PPKE ITK

A számítógépes grafika alapjai

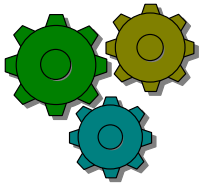
Geometriai modellezés

Előadó: Benedek Csaba

Tananyag : Szirmay-Kalos László, Benedek Csaba

Fides et Ratio

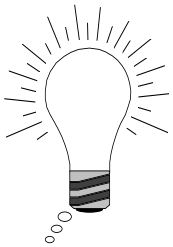
Visszatekintő: a számítógépes grafika feladatai



geometria

mozgás

textúrák



világítás

kamera

modellezés

Virtuális világ modell:
- objektumok
- primitívek
- számok

képszintézis

számítás

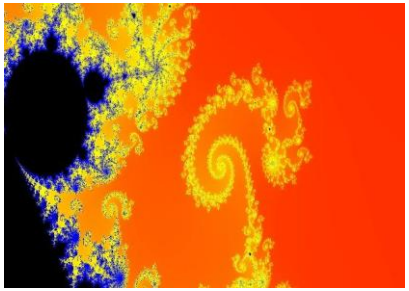
mérés

Modellezés

- *2D geometria:* pontok, görbék, síkbeli területek

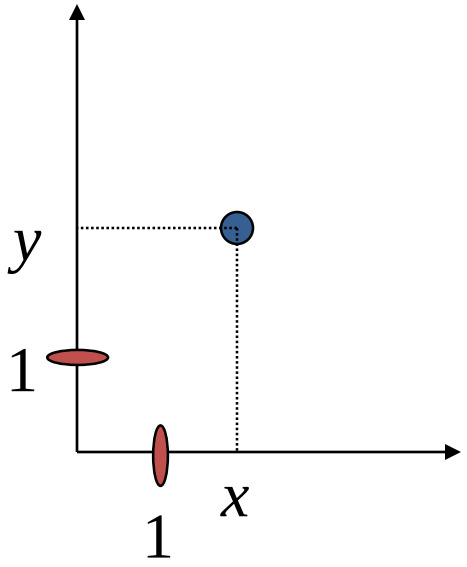
- *3D geometria:* mint 2D + térbeli felületek, 3D-s testek

- *Magasabb dimenziójú* adatok vizualizációja (mérnöki gyakorlat): alsóbb dimenziós altérbe vetítéssel
- *Törtdimenzió*: fraktálok

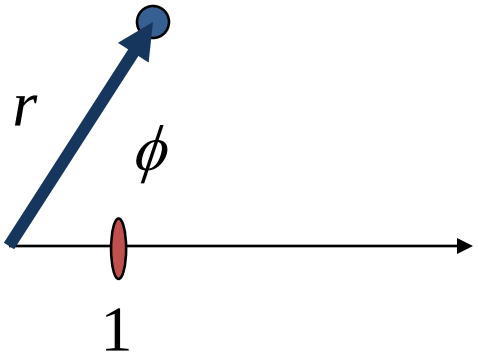


Pontok definiálása (2D)

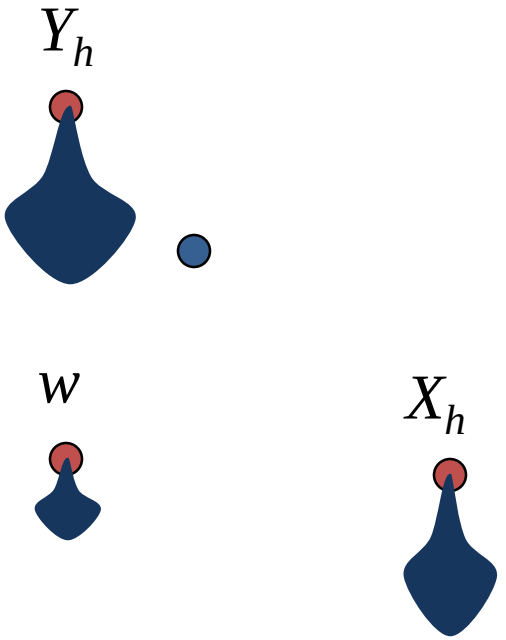
- Koordináták megadása alkalmas koordináta rendszerben:



Descartes koord. rsz.
eltolás



Polár koord. rsz.
elforgatás



Baricentrikus
Homogén
vetítés (lásd később)

Görbék definiálása (2D)

- Koordinátáik (helyvektoraik) kielégítenek egy egyenletet :

implicit: $f(x, y) = 0$,	$f(\mathbf{r}) = 0$
2D egyenes:	
$ax + by + c = 0$,	$\mathbf{n}\cdot(\mathbf{r} - \mathbf{r}_0) = 0$
Kör:	
$(x-x_0)^2 + (y-y_0)^2 - R^2 = 0$,	$\lvert\mathbf{r} - \mathbf{r}_0\rvert^2 - R^2 = 0$
paraméteres: $x = x(t)$, $y = y(t)$,	$\mathbf{r} = \mathbf{r}(t)$
2D egyenes: $t \in [-\infty,\infty]$	
$x(t) = x_0 + v_x t$,	$\mathbf{r} = \mathbf{r}_0 + \mathbf{v}\, t$
$y(t) = y_0 + v_y t$,	
Kör: $t \in [0,1]$	
$x(t) = x_0 + R \cos 2\pi t$	$\mathbf{r} = \mathbf{r}_0 + R(\cos 2\pi t, \sin 2\pi t)$
$y(t) = y_0 + R \sin 2\pi t$	

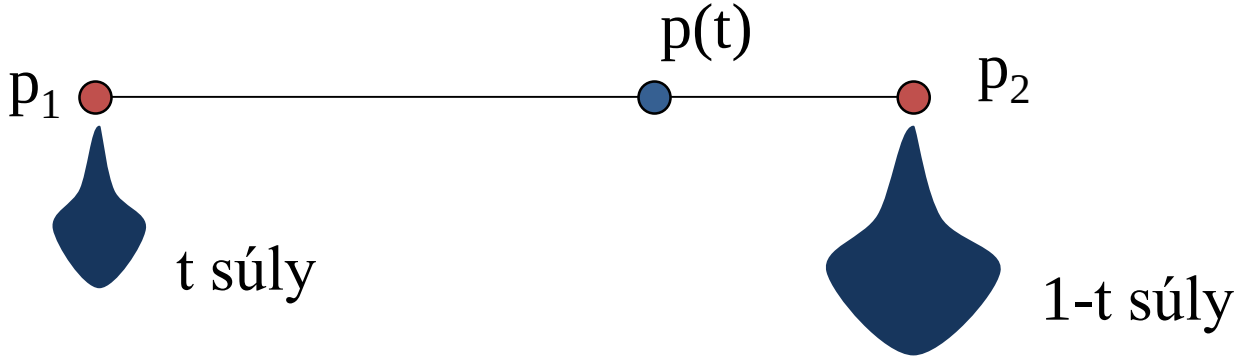
3D szakasz

- $p_1 = (x_1, y_1, z_1)$ -től $p_2 = (x_2, y_2, z_2)$ -ig tartó szakasz egyenlete:

$$x(t) = x_1 \cdot t + x_2 \cdot (1 - t),$$

$$y(t) = y_1 \cdot t + y_2 \cdot (1 - t), \qquad t \in [0,1]$$

$$z(t) = z_1 \cdot t + z_2 \cdot (1 - t),$$



Szabadformájú görbék

- Klasszikus görbeszegmensek – egyszerű analitikus egyenlet pl szakasz, körvonal
- Általános eset nem írható le klasszikus görbeszegmensekkel
- Szakaszok sorozatával közelítés – nem differenciálható a kapcsolódási pontokban
  - pl mechanikai alkalmazások esetén ill. animációknál megengedhetetlen – pl út-görbe esetén, a sebesség, gyorsulás nem változhat ugrásszerűen
  - most csak a modellezésről beszélünk (!) a végső rasztertizáció más kérdés...

Szabadformájú görbék

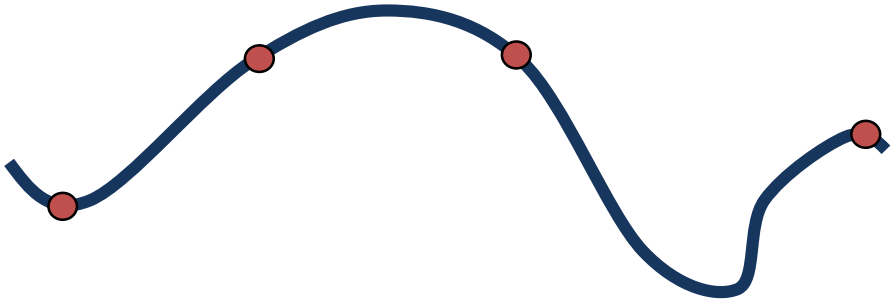
- Polinom: $x(t) = \sum_{i=0}^{n} a_i \cdot t^i, \quad y(t) = \sum_{i=0}^{n} b_i \cdot t^i$
  - vagy vektoros formában:

$$\vec{r}(t) = \sum_{i=0}^{n} [a_i, b_i] \cdot t^i, \; t \in [0,1]$$

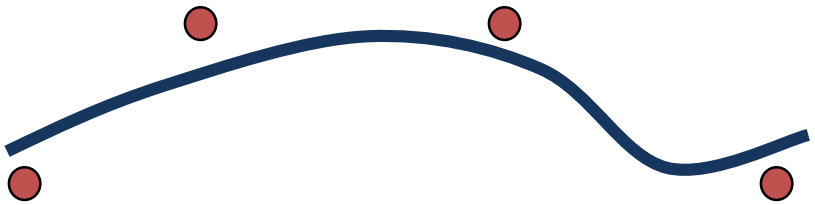
  - polinomegyütthatóknak nincs szemléletes tartalma, közvetlen származtatásuk nehézkes ☹
- Definíció kontrolpontokkal:
  - a görbe haladjon a mintapontokkal kijelölt út mentén!

Szabadformájú görbék

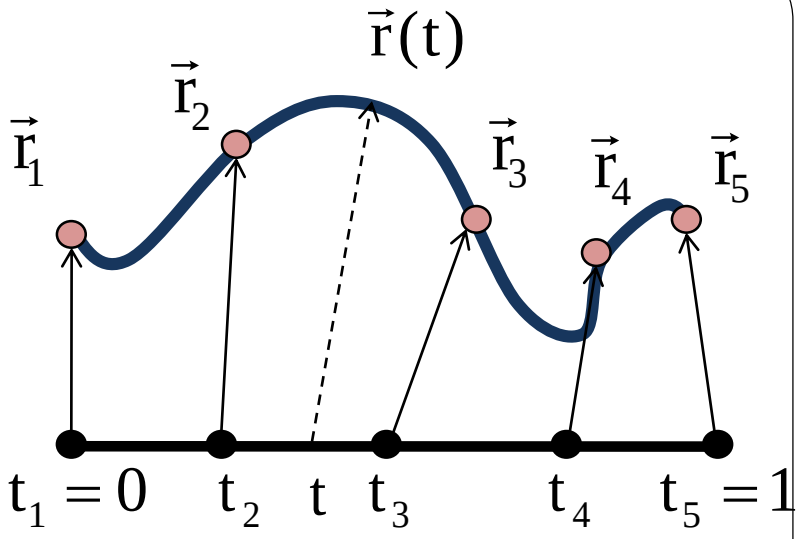
- Kontrollpontos definíció fajtái:
  - Interpoláció
    - megköveteljük, hogy a görbe átmenjen a vezérlőpontokon



  - Approximáció
    - csak azt írjuk elő, a görbe „nagyjából” kövesse a kijelölt irányvonalat – cserébe más jó tulajdonságokat várunk



Lagrange interpoláció



- Kontrolpontok:

$$\vec{r}_1, \vec{r}_2, \ldots, \vec{r}_n$$

- Csomópont (knot) vektor

$$(t_1, t_2, \ldots, t_n)$$

- Keressük azt az $\vec{r}(t) = \sum_i [a_i, b_i] \cdot t^i$ -t, amelyre

$$\vec{r}(t_1) = \vec{r}_1, \vec{r}(t_2) = \vec{r}_2, \ldots, \vec{r}(t_n) = \vec{r}_n$$

azaz: $$\vec{r}(t_j) = [x(t_j), y(t_j)] = \sum_{i=0}^{n-1} [a_i, b_i] \cdot t_j^i = \vec{r}_j$$

$$j = 1,2,\ldots,n$$

Lagrange interpoláció

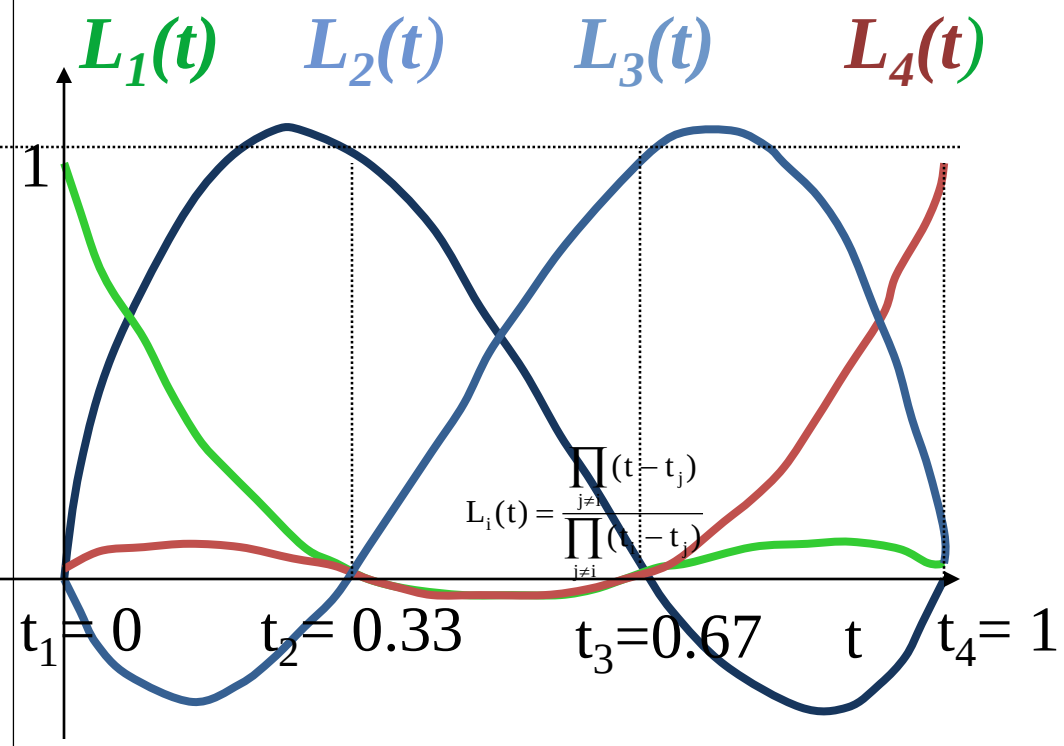
- Megoldás:

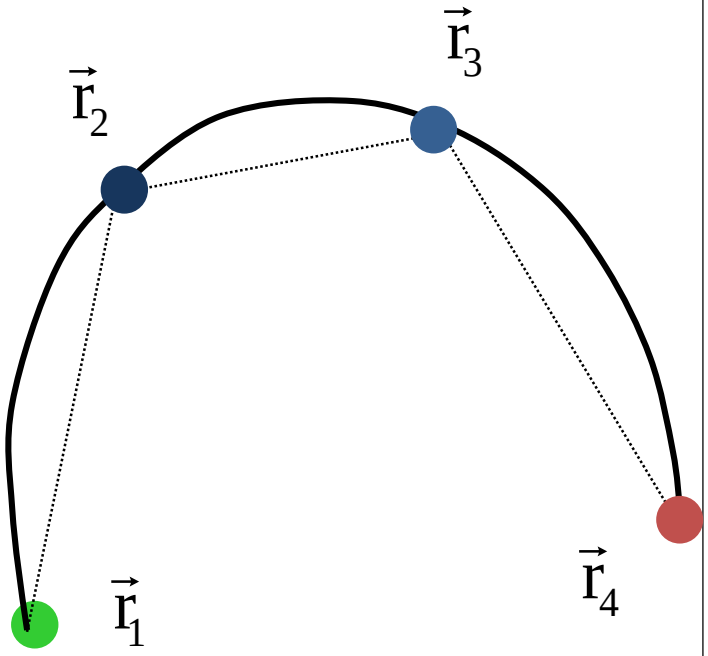
$$\vec{r}(t) = \sum_{i=1}^{n} L_i(t) \cdot \vec{r}_i \qquad \text{ahol} \qquad L_i(t) = \frac{\prod_{j \neq i} (t - t_j)}{\prod_{j \neq i} (t_i - t_j)}$$

- Pl n=3

$$r(t) = \frac{(t-t_2)(t-t_3)}{(t_1-t_2)(t_1-t_3)} \cdot \vec{r}_1 + \frac{(t-t_1)(t-t_3)}{(t_2-t_1)(t_2-t_3)} \cdot \vec{r}_2 + \frac{(t-t_1)(t-t_2)}{(t_3-t_1)(t_3-t_2)} \cdot \vec{r}_3$$

Lagrange interpoláció bázisfüggvényei

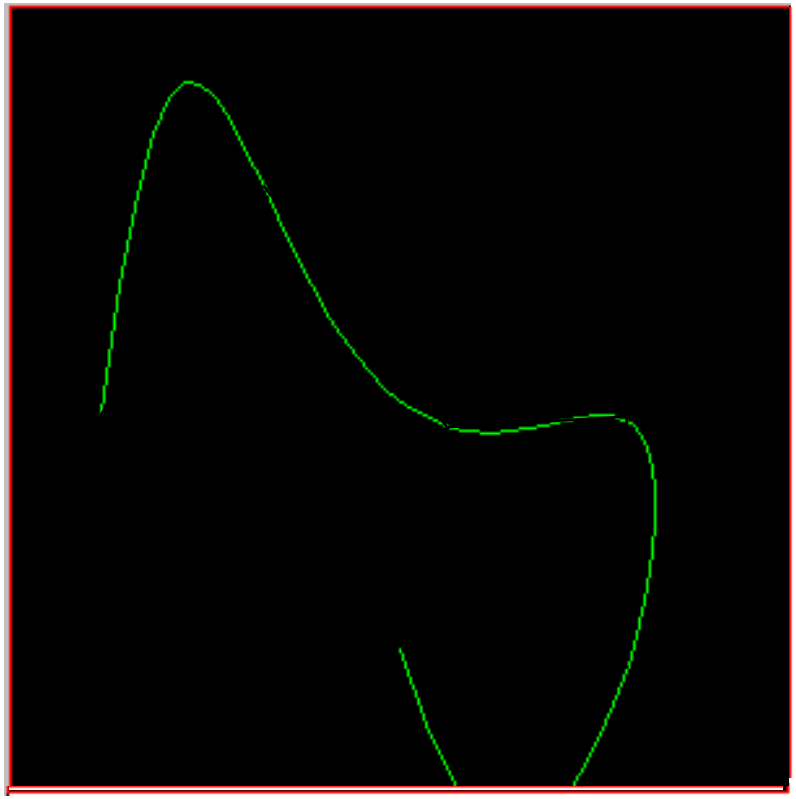




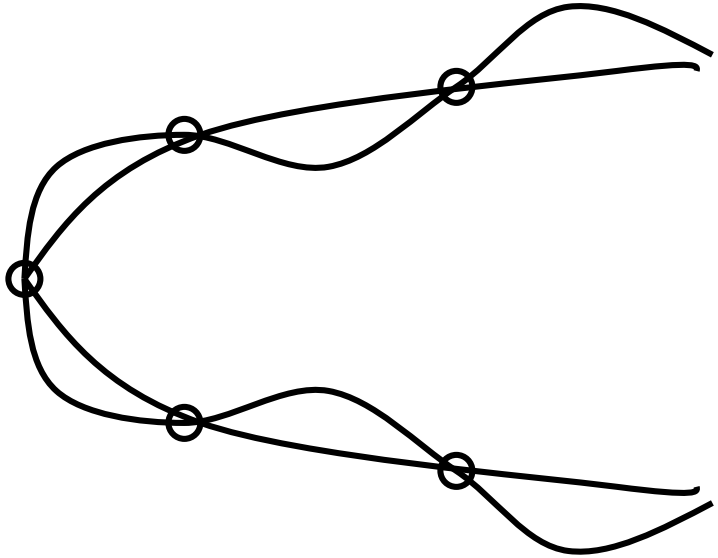
$$\vec{r}(t) = \sum_{i=1}^{n} L_i(t) \cdot \vec{r}_i$$

Görbeszerkesztés Lagrange interpolációval

Gond 1: egy-egy új/módosított vezérlőpont az egész görbe alakját változtatja



Gond 2: nem várt görbületek



Lagrange görbe implementációja

Együttható számítás:

```
double L( int i, double tt ) {
  double Li = 1.0;
  for(int j = 0; j < ptnum; j++) {
       if (i != j) Li *= (tt - knotVector[j]) /
  (knotVector[i] -knotVector[j]);
  }
  return Li;
}
```

Egyenletes knot vektor inicializálás:

```
 for (int i=0;i<ptnum;i++) {
       knotVector[i]=(double)i/(double(ptnum-1));
   }
```

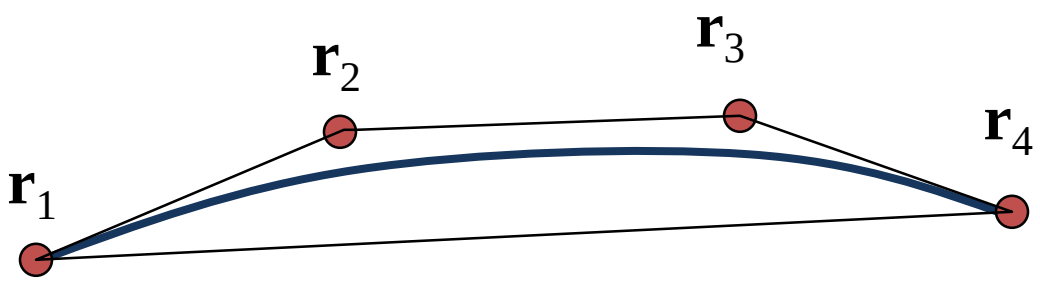
Lagrange görbe implementációja

(Csak pszeudo kód!!!)

```
MyPoint CalcLagrangePoint(float t) {

   MyPoint actPT(0,0);

   for(int i = 0; i < ptnum; i++)
       actPT+=ctrlPoint[i]*L(i,t);

  }
  return actPT;
}
```

Approximáció vs interpoláció

- Cél: ne legyen felesleges hullámosság
- Könnyítés: nem írjuk elő hogy a görbe átmenjen az összes vezérlőponton, csupán
  - a görbe minden pontja legyen a vezérlőpontok konvex burkán belül
  - az első és az utolsó vezérlőpontra pontosan illeszkedjen

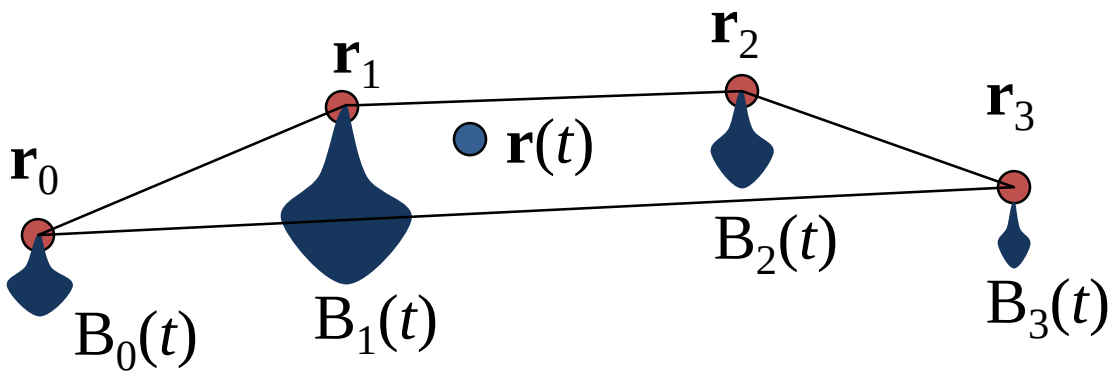


Bezier approximáció

- Keresett görbe: $$\vec{r}(t) = \sum_{i=0}^{m} B_i(t) \cdot \vec{r}_i$$
- $B_i(t)$ ne okozzon indokolatlan hullámokat
  - Konvex burok tulajdonság elégséges feltétele:

$$B_i(t) \geq 0, \quad \forall t \in [0,1], \quad i = 0,1,\ldots,m$$

$$\sum_{i=0}^{m} B_i(t) = 1, \ \forall t \in [0,1]$$



Bézier approximáció

- Súlyfüggvények a Bernstein polinomok

$$\vec{r}(t) = \sum_{i=0}^{m} B_i^{(m)}(t) \cdot \vec{r}_i \qquad B_i^{(m)}(t) = \binom{m}{i} t^i \cdot (1-t)^{m-i}$$

- Nemnegatívitás triviális

$$B_i^{(m)}(t) \geq 0, \quad \forall m, i, t$$

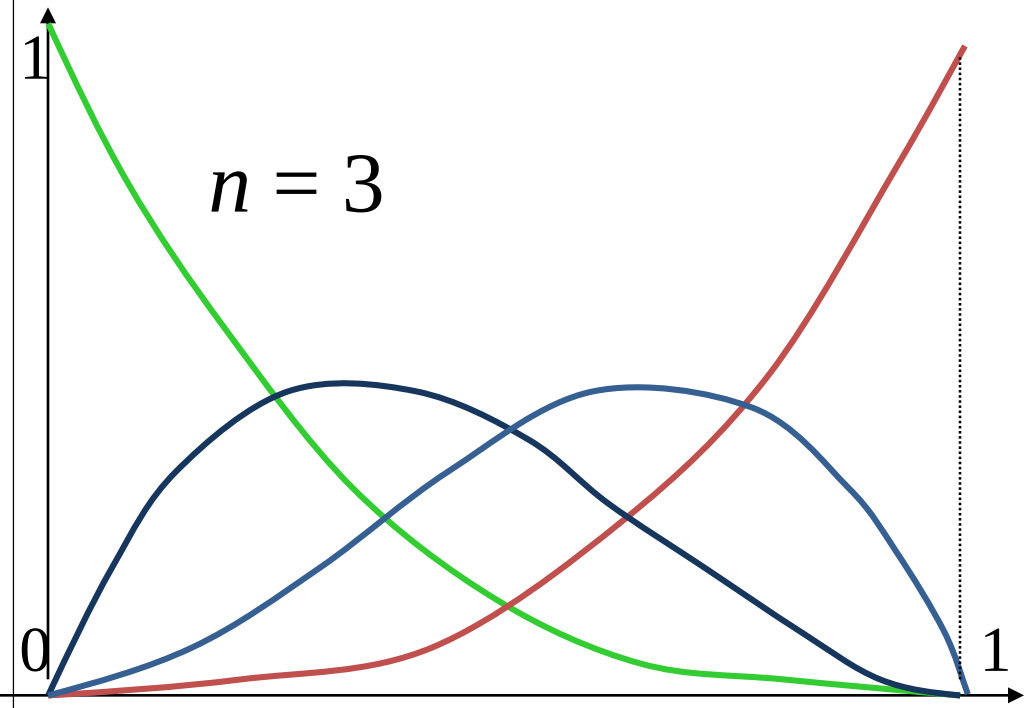
- Súlyfüggvények összege mindig 1 – binomiális tétel:

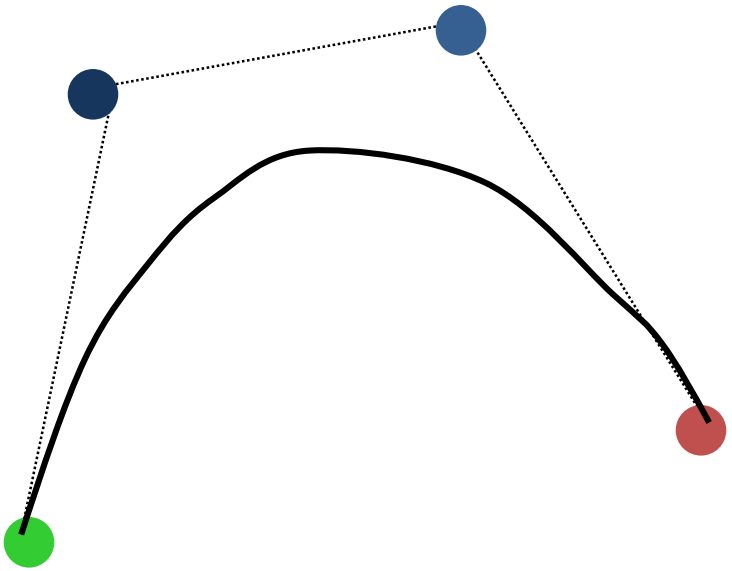
$$1 = \left(t + (1-t)\right)^m = \sum_{i=0}^{m} \binom{m}{i} t^i \cdot (1-t)^{m-i} = \sum_{i=0}^{m} B_i^{(m)}(t), \quad \forall t \in [0,1]$$

- Kezdet-vég feltétel teljesül, mivel:

$$B_0^{(m)}(0) = 1 \qquad B_m^{(m)}(1) = 1$$

Bezier approximáció bázisfüggvényei



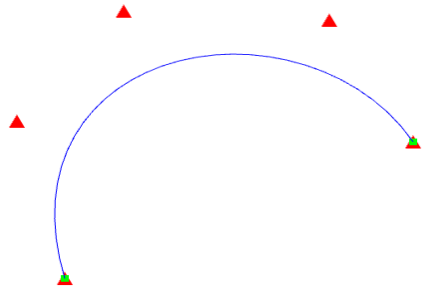


$$B_i^{(m)}(t) = \binom{m}{i} t^i \cdot (1-t)^{m-i}$$

Bernstein polinomok

$$\vec{r}(t) = \sum_{i=0}^{m} B_i^{(m)}(t) \cdot \vec{r}_i$$

BezierCurve implementáció



```
  MyPoint ctrlpoints[MAXPTNUM];
  int ptnum;
                …
 float B(int i, float t) {
      GLfloat Bi = 1.0;
      for(int j = 1; j <= i; j++) Bi *= t * (ptnum-j)/j;
      for( ; j < ptnum; j++) Bi *= (1-t);
      return Bi;
}

MyPoint CalcBezierPoint (float t) { //Pszeudo Point
       MyPoint actPT(0,0);
     for(int i = 0; i < ptnum; i++) {
            actPT+=ctrlpoints [i]*L(i,t);
      }
      return actPT;
}
```

Bezier görbe OpenGL implementációja

- `void` **`glMap1{fd}`**`(Glenum target, TYPE u1, TYPE u2, GLInt stride, GLInt order, const TYPE * points)`
  - Egydimenziós leképezés,
    - `target`: mit reprezentálnak a kontrollpontok: modelltérbeli pontot (`GL_MAP1_VERTEX_3`) vagy színt (`GL_MAP1_COLOR_4`) stb
    - `u1, u2`: paramétertartomány (nálunk [0,1])
    - `stride`: nálunk a pontok dimenziója (itt 3)
    - `order`: görbe rendje (kpontok száma+1)
    - `points`: kontrollpontokat tartalmazó tömb
  - Előtte engedélyezni kell az opciót: `glEnable(GL_MAP1_VERTEX_3);`

Bezier görbe OpenGL implementációja

- Kontroll pontok definiálása
  - `GLfloat ctrlpoints[MAXPTNUM][3];`
    - ctrlpoints[i][j] az i-edik kontrolpont j-edik koordinátája
    - 3D pontokkal dolgozik, a pont koordinátái rendre: [x,y,z], 2D-ben z=0-t használjunk

Bezier görbe OpenGL implementációja

- `void` **`glEvalCoord1{fd}`**`(TYPE u);`
  - az `u` paraméterértéknél kiértékeli a görbét, azaz meghatározza az aktuális pontot és esetünkben a `glVertex*()` parancsot is automatikusan végrehajtja rá (tehát azonnal meg is jeleníti)

Bonyolult görbék

- Sok vezérlőpont
- Ha egy polinomot illesztünk: nagyon magas fokszám kell (hullámosság, nem teljesül a lokális vezérelhetőség)
- Összetett görbék:
  - Több alacsony fokszámú + folytonos illesztés

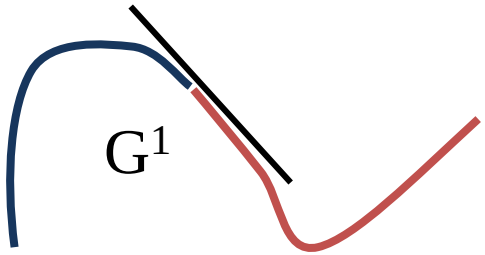
Folytonossági kategóriák

- C^n folytonos: a két görbe szegmens n-edik deriváltig megegyezik az illeszkedési pontban: $r_1^{(n)}(t_{veg}) = r_2^{(n)}(t_{kezd})$

C^0:

$\mathbf{r}_1(t_{veg}) = \mathbf{r}_2(t_{kezd})$

$G^0 = C^0$



C^1:

$\mathbf{r}_1'(t_{veg}) = \mathbf{r}_2'(t_{kezd})$

$C^1 \subseteq G^1$

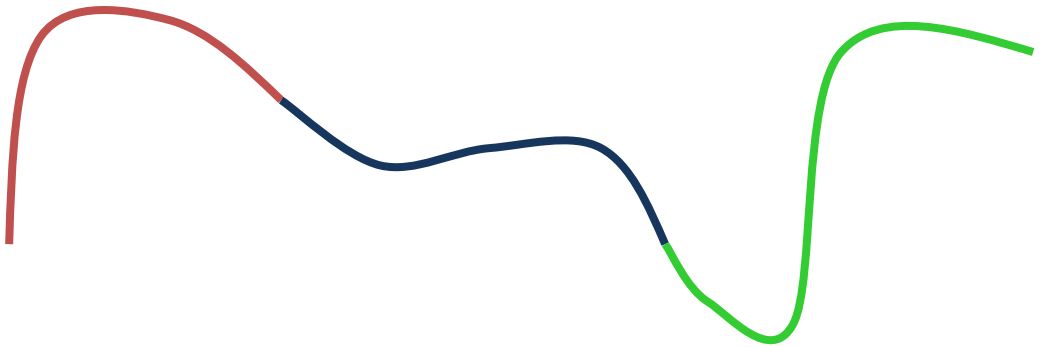
C^2:

$\mathbf{r}_1''(t_{veg}) = \mathbf{r}_2''(t_{kezd})$

- pl rugó animáció: a pozíciókoordináták C^n folytonossága biztosítja az erő sima változását (F=ma=mx")

Spline

- Spline: C^2 folytonos összetett görbe
  - Harmadfokú spline
  - B-spline



Harmadfokú spline

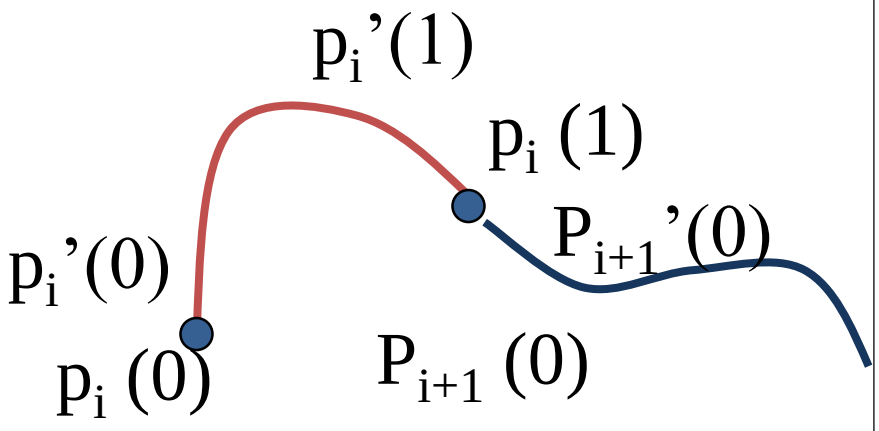
- $p(t) = a_3 t^3 + a_2 t^2 + a_1 t^1 + a_0$
- Új szemléletes reprezentáció:

  $p(0) = a_0$

  $p(1) = a_3 + a_2 + a_1 + a_0$

  $p'(0) = a_1$

  $p'(1) = 3a_3 + 2a_2 + a_1$



- $(p(0),p(1),p'(0),p'(1)), \leftrightarrow (a_3,a_2,a_1,a_0)$
- $p_i(0)=r_i$, $p_i(1)=r_{i+1}=p_{i+1}(0)$
- C^1 folytonosság: 2 paraméter közös
- C^2 folytonosság: $p_i''(1) = p_{i+1}''(0)$

Harmadfokú spline

- Pl: Két egymást követő szegmens $p_1(t)$, $p_2(t)$ paramétereinek számítása,
  - Adott: r_1, r_2, r_3 vezérlőpontok
  - Ismeretlen: $p_1(0), p_1(1), p_1'(0), p'_1(1), p_2(0), p_2(1), p_2'(0), p'_2(1)$ paraméterek
- 6 egyenlet, 8 ismeretlen
  - $p_1(0)=r_1$,
  - $p_1(1)=r_2$
  - $p_2(0)=r_2$
  - $p_2(1)=r_3$
  - $p'_1(1)=p'_2(0)$
  - $p_1''(1) = p_2''(0)$:

    $p_1''(t) = 6a_{13}\,t + 2a_{12} = f(p_1(0), p_1(1), p_1'(0), p'_1(1))$,

    $p_2''(t) = 6a_{23}\,t + 2a_{22} = f(p_2(0), p_2(1), p_2'(0), p'_2(1))$,
- $p'_1(0)$ és $p'_2(1)$ rögzítésével teljesen határozott lesz

B-spline

- Válasszunk olyan reprezentációt, amely C^2 folytonos, ha 3-t közösen birtokolnak
- Reprezentáció: vezérlőpontok – egy görbeszegmenst 4 egymást követő vezérlő pont definiál

$$\mathbf{r}^{i}(t) = B_0(t)\mathbf{r}_i + B_1(t)\mathbf{r}_{i+1} + B_2(t)\mathbf{r}_{i+2} + B_3(t)\mathbf{r}_{i+3}$$

$$\mathbf{r}^{i+1}(t) = B_0(t)\mathbf{r}_{i+1} + B_1(t)\mathbf{r}_{i+2} + B_2(t)\mathbf{r}_{i+3} + B_3(t)\mathbf{r}_{i+4}$$

B-spline bázisfüggvények

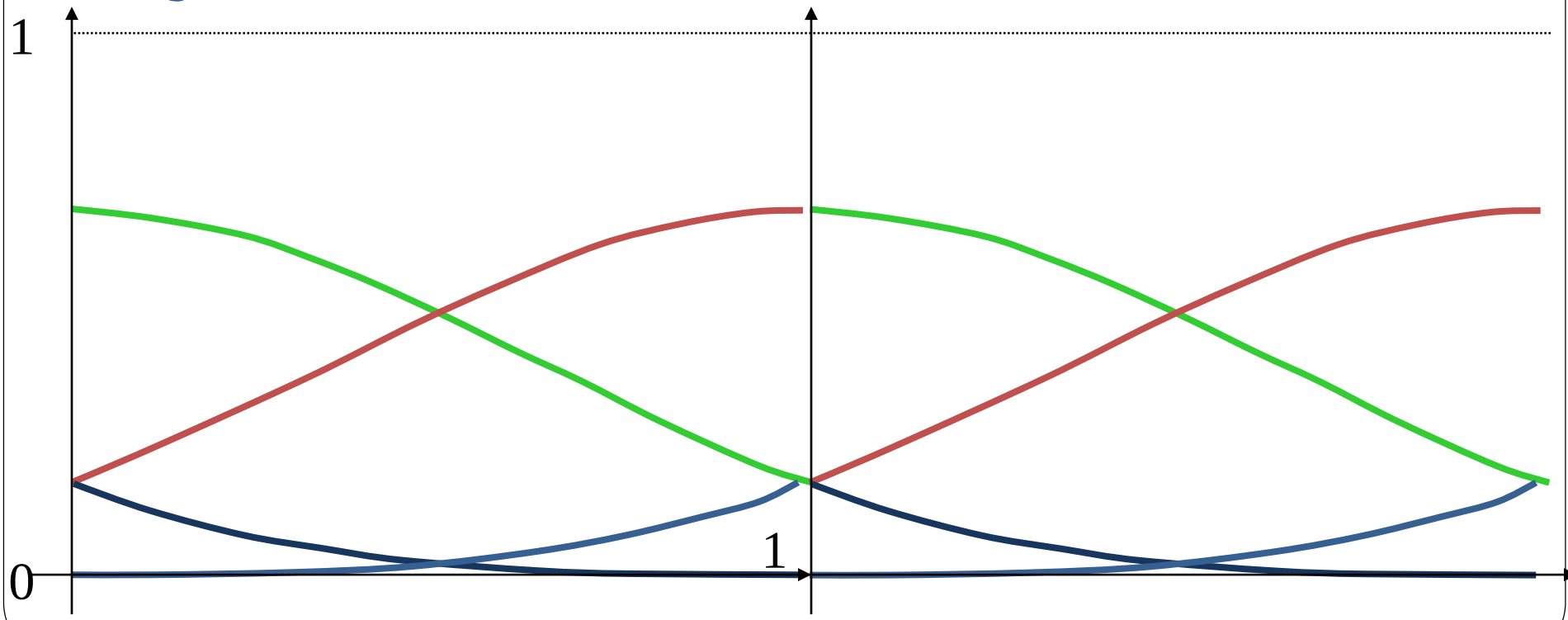
- Cirkuszi elefántok + Járulékos szempont: $\Sigma B_i(t) = 1$

$B_0(t) = (1-t)^3/6$

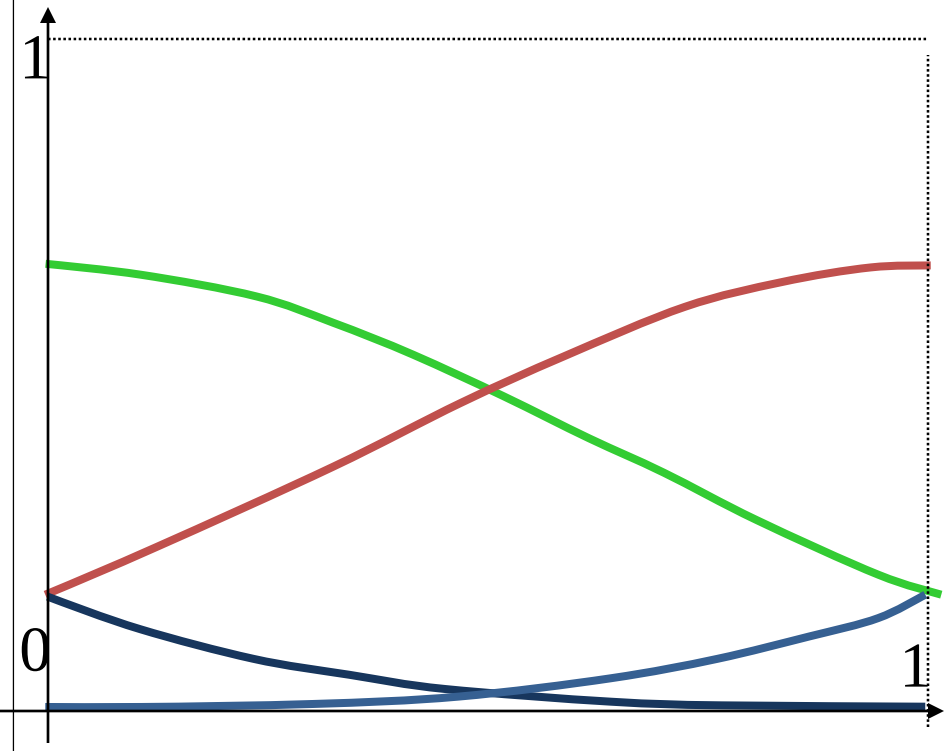
$B_1(t) = (1+3(1-t)+3t(1-t)^2)/$

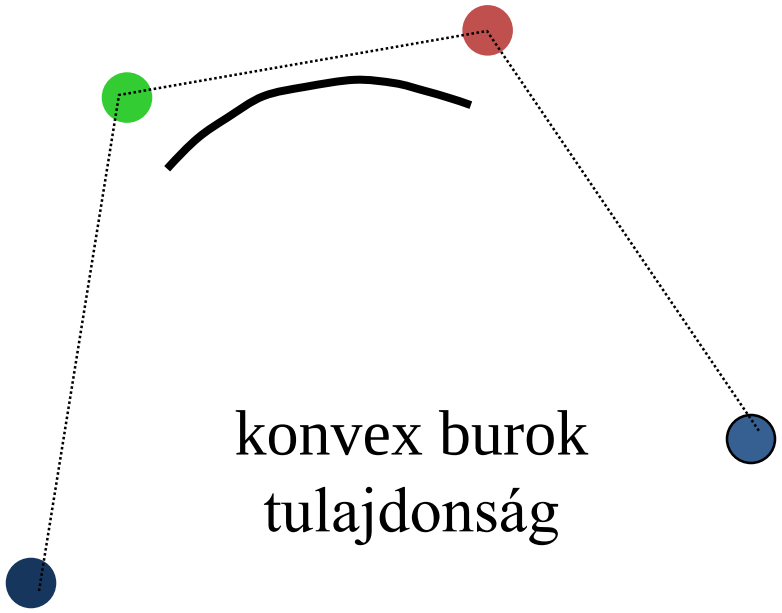
$B_3(t) = t^3/6$

$B_2(t) = (1+3t+3(1-t)t^2)/6$



B-spline görbeszegmens





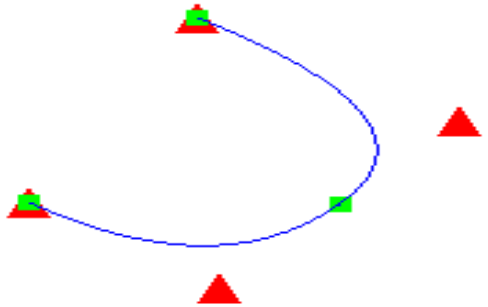
$B_0(t) = (1-t)^3 /6$

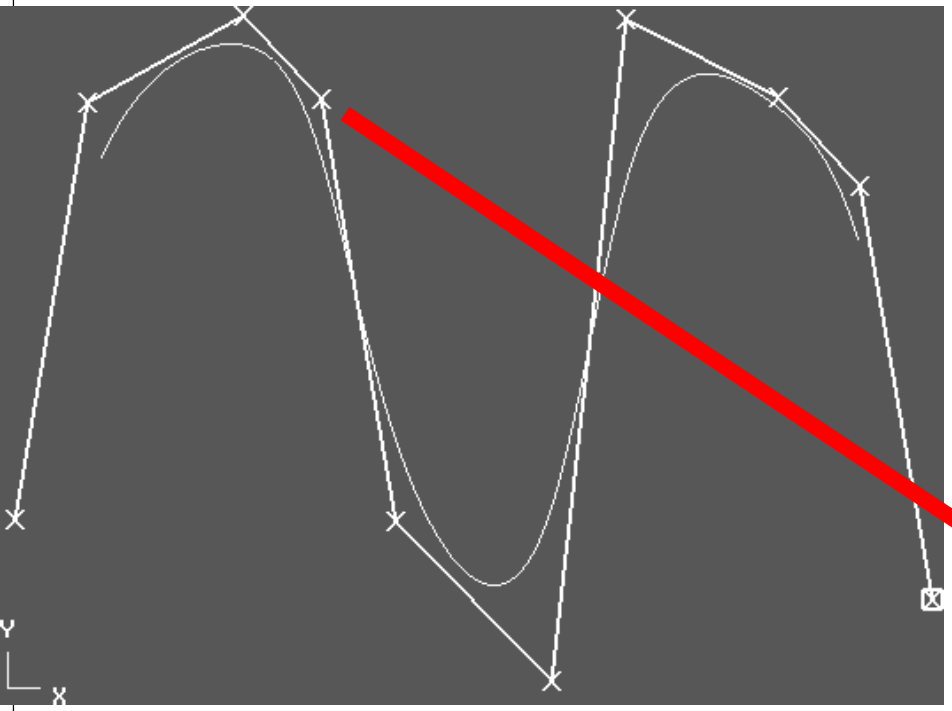
$B_1(t) = (1+3(1-t)+3t(1-t)^2) /6$

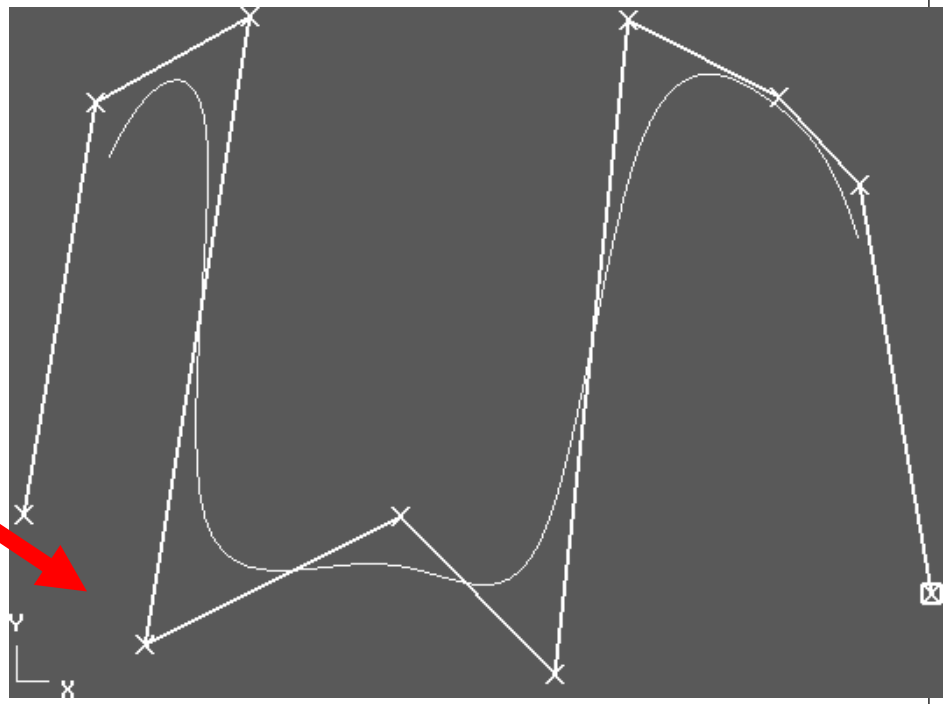
$B_3(t) = t^3 /6$

$B_2(t) = (1+3t+3(1-t)t^2) /6$

A B-spline lokálisan vezérelhető







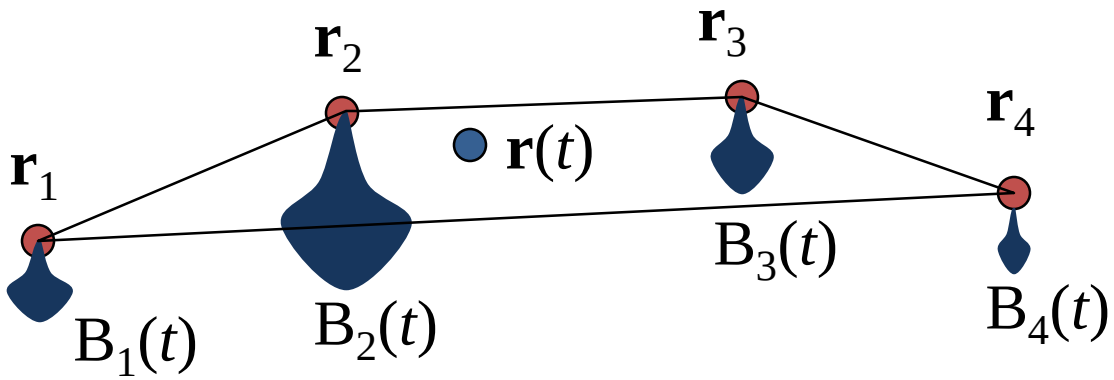
NUBS: Non-Uniform B-spline

- B-spline
  - minden szegmens 1 hosszú paramétertartomány
  - Akkor megy át a kontrol ponton, ha három egymás követő kontrolpont egymásra illeszkedik
- NUBS
  - az *i*. szegmens t_i -től t_{i+1} -ig.
  - Egy kontrolpont többször is számíthat:
    - A legalább 3-szoros pontokon a görbe átmegy

NUBS tulajdonságok

- Vezérlőpont „súlyozása” – közeli paramétertartomány kicsire választása
  - csak durva súlyozásra alkalmas, nehéz szemléletesen állítani a megfelelő hossz-paramétert
  - Zérus hosszúságú intervallumok – a görbe interpolálja a vezérlőpont
    - Elsőfokú (sátor) esetén elég 1 zérus hosszú intervallum
    - Másodfokú NUBS esetén 2, harmadfokúnál 3 egymást követő intervallumot kell zérusra állítani

Idáig: Nem racionális B-spline



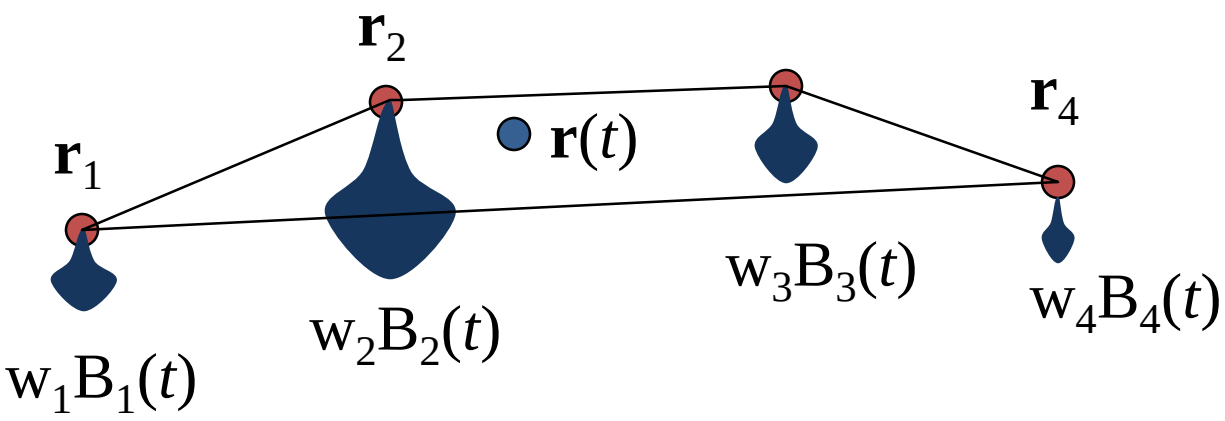
Idáig a súlyfüggvények: $\Sigma\, B_i(t) = 1$

Polinom!

Súlypont:

$$\mathbf{r}(t) = \frac{\Sigma\,(B_i(t)\,\mathbf{r}_i)}{\Sigma\,B_i(t)} = \Sigma\, B_i(t)\,\mathbf{r}_i$$

NURBS:
Non-uniform Rational B-spline

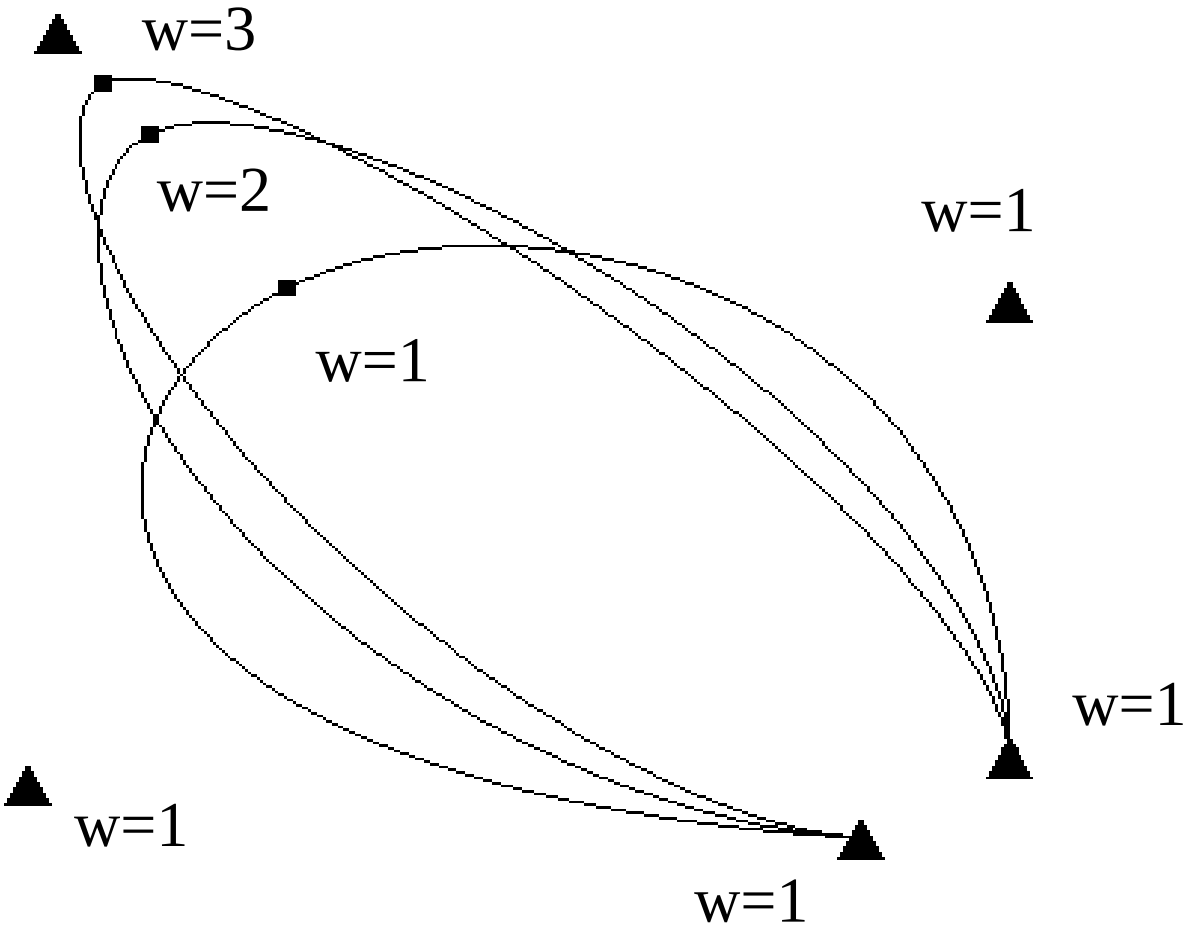


Polinom tört!
racionális

$$\mathbf{r}(t) = \frac{\Sigma\,(w_i B_i(t)\,\mathbf{r}_i)}{\Sigma\,w_j B_j(t)} = \Sigma \frac{w_i B_i(t)}{\Sigma\,w_j B_j(t)}\,\mathbf{r}_i$$

$B_i^*(t)$

NURBS súly



Nurbs görbe OpenGL implementációja

Adattagok:

```
GLUnurbsObj *theNurb; //NURBS objektum
ORDER //NURBS rendje – ellentétben
  Bézierrel, ez tőlünk függő szabad
  partaméter, nálunk legyen konst 3!
GLfloat ctrlpoints[MAXPTNUM][3];
  //kontrollpontok
GLfloat knots[MAXPTNUM+ORDER];
  //kiértékelés paraméterértékeit
  tartalmazó vektor
```

Nurbs görbe létrehozása

```
theNurb=gluNewNurbsRenderer();
gluNurbsProperty(theNurb,GLU_SAMPLIN_TOLERANCE, 25.0);
gluNurbsProperty(theNurb, GLU_DISPLAY_MODE, GLU_FILL);
ptnum= <<aktuális kontroll pontok
  száma>>//knot vektor hasznos része
  mindig ptnum+ORDER elemű
//Knot inicializáció: töltsük fel a KNOT
  vektor első ptnum+ORDER elemét osszuk be
  a [0 1] intervallumot egyenletesen
```

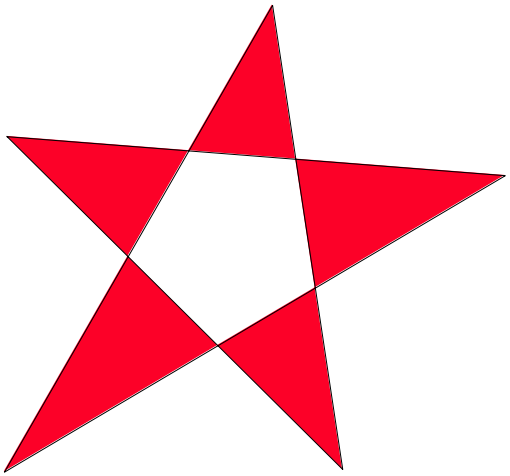
Nurbs görbe OpenGL implementációja

```
theNurb=gluNewNurbsRenderer();
…
gluNurbsProperty(theNurb,GLU_SAMPLIN_T
  OLERANCE, 25.0);
gluNurbsProperty(theNurb,
  GLU_DISPLAY_MODE, GLU_FILL);
```

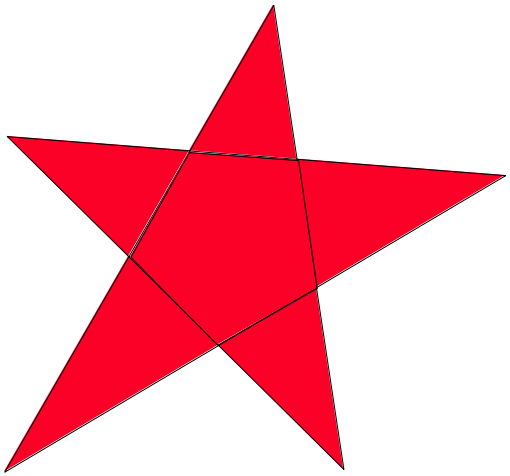
Területek

Határ + belső tartományok azonosítása

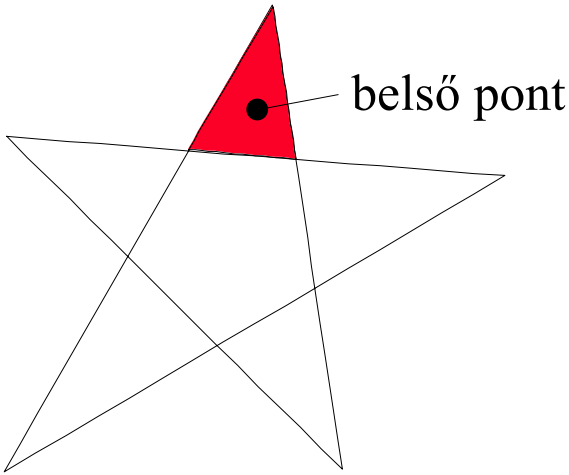
Belső tartományok:



1)



2)



3)

Felületek

- Felület 3D pontok halmaza:
  - koordinátáik kielégítenek egy egyenletet
  - implicit: $f(x, y, z) = 0$
    - gömb: $(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 - r^2 = 0$
  - paraméteres: $x = x(u,v)$, $y = y(u,v)$, $z = z(u,v)$,

    $u,v \in [0,1]$
    - gömb: $x = x_0 + r \cos 2\pi u \sin \pi v$

      $y = y_0 + r \sin 2\pi u \sin \pi v$

      $z = z_0 + r \cos \pi v \qquad u,v \in [0,1]$
- Klasszikus felületek
  - definíció = paraméterek megadása

Kvadratikus felületek

- $\underline{x}^T \mathbf{A} \underline{x} = 0$ $\underline{x}^T = [x, y, z, 1]$
- A koordináták legfeljebb másodfokon
- gömb, ellipszoid, sík, paraboloid, hiperboloid, hengerfelület,...

Ellipszoid

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} - 1 = 0$$

Végtelen kúp

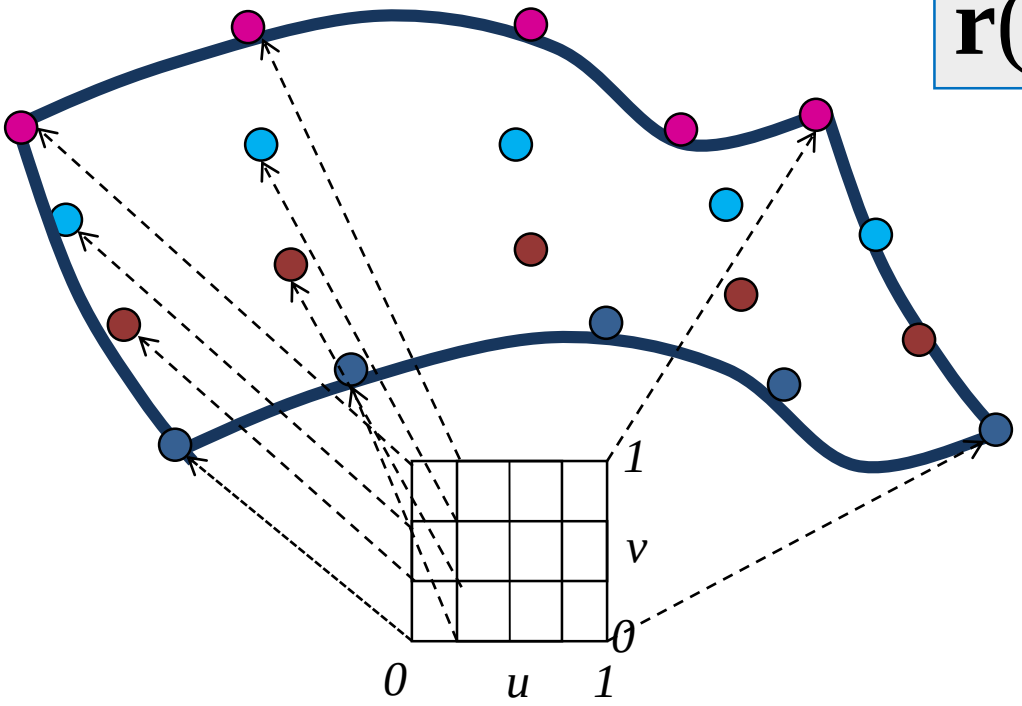
$$\frac{x^2}{a^2} + \frac{y^2}{b^2} - z^2 = 0$$

Végtelen henger

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} - 1 = 0$$

Szabadformájú felületek: r(u,v)

- Definíció kontrolpontokkal



$$\mathbf{r}(u,v) = \Sigma\,\Sigma\, B_{ij}(u,v)\,\mathbf{r}_{i,j}$$

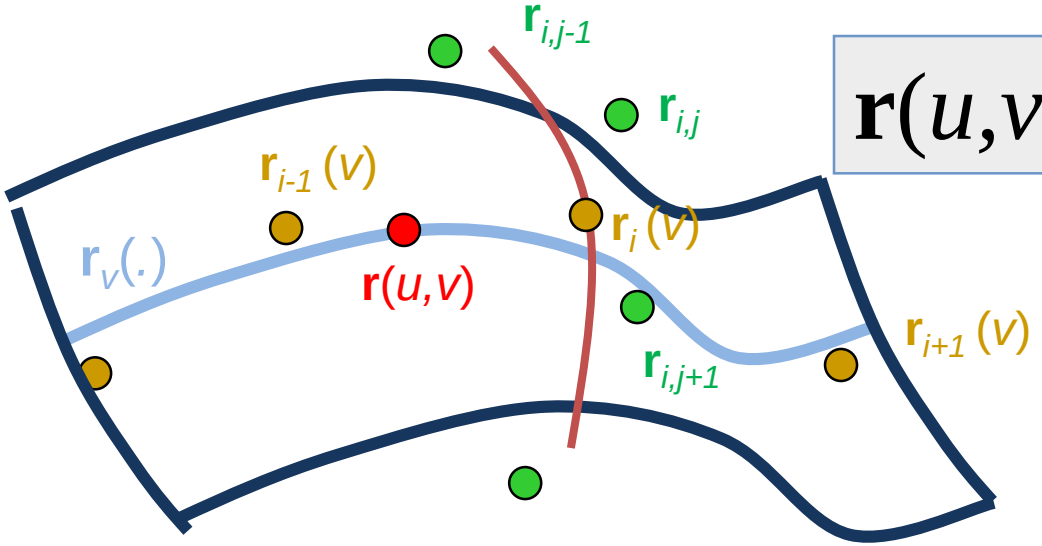
$$u,v \in [0,1]$$

$$\Sigma_{ij}\, B_{ij}(u,v) = 1 \text{ minden u,v-re}$$

- Egységnégyzet leképezése a felületre (2D csomópontmátrix)
- Kontrolpontok 2D tömbbe rendezettek

Szorzatfelületek

- Definíció kontrolpontokkal

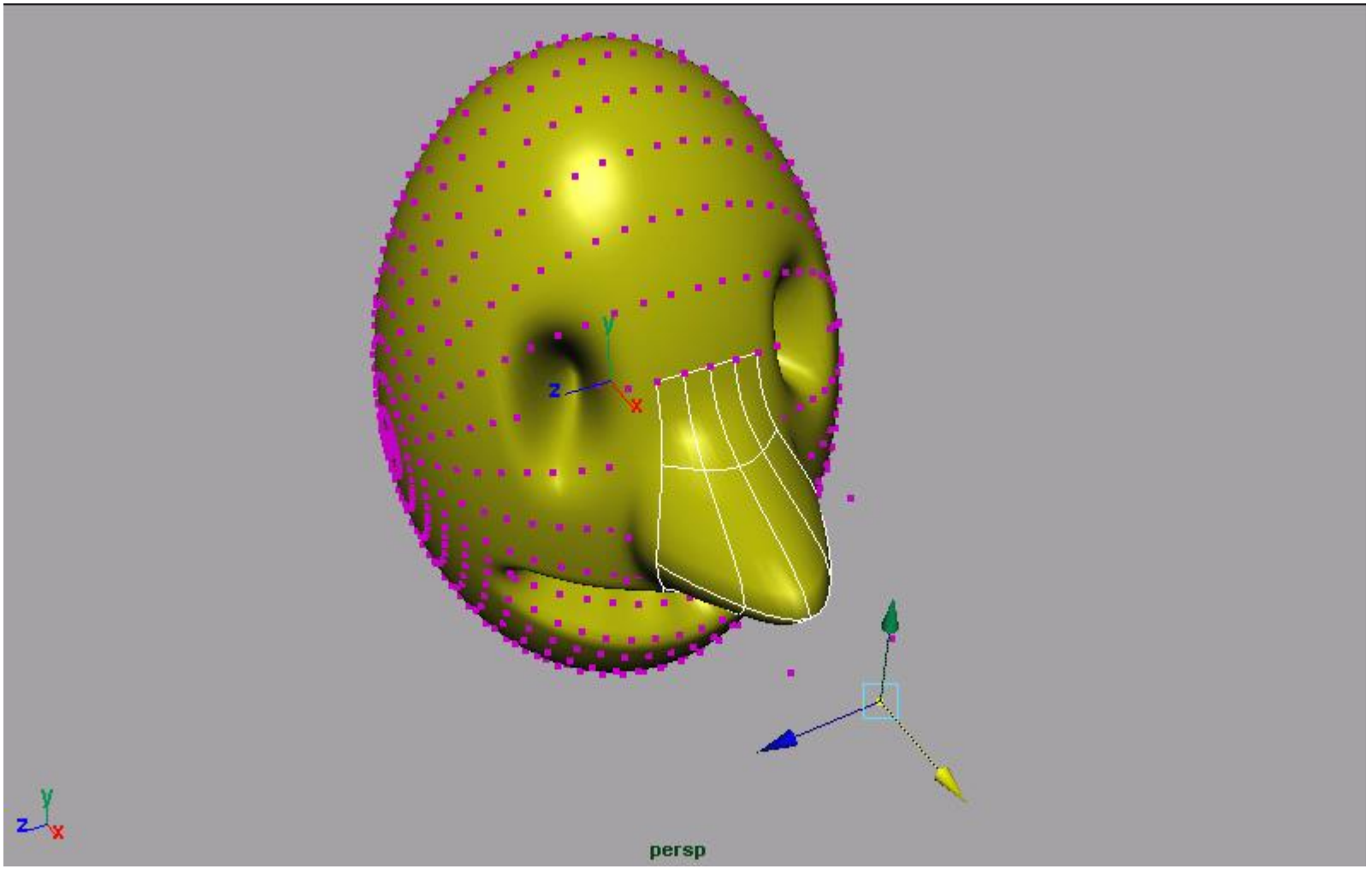


$$\mathbf{r}(u,v) = \Sigma\,\Sigma\, B_i(u)\, B_j(v)\, \mathbf{r}_{i,j}$$

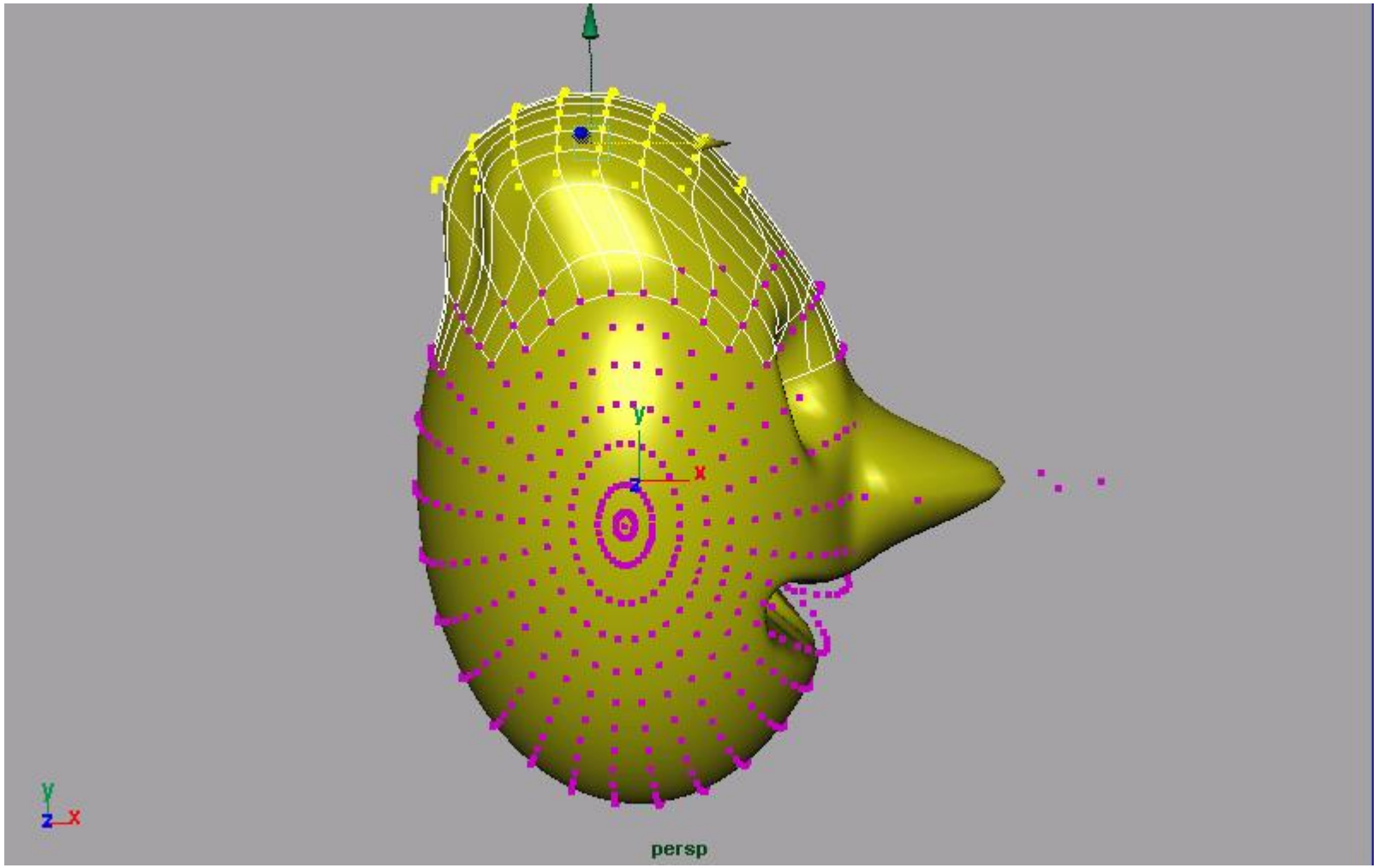
$$\mathbf{r}(u,v) = \mathbf{r}_v(u) = \Sigma_i\, B_i(u)\, \mathbf{r}_i(v)$$

$$\mathbf{r}_i(v) = \Sigma_j\, B_j(v)\, \mathbf{r}_{i,j}$$

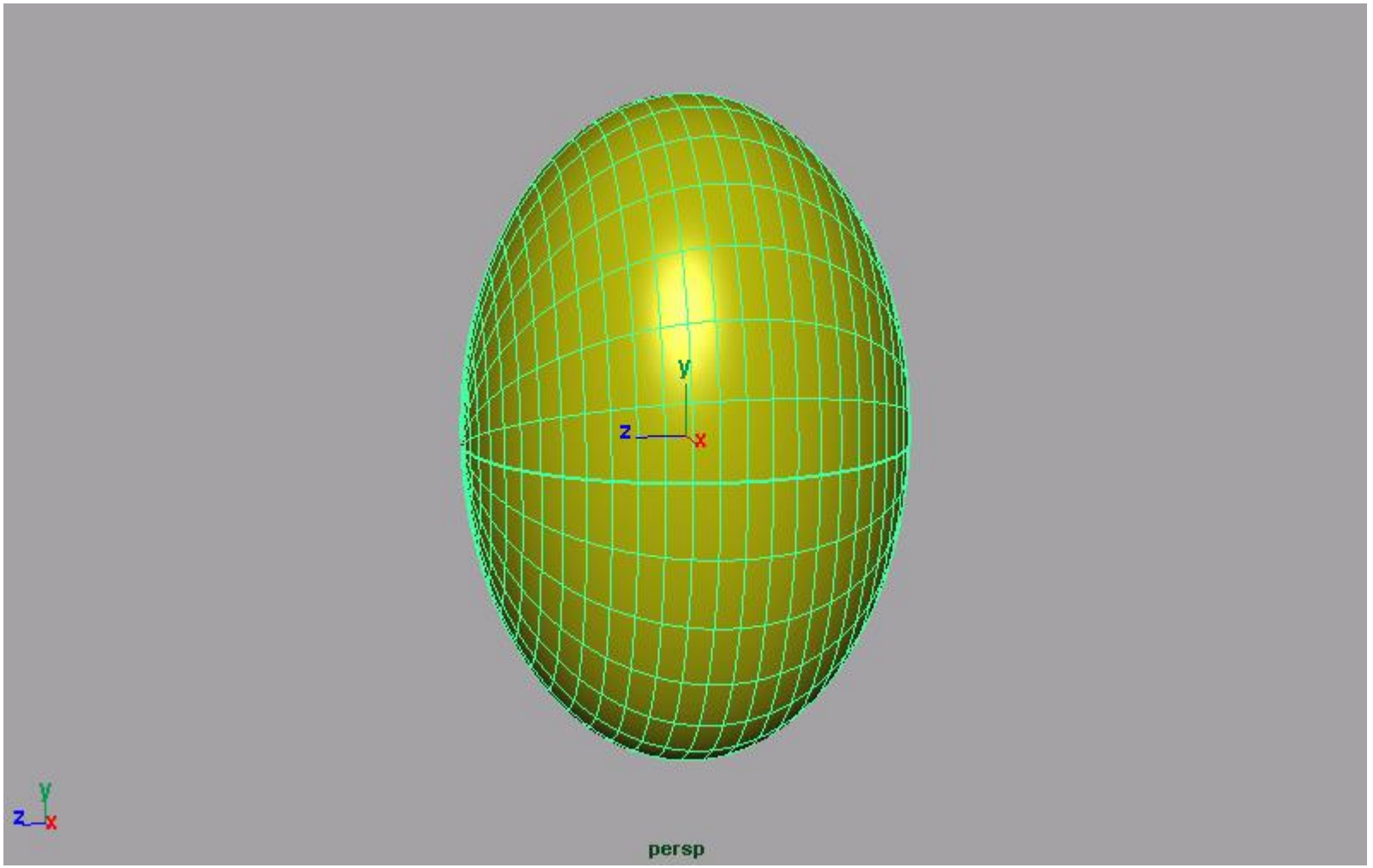
Vezérlőpontok, súlyok módosítása



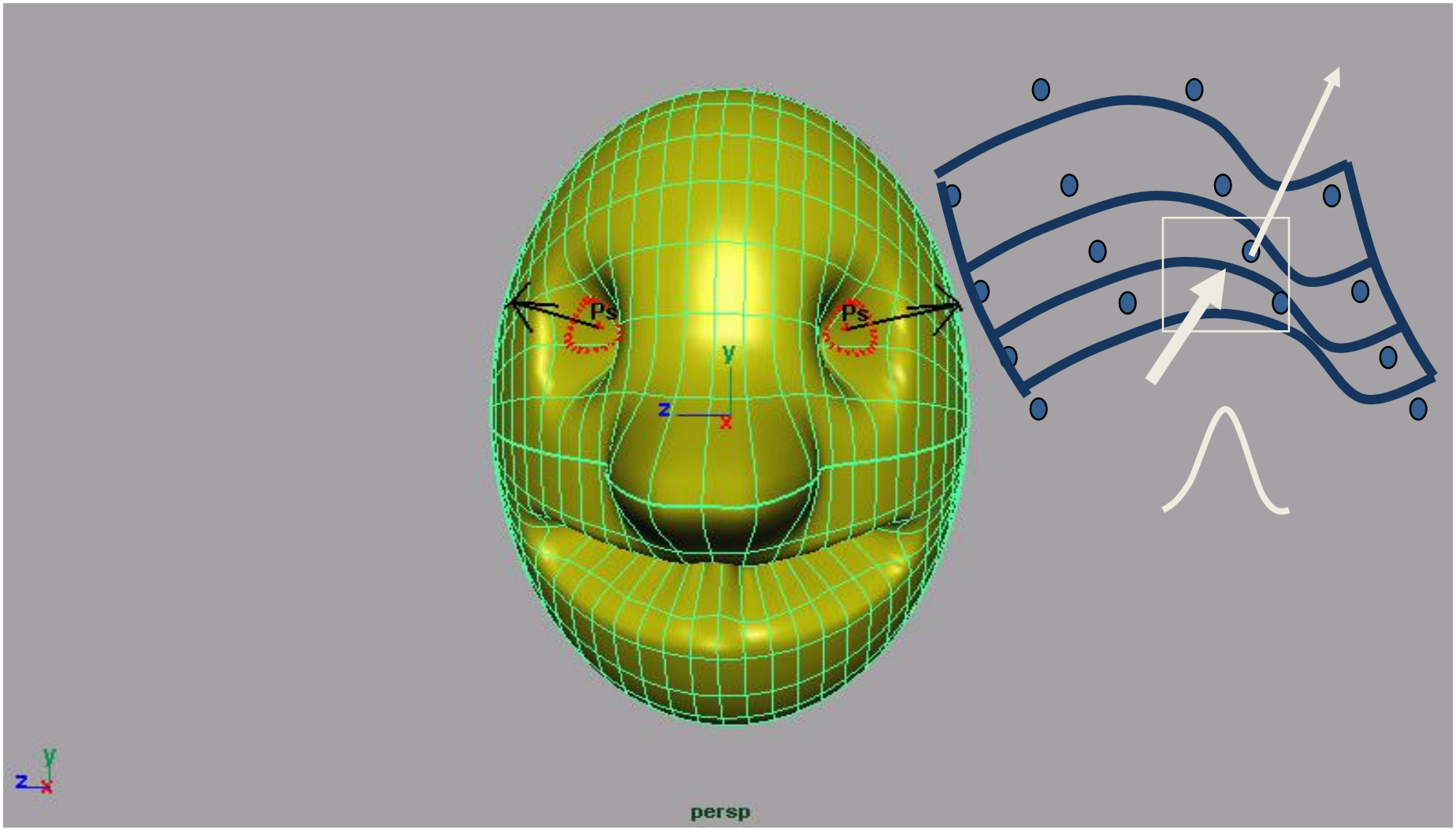
Vezérlőpontcsoportok módosítása



Szobrászkodás szabadformájú felületekkel



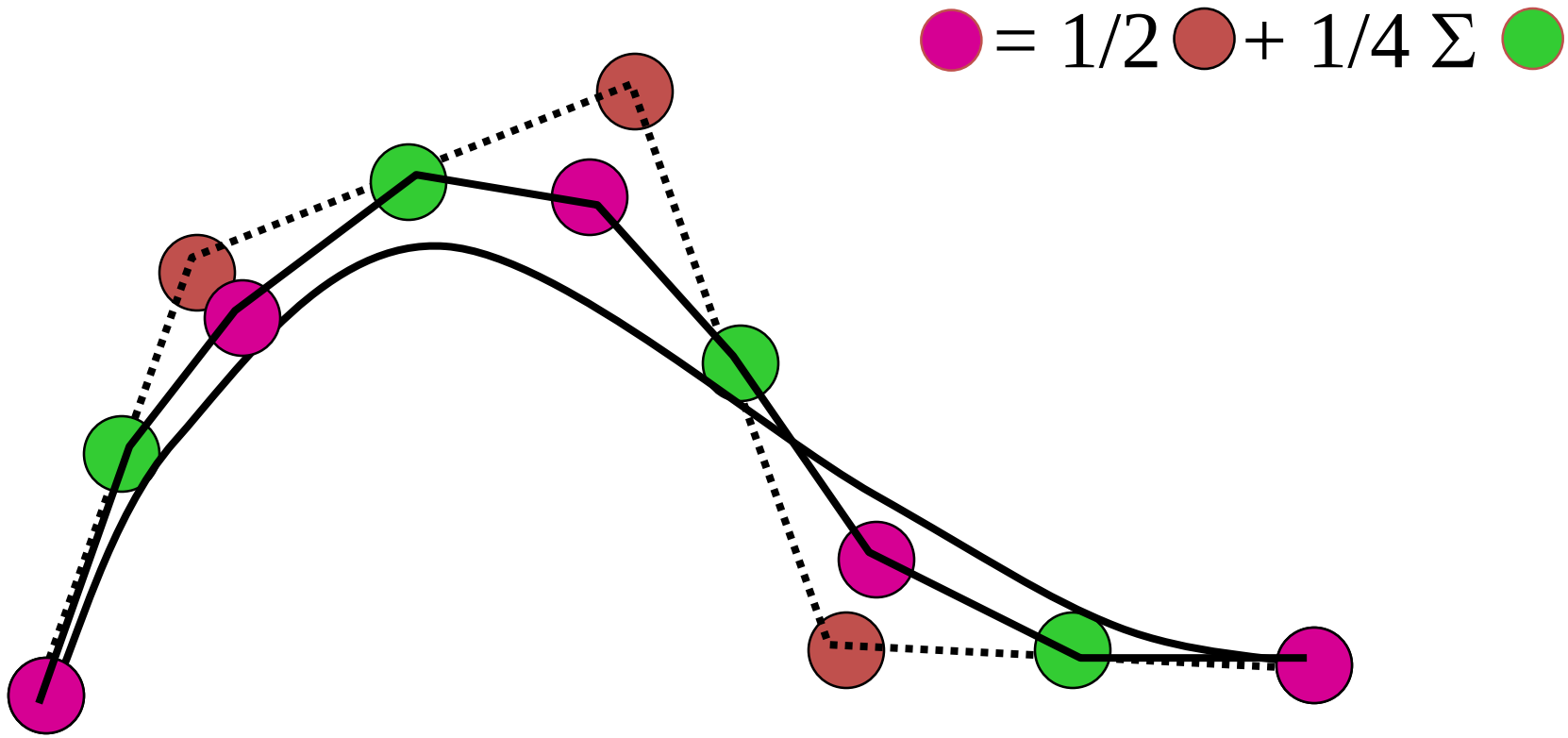
Szobrászkodás szabadformájú felületekkel



Felosztásos (subdivision) módszerek

- Durva poligonmodell simítása
  - NURBS illesztés, majd finomabb poligon közelítés
  - közvetlen finomítás (subdivision)

Felosztásos (subdivision) módszerek

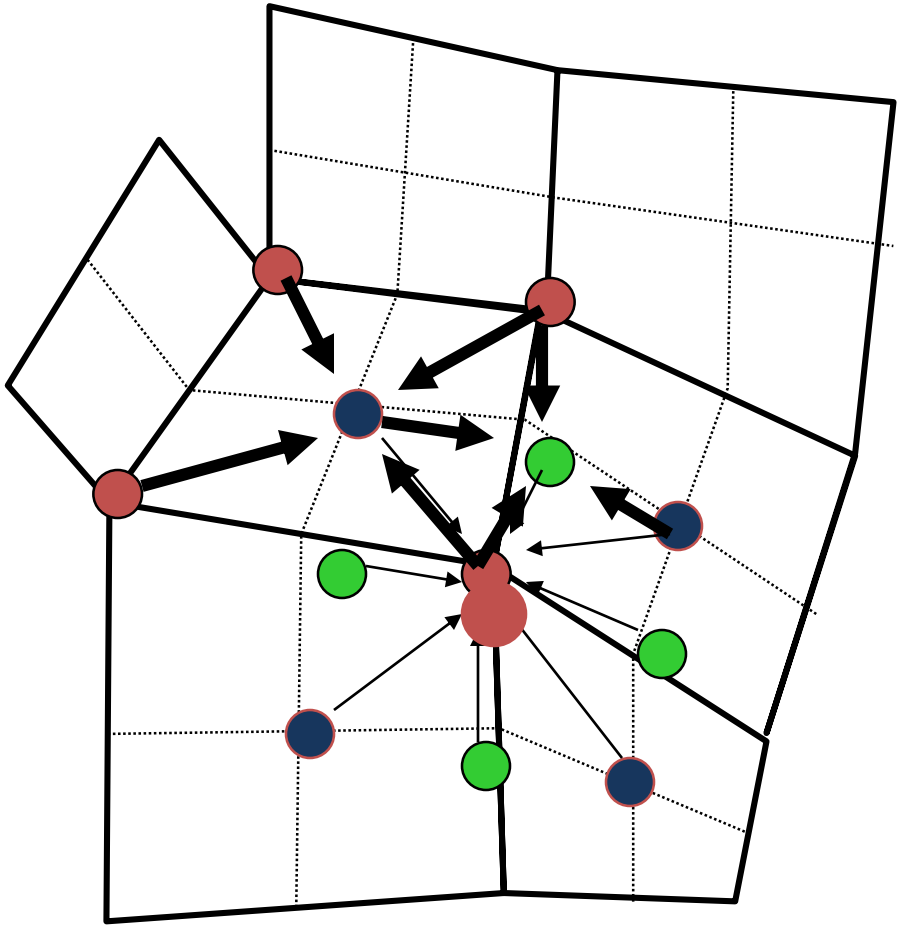


Subdivision felületek (Catmull-Clark)

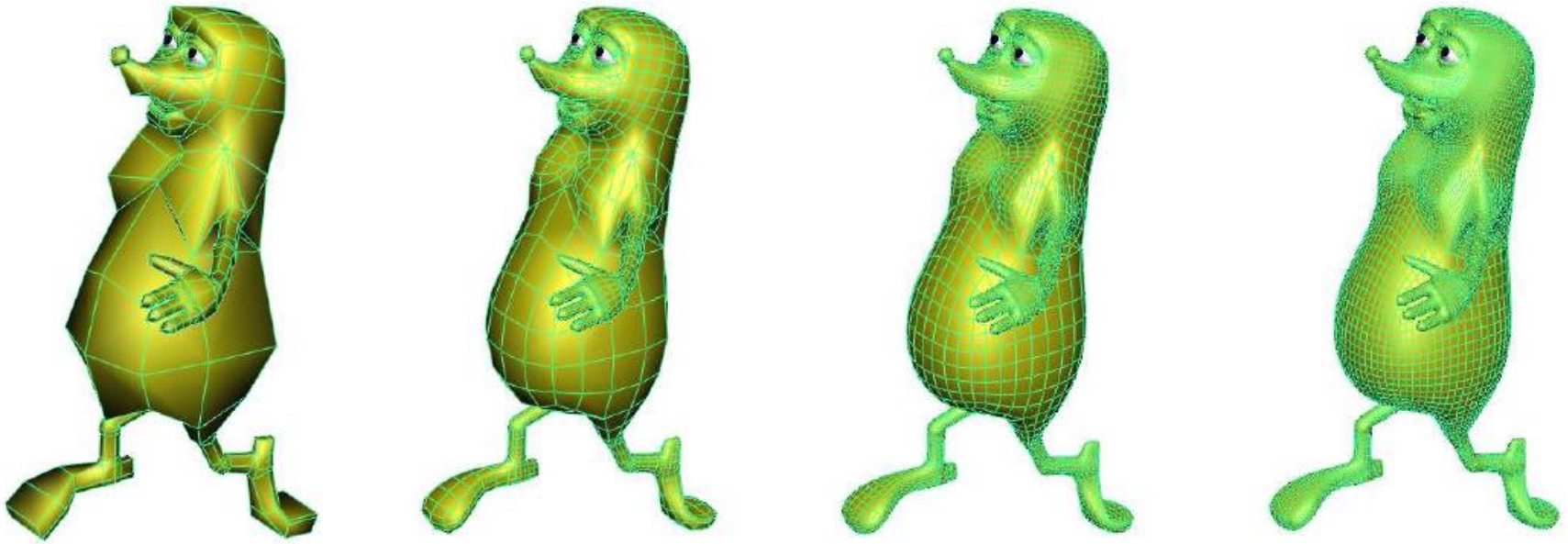
● = 1/4 Σ ●

● = 1/4 Σ ● + 1/4 Σ ●

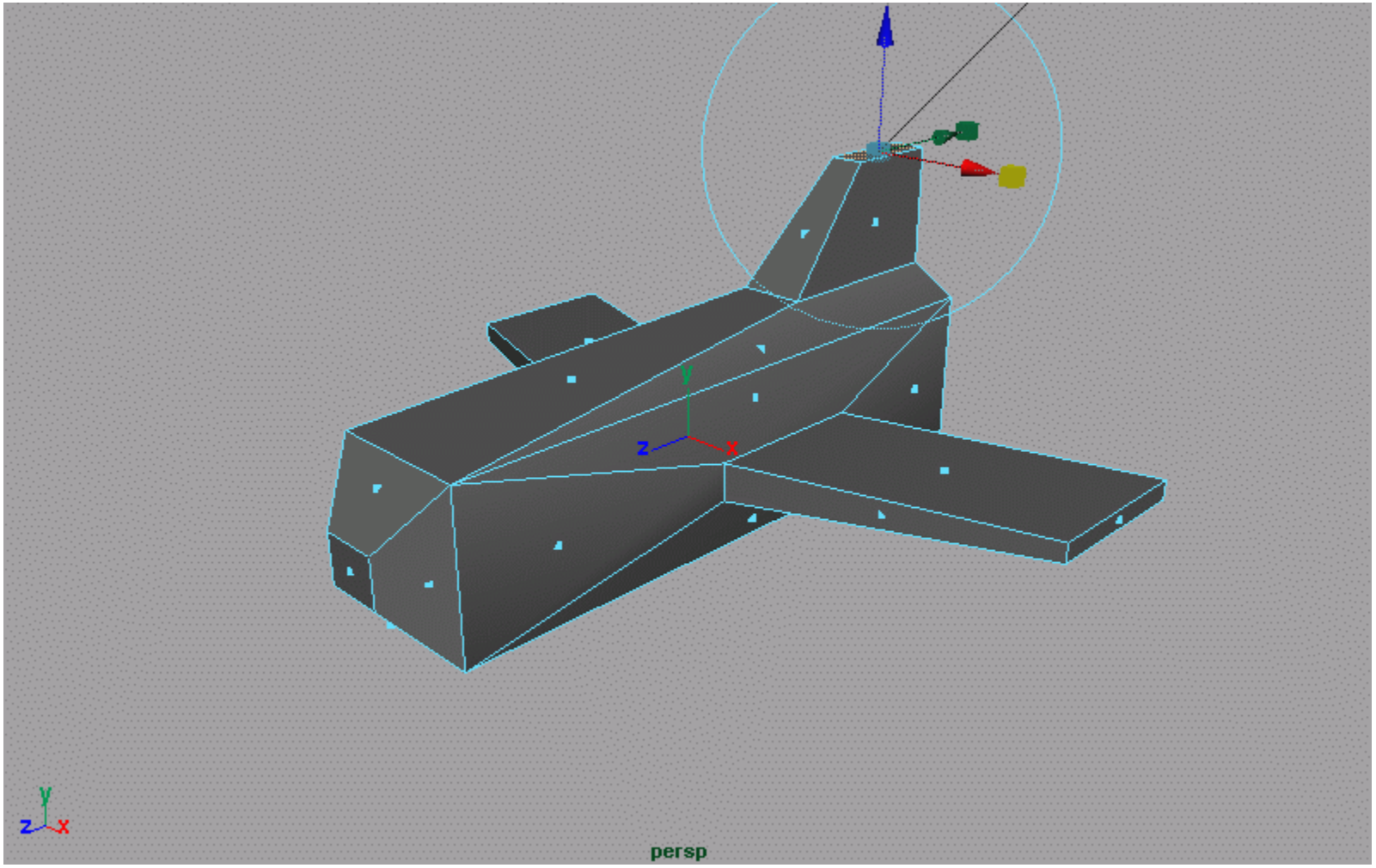
● = 1/2 ● + 1/16 Σ ●
+ 1/16 Σ ●



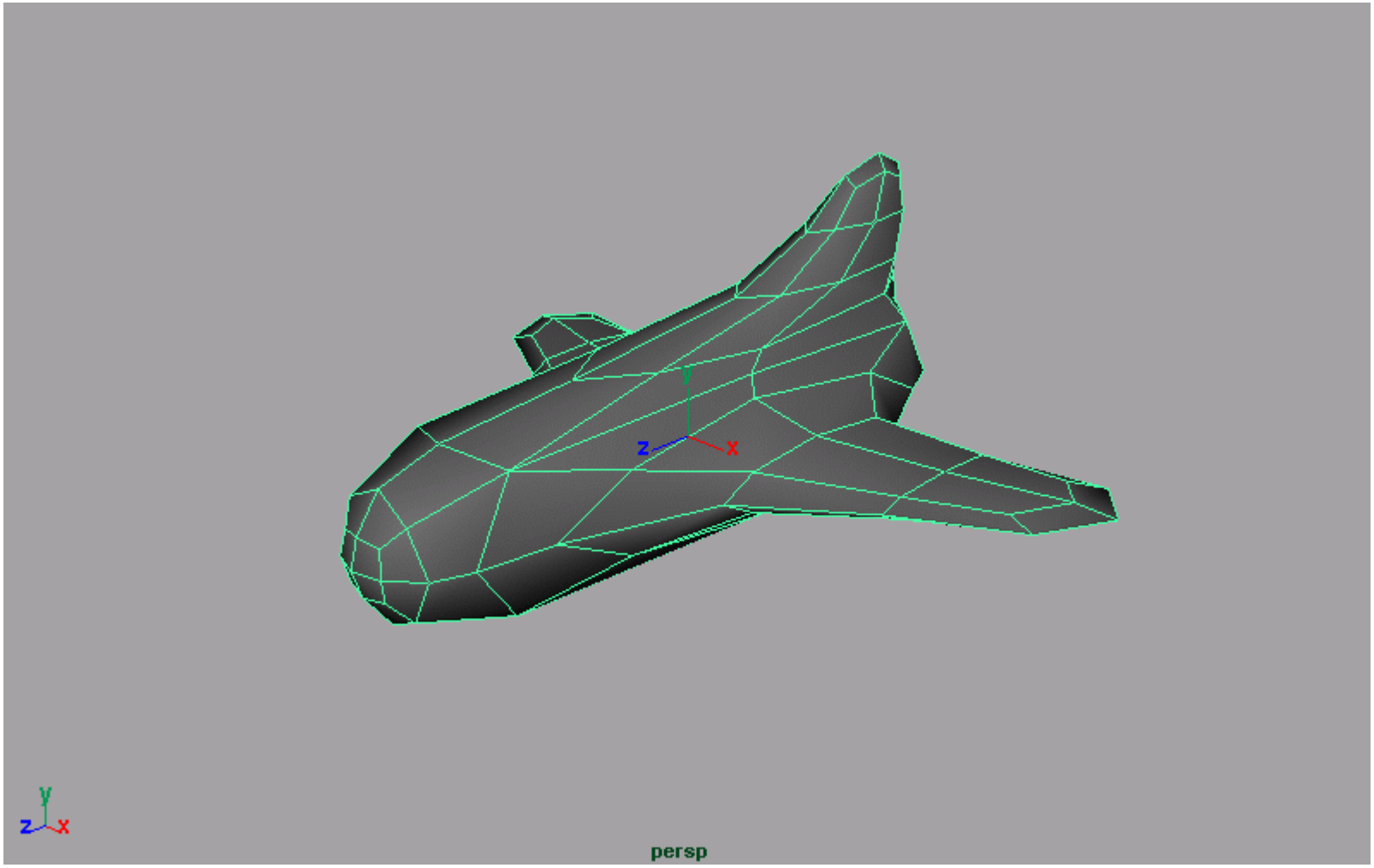
Subdivision felületek (Catmull-Clark)



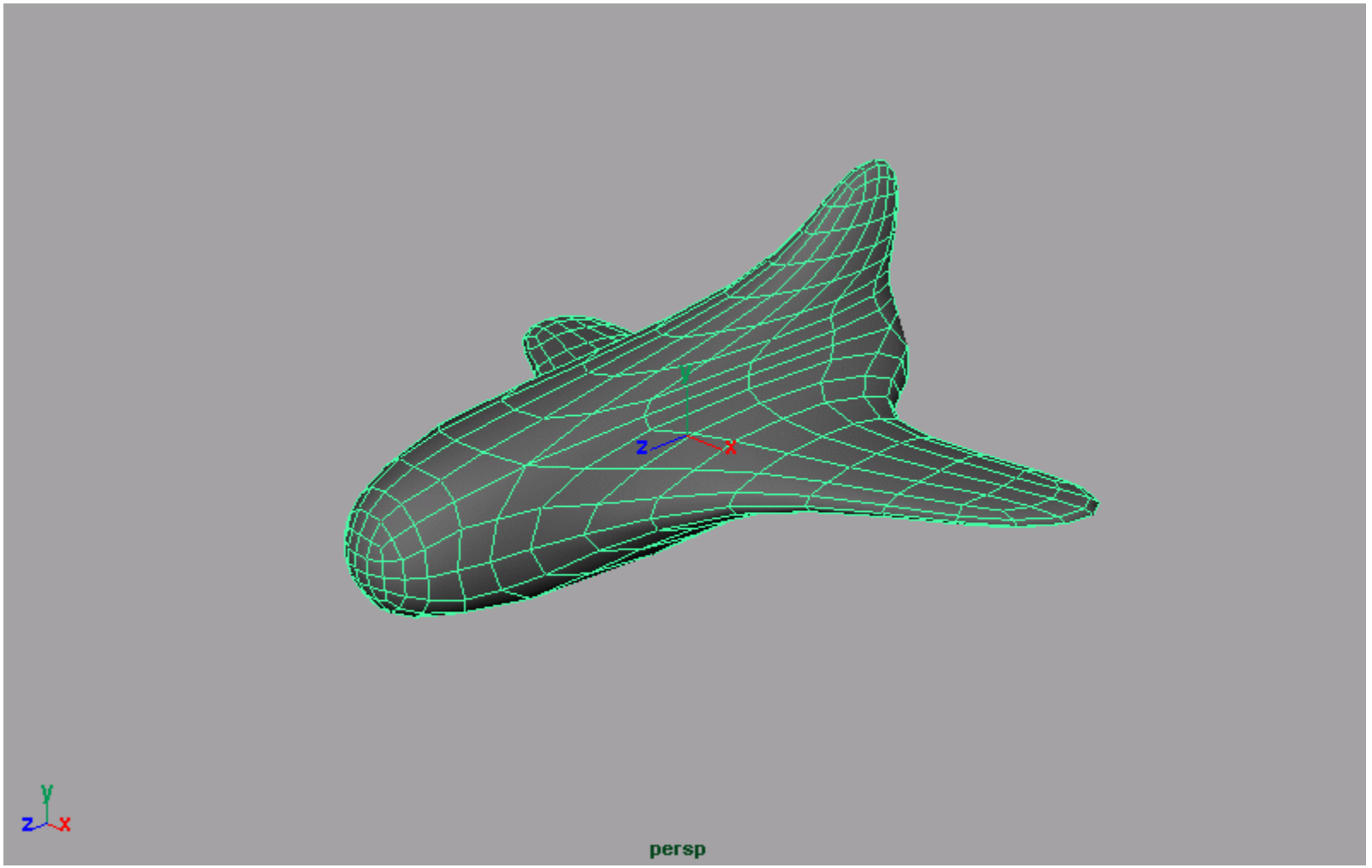
Durva poligon modell



Subdivision simítás: 1 szint



Subdivision simítás: 2. szint



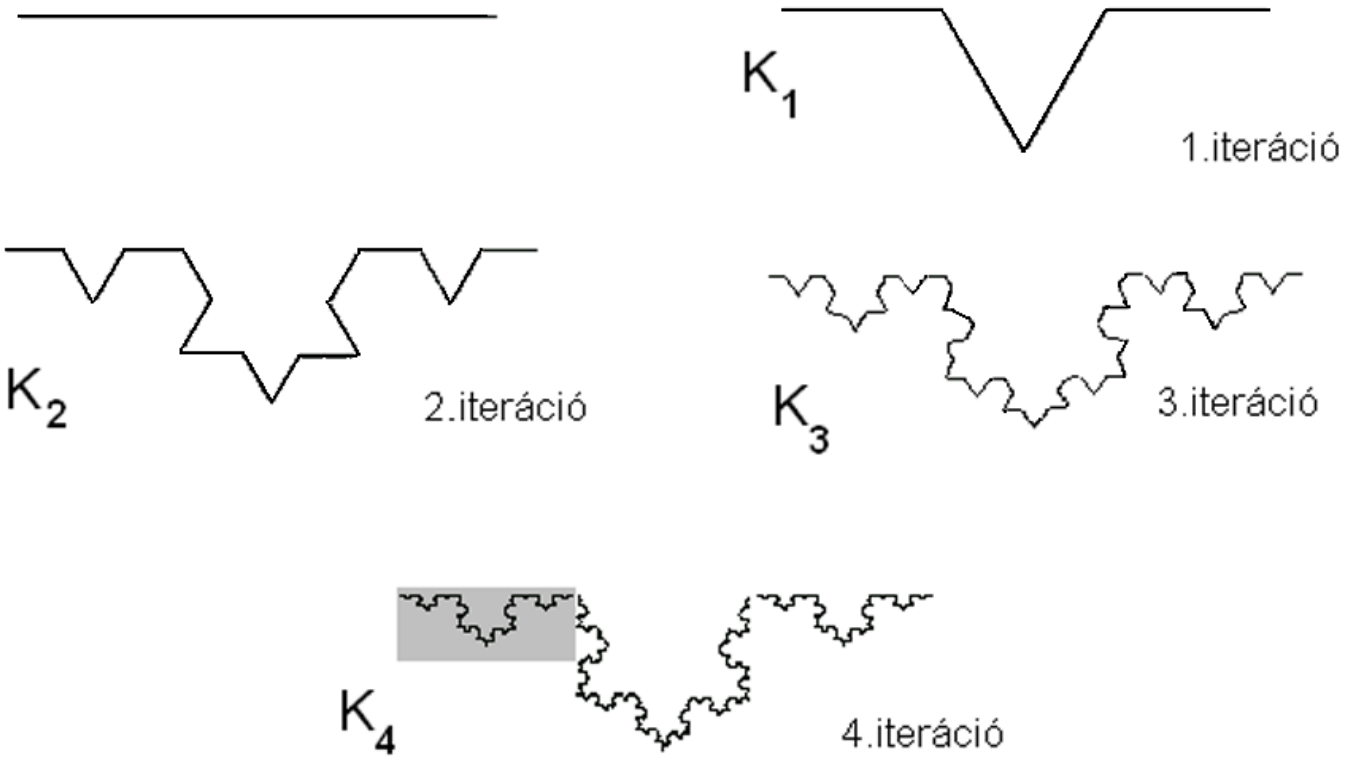
Progresszív hálók

- „Túl finom” poligonháló – nagy méret
- Közelítés kevesebb poligont tartalmazó hálóval
- Hoppe-féle progresszív háló: élzsugorítások sorozata
  - a poligonhálót legkevésbé módosító élt töröljük
  - élek prioritása – a legkisebb törlendő
  - heurisztika pl: tartsuk meg azokat az éleket akik hosszúak, illetve a rájuk illeszkedő lapok normálisa által bezárt szög nagy (nem garantálja a topológia megtartását)

Progresszív hálók

- Egyszerűsítés előnye:
  - kisebb leíró adatmennyiség
  - gyorsabb képszintézis (pl játék)
    - több részletezettségi szintet alkalmazó geometriai modellek
- Progresszív tárolás:
  - tároljuk a durva hálót és az egyszerűsítés műveletek inverzét
  - alkalmazás: pl lassú hálózati átvitel – először a durva modell érkezik, majd ez fokozatosan finomítható

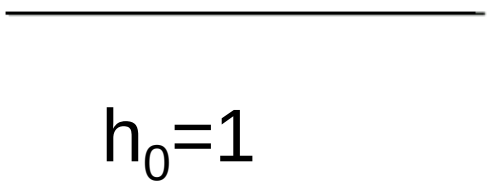
Koch görbe



1.5 ábra

Forrás: Máté László, BME

Koch görbe hossza végtelen



$h_0=1$

$h_1=4/3$

$h_2=(4/3)^2$

$h_3=(4/3)^3$

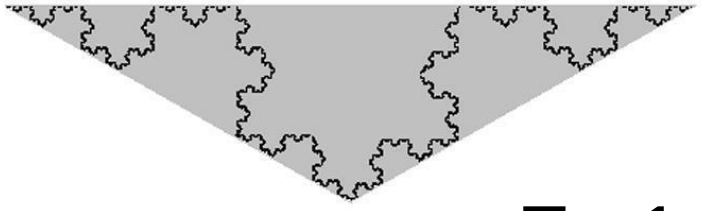
$h_4=(4/3)^4$

Görbe „hossza”:

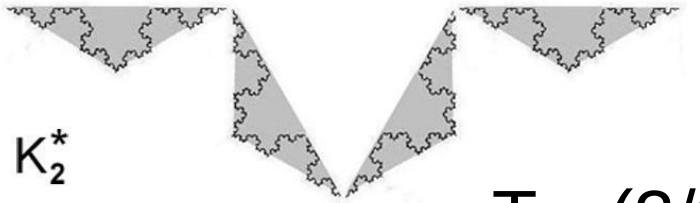
$h_{Koch}=\lim_{i\to\infty} h_i=$

$\lim_{i\to\infty}(4/3)^i=\infty$

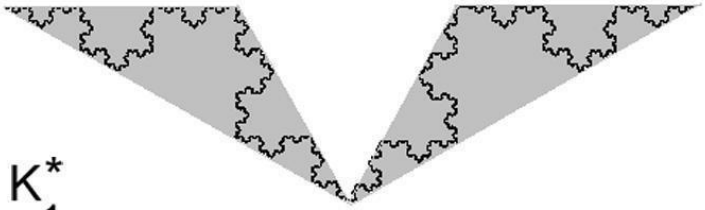
Koch görbe „területe” 0



$T_0=1$



$T_2=(2/3)^2$



$T_1=2/3$

$T_3=(2/3)^3$

Minden sorozatelem tartalmazza a teljes görbét

Görbe „területe”:

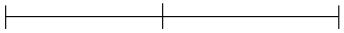
$T_{Koch} \leq \lim_{i\to\infty} T_i =$

$\lim_{i\to\infty} (2/3)^i = 0$

Fraktálok

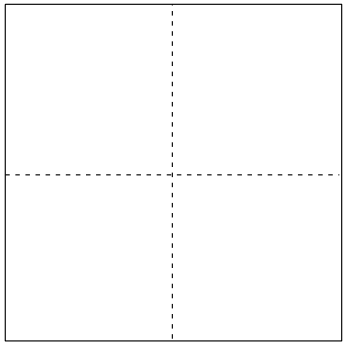
Hausdorff dimenzió

N = 2



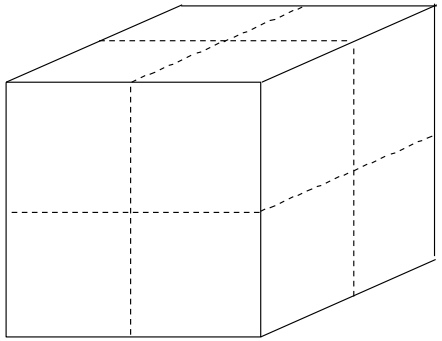
$r = \frac{1}{2}$

N = 4



$r = \frac{1}{2}$

N = 8



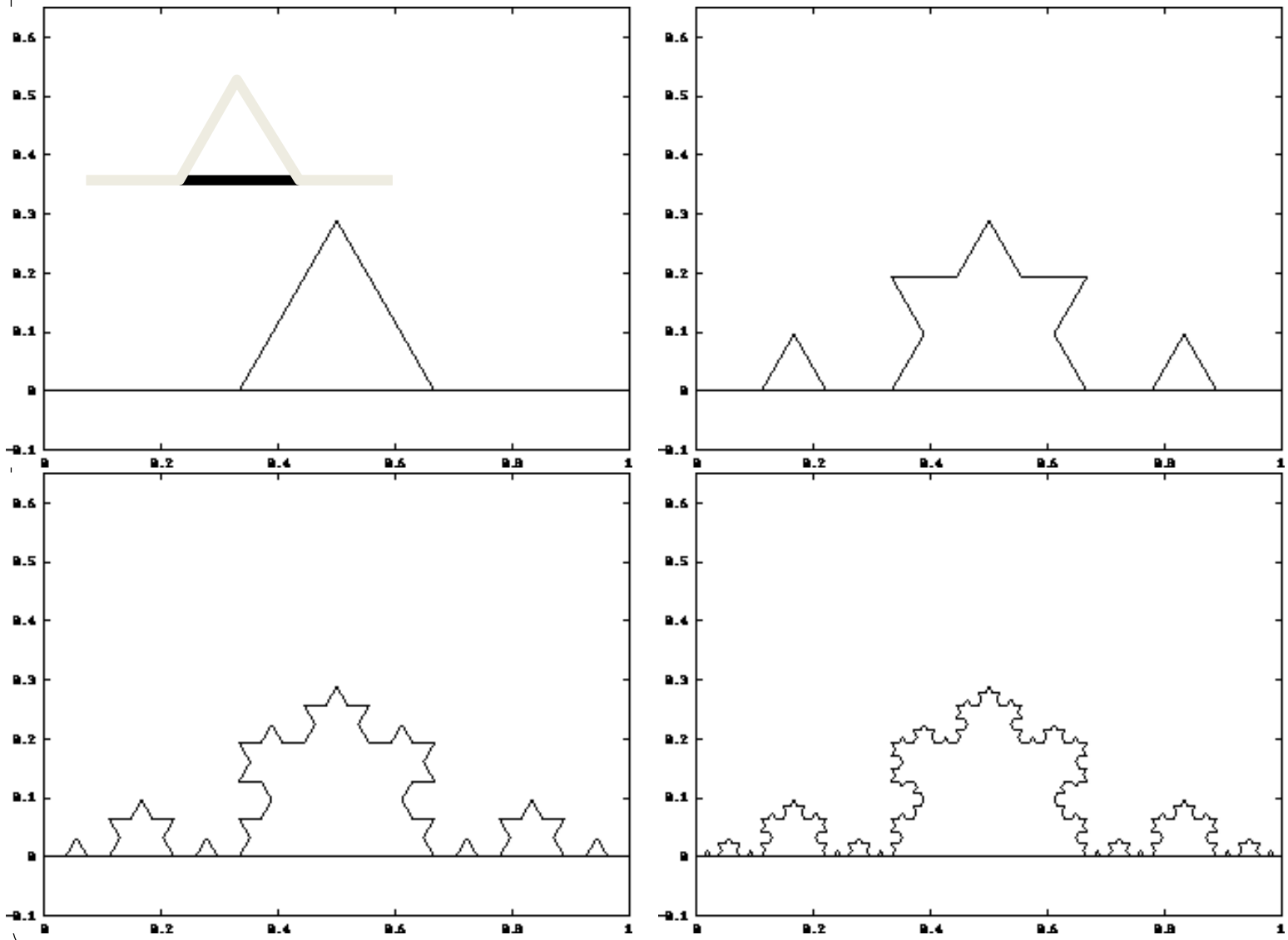
$r = \frac{1}{2}$

$N = 1/r^D$

$D = (\log N) / (\log 1/r)$

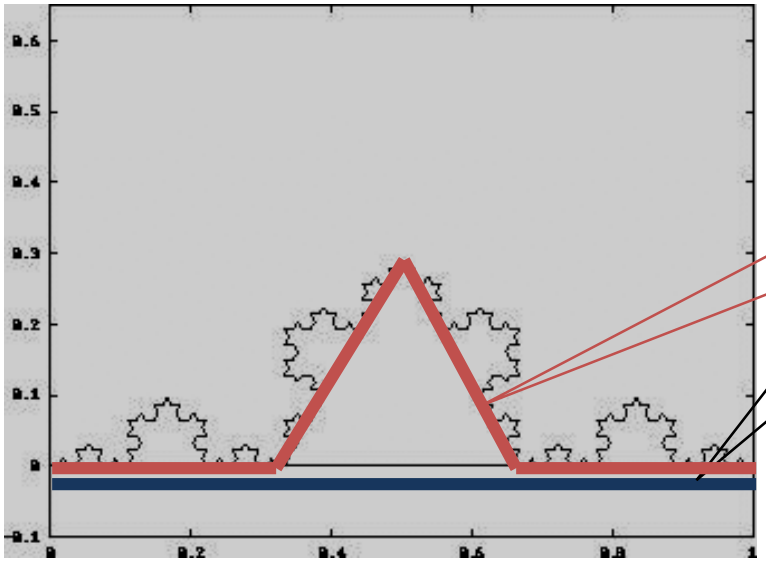
Koch görbe

$$D = (\log 4) / (\log 3) = 1.26$$



N = 4,
r = 1/3

Nem önhasonló objektumok dimenziója



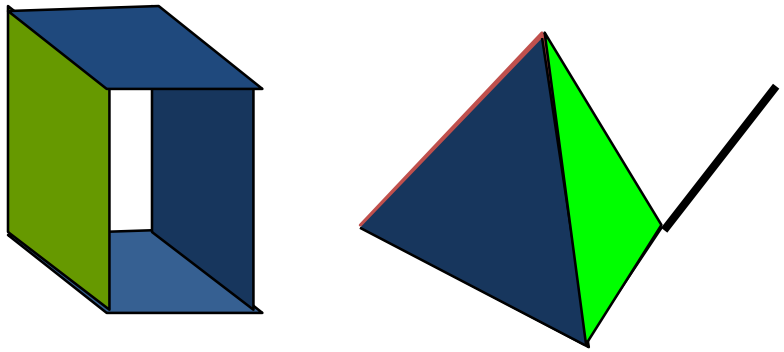
Vonalzó (l)	db
l	1
r =1/3	N = 4
r^2	N^2
r^m	N^m

$$\text{Hossz}(l) = l\ \text{db} = l\ N^m = l\ (1/r^D)^m = l\ (1/r^m)^D = 1/l^{D-1}$$

$$N = 1/r^D$$

$$D = -\log \text{Hossz}(l) / \log l + 1$$

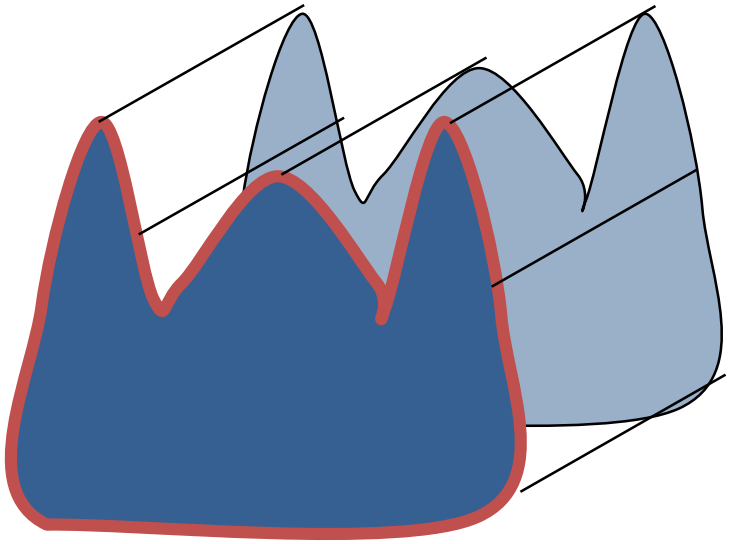
Testek



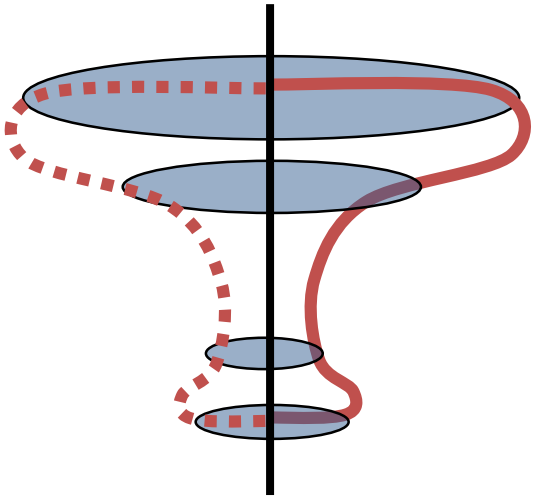
- Ellenpéldák

- Érvényes testek: reguláris halmaz
  - nem lehetnek alacsony dimenziós elfajulásai
  - minden határpont mellett van belső pont
- Garantáltan érvényes testet építő módszerek
  - 2.5 dimenziós eljárások
  - speciális felületi modellezés: B-rep
  - Konstruktív tömörtest geometria

2.5 dimenziós módszerek



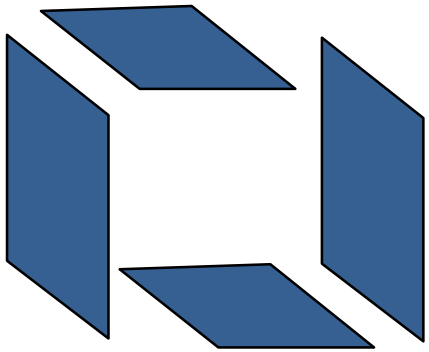
Kihúzás: extrude

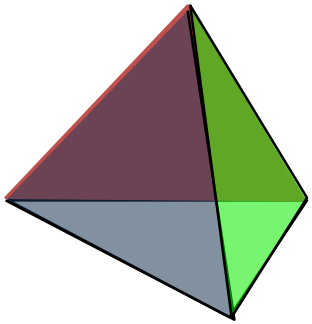


Forgatás: rotate

Felületmodellezők

- Test = határfelületek gyűjteménye



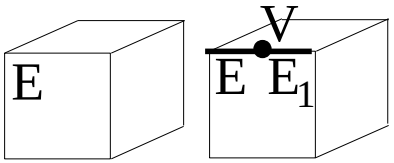


- Topológiai ellenőrzés (Euler tétel):

csúcs + lap = él + 2

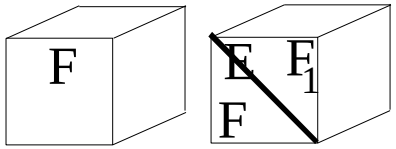
B-rep: Euler operátorok

MVE



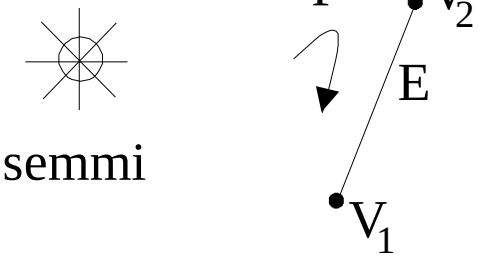
Csinálj csúcsot
és élt

MEF



Csinálj élt
és lapot

MEVVF



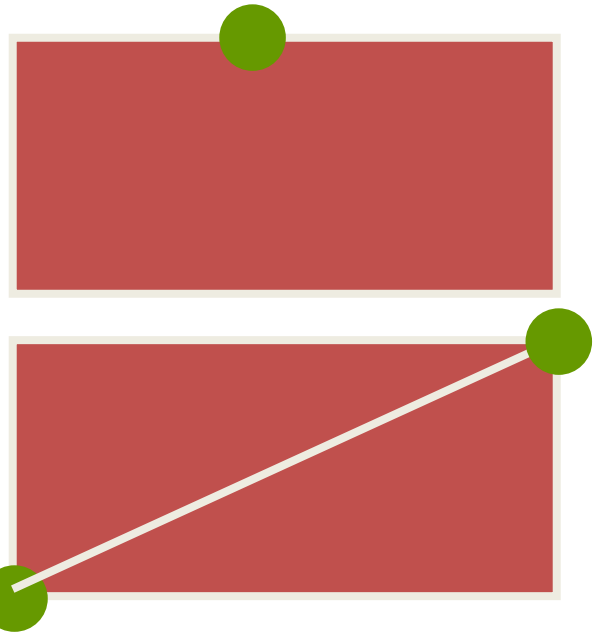
semmi

Csinálj élt,
két csúcsot
és lapot

Gyakorlati Euler operátorok

- Edge split

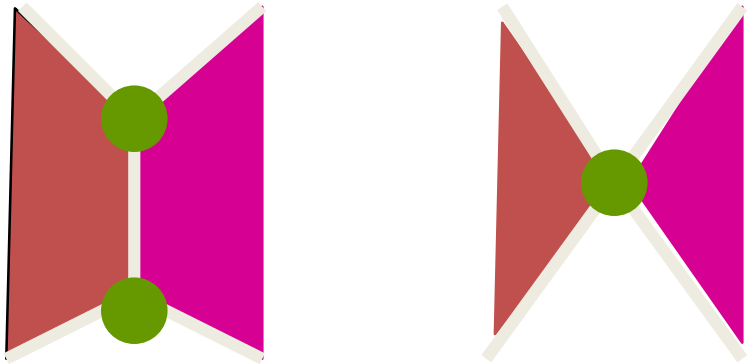
  csúcs + lap = él + 2



- Poligon split

- Élzsugorítás v. csúcspont összevonás

-Edge Collapse

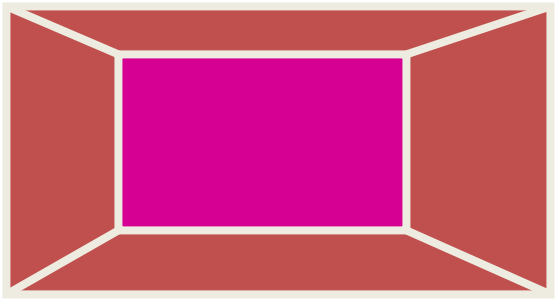


Gyakorlati Euler operátorok

- Poligon kihúzás

(Face extrude):

e_p: a poligon éleinek a száma

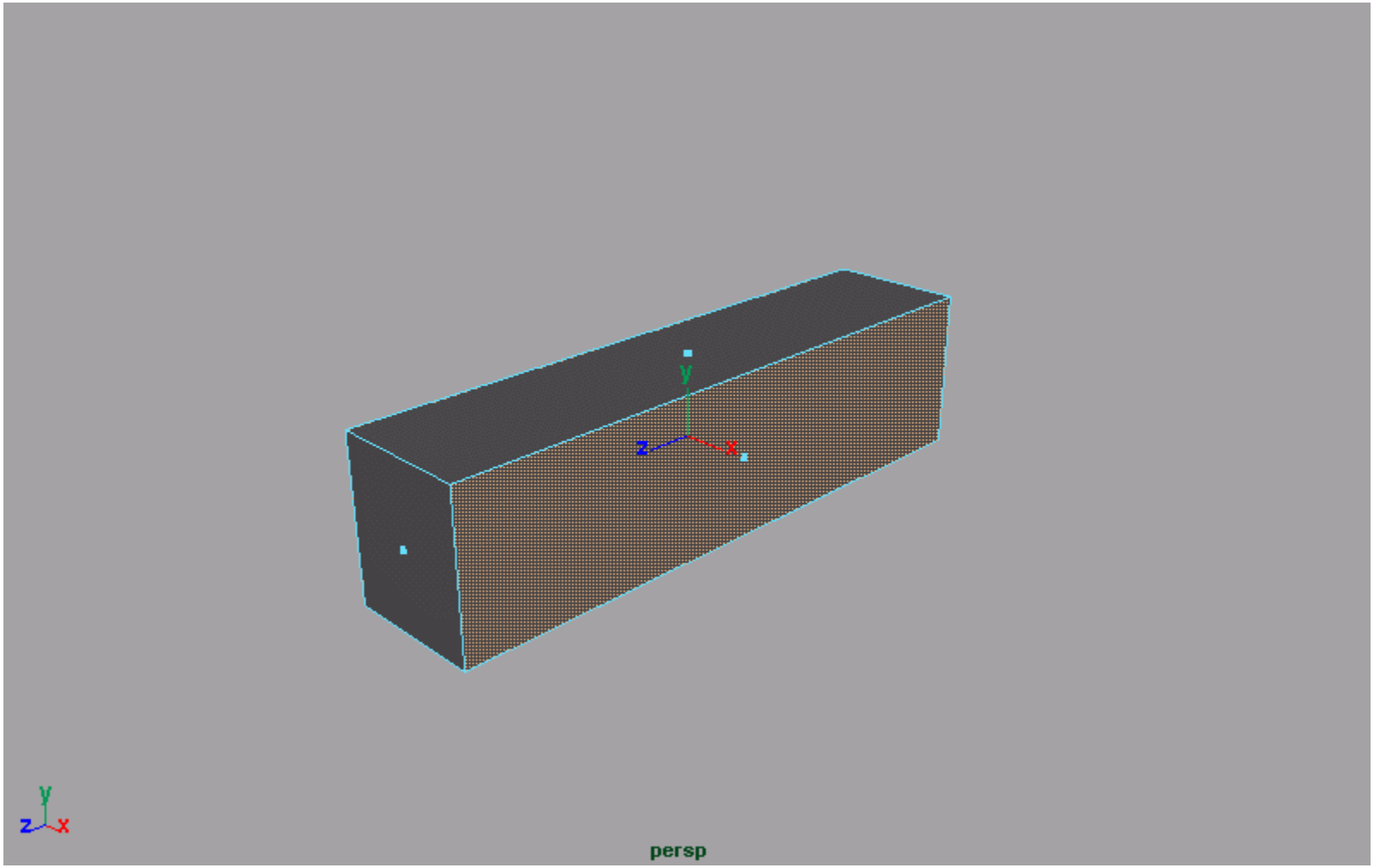


- $2e_p$ új él,
- e_p+1 új lap,
- e_p új csúcs
- 1 eltűnő lap

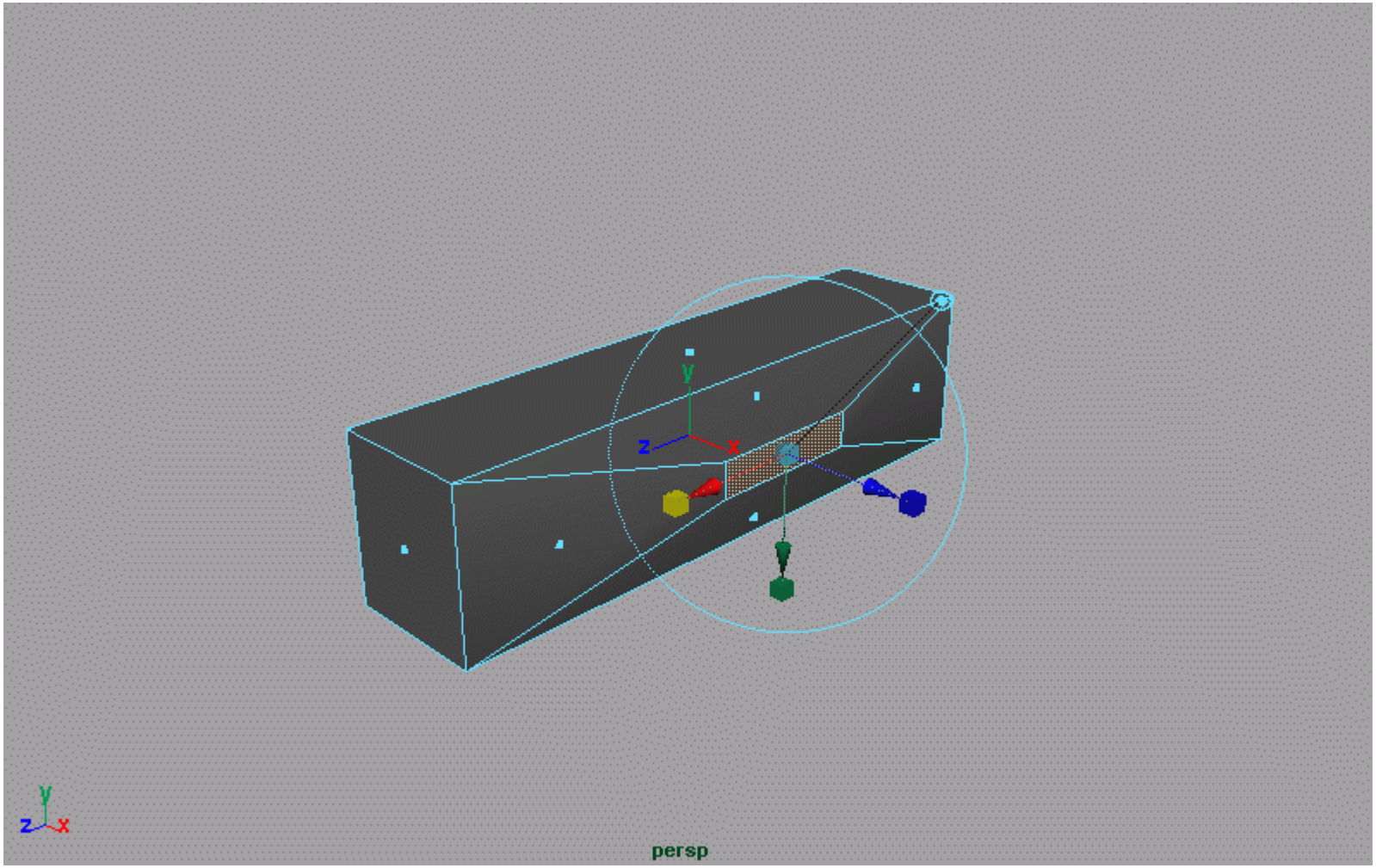
$$e'=e+2e_p \quad l'=l+e_p+1-1 \quad c'=c+e_p$$

$$l'+c'=l+c+2e_p=e+2+2e_p=e'+2$$

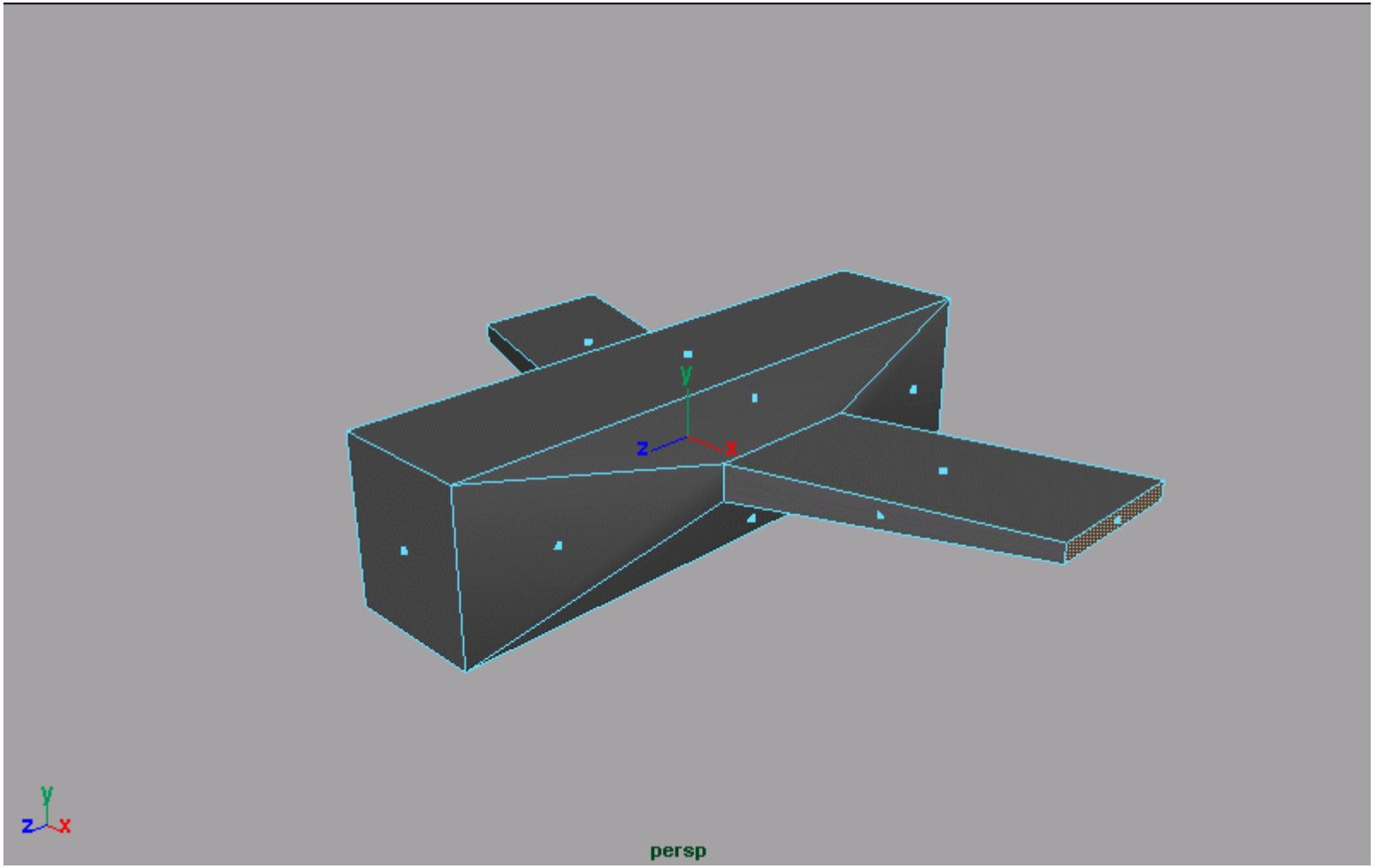
Poligon modellezés: téglatest



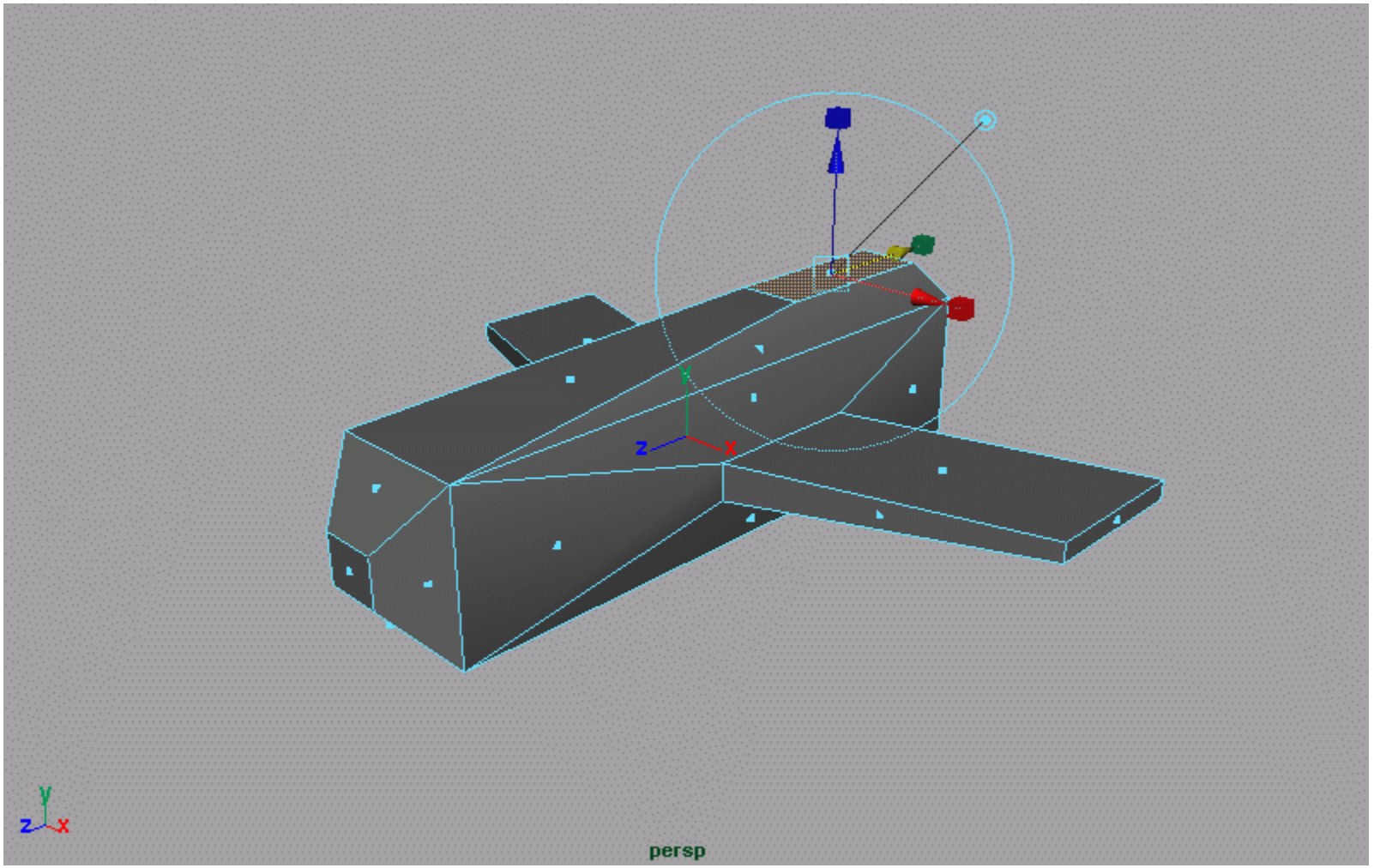
Poligon modellezés:
1. extruding



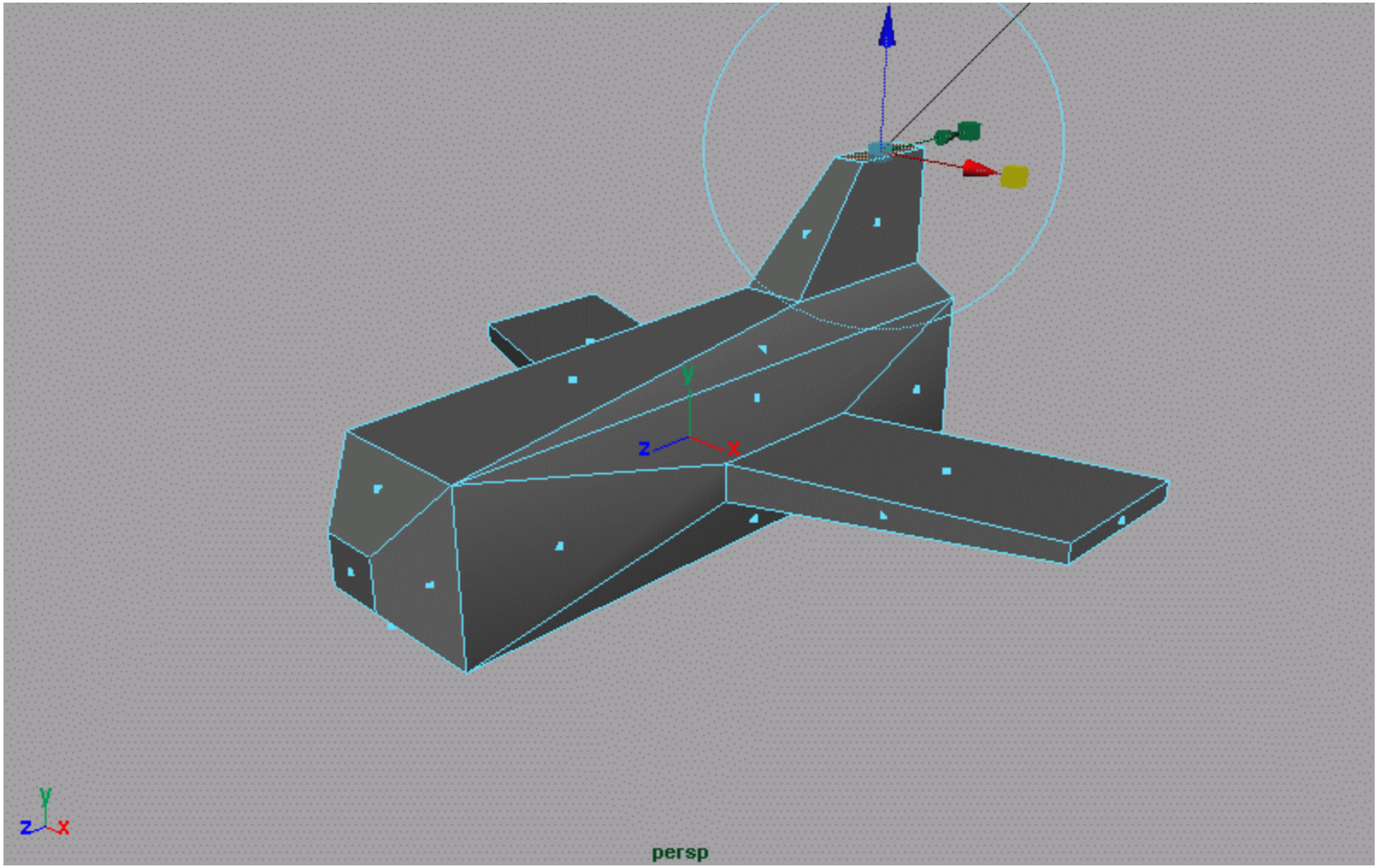
Poligon modellezés:
2. extruding



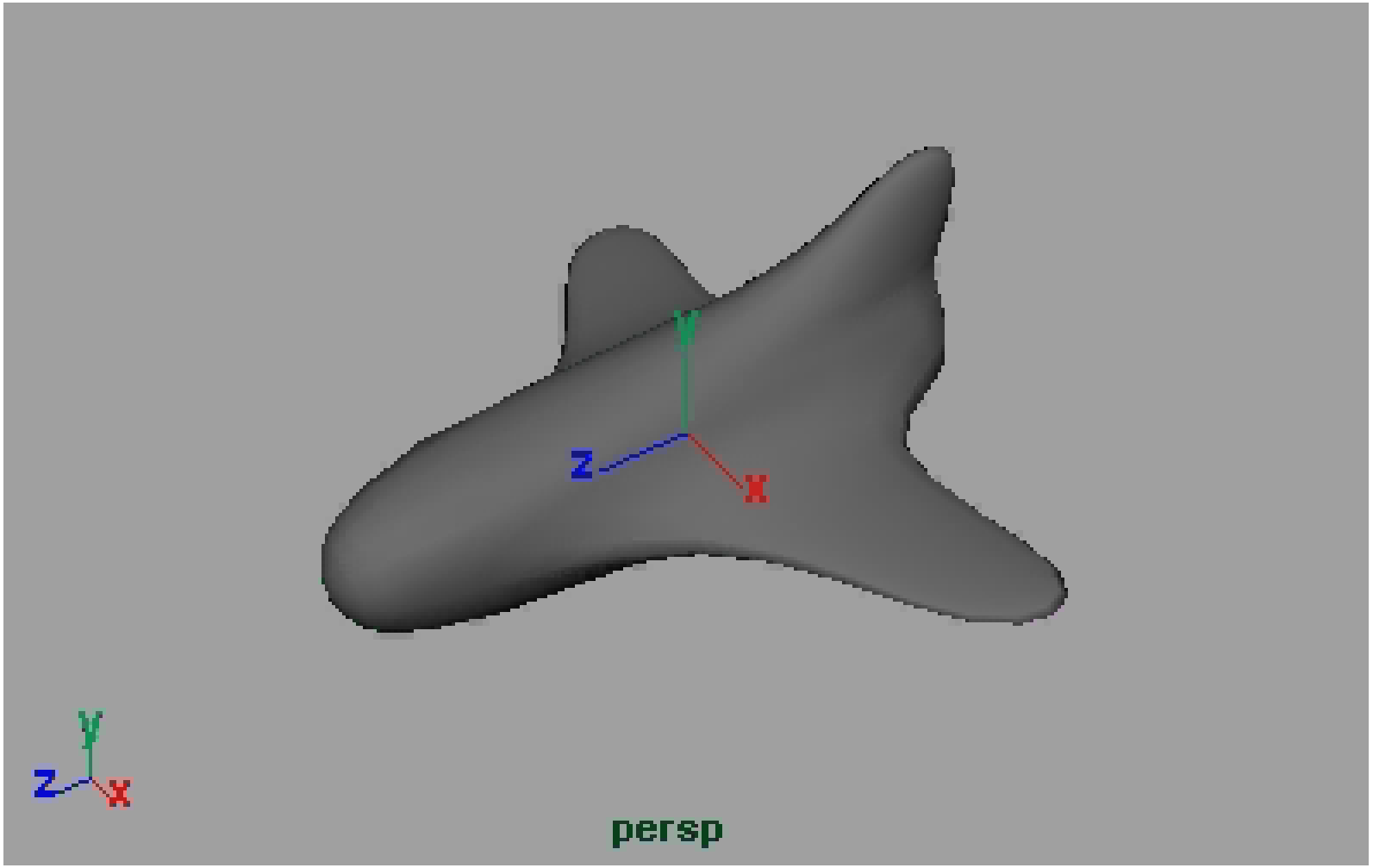
Poligon modellezés:
4. és 5. extruding



Poligon modellezés:
6. extruding

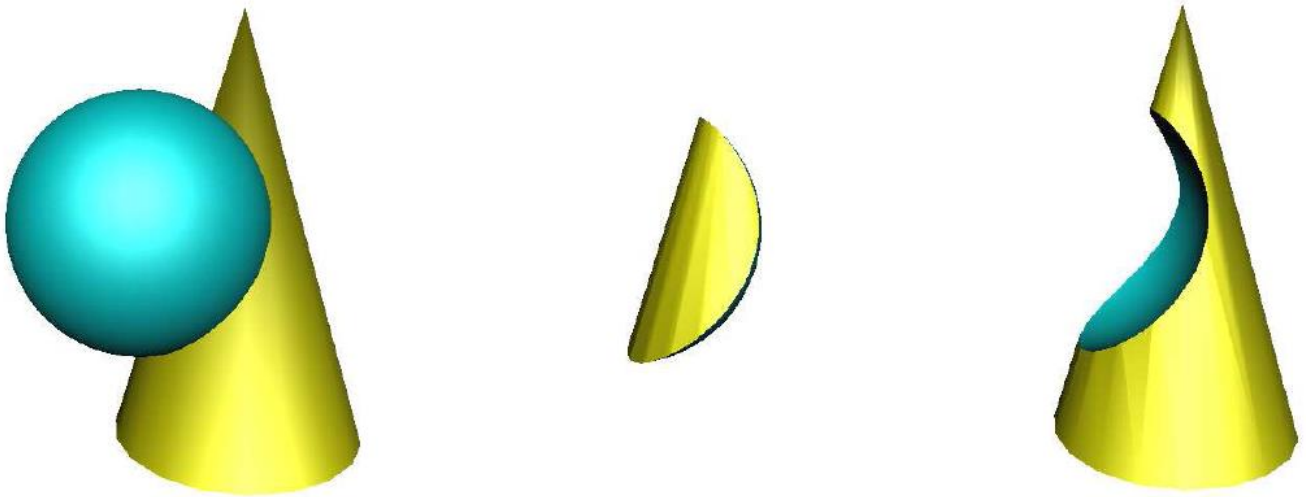


Subdivision simítás

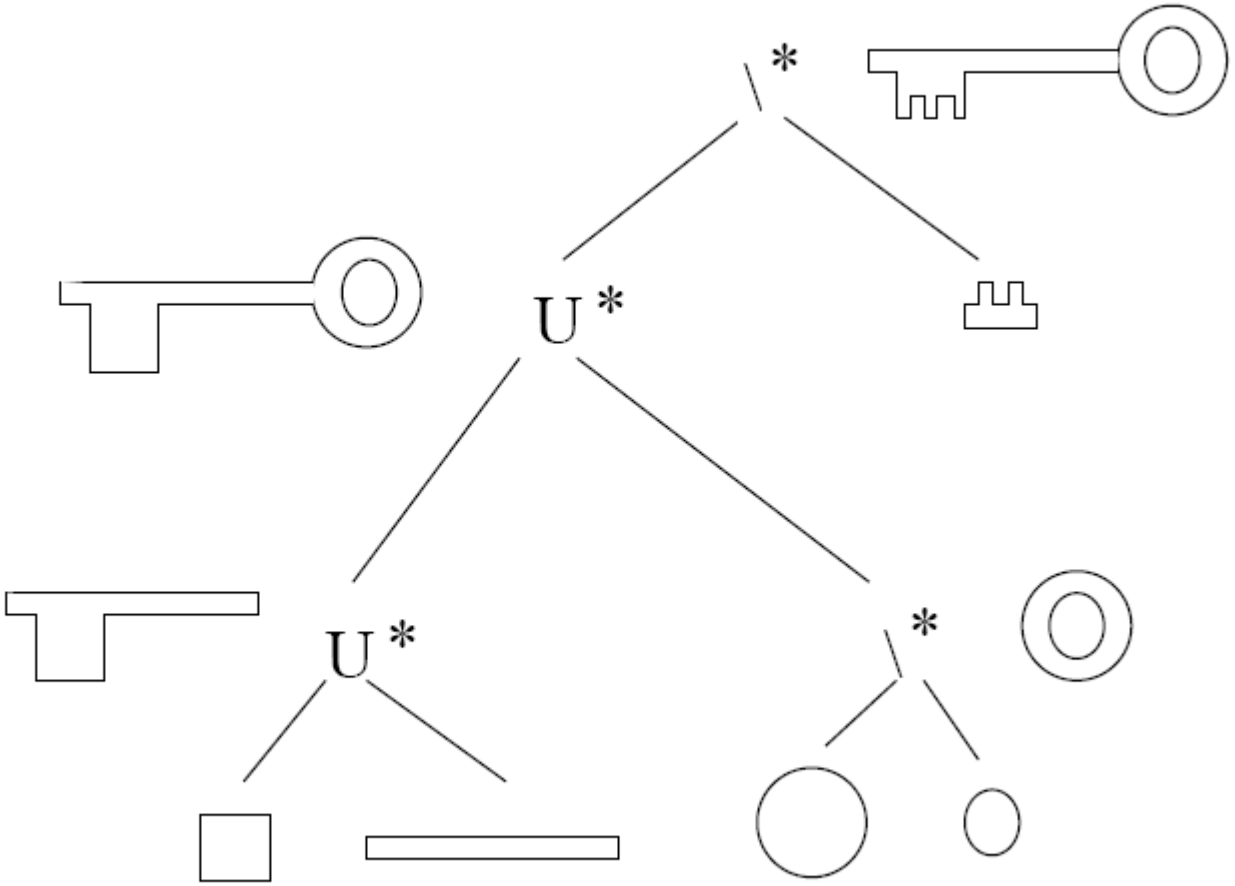


Konstruktív tömörtest geometria (Constuctive Solid Geometry (CSG)

- Összetett testeket primitív testekből halmazműveletek (egyesítés, metszet, különbség) alkalmazásával építi fel
- Regularizált műveletek



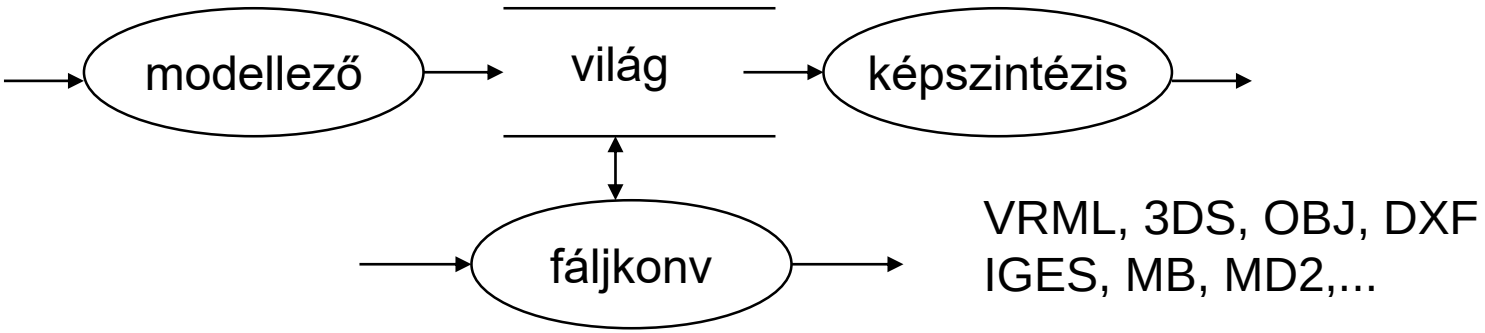
CSG fa



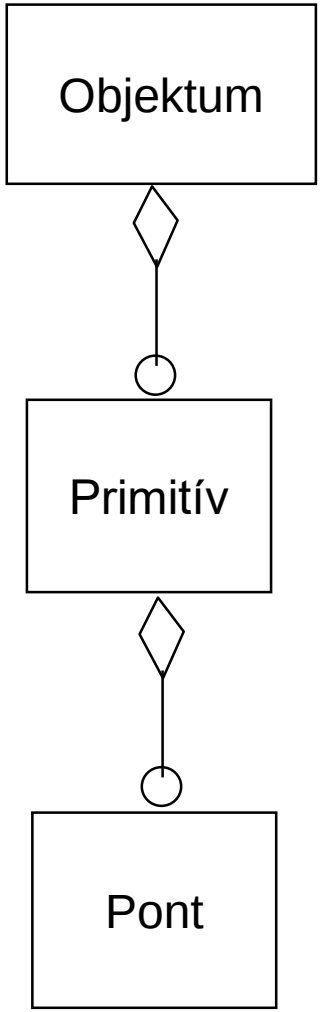
Virtuális világ tárolása

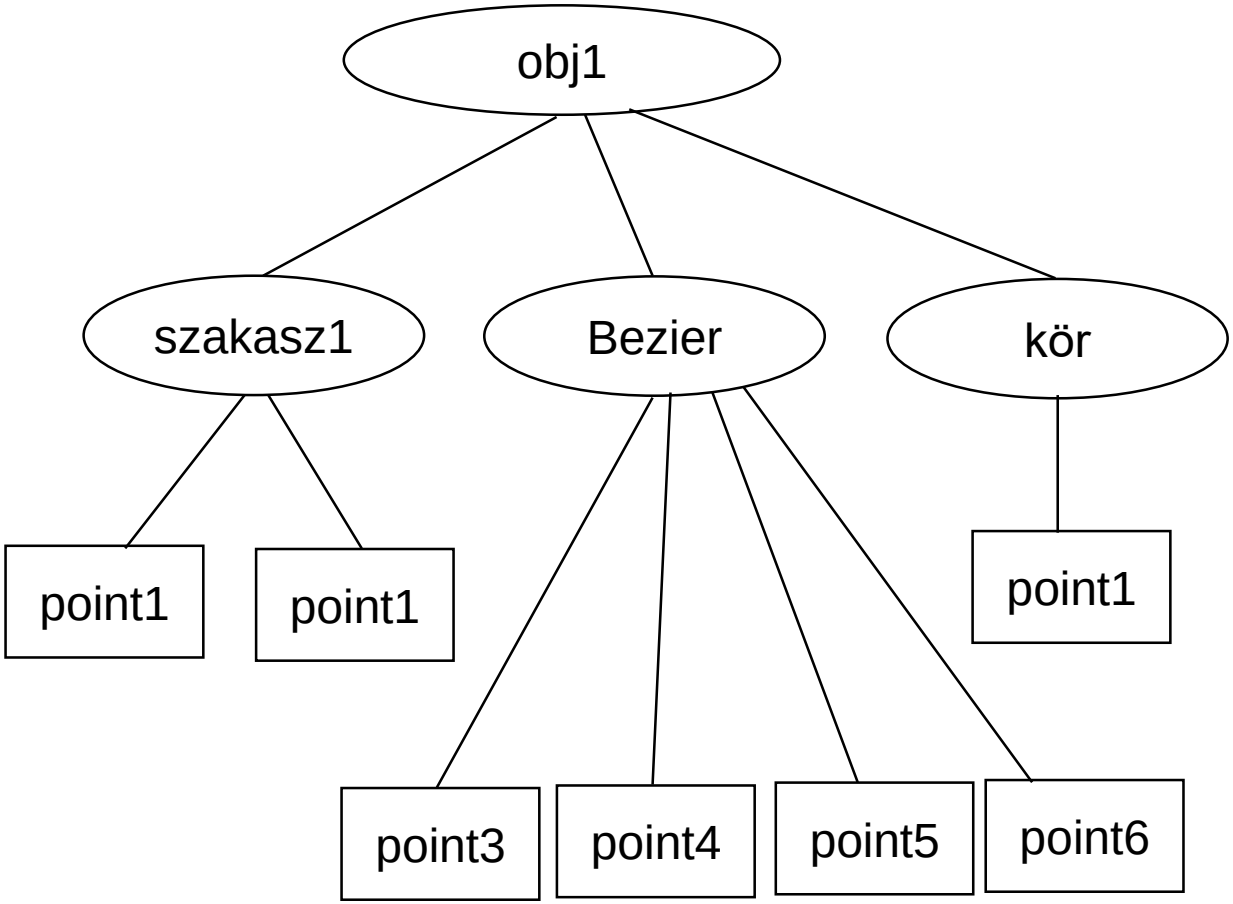
Belső világ tárolása

- Geometria: pontok koordinátái
- Topológia: élek-pontok; lapok-pontok;...
- hierarchia: objektum-lapok-élek-pontok
- transzformáció: lokális és világkoordináta rendszerek

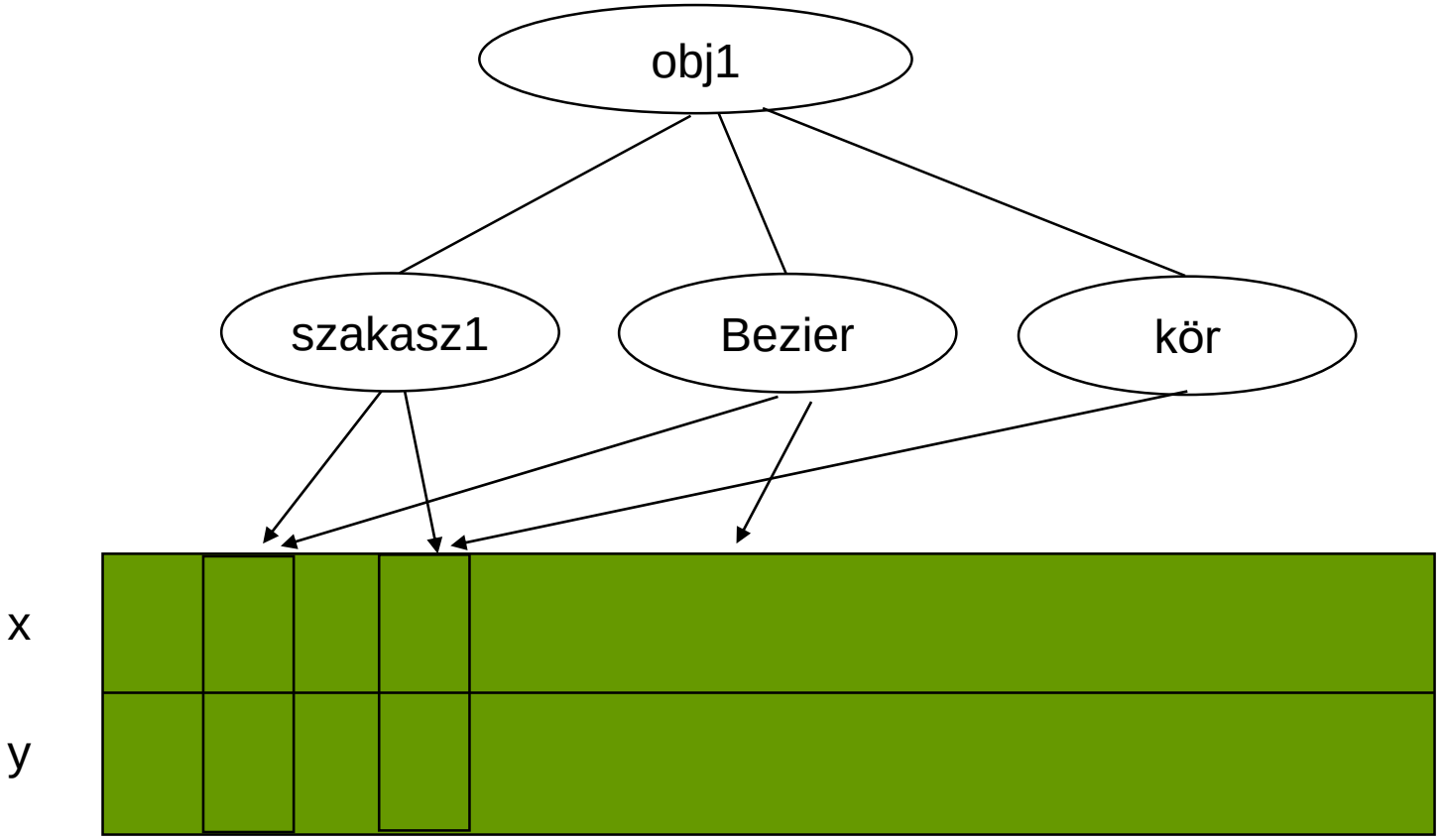


Egyszerű hierarchikus modell



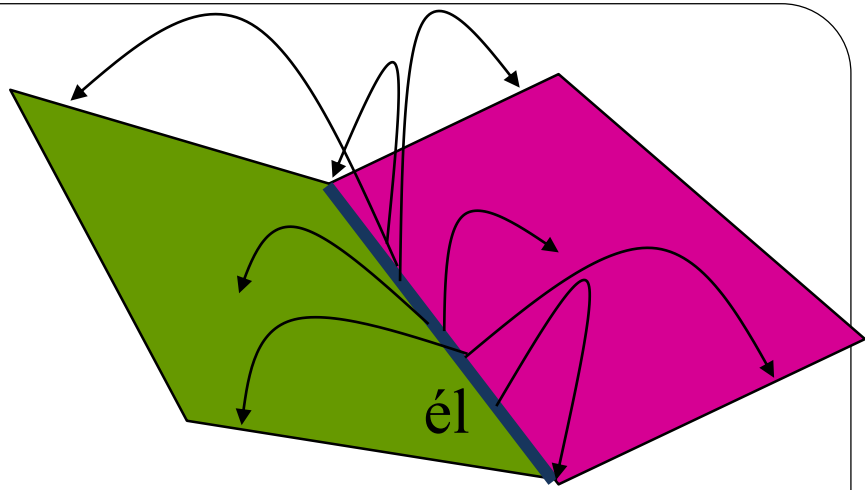


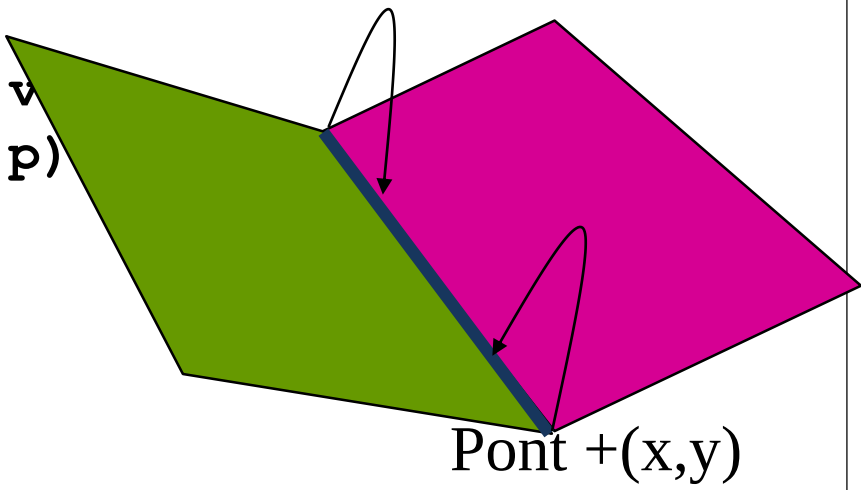
Geometria kiemelése

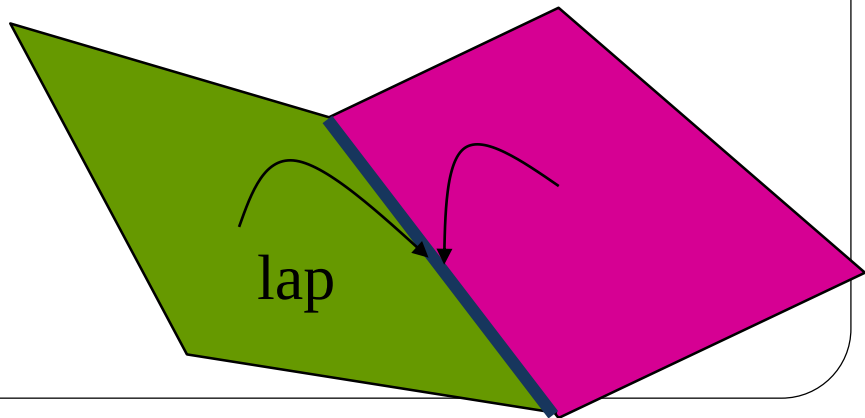


Szárnyasél adatstruktúra

```
class BRepCore {
…
public:
    void MEVVF(…);
    void MVE(float t, Edge& e);
    void MEF(Vertex& v1,Vertex& v
    void Move(Vertex& v, Vector p)
};

class BRep : BRepCore {
    void FaceExtrude( );
    void FaceSplit( );
    void EdgeCollapse( );
    void VertexSplit( );
    …
};
```







Hierarchikus színtér gráfok