Számítógépes Grafika

Bán Róbert
robert.ban102+cg@gmail.com

Eötvös Loránd Tudományegyetem
Informatikai Kar

2021-2022. őszi félév

Tartalom

Tartalom

Tartalom

Geometria közvetlen tárolása

- A mai modern grafikus API-k pont, szakasz és háromszög primitíveket használnak

Geometria közvetlen tárolása

- A mai modern grafikus API-k pont, szakasz és háromszög primitíveket használnak
- Ebben a pontban megnézzük, hogy milyen módon tárolhatóak el ezek hatékonyan

Geometria közvetlen tárolása

- A mai modern grafikus API-k pont, szakasz és háromszög primitíveket használnak
- Ebben a pontban megnézzük, hogy milyen módon tárolhatóak el ezek hatékonyan
- Továbbá azt is, hogy miképp tudunk szomszédossági információkat is kinyerni

Geometria tárolása – brute force

- Általánosabban nézve, legyenek a primitíveink síkbeli poligonok, amiket csúcspontjaik felsorolásával reprezentálunk (valamilyen rögzített bejárási irány szerint)

Geometria tárolása – brute force

- Általánosabban nézve, legyenek a primitíveink síkbeli poligonok, amiket csúcspontjaik felsorolásával reprezentálunk (valamilyen rögzített bejárási irány szerint)
- Poligonokkal kapcsolatos feladatok:
  - tárolás
  - transzformálás
  - szomszédossági lekérdezések

Geometria „*brute force*” tárolása

```
struct triangle {
        float x1,y1,z1;
        float x2,y2,z2;
        float x3,y3,z3;
};
```

A „*brute force*” tárolás elemzése

- *Tárolás:* ha vannak poligonoknak közös csúcsai, akkor ezeket többször tároljuk – feleslegesen $\rightarrow$ nem túl jó

A „*brute force*” tárolás elemzése

- *Tárolás:* ha vannak poligonoknak közös csúcsai, akkor ezeket többször tároljuk – feleslegesen $\rightarrow$ nem túl jó
- *Transzformálás:* a közös csúcsokra annyiszor fogjuk elvégezni a transzformációkat, ahányszor szerepelnek $\rightarrow$ nem hatékony

A „*brute force*” tárolás elemzése

- *Tárolás:* ha vannak poligonoknak közös csúcsai, akkor ezeket többször tároljuk – feleslegesen $\rightarrow$ nem túl jó
- *Transzformálás:* a közös csúcsokra annyiszor fogjuk elvégezni a transzformációkat, ahányszor szerepelnek $\rightarrow$ nem hatékony
- *Lekérdezések:* fogalmunk sincs, ki kinek a szomszédja, csak az összes csúcs bejárásával tudunk eredményre jutni $\rightarrow$ katasztrófa

A „*brute force*” tárolás elemzése

- *Tárolás:* ha vannak poligonoknak közös csúcsai, akkor ezeket többször tároljuk – feleslegesen $\rightarrow$ nem túl jó
- *Transzformálás:* a közös csúcsokra annyiszor fogjuk elvégezni a transzformációkat, ahányszor szerepelnek $\rightarrow$ nem hatékony
- *Lekérdezések:* fogalmunk sincs, ki kinek a szomszédja, csak az összes csúcs bejárásával tudunk eredményre jutni $\rightarrow$ katasztrófa
- Egyetlen előnye, hogy ennél egyszerűbben már nem is lehetne tárolni

Index pufferek

- Alapötlet: tároljunk minden csúcsot egyszer, egy nagy közös tömbben!

Index pufferek

- Alapötlet: tároljunk minden csúcsot egyszer, egy nagy közös tömbben!
- A poligonok csak hivatkozzanak a csúcsok tömbjének elemeire.

Index pufferek

- Alapötlet: tároljunk minden csúcsot egyszer, egy nagy közös tömbben!
- A poligonok csak hivatkozzanak a csúcsok tömbjének elemeire.
- Ez az *index puffer*.

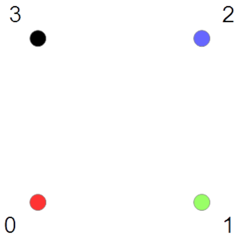
Index pufferek

- Alapötlet: tároljunk minden csúcsot egyszer, egy nagy közös tömbben!
- A poligonok csak hivatkozzanak a csúcsok tömbjének elemeire.
- Ez az *index puffer*.
- Minden GPU támogatja.

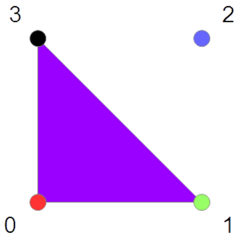
Index pufferek

```
struct triangle {
        unsigned int a,b,c;
};

struct vec3 {
        float x,y,z;
};

std::vector<vec3> vertexBuffer;
std::vector<unsigned int> indexBuffer;
```

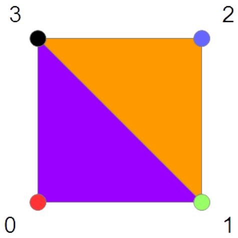
Index puffer



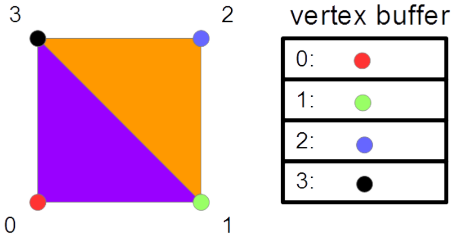
Index puffer



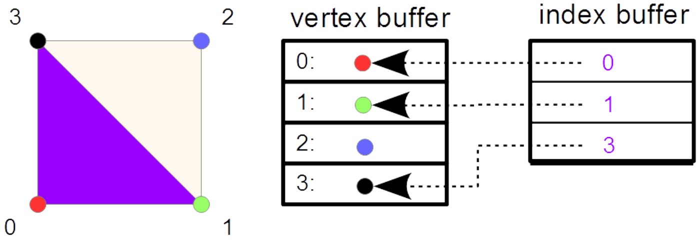
Index puffer



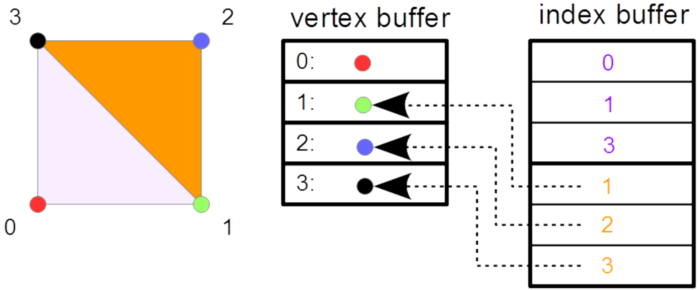
Index puffer



Index puffer



Index puffer



Példa

- Vegyünk egy $N \times N$ db négyzetből álló rácsot! Mennyi adatot kell eltárolnunk ehhez?

Példa

- Vegyünk egy $N \times N$ db négyzetből álló rácsot! Mennyi adatot kell eltárolnunk ehhez?
- Mérete *index puffer* nélkül: 4 csúcs/négyzet, $N \times N$ négyzet: $4N^2$ *csúcspont*.

Példa

- Vegyünk egy $N \times N$ db négyzetből álló rácsot! Mennyi adatot kell eltárolnunk ehhez?
- Mérete *index puffer* nélkül: 4 csúcs/négyzet, $N \times N$ négyzet: $4N^2$ *csúcspont*.
- Mérete *index puffer*-rel: $(N+1) \times (N+1) = N^2 + 2N + 1$ *csúcs* $(+\ 4N^2$ egész szám, *index*)

Példa

- Vegyünk egy $N \times N$ db négyzetből álló rácsot! Mennyi adatot kell eltárolnunk ehhez?
- Mérete *index puffer* nélkül: 4 csúcs/négyzet, $N \times N$ négyzet: $4N^2$ *csúcspont*.
- Mérete *index puffer*-rel: $(N+1) \times (N+1) = N^2 + 2N + 1$ *csúcs* $(+\ 4N^2$ egész szám, *index*)
- Mikor éri meg? Azaz hogyan viszonyul egymáshoz $4N^2$ és $N^2 + 2N + 1$?

$$\begin{aligned} 4N^2 &> N^2 + 2N + 1 \\ 0 &> -3N^2 + 2N + 1 \\ N &> 1, \text{ha } N \in \mathbb{Z}^+ \end{aligned}$$

Példa – folyt.

- Ha több mint egyetlen négyzetünk van, már megéri!

Példa – folyt.

- Ha több mint egyetlen négyzetünk van, már megéri!
- Pl. ha $N = 10$
- Mérete *index puffer* nélkül: 400 csúcsot tárolunk és transzformálunk
- Mérete *index puffer*-rel: 121 csúcsot tárolunk és transzformálunk

Index puffer-ek GPU-n

- Minden videokártya, „amit még nem gyűjtenek a múzeumok” (idézet 2010-ből) támogatja az *index puffer*-eket.

Index puffer-ek GPU-n

- Minden videokártya, „amit még nem gyűjtenek a múzeumok” (idézet 2010-ből) támogatja az *index puffer*-eket.
- A csúcspontok tömbje (*vertex buffer*) nem csak pozicókat tartalmaz, hanem normálvektorokat, textúra-koordinátákat, és még sok mást.

Index puffer-ek GPU-n

- Minden videokártya, „amit még nem gyűjtenek a múzeumok” (idézet 2010-ből) támogatja az *index puffer*-eket.
- A csúcspontok tömbje (*vertex buffer*) nem csak pozicókat tartalmaz, hanem normálvektorokat, textúra-koordinátákat, és még sok mást.
- Egy hivatkozás a *vertex buffer*-ra mindezekre együtt hivatkozik.

Kocka probléma

- Figyeljünk: az index puffer csak akkor tud segíteni, ha bár különböző háromszögeknél használjuk fel ugyanazt a csúcsot, de mindegyik háromszög *ugyananannak a felületnek* a közelítése!

Kocka probléma

- Figyeljünk: az index puffer csak akkor tud segíteni, ha bár különböző háromszögeknél használjuk fel ugyanazt a csúcsot, de mindegyik háromszög *ugyanannak a felületnek* a közelítése!
- Ha egy kockát rakunk össze háromszögekből, akkor egy sarokban lévő csúcs min. három, max. hat háromszögnek a csúcsa,

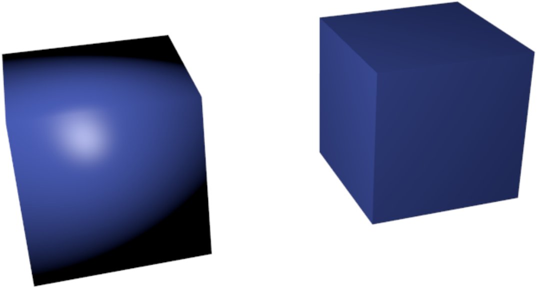
Kocka probléma

- Figyeljünk: az index puffer csak akkor tud segíteni, ha bár különböző háromszögeknél használjuk fel ugyanazt a csúcsot, de mindegyik háromszög *ugyanannak a felületnek* a közelítése!
- Ha egy kockát rakunk össze háromszögekből, akkor egy sarokban lévő csúcs min. három, max. hat háromszögnek a csúcsa, de minden egyes csúcspont három különböző felület (az ott összeérő három lap) pontja

Kocka probléma

- Figyeljünk: az index puffer csak akkor tud segíteni, ha bár különböző háromszögeknél használjuk fel ugyanazt a csúcsot, de mindegyik háromszög *ugyanannak a felületnek* a közelítése!
- Ha egy kockát rakunk össze háromszögekből, akkor egy sarokban lévő csúcs min. három, max. hat háromszögnek a csúcsa, de minden egyes csúcspont három különböző felület (az ott összeérő három lap) pontja
- Tehát bár *index puffer*-rel elég lenne csak 8 csúcsot nyilvántartani, amint megvilágítunk és felületi normálisokra lesz szükségünk baj lesz!

Kocka probléma



Kocka probléma

- A probléma: bár egyetlen térbeli pontot jelöl egy sarok, valójában három különböző felület *felületi pontja*

Kocka probléma

- A probléma: bár egyetlen térbeli pontot jelöl egy sarok, valójában három különböző felület *felületi pontja*
- Mivel a csúcspontokban pozíciókon kívül felületi tulajdonságokat is tárolunk (normális), ezért ilyenkor még index pufferrel sem spórolhatjuk ki a térbeli redundanciát!

Kocka probléma

- A probléma: bár egyetlen térbeli pontot jelöl egy sarok, valójában három különböző felület *felületi pontja*
- Mivel a csúcspontokban pozíciókon kívül felületi tulajdonságokat is tárolunk (normális), ezért ilyenkor még index pufferrel sem spórolhatjuk ki a térbeli redundanciát!
- Oldalanként külön meg kell adni a csúcsokat, ugyanolyan pozíció, de a három oldalnak megfelelően három különböző normális segítségével: összesen tehát 3×8 csúcs kerül az *vertex buffer*-ba

Az *index pufferes* tárolás elemzése

- *Tárolás:* ált. hatékony.

Az *index pufferes* tárolás elemzése

- *Tárolás:* ált. hatékony.
- *Transzformálás:* hatékony.

Az *index pufferes* tárolás elemzése

- *Tárolás:* ált. hatékony.
- *Transzformálás:* hatékony.
- *Lekérdezések:* közös csúcsokat már tudunk, de igazából még mindig fogalmunk sincs.

Tartalom

Szomszédsági viszonyok

- Néha kellenek a szomszédok, pl. felület-felosztásoknál, degenerált primitívek kiszűrésekor, egyes felhasználói inputok kezelésekor stb.

Szomszédsági viszonyok

- Néha kellenek a szomszédok, pl. felület-felosztásoknál, degenerált primitívek kiszűrésekor, egyes felhasználói inputok kezelésekor stb.
- Ismertek a csúcsok $\Rightarrow$ számítható mi, minek a szomszédja!

Szomszédsági viszonyok

- Néha kellenek a szomszédok, pl. felület-felosztásoknál, degenerált primitívek kiszűrésekor, egyes felhasználói inputok kezelésekor stb.
- Ismertek a csúcsok $\Rightarrow$ számítható mi, minek a szomszédja!
- Egy csúcsban tetszőleges számú poligon találkozhat $\Rightarrow$ dinamikus adatszerkezet kéne

Szomszédsági viszonyok

- Néha kellenek a szomszédok, pl. felület-felosztásoknál, degenerált primitívek kiszűrésekor, egyes felhasználói inputok kezelésekor stb.
- Ismertek a csúcsok $\Rightarrow$ számítható mi, minek a szomszédja!
- Egy csúcsban tetszőleges számú poligon találkozhat $\Rightarrow$ dinamikus adatszerkezet kéne
- Jobb megoldás: Szárnyas-él (*winged-edge*) adatszerkezet!

Szárnyas-él adatszerkezet

- Az alakzatokat határukkal leíró (*B-rep*) (boundary representation) reprezentáció egyik gyakran használt topológiatároló adatszerkezete manifold poliéderek esetén

Szárnyas-él adatszerkezet

- Az alakzatokat határukkal leíró (*B-rep*) (boundary representation) reprezentáció egyik gyakran használt topológiatároló adatszerkezete manifold poliéderek esetén
- Tárolás során *csúcsokat*, *éleket* és *lapokat* különböztet meg

Szárnyas-él adatszerkezet

- Az alakzatokat határukkal leíró (*B-rep*) (boundary representation) reprezentáció egyik gyakran használt topológiatároló adatszerkezete manifold poliéderek esetén
- Tárolás során *csúcsokat*, *éleket* és *lapokat* különböztet meg
- Az élek szempontjából tároljuk a felületet

Szárnyas-él adatszerkezet

- Az alakzatokat határukkal leíró (*B-rep*) (boundary representation) reprezentáció egyik gyakran használt topológiatároló adatszerkezete manifold poliéderek esetén
- Tárolás során *csúcsokat*, *éleket* és *lapokat* különböztet meg
- Az élek szempontjából tároljuk a felületet
- Minden élhez fix számú adat tartozik

Szárnyas-él adatszerkezet

- Az alakzatokat határukkal leíró (*B-rep*) (boundary representation) reprezentáció egyik gyakran használt topológiatároló adatszerkezete manifold poliéderek esetén
- Tárolás során *csúcsokat*, *éleket* és *lapokat* különböztet meg
- Az élek szempontjából tároljuk a felületet
- Minden élhez fix számú adat tartozik
- Segítségével pl. gyorsan körbe lehet járni egy poligon éleit, közben megkapva minden szomszédot

Szárnyas-él adatszerkezet

- Minden lapot egy élsorozat határol – minden laphoz tároljunk egy, az élsorozatához tartozó tetszőleges élre mutató pointert

Szárnyas-él adatszerkezet

- Minden lapot egy élsorozat határol – minden laphoz tároljunk egy, az élsorozatához tartozó tetszőleges élre mutató pointert
- A csúcspontok élekhez illeszkednek (vagy belőle indul ki, vagy ő a célja) – tároljuk valamelyiket a csúcsokhoz!

Egyetlen él adatai

- Egy él két csúcsot köt össze – tároljuk ezeket az élben

Egyetlen él adatai

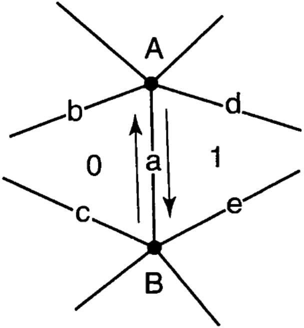
- Egy él két csúcsot köt össze – tároljuk ezeket az élben
- Egy él legfeljebb két laphoz tartozhat – az egyik a baloldali, a másik a jobboldali lap lesz, ezekre mutató pointereket (vagy indexeket) tárolunk

Egyetlen él adatai

- Egy él két csúcsot köt össze – tároljuk ezeket az élben
- Egy él legfeljebb két laphoz tartozhat – az egyik a baloldali, a másik a jobboldali lap lesz, ezekre mutató pointereket (vagy indexeket) tárolunk
- A fenti két lapon egyúttal egy-egy élsorozat (az adott lapot alkotó élsorozat) része is az adott él – mindkét élsorozatban tároljuk a rákövetkezőjét és megelőzőjét *az adott élnek az adott lap bejárási irányának megfelelően* (!)

Egyetlen él adatai

él	csúcs		lap		balra		jobbra	
	start	vég	bal	jobb	előző	köv.	előző	köv.
a	B	A	0	1	c	b	d	e



Egyéb táblázatok

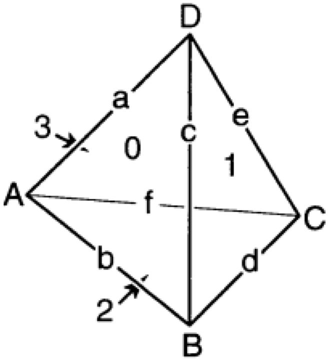
Csúcsok táblája

- csúcs ID
- csúcsból induló él

Lapok táblája

- lap ID
- lap egy éle

Példa: tetraéder



edge	vertex 1	vertex 2	face left	face right	pred left	succ left	pred right	succ right
a	A	D	3	0	f	e	c	b
b	A	B	0	2	a	c	d	f
c	B	D	0	1	b	a	e	d
d	B	C	1	2	c	e	f	b
e	C	D	1	3	d	c	a	f
f	C	A	3	2	e	a	b	d

vertex	edge
A	a
B	d
C	d
D	e

face	edge
0	a
1	c
2	d
3	a

Shirley, Fundamentals of Computer Graphics

Pl.: Egy lap összes szomszéd lapjának felsorolása

```
def allNeighbours(face, edges, vertices, faces):
  startEdge = faces[face]
  edge = startEdge
  do:
    if edges[edge].faceLeft == face:
      print edges[edge].faceRight
      edge = edges[edge].succLeft
    else:
      print edges[edge].faceLeft
      edge = edges[edge].succRight
  while edge != startEdge
```

Pl.: Egy lap összes szomszéd lapjának felsorolása

- Azaz: induljunk el az adott lap reprezentáns éléből (amit tárolunk a laphoz)

Pl.: Egy lap összes szomszéd lapjának felsorolása

- Azaz: induljunk el az adott lap reprezentáns éléből (amit tárolunk a laphoz)
- Ha az éppen vizsgált él bal oldali lapja az adott lap: kiírjuk a jobb oldali lapot és továbblépünk a bal lap következő élére

Pl.: Egy lap összes szomszéd lapjának felsorolása

- Azaz: induljunk el az adott lap reprezentáns éléből (amit tárolunk a laphoz)
- Ha az éppen vizsgált él bal oldali lapja az adott lap: kiírjuk a jobb oldali lapot és továbblépünk a bal lap következő élére
- Különben a jobb oldali lap az adott lap, a bal oldalit írjuk ki és a jobb rákövetkező élre lépünk

Pl.: Egy lap összes szomszéd lapjának felsorolása

- Azaz: induljunk el az adott lap reprezentáns éléből (amit tárolunk a laphoz)
- Ha az éppen vizsgált él bal oldali lapja az adott lap: kiírjuk a jobb oldali lapot és továbblépünk a bal lap következő élére
- Különben a jobb oldali lap az adott lap, a bal oldalit írjuk ki és a jobb rákövetkező élre lépünk
- Az iteráció érjen véget, amint visszaérünk az adott lap reprezentáns élébe

Pl.: Egy adott csúcsot tartalmazó összes lap felsorolása

```
def allFaces(vertex, edges, vertices, faces):
  startEdge = vertices[vertex]
  edge = startEdge
  do:
    if edges[edge].vertStart == vertex:
      print edges[edge].faceLeft
      edge = edges[edge].predLeft
    else:
      print edges[edge].faceRight
      edge = edges[edge].predRight
  while edge != startEdge
```

Fél-él adatszerkezet

- A winged-edge élét vegyük szét két fél-élre!

Fél-él adatszerkezet

- A winged-edge élét vegyük szét két fél-élre!
- $\rightarrow$ lényegében az élek lapra vett vetületével dolgozunk!

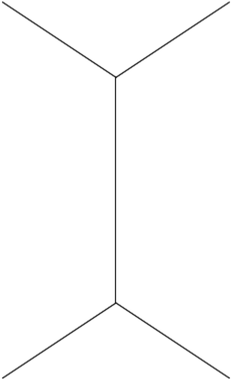
Fél-él adatszerkezet

- A winged-edge élét vegyük szét két fél-élre!
- $\rightarrow$ lényegében az élek lapra vett vetületével dolgozunk!
- A fél-élhez csak egy lap tartozhat + meg kell jegyeznünk a testvér fél-élét (az adott él másik oldali lapra vett vetületét)

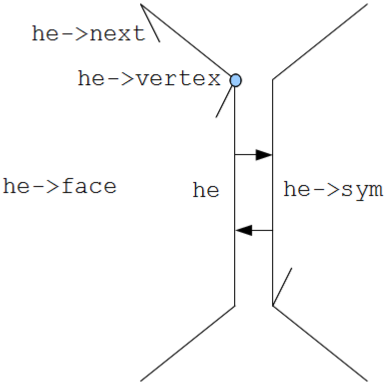
Fél-él adatszerkezet

- A winged-edge élét vegyük szét két fél-élre!
- $\rightarrow$ lényegében az élek lapra vett vetületével dolgozunk!
- A fél-élhez csak egy lap tartozhat + meg kell jegyeznünk a testvér fél-élét (az adott él másik oldali lapra vett vetületét)
- A reprezentáció központi eleme a fél-él

Half-edge



Half-edge



Fél-él adatszerkezet

- Szigorú értelemben véve egy fél-élhez pontosan egy csúcs, él és lap tartozik (de gyakorlatban ennél többet tárolni hasznos lehet)

Fél-él adatszerkezet

- Szigorú értelemben véve egy fél-élhez pontosan egy csúcs, él és lap tartozik (de gyakorlatban ennél többet tárolni hasznos lehet)
- A következőt tároljuk egy fél-élben: az fél-él cél csúcspontja (vertex), a fél-él testvére (sym), a fél-él lapja (face) és a lapot körbefogó fél-élsorozatban a rákövetkezője (next)

Fél-él adatszerkezet

- Szigorú értelemben véve egy fél-élhez pontosan egy csúcs, él és lap tartozik (de gyakorlatban ennél többet tárolni hasznos lehet)
- A következőt tároljuk egy fél-élben: az fél-él cél csúcspontja (vertex), a fél-él testvére (sym), a fél-él lapja (face) és a lapot körbefogó fél-élsorozatban a rákövetkezője (next)
- A lapokhoz egy tetszőleges alkotó fél-él pointerét jegyezzük fel

Fél-él adatszerkezet

- Szigorú értelemben véve egy fél-élhez pontosan egy csúcs, él és lap tartozik (de gyakorlatban ennél többet tárolni hasznos lehet)
- A következőt tároljuk egy fél-élben: az fél-él cél csúcspontja (vertex), a fél-él testvére (sym), a fél-él lapja (face) és a lapot körbefogó fél-élsorozatban a rákövetkezője (next)
- A lapokhoz egy tetszőleges alkotó fél-él pointerét jegyezzük fel
- A csúcspontokhoz egy befutó fél-élt

Fél-él adatszerkezet

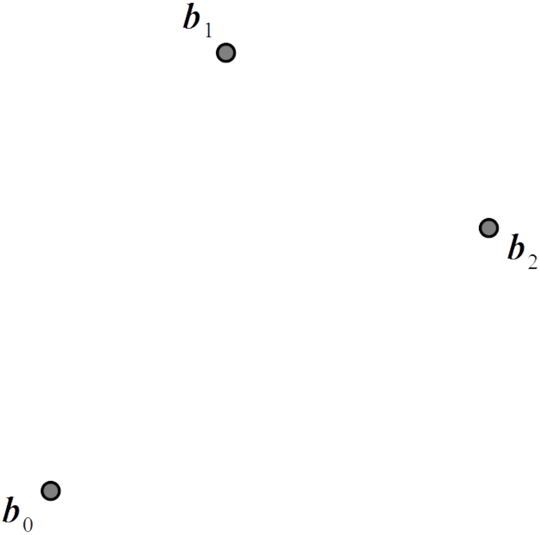
- Szigorú értelemben véve egy fél-élhez pontosan egy csúcs, él és lap tartozik (de gyakorlatban ennél többet tárolni hasznos lehet)
- A következőt tároljuk egy fél-élben: az fél-él cél csúcspontja (vertex), a fél-él testvére (sym), a fél-él lapja (face) és a lapot körbefogó fél-élsorozatban a rákövetkezője (next)
- A lapokhoz egy tetszőleges alkotó fél-él pointerét jegyezzük fel
- A csúcspontokhoz egy befutó fél-élt
- $\mathrm{HE}\to\mathrm{sym}\to\mathrm{sym} = \mathrm{HE}$, $\mathrm{HE}\to\mathrm{next}\to\mathrm{sym}\to\mathrm{vertex} = \mathrm{HE}\to\mathrm{vertex}$ stb.

Pl.: Lap éleinek körbejárása

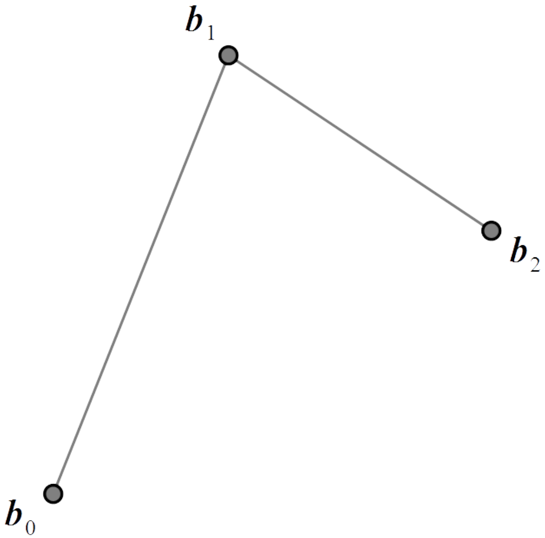
```
def faceLoop(HE):
  loop = HE
  do:
    print loop
    loop = loop->next
  while loop != HE
```

Tartalom

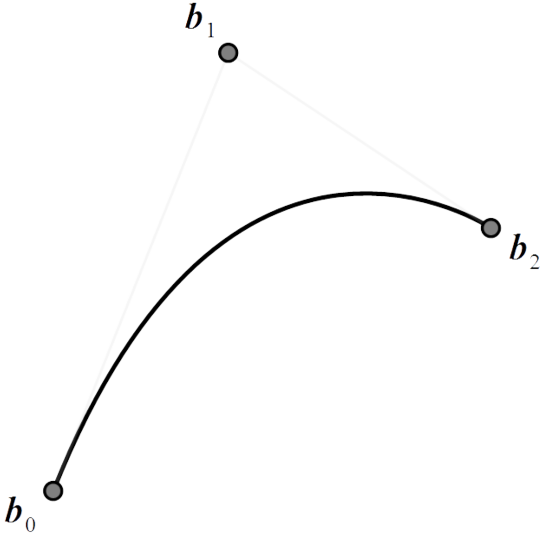
Reprezentáció



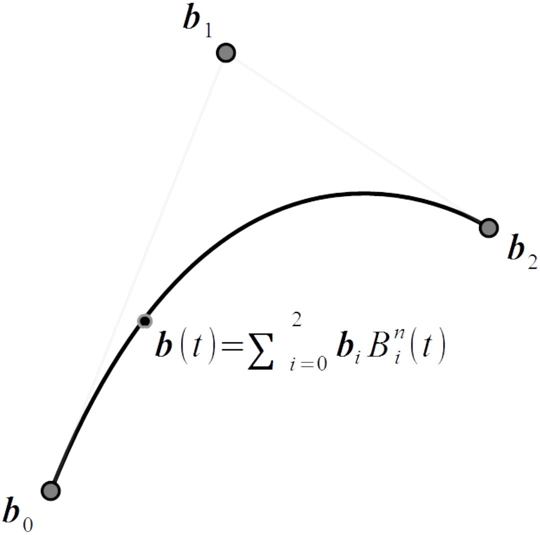
Reprezentáció



Reprezentáció



Reprezentáció



Görbék megadása

- Három módot láttunk már görbék reprezentációjára:
  - explicit: $y = f(x)$
  - parametrikus: $\mathbf{p}(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix}, t \in \mathbb{R}$
  - implicit: $x^2 + y^2 - 9 = 0$

Görbék megadása

- Most azt vizsgáljuk meg, hogy milyen adatokat kell eltárolni, hogy egy tetszőleges (ún. *szabadformájú*) görbét egyértelműen reprezentáljunk

Görbék megadása

- Most azt vizsgáljuk meg, hogy milyen adatokat kell eltárolni, hogy egy tetszőleges (ún. *szabadformájú*) görbét egyértelműen reprezentáljunk
- Ezt most a paramterikus megadási mód segítségével nézzük meg

Tartalom

Lineáris interpoláció

- Legyen adott két pont, $\mathbf{a}, \mathbf{b} \in \mathbb{E}^3$. A két ponton átmenő egyenes parametrikus egyenlete:

$$\mathbf{p}(t) = (1 - t)\mathbf{a} + t\mathbf{b},$$

  ahol $t \in \mathbb{R}$.
- Ha $t \in [0, 1]$, akkor az $\mathbf{a}, \mathbf{b}$ pontokat összekötő egyenes szakaszt kapjuk.

Lineáris interpoláció

- Az egyenes szakaszt egyértelműen azonosítja a szakasz két végpontja, **a** és **b**

Lineáris interpoláció

- Az egyenes szakaszt egyértelműen azonosítja a szakasz két végpontja, **a** és **b**
- Ez a legegyszerűbb „görbe” a két pont között

Lineáris interpoláció

- Az egyenes szakaszt egyértelműen azonosítja a szakasz két végpontja, **a** és **b**
- Ez a legegyszerűbb „görbe” a két pont között
- Hogyan lesz ebből „szép” görbe?

Lineáris interpoláció

- Az egyenes szakaszt egyértelműen azonosítja a szakasz két végpontja, **a** és **b**
- Ez a legegyszerűbb „görbe” a két pont között
- Hogyan lesz ebből „szép” görbe?
- „Szép” ≈ valami szépen, folytonosan változó

Szépség – parametrikus folytonosság

- C^0: a görbében/felületben nincsenek lyukak, nem szakad meg sehol

Szépség – parametrikus folytonosság

- C^0: a görbében/felületben nincsenek lyukak, nem szakad meg sehol
- C^1: a derivált is folytonosan változik (DE: a derivált *paraméterezéstől függ*!)

Szépség – parametrikus folytonosság

- C^0: a görbében/felületben nincsenek lyukak, nem szakad meg sehol
- C^1: a derivált is folytonosan változik (DE: a derivált *paraméterezéstől függ*!)
- C^2: a görbe/felület második deriváltjai is folytonosan változnak

Szépség – deriváltak

- Ne feledjük, a parametrikus görbe $\mathbf{p}(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix}$ alakú

Szépség – deriváltak

- Ne feledjük, a parametrikus görbe $\mathbf{p}(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix}$ alakú
- Tehát a deriváltak $\mathbf{p}'(t) = \begin{bmatrix} x'(t) \\ y'(t) \end{bmatrix}$, $\mathbf{p}''(t) = \begin{bmatrix} x''(t) \\ y''(t) \end{bmatrix}$ stb. alakúak

Szépség – deriváltak

- Ne feledjük, a parametrikus görbe $\mathbf{p}(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix}$ alakú
- Tehát a deriváltak $\mathbf{p}'(t) = \begin{bmatrix} x'(t) \\ y'(t) \end{bmatrix}$, $\mathbf{p}''(t) = \begin{bmatrix} x''(t) \\ y''(t) \end{bmatrix}$ stb. alakúak
- Példa: mi a deriváltja a $\mathbf{p}(t) = \begin{bmatrix} t^2 + t \\ t^3 \end{bmatrix}$ görbének a $t = 0$ és $t = 1$ pontokban?

Szépség – deriváltak

- Ne feledjük, a parametrikus görbe $\mathbf{p}(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix}$ alakú
- Tehát a deriváltak $\mathbf{p}'(t) = \begin{bmatrix} x'(t) \\ y'(t) \end{bmatrix}$, $\mathbf{p}''(t) = \begin{bmatrix} x''(t) \\ y''(t) \end{bmatrix}$ stb. alakúak
- Példa: mi a deriváltja a $\mathbf{p}(t) = \begin{bmatrix} t^2 + t \\ t^3 \end{bmatrix}$ görbének a $t = 0$ és $t = 1$ pontokban?
- Példa: mi lesz az első és második deriváltgörbéje (hodográfja) a $\mathbf{p}(t) = (1 - t)\mathbf{a} + t\mathbf{b}$ görbének?

Parametrikus folytonosság

Legyen adott két parametrikus görbe, $\mathbf{r}(t), \mathbf{s}(t) : [0, 1] \to \mathbb{E}^3$, amelyeknek van egy közös pontja, azaz pl. $\mathbf{r}(1) = \mathbf{s}(0) = \mathbf{p}$. Ekkor a két görbe a $\mathbf{p}$-ben

- $C^0 \Leftrightarrow \mathbf{r}(1) = \mathbf{s}(0)$

Parametrikus folytonosság

Legyen adott két parametrikus görbe, $\mathbf{r}(t), \mathbf{s}(t) : [0, 1] \to \mathbb{E}^3$, amelyeknek van egy közös pontja, azaz pl. $\mathbf{r}(1) = \mathbf{s}(0) = \mathbf{p}$. Ekkor a két görbe a **p**-ben

- $C^0 \Leftrightarrow \mathbf{r}(1) = \mathbf{s}(0)$
- $C^1 \Leftrightarrow C^0 \wedge \mathbf{r}'(1) = \mathbf{s}'(0)$

Parametrikus folytonosság

Legyen adott két parametrikus görbe, $\mathbf{r}(t), \mathbf{s}(t) : [0,1] \to \mathbb{E}^3$, amelyeknek van egy közös pontja, azaz pl. $\mathbf{r}(1) = \mathbf{s}(0) = \mathbf{p}$. Ekkor a két görbe a $\mathbf{p}$-ben

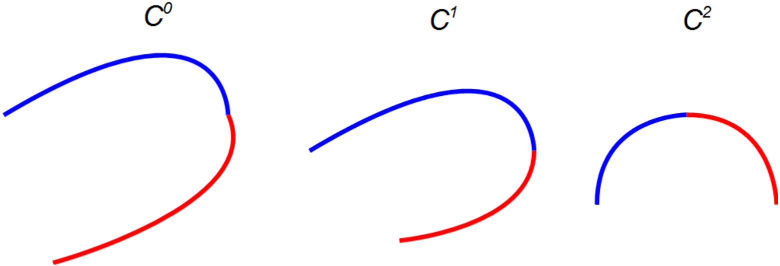
- $C^0 \Leftrightarrow \mathbf{r}(1) = \mathbf{s}(0)$
- $C^1 \Leftrightarrow C^0 \wedge \mathbf{r}'(1) = \mathbf{s}'(0)$
- $C^2 \Leftrightarrow C^1 \wedge \mathbf{r}''(1) = \mathbf{s}''(0)$

Parametrikus folytonosság

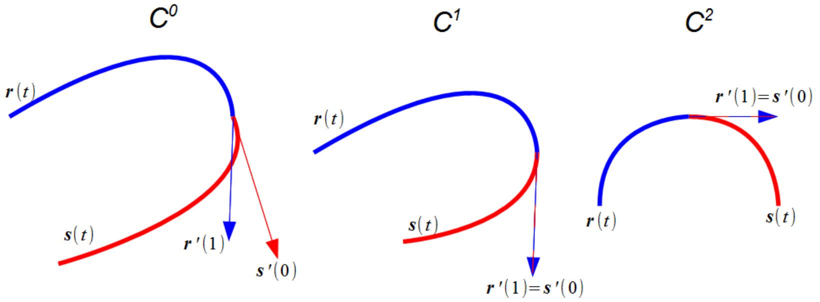
Legyen adott két parametrikus görbe, $\mathbf{r}(t), \mathbf{s}(t) : [0,1] \to \mathbb{E}^3$, amelyeknek van egy közös pontja, azaz pl. $\mathbf{r}(1) = \mathbf{s}(0) = \mathbf{p}$. Ekkor a két görbe a **p**-ben

- $C^0 \Leftrightarrow \mathbf{r}(1) = \mathbf{s}(0)$
- $C^1 \Leftrightarrow C^0 \wedge \mathbf{r}'(1) = \mathbf{s}'(0)$
- $C^2 \Leftrightarrow C^1 \wedge \mathbf{r}''(1) = \mathbf{s}''(0)$
- $C^n \Leftrightarrow C^{n-1} \wedge \mathbf{r}^{(n)}(1) = \mathbf{s}^{(n)}(0)$

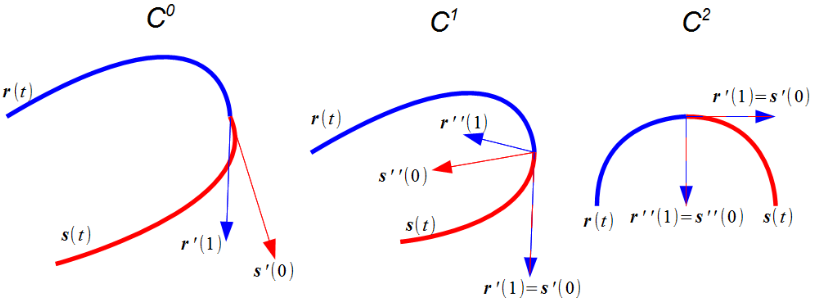
Parametrikus folytonosság



Parametrikus folytonosság



Parametrikus folytonosság



Törött vonal

- Adottak $\mathbf{p}_i \in \mathbb{E}^3, i = 0, ..., n$ pontok és mindegyik $\mathbf{p}_i, \mathbf{p}_{i+1}, i = 0, ..., n-1$ pontpárt kössünk össze egy szakasszal!

Törött vonal

- Adottak $\mathbf{p}_i \in \mathbb{E}^3, i = 0, ..., n$ pontok és mindegyik $\mathbf{p}_i, \mathbf{p}_{i+1}, i = 0, ..., n-1$ pontpárt kössünk össze egy szakasszal!
- Legyenek $t_0 \leq t_1 \leq ... \leq t_n \in \mathbb{R}$ paraméterek a $\mathbf{p}_i$ pontokhoz hozzárendelve

Törött vonal

- Adottak $\mathbf{p}_i \in \mathbb{E}^3, i = 0, ..., n$ pontok és mindegyik $\mathbf{p}_i, \mathbf{p}_{i+1}, i = 0, ..., n-1$ pontpárt kössünk össze egy szakasszal!
- Legyenek $t_0 \leq t_1 \leq ... \leq t_n \in \mathbb{R}$ paraméterek a $\mathbf{p}_i$ pontokhoz hozzárendelve
- Ekkor az eredmény törött vonal felírható egyetlen paraméterrel is: ha a $t \in [t_0, t_n]$ paraméter aktuális értékére $t \in [t_i, t_{i+1}]$ igaz, akkor a hozzátartozó pont

$$\frac{t_{i+1} - t}{t_{i+1} - t_i}\mathbf{p}_i + \frac{t - t_i}{t_{i+1} - t_i}\mathbf{p}_{i+1}$$

Törött vonal

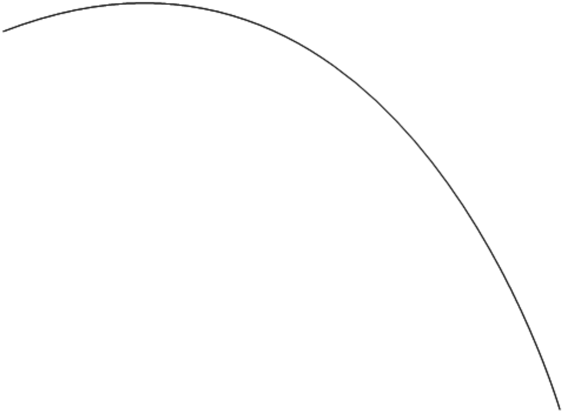
- Adottak $\mathbf{p}_i \in \mathbb{E}^3, i = 0, ..., n$ pontok és mindegyik $\mathbf{p}_i, \mathbf{p}_{i+1}, i = 0, ..., n-1$ pontpárt kössünk össze egy szakasszal!
- Legyenek $t_0 \leq t_1 \leq ... \leq t_n \in \mathbb{R}$ paraméterek a $\mathbf{p}_i$ pontokhoz hozzárendelve
- Ekkor az eredmény törött vonal felírható egyetlen paraméterrel is: ha a $t \in [t_0, t_n]$ paraméter aktuális értékére $t \in [t_i, t_{i+1}]$ igaz, akkor a hozzátartozó pont

$$\frac{t_{i+1} - t}{t_{i+1} - t_i}\mathbf{p}_i + \frac{t - t_i}{t_{i+1} - t_i}\mathbf{p}_{i+1}$$

- Ez egy interpoláló megközelítés, azaz a reprezentációt képező ponthalmaz összes elemén áthalad a reprezentált görbe

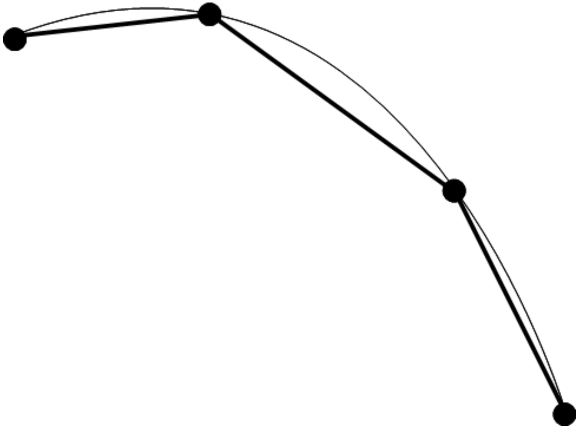
Törött vonal

Ha egy „sima” görbét szeretnénk ábrázolni, akkor szakaszokkal kell közelítenünk (lényegében: tesszellálni)



Törött vonal

Ha egy „sima” görbét szeretnénk ábrázolni, akkor szakaszokkal kell közelítenünk (lényegében: tesszellálni)



Tartalom

Polinomiális görbék

- Ha C^0-nál magasabb fokú **garantált** (!) folytonosságot akarunk, akkor próbálkozhatunk polinomokkal

Polinomiális görbék

- Ha C^0-nál magasabb fokú **garantált** (!) folytonosságot akarunk, akkor próbálkozhatunk polinomokkal
- A $\mathbf{p}_0, \ldots, \mathbf{p}_n$ pontokra illeszthető egy n-edfokú polinom

Polinomiális görbék

- Ha C^0-nál magasabb fokú **garantált** (!) folytonosságot akarunk, akkor próbálkozhatunk polinomokkal
- A $\mathbf{p}_0, \dots, \mathbf{p}_n$ pontokra illeszthető egy n-edfokú polinom
- A jól ismert hatványbázisban ez $\mathbf{p}(t) = \sum_{i=0}^{n} \mathbf{a}_i t^i, \ t \in \mathbb{R}$ alakú (pl.: $\mathbf{p}(t) = \begin{bmatrix}1\\2\end{bmatrix} t^2 + \begin{bmatrix}0\\1\end{bmatrix} t + \begin{bmatrix}1\\1\end{bmatrix}$)

Polinomiális görbék

- Ha C^0-nál magasabb fokú **garantált** (!) folytonosságot akarunk, akkor próbálkozhatunk polinomokkal
- A $\mathbf{p}_0, \dots, \mathbf{p}_n$ pontokra illeszthető egy n-edfokú polinom
- A jól ismert hatványbázisban ez $\mathbf{p}(t) = \sum_{i=0}^{n} \mathbf{a}_i t^i$, $t \in \mathbb{R}$ alakú (pl.: $\mathbf{p}(t) = \begin{bmatrix}1\\2\end{bmatrix} t^2 + \begin{bmatrix}0\\1\end{bmatrix} t + \begin{bmatrix}1\\1\end{bmatrix}$)
- De mi lenne a geometriai értelmezése az $\mathbf{a}_i \in \mathbb{R}^3$ együtthatóknak? Mit ábrázolnak?

Polinomiális görbék

- Ha C^0-nál magasabb fokú **garantált** (!) folytonosságot akarunk, akkor próbálkozhatunk polinomokkal
- A $\mathbf{p}_0, ..., \mathbf{p}_n$ pontokra illeszthető egy n-edfokú polinom
- A jól ismert hatványbázisban ez $\mathbf{p}(t) = \sum_{i=0}^{n} \mathbf{a}_i t^i$, $t \in \mathbb{R}$ alakú (pl.: $\mathbf{p}(t) = \begin{bmatrix}1\\2\end{bmatrix} t^2 + \begin{bmatrix}0\\1\end{bmatrix} t + \begin{bmatrix}1\\1\end{bmatrix}$)
- De mi lenne a geometriai értelmezése az $\mathbf{a}_i \in \mathbb{R}^3$ együtthatóknak? Mit ábrázolnak?
- $\rightarrow$ másik bázist keressünk inkább, ahol a reprezentációt képező elemeknek szemléletesebb jelentése van

Polinomiális görbék

- Ha C^0-nál magasabb fokú **garantált** (!) folytonosságot akarunk, akkor próbálkozhatunk polinomokkal
- A $\mathbf{p}_0, \ldots, \mathbf{p}_n$ pontokra illeszthető egy n-edfokú polinom
- A jól ismert hatványbázisban ez $\mathbf{p}(t) = \sum_{i=0}^{n} \mathbf{a}_i t^i,\ t \in \mathbb{R}$ alakú (pl.: $\mathbf{p}(t) = \begin{bmatrix}1\\2\end{bmatrix} t^2 + \begin{bmatrix}0\\1\end{bmatrix} t + \begin{bmatrix}1\\1\end{bmatrix}$)
- De mi lenne a geometriai értelmezése az $\mathbf{a}_i \in \mathbb{R}^3$ együtthatóknak? Mit ábrázolnak?
- $\rightarrow$ másik bázist keressünk inkább, ahol a reprezentációt képező elemeknek szemléletesebb jelentése van
- De előtte oldjuk még meg a feladatot!

Polinomiális görbék

- Legyenek adottak $\mathbf{p}_0, \ldots, \mathbf{p}_n \in \mathbb{E}^3$ pontok és $t_0 < t_1 < .. < t_n \in \mathbb{R}$ paraméterértékek

Polinomiális görbék

- Legyenek adottak $\mathbf{p}_0, \ldots, \mathbf{p}_n \in \mathbb{E}^3$ pontok és $t_0 < t_1 < .. < t_n \in \mathbb{R}$ paraméterértékek
- Keressük azt az n-edfokú $\mathbf{p}(t) = \sum_{i=0}^{n} \mathbf{a}_i t^i, \;\; t \in \mathbb{R}$ parametrikus görbét, amely az adott pontokat interpolálja az előírt paraméterértékeknél, azaz amelyre

$$\mathbf{p}(t_i) = \mathbf{p}_i, \quad i = 0, 1, .., n$$

Polinomiális görbék

- Megoldandó tehát az

$$\underbrace{\begin{bmatrix} 1 & t_0 & t_0^2 & .. & t_0^n \\ 1 & t_1 & t_1^2 & .. & t_1^n \\ .. & .. & .. & .. & .. \\ 1 & t_n & t_n^2 & .. & t_n^n \end{bmatrix}}_{V} \underbrace{\begin{bmatrix} \mathbf{a}_0 \\ \mathbf{a}_1 \\ .. \\ \mathbf{a}_n \end{bmatrix}}_{\mathbf{a}} = \underbrace{\begin{bmatrix} \mathbf{p}_0 \\ .. \\ \mathbf{p}_n \end{bmatrix}}_{\mathbf{b}}$$

lineáris egyenletrendszer, az ismeretlen $\mathbf{a}_0, .., \mathbf{a}_n \in \mathbb{R}^3$ együtthatókra

Polinomiális görbék

- Megoldandó tehát az

$$\underbrace{\begin{bmatrix} 1 & t_0 & t_0^2 & .. & t_0^n \\ 1 & t_1 & t_1^2 & .. & t_1^n \\ .. & .. & .. & .. & .. \\ 1 & t_n & t_n^2 & .. & t_n^n \end{bmatrix}}_{V} \underbrace{\begin{bmatrix} \mathbf{a}_0 \\ \mathbf{a}_1 \\ .. \\ \mathbf{a}_n \end{bmatrix}}_{\mathbf{a}} = \underbrace{\begin{bmatrix} \mathbf{p}_0 \\ .. \\ \mathbf{p}_n \end{bmatrix}}_{\mathbf{b}}$$

  lineáris egyenletrendszer, az ismeretlen $\mathbf{a}_0, .., \mathbf{a}_n \in \mathbb{R}^3$ együtthatókra
- Ha $det(V) \neq 0$, akkor van megoldás

Polinomiális görbék

- Megoldandó tehát az

$$\underbrace{\begin{bmatrix} 1 & t_0 & t_0^2 & .. & t_0^n \\ 1 & t_1 & t_1^2 & .. & t_1^n \\ .. & .. & .. & .. & .. \\ 1 & t_n & t_n^2 & .. & t_n^n \end{bmatrix}}_{V} \underbrace{\begin{bmatrix} \mathbf{a}_0 \\ \mathbf{a}_1 \\ .. \\ \mathbf{a}_n \end{bmatrix}}_{\mathbf{a}} = \underbrace{\begin{bmatrix} \mathbf{p}_0 \\ .. \\ \mathbf{p}_n \end{bmatrix}}_{\mathbf{b}}$$

  lineáris egyenletrendszer, az ismeretlen $\mathbf{a}_0, .., \mathbf{a}_n \in \mathbb{R}^3$ együtthatókra

- Ha $det(V) \neq 0$, akkor van megoldás
- De vegyük észre: V egy Vandermonde mátrix $\rightarrow$ determinánsa nem nulla (mivel nincs $i \neq j$, hogy $t_i = t_j$)

Polinomiális görbék

- Megoldandó tehát az

$$\underbrace{\begin{bmatrix} 1 & t_0 & t_0^2 & .. & t_0^n \\ 1 & t_1 & t_1^2 & .. & t_1^n \\ .. & .. & .. & .. & .. \\ 1 & t_n & t_n^2 & .. & t_n^n \end{bmatrix}}_{V} \underbrace{\begin{bmatrix} \mathbf{a}_0 \\ \mathbf{a}_1 \\ .. \\ \mathbf{a}_n \end{bmatrix}}_{\mathbf{a}} = \underbrace{\begin{bmatrix} \mathbf{p}_0 \\ .. \\ \mathbf{p}_n \end{bmatrix}}_{\mathbf{b}}$$

  lineáris egyenletrendszer, az ismeretlen $\mathbf{a}_0, .., \mathbf{a}_n \in \mathbb{R}^3$ együtthatókra
- Ha $det(V) \neq 0$, akkor van megoldás
- De vegyük észre: V egy Vandermonde mátrix $\rightarrow$ determinánsa nem nulla (mivel nincs $i \neq j$, hogy $t_i = t_j$)
- A keresett együtthatók tehát $\mathbf{a} = V^{-1}\mathbf{b}$

Polinomiális görbék – parabola példa

- Tehát ha például egy parabolával szeretnénk a $t = 0, 1, 2$ pontokban interpolálni a $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2$ pontokat, akkor

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{a}_0 \\ \mathbf{a}_1 \\ \mathbf{a}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \end{bmatrix}$$

egyenletet kell megoldanunk

Polinomiális görbék – parabola példa

- Tehát ha például egy parabolával szeretnénk a $t = 0, 1, 2$ pontokban interpolálni a $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2$ pontokat, akkor

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{a}_0 \\ \mathbf{a}_1 \\ \mathbf{a}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \end{bmatrix}$$

  egyenletet kell megoldanunk
- Azaz a keresett együtthatók

$$\begin{bmatrix} \mathbf{a}_0 \\ \mathbf{a}_1 \\ \mathbf{a}_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{3}{2} & 2 & -\frac{1}{2} \\ \frac{1}{2} & -1 & \frac{1}{2} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \end{bmatrix}$$

Polinomiális görbék – parabola példa

- Tehát ha például egy parabolával szeretnénk a $t = 0, 1, 2$ pontokban interpolálni a $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2$ pontokat, akkor

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{a}_0 \\ \mathbf{a}_1 \\ \mathbf{a}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \end{bmatrix}$$

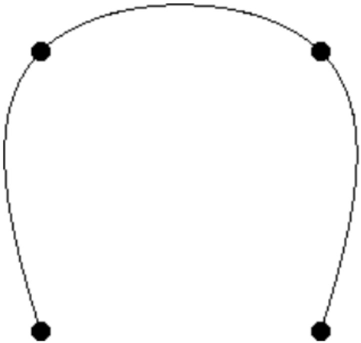
  egyenletet kell megoldanunk
- Azaz a keresett együtthatók

$$\begin{bmatrix} \mathbf{a}_0 \\ \mathbf{a}_1 \\ \mathbf{a}_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{3}{2} & 2 & -\frac{1}{2} \\ \frac{1}{2} & -1 & \frac{1}{2} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \end{bmatrix}$$

- A parabola pedig $\mathbf{p}(t) = \mathbf{a}_2 \cdot t^2 + \mathbf{a}_1 \cdot t + \mathbf{a}_0$

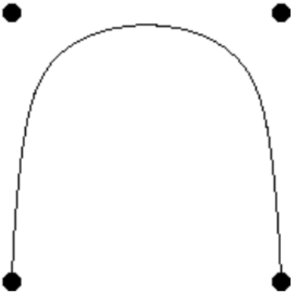
Interpoláció vagy approximáció?

Interpoláció: a görbének át kell haladnia a vezérlőpontokon



Interpoláció vagy approximáció?

Approximáció: a görbének csak közelítenie kell a vezérlőpontokat



Sok vezérlőpont esete

- A vezérlőpontok számával együtt nő a fokszám $\rightarrow$ egy idő után lehet meg kell elégednünk az approximációval, de az sem lesz jó

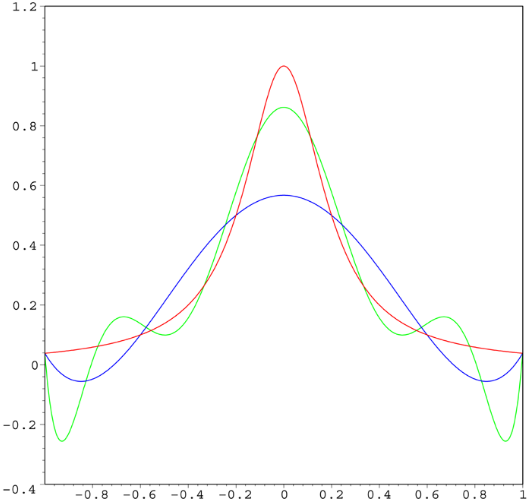
Sok vezérlőpont esete

- A vezérlőpontok számával együtt nő a fokszám $\rightarrow$ egy idő után lehet meg kell elégednünk az approximációval, de az sem lesz jó
- A magas fokszámú polinomok erősen „hullámozhatnak”

Sok vezérlőpont esete

- A vezérlőpontok számával együtt nő a fokszám $\rightarrow$ egy idő után lehet meg kell elégednünk az approximációval, de az sem lesz jó
- A magas fokszámú polinomok erősen „hullámozhatnak”
- Runge-problémája: az $f(x) = \frac{1}{1+25x^2}$ függvény (piros) közelítése során ötödfokú (kék) és kilencedfokú (zöld) polinomokkal

Sok vezérlőpont esete



Spline görbék

- Ne egyetlen polinommal próbáljuk interpolálni vagy approximálni az adatpontjainkat (vezérlőpontjainkat)!

Spline görbék

- Ne egyetlen polinommal próbáljuk interpolálni vagy approximálni az adatpontjainkat (vezérlőpontjainkat)!
- Használjunk *összetett görbét*, amely alacsonyabb fokszámú szegmensekből áll

Spline görbék

- Ne egyetlen polinommal próbáljuk interpolálni vagy approximálni az adatpontjainkat (vezérlőpontjainkat)!
- Használjunk *összetett görbét*, amely alacsonyabb fokszámú szegmensekből áll
- Így kiküszöböhető a magas fokszámmal járó oszcilláció, illetve a kontrollpontok módosításának kihatása az egész görbére, de a polinomiális szegmensek folytonos csatlakozására külön figyelni kell

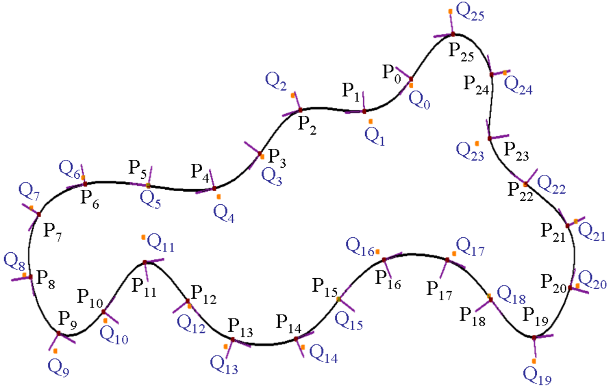
Spline görbék

- Ne egyetlen polinommal próbáljuk interpolálni vagy approximálni az adatpontjainkat (vezérlőpontjainkat)!
- Használjunk *összetett görbét*, amely alacsonyabb fokszámú szegmensekből áll
- Így kiküszöböhető a magas fokszámmal járó oszcilláció, illetve a kontrollpontok módosításának kihatása az egész görbére, de a polinomiális szegmensek folytonos csatlakozására külön figyelni kell
- És még sok minden másra is, de...

Spline görbék

- Ne egyetlen polinommal próbáljuk interpolálni vagy approximálni az adatpontjainkat (vezérlőpontjainkat)!
- Használjunk *összetett görbét*, amely alacsonyabb fokszámú szegmensekből áll
- Így kiküszöböhető a magas fokszámmal járó oszcilláció, illetve a kontrollpontok módosításának kihatása az egész görbére, de a polinomiális szegmensek folytonos csatlakozására külön figyelni kell
- És még sok minden másra is, de...
- Részletek: NumAnal - BSc.; Geometriai Modellezés, Felület és Testmodellezés - MSc.

Spline görbék



Tartalom

Harmadfokú Hermite interpoláció

- A görbénk tárolásához bizonyos pontokban jegyezzük fel a görbe pozícióját és egy bejárásához tartozó sebesség, gyorsulás stb. ... vektorát (azaz legyenek adottak $\mathbf{x}(t_i), \mathbf{x}'(t_i), \mathbf{x}''(t_i), ...$) pont és derivált adatok, $i = 0, 1, ...$)

Harmadfokú Hermite interpoláció

- A görbénk tárolásához bizonyos pontokban jegyezzük fel a görbe pozícióját és egy bejárásához tartozó sebesség, gyorsulás stb. ... vektorát (azaz legyenek adottak $\mathbf{x}(t_i), \mathbf{x}'(t_i), \mathbf{x}''(t_i), \ldots$) pont és derivált adatok, $i = 0, 1, \ldots$)
- Keressünk egy olyan bázist, amibe ezeket a fenti adatokat *koordinátaként* beírva az adott szinten a beírt adatot kapjuk vissza

Harmadfokú Hermite interpoláció

- Két pont között, $[0,1]$-re megfogalmazza, keressük a

$$\mathbf{p}(t) = H_0^0(t)\mathbf{p}_0 + H_0^1(t)\mathbf{m}_0 + H_1^0(t)\mathbf{p}_1 + H_1^1(t)\mathbf{m}_1$$

alakú parametrikus görbét amelyre

$$\begin{aligned}\mathbf{p}(0) &= \mathbf{p}_0\\ \mathbf{p}(1) &= \mathbf{p}_1\\ \mathbf{p}'(0) &= \mathbf{m}_0\\ \mathbf{p}'(1) &= \mathbf{m}_1\end{aligned}$$

Harmadfokú Hermite bázisfüggvények

$$\mathbf{p}(t) = H_0^0(t)\mathbf{p}_0 + H_0^1(t)\mathbf{m}_0 + H_1^0(t)\mathbf{p}_1 + H_1^1(t)\mathbf{m}_1$$

Harmadfokú Hermite bázisfüggvények

$$\mathbf{p}(t) = H_0^0(t)\mathbf{p}_0 + H_0^1(t)\mathbf{m}_0 + H_1^0(t)\mathbf{p}_1 + H_1^1(t)\mathbf{m}_1$$

- Keressük harmadfokú, egész polinomként az ismeretlen $H_i^j(t)$ bázisfüggvényeket, azaz legyen

$$H_i^j(t) = a_{ij}t^3 + b_{ij}t^2 + c_{ij}t + d_{ij}$$

Harmadfokú Hermite bázisfüggvények

$$\mathbf{p}(t) = H_0^0(t)\mathbf{p}_0 + H_0^1(t)\mathbf{m}_0 + H_1^0(t)\mathbf{p}_1 + H_1^1(t)\mathbf{m}_1$$

- Keressük harmadfokú, egész polinomként az ismeretlen $H_i^j(t)$ bázisfüggvényeket, azaz legyen

$$H_i^j(t) = a_{ij}t^3 + b_{ij}t^2 + c_{ij}t + d_{ij}$$

- Azt akarjuk, hogy $\mathbf{p}(0) = \mathbf{p}_0$, $\mathbf{p}(1) = \mathbf{p}_1$, $\mathbf{p}'(0) = \mathbf{m}_0$, $\mathbf{p}'(1) = \mathbf{m}_1$ teljesüljenek

Harmadfokú Hermite bázisfüggvények

$$\mathbf{p}(t) = H_0^0(t)\mathbf{p}_0 + H_0^1(t)\mathbf{m}_0 + H_1^0(t)\mathbf{p}_1 + H_1^1(t)\mathbf{m}_1$$

- Keressük harmadfokú, egész polinomként az ismeretlen $H_i^j(t)$ bázisfüggvényeket, azaz legyen

$$H_i^j(t) = a_{ij}t^3 + b_{ij}t^2 + c_{ij}t + d_{ij}$$

- Azt akarjuk, hogy $\mathbf{p}(0) = \mathbf{p}_0$, $\mathbf{p}(1) = \mathbf{p}_1$, $\mathbf{p}'(0) = \mathbf{m}_0$, $\mathbf{p}'(1) = \mathbf{m}_1$ teljesüljenek
- Ekkor megoldandó $a_{ij}, b_{ij}, c_{ij}, d_{ij}$, $i, j = 0, 1$-re

$$\begin{array}{cccc}
H_0^0(0) = 1 & H_0^1(0) = 0 & H_1^0(0) = 0 & H_1^1(0) = 0 \\
(H_0^0)'(0) = 0 & (H_0^1)'(0) = 1 & (H_1^0)'(0) = 0 & (H_1^1)'(0) = 0 \\
H_0^0(1) = 0 & H_0^1(1) = 0 & H_1^0(1) = 1 & H_1^1(1) = 0 \\
(H_0^0)'(1) = 0 & (H_0^1)'(1) = 0 & (H_1^0)'(1) = 0 & (H_1^1)'(1) = 1
\end{array}$$

Harmadfokú Hermite bázis

- A harmadfokú bázisunk ekkor a következő lesz:

$$
\begin{aligned}
H_0^0(t) &= 2t^3 - 3t^2 + 1 \\
H_0^1(t) &= t^3 - 2t^2 + t \\
H_1^0(t) &= -2t^3 + 3t^2 \\
H_1^1(t) &= t^3 - t^2
\end{aligned}
$$

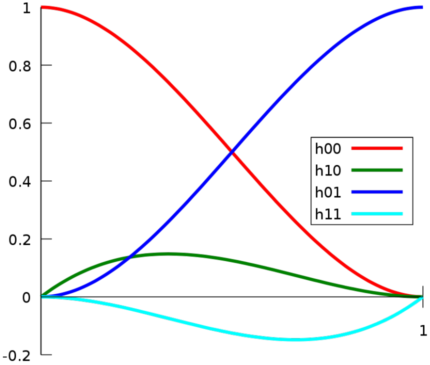
Harmadfokú Hermite bázis

- A harmadfokú bázisunk ekkor a következő lesz:

$$
\begin{aligned}
H_0^0(t) &= 2t^3 - 3t^2 + 1 \\
H_0^1(t) &= t^3 - 2t^2 + t \\
H_1^0(t) &= -2t^3 + 3t^2 \\
H_1^1(t) &= t^3 - t^2
\end{aligned}
$$

- Ha van $n+1$ bejövő adatunk, akkor minden pár közé szerkesztve egy-egy görbét az összeillesztésből kapott n harmadfokú szegmensből álló *spline* C^1 lesz

Harmadfokú Hermite bázis



Kitérő – geometriai folytonosság

- A szemünknek nem csak az lesz folytonos, ami parametrikusan is az

Kitérő – geometriai folytonosság

- A szemünknek nem csak az lesz folytonos, ami parametrikusan is az
- Nem teljesen szabatos matematikai definíciókat következnek

Kitérő – geometriai folytonosság

- A szemünknek nem csak az lesz folytonos, ami parametrikusan is az
- Nem teljesen szabatos matematikai definíciókat következnek
- A geometriai folytonosságnál paraméterezéstől független folytonossági megkötéseket teszünk:
  - G^0: a görbében/felületben nincsenek lyukak, nem szakad meg sehol

Kitérő – geometriai folytonosság

- A szemünknek nem csak az lesz folytonos, ami parametrikusan is az
- Nem teljesen szabatos matematikai definíciókat következnek
- A geometriai folytonosságnál paraméterezéstől független folytonossági megkötéseket teszünk:
  - G^0: a görbében/felületben nincsenek lyukak, nem szakad meg sehol
  - G^1: a csatlakozásoknál ha a deriváltak nem is egyeznek meg, de $\exists\alpha > 0$ úgy, hogy $\mathbf{m}_i = \alpha\mathbf{m}_{i+1}$

Kitérő – geometriai folytonosság

- A szemünknek nem csak az lesz folytonos, ami parametrikusan is az
- Nem teljesen szabatos matematikai definíciókat következnek
- A geometriai folytonosságnál paraméterezéstől független folytonossági megkötéseket teszünk:
  - G^0: a görbében/felületben nincsenek lyukak, nem szakad meg sehol
  - G^1: a csatlakozásoknál ha a deriváltak nem is egyeznek meg, de $\exists \alpha > 0$ úgy, hogy $\mathbf{m}_i = \alpha \mathbf{m}_{i+1}$
  - G^2: a görbe/felület görbülete folytonos a csatlakozásban is

Catmull-Rom spline

- Ne legyen közvetlenül adott a derivált, hanem a pontokból számoljuk őket a következőképp:

$$\mathbf{m}_i = \frac{\mathbf{p}_{i+1} - \mathbf{p}_{i-1}}{t_{i+1} - t_{i-1}}$$

Catmull-Rom spline

- Ne legyen közvetlenül adott a derivált, hanem a pontokból számoljuk őket a következőképp:

$$\mathbf{m}_i = \frac{\mathbf{p}_{i+1} - \mathbf{p}_{i-1}}{t_{i+1} - t_{i-1}}$$

- Ezután a megadott $\mathbf{p}_i$ és a fentiek szerint számított m_i adatokra páronként illesszünk Hermite-görbéket

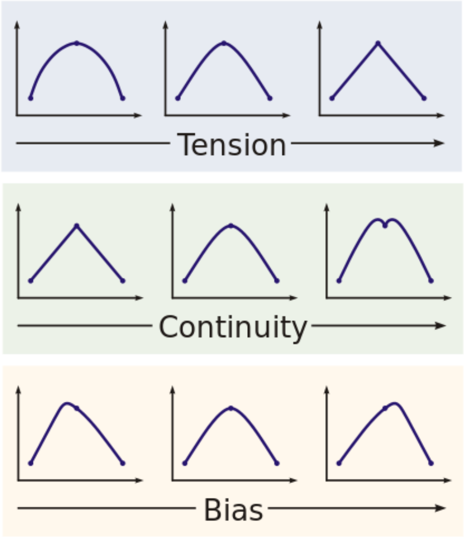
Kochanek-Bartels spline

- A Catmull-Rom spline-hoz hasonlóan itt is számított adat lesz a tangens, de most három paraméter is adott hozzá:
  - T: tension, $T \in [-1, 1]$
  - B: bias, $B \in [-1, 1]$
  - C: folytonosság (simaság), $C \in [-1, 1]$

Kochanek-Bartels spline

- A Catmull-Rom spline-hoz hasonlóan itt is számított adat lesz a tangens, de most három paraméter is adott hozzá:
  - T: tension, $T \in [-1, 1]$
  - B: bias, $B \in [-1, 1]$
  - C: folytonosság (simaság), $C \in [-1, 1]$
- A Catmull-Rom spline-t kapjuk, ha $T = B = C = 0$

Kochanek-Bartels spline



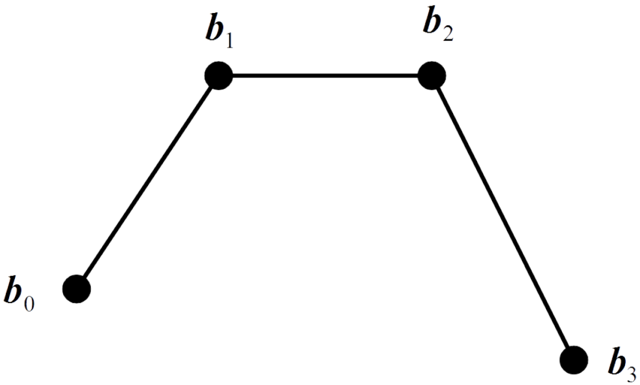
Kochanek-Bartels spline

- A fentiek felhasználásával az i-edik szegmens két végpontjának derivált-értékei legyenek

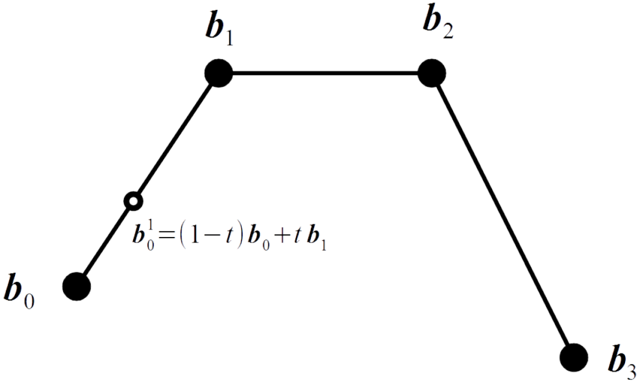
$$
\begin{aligned}
\mathbf{m}_i &= \frac{(1-T)(1+B)(1+C)}{2}(\mathbf{p}_i - \mathbf{p}_{i-1}) \\
&+ \frac{(1-T)(1-B)(1-C)}{2}(\mathbf{p}_{i+1} - \mathbf{p}_i) \\
\mathbf{m}_{i+1} &= \frac{(1-T)(1+B)(1-C)}{2}(\mathbf{p}_{i+1} - \mathbf{p}_i) \\
&+ \frac{(1-T)(1-B)(1+C)}{2}(\mathbf{p}_{i+2} - \mathbf{p}_{i+1})
\end{aligned}
$$

Tartalom

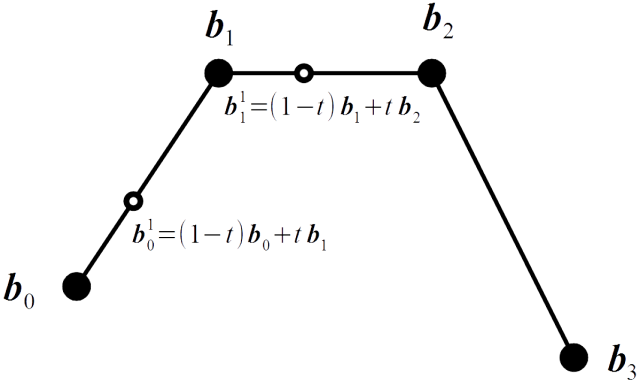
Bézier-görbe – de Casteljau algoritmus



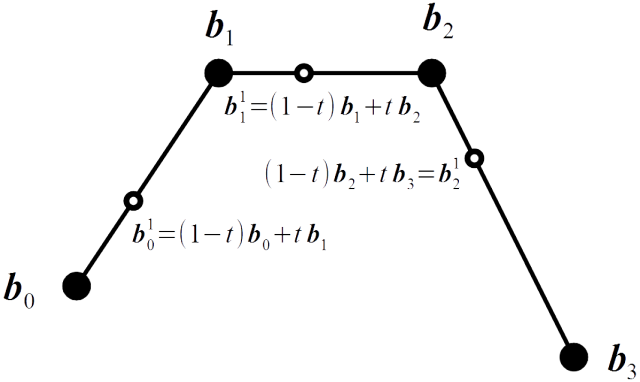
Bézier-görbe – de Casteljau algoritmus



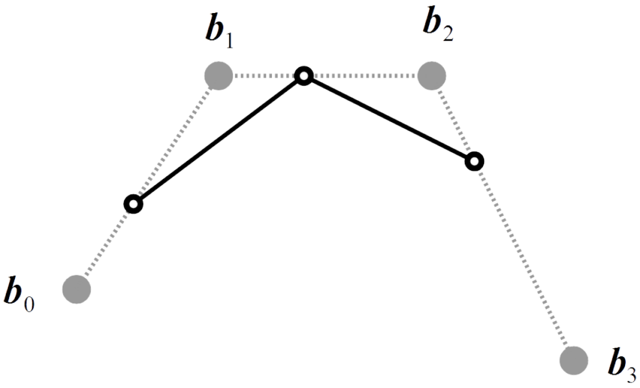
Bézier-görbe – de Casteljau algoritmus



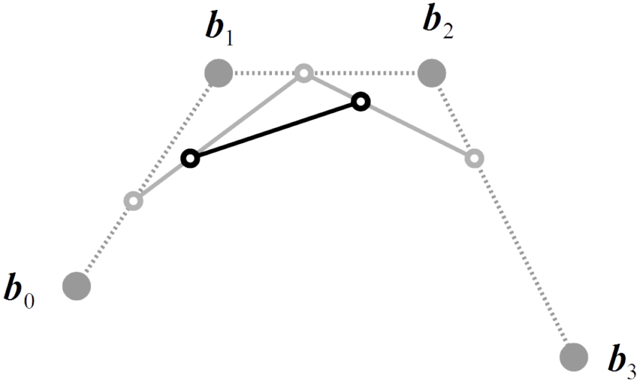
Bézier-görbe – de Casteljau algoritmus



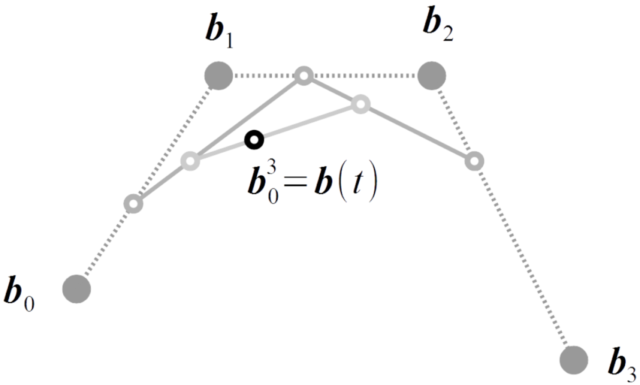
Bézier-görbe – de Casteljau algoritmus



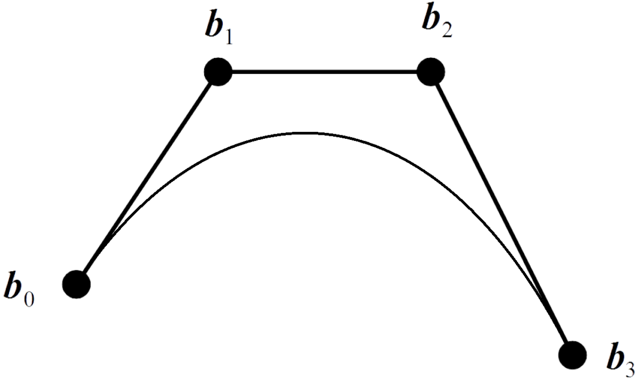
Bézier-görbe – de Casteljau algoritmus



Bézier-görbe – de Casteljau algoritmus



Bézier-görbe – de Casteljau algoritmus



Bézier-görbe – de Casteljau algoritmus

$\boldsymbol{b}_0$

$\boldsymbol{b}_1$

$\boldsymbol{b}_2$

$\boldsymbol{b}_3$

Bézier-görbe – de Casteljau algoritmus

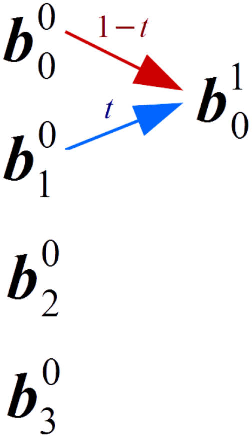
$$b_0^0$$

$$b_1^0$$

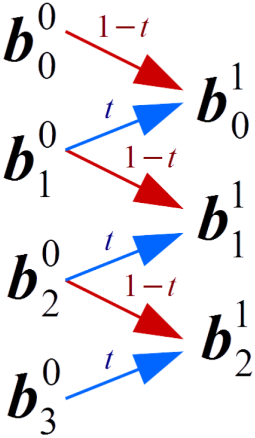
$$b_2^0$$

$$b_3^0$$

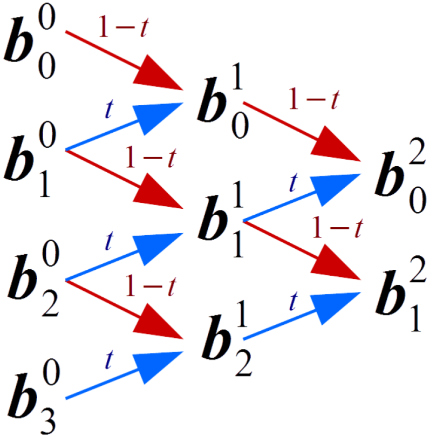
Bézier-görbe – de Casteljau algoritmus



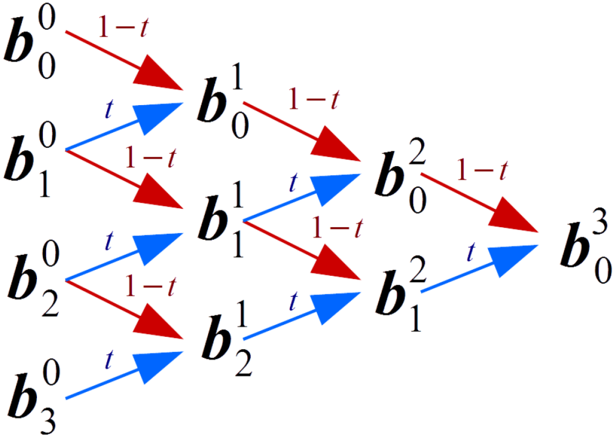
Bézier-görbe – de Casteljau algoritmus



Bézier-görbe – de Casteljau algoritmus



Bézier-görbe – de Casteljau algoritmus



Bézier-görbe – de Casteljau algoritmus

- Általánosan: egy n-edfokú Bézier-görbének $n+1$ kontrollpontja van, ezeket jelöljük $\mathbf{b}_i \in \mathbb{E}^3$-mal, $i = 0, 1, .., n$

Bézier-görbe – de Casteljau algoritmus

- Általánosan: egy n-edfokú Bézier-görbének $n+1$ kontrollpontja van, ezeket jelöljük $\mathbf{b}_i \in \mathbb{E}^3$-mal, $i = 0, 1, .., n$
- A de Casteljau algoritmus egy rekurzív kiértékelése a görbének:

$$\mathbf{b}_i^{k+1} = (1-t)\mathbf{b}_i^k + t\mathbf{b}_{i+1}^k$$

ahol $\mathbf{b}_i^0 := \mathbf{b}_i$, $k = 0, 1, .., n$, $i = 0, .., n-k$.

Bézier-görbe – de Casteljau algoritmus

- Általánosan: egy n-edfokú Bézier-görbének $n+1$ kontrollpontja van, ezeket jelöljük $\mathbf{b}_i \in \mathbb{E}^3$-mal, $i = 0, 1, .., n$
- A de Casteljau algoritmus egy rekurzív kiértékelése a görbének:

$$\mathbf{b}_i^{k+1} = (1-t)\mathbf{b}_i^k + t\mathbf{b}_{i+1}^k$$

ahol $\mathbf{b}_i^0 := \mathbf{b}_i$, $k = 0, 1, .., n$, $i = 0, .., n-k$.

- A görbe t paraméteréhez tartozó pontja

$$\mathbf{b}(t) := \mathbf{b}_0^n$$

Bézier-görbe

- Használjuk a *Bernstein-bázist*: $B_i^n(t) := \binom{n}{i} t^i (1-t)^{n-i}$

Bézier-görbe

- Használjuk a *Bernstein-bázist*: $B_i^n(t) := \binom{n}{i} t^i(1-t)^{n-i}$
- A $\mathbf{b}_0, \ldots, \mathbf{b}_n \in \mathbb{R}^3$ *kontrollpontok* által meghatározott n-edfokú Bézier-görbe ekkor

$$\mathbf{b}(t) = \sum_{i=0}^{n} B_i^n(t)\mathbf{b}_i,$$

ahol $t \in [0, 1]$.

Bézier-görbe

- Használjuk a *Bernstein-bázist*: $B_i^n(t) := \binom{n}{i} t^i(1-t)^{n-i}$
- A $\mathbf{b}_0, ..., \mathbf{b}_n \in \mathbb{R}^3$ *kontrollpontok* által meghatározott n-edfokú Bézier-görbe ekkor

$$\mathbf{b}(t) = \sum_{i=0}^{n} B_i^n(t)\mathbf{b}_i,$$

ahol $t \in [0, 1]$.
- HF: $\sum_{i=0}^n B_i^n(t) = 1$ teljesül, $\forall t \in [0, 1]$

Bézier-görbe

- Használjuk a *Bernstein-bázist*: $B_i^n(t) := \binom{n}{i} t^i(1-t)^{n-i}$
- A $\mathbf{b}_0, ..., \mathbf{b}_n \in \mathbb{R}^3$ *kontrollpontok* által meghatározott n-edfokú Bézier-görbe ekkor

$$\mathbf{b}(t) = \sum_{i=0}^{n} B_i^n(t)\mathbf{b}_i,$$

  ahol $t \in [0,1]$.
- HF: $\sum_{i=0}^{n} B_i^n(t) = 1$ teljesül, $\forall t \in [0,1]$
- A görbe „nagyjából” követi a vezérlőpontok poligonjának az alakját, de nem halad át mindegyiken! Ez egy *approximáló* séma

Bézier-görbe

- Használjuk a *Bernstein-bázist*: $B_i^n(t) := \binom{n}{i} t^i(1-t)^{n-i}$
- A $\mathbf{b}_0, ..., \mathbf{b}_n \in \mathbb{R}^3$ *kontrollpontok* által meghatározott n-edfokú Bézier-görbe ekkor

$$\mathbf{b}(t) = \sum_{i=0}^{n} B_i^n(t)\mathbf{b}_i,$$

  ahol $t \in [0, 1]$.
- HF: $\sum_{i=0}^n B_i^n(t) = 1$ teljesül, $\forall t \in [0, 1]$
- A görbe „nagyjából” követi a vezérlőpontok poligonjának az alakját, de nem halad át mindegyiken! Ez egy *approximáló* séma
- További részletek: Geometriai Modellezés MSc, Analízis (Stone-Weierstrass approximációs tétel)

Bézier-görbe

- Egyetlen kontrollpont módosítása az egész görbére hat

Bézier-görbe

- Egyetlen kontrollpont módosítása az egész görbére hat
- Bernstein-bázis néhány *n*-re:

Bézier-görbe

- Egyetlen kontrollpont módosítása az egész görbére hat
- Bernstein-bázis néhány *n*-re:
  - $B_0^1(t) = 1 - t, B_1^1(t) = t$

Bézier-görbe

- Egyetlen kontrollpont módosítása az egész görbére hat
- Bernstein-bázis néhány *n*-re:
  - $B_0^1(t) = 1 - t, B_1^1(t) = t \rightarrow$ lineáris interpoláció!

Bézier-görbe

- Egyetlen kontrollpont módosítása az egész görbére hat
- Bernstein-bázis néhány n-re:
  - $B_0^1(t) = 1 - t, B_1^1(t) = t \rightarrow$ lineáris interpoláció!
  - $B_0^2(t) = (1-t)^2, B_1^2(t) = 2t(1-t), B_2^2(t) = t^2$

Bézier-görbe

- Egyetlen kontrollpont módosítása az egész görbére hat
- Bernstein-bázis néhány n-re:
  - $B_0^1(t) = 1 - t, B_1^1(t) = t \rightarrow$ lineáris interpoláció!
  - $B_0^2(t) = (1-t)^2, B_1^2(t) = 2t(1-t), B_2^2(t) = t^2$
  - $n = 3$: HF

Bézier-görbe

- Egyetlen kontrollpont módosítása az egész görbére hat
- Bernstein-bázis néhány n-re:
  - $B_0^1(t) = 1 - t, B_1^1(t) = t \rightarrow$ lineáris interpoláció!
  - $B_0^2(t) = (1-t)^2, B_1^2(t) = 2t(1-t), B_2^2(t) = t^2$
  - $n = 3$: HF
- Lényegében „összemossuk” a kontrollpontjainkat egymással a fenti függvényekkel súlyozva őket, ezzel megmondva, hogy egy adott $t \in [0, 1]$ paraméterértéknél melyik kontrollpont mennyire játszik „fontos” szerepet a görbe alakjának meghatározásában

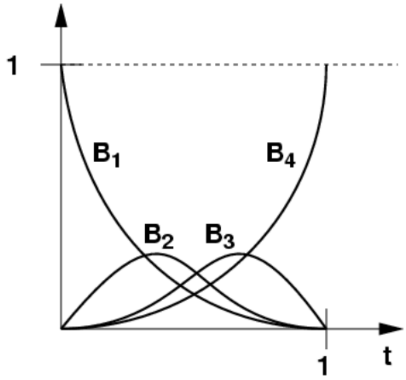
Bézier-görbe tulajdonságok

- A Bézier-görbe áthalad a két végponton ($\mathbf{b}_0$ és $\mathbf{b}_n$ pontokon)

Bézier-görbe tulajdonságok

- A Bézier-görbe áthalad a két végponton ($\mathbf{b}_0$ és $\mathbf{b}_n$ pontokon)
- A Bézier-görbe *invariáns az affin transzformációkra*

Bézier-görbe – bázisfüggvények



Tartalom

Subdivision görbék

- Egy ötlet: az eddigiek során rögtön egy polinomot állítottunk elő a kontrollpontjainkból (lényegében: a kontrollpontok által meghatározott kontroll-poligonból)

Subdivision görbék

- Egy ötlet: az eddigiek során rögtön egy polinomot állítottunk elő a kontrollpontjainkból (lényegében: a kontrollpontok által meghatározott kontroll-poligonból)
- Megjelenítés során viszont úgyis szakaszokkal kell közelíteni!

Subdivision görbék

- Egy ötlet: az eddigiek során rögtön egy polinomot állítottunk elő a kontrollpontjainkból (lényegében: a kontrollpontok által meghatározott kontroll-poligonból)
- Megjelenítés során viszont úgyis szakaszokkal kell közelíteni! $\rightarrow$ dolgozzunk magán a kontroll-poligonon!

Subdivision görbék

- Egy ötlet: az eddigiek során rögtön egy polinomot állítottunk elő a kontrollpontjainkból (lényegében: a kontrollpontok által meghatározott kontroll-poligonból)
- Megjelenítés során viszont úgyis szakaszokkal kell közelíteni! $\rightarrow$ dolgozzunk magán a kontroll-poligonon!
- A subdivision, vagy rekurzív felosztással definiált sémák a kiinduló ponthalmazunkat (kontrollpontok halmazát) rekurzívan sűrítik, egyre finomabb lineáris közelítést is adva (legtöbbször)

Subdivision görbék

- A kiinduló ponthalmaz által meghatározott görbének a rekurzív sűrítést határgörbéjét („végtelen sok” finomítás utáni pontok halmazát) tekintjük

Subdivision görbék

- A kiinduló ponthalmaz által meghatározott görbének a rekurzív sűrítést határgörbéjét („végtelen sok” finomítás utáni pontok halmazát) tekintjük
- Nagy kifejezőerő (például a Chaikin saroklevágási algoritmus egy másodfokú B-spline görbét ad), de görbéknél lehet hatékonyabban is számolni sok esetben

Subdivision görbék – Chaikin saroklevágási algoritmus

- Legyen az aktuális vezérlőpont halmazunk $\{\mathbf{p}_i \in \mathbb{R}^3\}_{i=0}^n$

Subdivision görbék – Chaikin saroklevágási algoritmus

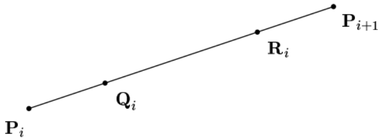
- Legyen az aktuális vezérlőpont halmazunk $\{\mathbf{p}_i \in \mathbb{R}^3\}_{i=0}^{n}$
- Az iterációs lépés során az új vezérlőpont halmazunk $\{\mathbf{q}_i, \mathbf{r}_i \in \mathbb{R}^3\}_{i=0}^{n-1}$ lesz, ahol

$$\mathbf{q}_i = \frac{3}{4}\mathbf{p}_i + \frac{1}{4}\mathbf{p}_{i+1}$$
$$\mathbf{r}_i = \frac{1}{4}\mathbf{p}_i + \frac{3}{4}\mathbf{p}_{i+1}$$

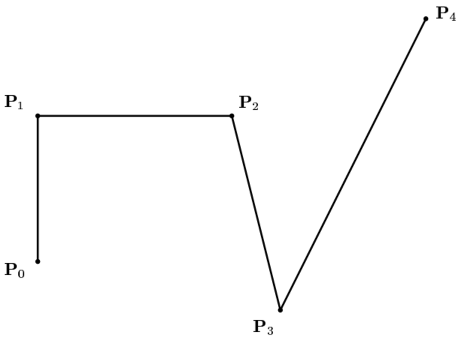
Subdivision görbék – Chaikin saroklevágási algoritmus

- Legyen az aktuális vezérlőpont halmazunk $\{\mathbf{p}_i \in \mathbb{R}^3\}_{i=0}^{n}$
- Az iterációs lépés során az új vezérlőpont halmazunk $\{\mathbf{q}_i, \mathbf{r}_i \in \mathbb{R}^3\}_{i=0}^{n-1}$ lesz, ahol

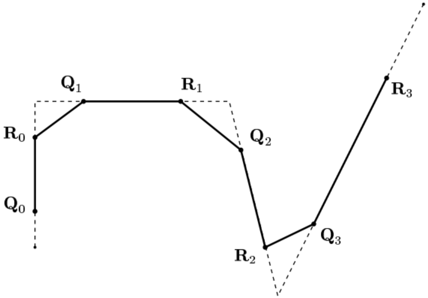
$$\mathbf{q}_i = \frac{3}{4}\mathbf{p}_i + \frac{1}{4}\mathbf{p}_{i+1}$$
$$\mathbf{r}_i = \frac{1}{4}\mathbf{p}_i + \frac{3}{4}\mathbf{p}_{i+1}$$



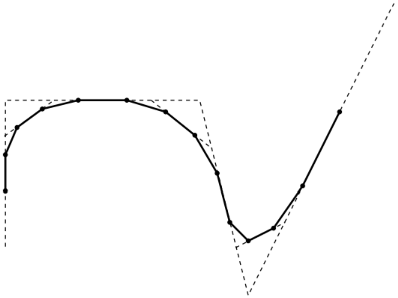
Subdivision görbék – Chaikin saroklevágási algoritmus



Subdivision görbék – Chaikin saroklevágási algoritmus



Subdivision görbék – Chaikin saroklevágási algoritmus



Subdivision görbék – Chaikin saroklevágási algoritmus