Kõverad ja pinnad: Osa II

Konstantin Tretjakov (kt@ut.ee)

14. november 2005

Eelmine kord

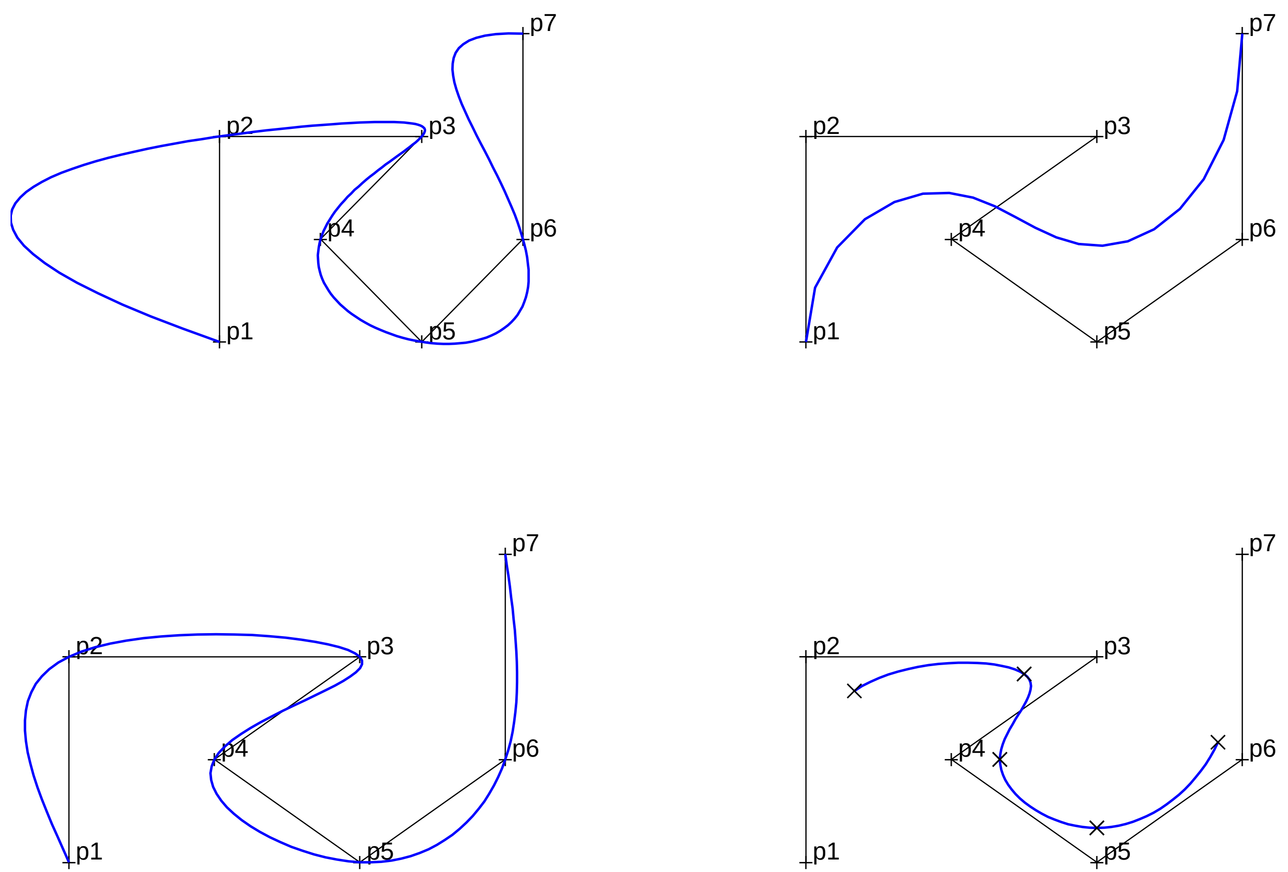
- Kõverjooned ja pinnad lubavad realistlikumalt esitada objektide kuju.
- Kõverate abil saab määrata ka objektide liikumist.
- Joonte ja pindade esitamiseks kasutame *parameetrilist* kuju: $\mathbf{p}(t) = \mathbf{f}(t)$
- Vaatleme ainult polünomiaalseid või tükiti-polünomiaalseid kõveraid:

$$\mathbf{p}(t) = \mathbf{c}_0 + \mathbf{c}_1 t + \cdots + \mathbf{c}_n t^n =: \mathbf{C}\mathbf{T}_n(t)$$

Eelmine kord

- Kõvera konstrueerime *kontrollpunktide* abil.
  - Kõver *interpoleerib* punkte (Lagrange'i kõver, interpoleeriv splain).
  - Kõver *aproksimeerib* punkte (Bezier' kõver, B-splain).

Guess the curves!



Eelmine kord

- Kõvera esitus geomeetria- ja baasimaatriksite kaudu:

$$\mathbf{p}(t) = \mathbf{GMT}(t)$$

- Esitus segamis (ehk baas-)funktsioonide kaudu:

$$\mathbf{p}(t) = \sum_{i=0}^{n} b_i(t)\mathbf{p}_i, \qquad \sum_{i=0}^{n} b_i(t) = 1$$

- Mõnede kuupkõverate baasimaatriksid: $\mathbf{M}_L, \mathbf{M}_B, \mathbf{M}_H$

Seekord

- B-splain: populaarseim kõver graafikas.
- Ratsionaalne B-splain, NURBS.
- Pinnad: tensorkorrutise pinnad.
- Kõverate ja pindade renderdamine.
- Kõverad, pinnad & OpenGL.

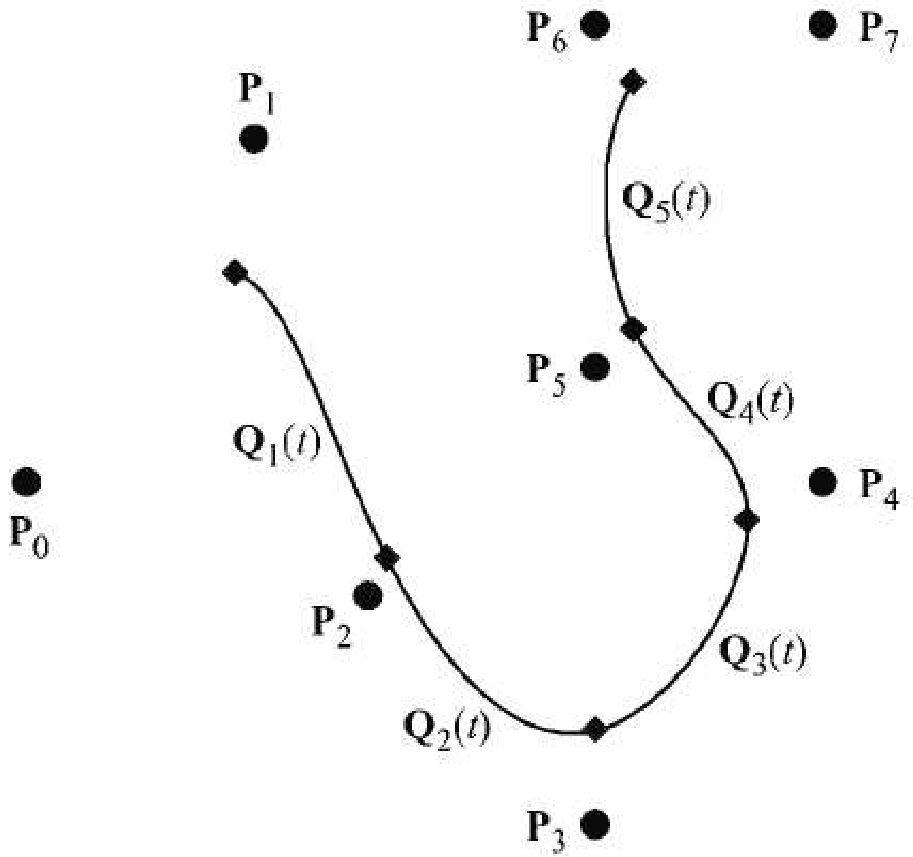
Milleks B-Splain?

- Olgu meil antud n punkti, ning soovime konstrueerida nende abil C^2-sileda kõvera.
- $n-1$ astme polünomiaalne kõver (Lagrange, Bezier) ei sobi kuna ta on liiga ebaefektiivne (kõrge astme polünoom).
- Tavaline interpoleeriv kuupsplain võib mitte sobida sest teda ei saa konstrueerida inkrementaalselt.
- Proovime leida kõverat mis oleks tükiti kuupkõver, oleks C^2-sile sõlmepunktides ning lubaks inkrementaalset spetsifitseerimist.

B-Splain

- B-splain on tükiti kuupkõver, mille iga tükk on konstrueeritud 4 järjestikuse punkti järgi, ning tükid rahuldavad järgmist omadust:
  - Olgu tükk $\mathbf{q}_1(t)$ määratud punktide $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$ poolt, ning tükk $\mathbf{q}_2(t)$ määratud punktide $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4$ poolt.
  - Siis nendest kahest tükist kokkupandud kõver on C^2-pidev.
- Seda tingimust rahuldavate kõverajupide abil saab konstrueerida kuitahes pika n-punkte aproksimeeriva kõvera.

B-Splain



B-Splaini konstrueerimine

- Esitame B-splaini segamisfunktsioonide kaudu:

$$\mathbf{q}(t) = \sum_{i=0}^{3} b_i(t)\mathbf{p}_i$$

- Olgu jupp $\mathbf{q}_1(t)$ määratud kontrollpunktide $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$ poolt ning jupp $\mathbf{q}_2(t)$ — kontrollpunktide $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4$ poolt:

$$\mathbf{q}_1(t) = b_0(t)\mathbf{p}_0 + b_1(t)\mathbf{p}_1 + b_2(t)\mathbf{p}_2 + b_3(t)\mathbf{p}_3$$

$$\mathbf{q}_2(t) = b_0(t)\mathbf{p}_1 + b_1(t)\mathbf{p}_2 + b_2(t)\mathbf{p}_3 + \mathbf{p}_3(t)\mathbf{p}_4$$

B-Splaini konstrueerimine

- Kõver $(\mathbf{q}_1(t), \mathbf{q}_2(t))$ peab olema pidev, seega:

$$\mathbf{q}_1(1) = \mathbf{q}_2(0)$$

$$\Leftrightarrow$$

$$\mathbf{q}_1(1) = b_0(1)\mathbf{p}_0 + b_1(1)\mathbf{p}_1 + b_2(1)\mathbf{p}_2 + b_3(1)\mathbf{p}_3 =$$

$$= b_0(0)\mathbf{p}_1 + b_1(0)\mathbf{p}_2 + b_2(0)\mathbf{p}_3 + b_3(0)\mathbf{p}_4 = \mathbf{q}_2(0)$$

- Järelikult:

$$b_0(1) = 0 \qquad b_1(1) = b_0(0) \qquad b_2(1) = b_1(0)$$

$$b_3(1) = b_2(0) \qquad b_3(0) = 0$$

B-Splaini konstrueerimine

- Kõver $(\mathbf{q}_1(t), \mathbf{q}_2(t))$ peab olema C^1-pidev, seega:

$$\mathbf{q}_1'(1) = \mathbf{q}_2'(0)$$

$$\Leftrightarrow$$

$$\begin{aligned}\mathbf{q}_1'(1) &= b_0'(1)\mathbf{p}_0 + b_1'(1)\mathbf{p}_1 + b_2'(1)\mathbf{p}_2 + b_3'(1)\mathbf{p}_3 = \\ &= b_0'(0)\mathbf{p}_1 + b_1'(0)\mathbf{p}_2 + b_2'(0)\mathbf{p}_3 + b_3'(0)\mathbf{p}_4 = \mathbf{q}_2'(0)\end{aligned}$$

- Järelikult:

$$b_0'(1) = 0 \qquad b_1'(1) = b_0'(0) \qquad b_2'(1) = b_1'(0)$$

$$b_3'(1) = b_2'(0) \qquad b_3'(0) = 0$$

B-Splaini konstrueerimine

- Analoogselt, C^2-pidevuse nõuest saame:

$$b_0''(1) = 0 \qquad b_1''(1) = b_0''(0) \qquad b_2''(1) = b_1''(0)$$
$$b_3''(1) = b_2''(0) \qquad b_3''(0) = 0$$

- Kokku 15 võrrandit ning 16 tundmatut. Viimase võrrandina lisame nõuet

$$b_0(0) + b_1(0) + b_2(0) + b_3(0) = 1$$

- Tulemusena on üheselt lahenduv LVS kust saame kätte polünoome $b_i(t)$.

B-Splaini segamisfunktsioonid

$$b_0(t) = \frac{(1-t)^3}{6}$$

$$b_1(t) = \frac{4 - 6t^2 + 3t^3}{6}$$

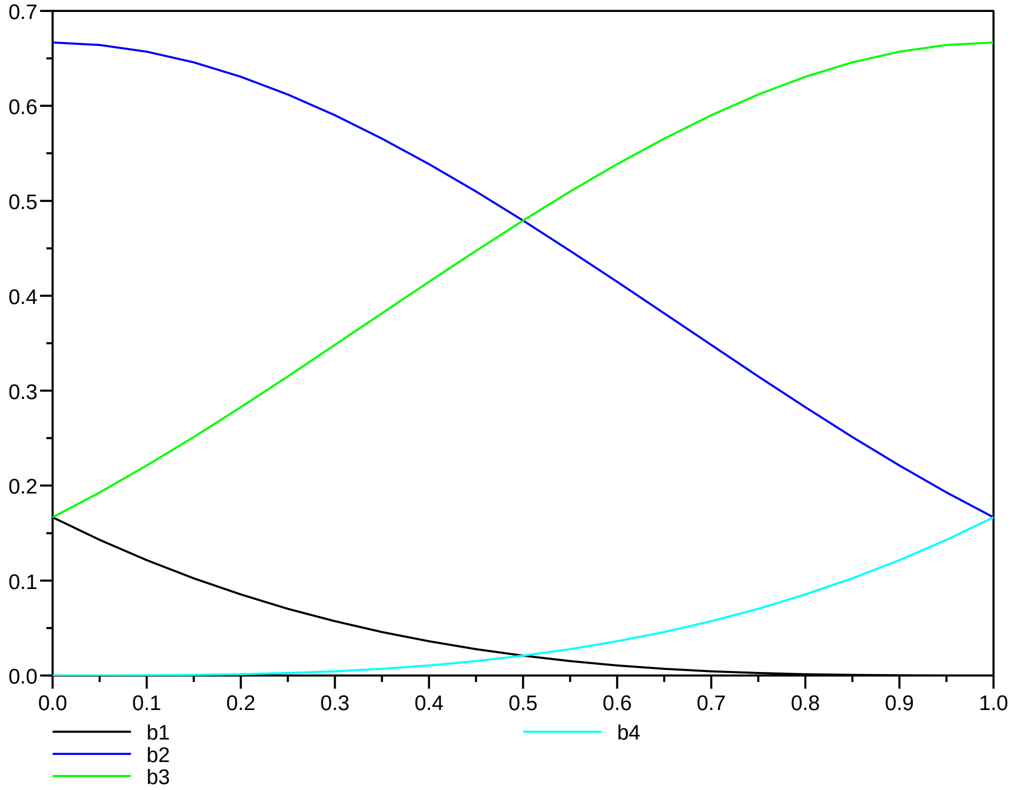
$$b_2(t) = \frac{1 + 3t + 3t^2 - 3t^3}{6}$$

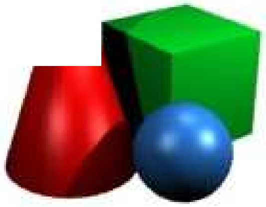
$$b_3(t) = \frac{t^3}{6}$$

B-Splaini baasimaatriks

$$M_{BS} = \frac{1}{6}\begin{pmatrix} 1 & -3 & 3 & -1 \\ 4 & 0 & -6 & 3 \\ 1 & 3 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

B-Splaini segamisfunktsioonid

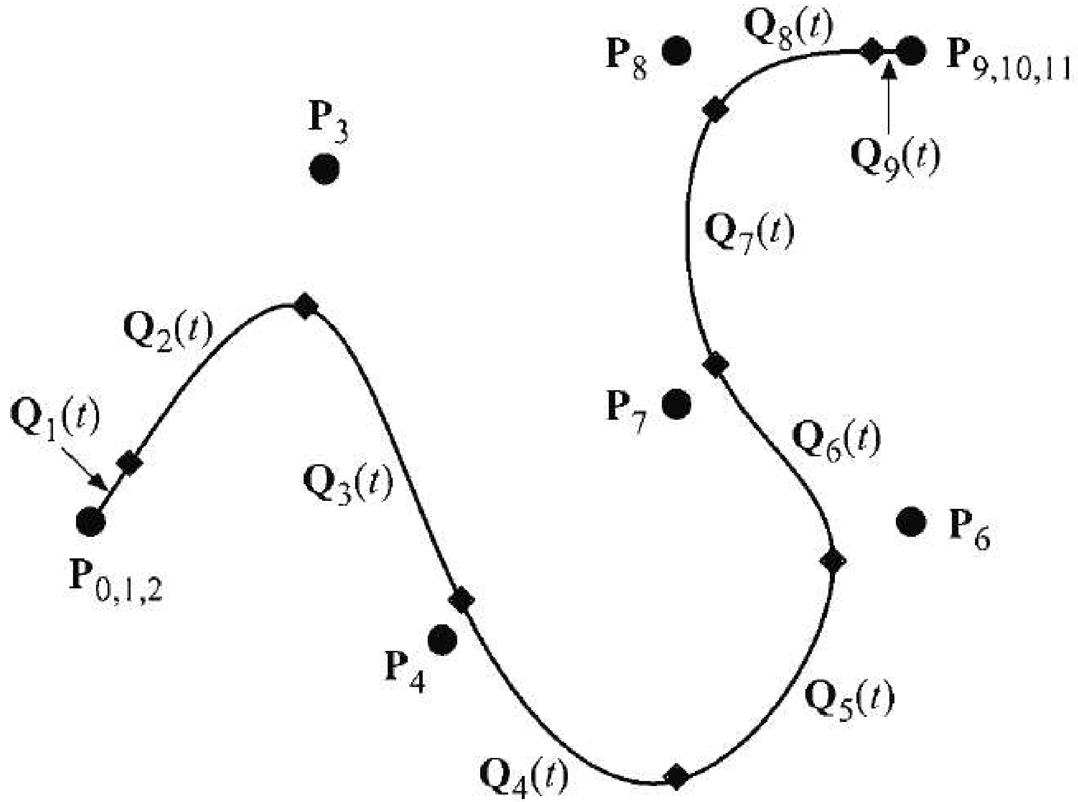




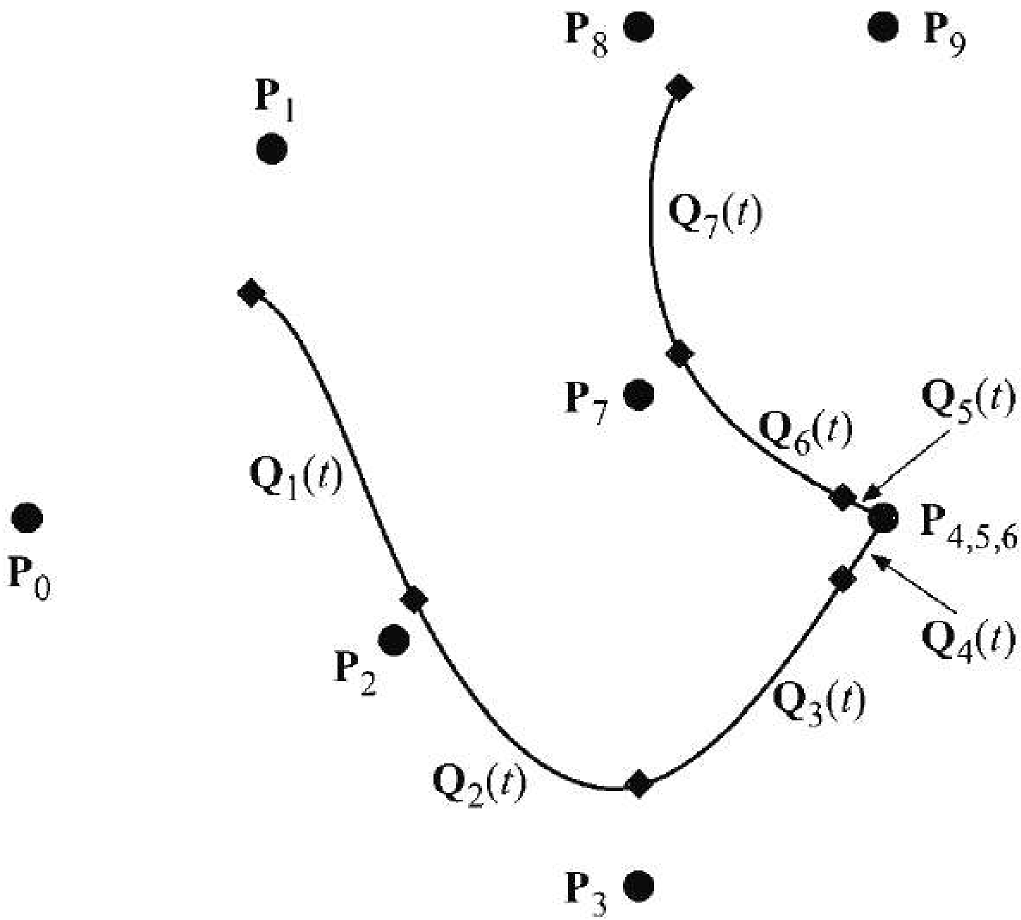
B-Splain

- Kontrollpunkte võib korrata, siis nihkub joon nendele lähemale.
- Kui mõni punkt on korratud 3 korda, joon läbib seda.
- Iga kordus kaotab ühte siledusastet.
- Kontrollpunktide kordamist kasutatakse tihti esimese ja viimase kontrollpunkti puhul.

Korduvad lõpp-punktid



Korduv vahepunkt



B-splain = basis-spline

- Defineerisime iga B-splaini tükki $\mathbf{q}_i(t)$ kui funktsiooni väärtustega lõigus $[0, 1]$.
- Paneme nüüd kõik tükid kokku üheks funktsiooniks $\mathbf{q}(t)$ määratud lõigus $[1, n-1]$:

$$\mathbf{q}(t) = \mathbf{q}_i(t-i), \quad \text{kui } i \le t \le i+1$$

- Osutub et funktsioon $\mathbf{q}(t)$ avaldub kui:

$$\mathbf{q}(t) = \sum_{i=1}^{n-1} N_i(t)\mathbf{p}_i$$

kus $N_i(t)$ on *baas*-funktsioonid.

B-splaini baasfunktsioonid

- Iga punkt $\mathbf{p}_i$ mõjutab kõverat $\mathbf{q}(t)$ neljas lõigus:
  - Ta on kaalutud funktsiooniga $b_0(t-(i+1))$ lõigus $[i+1, i+2]$
  - Ta on kaalutud funktsiooniga $b_1(t-i)$ lõigus $[i, i+1]$
  - Ta on kaalutud funktsiooniga $b_2(t-(i-1))$ lõigus $[i-1, i]$
  - Ta on kaalutud funktsiooniga $b_3(t-(i-2))$ lõigus $[i-2, i-1]$
  - Mujal on ta "kaalutud" funktsiooniga 0.

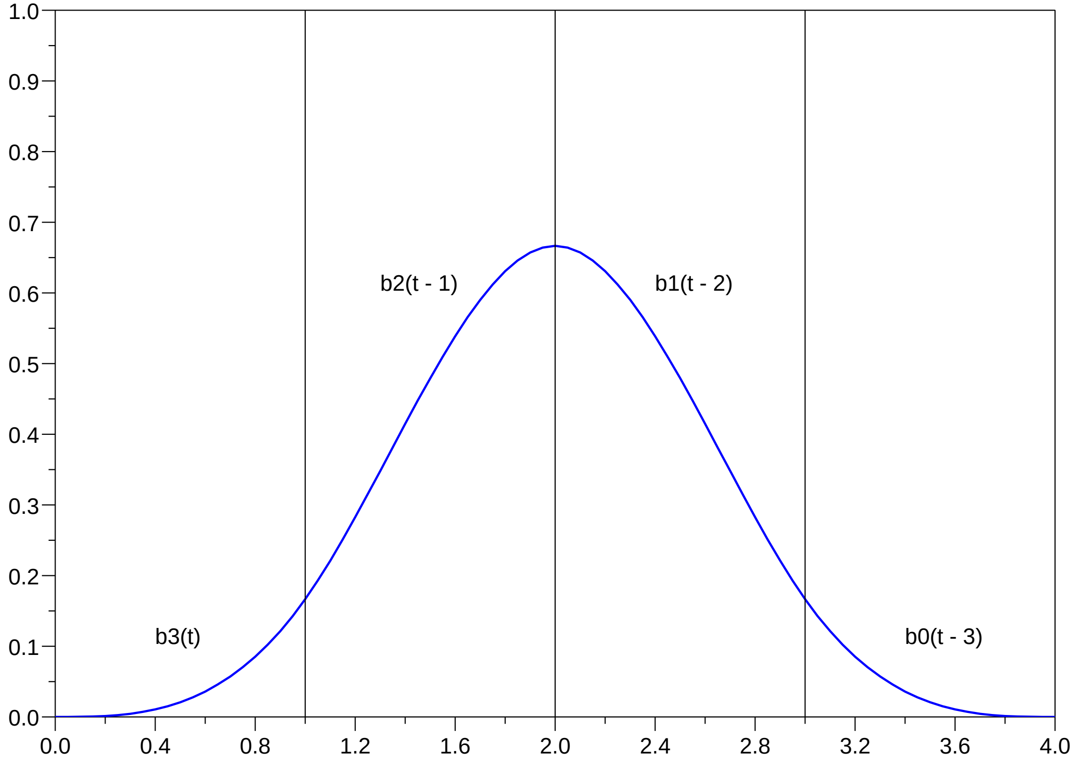
B-splaini baasfunktsioonid

- Seega võime lihtsalt öelda et iga punkt $\mathbf{p}_i$ on kaalutud lõigus $[1, n-1]$ funktsiooniga $N_i(t)$:

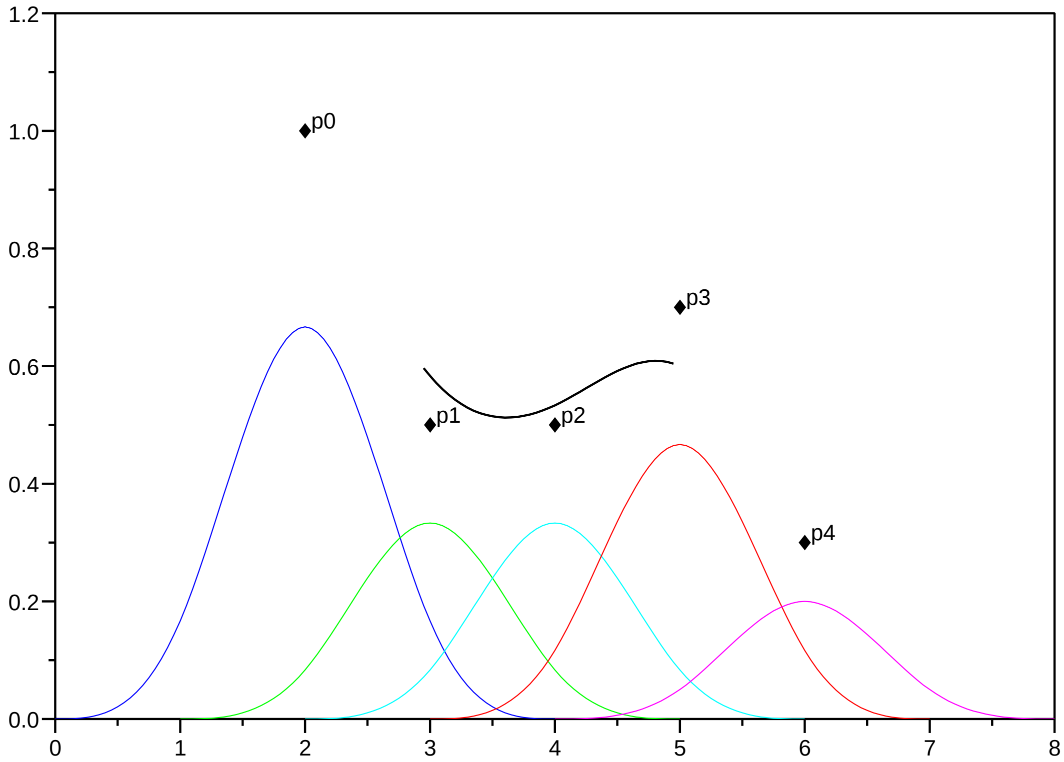
$$N_i(t) = \begin{cases} b_0(t-(i+1)), & t \in [i+1, i+2] \\ b_1(t-i), & t \in [i, i+1] \\ b_2(t-(i-1)), & t \in [i-1, i] \\ b_3(t-(i-2)), & t \in [i-2, i-1] \\ 0, & \text{otherwise} \end{cases}$$

$$\mathbf{q}(t) = \sum_{i=1}^{n-1} N_i(t)\mathbf{p}_i$$

B-splaini baasfunktsioon



B-splain



Üldistatud B-splain

- B-splaini mõistet saab üldistada suvalisele astmele.

$$\mathbf{q}(t) = \sum_{i=0}^{n} N_{i,k}(t)\mathbf{p}_i \qquad t \in [k, n+1]$$

  kus $N_{i,k}$ on k-nda astme B-splaini i-s baasfunktsioon.
- $N_{i,k}$ on C^{k-1}-sile tükiti k-nda astme polünoom

Cox – De Boor'i valemid

- Osutub et

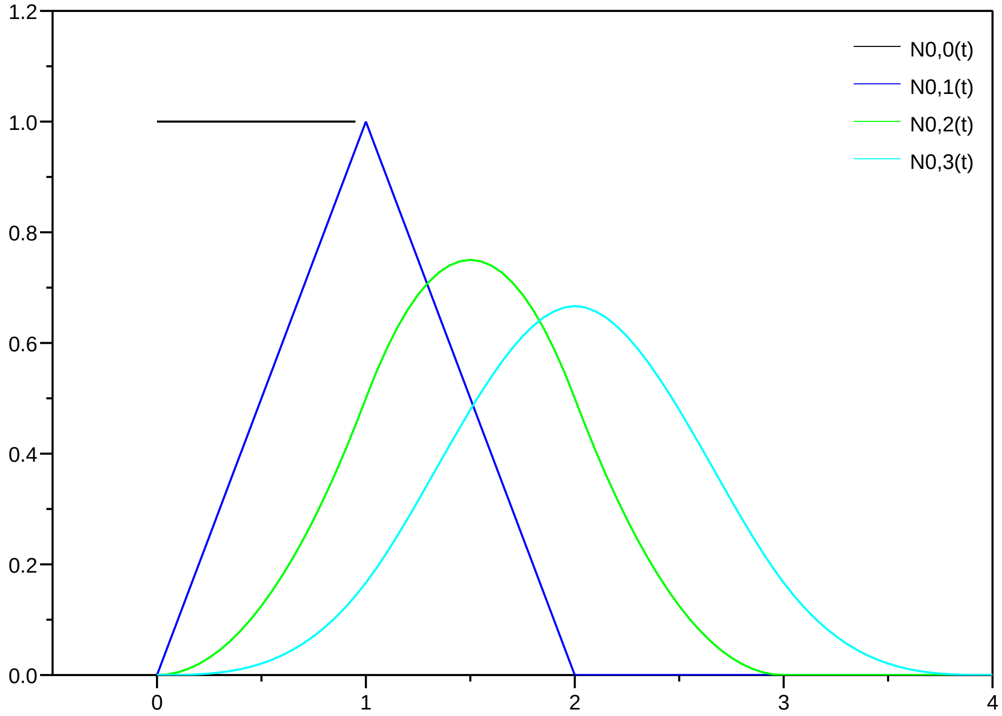
$$N_{i,0}(t) = \begin{cases} 1 & t \in [i, i+1) \\ 0 & \text{otherwise} \end{cases}$$

$$N_{i,k}(t) = \frac{t-i}{k} N_{i,k-1}(t) + \frac{i+k+1-t}{k} N_{i+1,k-1}(t)$$

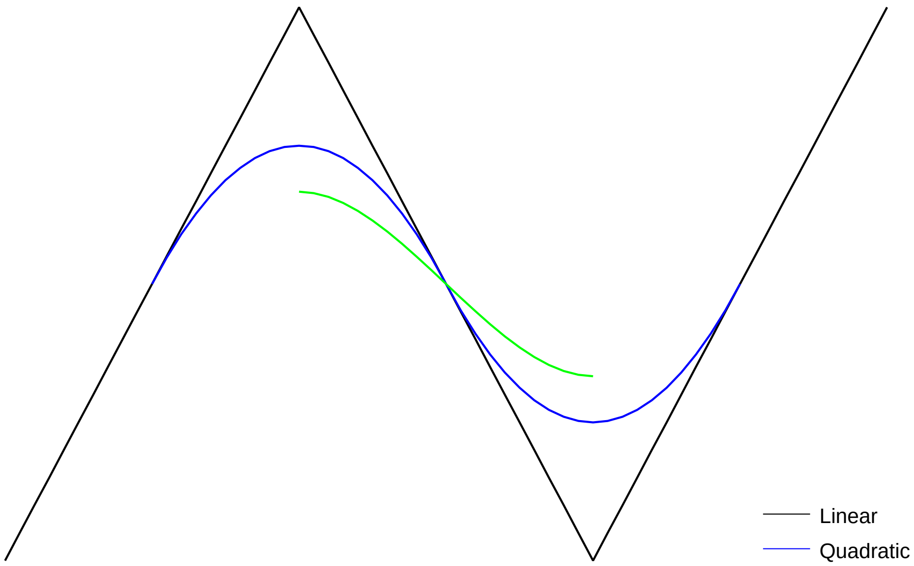
kus $N_{i,k}$ on k-nda astme B-splaini i-s baasfunktsioon.

- $N_{i,k}(t)$ on (konstandi täpsuseni) ainus funktsioon, mis teeb B-splaini C^{k-1}-siledaks.

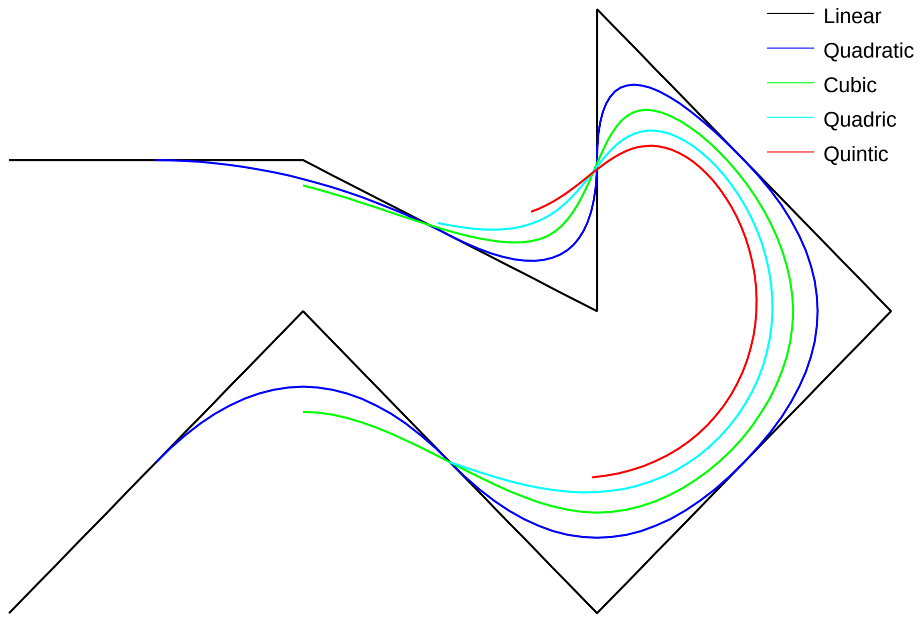
Üldistatud B-splaini baasf-nid $N_{0,k}$



Üldistatud B-splain



Üldistatud B-splain



B-splain mitteühtlasel võrgul

- B-splaini üks potentsiaalne puudus on selles et punktidega seotud baasfunktsioonid on ühe laiusega.
- See tähendab, et punktidele vastavad kõvera sõlmepunktid on parameetri suhtes sama kaugusel üks teisest ning igat parameetrivahemiku mõjutab sama arv kontrollpunkte.
- Mõnikord on kasuks määrata mõnes parameetri-vahemikus rohkem kontrollpunkte kui teises.
- Näiteks mõnikord on mugav lisada uusi kontrollpunkte teiste punktide “vahele”.

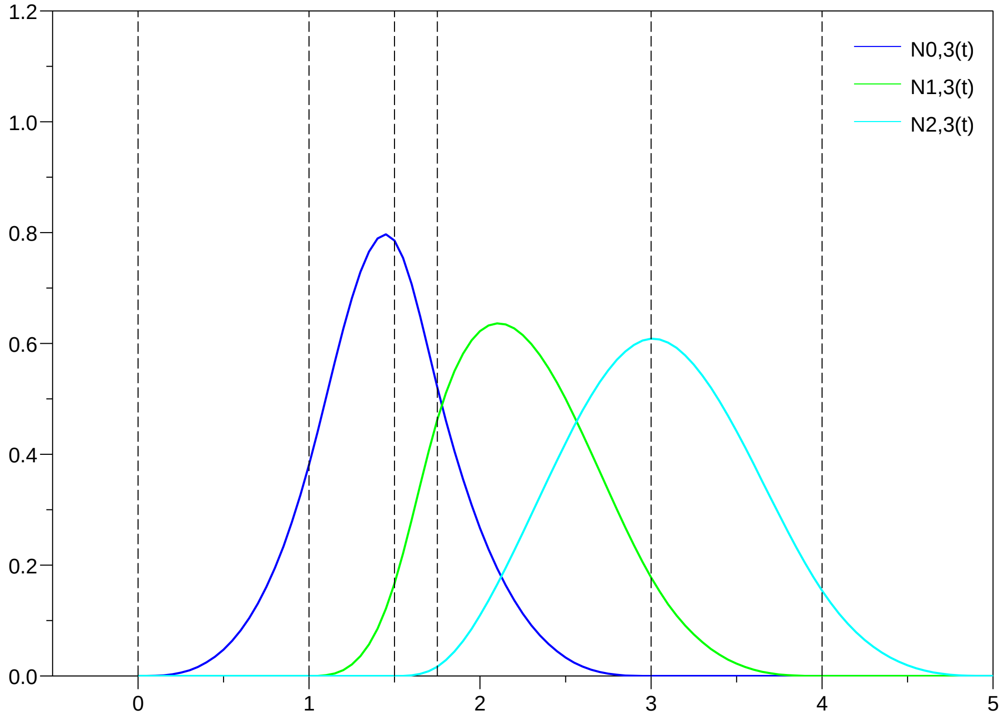
B-splain mitteühtlasel võrgul

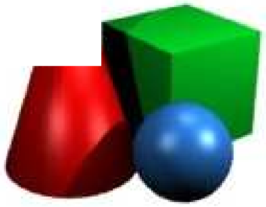
- Olgu siis koos kontrollpunktidega antud parameetri *sõlmeväärtused*: $t_0, t_1, \ldots, t_{n+k+1}$.
- Defineerime baasfunktsioone järgmiselt (*Cox–De Boor'i valemid*):

$$N_{i,0}(t) = \begin{cases} 1 & t \in [t_i, t_{i+1}) \\ 0 & \text{otherwise} \end{cases}$$

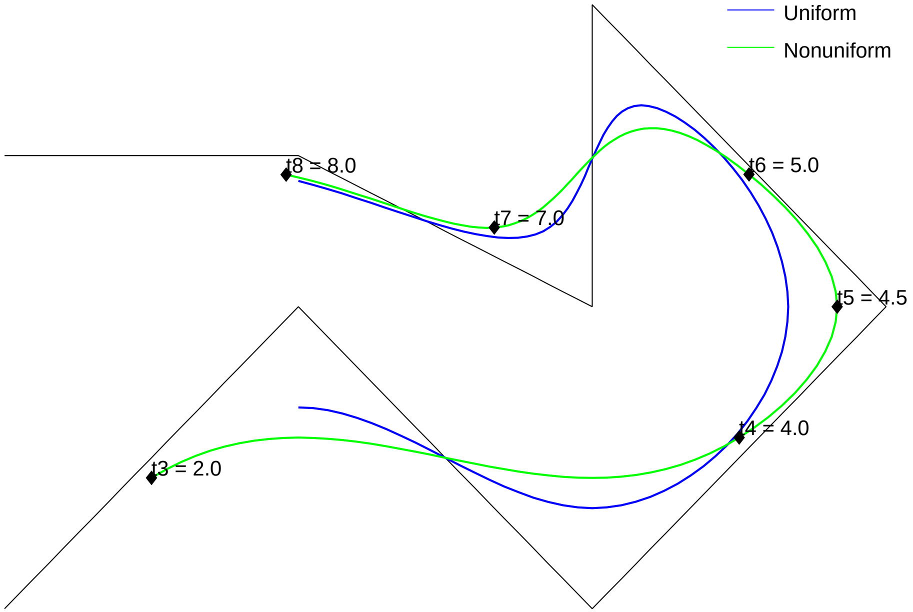
$$N_{i,k}(t) = \frac{t - t_i}{t_{i+k} - t_i} N_{i,k-1}(t) + \frac{t_{i+k+1} - t}{t_{i+k+1} - t_{i+1}} N_{i+1,k-1}(t)$$

Non-uniform B-spline



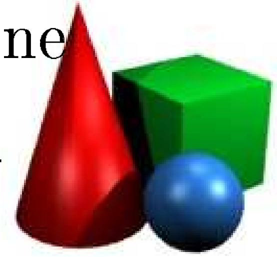


Non-uniform B-spline



Ratsionaalne B-splain

- Kõvera parameetriline esitus on sama kujuga nii 2, 3 kui ka 4-mõõtmelises ruumis.
- Kolmemõõtmelises graafikas kasutame põhiliselt neljamõõtmelisi homogeenseid koordinaate $(xw, yw, zw, w)^T$.
- Võime seega konstrueerida kõverat kohe homogeensetes koordinaatides, ning anda ette neljamõõtmelisi kontrollpunkte: $\mathbf{p}_i = (x_i w_i, y_i w_i, z_i w_i, w_i)^T$.
- Selliselt konstrueeritud kõver on invariantne perspektiivteisenduse suhtes!



Ratsionaalne B-splain

$$\begin{pmatrix} xw \\ yw \\ zw \\ w \end{pmatrix} = \sum_{i=0}^{n} N_{i,k}(t) w_i \begin{pmatrix} x_i \\ y_i \\ z_i \\ 1 \end{pmatrix}$$

kust

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \frac{\sum_{i=0}^{n} N_{i,k}(t) w_i \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix}}{w}$$

Ratsionaalne B-splain

kust

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \frac{\sum_{i=0}^{n} N_{i,k}(t) w_i \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix}}{\sum_{i=0}^{n} N_{i,k}(t) w_i}$$

ehk

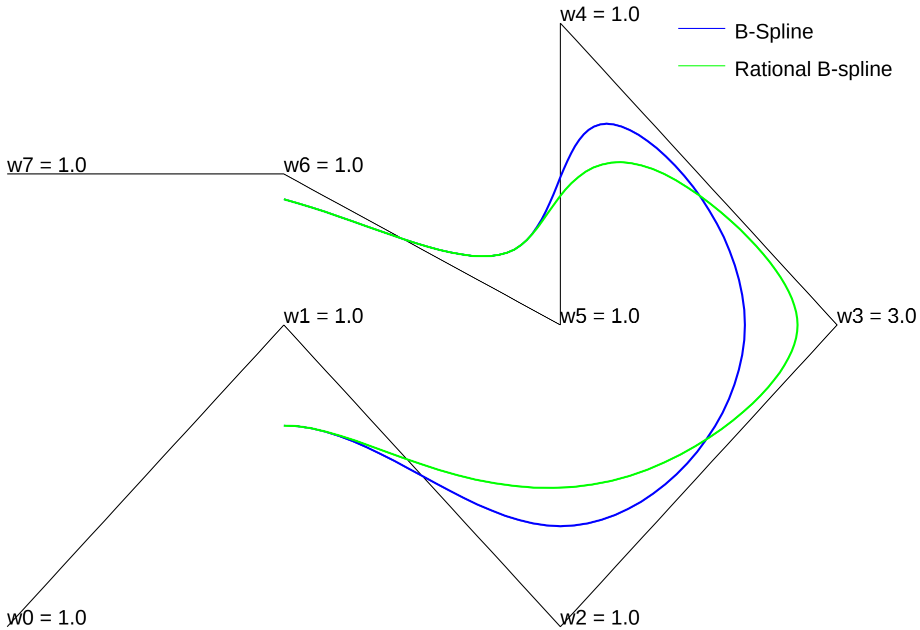
$$\mathbf{p}(t) = \frac{\sum_{i=0}^{n} N_{i,k}(t) w_i \mathbf{p}_i}{\sum_{i=0}^{n} N_{i,k}(t) w_i}$$

Ratsionaalne B-splain

$$\mathbf{p}(t) = \frac{\sum_{i=0}^{n} N_{i,k}(t) w_i \mathbf{p}_i}{\sum_{i=0}^{n} N_{i,k}(t) w_i}$$

- Kõver on invariantne nii afiinse kui ka perspektiivteisenduse suhtes (saab alguses teisendada kontrollpunkte ja alles siis konstrueerida kõverat).
- Väärtuste w_i abil saab anda kontrollpunktidele erinevad “kaalud”.

Ratsionaalne B-splain



NURBS

- Üldjuhul võivad parameetri sõlmväärtused olla määratud mitteühtlaselt, siis nimetatakse sellist kõverat *Non-Uniform Rational B-Spline (NURBS)*.
- *NURBS*-kõverad ja pinnad on väga paindlikud ning lubavad määrata nii standartseid kujud (ringjoon, sfäär, toor) kui ka suvaliselt modelleeritud.
- Selle pärast *NURBS* on praegu de-facto standard CAD/CAM-rakendustes ning ka arvutigraafika-kunstis.

Kokkuvõte: Kõverad

- Interpoleerivad:
  - Lagrange'i polünoom (suht kasutu)
  - Interpoleeriv splain (CAD/CAM)
- Aproksimeerivad:
  - Bezier kõver (Photoshop/GIMP/jne, mugav vaheesitus)
  - B-splain (trajektooride esitus)
  - NURBS (CAD/CAM, Blender/Maya/jne)

Pinnad

- Pindade puhul on asjad analoogsed kõveratele:
  - Parameetriline esitus: $\mathbf{p}(u,v) = \mathbf{f}(u,v)$
  - Polünomiaalne pind: $x(u,v), y(u,v), z(u,v)$ on polünoomid u ja v-st:

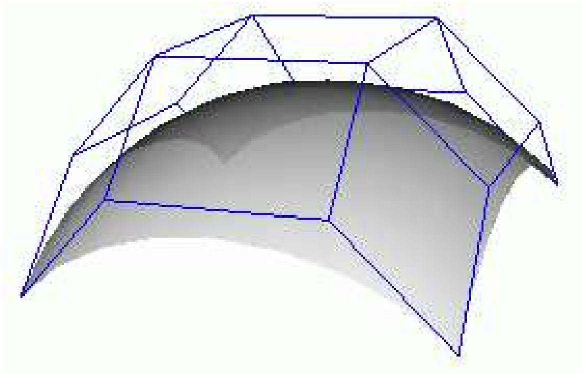
$$x(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{n} c_{xij} u^i v^j = \mathbf{U}_n(u)^T \mathbf{C}_x \mathbf{V}_n(v)$$

$$\mathbf{U}_n(u) = (1, u, u^2, \ldots, u^n)^T$$

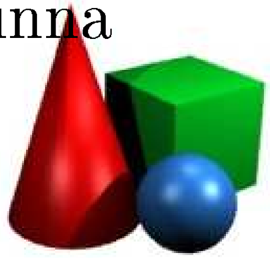
$$\mathbf{V}_n(v) = (1, v, v^2, \ldots, v^n)^T$$

Pinna kontrollpunktid

- Pinnajupid konstrueerime kontrollpunktide abil, d-astme pinna jaoks läheb vaja $(d+1)^2$ kontrollpunkti.



- Nagu ka kõverate puhul on kõige levinumad kuuppinnad ja tükiti-kuuppinnad. Kuuppinna määramiseks on vaja 16 kontrollpunkti.



Pinna segamisfunktsioonid

- Analoogselt kõveratele, peavad pinnad esituma kontrollpunktide lineaarkombinatsioonina, mille kordajad on määratud *segamisfunktsioonide* poolt:

$$\mathbf{p}(u,v) = \sum_i \sum_j b_{ij}(u,v)\mathbf{p}_{ij}$$

Tensorkorrutise pind

- Kõik praktikas kasutatud pinnad on nn *tensorkorrutise pinnad (tensor product surfaces)*, nimelt esituvad nad kujul:

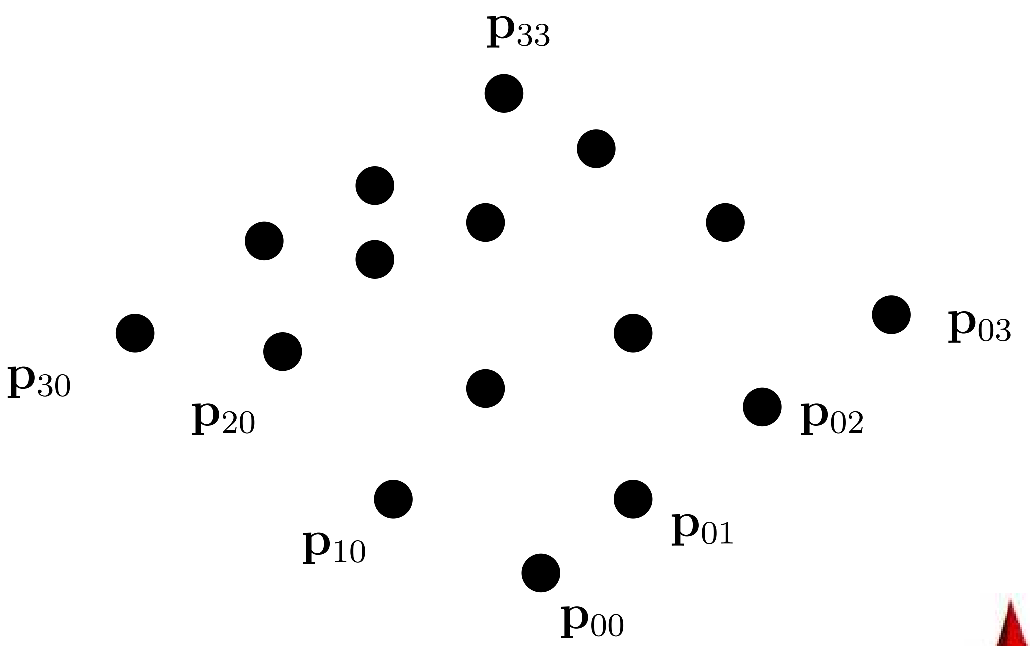
$$\mathbf{p}(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{n} b_i(u)b_j(v)\mathbf{p}_{ij} = \mathbf{b}(u)^T\mathbf{P}_*\mathbf{b}(v)$$

$$= \mathbf{U}_n(u)^T\mathbf{M}^T\mathbf{P}_*\mathbf{M}\mathbf{V}_n(v)$$

kus $b_i(t)$ on mõnda kõvera baasfunktsioonid.

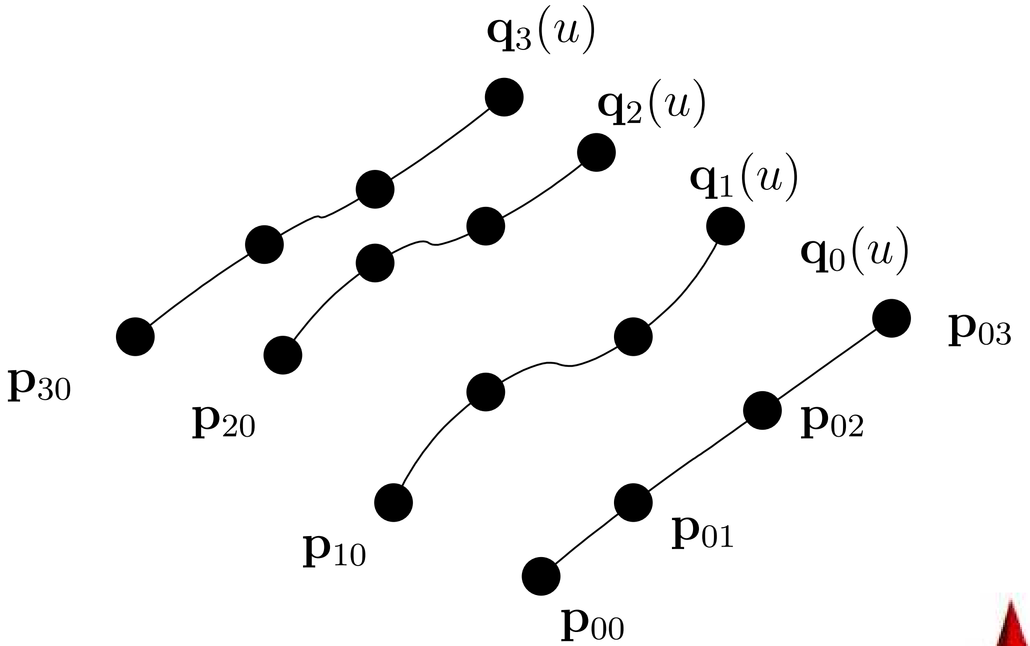
Tensorkorrutise pind

- Olgu meil antud 16 kontrollpunkti $\mathbf{p}_{ij}$, mille järgi proovime konstrueerida mõnda kuuppinna.



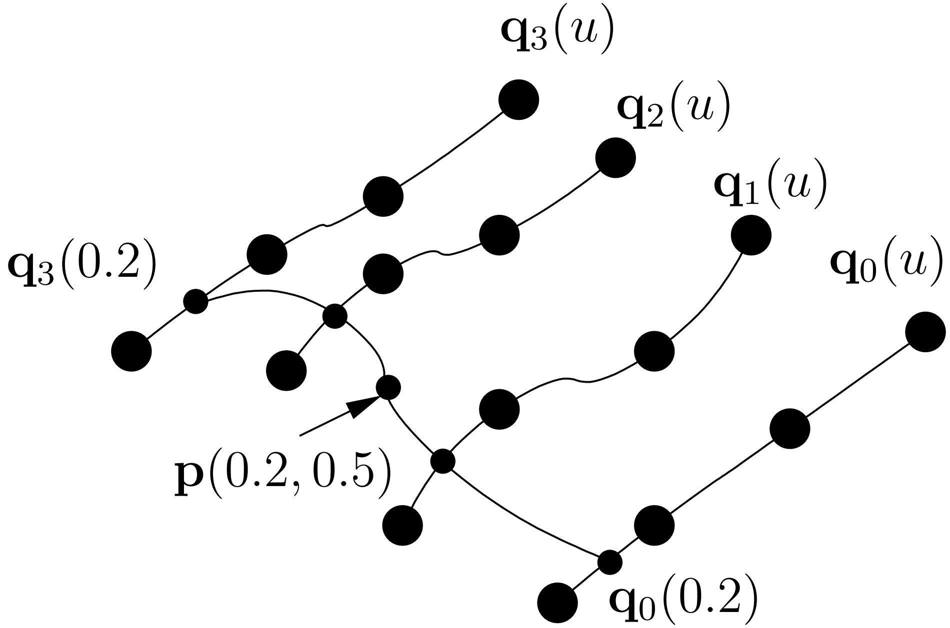
Tensorkorrutise pind

- Olgu kõverad $\mathbf{q}_i(u)$ konstrueeritud kontrollpunktide $\mathbf{p}_{i0}, \mathbf{p}_{i1}, \mathbf{p}_{i2}, \mathbf{p}_{i3}$ abil:



Tensorkorrutise pind

- Nüüd iga fikseeritud u jaoks saame konstrueerida kõvera $\mathbf{p}(u,v)$ kasutades kontrollpunktidena $\mathbf{q}_0(u), \mathbf{q}_1(u), \mathbf{q}_2(u), \mathbf{q}_3(u)$:



Tensorkorrutise pind

- Seega

$$\mathbf{p}(u,v) = \sum_{j=0}^{3} b_j(v)\mathbf{q}_j(u) =$$

$$= \sum_{j=0}^{3} b_j(v) \left(\sum_{i=0}^{3} b_i(u)\mathbf{p}_{ij} \right) =$$

$$= \sum_{i} \sum_{j} b_i(u) b_j(v) \mathbf{p}_{ij}$$

Tensorkorrutise pind

- Selge et selline konstruktsioon ei sõltu segamisfunktsioonidest $b_i(t)$ seega niimoodi saame konstrueerida:
  - Lagrange-i interpoleeriva pinna
  - Interpoleeriva splain-pinna.
  - Bezier pinna
  - B-splain pinna
  - NURBS-pinna

Kõverate ja pindade renderdamine

- Kolm põhilist viisi kõverate renderdamiseks:
  - Konverteerime meshiks etteantud tipude arvuga.
  - Konverteerime meshiks rekursiivse alajaotamise meetodiga.
  - Raytracing.

Meshiks konverteerimine

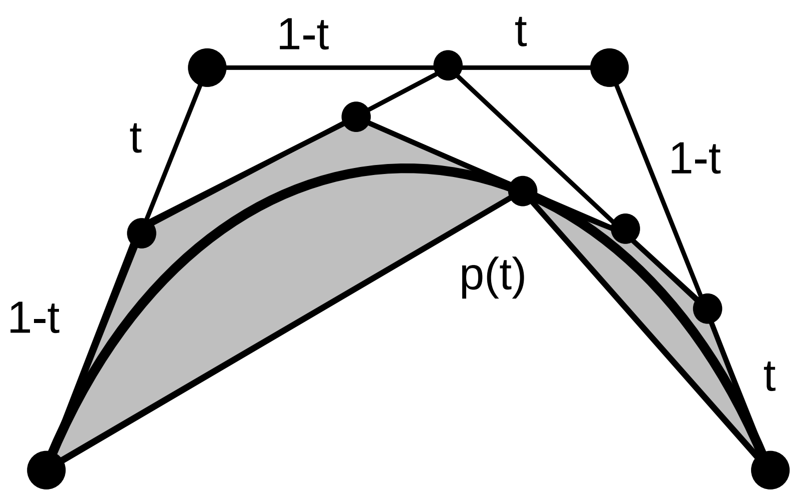
- Olgu $\mathbf{p}(u, v)$, $u \in [0, 1], v \in [0, 1]$ mingi pind või pinna tükk.
- Sellest saab konstrueerida meshi etteantud tipude arvuga $(n + 1) \times (m + 1)$:

$$\mathbf{m}_{ij} = \mathbf{p}(i/n, j/m)$$

- Meetodi põhiline miinus on selles et tuleb eelnevalt määrata kui palju tippe peab meshil olema, lisaks on meshi resolutsioon samasugune terve pinna peal.

Rekursiivne alajaotamine

- Parem oleks arvestada ekraani resolutsiooniga ning pinna keerukusega, seda teeb Bezier kõvera *rekursiivse alajaotamise* meetod (mis on ka väga efektiivne):



Rekursiivne alajaotamine

- Alajaotamisalgoritm annab võimalust konverteerida ühe suure Bezier kõvera kaheks väiksemaks.
- Sellist alajaotamist saame korrata rekursiivselt seni kuni tükid on piisavalt sirged või neid piisavalt palju, sealt alates joonistame välja objekti polügonid.
- Mitte-Bezier kõveraid peame eelnevalt konverteerima Bezier kujule.

Bezier kujuks viimine

- Olgu B-splaini kontrollpunktid antud maatriksis $\mathbf{P}_{BS}$. Kuidas leida samasuguse Bezier kõvera kontrollpunkte $\mathbf{P}_B$?

$$\begin{aligned}
\mathbf{P}_{BS}\mathbf{M}_{BS}\mathbf{T}(t) &= \mathbf{P}_B\mathbf{M}_B\mathbf{T}(t) \\
\mathbf{P}_{BS}\mathbf{M}_{BS} &= \mathbf{P}_B\mathbf{M}_B \\
\mathbf{P}_B &= \mathbf{P}_{BS}\mathbf{M}_{BS}\mathbf{M}_B^{-1}
\end{aligned}$$

Raytracing

- Ruut- ja kuuppindade ja -kõverate puhul on võimalik leida lõikumispunkte kiirtega raytracing algoritmides.
- Selleks tuleb lahendada vastava ruut või kuupvõrrandit.
- Eelistatakse ruutpindasid ilmutamata kujul *(quadratic patches)*.

Kõverad OpenGL-is

- OpenGL toetab sisseehitatult ainult Bezier-kujul kõveraid. Kõike muud tuleb Bezier kujuks viia.
- NURBS-kõveraid toetab GLU teek (mis teisendabki neid Bezier kujule ise).
- Kõverate joonistamiseks kasutatakse nn. *evaluatorit*.
- Üldine kasutusskeem on järgmine:
  - Määra kõvera parameetrid `glMap1f` abil.
  - Kõvera punkte saab kätte `glEvalCoord1f` abil.

Kõverad OpenGL-is

```
GLfloat control_points[4][3] =
  {{0, 0, 0}, {1, 1, 0},
   {3, 1, 0}, {4, 0, 1}};

glEnable(GL_MAP1_VERTEX_3);
glMap1f(GL_MAP1_VERTEX_3, 0.0, 1.0,
                3, 4, &control_points[0][0]);
```

Kõverad OpenGL-is

- Kõvera joonistamine:

```
glBegin(GL_LINE_STRIP);
  for (i = 0; i <= 20; i++)
    glEvalCoord1f((GLfloat) i/20.0);
glEnd();
```

- või samaväärselt

```
glMapGrid1f(20, 0.0, 1.0);
glEvalMesh1(GL_LINE, 0, 20);
```

Kõverad OpenGL-is

- Evaluator oskab genereerida mitte ainult meshide tipud vaid ka normaalid, värvid ja tekstuuri koordinaadid.
- Näiteks kui on vaja interpoleerida värvi mööda kõverat saab öelda:

```
glEnable(GL_MAP1_COLOR_4);
glMap1f(GL_MAP1_COLOR_4, 0.0, 1.0,
        4, 4, &control_colors[0][0]);
```

Pinnad OpenGL-is

- Pindadega samamoodi, initsialiseerime evaluator-i:

```
GLfloat ctrl_points[4][4][3] = ...;
glEnable(GL_MAP2_VERTEX_3, 0.0, 1.0, 3, 4,
   0.0, 1.0, 12, 4, &ctrl_points[0][0][0]);
```

- ning kasutame seda kas "käsitsi" `glEvalCoord2f` abil või siis "automaatselt":

```
glMapGrid2f(10, 0.0, 1.0, 10, 0.0, 1.0);
glEvalMesh2(GL_FILL, 0, 10, 0, 10);
```

NURBS

```
GLUnurbsObj* nurb = gluNewNurbsRenderer();
gluNurbsProperty(nurb, GLU_DISPLAY_MODE,
                       GLU_FILL);
gluBeginSurface(nurb);
gluNurbsSurface(nurb, 8, knots, 8, knots,
   12, 3, &control_points[0][0][0],
   4, 4, GL_MAP2_VERTEX_3);
gluEndSurface(nurb);
gluDeleteNurbsRenderer(nurb);
```

Kokkuvõte

- Polünomiaalsed pinnad ja kõverad:
  - Interpoleerivad: Lagrange, interpoleeriv splain.
  - Aproksimeerivad: Bezier, B-splain, NURBS.
- Esitus: segamisfunktsioonid, baasimaatriks.
- Renderdamine: teisendamine mesh-iks, raytracing.
- OpenGL: `glMap`, `glEvalCoord`, `glMapGrid`, `glEvalMesh`
- GLU: `gluNewNurbsRenderer`, `gluNurbsSurface`, etc.

Küsimused