Geometriai modellezés

Szirmay-Kalos László

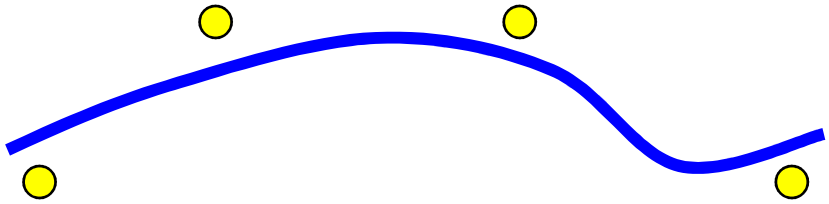
Görbék: 1D ponthalmazok

A pontok koordinátái kielégítenek egy egyenletet:

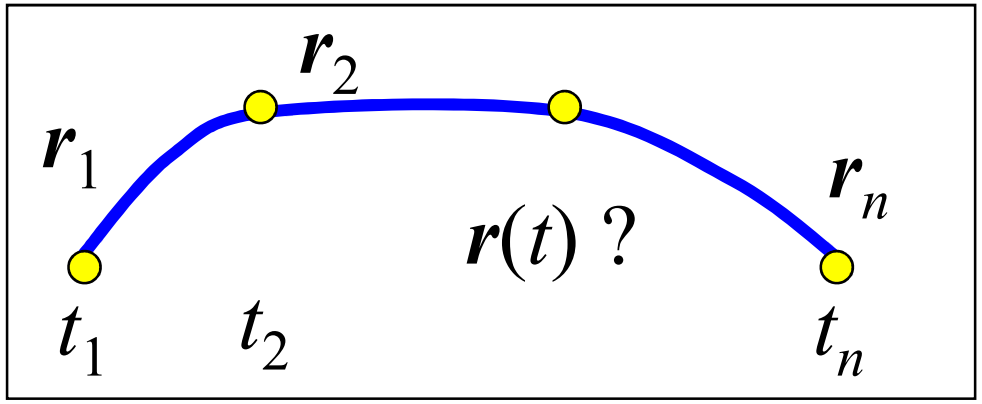
– **implicit**: $f(x, y) = 0$	$f(\boldsymbol{r}) = 0$
• 2D egyenes: $ax + by + c = 0$	$\boldsymbol{n}\cdot(\boldsymbol{r} - \boldsymbol{r}_0) = 0$
• Kör: $(x–x_0)^2+(y–y_0)^2–R^2 = 0$	$\lvert\boldsymbol{r} - \boldsymbol{r}_0\rvert^2 - R^2 = 0$
– **parametrikus**: $x = x(t), y = y(t)$	$\boldsymbol{r} = \boldsymbol{r}(t)$
• 3D egyenes: $x = x_0 + v_x\, t$ $y = y_0 + v_y\, t$ $z = z_0 + v_z\, t$	$\boldsymbol{r} = \boldsymbol{r}_0 + \boldsymbol{v}\, t, \quad t \in [-\infty, \infty]$
• Kör: $t \in [0,1]$ $x(t) = x_0 + R\cos 2\pi t$ $y(t) = y_0 + R\sin 2\pi t$	
– **explicit:** $y = F(x)$	
• 2D egyenes: $y = mx + b$	

Szabadformájú görbék

- Definíció vezérlőpontokkal



- Polinom: $x(t) = \Sigma\, a_i\, t^i, \quad y(t) = \Sigma\, b_i\, t^i$
- Polinom együtthatók:
  - Interpoláció
  - Approximáció



Lagrange interpoláció

- Vezérlő pontok: $\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{r}_3, \ldots, \boldsymbol{r}_n$
- Keresd: $\boldsymbol{r}(t) = \Sigma\, [a_i, b_i]\, t^i$ amelyre

  $\boldsymbol{r}(t_1) = \boldsymbol{r}_1, \boldsymbol{r}(t_2) = \boldsymbol{r}_2, \ldots, \boldsymbol{r}(t_n) = \boldsymbol{r}_n,$
- Megoldás:

$$\boldsymbol{r}(t) = \Sigma_i\, L_i(t)\, \boldsymbol{r}_i \quad \rightarrow \quad \boldsymbol{r}(t_k) = \Sigma_i\, L_i(t_k) \boldsymbol{r}_i = \boldsymbol{r}_k$$

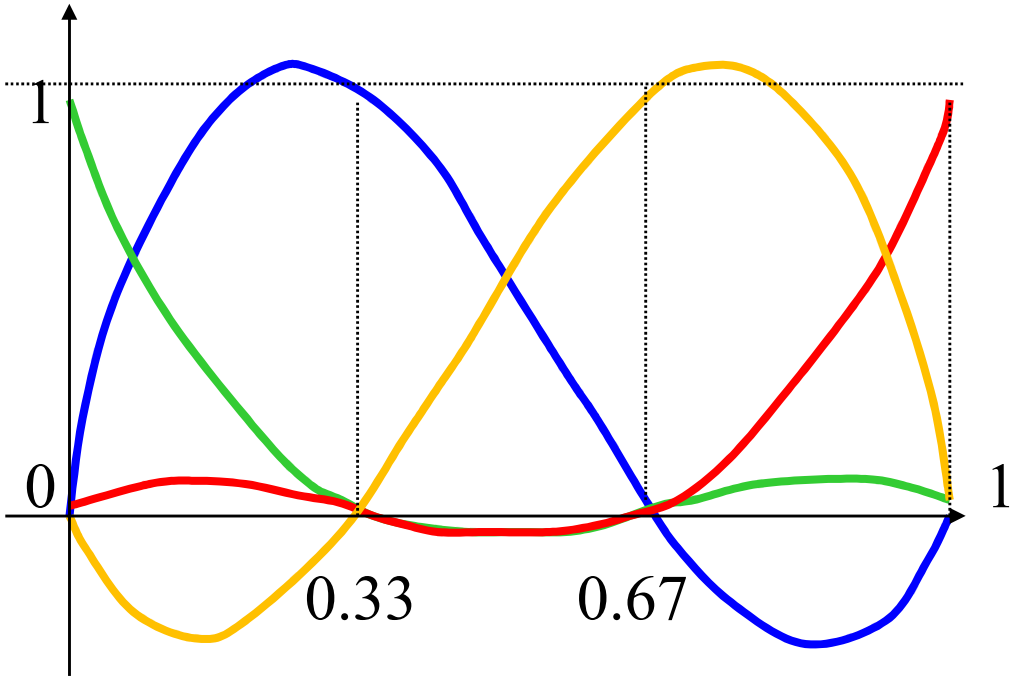
$$L_i(t) = \frac{\Pi_{j \neq i}\,(t\text{-}t_j)}{\Pi_{j \neq i}\,(t_i\text{-}t_j)}$$

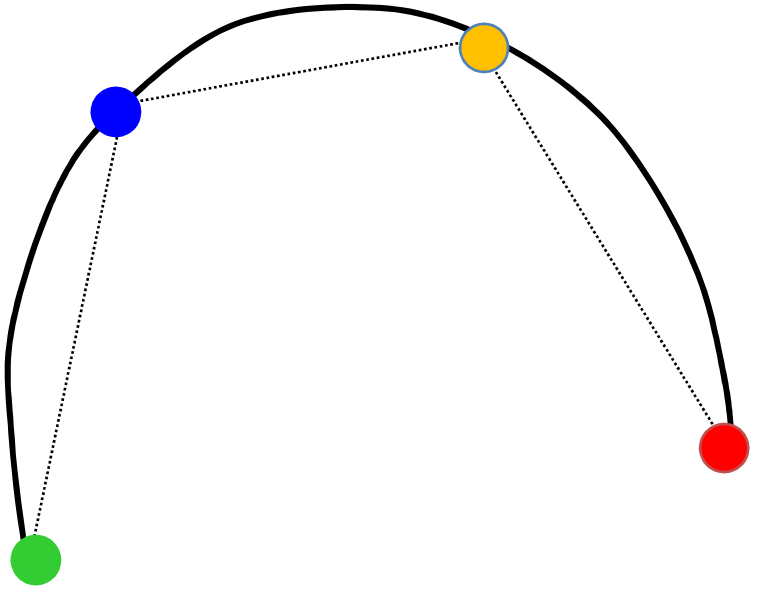
$$L_i(t_k) = \frac{\Pi_{j \neq i}\,(t_k\text{-}t_j)}{\Pi_{j \neq i}\,(t_i\text{-}t_j)} \quad \begin{cases} 1 \text{ if } i=k \\ 0 \text{ if } i \neq k \end{cases}$$

LagrangeCurve

```
class LagrangeCurve {
    vector<vec3>  cps;      // control points
    vector<float> ts;       // parameter (knot) values

    float L(int i, float t) {
        float Li = 1.0f;
        for(int j = 0; j < cps.size(); j++)
                if (j != i) Li *= (t - ts[j])/(ts[i] - ts[j]);
        return Li;
    }
public:
    void AddControlPoint(vec3 cp) {
        float ti = cps.size();     // or something better
        cps.push_back(cp); ts.push_back(ti);
    }

    vec3 r(float t) {
        vec3 rr(0, 0, 0);
        for(int i = 0; i < cps.size(); i++) rr += cps[i] * L(i,t);
        return rr;
    }
};
```

Lagrange interpoláció bázisfüggvényei

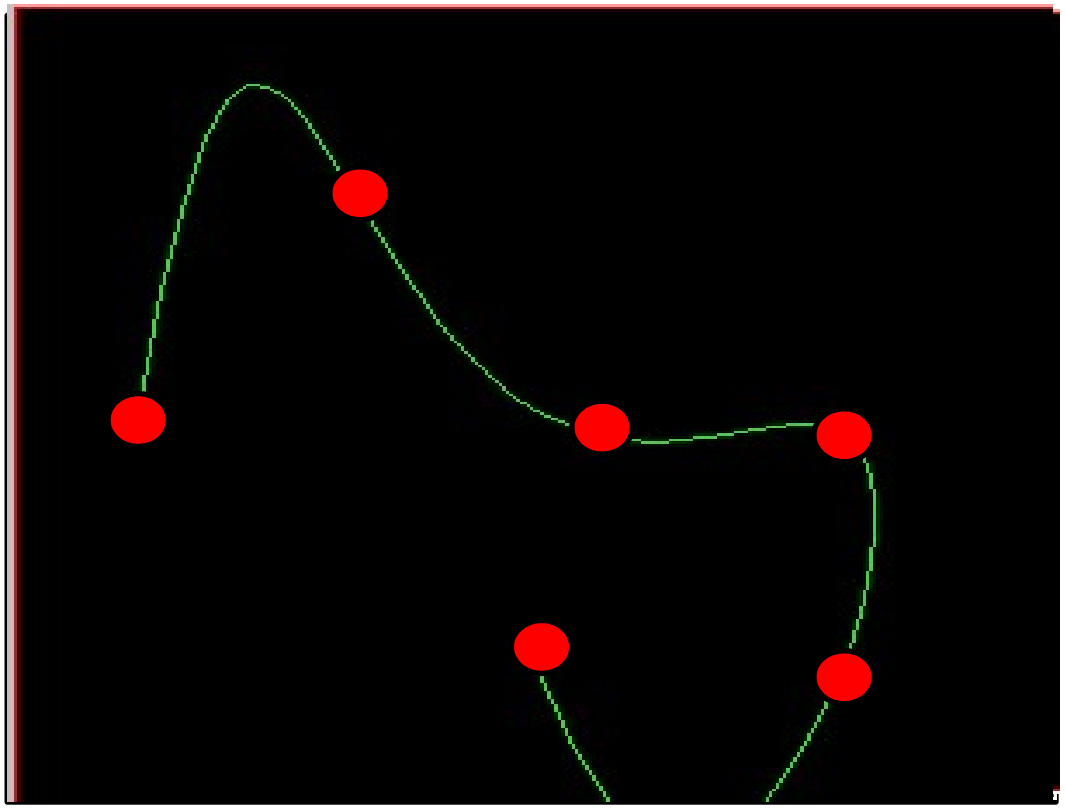


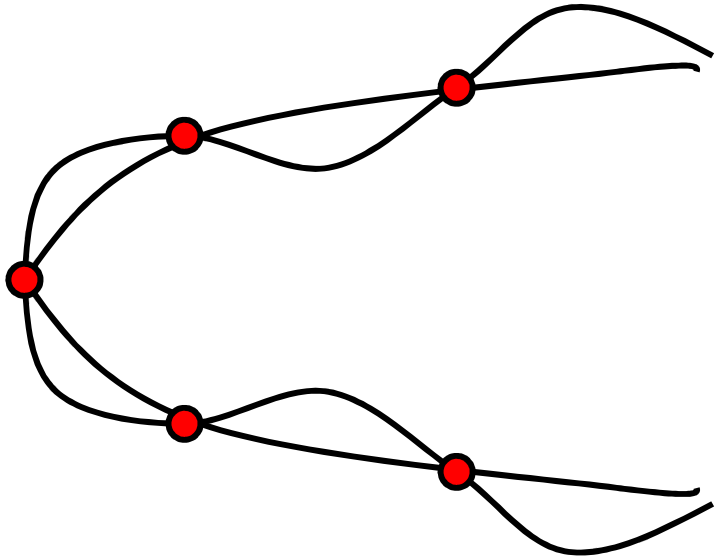


$$L_i(t) = \frac{\prod_{j \neq i} (t - t_j)}{\prod_{j \neq i} (t_i - t_j)}$$

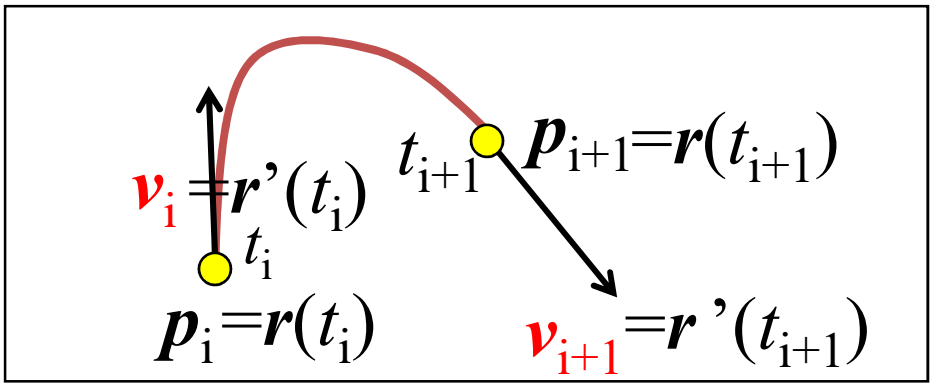
$$\boldsymbol{r}(t) = \Sigma_i \, L_i(t) \, \boldsymbol{r}_i$$

Lagrange interpoláció problémái





Hermite interpoláció



- $r(t) = a_3(t-t_i)^3+a_2(t-t_i)^2+a_1(t-t_i)+a_0$
- $r_i'(t) = 3a_3(t-t_i)^2+2a_2(t-t_i)+a_1$

$$r(t_i) = a_0 = p_i$$

$$r(t_{i+1}) = a_3(t_{i+1}-t_i)^3+a_2(t_{i+1}-t_i)^2+a_1(t_{i+1}-t_i)+a_0 = p_{i+1}$$

$$r'(t_i) = a_1 = v_i$$

$$r'(t_{i+1}) = 3a_3(t_{i+1}-t_i)^2+2a_2(t_{i+1}-t_i)+a_1 = v_{i+1}$$

$$a_0 = p_i$$

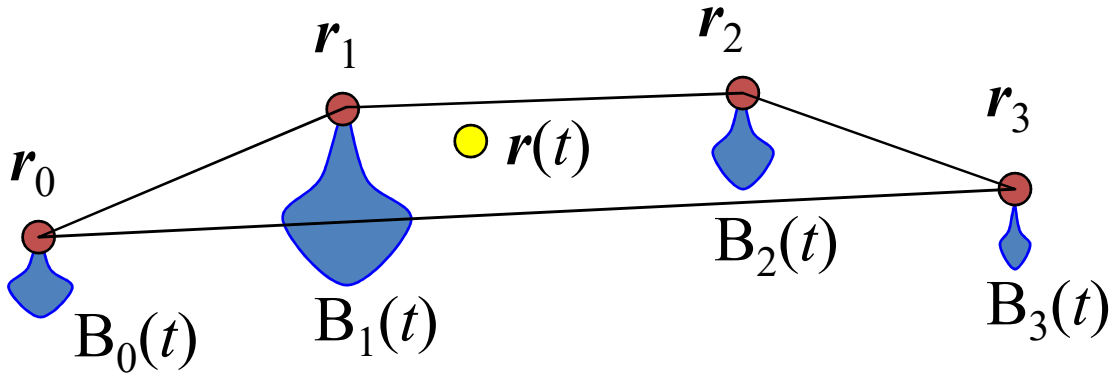
$$a_1 = v_i$$

$$a_2 = \frac{3(p_{i+1}-p_i)}{(t_{i+1}-t_i)^2} - \frac{v_{i+1}+2v_i}{(t_{i+1}-t_i)}$$

$$a_3 = \frac{2(p_i-p_{i+1})}{(t_{i+1}-t_i)^3} + \frac{v_{i+1}+v_i}{(t_{i+1}-t_i)^2}$$

Bezier approximáció

- Keresd: $\boldsymbol{r}(t) = \Sigma_i \, \mathrm{B}_i(t) \, \boldsymbol{r}_i$
  - $\mathrm{B}_i(t)$: ne oszcilláljon
  - Konvex burok tulajdonság
  - $\mathrm{B}_i(t) \geq 0, \qquad \Sigma \, \mathrm{B}_i(t) = 1$



Bernstein polinomok

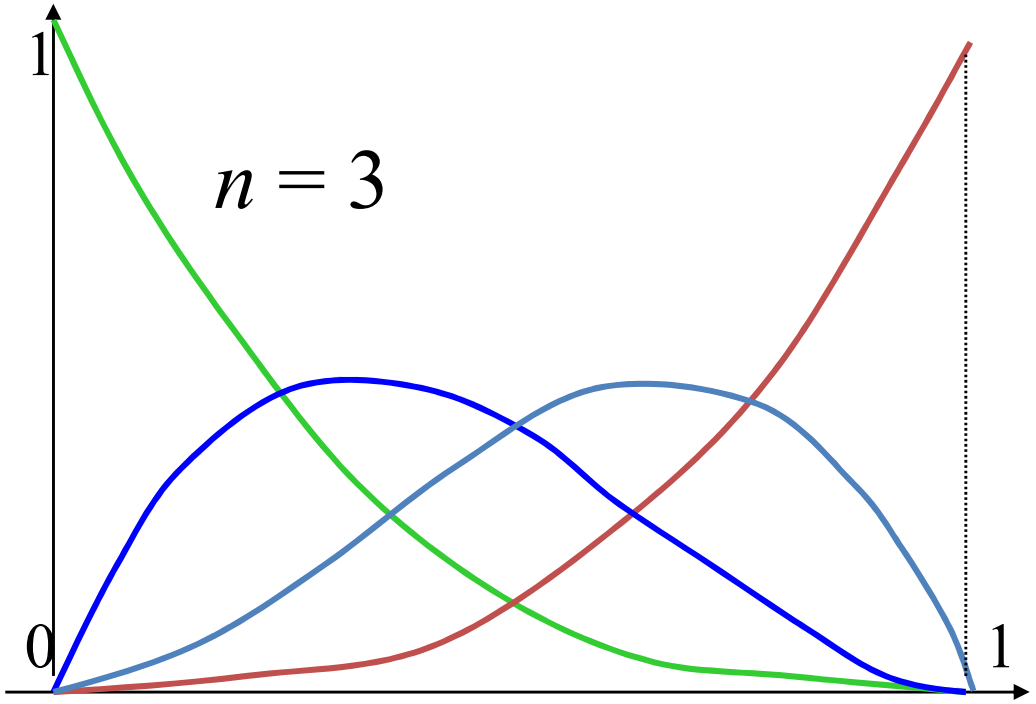
Newton féle binomiális tétel

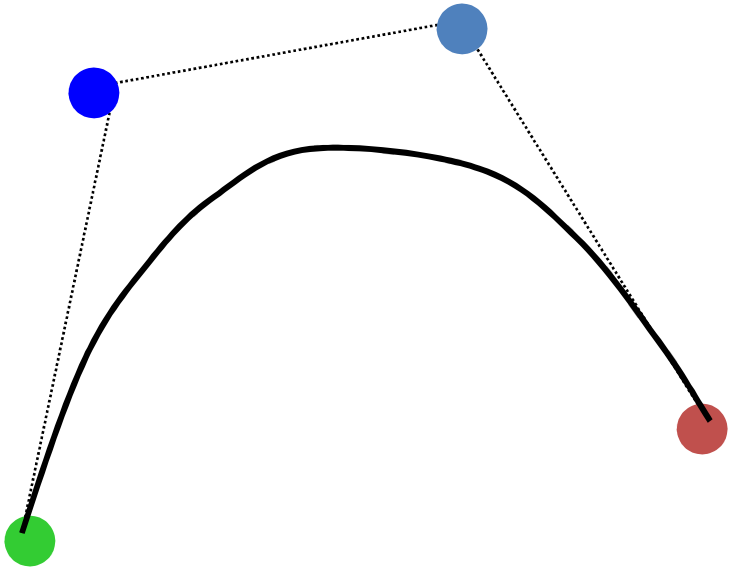
$$1^n = (t+(1-t))^n = \sum_{i=0}^{n} \binom{n}{i} t^i (1-t)^{n-i}$$

$$B_i(t)$$

$$B_i(t) \geq 0, \ \Sigma B_i(t) = 1: \ \text{OK}$$

Bezier approximáció

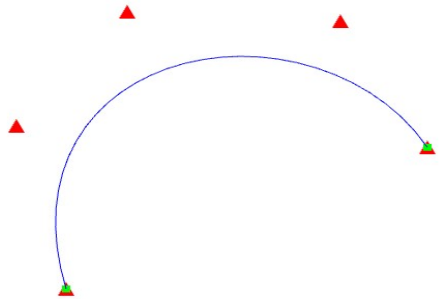




$$B_i(t) = \binom{n}{i} t^i (1-t)^{n-i}$$

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

BezierCurve

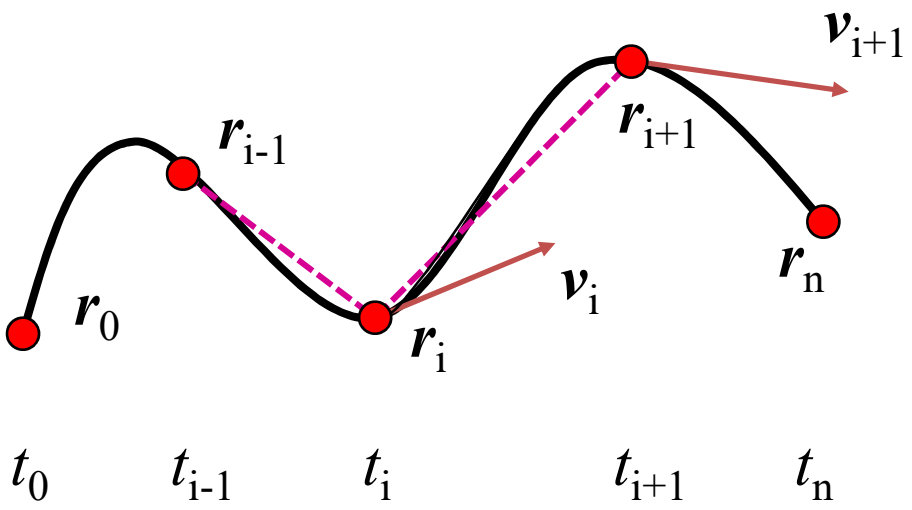


```
class BezierCurve {
    vector<vec3> cps;// control points

    float B(int i, float t) {
        int n = cps.size()-1; // n deg polynomial = n+1 pts!
        float choose = 1;
        for(int j = 1; j <= i; j++) choose *= (float)(n-j+1)/j;
        return choose * pow(t, i) * pow(1-t, n-i);
    }
public:
    void AddControlPoint(vec3 cp) { cps.push_back(cp); }

    vec3 r(float t) {
        vec3 rr(0, 0);
        for(int i = 0; i < cps.size(); i++) rr += cps[i] * B(i,t);
        return rr;
    }
};
```

Catmull-Rom spline

Minden két vezérlőpont közé egy görbe szegmens

Simaság: a sebesség is legyen közös két egymás utánira



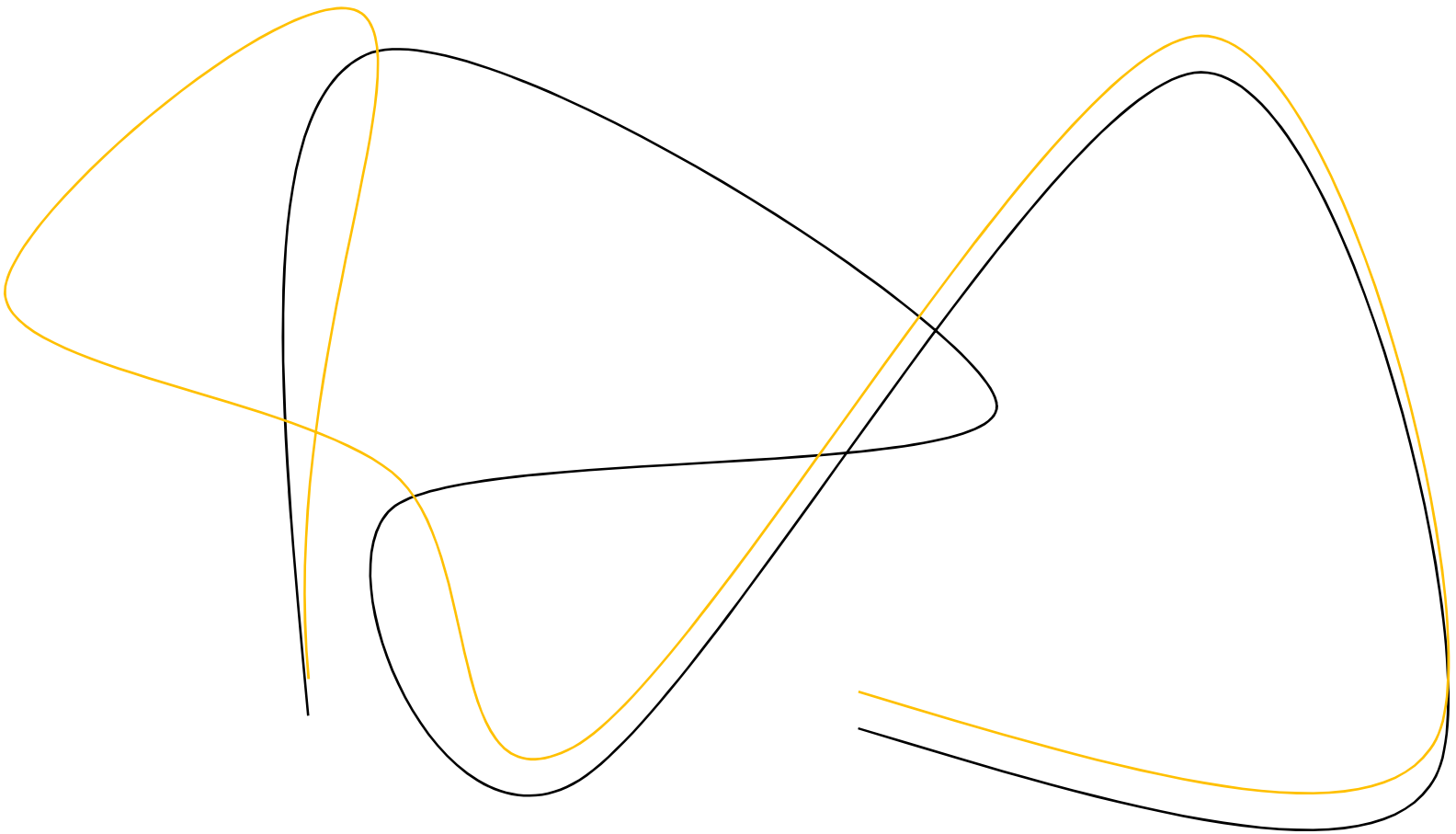
$$\boldsymbol{v}_i = \frac{1}{2}\left(\frac{\boldsymbol{r}_{i+1} - \boldsymbol{r}_i}{t_{i+1} - t_i} + \frac{\boldsymbol{r}_i - \boldsymbol{r}_{i-1}}{t_i - t_{i-1}}\right)$$

Egy görbeszegmens: Hermite interpoláció

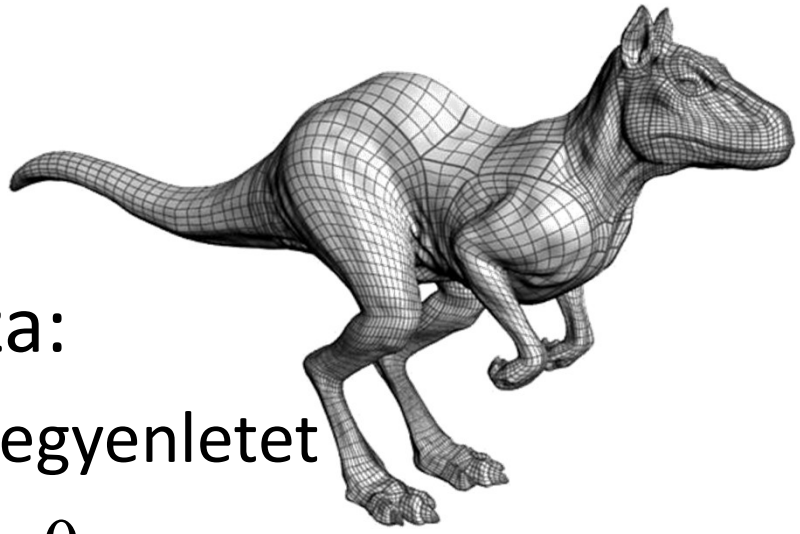
Legeslegelső és legutolsó sebesség explicite

CatmullRom

```
class CatmullRom {
    vector<vec3> cps;        // control points
    vector<float>  ts;       // parameter (knot) values

    vec3 Hermite( vec3 p0, vec3 v0, float t0,
                  vec3 p1, vec3 v1, float t1,
                  float t ) {
        ???
    }

public:
    void AddControlPoint(vec3 cp, float t) { ??? }

    vec3 r(float t) {
        for(int i = 0; i < cps.size() - 1; i++) {
            if (ts[i] <= t && t <= ts[i+1]) return Hermite(…);
        }
    }
};
```



Felületek



Felület a 3D tér 2D részhalmaza:

- Koordináták kielégítenek egy egyenletet
- **implicit**: $f(x, y, z) = 0$
  - gömb: $(x - x0)^2 + (y - y0)^2 + (z - z0)^2 - R^2 = 0$
- **parametrikus**: $x = x(u,v),\ y = y(u,v),\ z = z(u,v),$

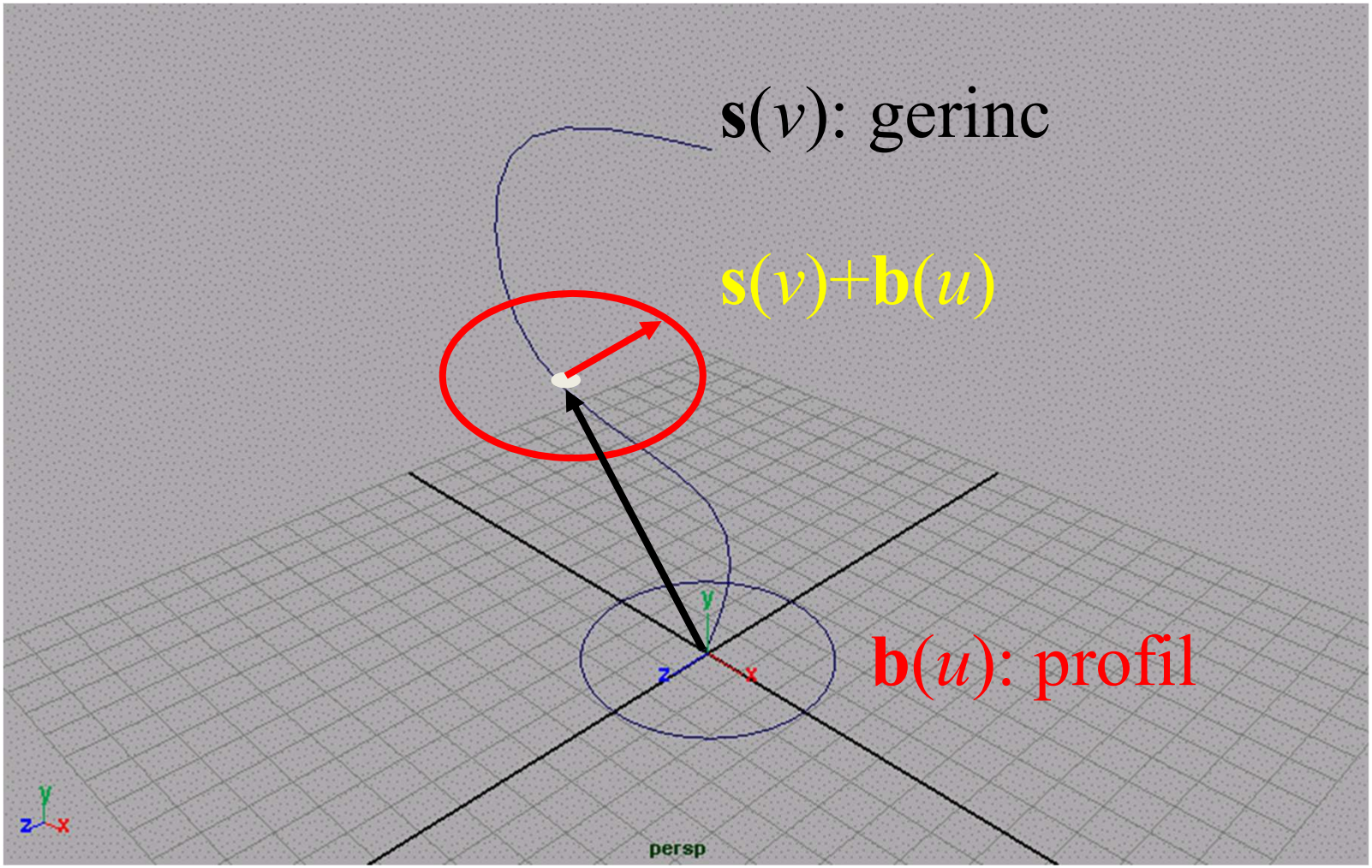
  $u,v \in [0,1]$
  - gömb

$$x = x0 + R \cos 2\pi u \sin \pi v$$
$$y = y0 + R \sin 2\pi u \sin \pi v$$
$$z = z0 + R \cos \pi v \qquad u,v \in [0,1]$$

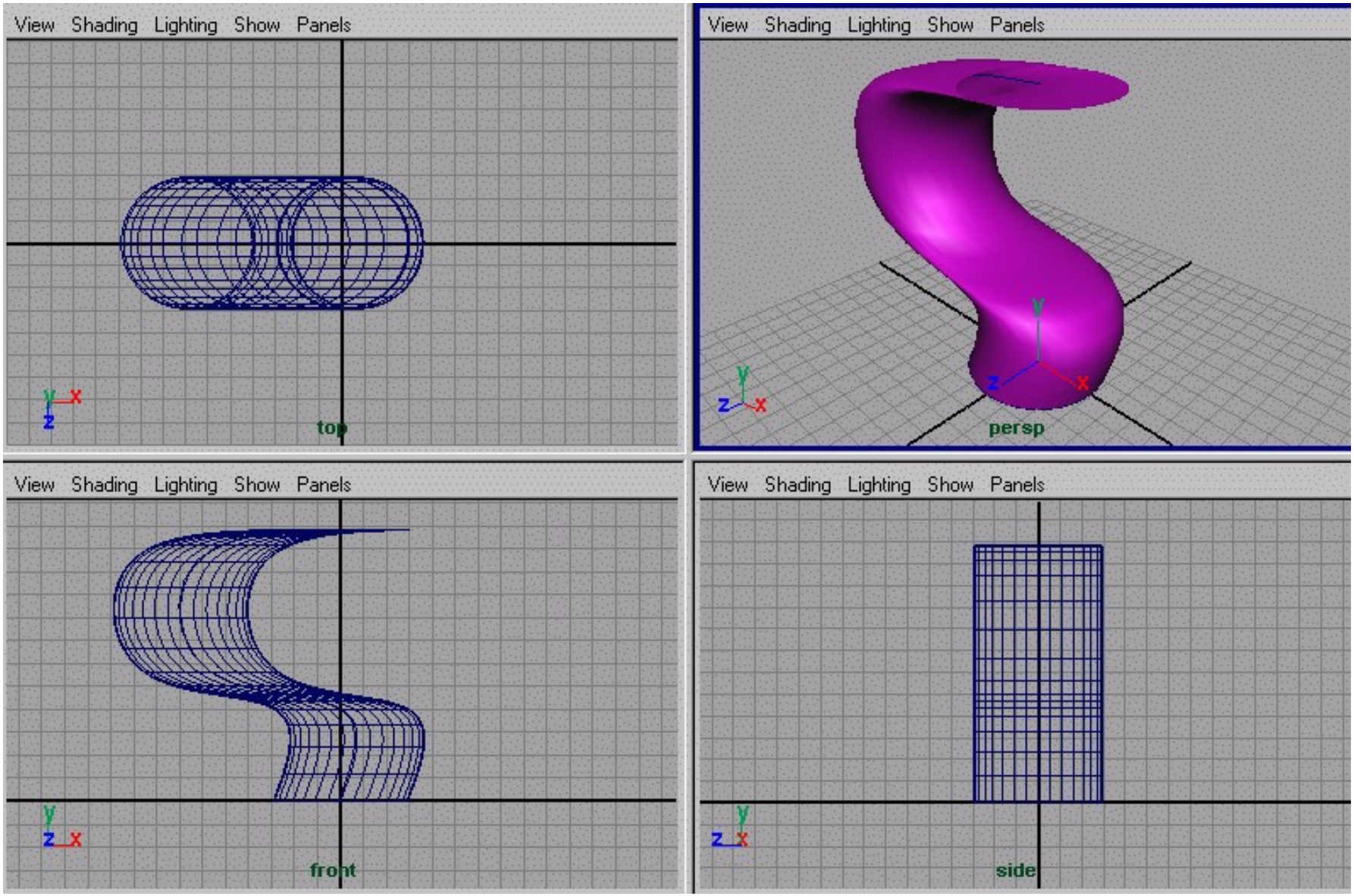
- **explicit** (magasságmező):

$$z = h(x,y)$$

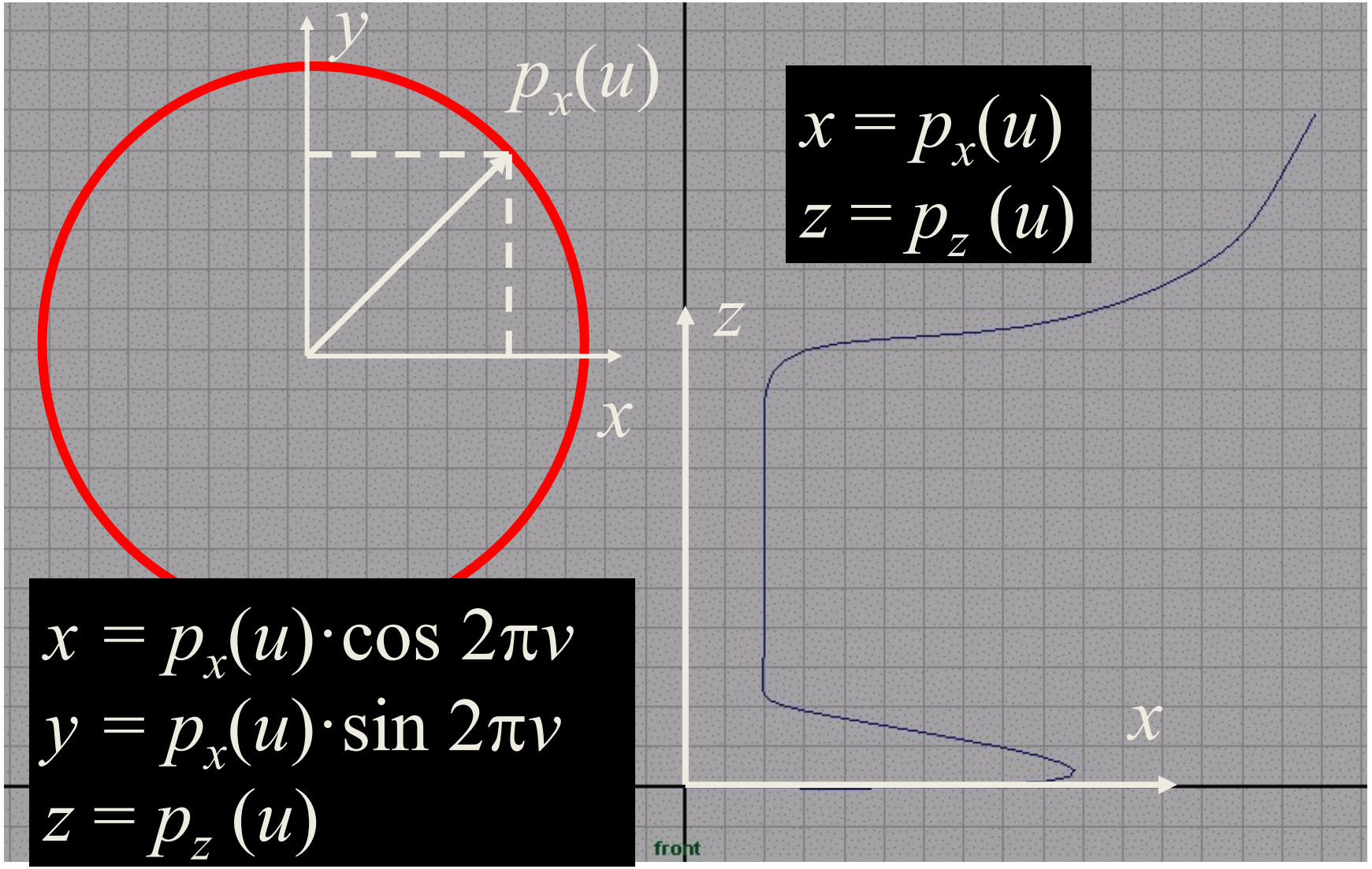
Kihúzás (extrude)



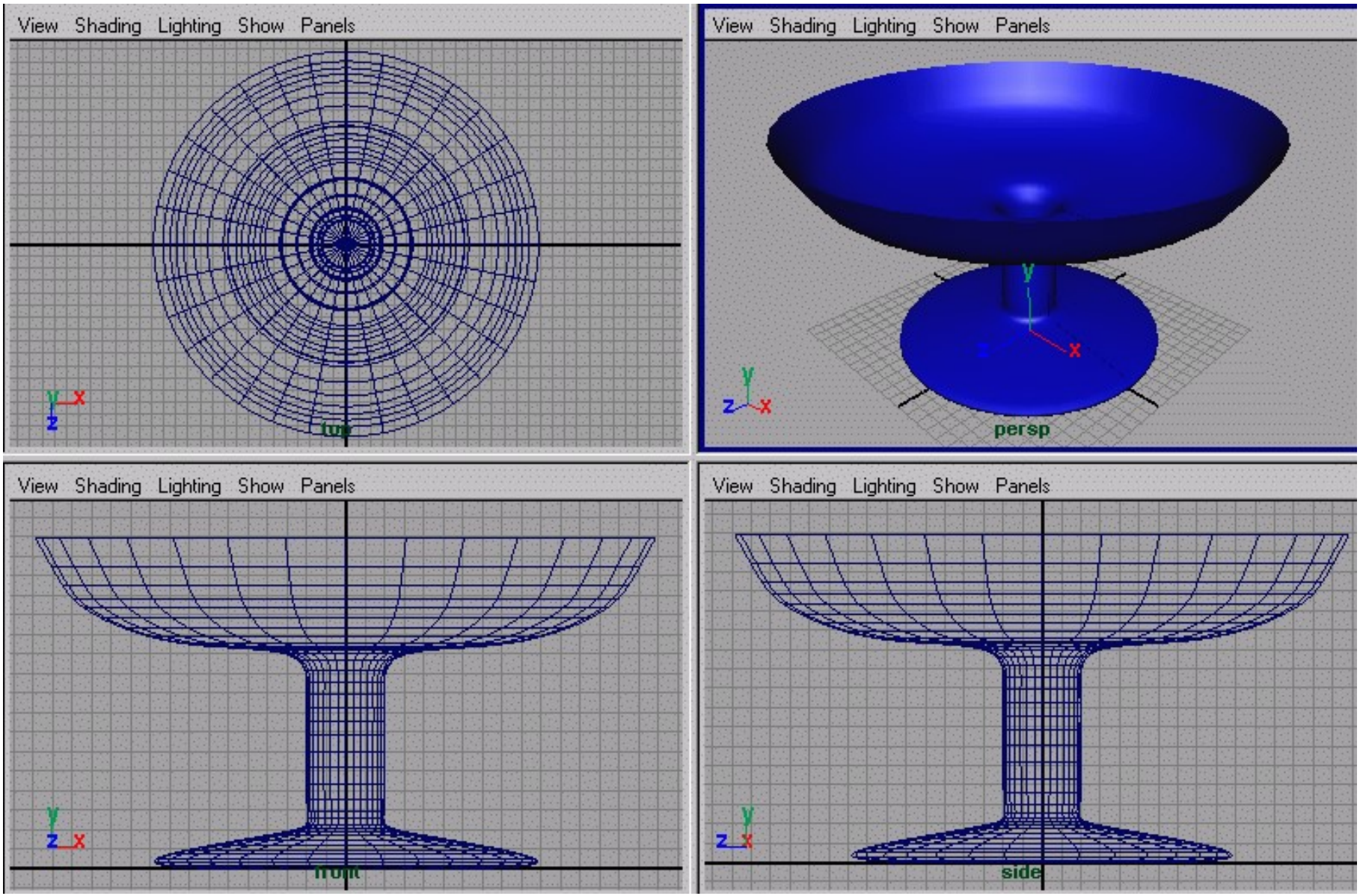
Lapos kihúzás: $\mathbf{r}(u,v)=\mathbf{b}(u)+\mathbf{s}(v)$



Forgatás

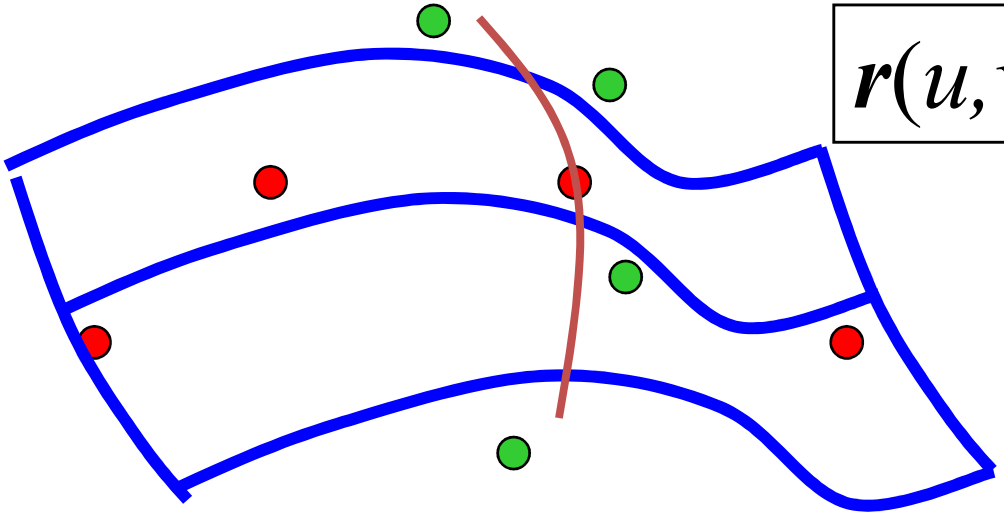

y
$p_x(u)$
x
$x = p_x(u)$
$z = p_z(u)$
z
x
front
$x = p_x(u) \cdot \cos 2\pi v$
$y = p_x(u) \cdot \sin 2\pi v$
$z = p_z(u)$

Forgatás


View Shading Lighting Show Panels
top
View Shading Lighting Show Panels
persp
View Shading Lighting Show Panels
front
View Shading Lighting Show Panels
side

Szabadformájú felület

Definíció vezérlőpontokkal

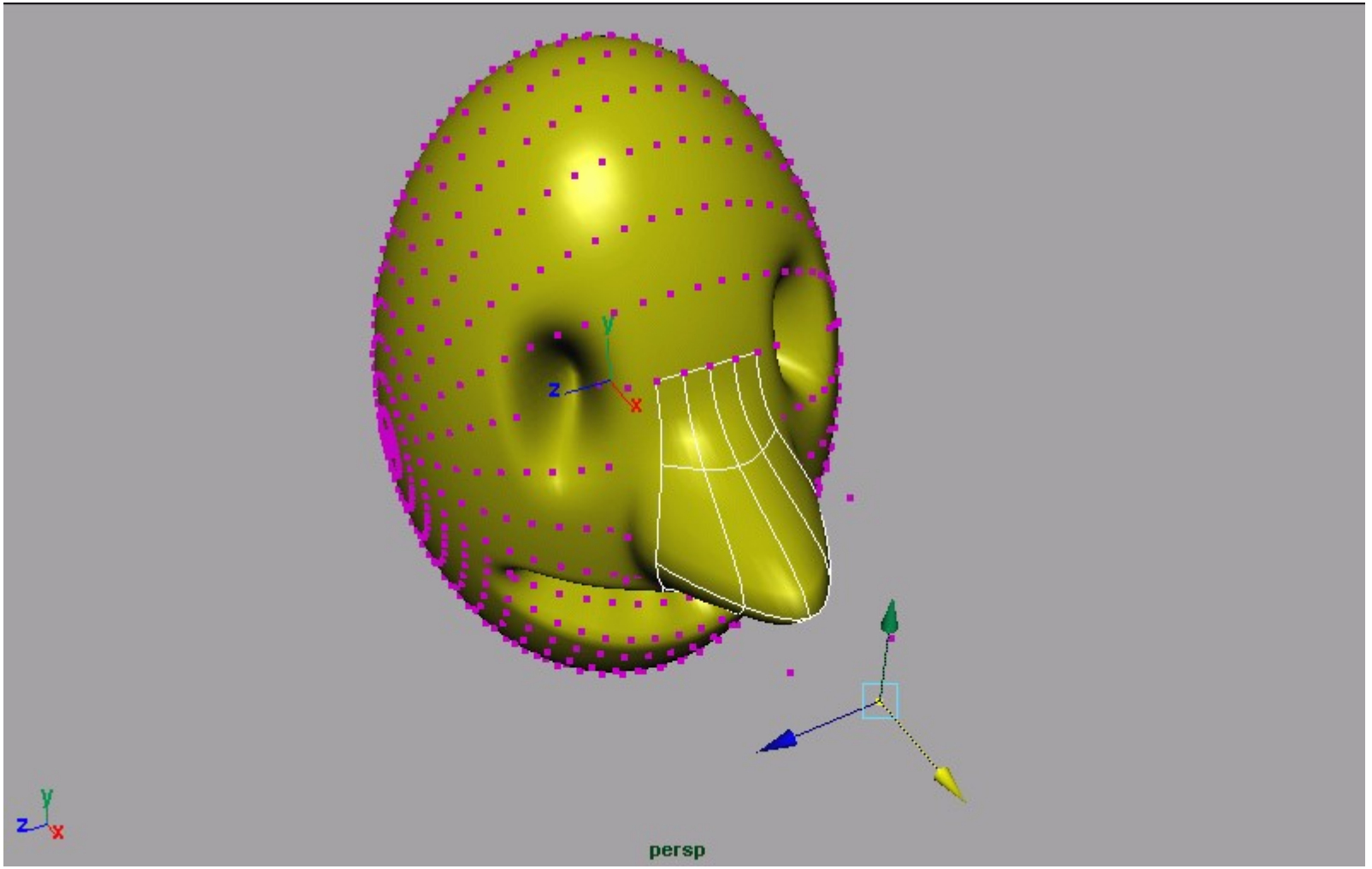


$$\boldsymbol{r}(u,v) = \Sigma\,\Sigma\, B_i(u)\, B_j(v)\, \boldsymbol{r}_{i,j}$$

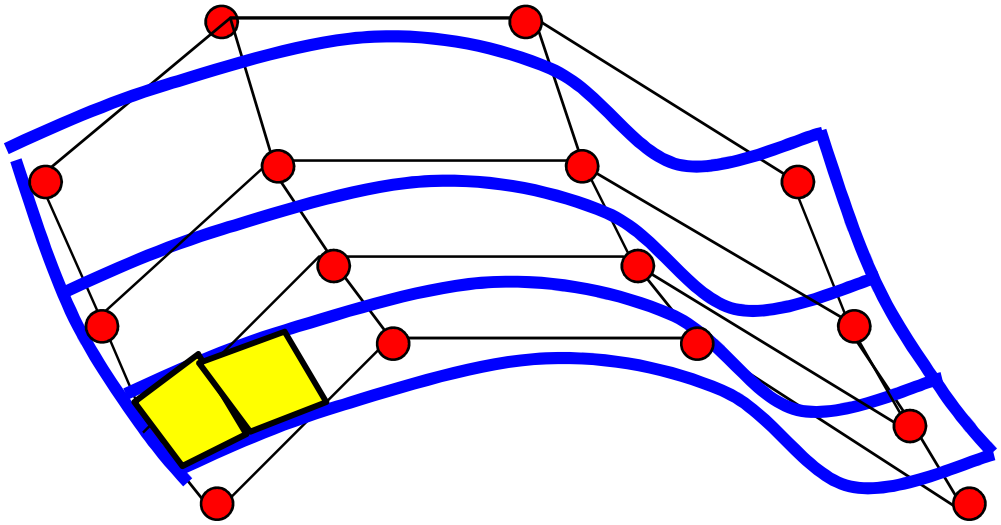
$$\boldsymbol{r}(u,v) = \boldsymbol{r}_v(u) = \Sigma\, B_i(u)\, \boldsymbol{r}_i(v)$$

$$\boldsymbol{r}_i(v) = \Sigma\, B_j(v)\, \boldsymbol{r}_{i,j}$$

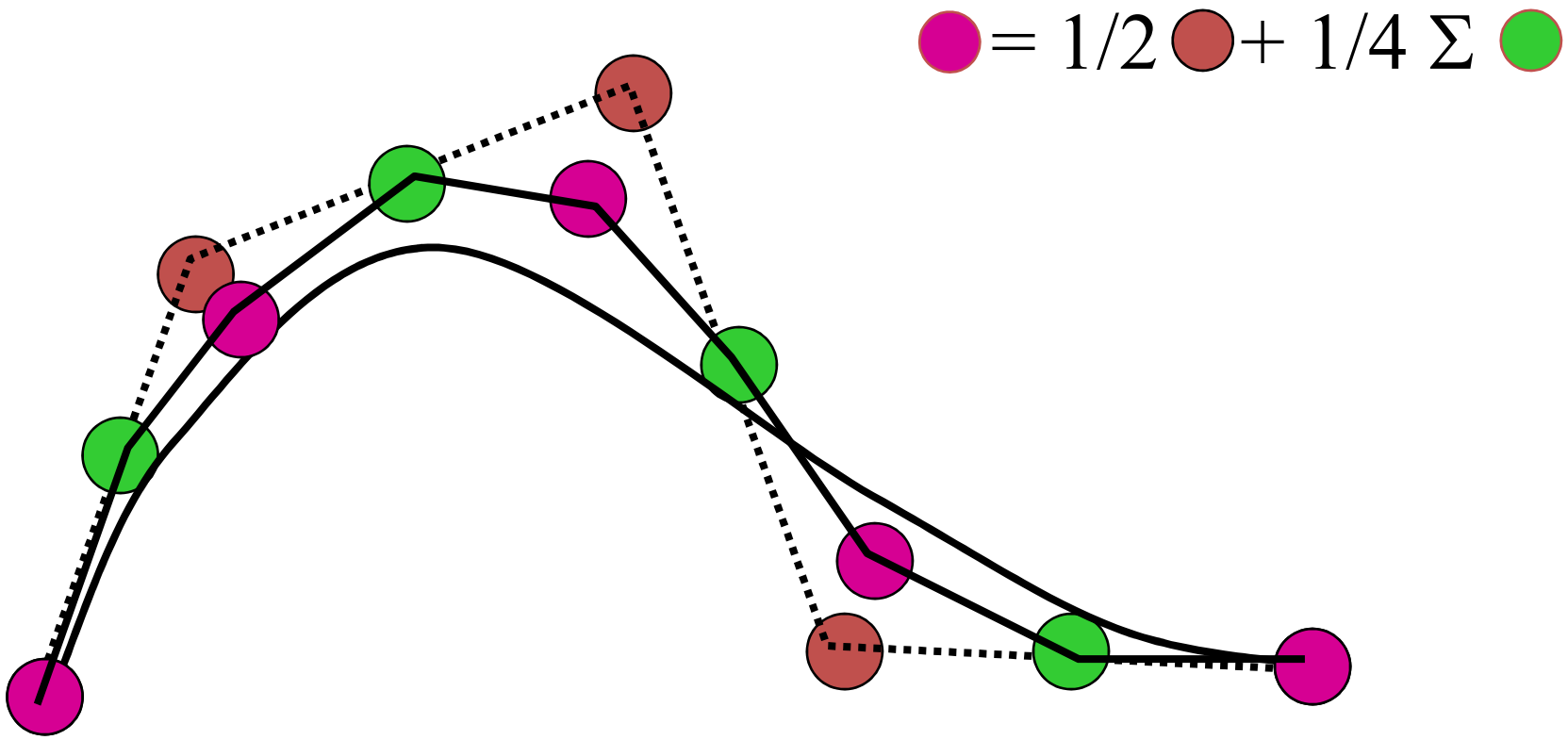
Vezérlőpontok módosítása



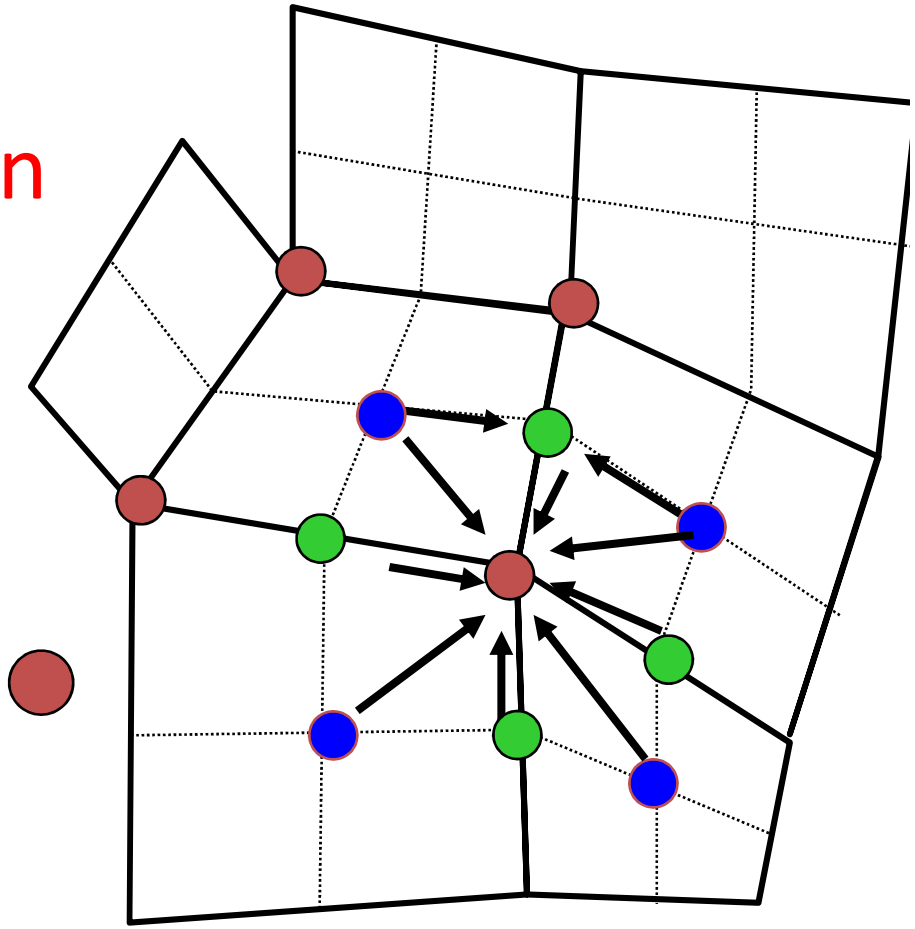
Poligonháló finomítása



Subdivision görbék



Catmull-Clark subdivision felület

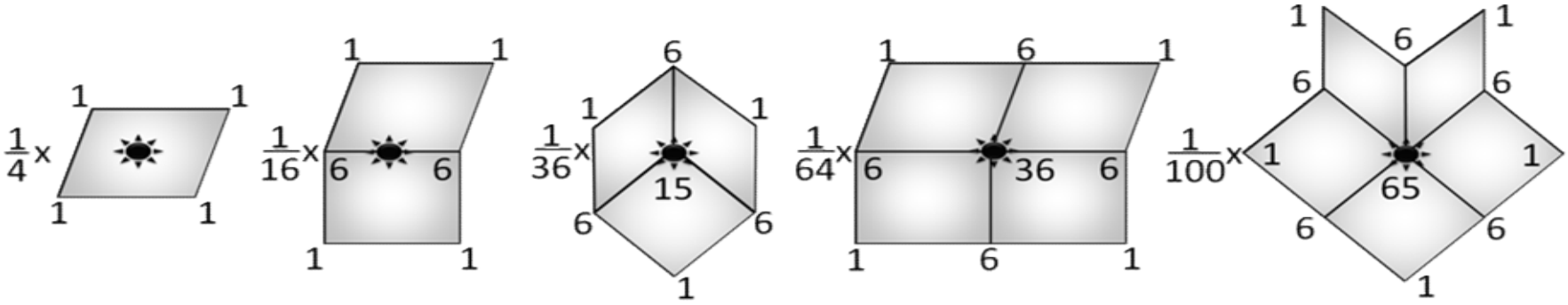


● = 1/4Σ ●

i = 1/2Σ ●

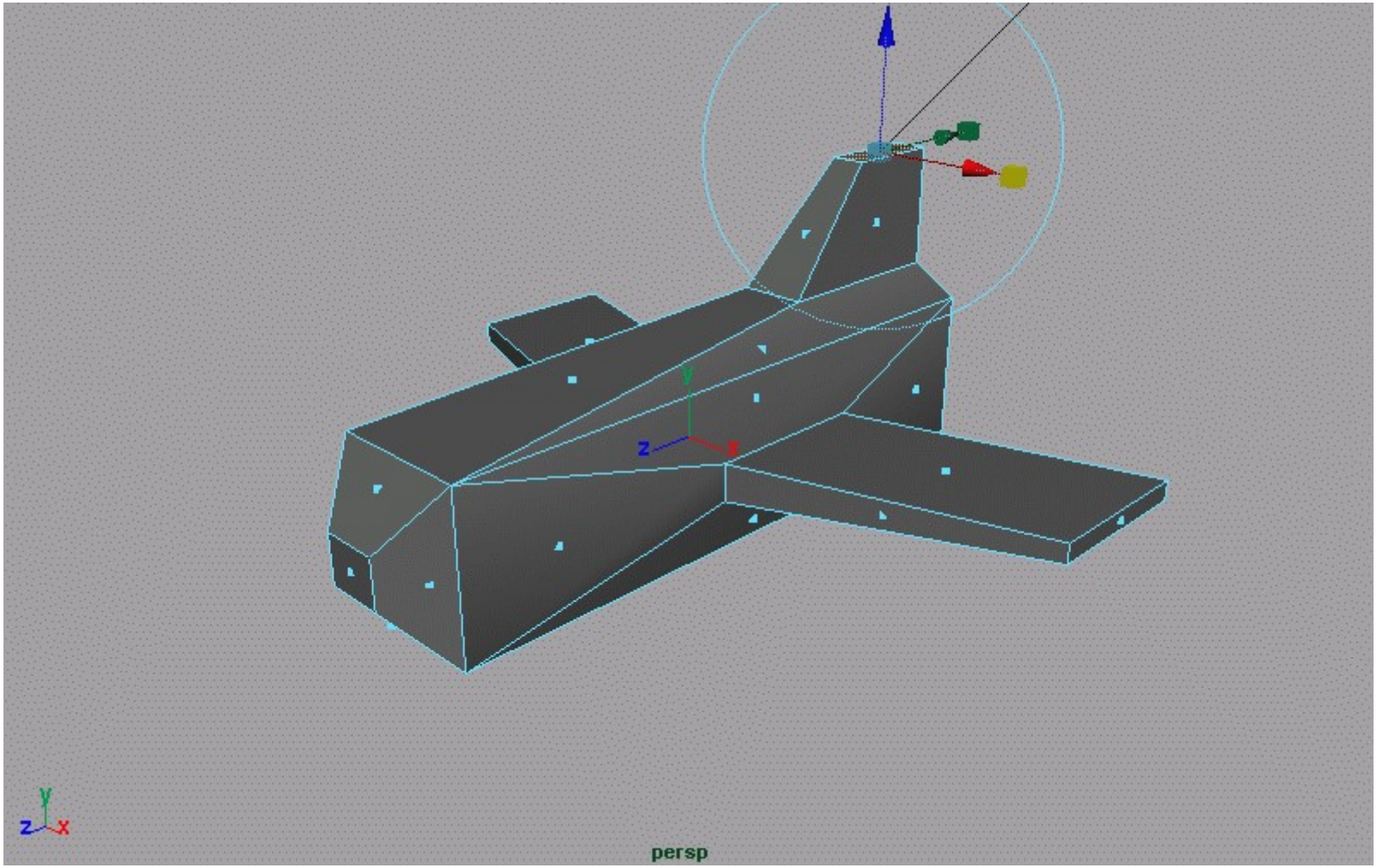
● = $1/v^2\Sigma$ ● $+2/v^2\Sigma$ *i* $+(v\text{-}3)/v$ ●

● = 1/4Σ ● + 1/2 *i*

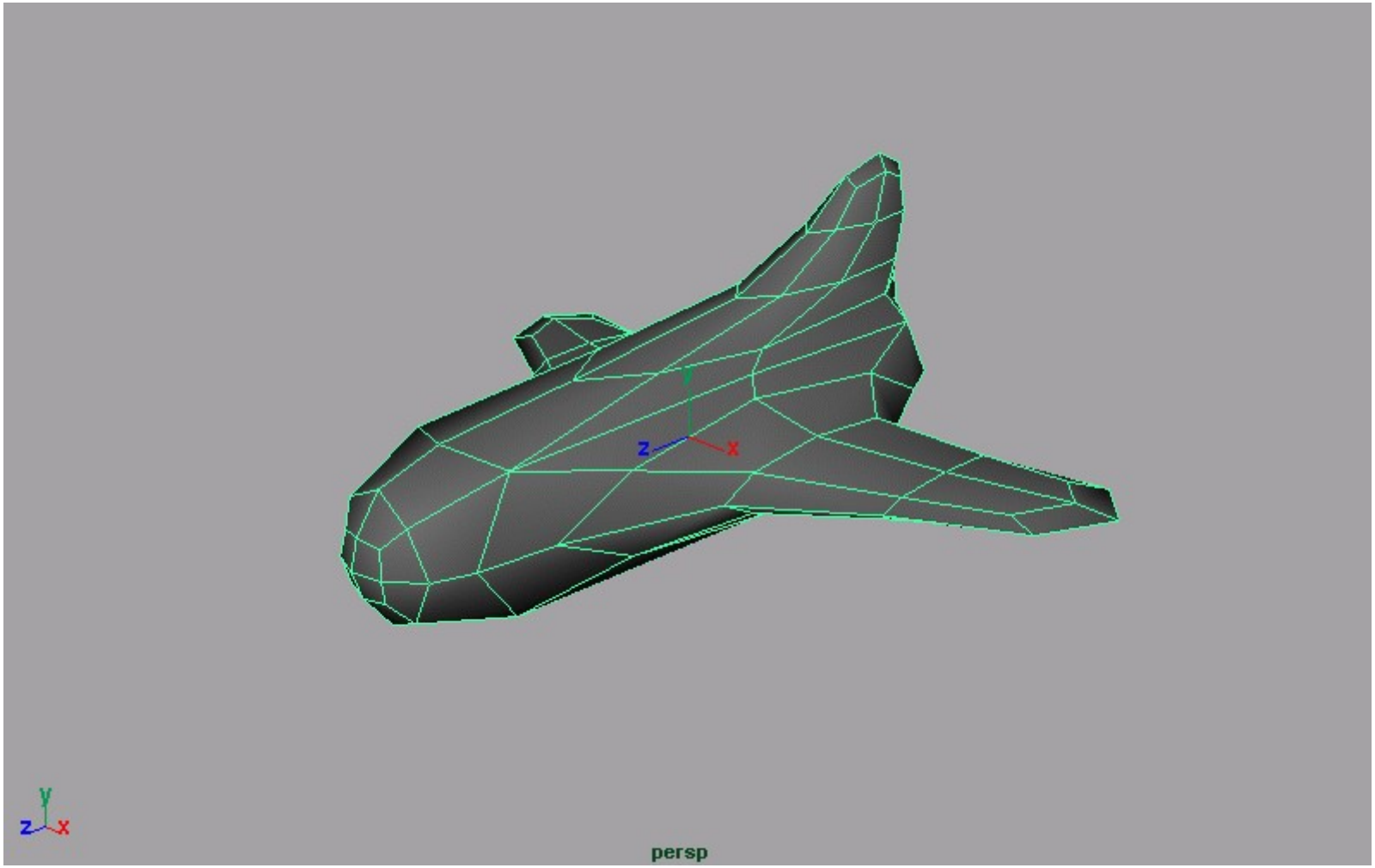


Face Point | Edge Point | Valence 3 Vertex | Valence 4 Vertex | Valance 5 Vertex

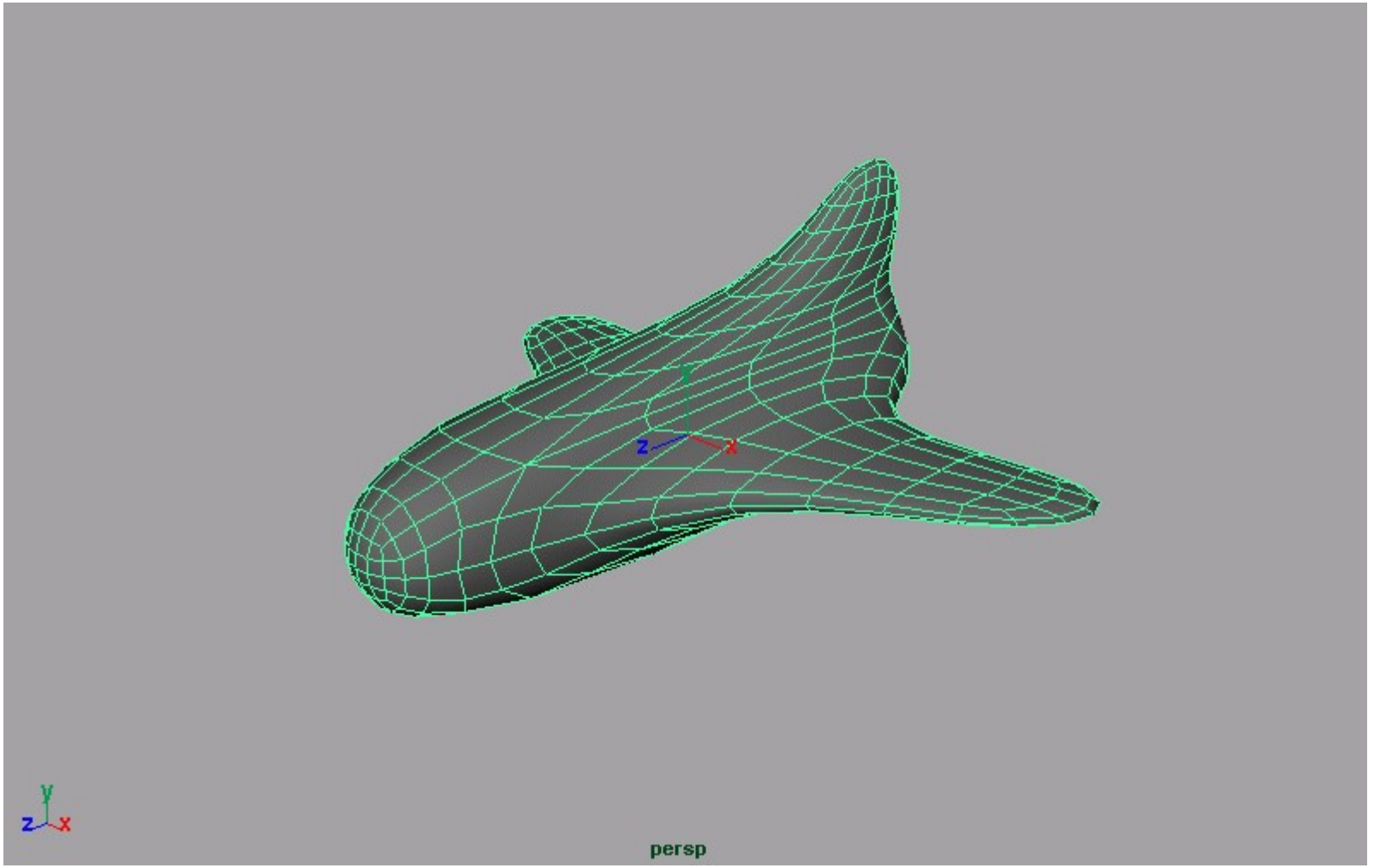
Durva poligon modell



Subdivision 1

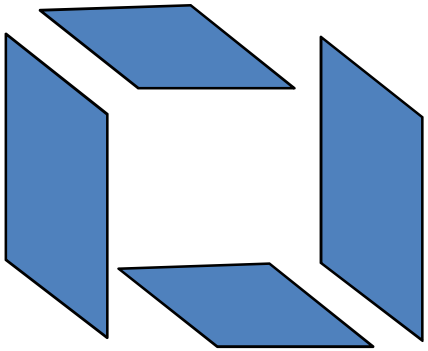


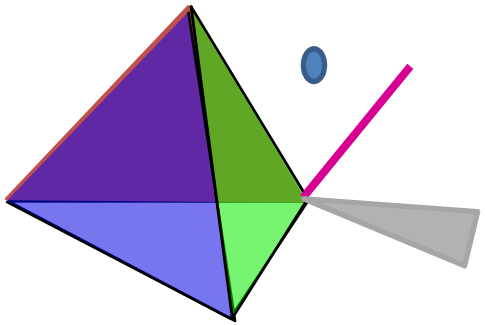
Subdivision 2



B-rep

- Test = határoló felületek



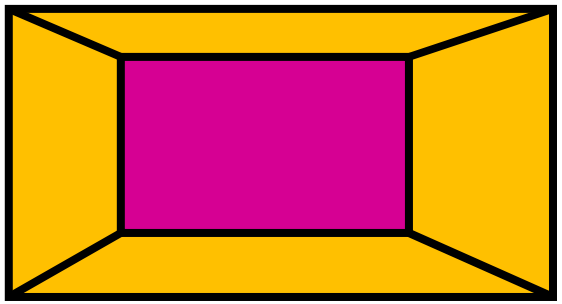


- Topológiai érvényesség:
  - Csúcspont élhez, él laphoz tartozik
  - Euler tétel ha egy darabból áll és nem lyukas:

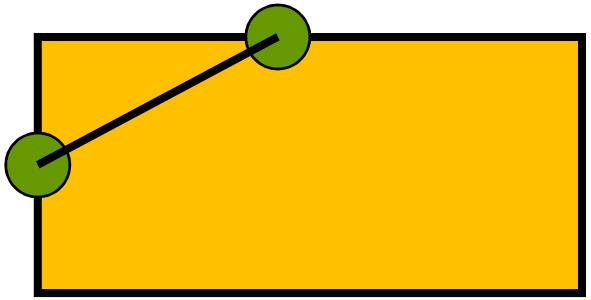
$$\text{csúcsok} + \text{lapok} = \text{élek} + 2$$

Euler operátorok

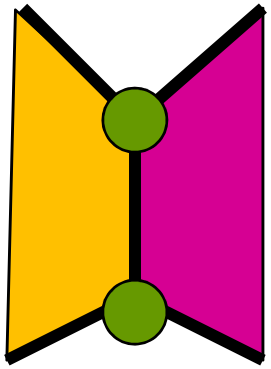
- Lap kihúzás

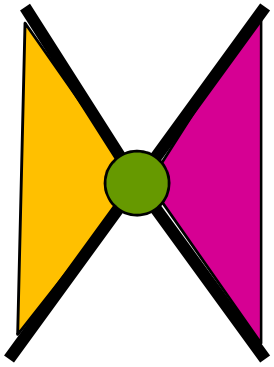


- Lap felvágás

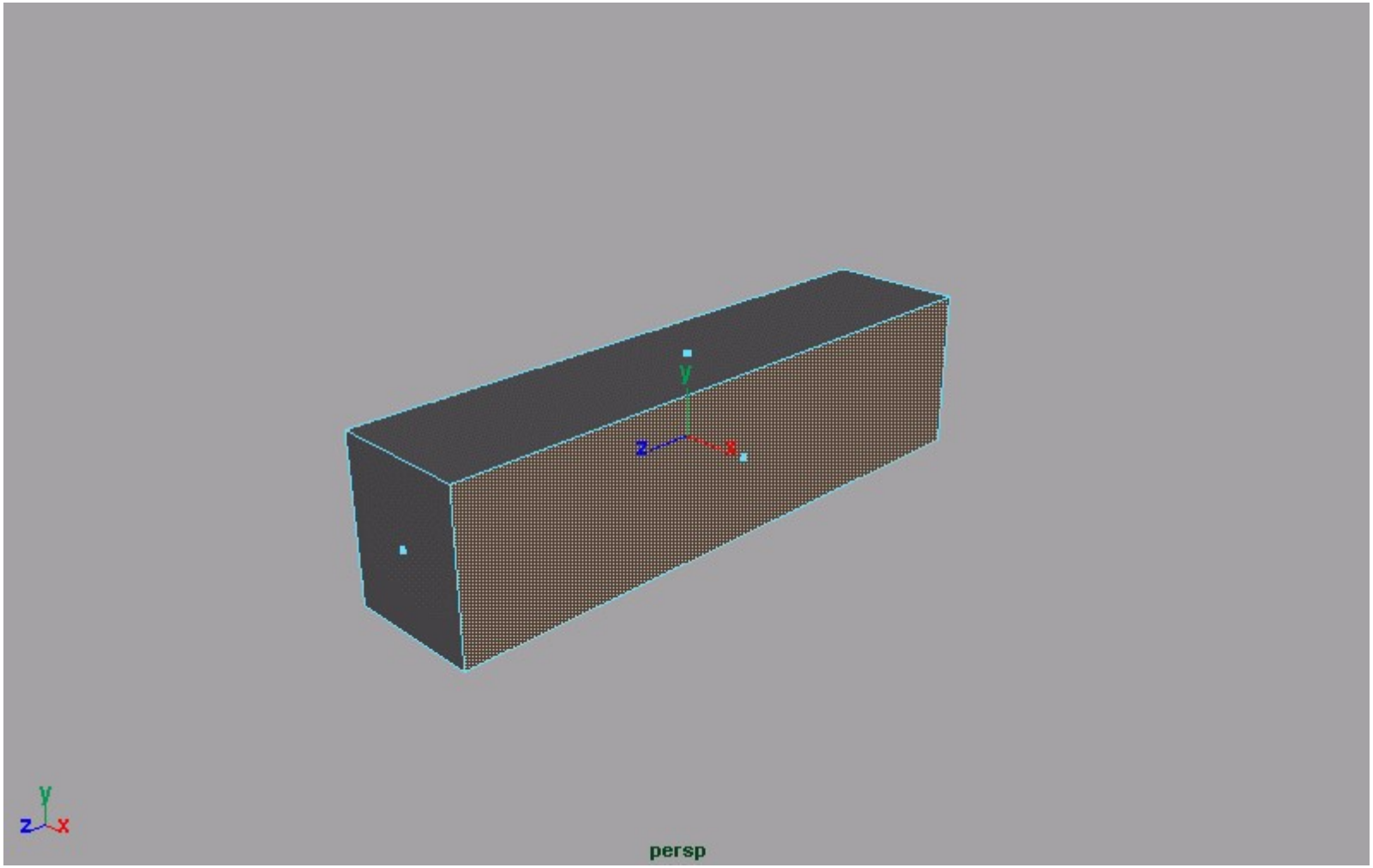


- Él törlés
- Csúcs szétvágás

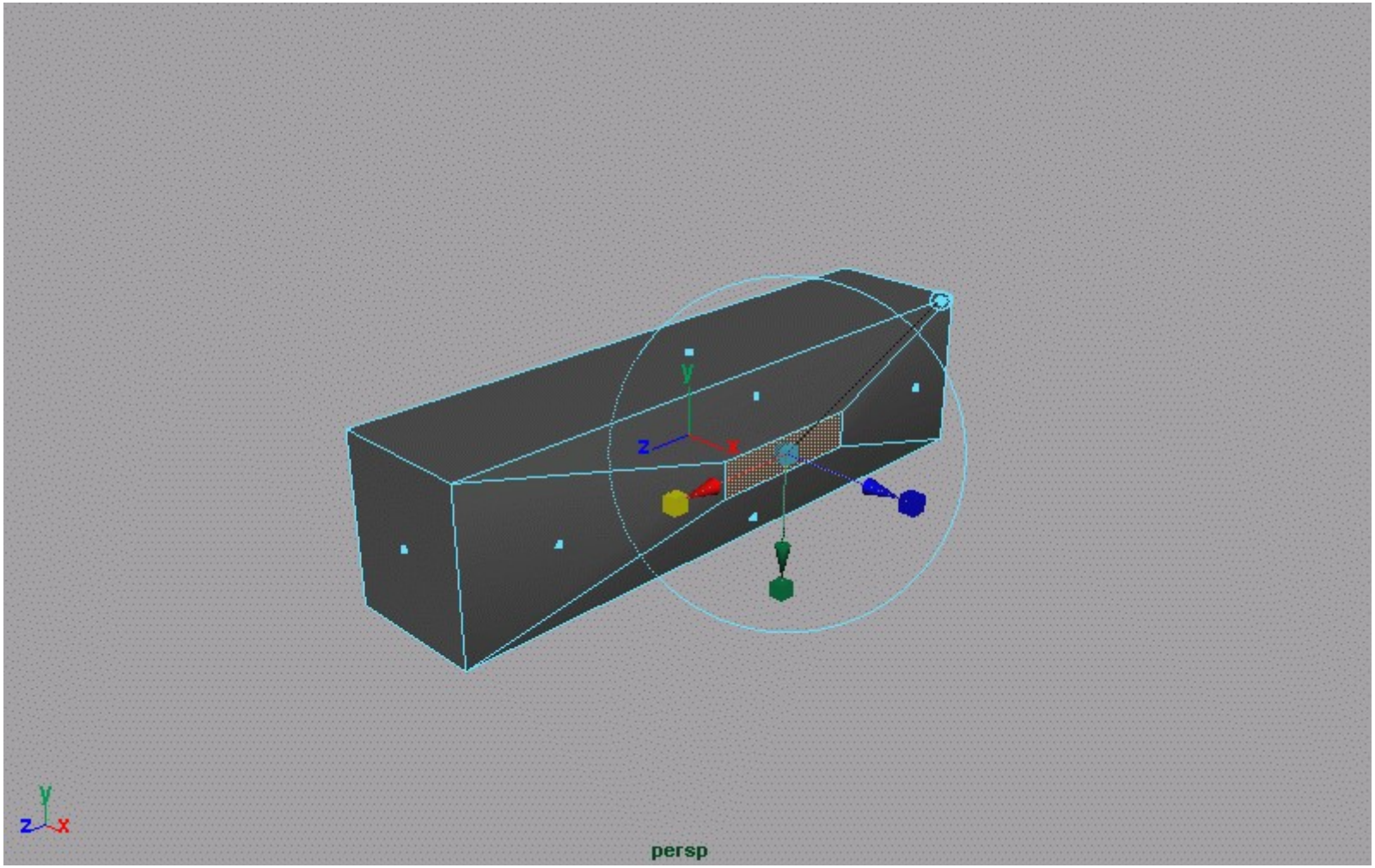




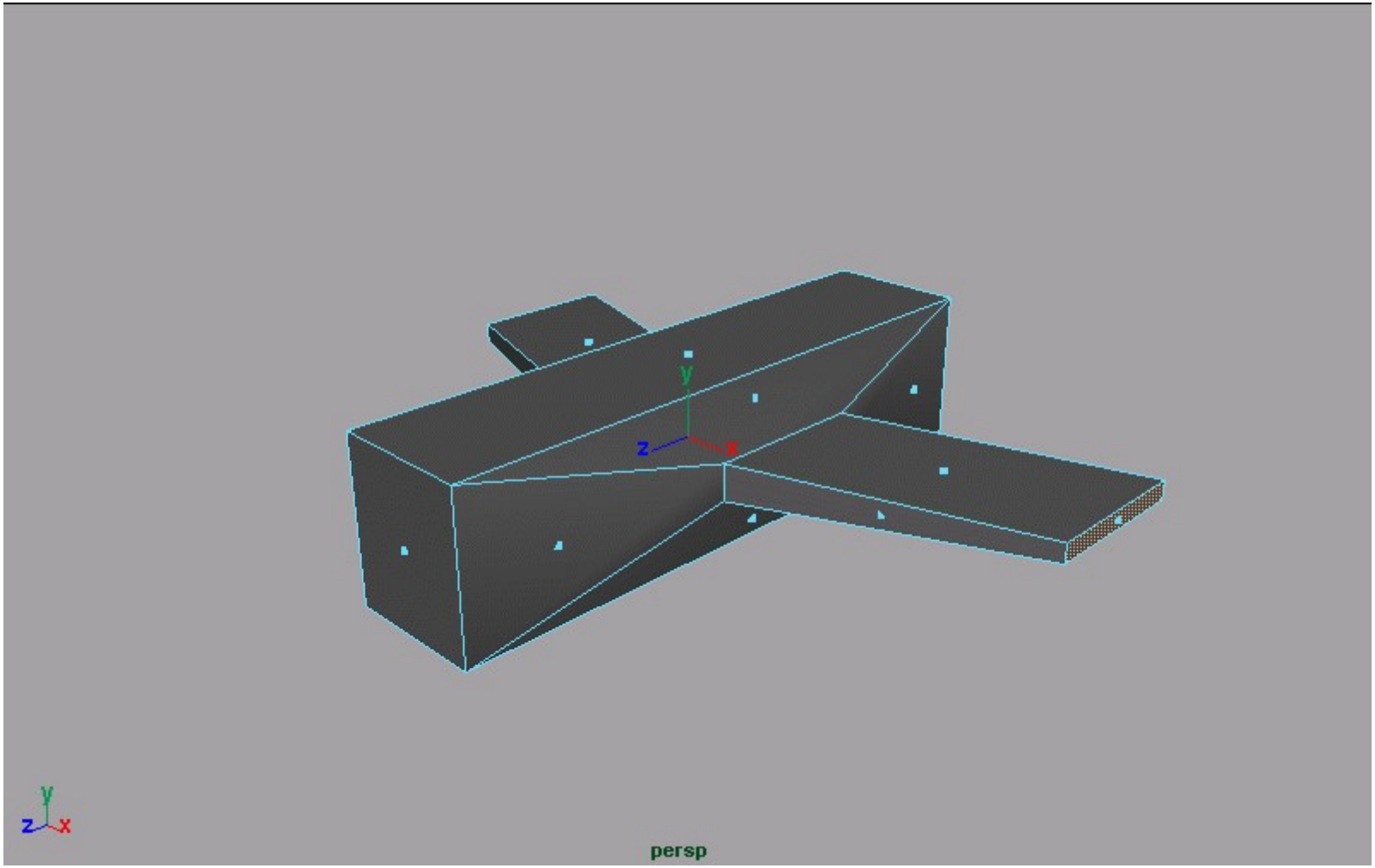
Kezdet: érvényes téglatest



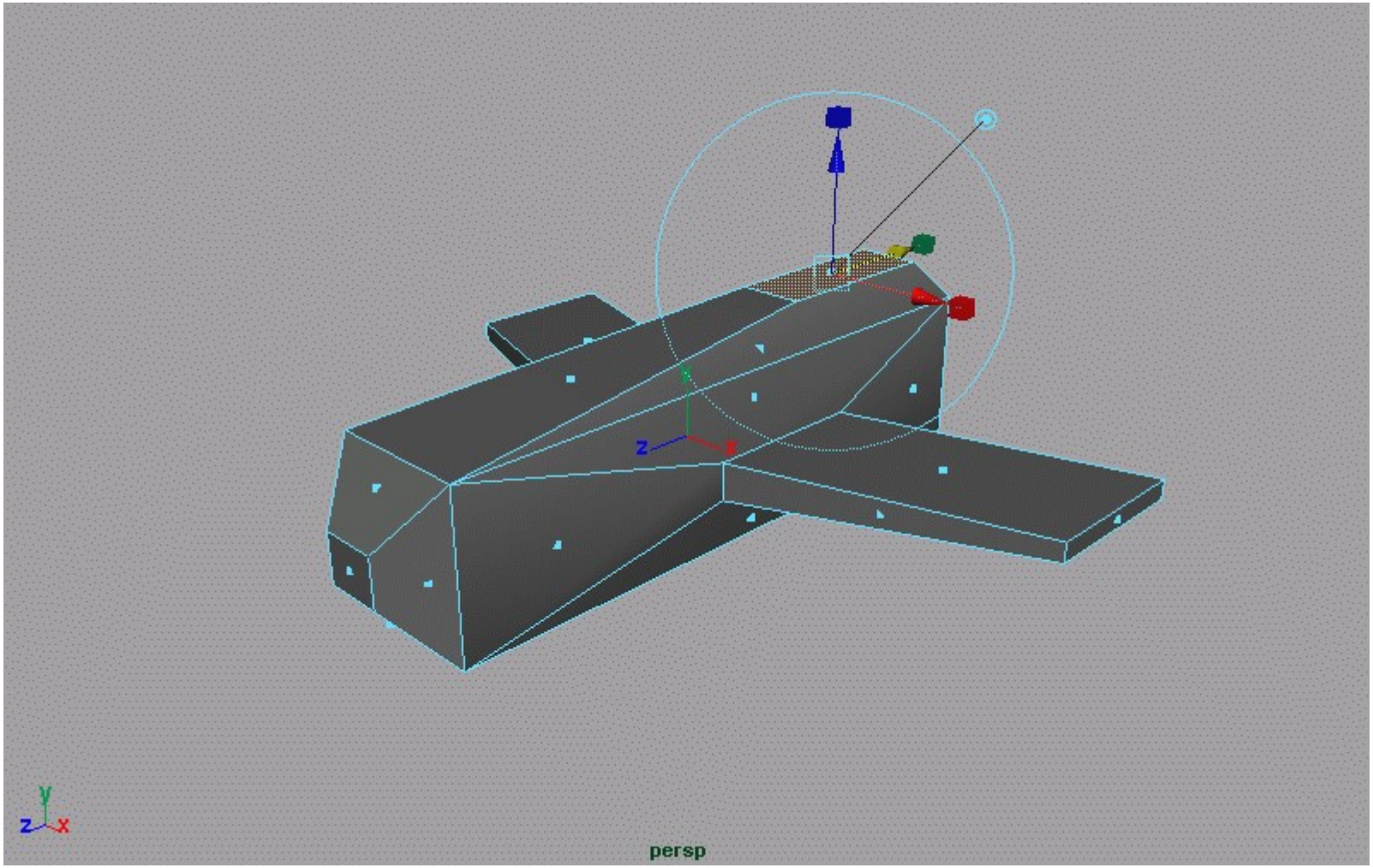
Lap kihúzás



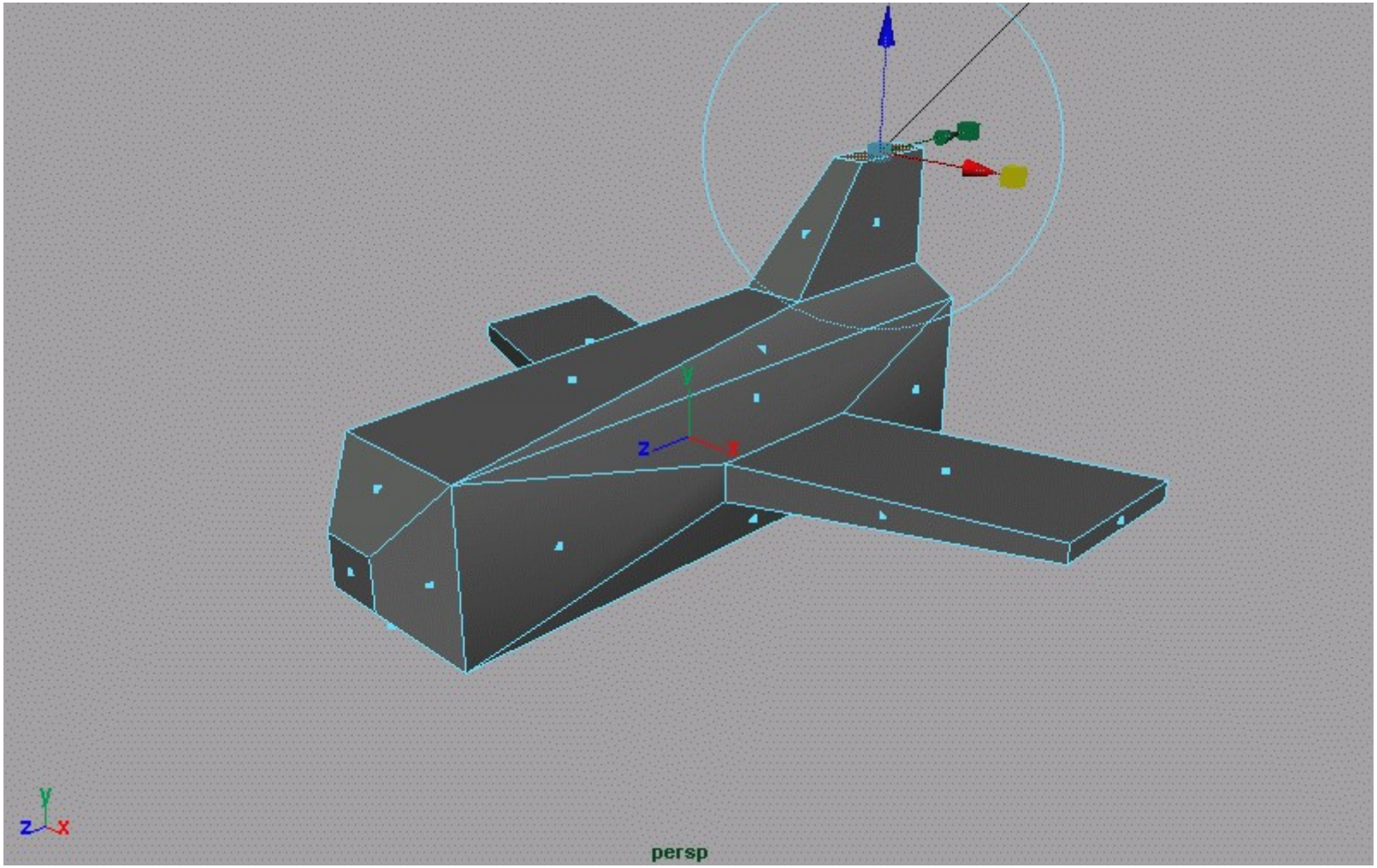
Lap kihúzás



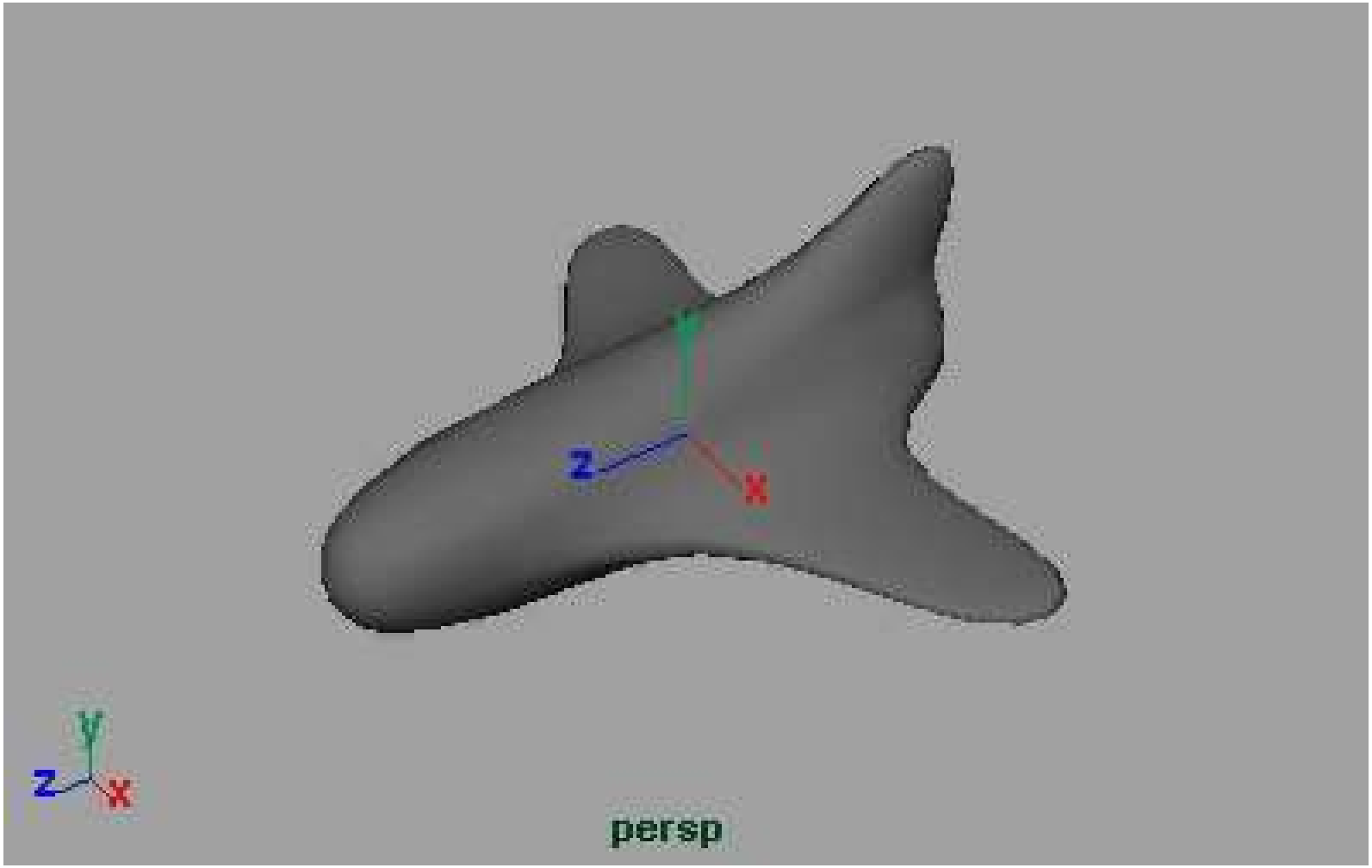
Lap kihúzás



Lap kihúzás



Subdivision simítás



Ellenőrző kérdések

- Írja fel a tórusz paraméteres és implicit egyenleteit.
- Milyen szabadformájú görbe van a PowerPoint-ban? Implementálja a saját programjában!
- Soroljon fel minél több Euler operátort!
- Tervezzen adatstruktúrát egy poliéderhez! Hogyan implementálható azon a Catmull-Clark Subdivision?
- Adja meg az egyenes másodfokú egyenletét!
- Általánosítsa az Euler tételt lyukas objektumokra.
- Írjon óraprogramot, ahol a számjegyek animált Catmull-Rom spline-ok.
- Írjon programot, amely egy diszkrét pontokban megadott függvényt interpolál, integrál és differenciál Lagrange és Catmull-Rom interpoláció felhasználásával.
- Animálja végig a sebesség és gyorsulás vektorokat a Bézier, Lagrange és Catmull-Rom görbéken.