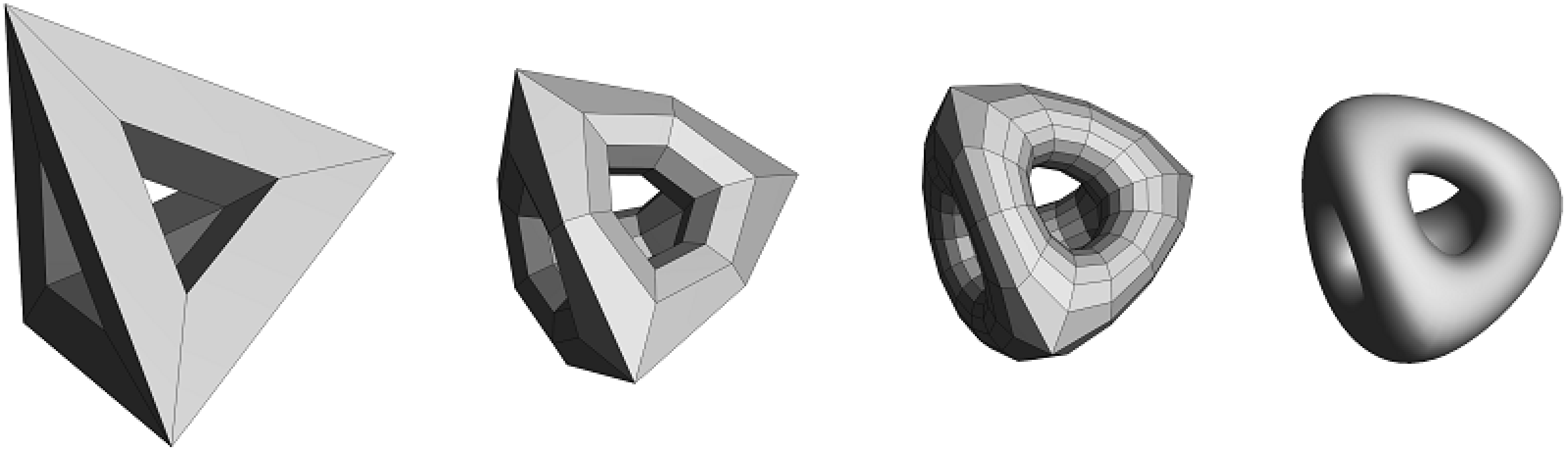
コンピュータグラフィクス論

– モデリング(2) –

2015年4月23日

高山 健志

サブディビジョン曲面

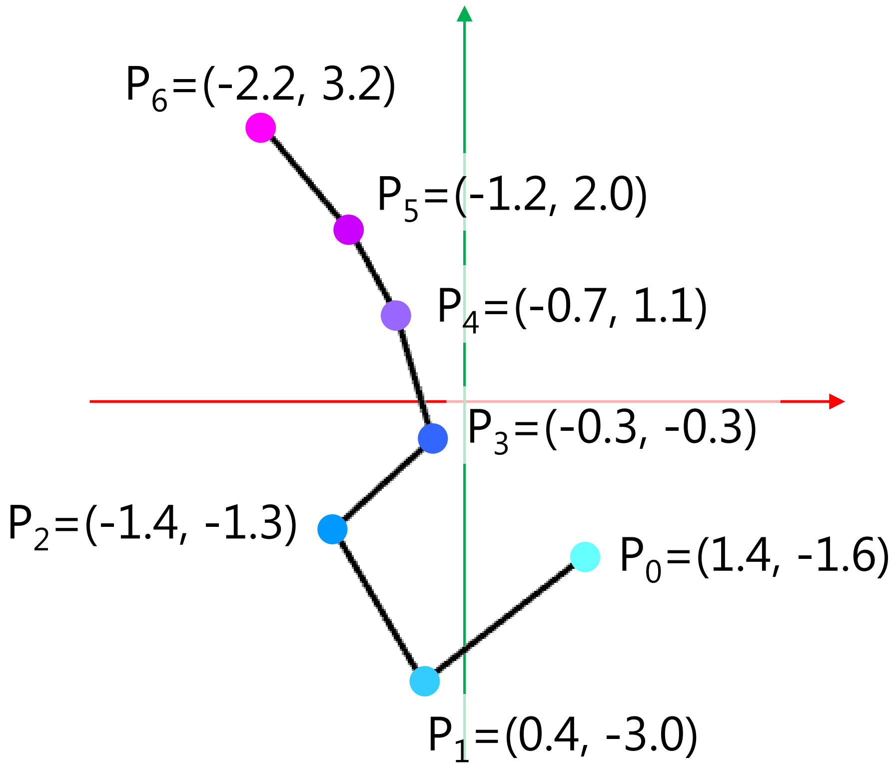


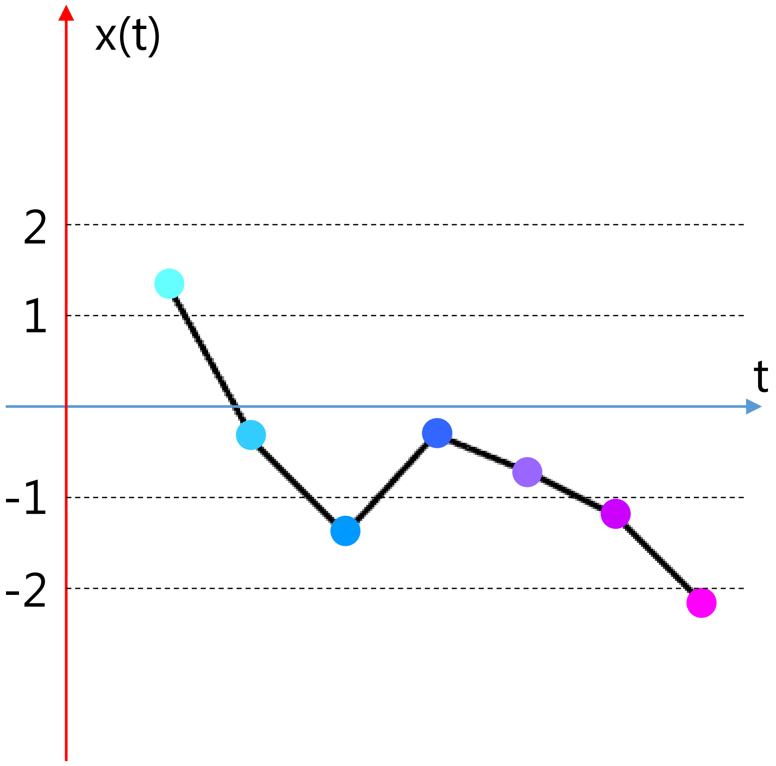
その前に・・・

Bスプライン (**B**asis-Spline)

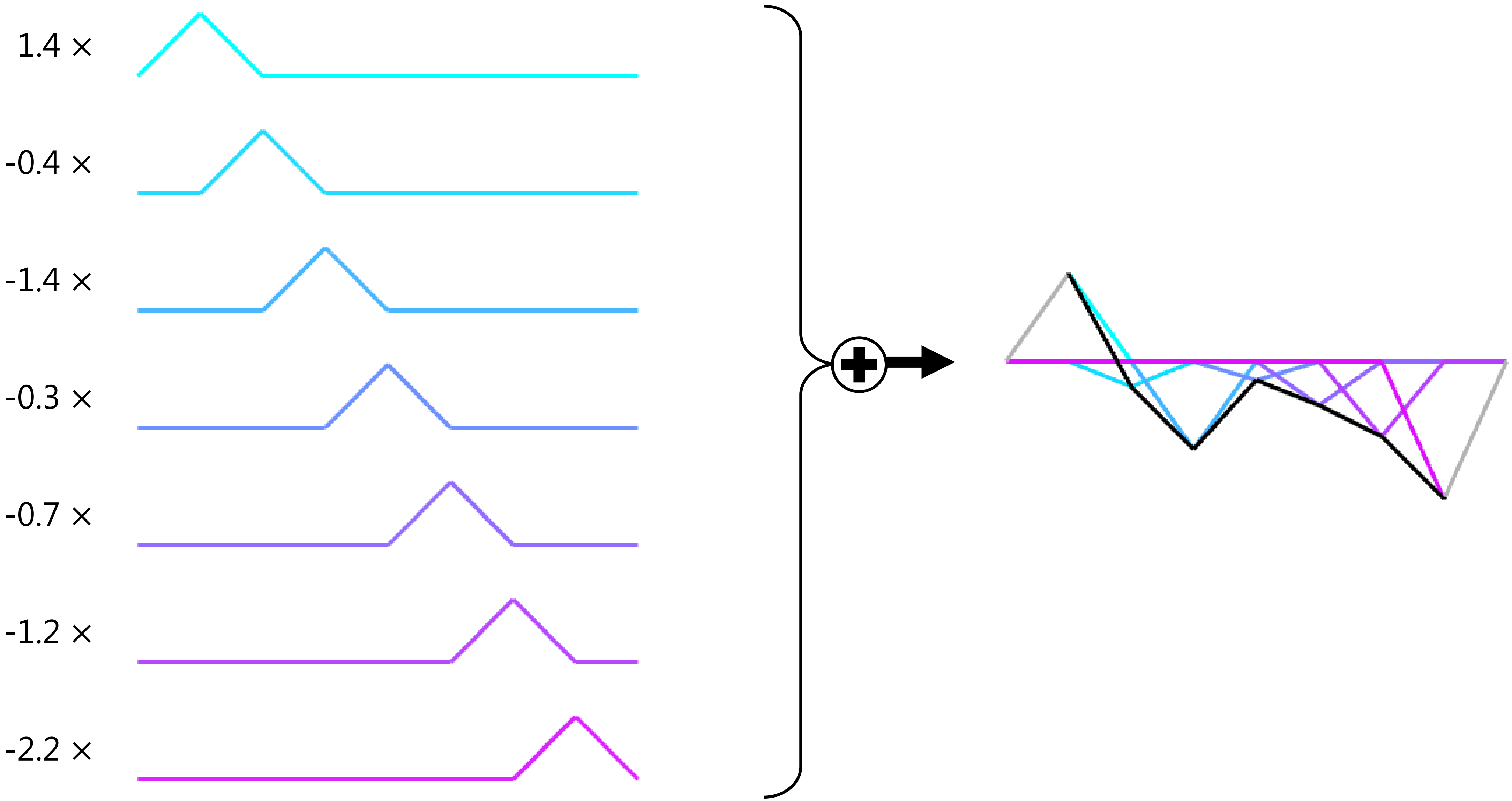
サブディビジョンを理解する上で大事なので、
やっぱり説明します

折れ線再考



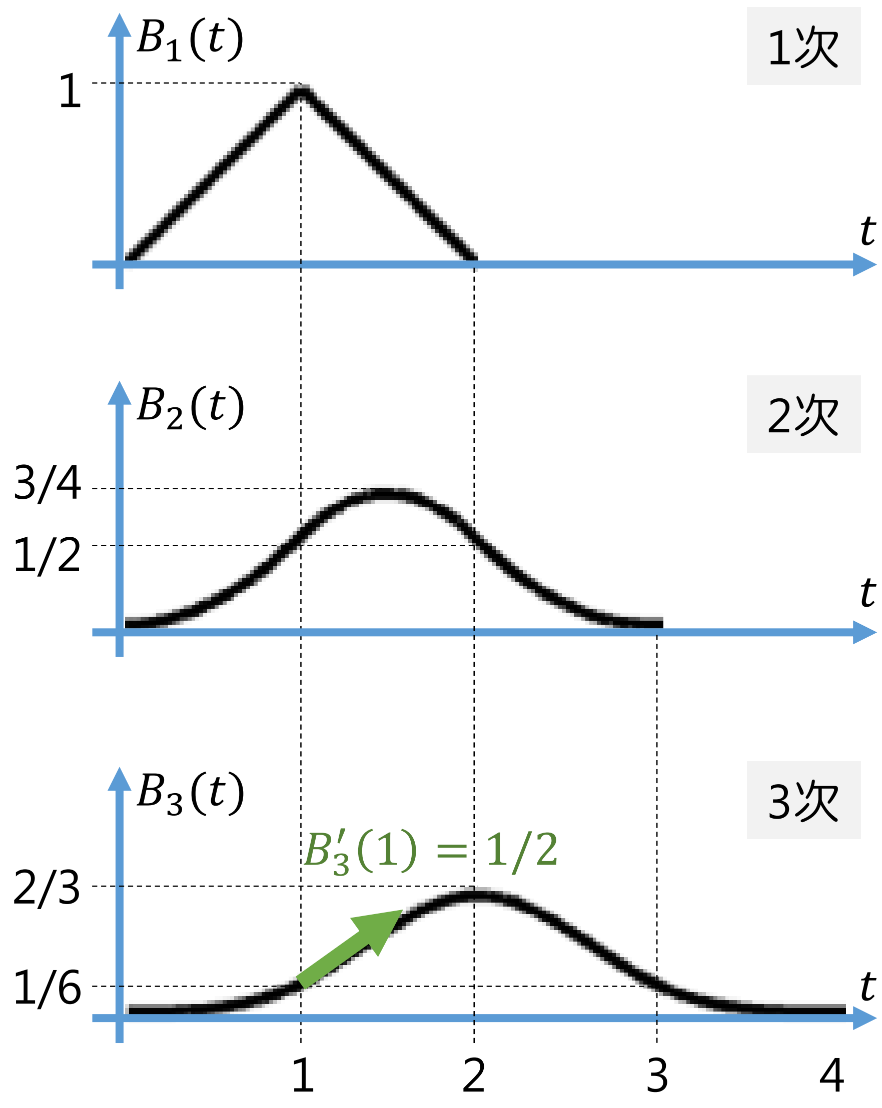


1次の基底関数を用いた折れ線の表現

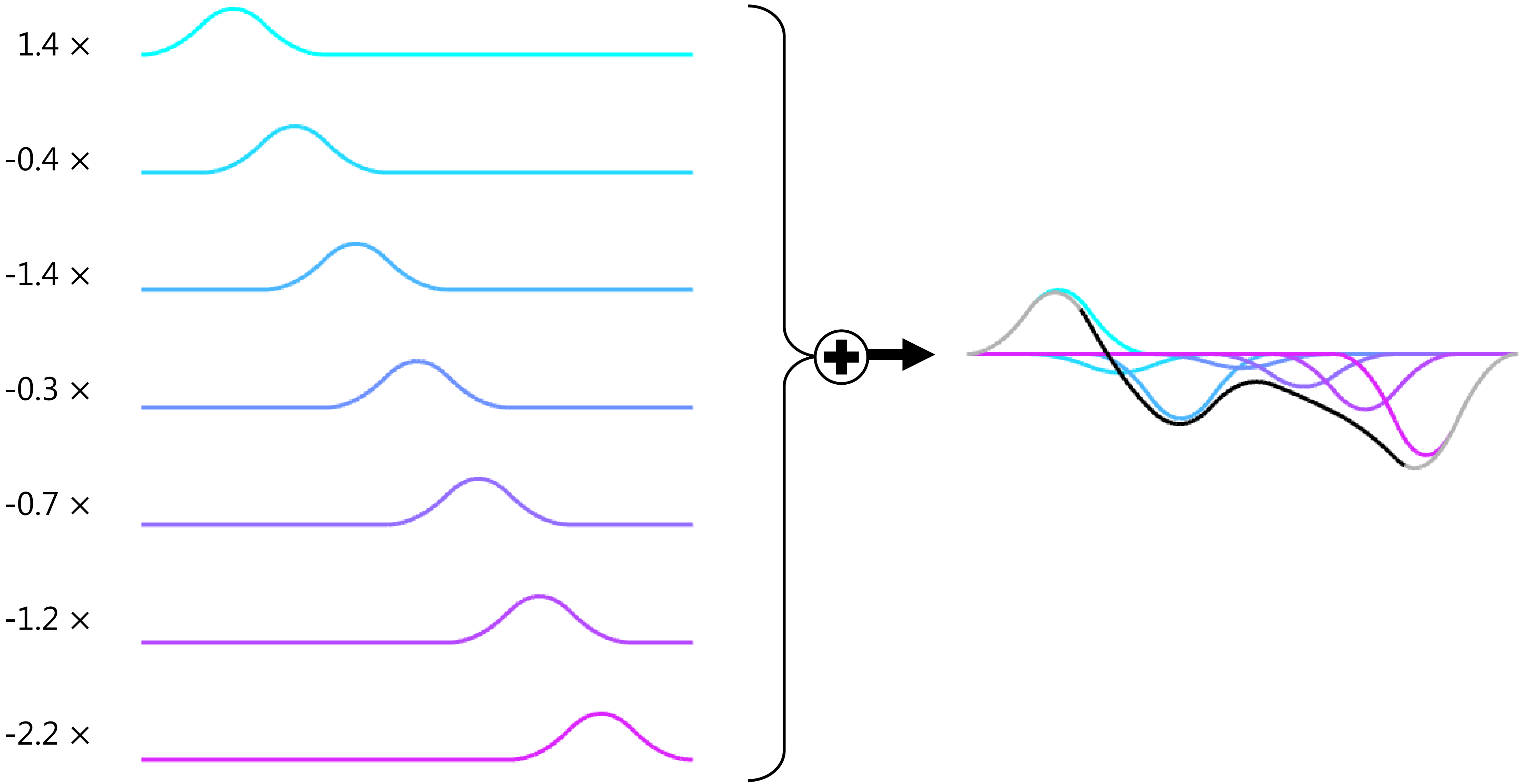


de Boor の n次基底関数

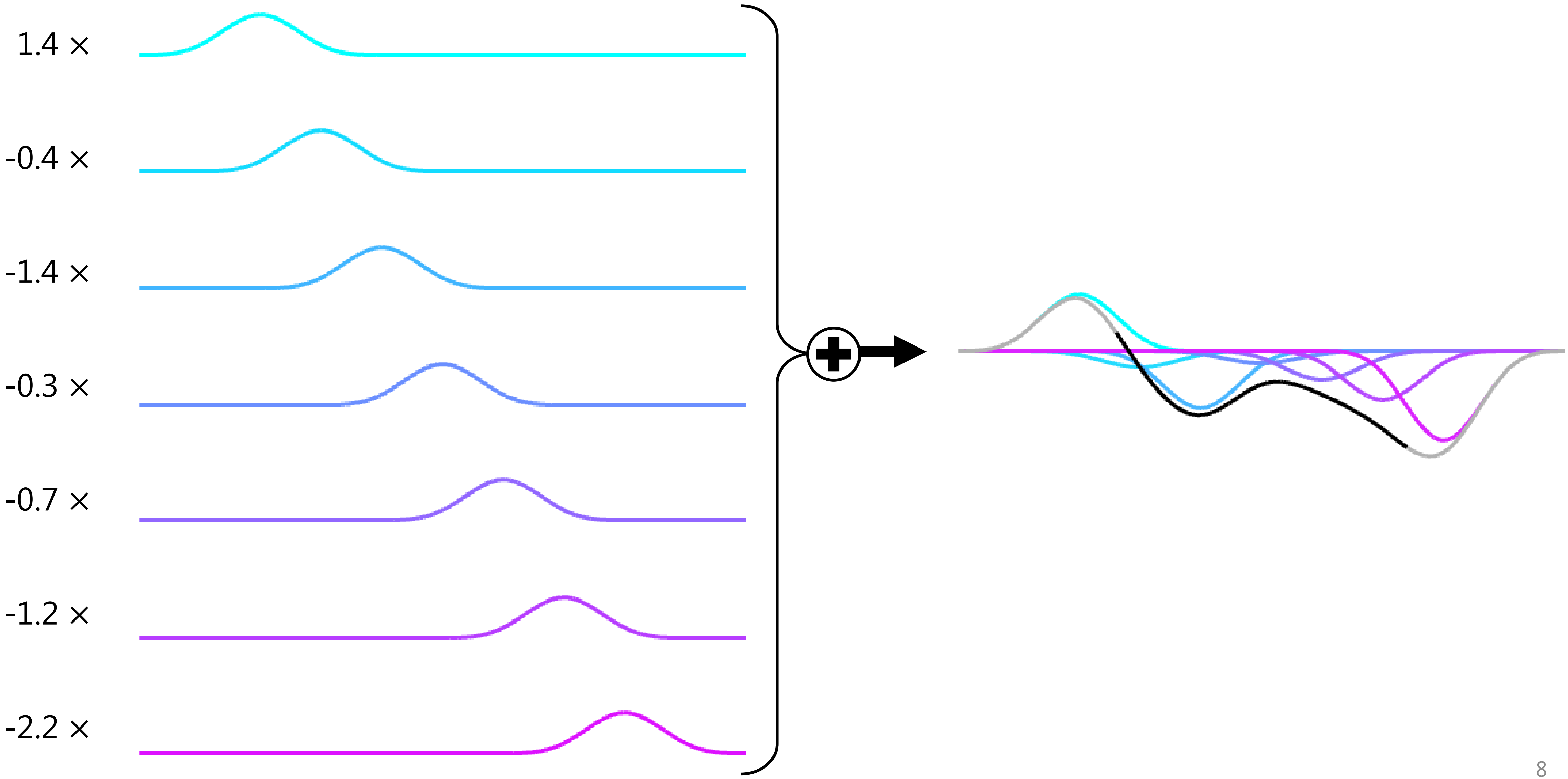
- 再帰的な定義：
  - $B_0(t) = \begin{cases} 1, & 0 \le t < 1 \\ 0, & \text{otherwise} \end{cases}$
  - $B_n(t) = \frac{t}{n} B_{n-1}(t) + \frac{n+1-t}{n} B_{n-1}(t-1)$
- 性質：
  - n次区分多項式
  - [0, n+1] の外では常にゼロ (local support)
  - C^{n-1}連続



2次の基底関数を使う ➔ 2次Bスプライン



3次の基底関数を使う ➔ 3次Bスプライン



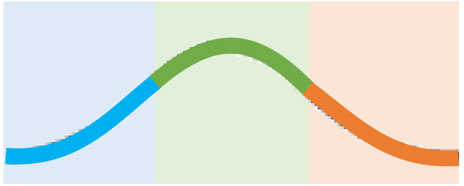
基底関数の大事な性質：partition-of-unity

- Bスプライン曲線のX座標： $x(t) = \sum_i x_i \, B_n(t-i)$

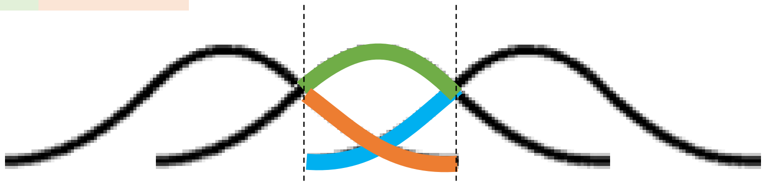
- すべての制御点の座標 x_i を定数 c だけ平行移動することを考える：
  - $x(t) = \sum_i (x_i + c) \, B_n(t-i)$

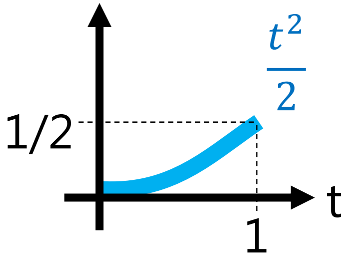
    $$= \sum_i x_i B_n(t-i) + c \underbrace{\sum_i B_n(t-i)}_{1}$$

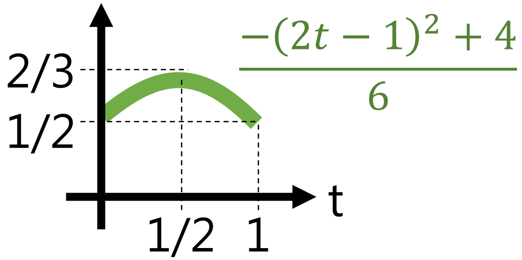
  - partition-of-unityを満たせば、補間結果も c だけ平行移動したものになる

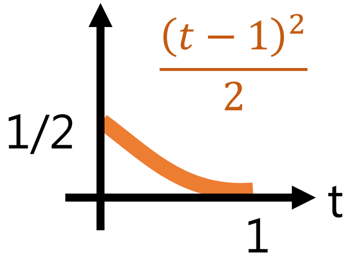


2次の場合

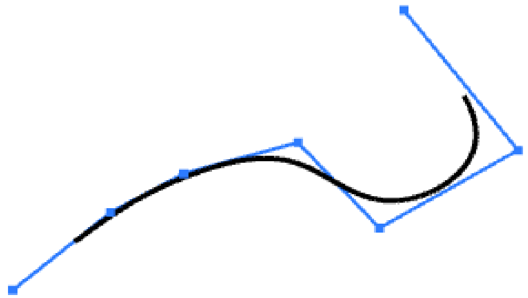
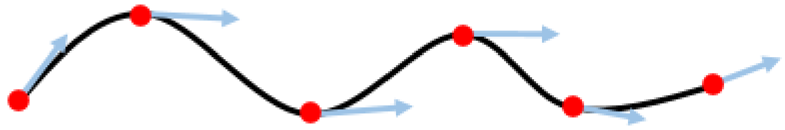








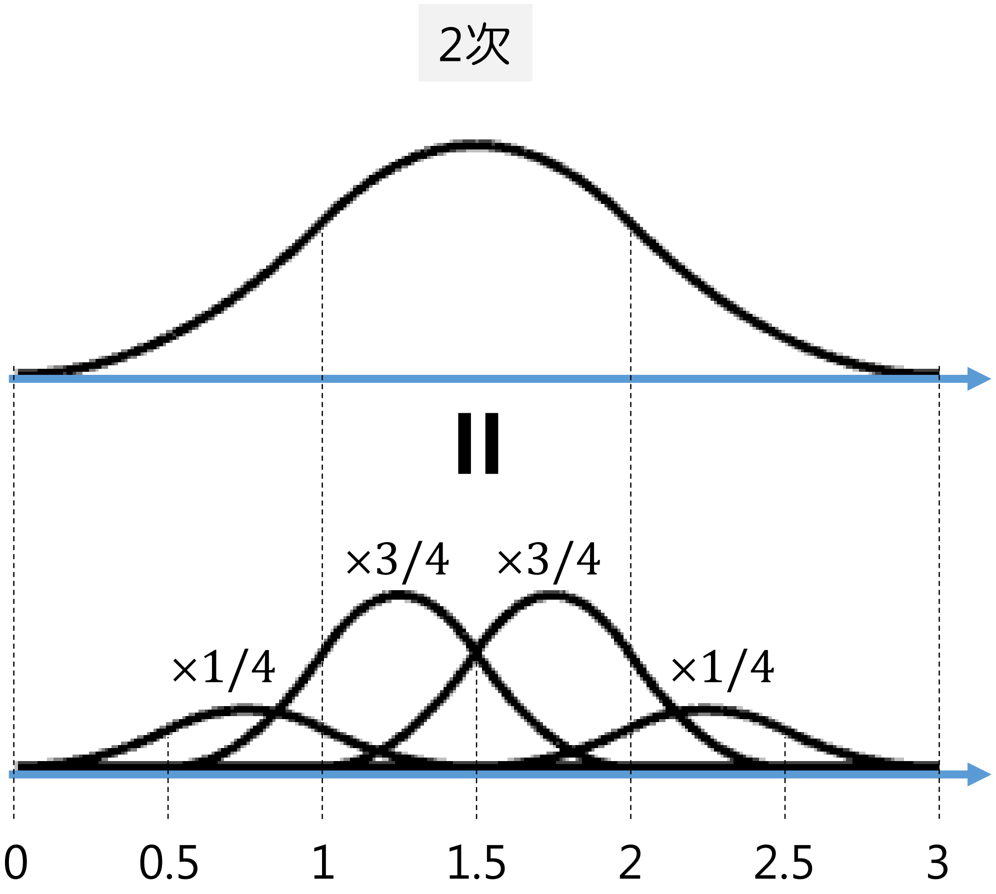
3次Bスプライン曲線と3次Catmull-Rom曲線

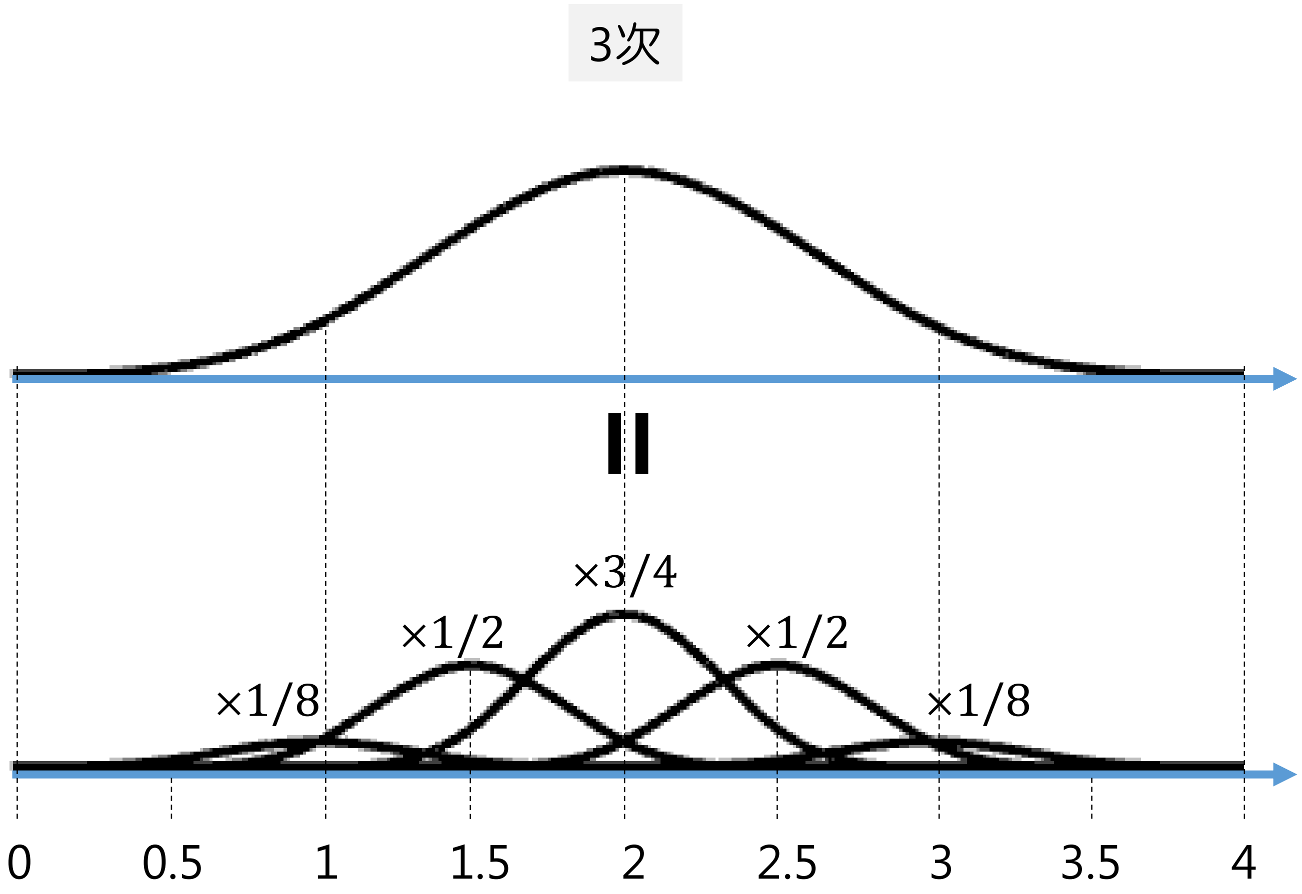
		
数学的な表現	区分3次曲線	区分3次曲線
定義の方法	3次基底関数の線形結合	$t = t_k$における座標値を指定すると、そこでの微分値を自動設定 各区間を3次エルミート補間
制御点を通る？	通らない	通る
区間境界の連続性	C^2連続	C^1連続

Bスプラインからサブディビジョンへ

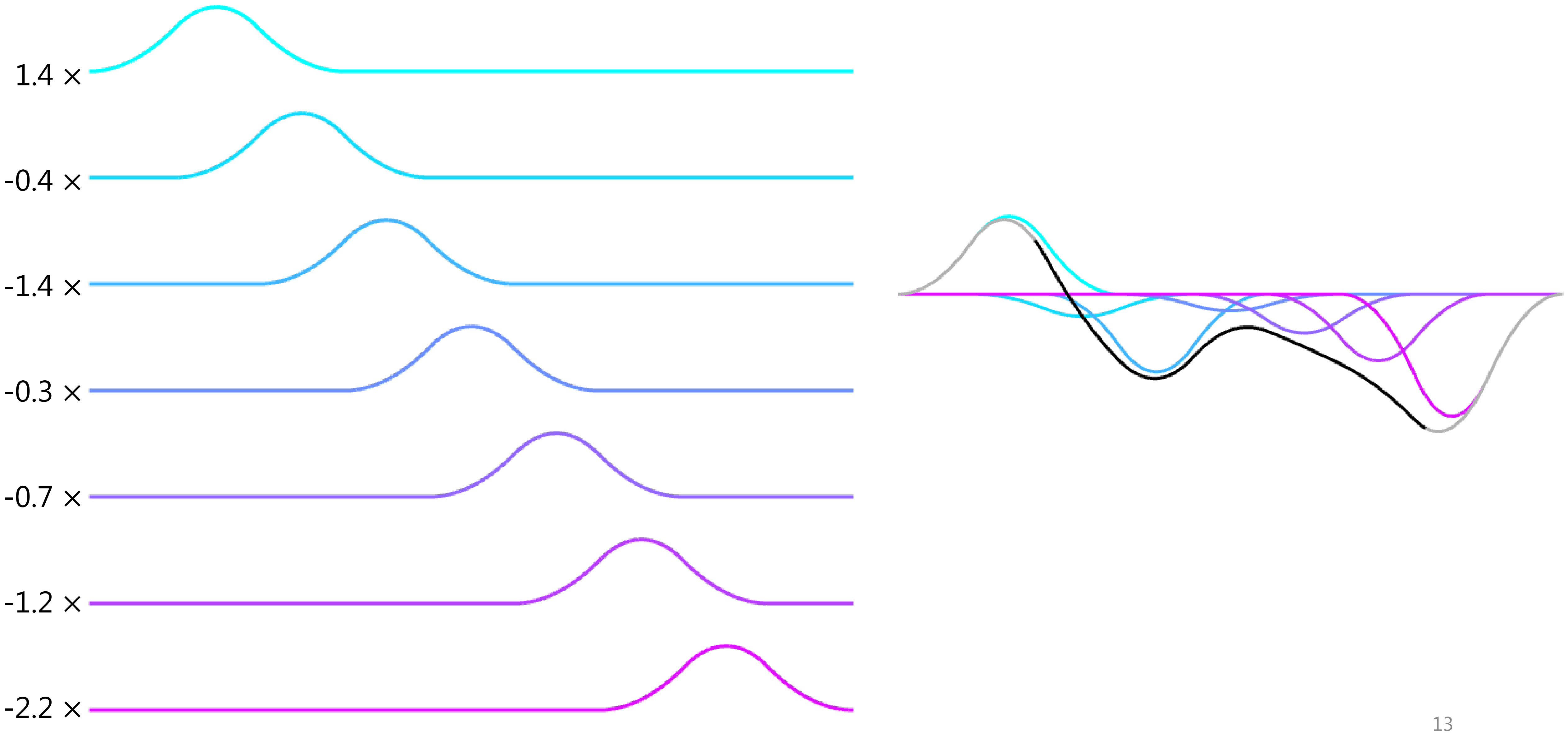
基底関数のもう一つの重要な性質

- 同じ基底関数の local support を半分にしたものの重み付け和に分解可能

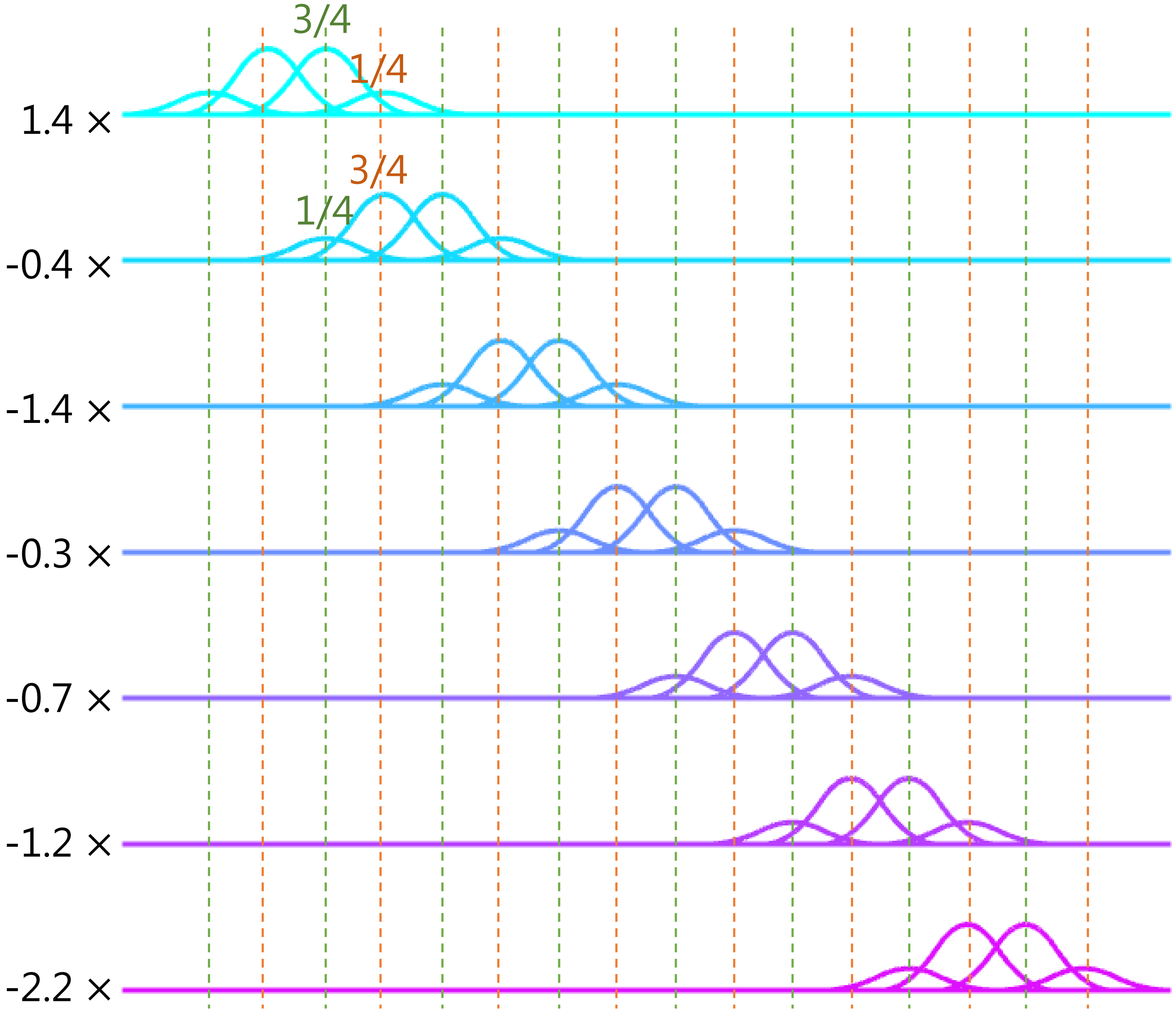


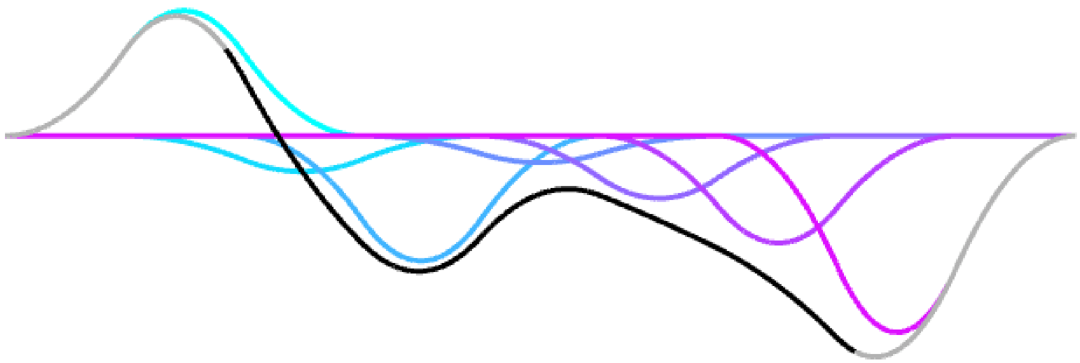


2次Bスプライン曲線の分割

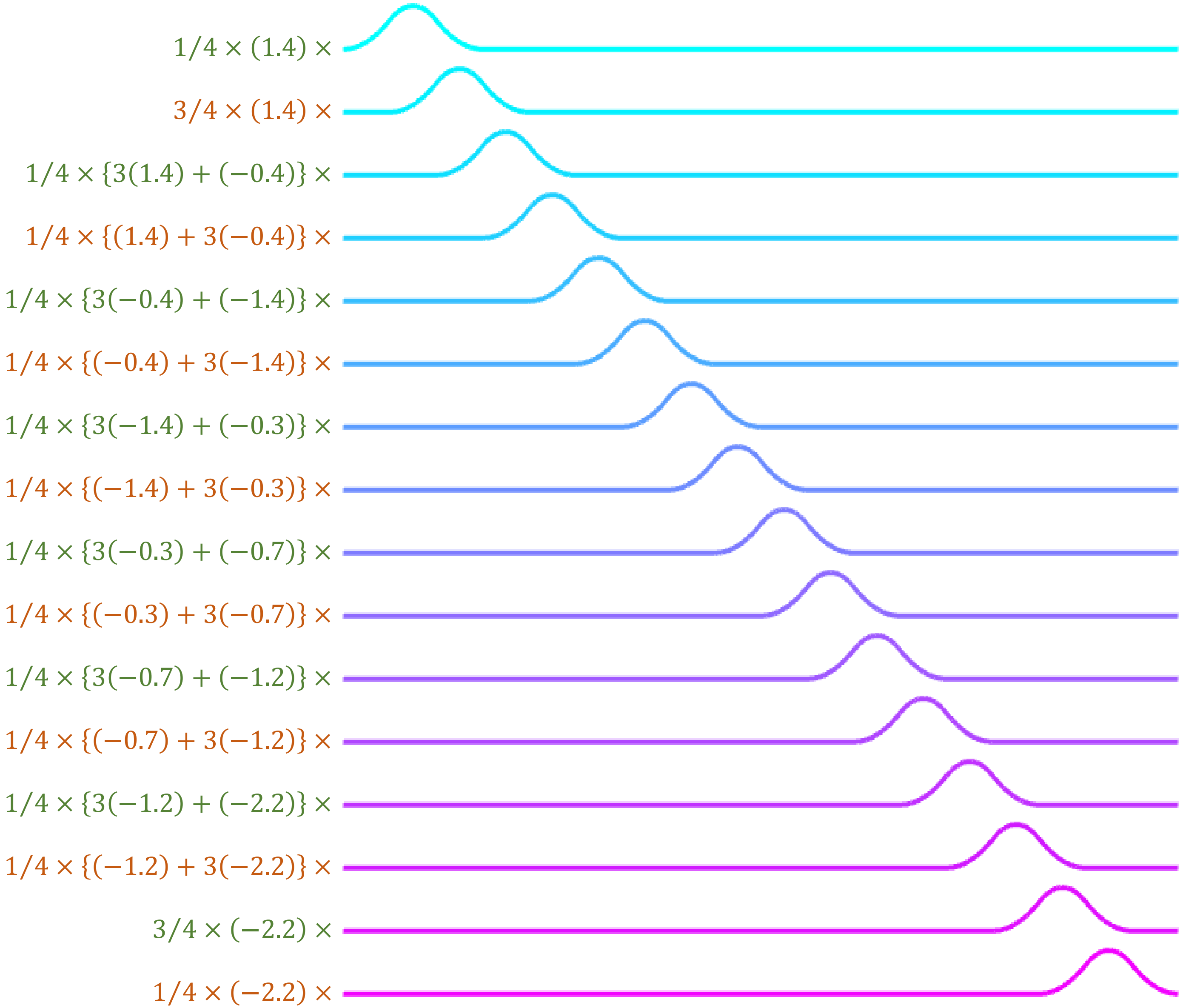


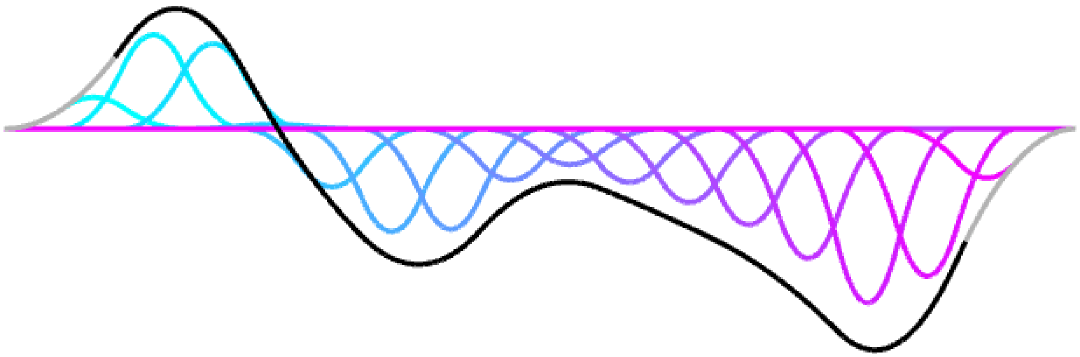
2次Bスプライン曲線の分割



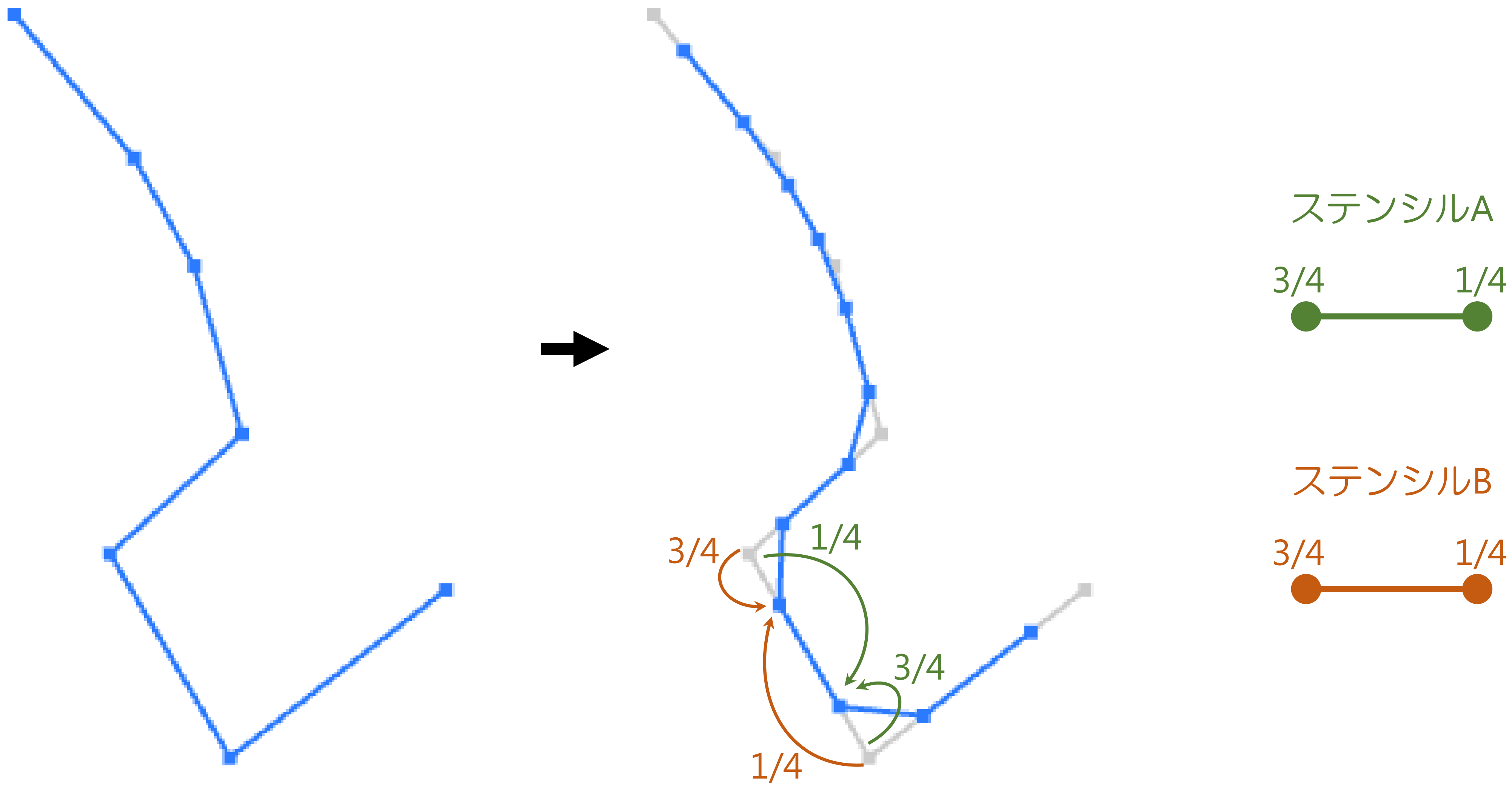


2次Bスプライン曲線の分割

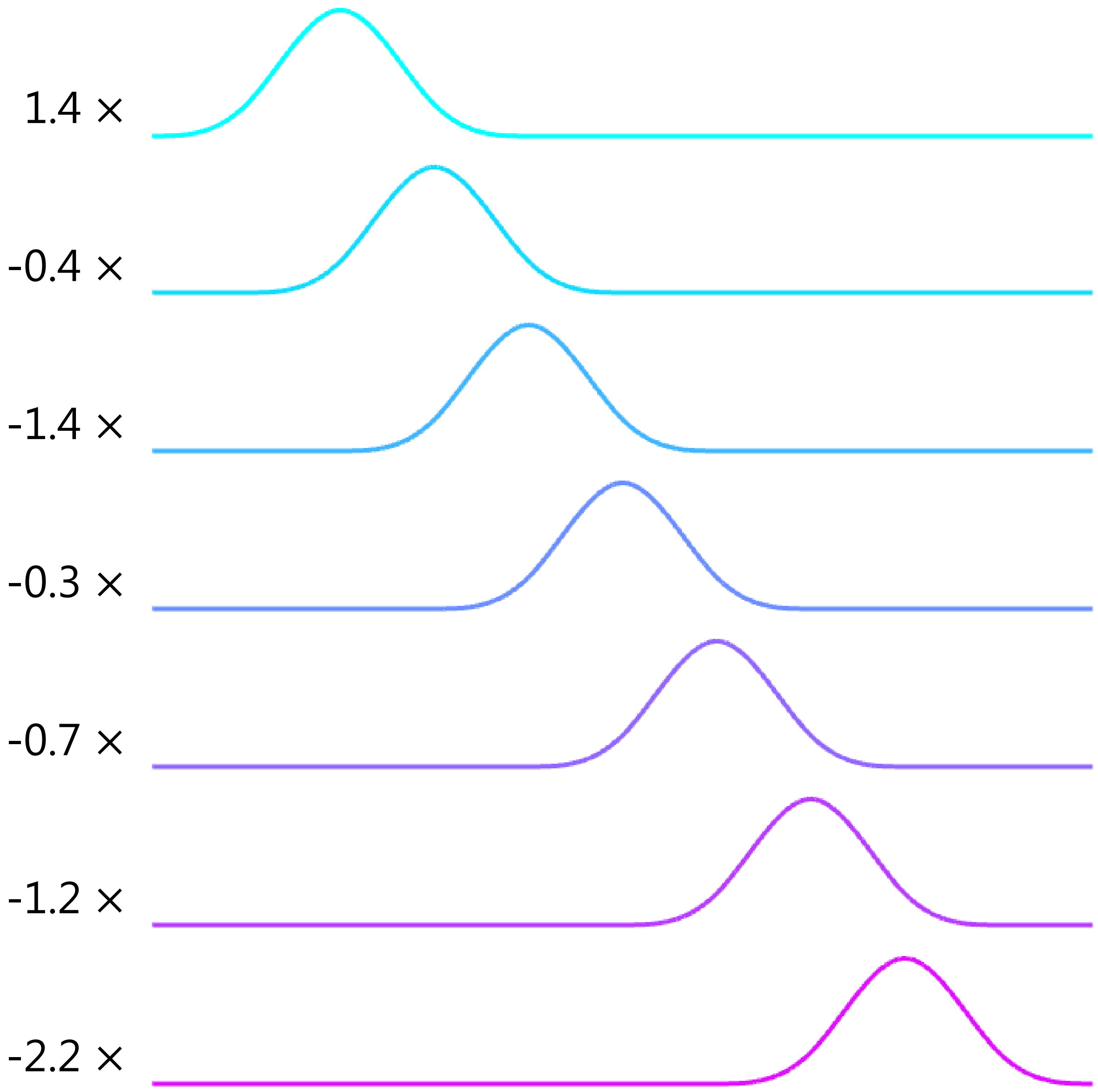


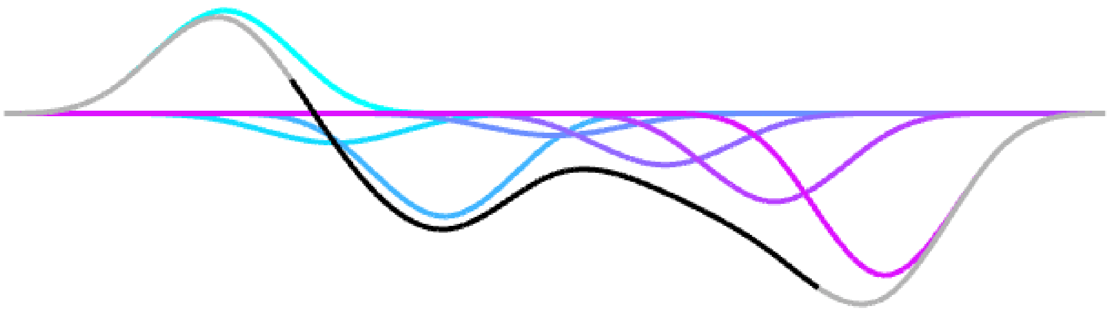


サブディビジョンによる2次曲線の生成

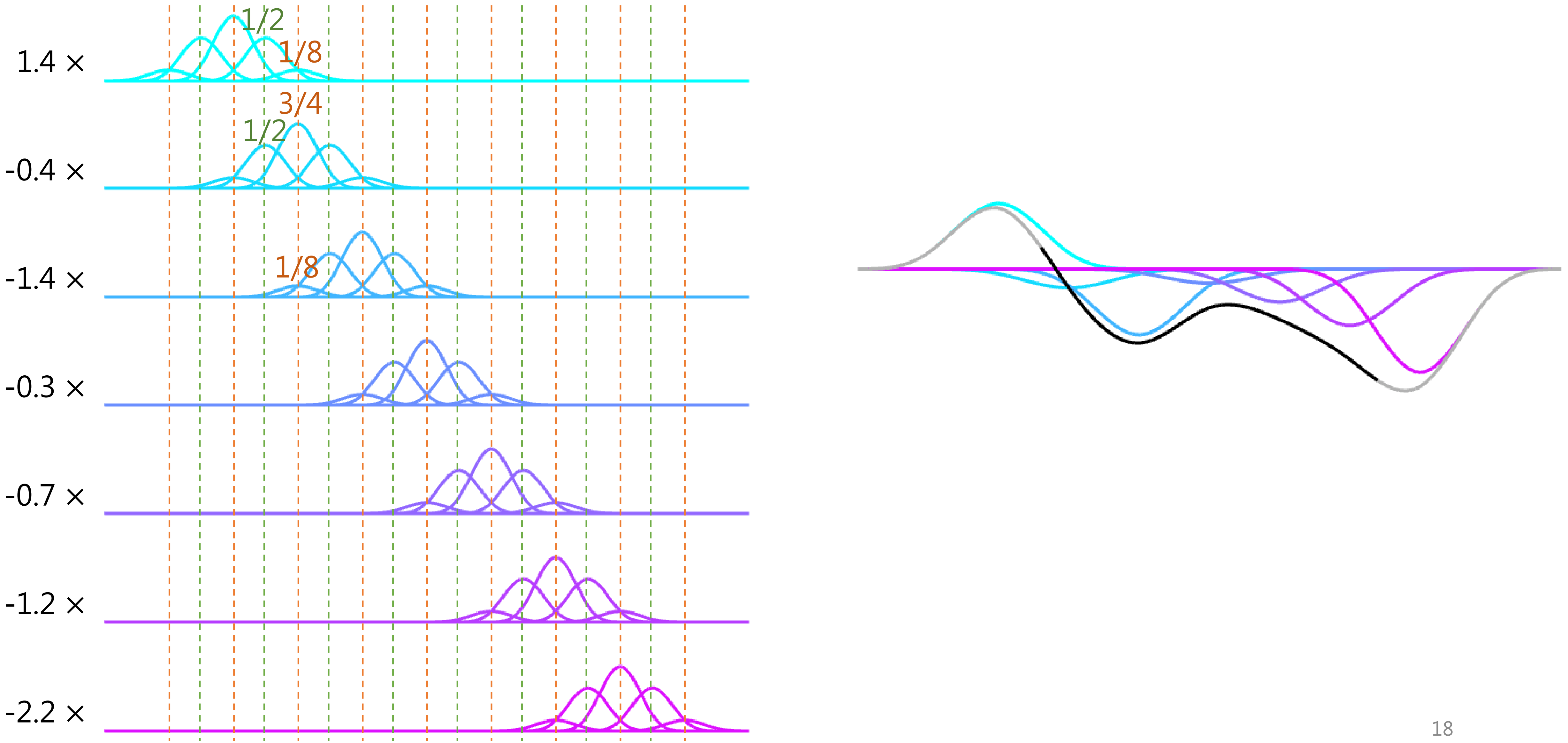


3次Bスプライン曲線の分割





3次Bスプライン曲線の分割



3次Bスプライン曲線の分割

$1/8 \times (1.4) \times$

$1/2 \times (1.4) \times$

$1/8 \times \{6(1.4) + (-0.4)\} \times$

$1/2 \times \{(1.4) + (-0.4)\} \times$

$1/8 \times \{(1.4) + 6(-0.4) + (-1.4)\} \times$

$1/2 \times \{(-0.4) + (-1.4)\} \times$

$1/8 \times \{(-0.4) + 6(-1.4) + (-0.3)\} \times$

$1/2 \times \{(-1.4) + (-0.3)\} \times$

$1/8 \times \{(-1.4) + 6(-0.3) + (-0.7)\} \times$

$1/2 \times \{(-0.3) + (-0.7)\} \times$

$1/8 \times \{(-0.3) + 6(-0.7) + (-1.2)\} \times$

$1/2 \times \{(-0.7) + (-1.2)\} \times$

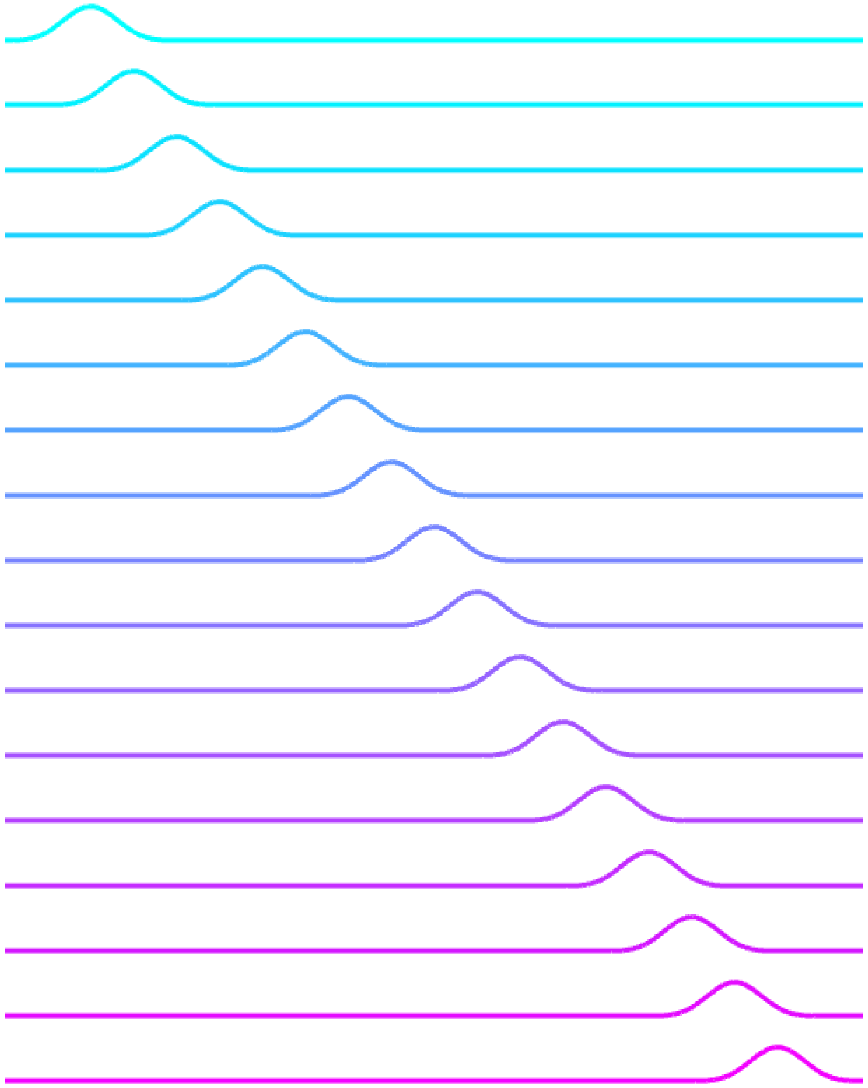
$1/8 \times \{(-0.7) + 6(-1.2) + (-2.2)\} \times$

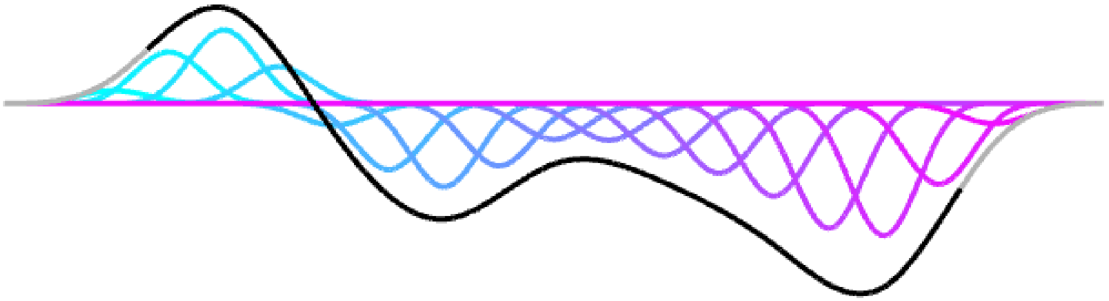
$1/2 \times \{(-1.2) + (-2.2)\} \times$

$1/8 \times \{(-1.2) + 6(-2.2)\} \times$

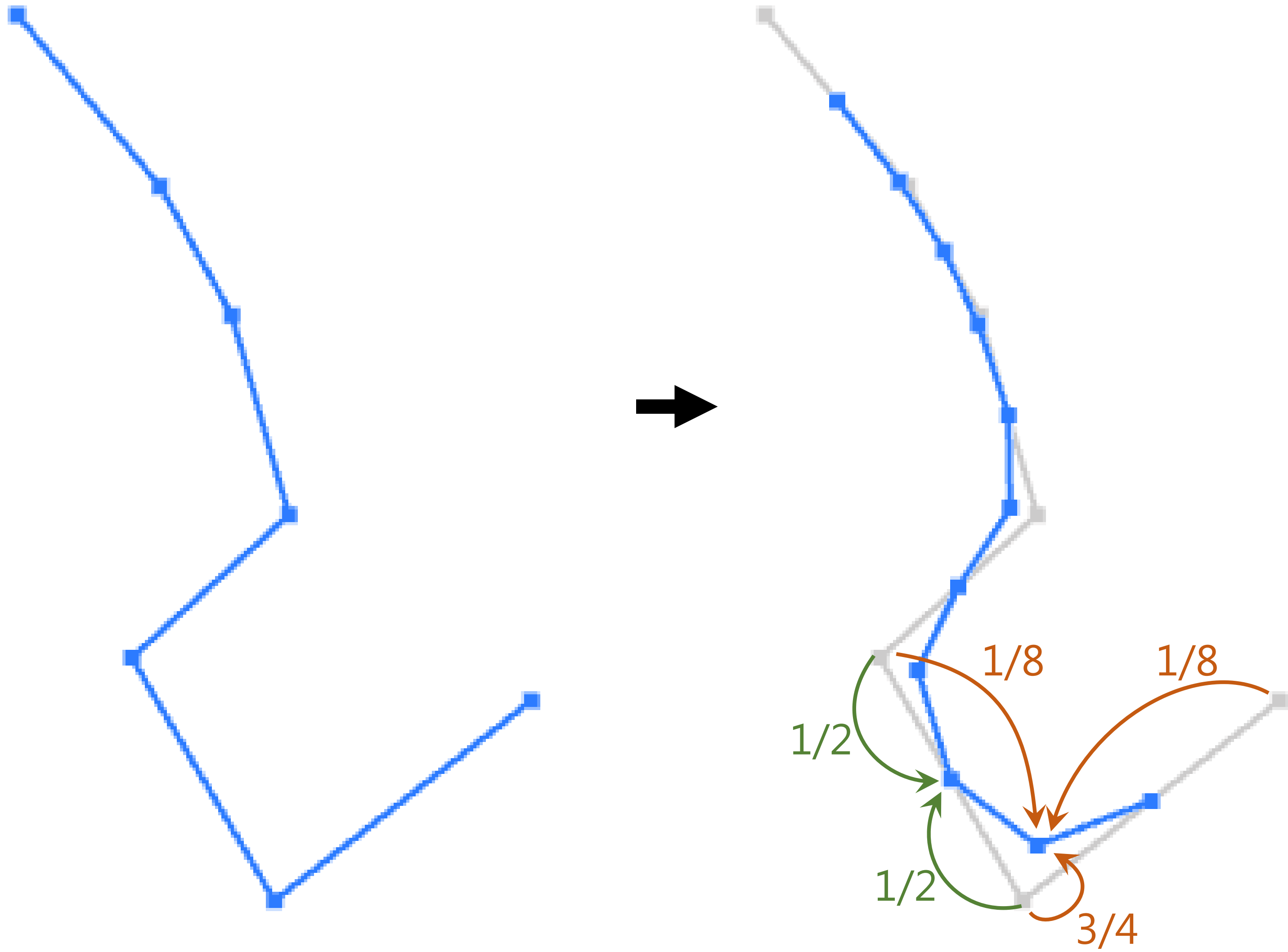
$1/2 \times (-2.2) \times$

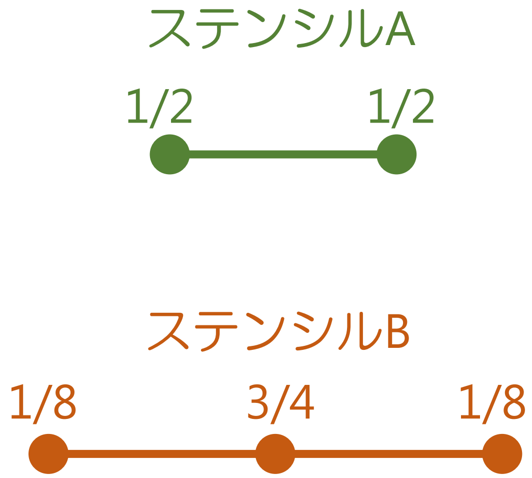
$1/8 \times (-2.2) \times$



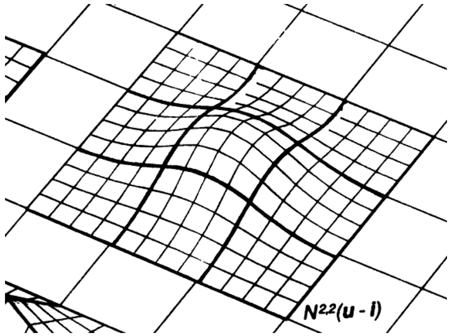


サブディビジョンによる3次曲線の生成

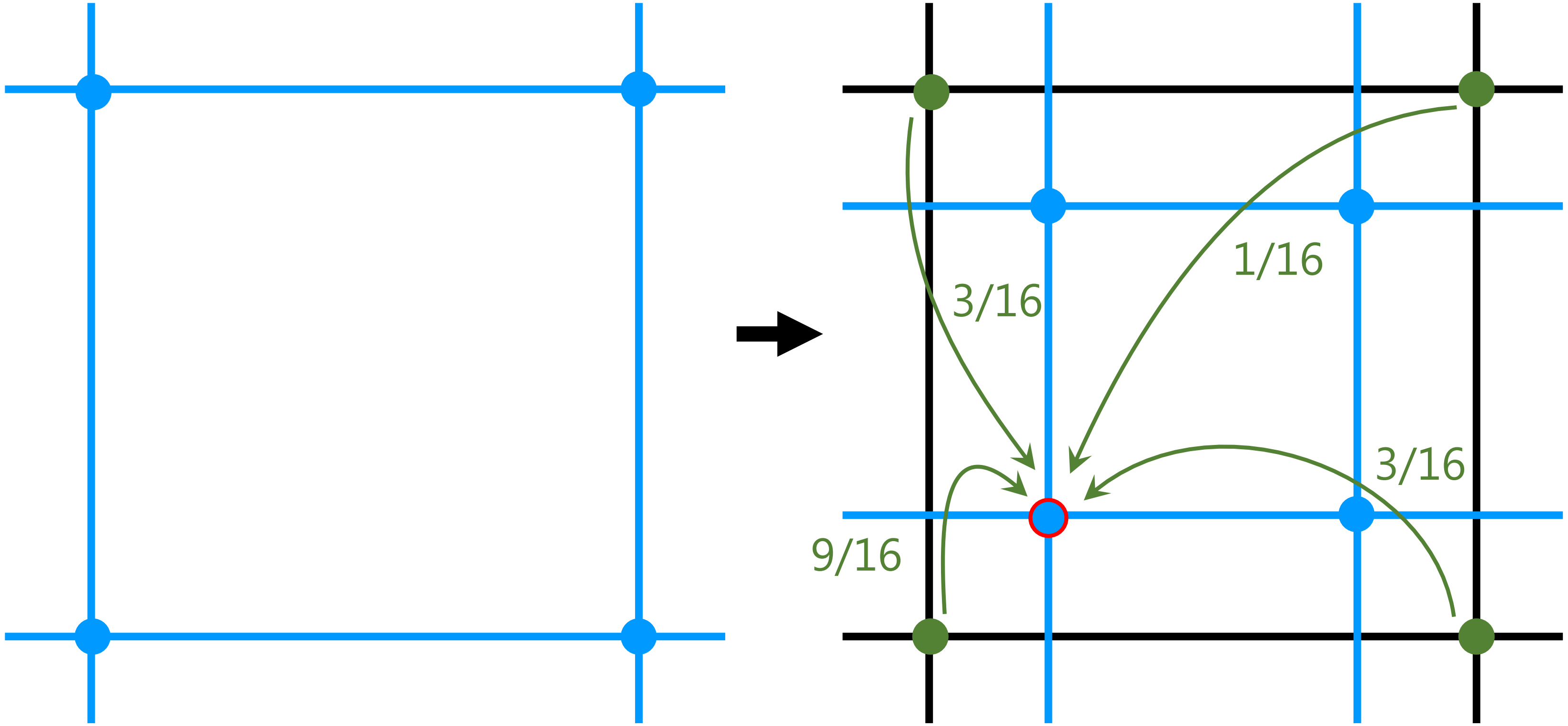


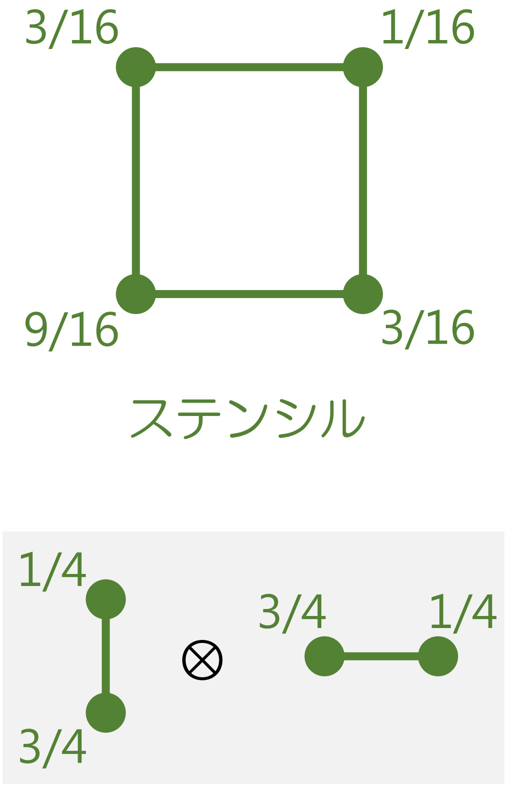


サブディビジョンによる 2次曲面の生成

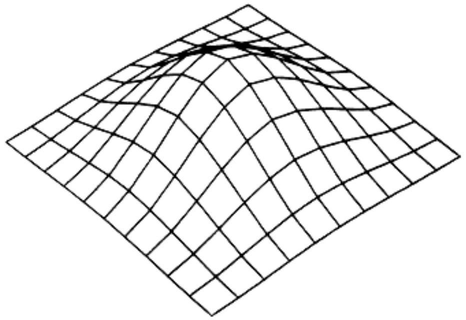


双2次基底関数
$B_{2,2}(s,t) = B_2(s)\,B_2(t)$

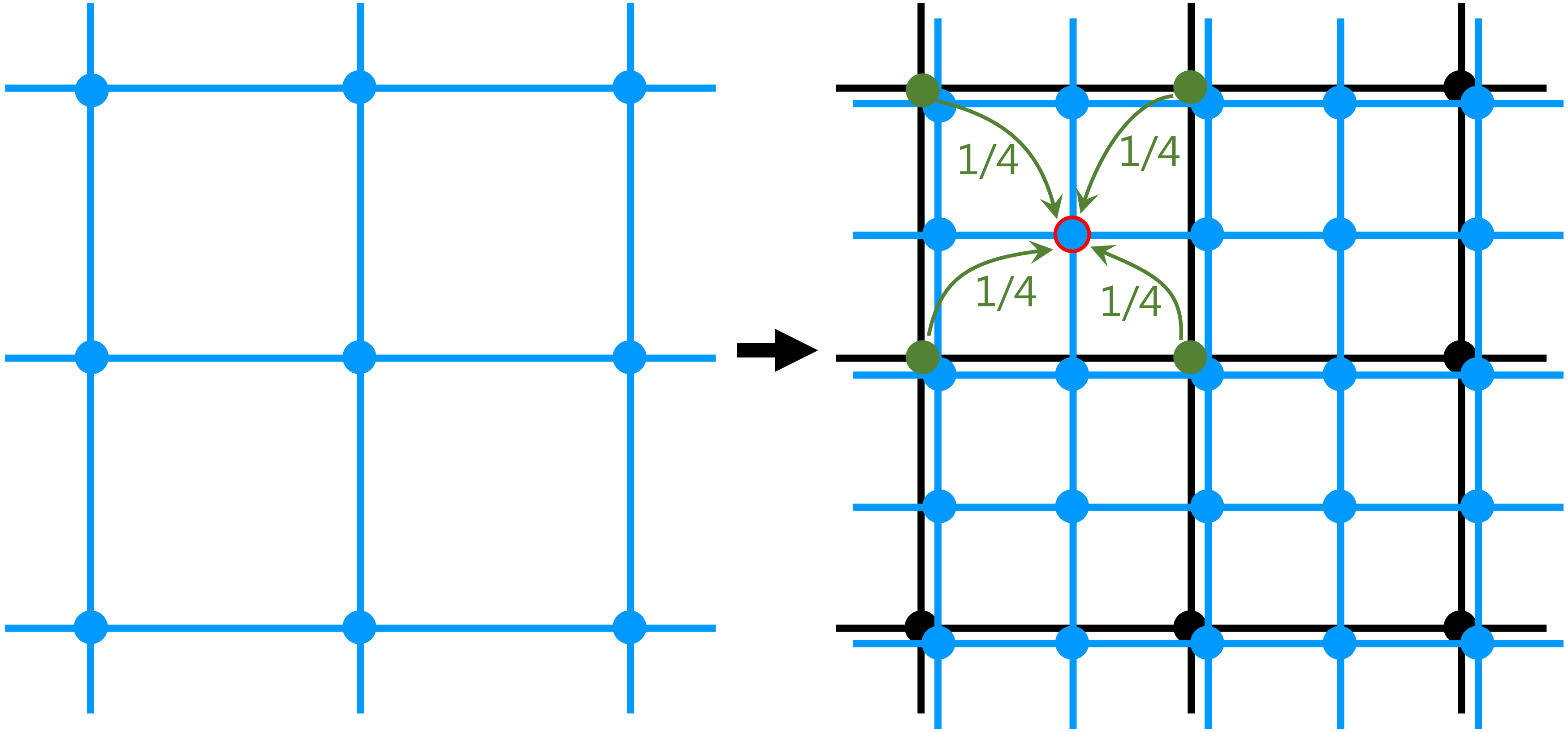




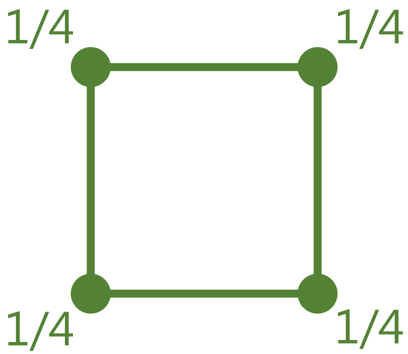
サブディビジョンによる3次曲面の生成

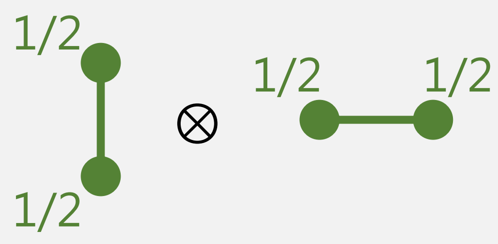


双3次基底関数
$B_{3,3}(s,t) = B_3(s)\, B_3(t)$

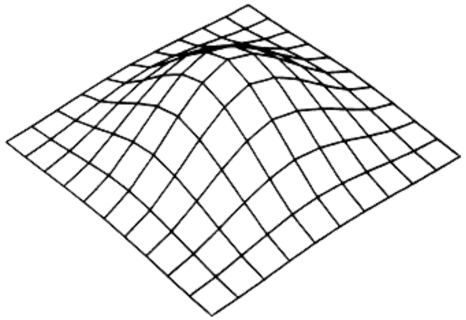


“face point” ステンシル

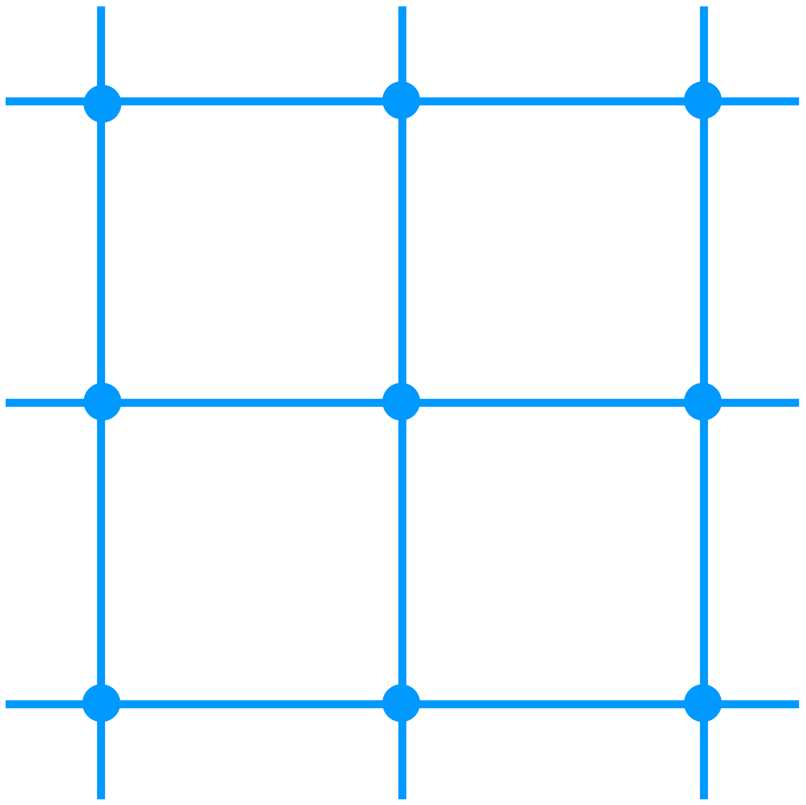


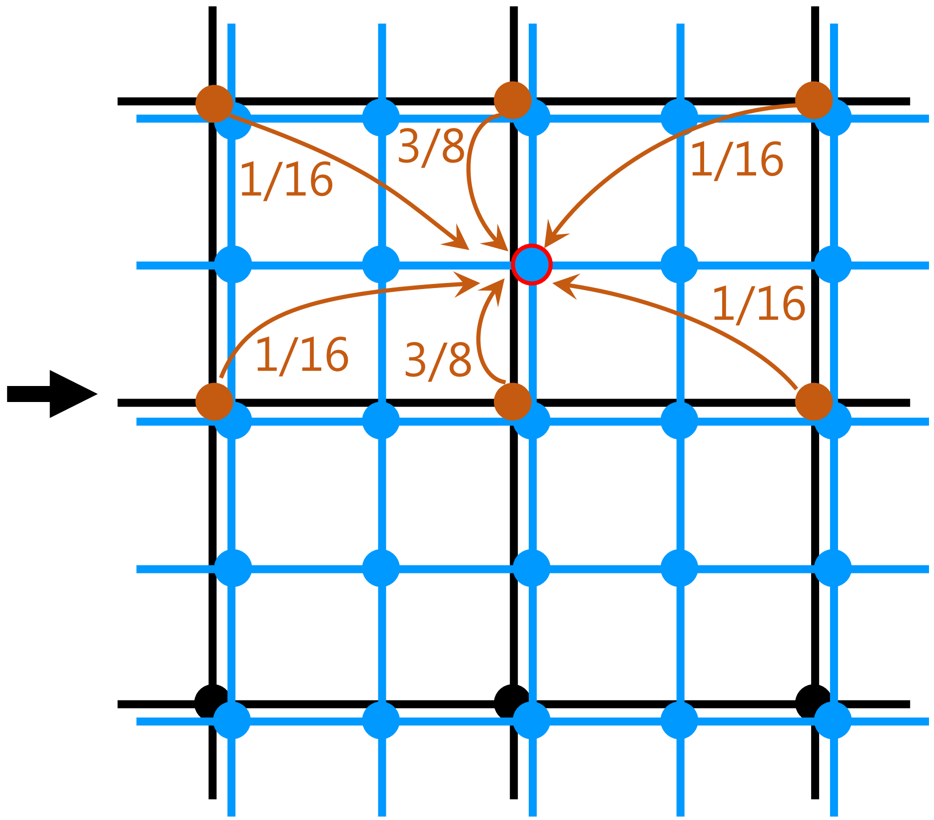


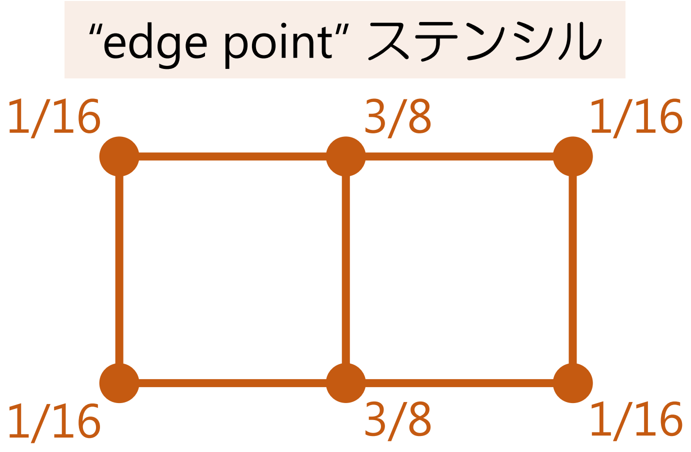
サブディビジョンによる3次曲面の生成

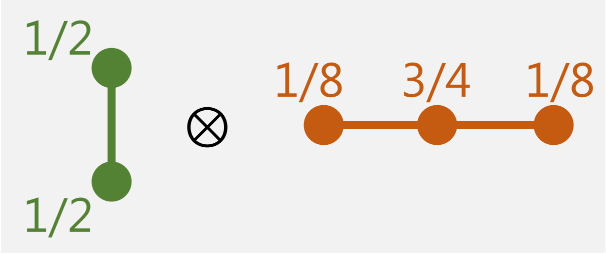


双3次基底関数
$B_{3,3}(s,t) = B_3(s)\,B_3(t)$

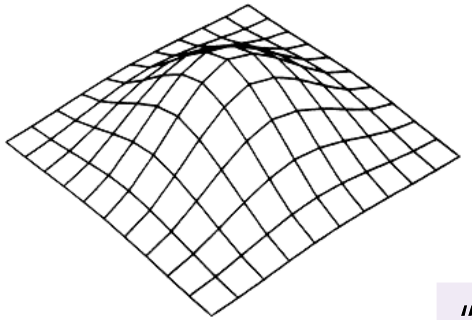




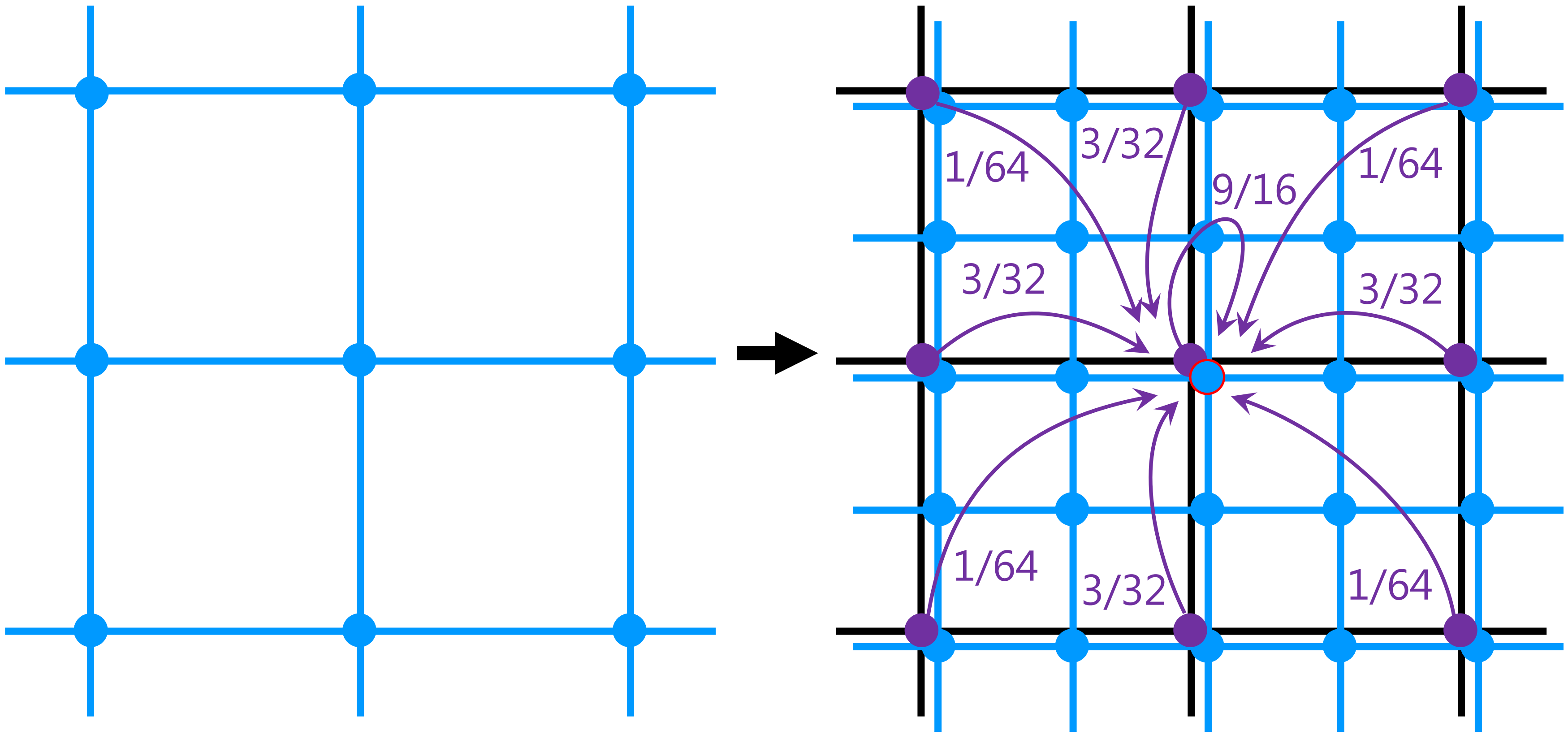




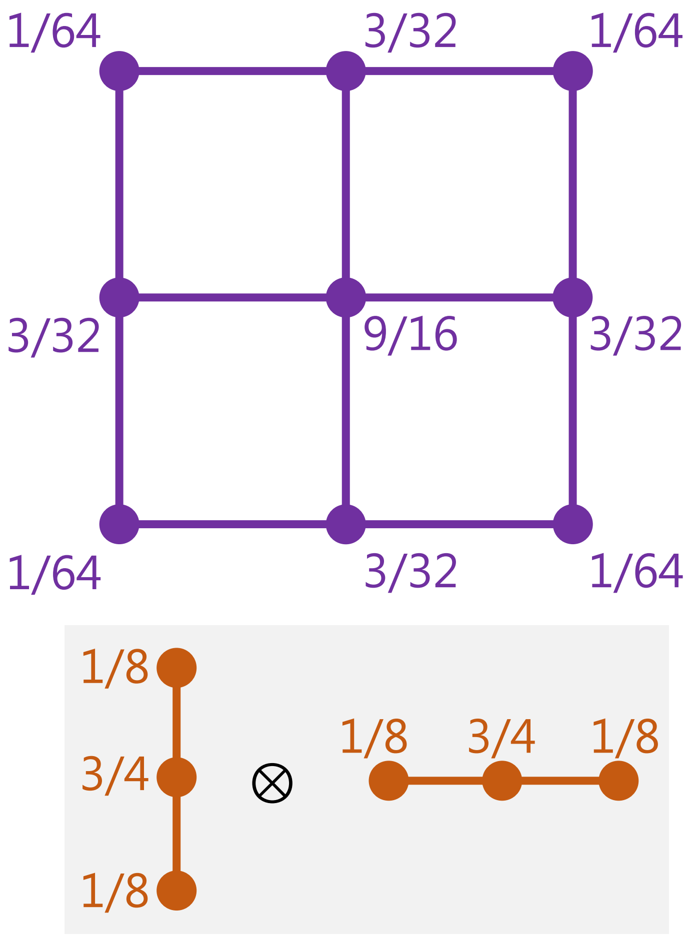
サブディビジョンによる3次曲面の生成



双3次基底関数
$B_{3,3}(s,t) = B_3(s)\,B_3(t)$



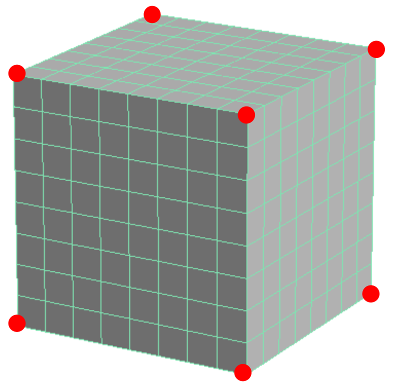
“vertex point” ステンシル

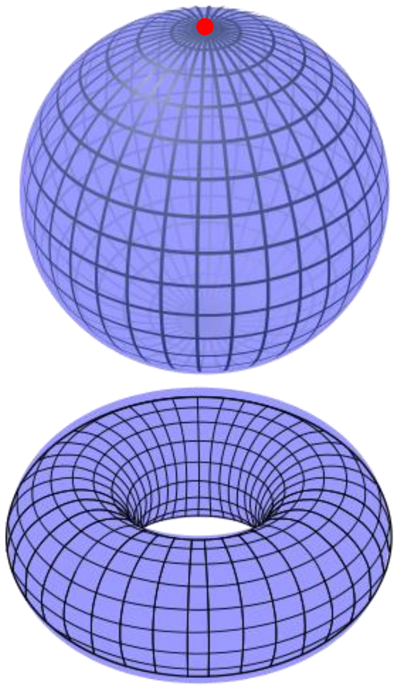


サブディビジョンの一般化

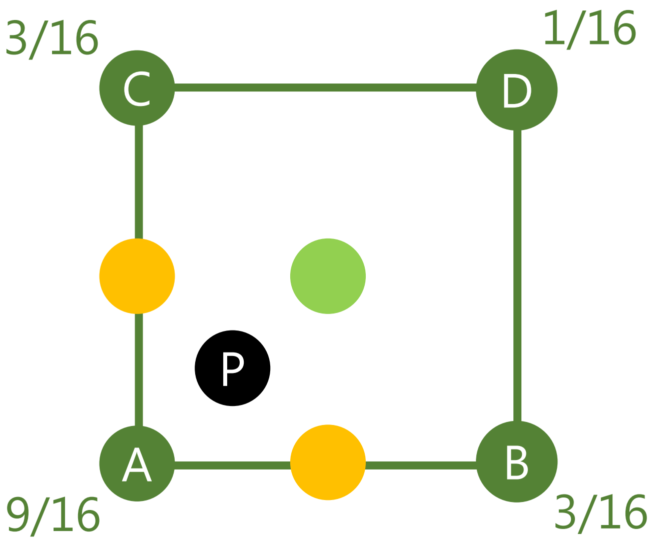
Bスプラインの本質的な限界

- 領域を「きれいな」四角形メッシュに分割できることが前提条件
  - 「きれいな」頂点：隣接する面の数 (valence) が4つ
    - valence が4でない頂点：特異点
  - 特別な場合を除き、原理的に不可能
    - 特別な場合：ドーナツ (トーラス)
- 特異点の周囲にBスプライン曲面パッチを配置し、滑らかに繋がるように微調整するのは大変 ☹
- サブディビジョン法の利点：特異点を扱える ☺
  - Bスプラインのステンシルを、幾何的な解釈から一般化





2次曲面ステンシルの一般化 (Doo-Sabin法)

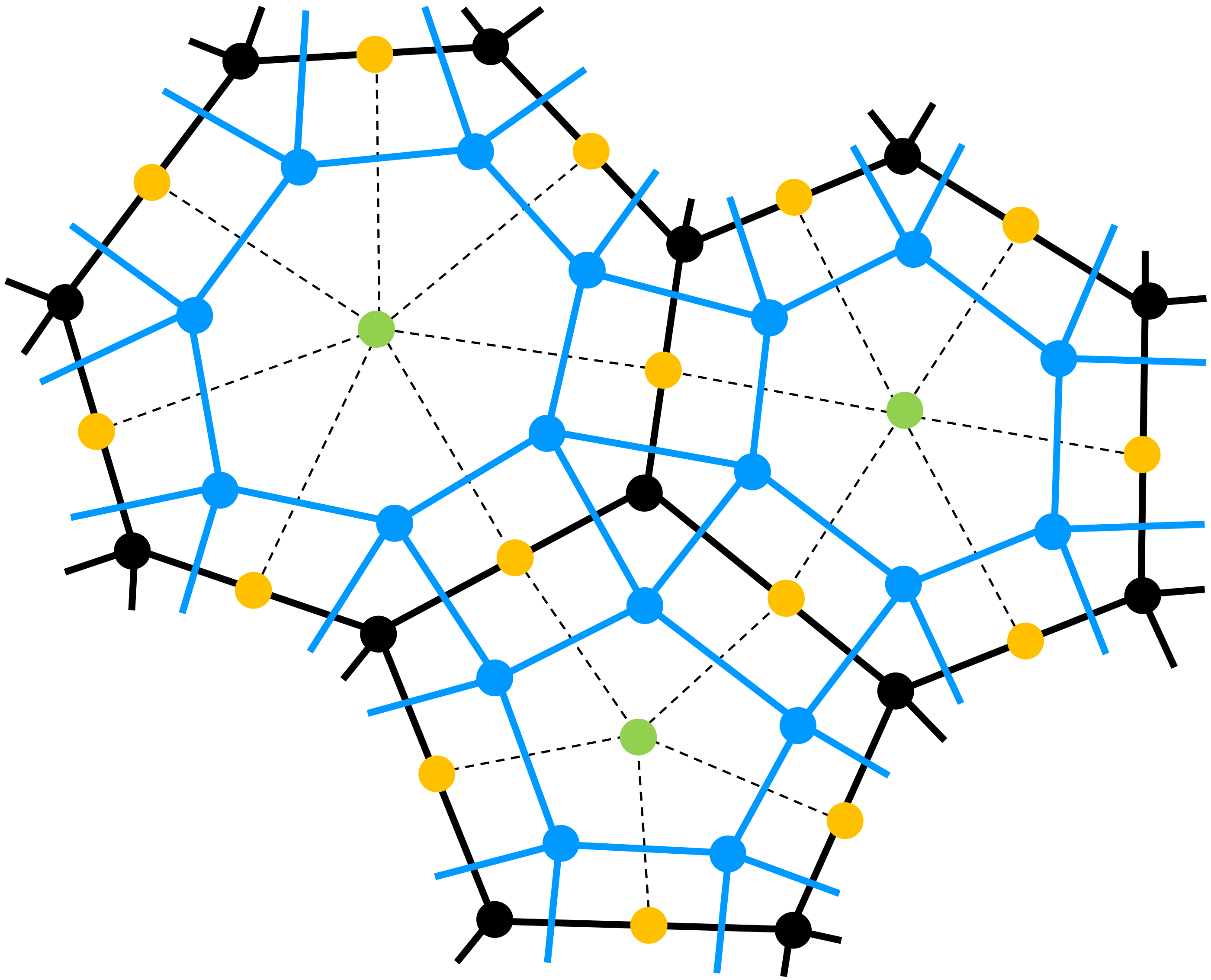


$$P = \frac{1}{16}(9\,A + 3\,B + 3\,C + D)$$

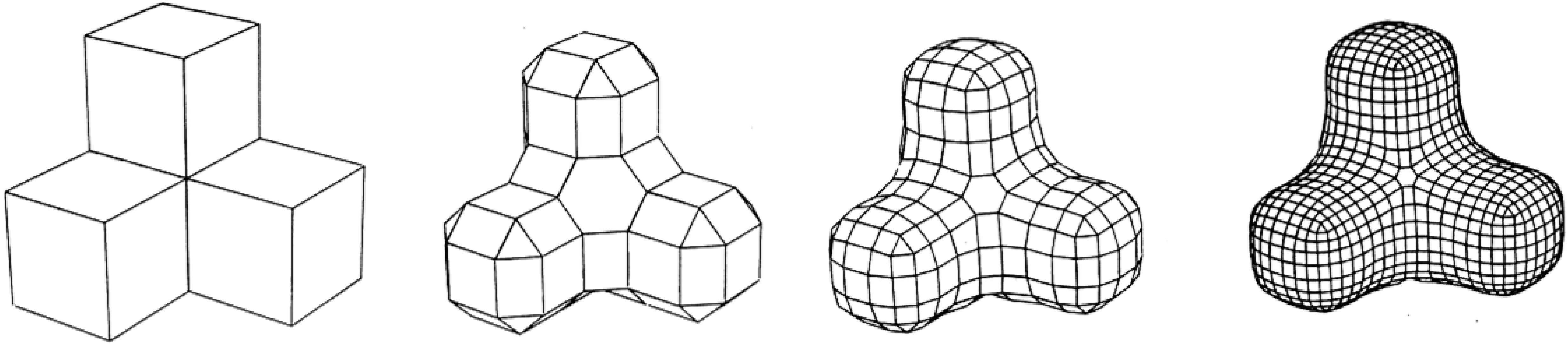
$$= \frac{\overbrace{\frac{A+B+C+D}{4}}^{\text{重心}} + \overbrace{\frac{A+B}{2}}^{\text{中点}} + \overbrace{\frac{A+C}{2}}^{\text{中点}} + A}{4}$$

➔ 一般のポリゴンメッシュに適用できる

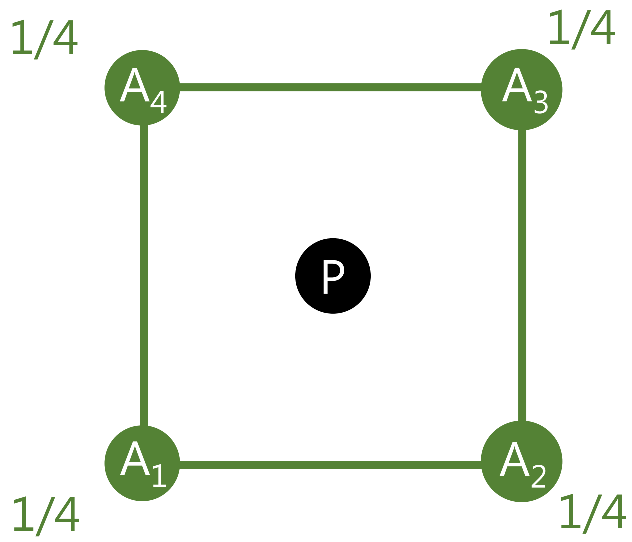
Doo-Sabin法の適用例



Doo-Sabin法の適用例



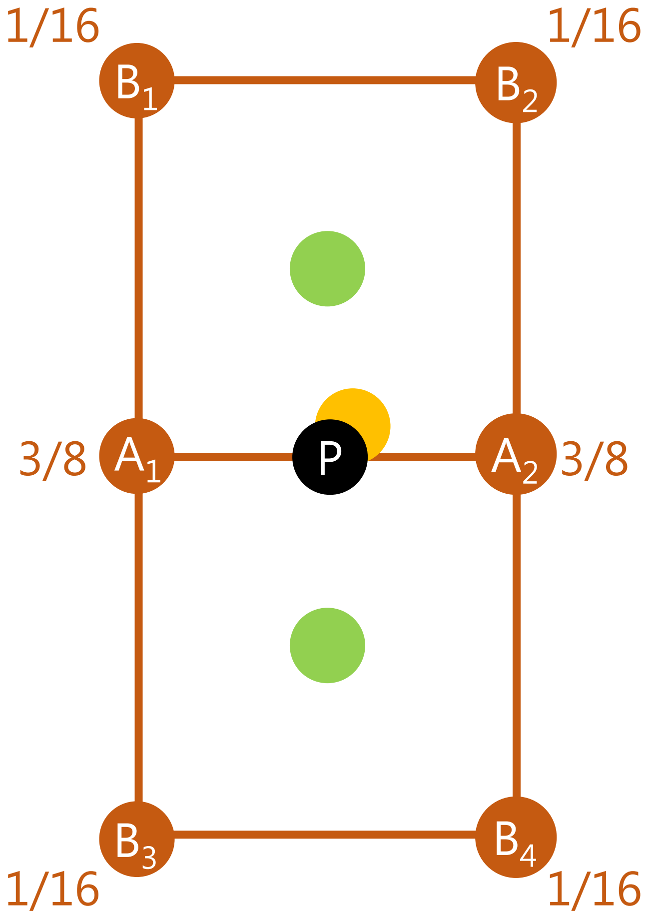
3次曲面ステンシルの一般化 (Catmull-Clark法)



$$P = \frac{A_1 + A_2 + A_3 + A_4}{4}$$

➔ 一般のポリゴンメッシュに適用できる

3次曲面ステンシルの一般化 (Catmull-Clark法)

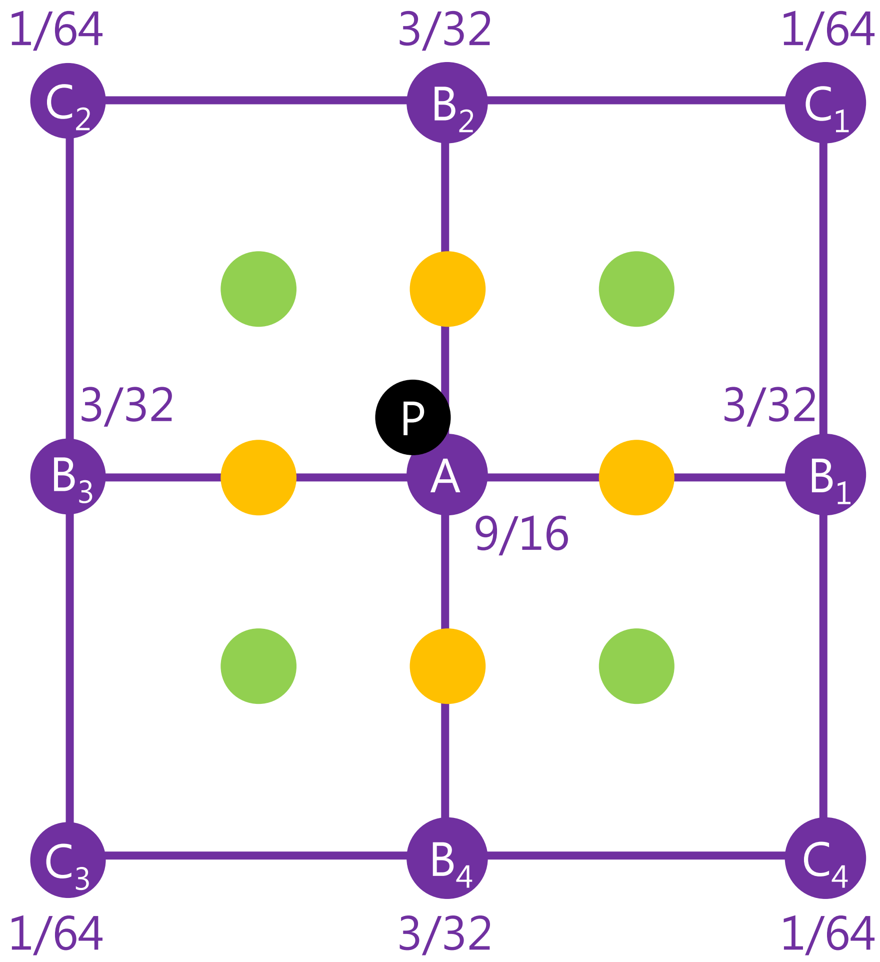


$$P = \frac{3}{8}(A_1 + A_2) + \frac{1}{16}(B_1 + B_2 + B_3 + B_4)$$

$$= \frac{\frac{\overbrace{\frac{A_1 + A_2 + B_1 + B_2}{4}}^{\text{重心}} + \overbrace{\frac{A_1 + A_2 + B_3 + B_4}{4}}^{\text{重心}}}{2} + \overbrace{\frac{A_1 + A_2}{2}}^{\text{中点}}}{2}$$

➔ 一般のポリゴンメッシュに適用できる

3次曲面ステンシルの一般化 (Catmull-Clark法)



$$P = \frac{9}{16}A + \frac{3}{32}(B_1 + B_2 + B_3 + B_4) + \frac{1}{64}(C_1 + C_2 + C_3 + C_4)$$

$$= \frac{1}{4}\left\{\underbrace{\frac{\overset{\text{重心}}{\frac{A+B_1+C_1+B_2}{4}} + \overset{\text{重心}}{\frac{A+B_2+C_2+B_3}{4}} + \overset{\text{重心}}{\frac{A+B_3+C_3+B_4}{4}} + \overset{\text{重心}}{\frac{A+B_4+C_4+B_1}{4}}}{4}}_{Q}\right\}$$

$$+ \frac{2}{4}\left\{\underbrace{\frac{\overset{\text{中点}}{\frac{A+B_1}{2}} + \overset{\text{中点}}{\frac{A+B_2}{2}} + \overset{\text{中点}}{\frac{A+B_3}{2}} + \overset{\text{中点}}{\frac{A+B_4}{2}}}{4}}_{R}\right\}$$

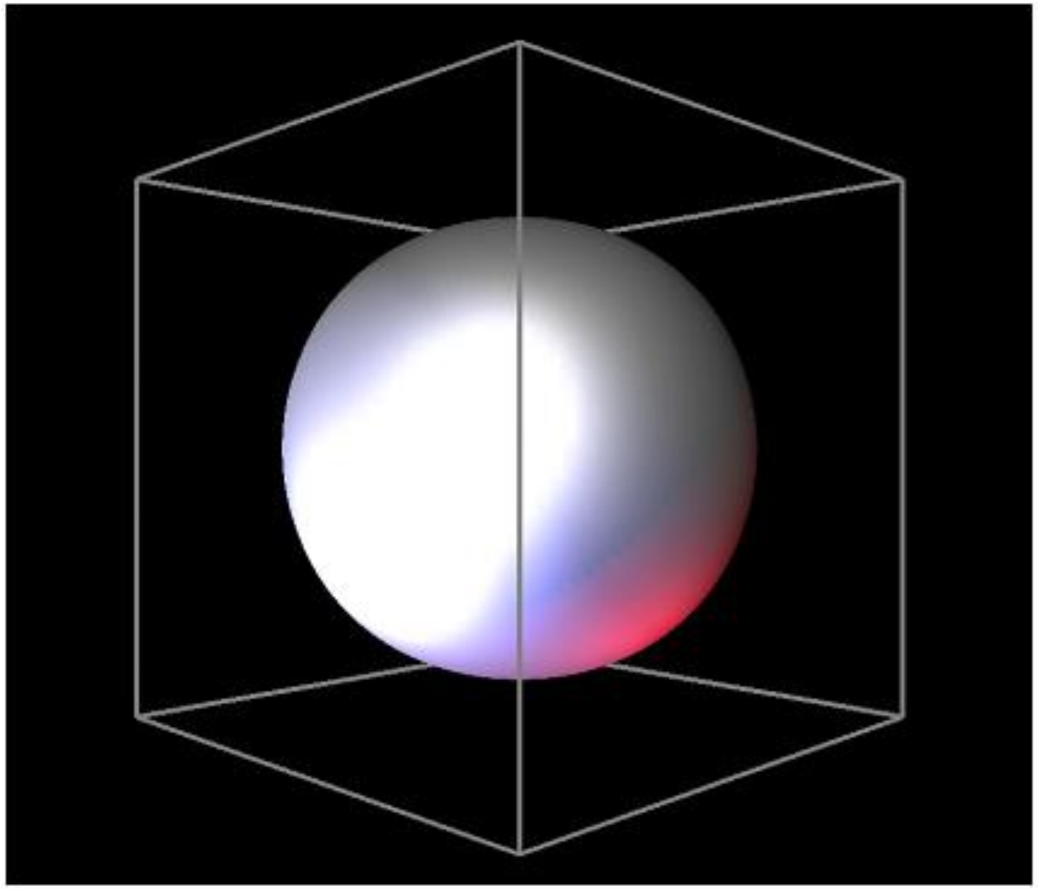
$$+ \frac{1}{4}A$$

➔ 一般のポリゴンメッシュに適用できる

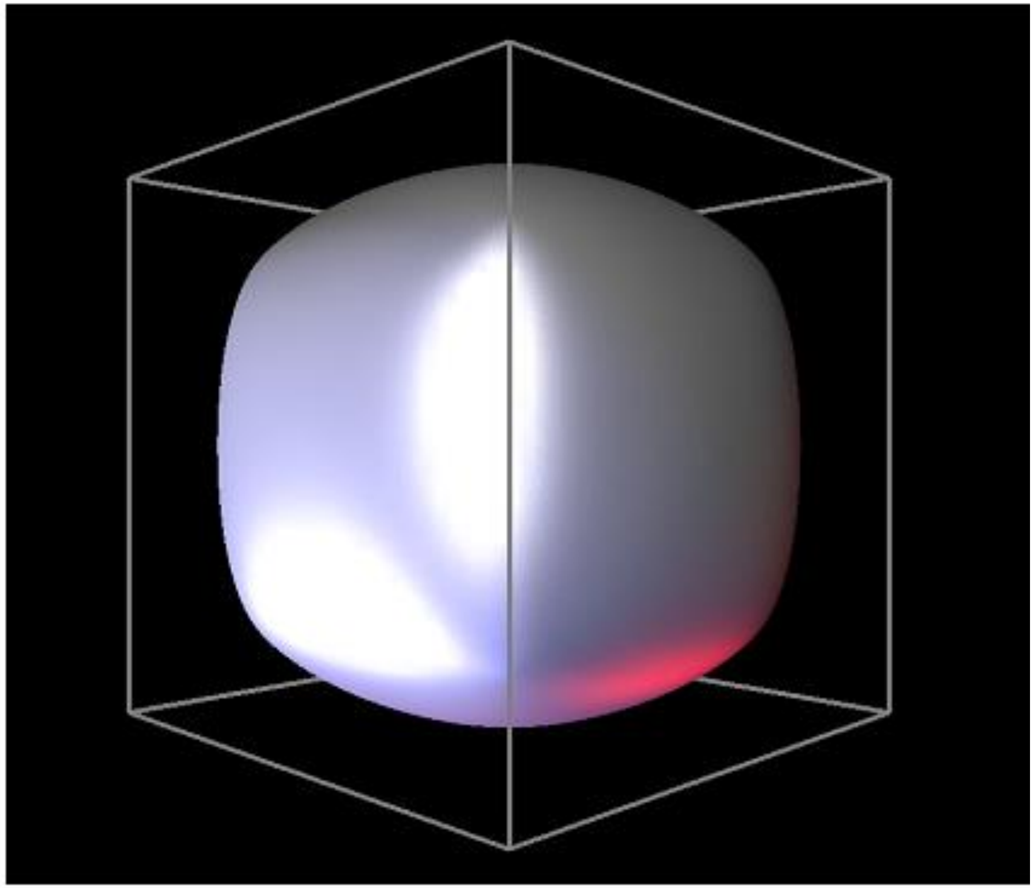
A の valence が n のとき、

$$P = \frac{1}{n}Q + \frac{2}{n}R + \frac{n-3}{n}A$$

Doo-Sabin法とCatmull-Clark法の比較



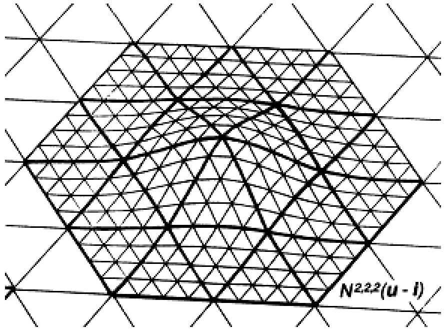
Catmull-Clark

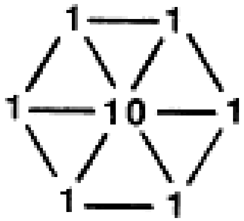


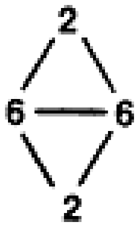
Doo-Sabin

三角形メッシュのサブディビジョン (Loop法)

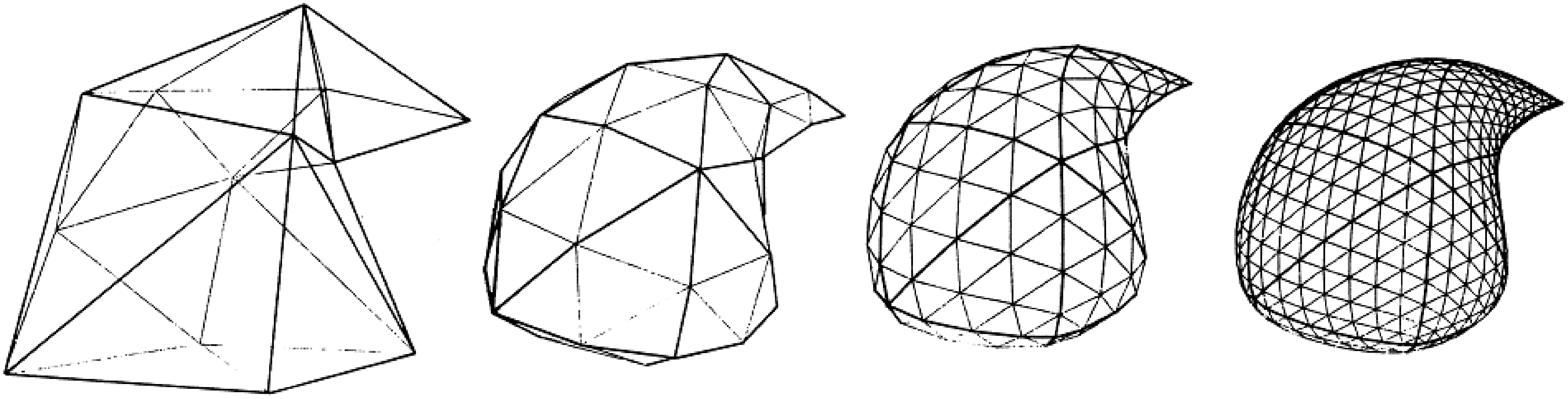
- 三角形格子上のBスプラインに基づいて設計



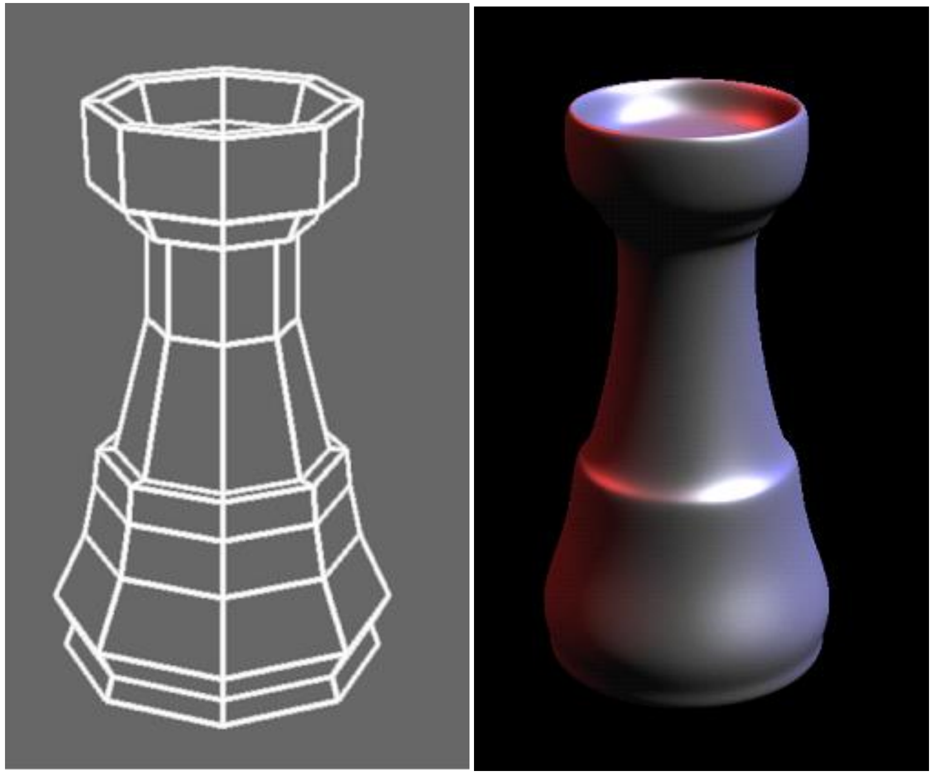




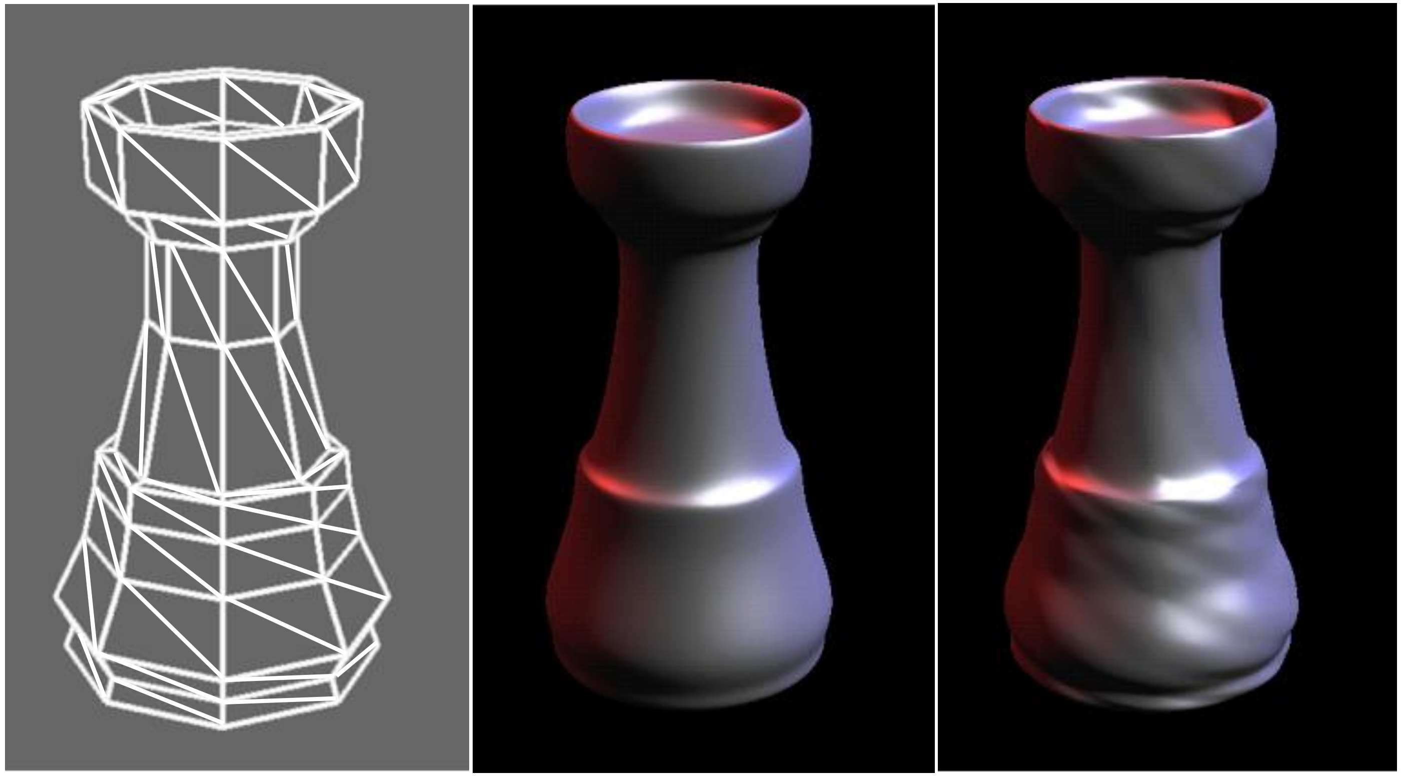
- 特異点以外ではC^2連続な3次曲面



Catmull-Clark法とLoop法の比較



Catmull-Clark

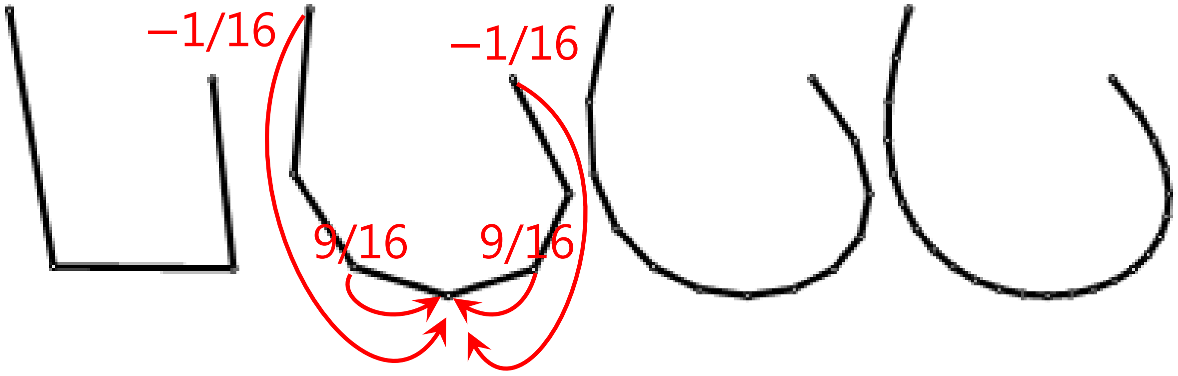


Loop　Catmull-Clark

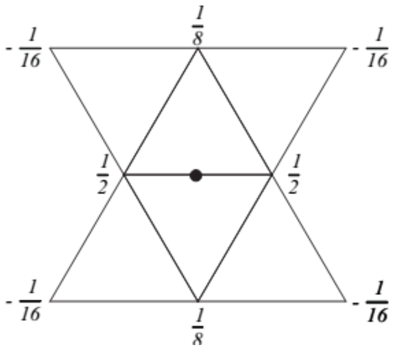
- CG業界ではCatmull-Clark法が圧倒的にポピュラー
  - 四角形メッシュだと二つの主曲率方向を自然に表せる

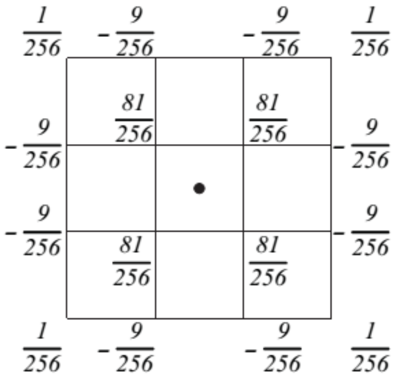
その他のサブディビジョン手法

- four-point法
  - 制御点を通る (interpolating)
    - ←→ approximating
  - 多項式として表現できない (?)
  - C^1連続
  - 曲面バージョン：Butterfly法



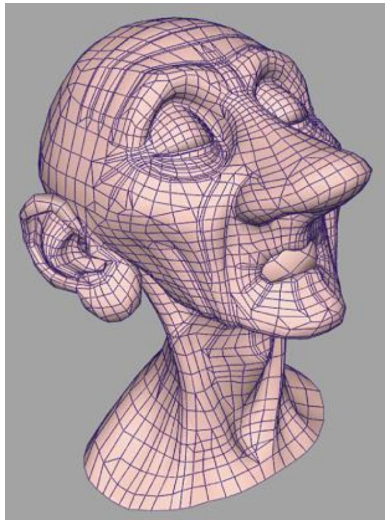
- この他にも亜種が多数存在
  - Kobbelt法
  - $\sqrt{3}$法
  - etc...





Geri’s Game (Pixar, 1997)

- サブディビジョンを使った最初の映画
  - それ以前 (Toy Story) はBスプラインで多大な労力をかけていた

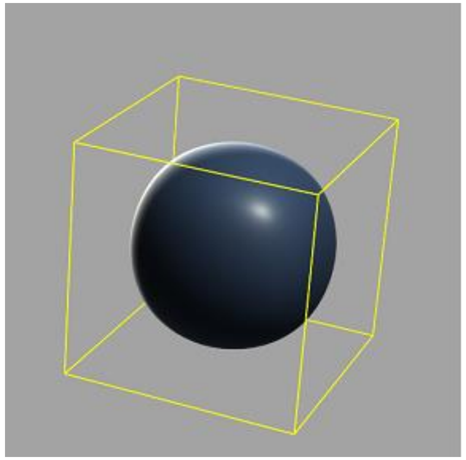


https://www.youtube.com/watch?v=9IYRC7g2ICg

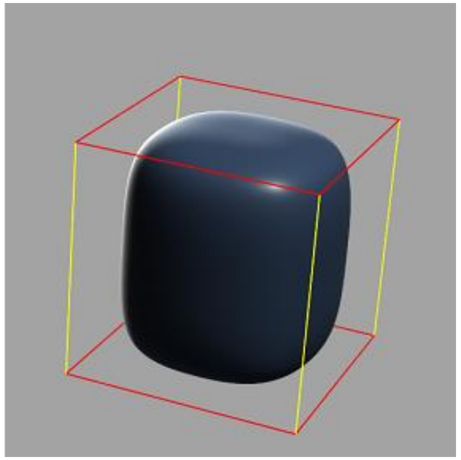
Subdivision surfaces in character animation [DeRose,Kass,Truong,SIGGRAPH98]

滑らかさの制御

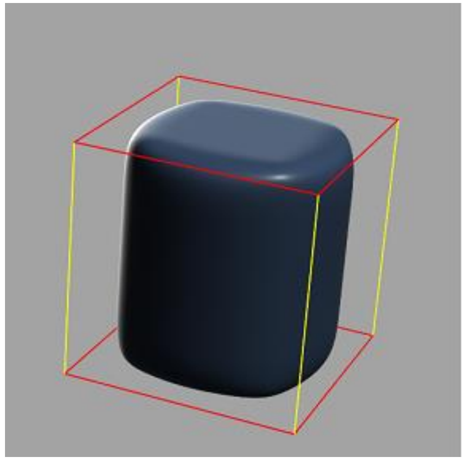
- サブディビジョンのルールを少し変えると、鋭いエッジを表現できる



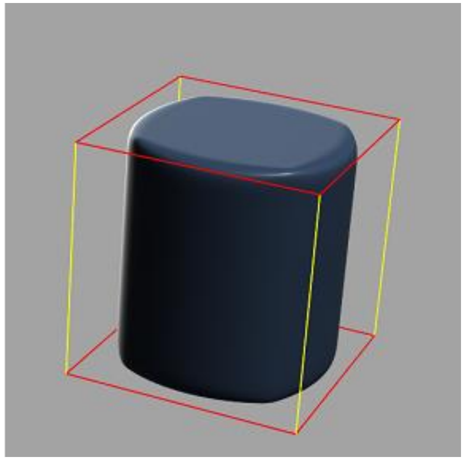
sharpness=0



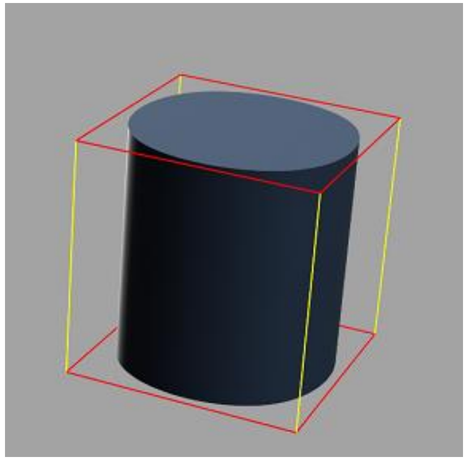
sharpness=1



sharpness=2



sharpness=3



sharpness=∞

Subdivision surfaces in character animation [DeRose,Kass,Truong,SIGGRAPH98]

滑らかさの制御

- サブディビジョンのルールを少し変えると、鋭いエッジを表現できる

Subdivision surfaces in character animation [DeRose,Kass,Truong,SIGGRAPH98]

サブディビジョンの解説資料

- Smooth Subdivision Surfaces Based on Triangles [Loop, MSc. Thesis 87]
  - Doo-Sabin法やCatmull-Clark法を含め、考え方を丁寧に図解
  - 間違いがあるので注意：
    http://www.cs.berkeley.edu/~sequin/CS284/TEXT/LoopErrata.txt
- Subdivision for Modeling and Animation [SIG00 Course]
  - サブディビジョンの概説としては最も有名。でも微妙に難解
  - http://www.cs.nyu.edu/~dzorin/sig00course/
- OpenSubdiv from research to industry adoption [SIG13 Course]
  - 最新の話題など
  - http://dx.doi.org/10.1145/2504435.2504451

ハーフエッジデータ構造

頂点リストと面リストによるメッシュ表現

OFF file format

```
OFF
8 6 0
-0.5 -0.5 0.5
0.5 -0.5 0.5
-0.5 0.5 0.5
0.5 0.5 0.5
-0.5 0.5 -0.5
0.5 0.5 -0.5
-0.5 -0.5 -0.5
0.5 -0.5 -0.5
4 0 1 3 2
4 2 3 5 4
4 4 5 7 6
4 6 7 1 0
4 1 7 5 3
4 6 0 2 4
```

ジオメトリ情報

トポロジ情報

← 頂点数、面数

← 0番目の頂点の xyz 座標

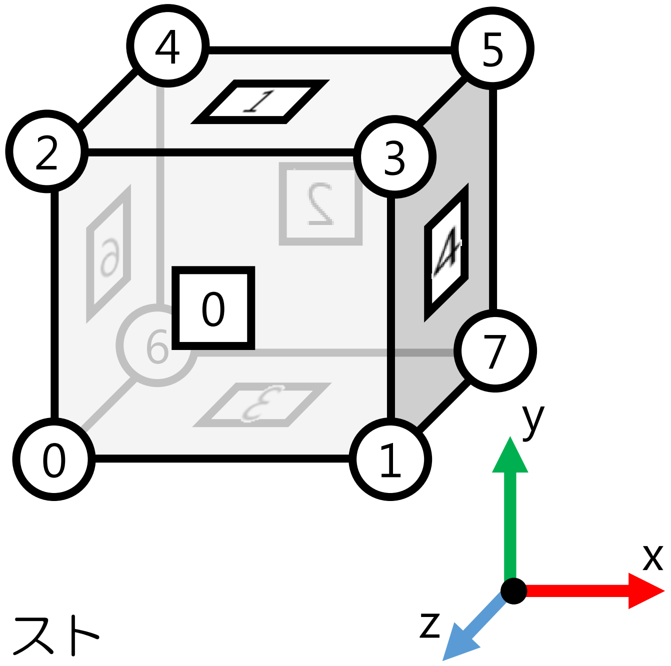
⋮

← 7番目の頂点の xyz 座標

← 0番目の面の頂点数と頂点インデックスリスト

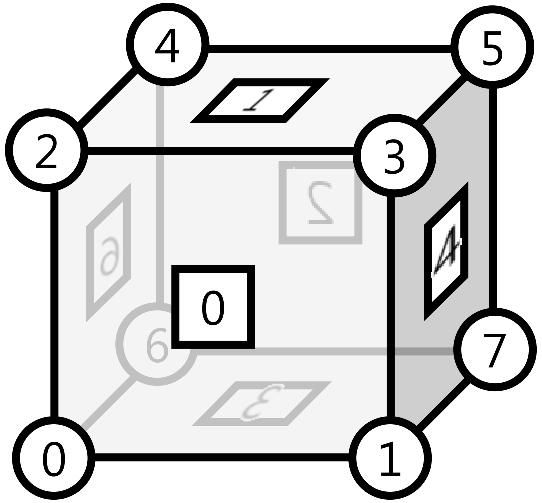
⋮

← 6番目の面の頂点数と頂点インデックスリスト



頂点リストと面リストによるメッシュ表現

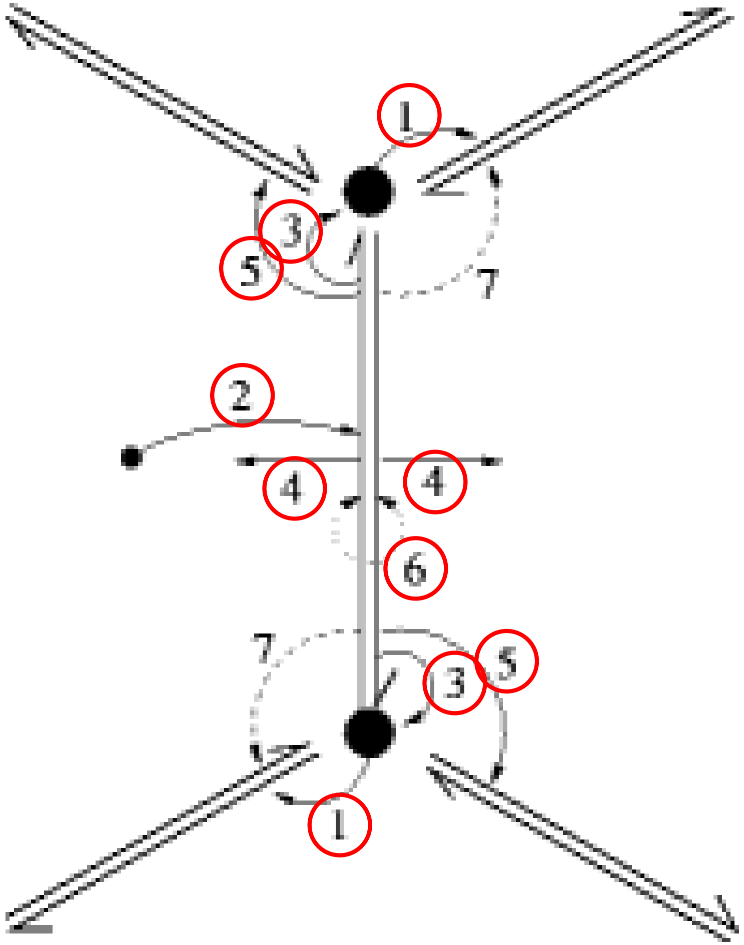
- メッシュ処理 (サブディビジョン等) で使う情報
  - ある頂点に隣接する頂点の集合
  - ある面に隣接する面の集合
  - あるエッジの両端の頂点
  - あるエッジの両側の面
  - etc…



- 「配列の配列」で保持しても良いが、メモリ消費が大きい ☹

ハーフエッジデータ構造

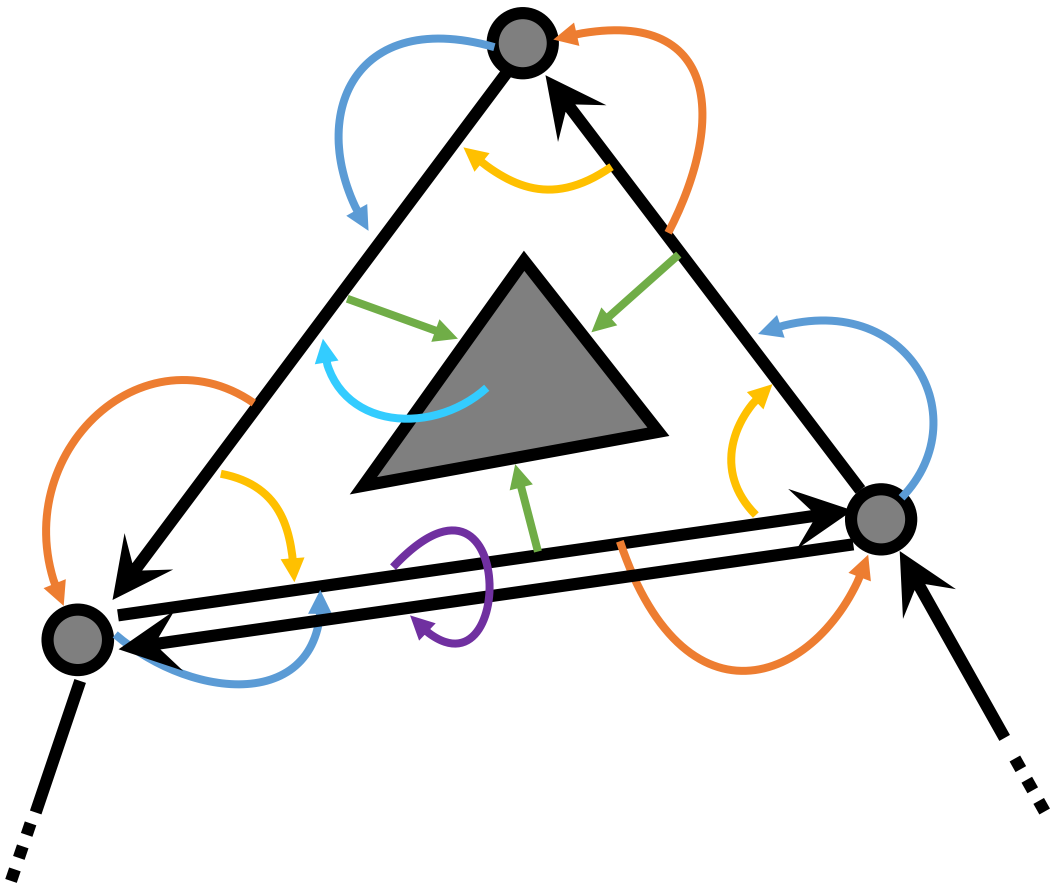
- リンク情報を保持：
  - (1) 頂点→その頂点から出るハーフエッジの一つ
  - (2) 面→その面を構成するハーフエッジの一つ
  - (3) ハーフエッジ→行き先の頂点
  - (4) ハーフエッジ→それが構成する面
  - (5) ハーフエッジ→次のハーフエッジ
  - (6) ハーフエッジ→反対側のハーフエッジ
- ある面の周りの要素をループ：
  - (2) → (5) → (5) → …
- ある頂点の周りの要素をループ：
  - (1) → (6) → (5) → (6) → (5) → …



http://www.openmesh.org/

面を追加する際の処理

- ハーフエッジを生成
- 頂点→ハーフエッジをリンク (1)
- ハーフエッジ→頂点をリンク (3)
- 次のハーフエッジをリンク (5)
- ハーフエッジ→面をリンク (4)
- 面→ハーフエッジをリンク (2)
- ハーフエッジ→反対向きのハーフエッジを探してリンク (6)



論文

- Recursively generated B-spline surfaces on arbitrary topological meshes [Catmull,Clark,CAD78]
- A 4-point interpolatory subdivision scheme for curve design [Dyn,Levin,CAGD87]
- A butterfly subdivision scheme for surface interpolation with tension control [Dyn,Levine,Gregory,TOG90]
- Sqrt(3)-subdivision [Kobbelt,SIGGRAPH00]
- Exact evaluation of Catmull-Clark subdivision surfaces at arbitrary parameter values [Stam,SIGGRAPH98]
- Interactive multiresolution mesh editing [Zorin,Schroder,Sweldens,SIGGRAPH97]
- Interpolating subdivision for meshes with arbitrary topology [Zorin,Schroder,Sweldens,SIGGRAPH96]