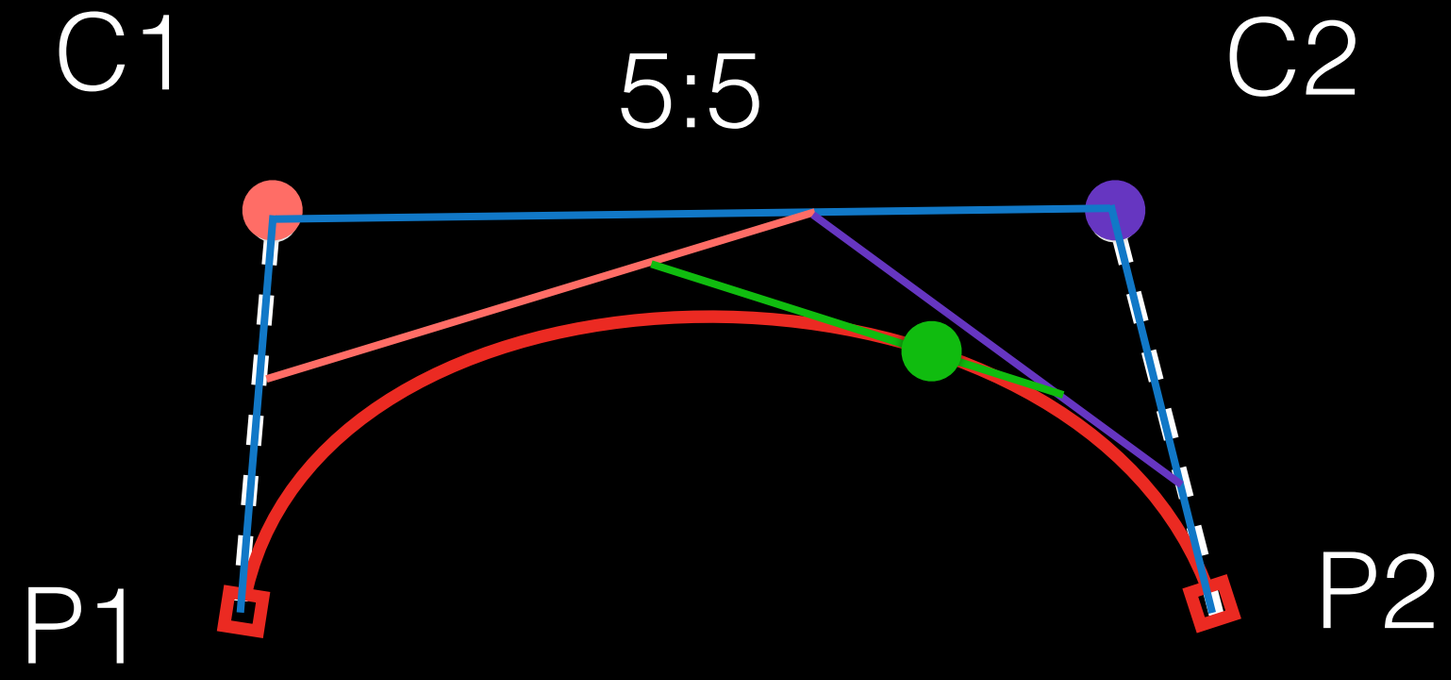
参考：「中学生でもわかるベジェ曲線」
http://ruiueyama.tumblr.com/post/11197882224

- Bézierについてのお話



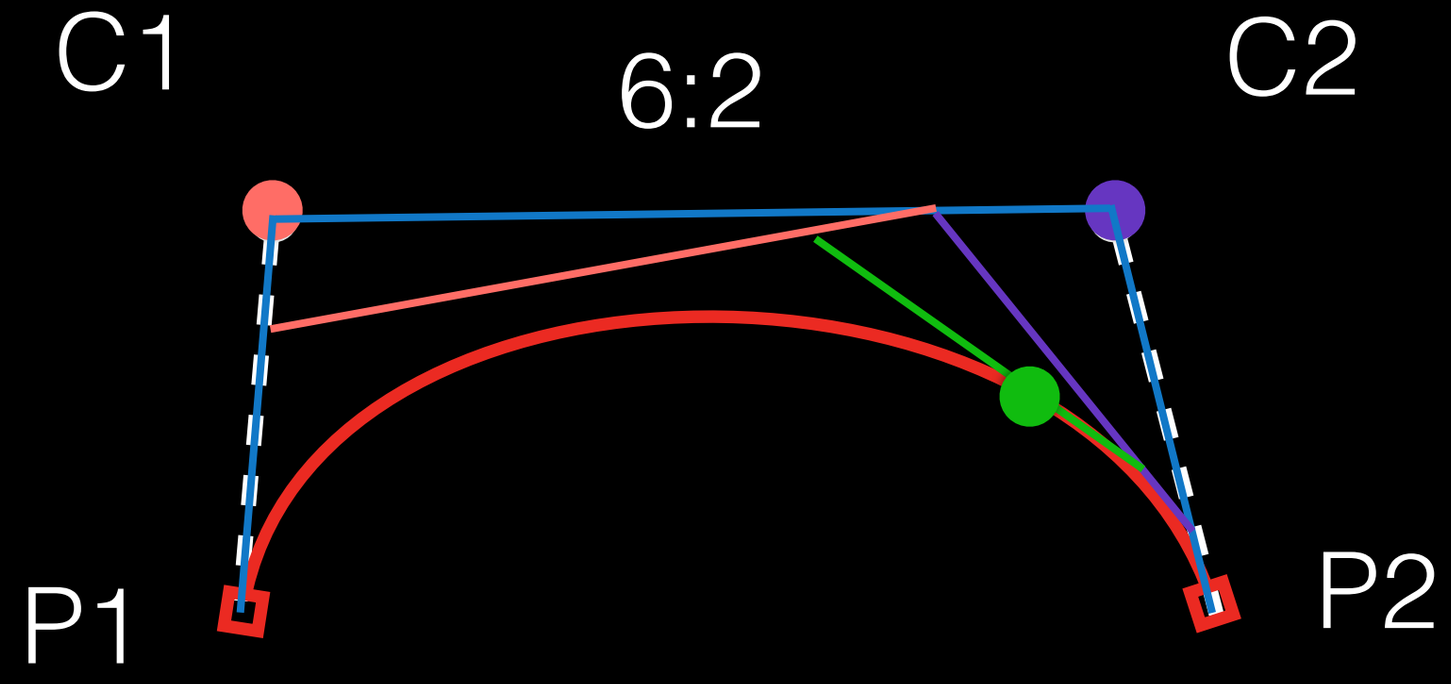
参考：「中学生でもわかるベジェ曲線」
http://ruiueyama.tumblr.com/post/11197882224

- Bézierについてのお話



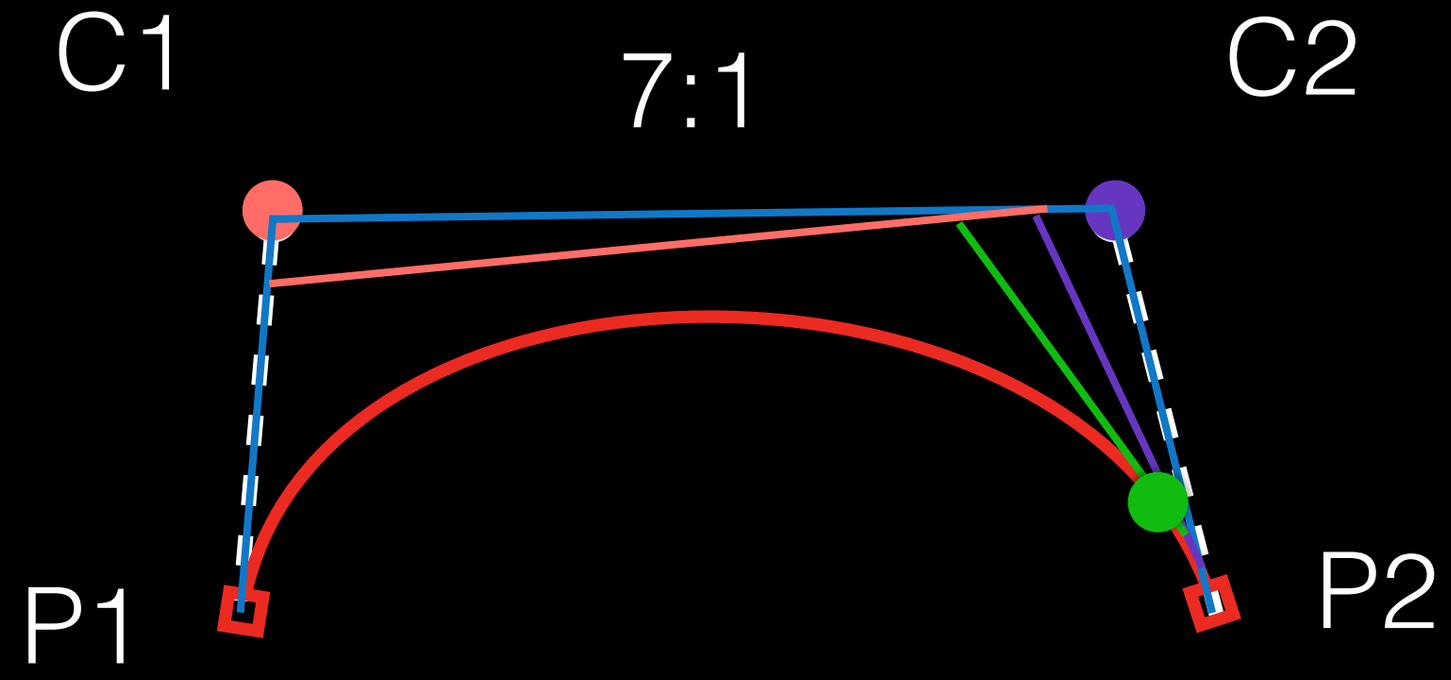
参考：「中学生でもわかるベジェ曲線」
http://ruiueyama.tumblr.com/post/11197882224

- Bézierについてのお話



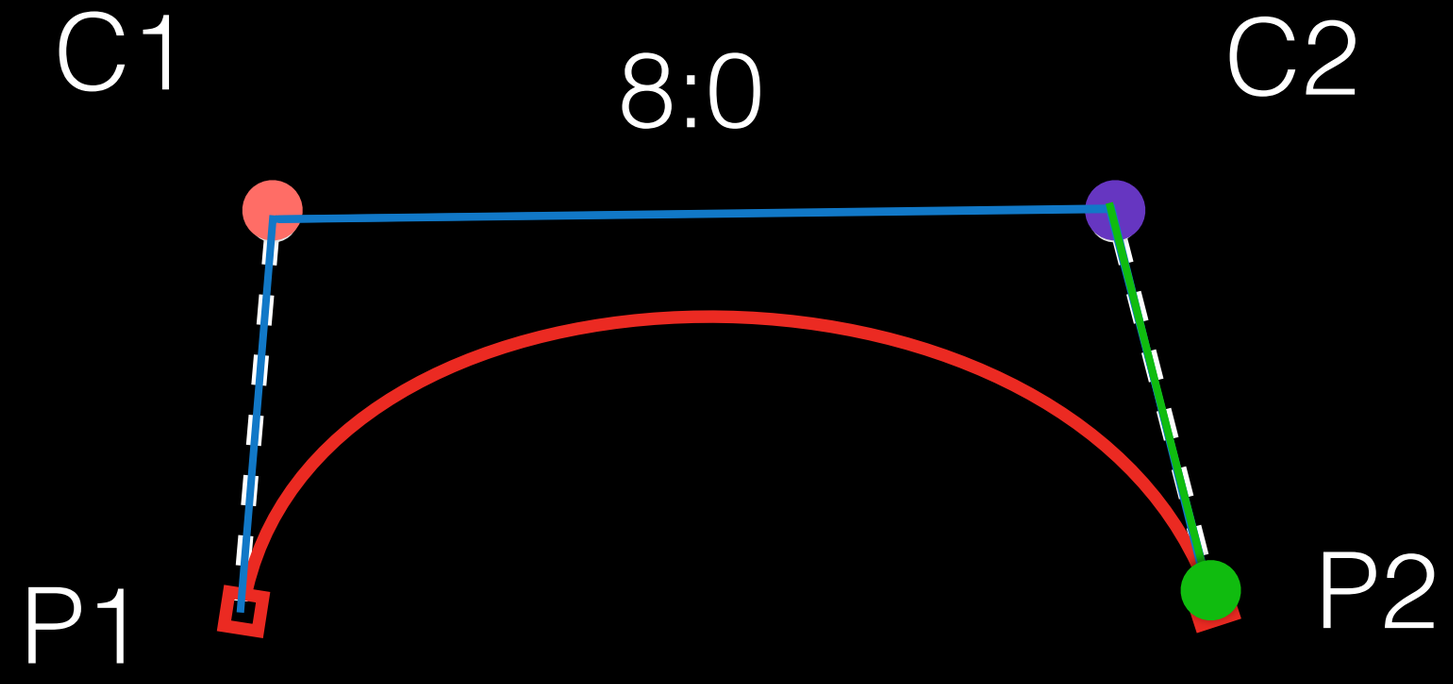
参考：「中学生でもわかるベジェ曲線」
http://ruiueyama.tumblr.com/post/11197882224

- Bézierについてのお話



参考：「中学生でもわかるベジェ曲線」
http://ruiueyama.tumblr.com/post/11197882224

- Bézierについてのお話



参考：「中学生でもわかるベジェ曲線」
http://ruiueyama.tumblr.com/post/11197882224

- 三角形だから中学生でもわかりますよね。

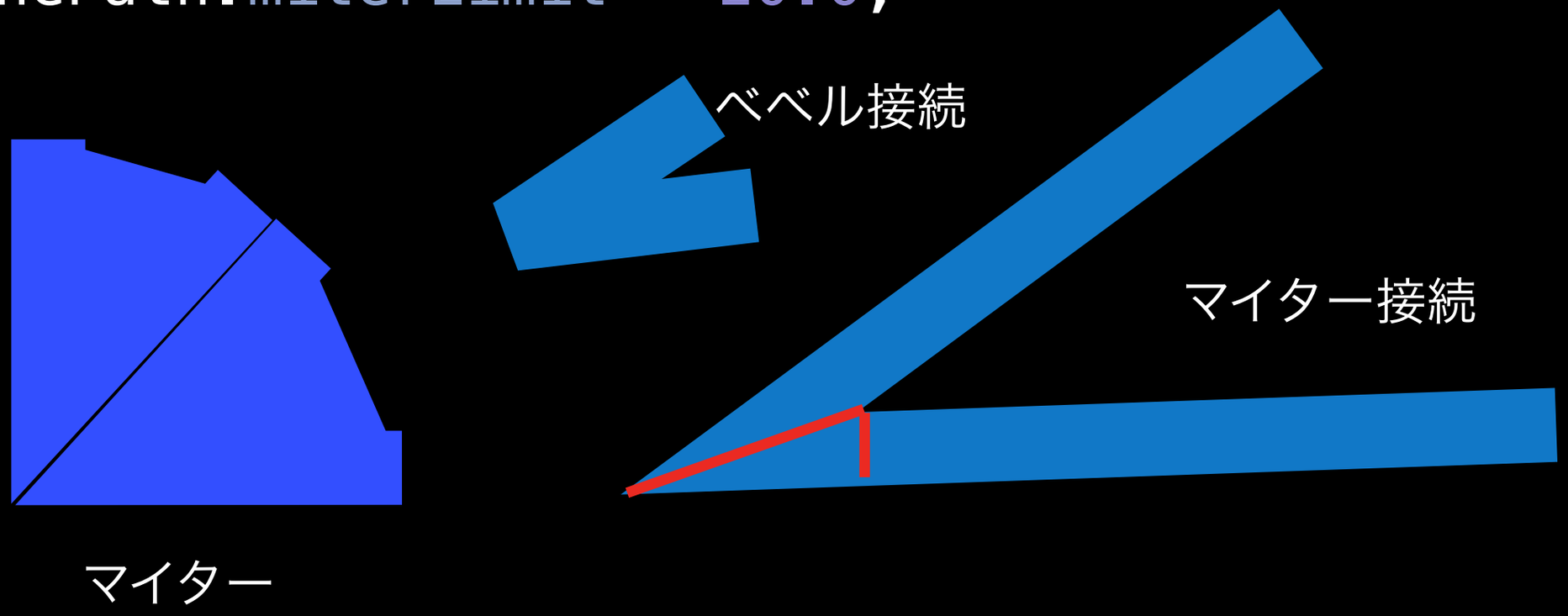
角の処理

マイター、ベベル

マイターリミット（ポイント接続部の角の処理）

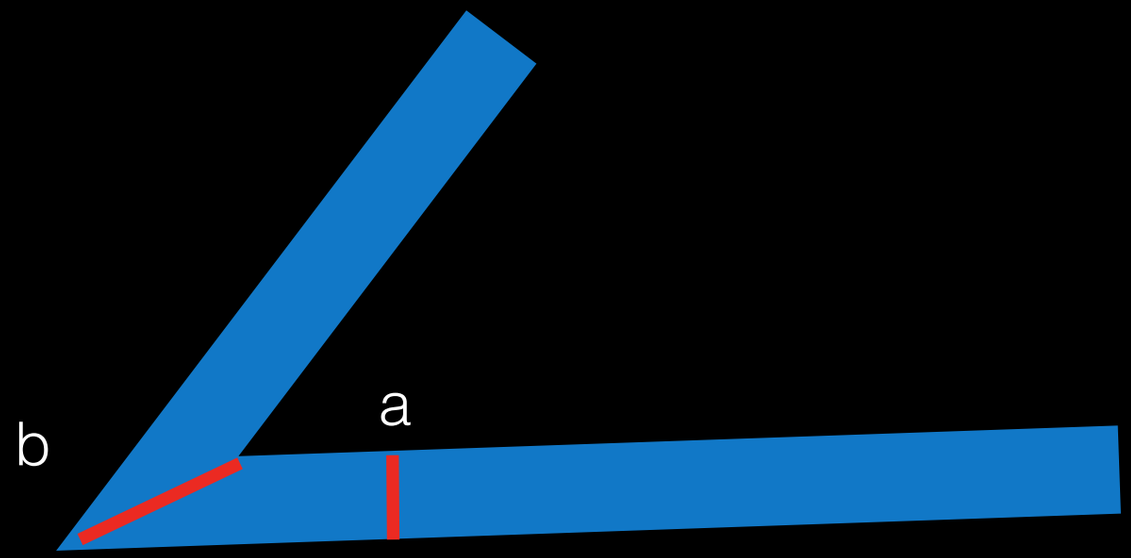
- 線幅との比率（デフォルトは10.0）
- リミット値を超えるとマイター接続になります。
- （それまではベベル接続）

```
thePath.miterLimit = 20.0;
```



- 線幅との比率（デフォルトは10.0）

  60度なら(a:b = 1:2)



- DEMO

- BezierMiter

破線

ラインダッシュ

```
float array[5];

array[0] = 8.0;
array[1] = 3.0;
array[2] = 20.0;
array[3] = 3.0;
[outPath setLineDash: array count: 4
phase: 0.0];
```

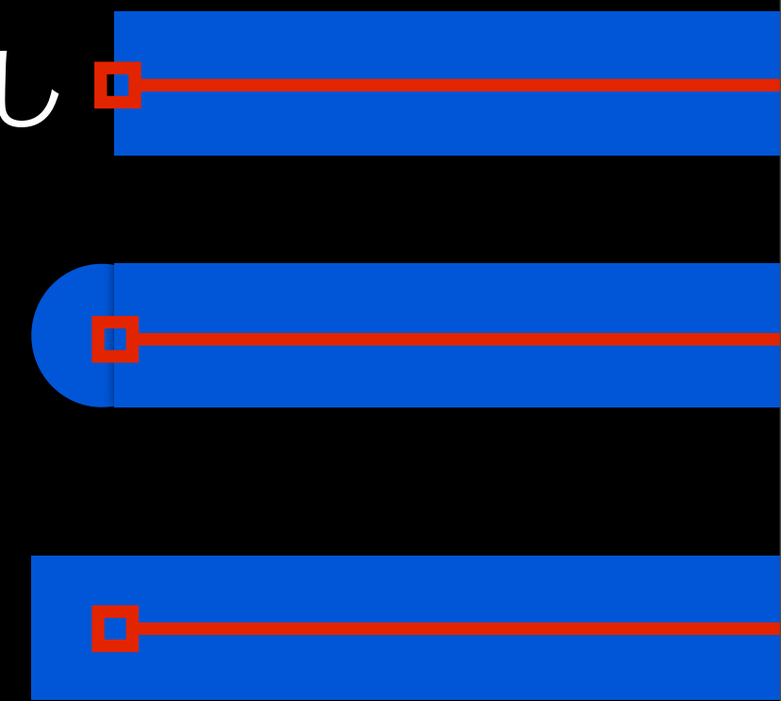
- DEMO

- BezierLineDash

端の処理

ラインキャップスタイル

- lineCapStyleプロパティ
- CGLineCap
- kCGLineCapButt　キャップなし
- kCGLineCapRound　丸い端
- kCGLineCapSquare　四角



ヒットテスト

その位置にパスがあるか

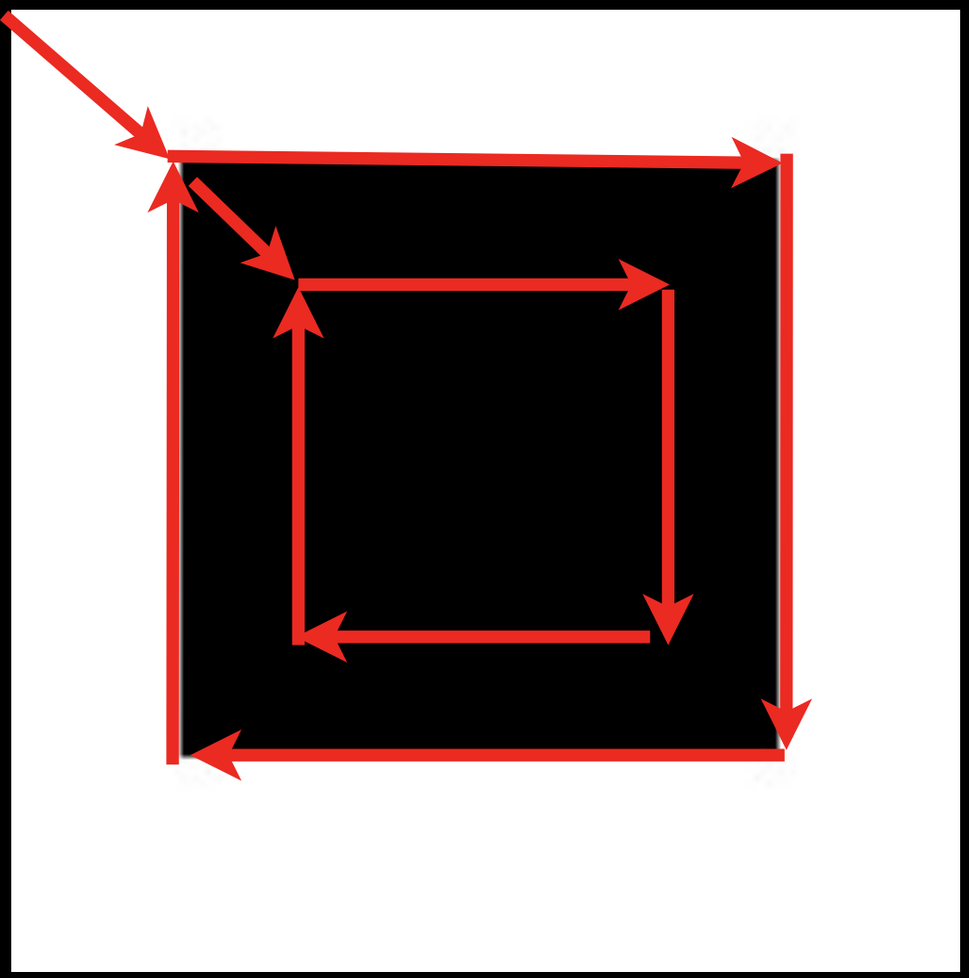
- - (BOOL)containsPoint:(CGPoint)point
- 指定した点をパスのオブジェクト無いに含むか？
- 線のセグメント上のヒットテストは後述

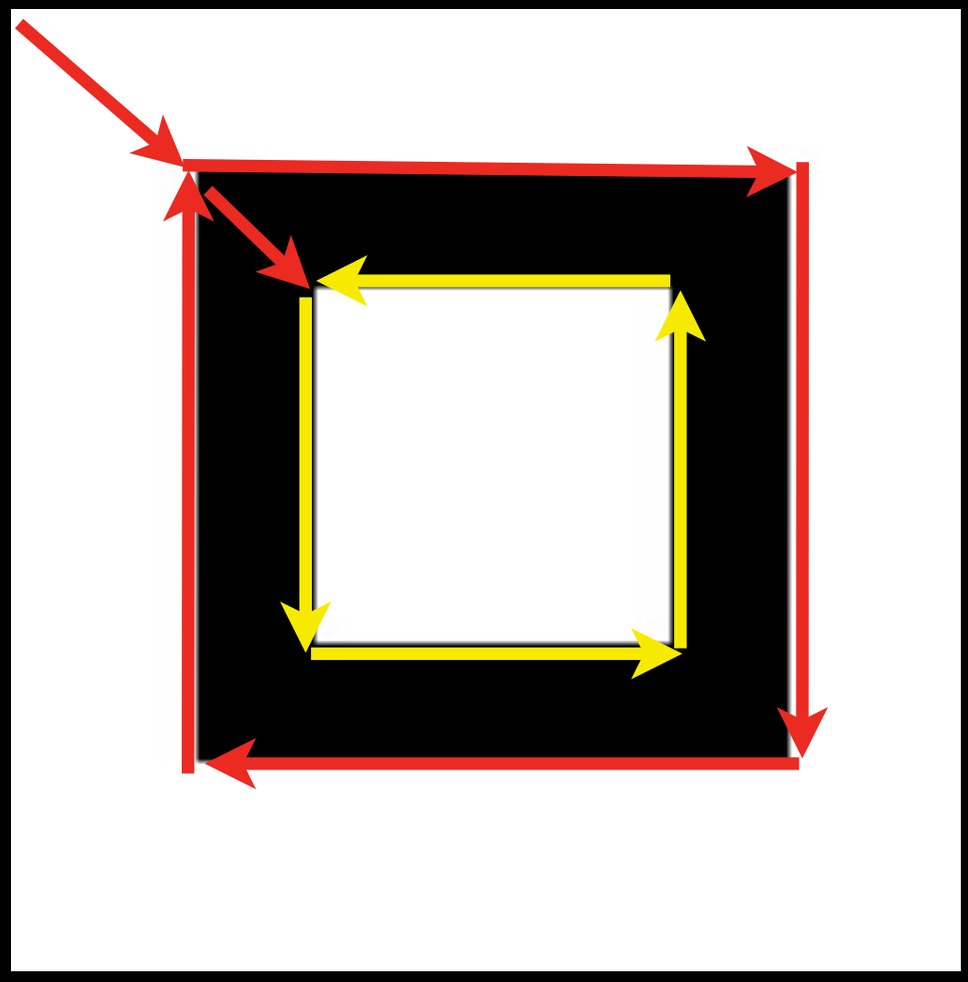
くりぬき

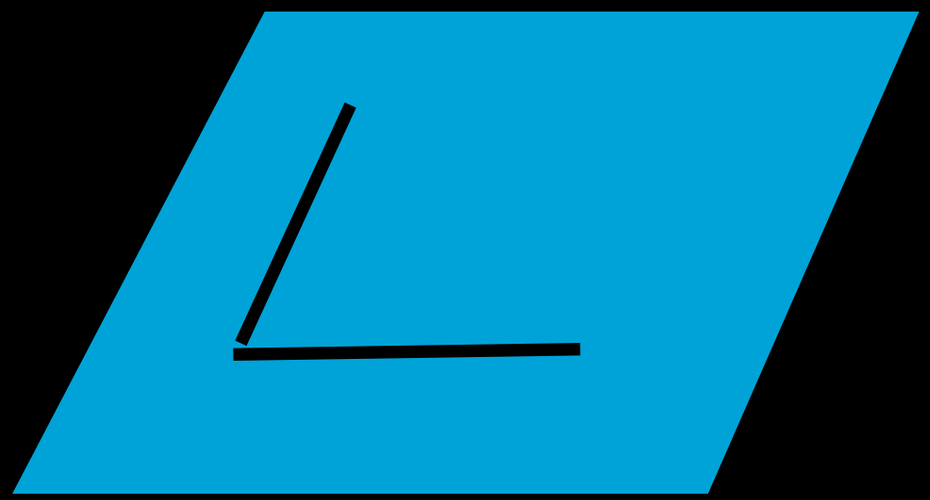
2つの処理方法

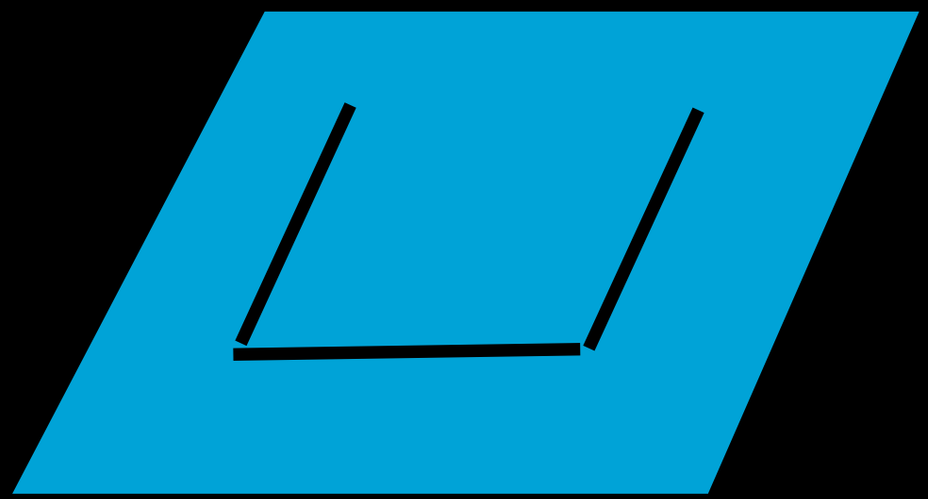
- くりぬき方法（デフォルト）
- Non-Zero ルール

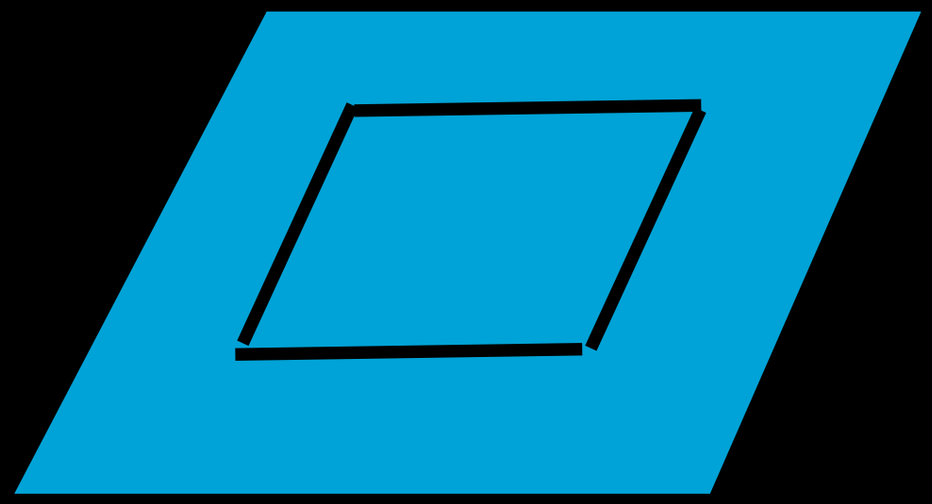
```
thePath.usesEvenOddFillRule = NO;
```







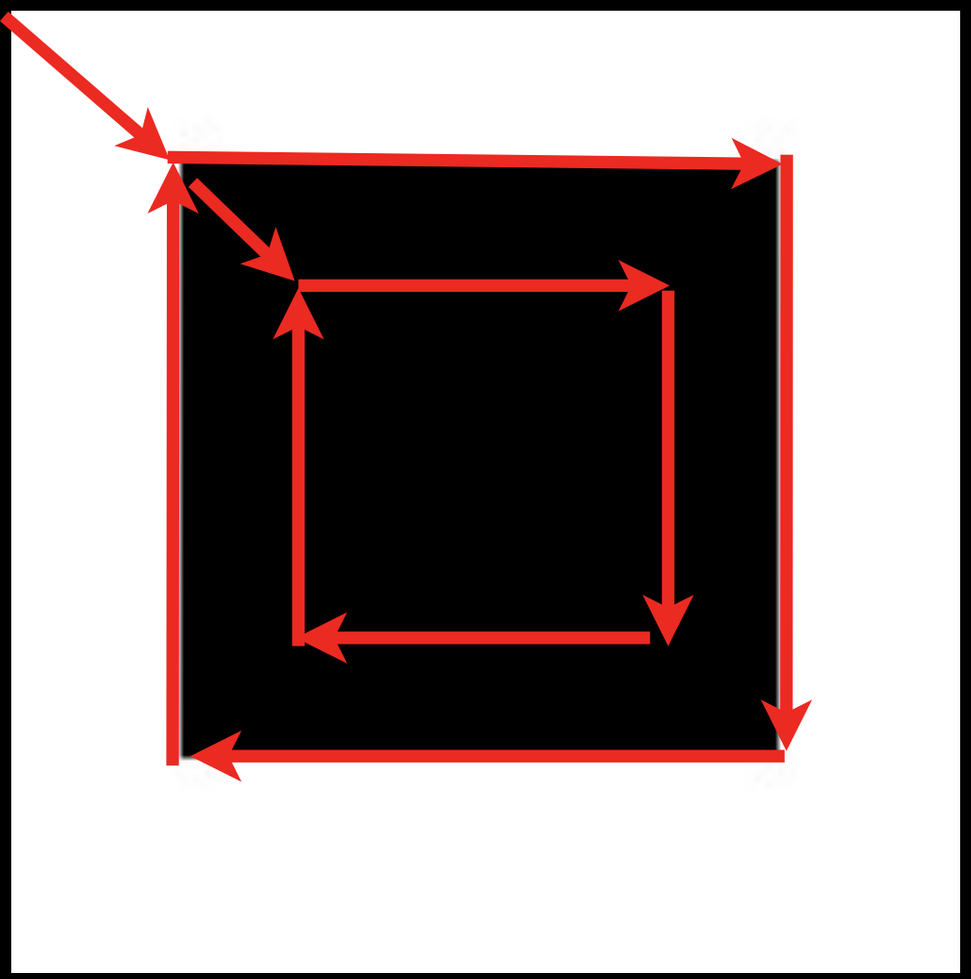


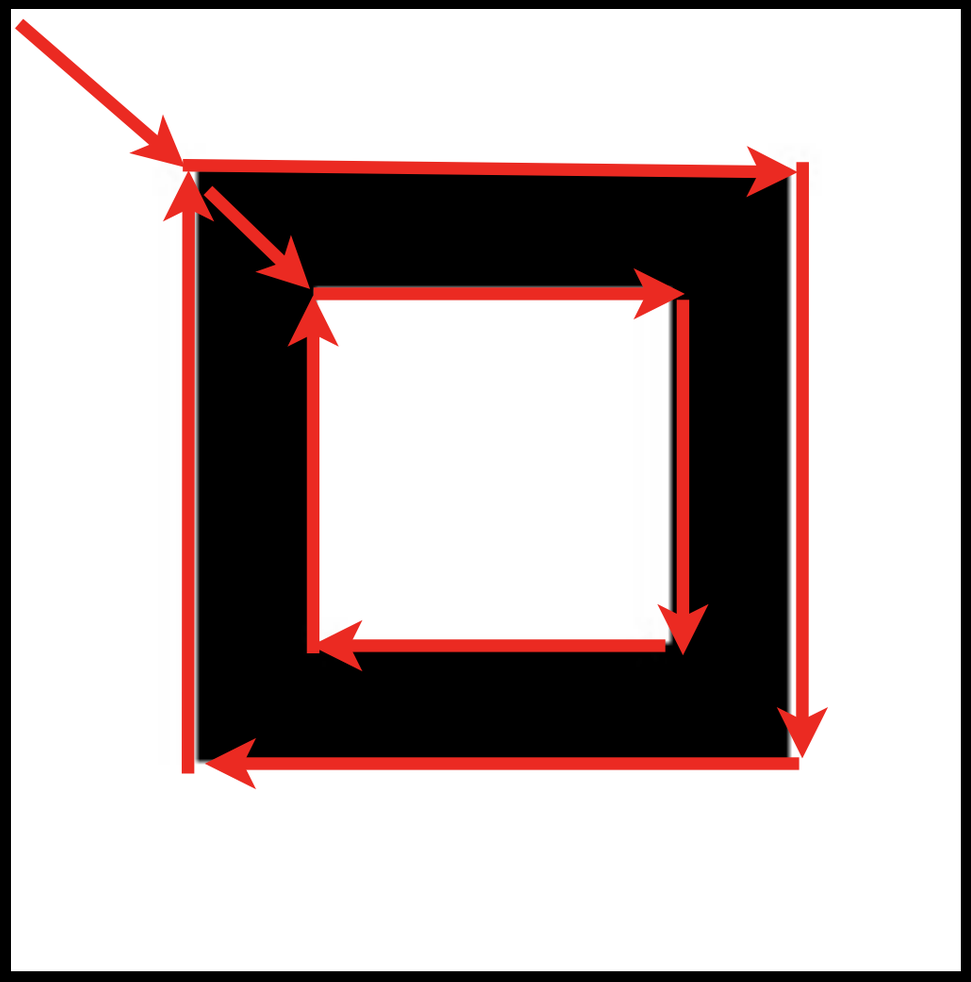


- 参考
- パスの順序を反転する
- - (UIBezierPath *)bezierPathByReversingPath NS_AVAILABLE_IOS(6_0);

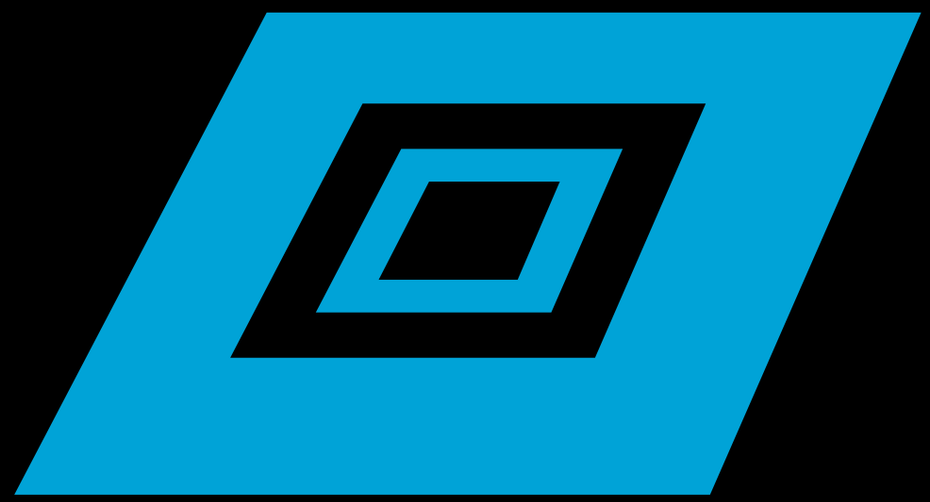
- くりぬき方法
- Even-Odd ルール
- （usesEvenOddFillRuleプロパティを使用）

```
thePath.usesEvenOddFillRule = NO;thePath.usesEvenOddFillRule = YES;
```





2013年 3月 30日 土曜日



- DEMOはしませんが、サンプル置いときます
- BezierNonZero
- BezierEvenOdd

クリッピング

画像をマスク

- 画像のクリッピング
- 画像をパスで切り抜きできます。

```
- [outPath2 addClip];
```

- 画像のクリッピング
- `[outPath2 addClip];`

- DEMO
- BezierAddclip

ブレンド

合成方法の指定

- ブレンドモード
- 塗り、線で下地との混ざり具合を調整します

- ブレンドモード

- `[outPath3 fillWithBlendMode:(CGBlendMode)kCGBlendModeSourceAtop alpha:0.8];`

- CGBlendMode

kCGBlendModeNormal,

kCGBlendModeMultiply,

kCGBlendModeScreen,

kCGBlendModeOverlay,

kCGBlendModeDarken,

kCGBlendModeLighten,

kCGBlendModeColorDodge,

kCGBlendModeColorBurn,

kCGBlendModeSoftLight,

kCGBlendModeHardLight,

kCGBlendModeDifference,

kCGBlendModeExclusion,

kCGBlendModeHue,

kCGBlendModeSaturation,

kCGBlendModeColor,

kCGBlendModeLuminosity,

kCGBlendModeClear,

kCGBlendModeCopy

kCGBlendModeSourceIn

kCGBlendModeSourceOut

kCGBlendModeSourceAtop

kCGBlendModeDestinationOver

kCGBlendModeDestinationIn

kCGBlendModeDestinationOut,

kCGBlendModeDestinationAtop,

kCGBlendModeDestinationOver

kCGBlendModeDestinationIn

kCGBlendModeDestinationOut

kCGBlendModeDestinationAtop

kCGBlendModeXOR

kCGBlendModePlusDarker

kCGBlendModePlusLighter

- DEMO

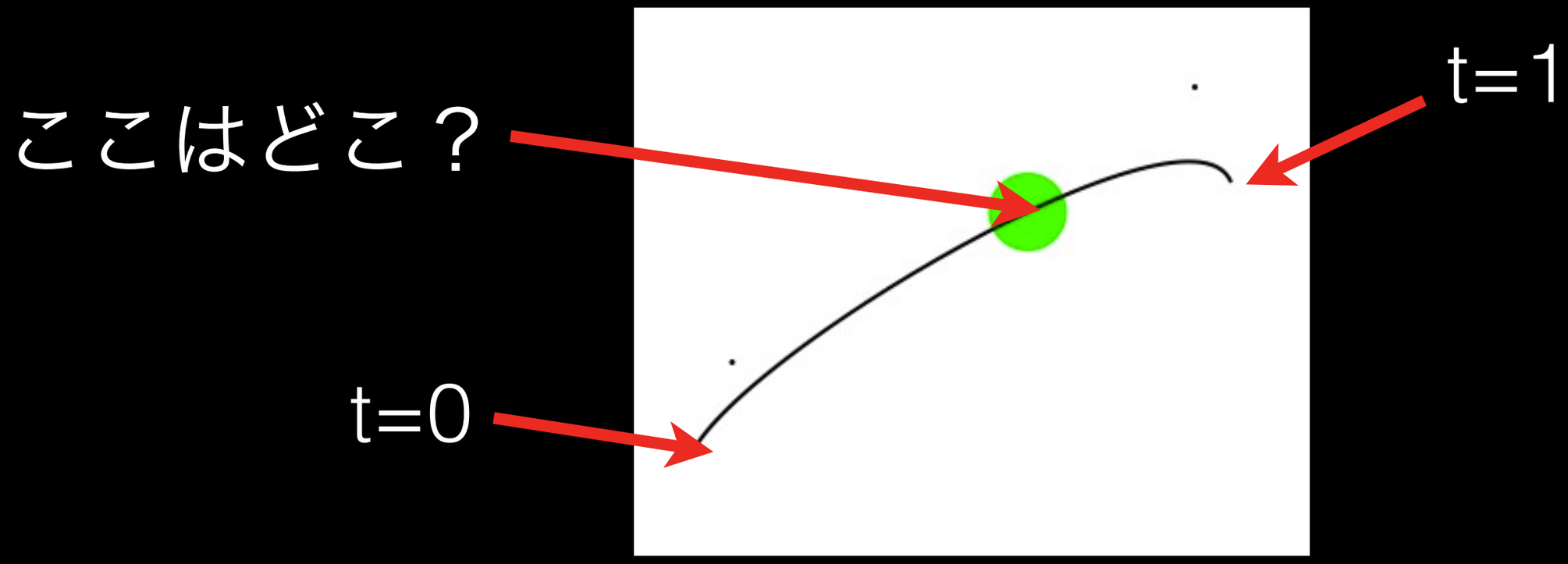
- BezierBlendMode

座標計算

パスの位置を求める

- セグメント上の座標を求める（０≦ｔ≦１）

```
float tp = (1-t);

CGFloat x = t*t*t*x2 + 3*t*t*tp*cp2x + 3*t*tp*tp*cp1x +
tp*tp*tp*x1;

CGFloat y = t*t*t*y2 + 3*t*t*tp*cp2y + 3*t*tp*tp*cp1y +
tp*tp*tp*y1;
```



- DEMO

- BezierSegmentPoint

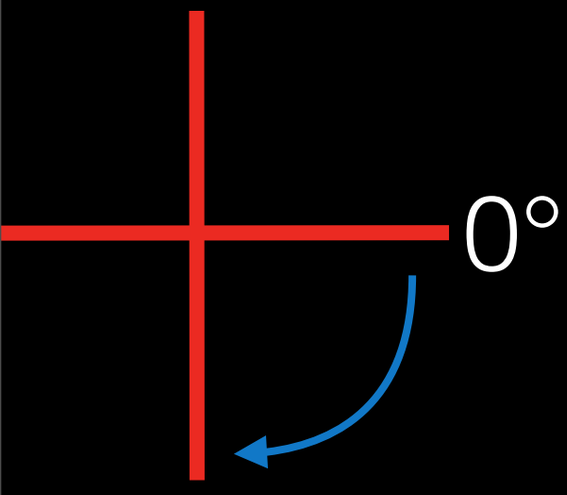
接線の角度

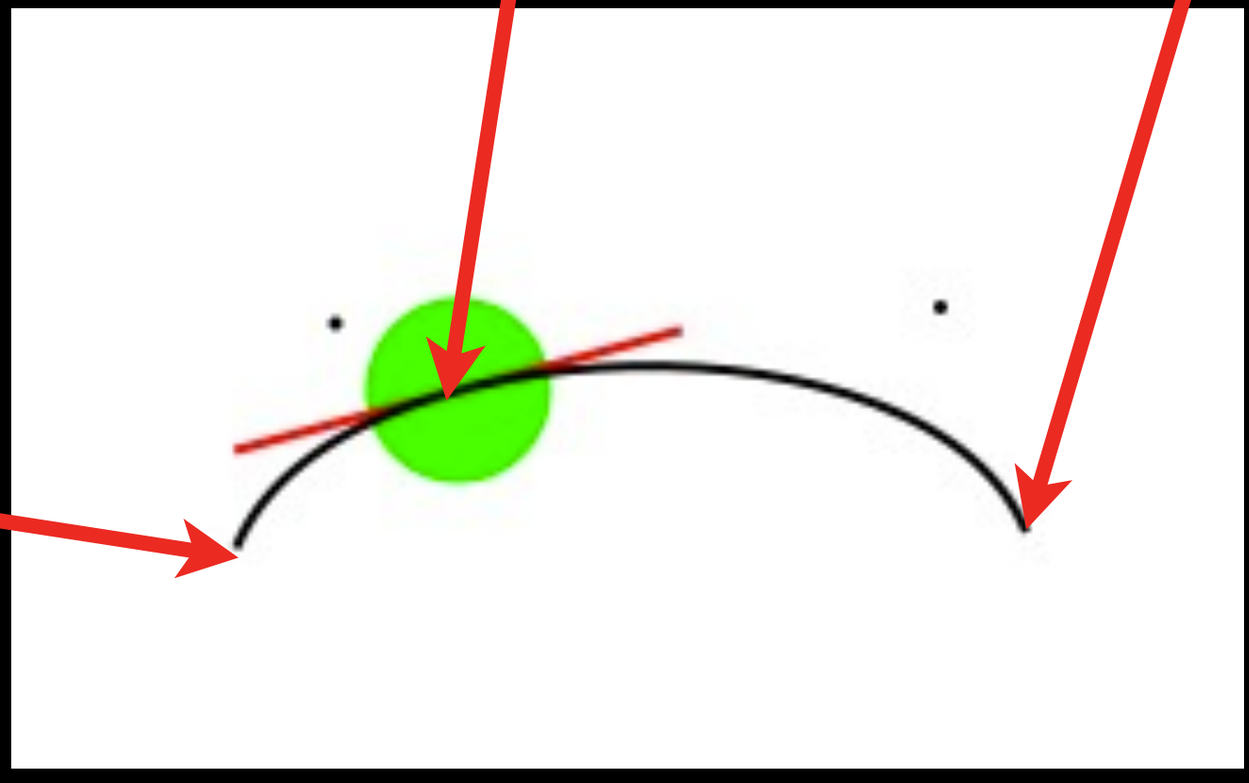
ある地点の傾き

- 接線の角度を求める（０≦ｔ≦１）

```
float tp = (1-t);


CGFloat dx = 3*(t*t*(x2-cp2x)+2*t*tp*(cp2x-cp1x)+tp*tp*(cp1x-
x1));
CGFloat dy = 3*(t*t*(y2-cp2y)+2*t*tp*(cp2y-cp1y)+tp*tp*(cp1y-
y1));

NSLog(@"%.2f,(%.2f,%.2f)",degrees(atan2(dy,dx)),dx,dy);
```

ラジアンを度に変換しているマクロ関数





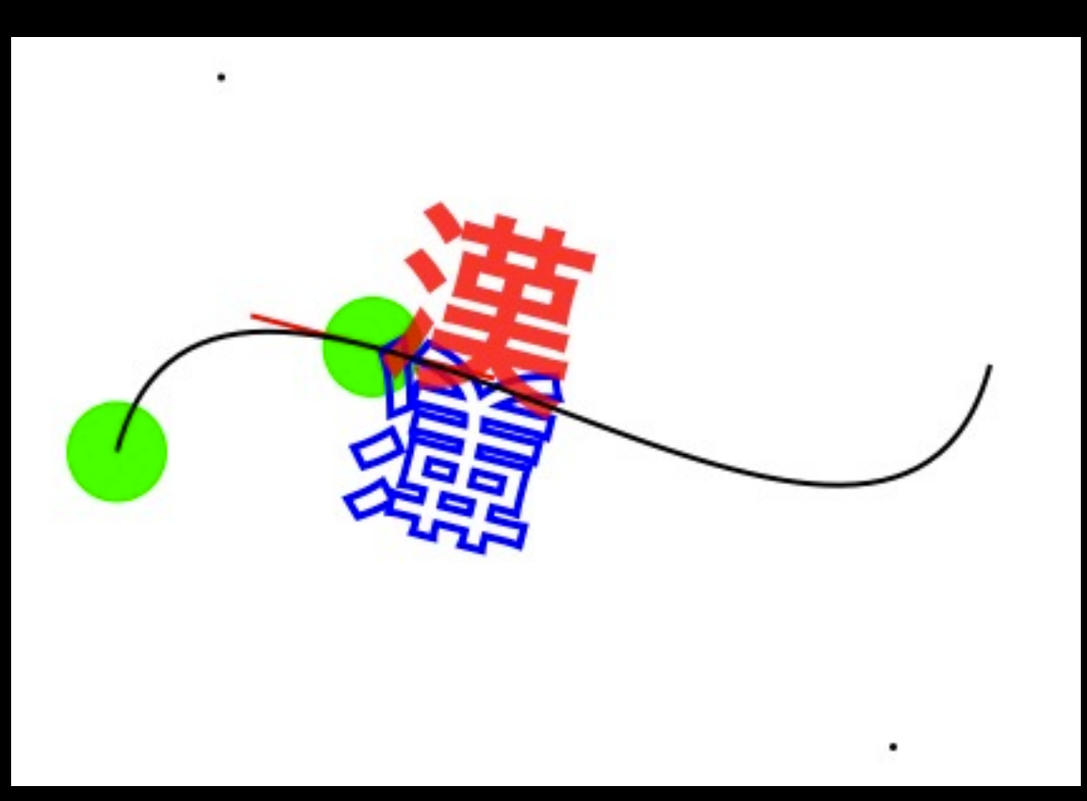
- DEMO

- BezierSegmentIncline

パスの変形

アフィン変換

- パスに沿って文字を描画。
- グリフとグリフから取ったパス、両方で描画しています。



グリフのパス取得はソースを見てね

パスの変形

```
CGAffineTransform affine =
CGAffineTransformMakeTranslation(x,y);
```

```
affine =
CGAffineTransformRotate(affine,
atan2(dy,dx));

[glyphBezierPath
applyTransform:affine];
```

- DEMO

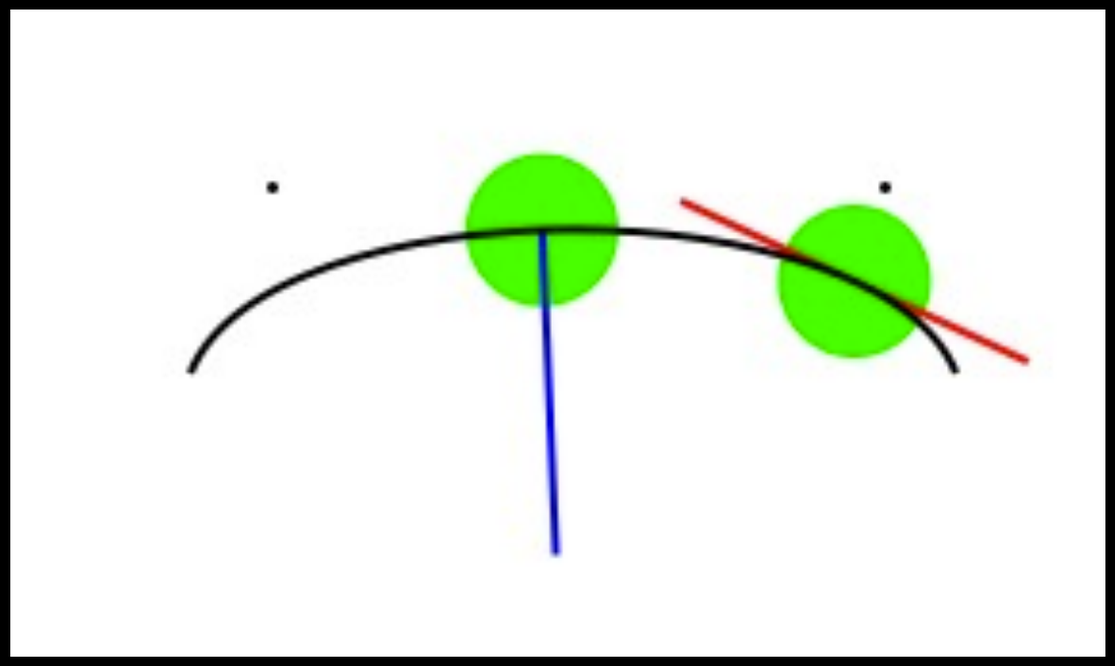
- BezierSegmentInclineString

ヒットテスト

セグメント上で

- 任意の点から一番近いポイントを求める。

- 計算でできるみたいなんだけど、よくわからなかったので、t=0からt=1までのセグメント上の点の位置を計算して、セグメント上の距離を比較してみました。



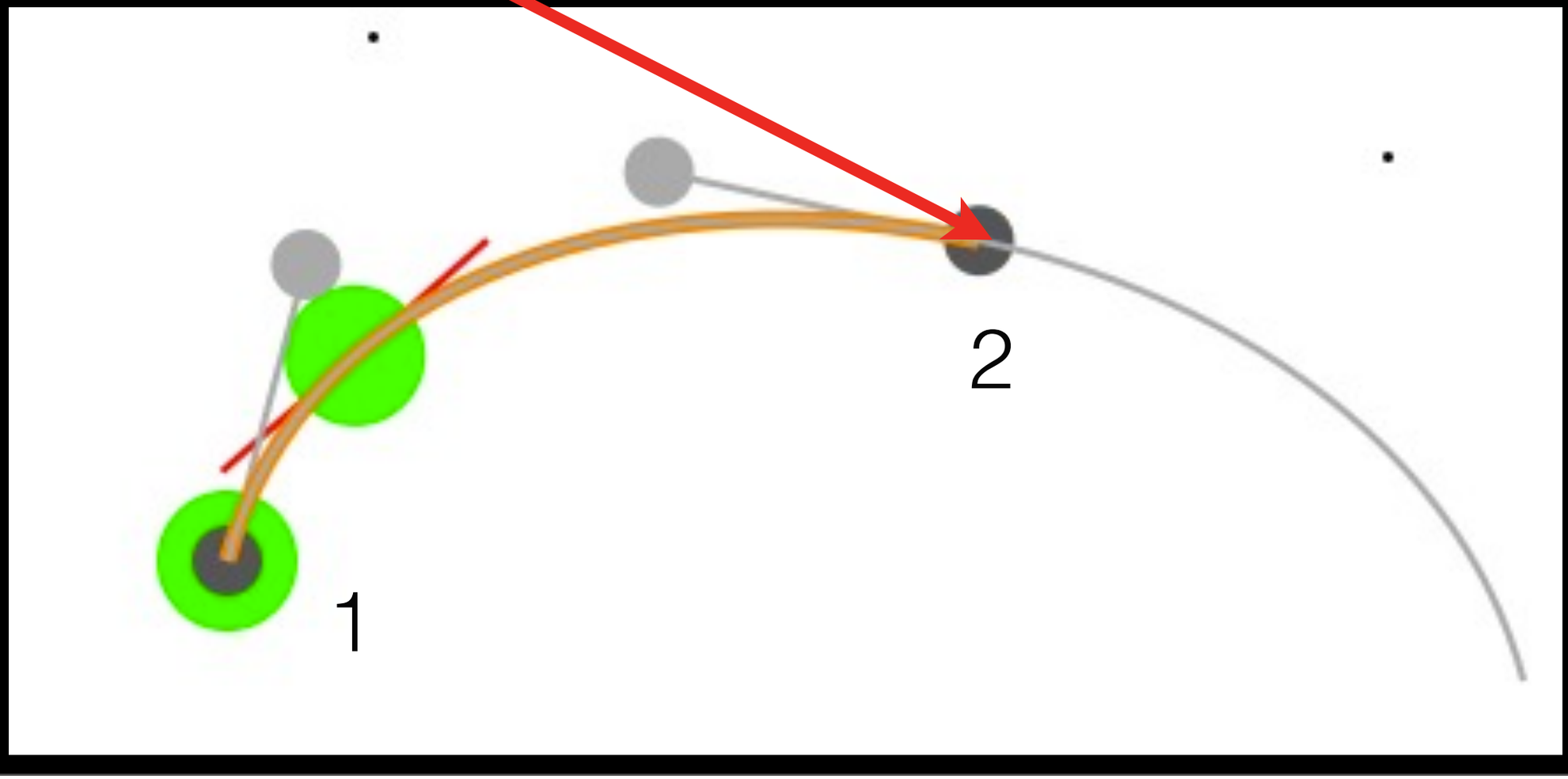
- 線上でのヒットテストに使うことができます

- DEMO

- BezierSegmentNearPoint

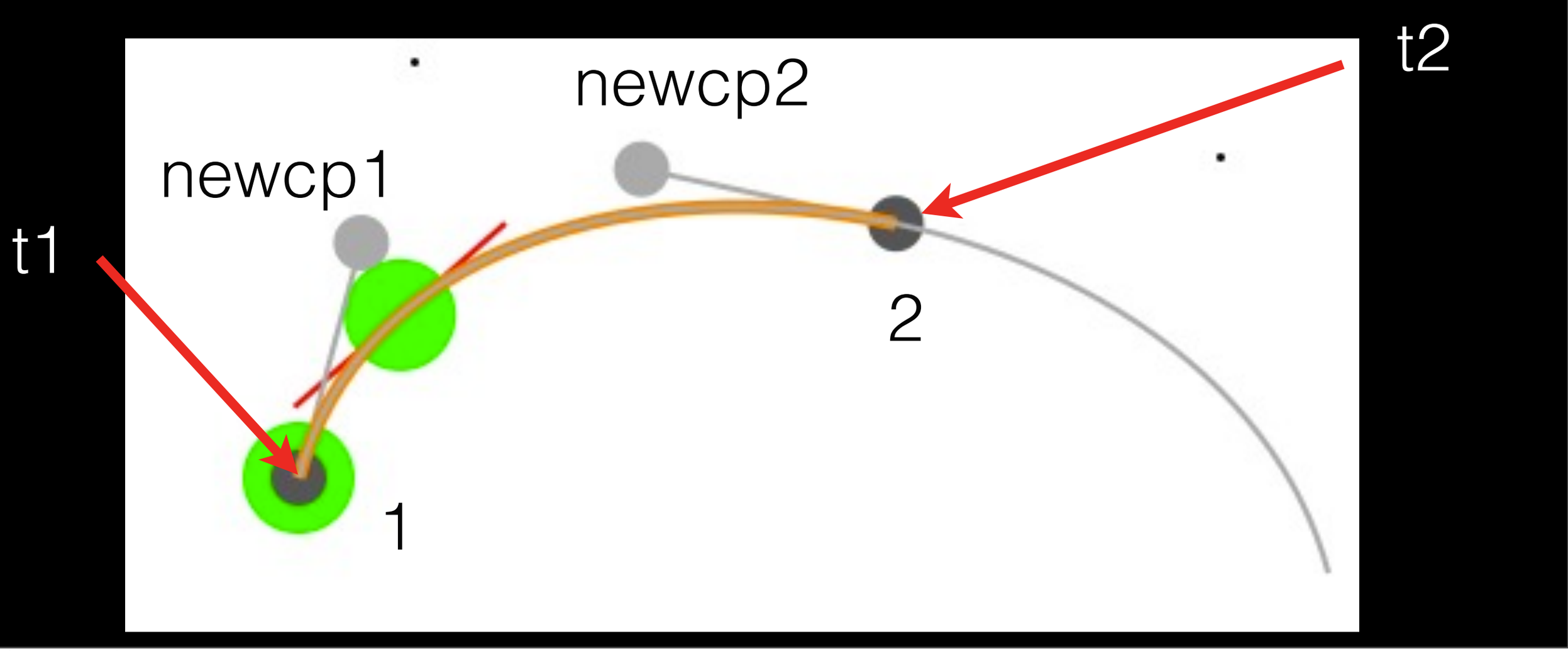
パスの分割

- パスの分割
- 2の位置で分割する（$0 \leqq t1 \leqq 1, 0 \leqq t2 \leqq 1,\ t1 < t2$）



- パスの分割

```
double t1p = 1-t1;
double t2p = 1-t2;
double newx1 = t1p*t1p*t1p*x1 + 3*t1*t1p*t1p*cp1x + 3*t1*t1*t1p*cp2x + t1*t1*t1*x2;
double newy1 = t1p*t1p*t1p*y1 + 3*t1*t1p*t1p*cp1y + 3*t1*t1*t1p*cp2y + t1*t1*t1*y2;
double newcp1x = t1p*t1p*(t2p*x1+t2*cp1x) + 2*t1p*t1*(t2p*cp1x+t2*cp2x) + t1*t1*(t2p*cp2x+t2*x2);
double newcp1y = t1p*t1p*(t2p*y1+t2*cp1y) + 2*t1p*t1*(t2p*cp1y+t2*cp2y) + t1*t1*(t2p*cp2y+t2*y2);
double newcp2x = t2p*t2p*(t1p*x1+t1*cp1x) + 2*t2p*t2*(t1p*cp1x+t1*cp2x) + t2*t2*(t1p*cp2x+t1*x2);
double newcp2y = t2p*t2p*(t1p*y1+t1*cp1y) + 2*t2p*t2*(t1p*cp1y+t1*cp2y) + t2*t2*(t1p*cp2y+t1*y2);
double newx2 = t2p*t2p*t2p*x1 + 3*t2*t2p*t2p*cp1x + 3*t2*t2*t2p*cp2x + t2*t2*t2*x2;
double newy2 = t2p*t2p*t2p*y1 + 3*t2*t2p*t2p*cp1y + 3*t2*t2*t2p*cp2y + t2*t2*t2*y2;
```



- DEMO

- BezierSplit

最後に

- 最初は取っ付きにくいですが、慣れるといろいろ自由な曲線を書くことができます。

- いろいろ試してみてください。

- ありがとうございました。

- ありがとうございました。