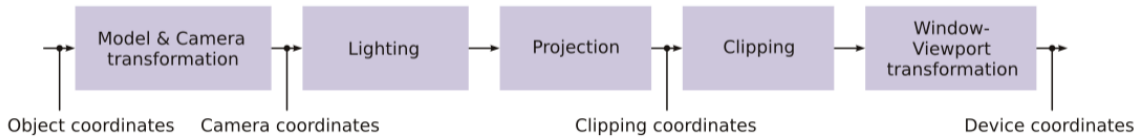
Kriptográfia

December 11, 2025

1 Geometry Pipeline



A grafikus szerelőszalag (pipeline) azon lépések sorozata, amelyeken keresztül a 3D-s geometriai adatok áthaladnak, míg végül 2D-s képpontokká (pixelekké) válnak a képernyőn. A modern GPU-k (Graphics Processing Unit) esetében ez a folyamat nagymértékben párhuzamosított és bizonyos szakaszai programozhatók.

A **vertex (plural: vertices)** is a point in the world. Many points are used to join the surfaces. In special cases, point clouds are drawn directly, but this is still the exception.

2 A programozható grafikus szerelőszalag matematikai háttere

A grafikus csővezeték (pipeline) alapja a lineáris algebra, különösen a mátrixszorzások és a homogén koordináták használata.

2.1 Homogén koordináták és transzformációk

A 3D pontokat (x, y, z) helyett homogén koordinátákkal (x, y, z, w) ábrázoljuk (általában $w = 1$). Ez lehetővé teszi, hogy az eltolást is mátrixszorzásként írjuk le.

2.1.1 Alapvető transzformációs mátrixok

- **Eltolás (Translation):**

$$T(d_x, d_y, d_z) = \begin{pmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- **Forgatás tengelyek körül (Rotation):**

  X tengely körül:

$$R_x(\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

  Y tengely körül:

$$R_y(\theta) = \begin{pmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

  Z tengely körül:

$$R_z(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- **Skálázás (Scaling):**

$$S(s_x, s_y, s_z) = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

2.2 A koordináta-rendszerek átmenetei

A vertex shaderben a csúcsok (v) a következő transzformációkon mennek keresztül:

$$v_{clip} = M_{proj} \cdot M_{view} \cdot M_{model} \cdot v_{local}$$

3 Inkrementális primitívrajzoló algoritmusok

3.1 DDA (Digital Differential Analyzer) Algoritmus

A DDA egy egyszerű, inkrementális szakaszrajzoló eljárás. Az alapötlet a szakasz egyenletének ($y = mx + b$) deriváltjából származik. Mivel az x koordinátát egységenként növeljük, az y koordináta a meredekséggel (m) változik minden lépésben.

$$m = \frac{y_1 - y_0}{x_1 - x_0} = \frac{\Delta y}{\Delta x}$$

3.1.1 Algoritmus lépései (feltéve, hogy $|m| \leq 1$)

1. **Inicializálás:** Kezdőpont (x_0, y_0), aktuális $y = y_0$.
2. **Ciklus** $x = x_0$-tól x_1-ig:
   - Rajzoljuk ki a pixelt: $(x, \text{round}(y))$.

- $x \leftarrow x + 1$
- $y \leftarrow y + m$

Hátránya: Lebegőpontos műveleteket használ (összeadás és kerekítés), ami lassabb lehet az egész számos aritmetikánál, és a kerekítési hibák felhalmozódhatnak hosszú szakaszoknál.

3.2 Bresenham Szakaszrajzoló Algoritmus

Az algoritmus eldönti, hogy egy adott x lépésnél az y koordináta maradjon-e, vagy növekedjen. Az alábbi ábra szemlélteti a rácsot (pixelek) és az ideális vonalat.

A döntési változó: $p_k = 2\Delta y \cdot x_k - 2\Delta x \cdot y_k + C$. Ez határozza meg, hogy a vonal az alsó (d_1) vagy felső (d_2) ponthoz van-e közelebb.

3.2.1 Algoritmus lépései (Egész aritmetika)

1. **Inicializálás:** $p_0 = 2\Delta y - \Delta x$.
2. **Ciklus** $k = 0$-tól $\Delta x - 1$-ig:
   - Ha $p_k < 0 \implies (x_{k+1}, y_k)$, $p_{k+1} = p_k + 2\Delta y$.
   - Ha $p_k \geq 0 \implies (x_{k+1}, y_k + 1)$, $p_{k+1} = p_k + 2\Delta y - 2\Delta x$.

3.3 Midpoint Körrajzoló Algoritmus

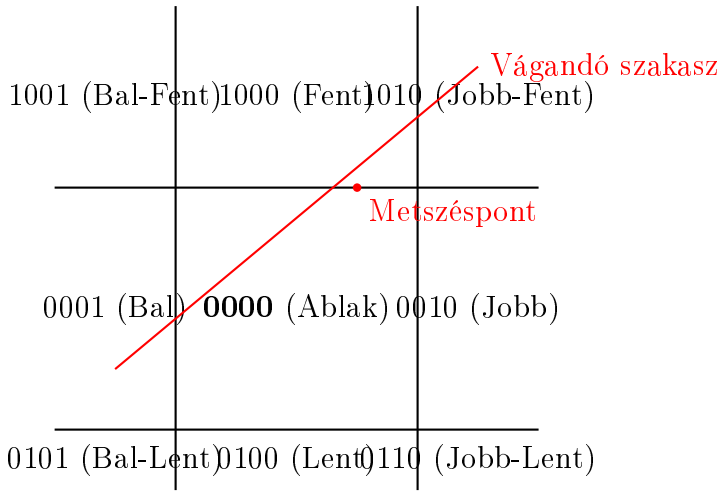
Kör egyenlete: $x^2 + y^2 - R^2 = 0$. A döntési változó (P_k) a körív $(x_k + 1, y_k - 1/2)$ felezőpontjának helyzetét vizsgálja.

$$P_k = (x_k + 1)^2 + (y_k - 1/2)^2 - R^2$$

4 Vágási algoritmusok (Clipping)

4.1 Cohen-Sutherland: Tartománykódok

A vágóablak a síkot 9 részre osztja. Minden tartományhoz egy 4 bites kódot rendelünk (Outcode: Fent, Lent, Jobbra, Balra).

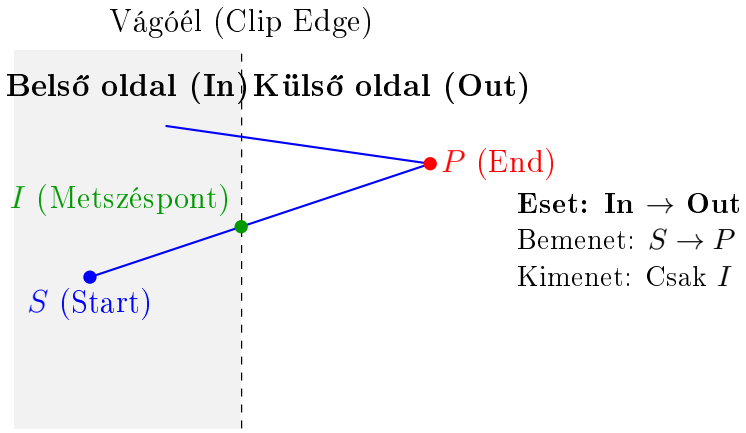


Metszéspontok képletei: Ha *Outcode* $\neq 0000$, akkor metszeni kell. Például a felső szél (y_{max}) esetén:

$$x = x_0 + \frac{1}{m}(y_{max} - y_0), \quad y = y_{max}$$

4.2 Sutherland-Hodgman Poligonvágás

Ez az algoritmus tetszőleges konvex vágóablakra (pl. téglalap) képes levágni tetszőleges poligont. A módszer lényege, hogy a poligont a vágóablak minden egyes éle mentén sorban vágjuk (pipeline-szerűen). A kimenet egy új csúcslista, amely a következő vágóél bemenete lesz.



A 4 eset egy él feldolgozásakor (ahol S az előző, P az aktuális csúcs):

1. **In → In:** Mindkét pont belül van. $\Longrightarrow$ P-t hozzáadjuk a kimeneti listához.
2. **In → Out:** A szakasz kilép. $\Longrightarrow$ A metszéspontot (I) számítjuk ki és adjuk hozzá.
3. **Out → Out:** Mindkettő kívül van. $\Longrightarrow$ Nem adunk hozzá semmit.
4. **Out → In:** A szakasz belép. $\Longrightarrow$ A metszéspontot (I), majd P-t adjuk hozzá.

5 Kitöltési algoritmusok

5.1 Pásztázó (Scan-line) algoritmus

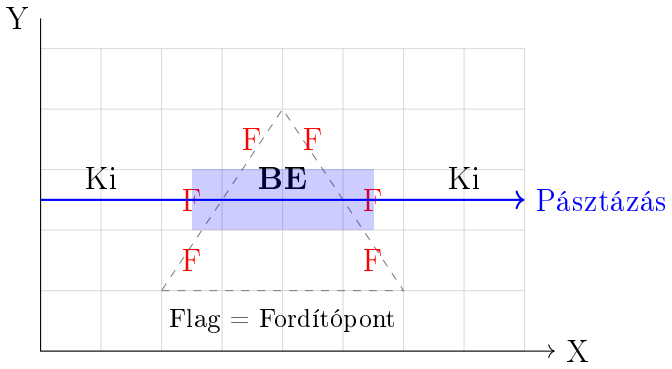
A sokszögkitöltés kulcsa az élek metszéspontjainak rendezése.

1. **ET (Edge Table):** Élek tárolása y_{min} szerint.
2. **AET (Active Edge Table):** Az aktuális pásztázó sort metsző élek.
3. **Párosítás:** A rendezett metszéspontok között (x_{2i}, x_{2i+1}) színezzük a pixeleket.

5.2 Él-flag (Edge-Flag) módszer

Ez az algoritmus a kitöltést két különálló lépésben végzi a képmemóriában. Hardveresen könnyen gyorsítható.

1. **1. menet (Kontúrrajzolás):** A sokszög körvonalát kirajzoljuk, de ahelyett, hogy színt írnánk a pixelbe, invertáljuk (XOR) a memóriában lévő értéket (vagy egy külön jelzőbitet). Fontos szabályok:
   - A vízszintes éleket figyelmen kívül hagyjuk.
   - A csúcspontoknál a szomszédos élek Y irányát figyeljük (lokális minimum/-maximum kezelése).
2. **2. menet (Kitöltés):** Végigolvassuk a képernyőt (soronként, balról jobbra). Egy *Inside* (Bent) logikai változót tartunk karban.
   - Kezdetben Inside = False.
   - Ha olyan pixelhez érünk, ahol a Flag be van állítva: Inside = !Inside.
   - Ha Inside == True: A pixelt kiszínezzük a kitöltő színnel.

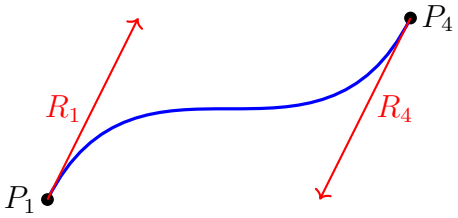


6 Görbék matematikai leírása

6.1 Hermite-görbe (Interpoláló)

A Hermite-görbe egy harmadfokú görbe, amelyet két végpontja (P_1, P_4) és a végpontokban vett érintővektorok (R_1, R_4) határoznak meg.

Interpolálja a végpontokat,
követi az érintőket.



6.1.1 Matematikai alak (Polinom)

A görbe egyenlete felírható a súlyfüggvények segítségével:

$$Q(t) = H_1(t)P_1 + H_2(t)P_4 + H_3(t)R_1 + H_4(t)R_4$$

Ahol a $H(t)$ bázisfüggvények (geometriai súlyfüggvények):

$$H_1(t) = 2t^3 - 3t^2 + 1 \quad \text{(Kezdőpont súlya)}$$

$$H_2(t) = -2t^3 + 3t^2 \quad \text{(Végpont súlya)}$$

$$H_3(t) = t^3 - 2t^2 + t \quad \text{(Kezdő érintő súlya)}$$

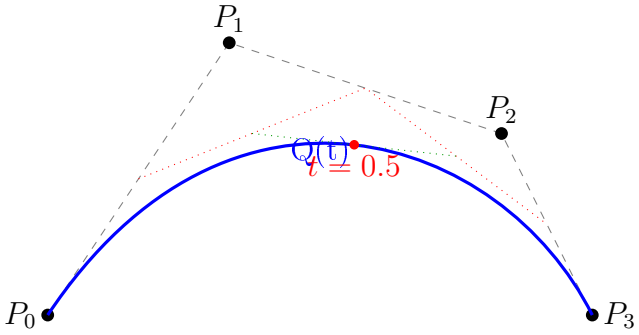
$$H_4(t) = t^3 - t^2 \quad \text{(Vég érintő súlya)}$$

6.1.2 Mátrixos alak

$$Q(t) = \begin{pmatrix} t^3 & t^2 & t & 1 \end{pmatrix} \begin{pmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{pmatrix}$$

6.2 Bézier-görbék

A harmadfokú (kubikus) Bézier-görbe 4 kontrollponttal (P_0, P_1, P_2, P_3) definiálható. A görbe a P_0-ból indul és P_3-ba érkezik, miközben P_1 és P_2 vonzza magához az ívet.



6.2.1 Matematikai alak (Bernstein-polinomok)

A görbe általános alakja a Bernstein-polinomokkal:

$$Q(t) = \sum_{i=0}^{3} B_{i,3}(t)P_i = (1-t)^3P_0 + 3t(1-t)^2P_1 + 3t^2(1-t)P_2 + t^3P_3$$

A súlyfüggvények tulajdonsága, hogy összegük minden t-re 1 (ezért marad a görbe a konvex burokban).

6.2.2 Mátrixos alak

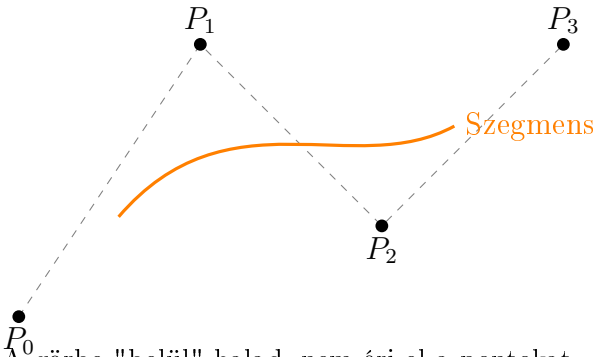
$$Q(t) = \begin{pmatrix} t^3 & t^2 & t & 1 \end{pmatrix} \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{pmatrix}$$

6.3 B-Spline görbék

A B-Spline görbék (Basis Spline) általánosítják a Bézier-görbéket, lehetővé téve a **lokális kontrollt**. Ez azt jelenti, hogy egy kontrollpont mozgatása csak a görbe egy kis szakaszát befolyásolja. Nem feltétlenül interpolálja a végpontokat.

6.3.1 Uniform Kubikus B-Spline

A leggyakoribb típus a grafikában. Egy görbeszegmenst 4 kontrollpont határoz meg $(P_{i-1}, P_i, P_{i+1}, P_{i+2})$, de a görbe nem halad át egyikükön sem (hacsak nem esnek egybe). A görbék C^2 folytonossággal kapcsolódnak.



A görbe "belül" halad, nem éri el a pontokat.

6.3.2 Matematikai alak (Bázisfüggvények)

A görbeszegmens egyenlete a súlyfüggvényekkel:

$$Q_i(t) = N_0(t)P_{i-1} + N_1(t)P_i + N_2(t)P_{i+1} + N_3(t)P_{i+2}$$

Ahol a súlyfüggvények:

$$N_0(t) = \frac{1}{6}(1-t)^3$$

$$N_1(t) = \frac{1}{6}(3t^3 - 6t^2 + 4)$$

$$N_2(t) = \frac{1}{6}(-3t^3 + 3t^2 + 3t + 1)$$

$$N_3(t) = \frac{1}{6}t^3$$

6.3.3 Mátrixos alak (egy szegmensre)

$$Q_i(t) = \frac{1}{6}\begin{pmatrix} t^3 & t^2 & t & 1 \end{pmatrix}\begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{pmatrix}\begin{pmatrix} P_{i-1} \\ P_i \\ P_{i+1} \\ P_{i+2} \end{pmatrix}$$

6.3.4 Cox-de Boor formula (Általános eset)

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

7 Koordináta-rendszerek

A számítógépes grafikában a geometria leírásához elengedhetetlen a megfelelő koordináta-rendszer kiválasztása.

7.1 Descartes-féle koordináta-rendszer (Cartesian)

Ez a legismertebb rendszer, amely egymásra merőleges tengelyekből áll (X, Y síkban, X, Y, Z térben).

- **Bázisvektorok:** $i(1,0,0), j(0,1,0), k(0,0,1)$.
- **Jobbsodrású rendszer (Right-handed):** Ha a jobb kéz hüvelykujja X, mutatóujja Y, akkor a középső ujj Z irányába mutat (OpenGL alapértelmezés).
- **Balsodrású rendszer (Left-handed):** A Z tengely a képernyőbe befelé mutat (DirectX alapértelmezés).

7.2 Baricentrikus koordináta-rendszer

Különösen fontos a háromszögek raszterizálásánál és a sugárkövetésnél (ray tracing). Egy síkbeli pontot egy háromszög csúcspontjaihoz (A, B, C) viszonyítva adunk meg.

Egy P pont előállítható a csúcsok súlyozott összegeként:

$$P = \alpha A + \beta B + \gamma C$$

Ahol a súlyokra igaz, hogy:

$$\alpha + \beta + \gamma = 1$$

- Ha $0 \leq \alpha, \beta, \gamma \leq 1$, akkor a pont a háromszög **belsejében** van.
- A koordináták arányosak a rész-háromszögek területeivel.

7.3 Homogén koordináta-rendszer

A projektív geometria eszköze. A síkbeli (x, y) pontot (x, y, w) hármassal, a térbeli (x, y, z) pontot (x, y, z, w) négyessel jelöljük.

- Áttérés Descartes-ra: $X = x/w, Y = y/w, Z = z/w$.
- Ha $w = 1$, az egy hagyományos pont.

- Ha $w = 0$, az egy **irányvektor** (végtelen távoli pont).

Jelentősége: Lehetővé teszi, hogy az **eltolást (transzláció)** és a **perspektivikus vetítést** is mátrixszorzásként írjuk le (lineáris transzformációvá teszi az affin transzformációkat).

8 Részletes Transzformációk: 2D és 3D Mátrixok

Ebben a fejezetben összefoglaljuk az alapvető geometriai transzformációkat. Megkülönböztetjük a hagyományos **Descartes-i (Heterogén)** alakot (egyenletrendszerek) és a számítógépes grafikában elterjedt **Homogén** mátrixos alakot.
Megjegyzés: A jelölésben oszlopvektorokat használunk ($v' = M \cdot v$).

8.1 Identitás (Egységmátrix)

Ez a transzformáció nem változtatja meg a pont helyét.

Heterogén alak: $x' = x, \quad y' = y, \quad z' = z.$

2D Homogén (3x3)	**3D Homogén (4x4)**
$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$I = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

8.2 Eltolás (Translation)

Az eltolás nem lineáris transzformáció a Descartes-i térben (nem reprezentálható 2×2-es vagy 3×3-as szorzással), csak vektorösszeadással. Homogén koordinátákkal azonban mátrixszorzássá válik.
Paraméterek: d_x, d_y (és d_z térben).

Heterogén alak:

- 2D: $x' = x + d_x, \quad y' = y + d_y$
- 3D: $x' = x + d_x, \quad y' = y + d_y, \quad z' = z + d_z$

2D Homogén (3x3)	**3D Homogén (4x4)**
$T = \begin{bmatrix} 1 & 0 & d_x \\ 0 & 1 & d_y \\ 0 & 0 & 1 \end{bmatrix}$	$T = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$

8.3 Méretezés / Skálázás (Scaling)

A tengelyek mentén történő nyújtás vagy zsugorítás. Ha $s_x = s_y (= s_z)$, akkor egyenletes (hasonlósági) skálázásról beszélünk.
Paraméterek: s_x, s_y (és s_z).

Heterogén alak:

- 2D: $x' = x \cdot s_x, \quad y' = y \cdot s_y$
- 3D: $x' = x \cdot s_x, \quad y' = y \cdot s_y, \quad z' = z \cdot s_z$

2D Homogén (3x3)	**3D Homogén (4x4)**
$S = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$S = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

8.4 Forgatás (Rotation)

A forgatás az origó (vagy térben egy tengely) körül történik α szöggel. Pozitív szög esetén az óramutató járásával ellentétes irányba.

8.4.1 2D Forgatás (Origó körül)

Heterogén alak:

$$x' = x\cos\alpha - y\sin\alpha$$
$$y' = x\sin\alpha + y\cos\alpha$$

Homogén (3x3):

$$R(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

8.4.2 3D Forgatás (Főtengelyek körül)

Térben 4x4-es mátrixokat használunk.

- **Z-tengely körül (síkforgatás kiterjesztése):**
  Heterogén: $x' = x\cos\alpha - y\sin\alpha, \quad y' = x\sin\alpha + y\cos\alpha, \quad z' = z$

$$R_z(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- **X-tengely körül:**
  Heterogén: $y' = y\cos\alpha - z\sin\alpha, \quad z' = y\sin\alpha + z\cos\alpha, \quad x' = x$

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- **Y-tengely körül:**
  Heterogén: $z' = z\cos\alpha - x\sin\alpha, \quad x' = z\sin\alpha + x\cos\alpha, \quad y' = y$

$$R_y(\alpha) = \begin{bmatrix} \cos\alpha & 0 & \sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

8.5 Tükrözés (Reflection)

A tükrözés speciális skálázásnak is felfogható, ahol a skálázási tényező -1. A determináns értéke -1.

2D Tükrözés (Tengelyekre)	**3D Tükrözés (Síkokra)**
X-tengelyre ($y \to -y$): *Heterogén:* $x' = x, y' = -y$ $M_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	XY síkra ($z \to -z$): *Heterogén:* $x' = x, y' = y, z' = -z$ $M_{xy} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Origóra ($x \to -x, y \to -y$): *Heterogén:* $x' = -x, y' = -y$ $M_o = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	Origóra: *Heterogén:* $x' = -x, y' = -y, z' = -z$ $M_o = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

8.6 Nyírás (Shearing)

A nyírás során az egyik koordináta értéke függ egy másik koordinátától. Affin transzformáció, párhuzamosságot megőriz, de a szögeket és távolságokat nem.

8.6.1 2D Nyírás

X-irányú nyírás (az X koordináta változik Y függvényében):
Heterogén: $x' = x + sh_x \cdot y, \quad y' = y.$

$$Sh_x = \begin{bmatrix} 1 & sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Y-irányú nyírás:
Heterogén: $x' = x, \quad y' = y + sh_y \cdot x.$

$$Sh_y = \begin{bmatrix} 1 & 0 & 0 \\ sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

8.6.2 3D Nyírás

Itt több lehetőség van. Például, ha a Z koordinátát nem változtatjuk, de X-et és Y-t a Z függvényében nyírjuk:
Heterogén: $x' = x + sh_x \cdot z, \quad y' = y + sh_y \cdot z, \quad z' = z.$

$$H_{xy(z)} = \begin{bmatrix} 1 & 0 & sh_x & 0 \\ 0 & 1 & sh_y & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

9 Transzformációk osztályozása és szorzata

A geometriai transzformációkat aszerint csoportosítjuk, hogy milyen invariánsokat (változatlan tulajdonságokat) őriznek meg.

9.1 Osztályozás (Klein-féle program szerint)

1. **Egybevágósági transzformációk (Izometriák):**
   - Megőrzi: Távolságot, szögeket, területeket.
   - Példa: Eltolás, Forgatás, Tükrözés.
   - Mátrix alak: $\det(M) = \pm 1$ (ortogonális rész).
2. **Hasonlósági transzformációk:**
   - Megőrzi: Szögeket, távolságok *arányát*.
   - Példa: Egyenletes skálázás ($s_x = s_y = s_z$).
3. **Affin transzformációk:**
   - Megőrzi: Párhuzamosságot, osztóviszonyt (szakaszok arányát egy egyenesen).
   - Példa: Nyírás (Shearing), nem egyenletes skálázás.

- Mátrix alakja: Az utolsó sor mindig $[0, 0, 0, 1]$.

$$M_{affin} = \begin{bmatrix} A & t \\ 0^T & 1 \end{bmatrix}$$

  ahol A a lineáris rész (3x3), t az eltolás.

4. **Projektív transzformációk:**
   - Megőrzi: Egyenest (egyenes képe egyenes), illeszkedést.
   - **NEM** őrzi meg a párhuzamosságot (enyészpontok keletkeznek).
   - Mátrix alakja: Az utolsó sor bármi lehet. Ez felelős a perspektív torzításért.

9.2 Transzformációk szorzata

Több transzformáció egymásutánja egyetlen mátrixszal helyettesíthető.

$$v' = M_n \cdot M_{n-1} \cdot \dots \cdot M_2 \cdot M_1 \cdot v$$

$$M_{ered} = M_n \cdot \dots \cdot M_1$$

Fontos: A mátrixszorzás **nem kommutatív**! ($A \cdot B \neq B \cdot A$). Például: Nem mindegy, hogy előbb forgatunk és utána toljuk el a koordinátarendszert, vagy fordítva.

10 Koordináta-transzformációk

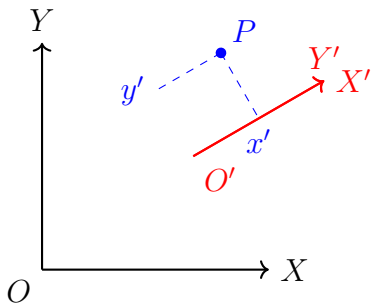
Kétféleképpen értelmezhetjük a transzformációkat:

1. **Ponttranszformáció:** A koordinátarendszer rögzített, az objektum pontjai mozognak benne.
2. **Koordináta-transzformáció (Báziscsere):** A pont a térben rögzített, de a koordinátarendszert (bázist) mozgatjuk alatta, így a pont koordinátái megváltoznak.

Ha egy K_1 rendszerből áttérünk egy K_2-be, és a K_2 rendszer origója és bázisvektorai K_1-ben kifejezve M mátrixszal írhatók le (ahol az oszlopok az új bázisvektorok), akkor a pont új koordinátái:

$$v_{K2} = M^{-1} \cdot v_{K1}$$

Tehát a koordináta-transzformáció mátrixa az objektum-transzformáció mátrixának az inverze.



11 Tér leképezése síkra (Viewing Pipeline)

A 3D színtér síkra vetítése több lépésből áll:

$$P_{screen} = M_{viewport} \cdot M_{proj} \cdot M_{view} \cdot M_{model} \cdot P_{local}$$

11.1 Viewing Transzformáció (Nézeti transzformáció)

A transzformáció célja, hogy a világkoordinátákat (P_{world}) átalakítsa a kamera koordinátarendszerébe (P_{view}). Ez technikailag a kamera mozgatásának inverze: ha a kamerát $+10$-zel toljuk Z-ben, az egyenértékű azzal, ha a világot -10-zel toljuk Z-ben.

11.1.1 Matematikai levezetés (Báziscsere)

Adott a kamera három paramétere (LookAt függvény):

- E (Eye): A kamera pozíciója (Origó lesz az új rendszerben).
- C (Center/Target): A pont, ahova a kamera néz.
- U (Up): A "felfelé" mutató irány (közelítőleg).

Az új koordinátarendszer (u, v, n bázisvektorok) kiszámítása (Jobbsodrású rendszer, kamera $-Z$ felé néz):

1. **Előre vektor (n vagy Z_{axis}):** A kamera $-Z$ irányba néz, tehát a $+Z$ irány a kamerától kifelé mutat (hátra).
$$n = \frac{E - C}{||E - C||}$$
2. **Jobbra vektor (u vagy X_{axis}):** Az Up vektor és az n vektor keresztszorzata.
$$u = \frac{U \times n}{||U \times n||}$$
3. **Fel vektor (v vagy Y_{axis}):** A korrigált felfelé irány (hogy a bázis ortogonális legyen).
$$v = n \times u$$

11.1.2 A Mátrix felépítése

A Viewing mátrix (M_{view}) két transzformációból áll: egy eltolásból (T), amely a kamera pozícióját (E) az origóba viszi, és egy forgatásból (R), amely a világ bázisát a kamera bázisába (u, v, n) forgatja. Mivel koordináta-rendszer transzformációról van szó (inverz), a forgatási mátrix sorai az új bázisvektorok lesznek.

$$M_{view} = R \cdot T$$

$$T = \begin{bmatrix} 1 & 0 & 0 & -E_x \\ 0 & 1 & 0 & -E_y \\ 0 & 0 & 1 & -E_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{és} \quad R = \begin{bmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ n_x & n_y & n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$M_{view} = \begin{bmatrix} u_x & u_y & u_z & -(u \cdot E) \\ v_x & v_y & v_z & -(v \cdot E) \\ n_x & n_y & n_z & -(n \cdot E) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

11.1.3 Számítási Példa

Legyen $E = (0, 0, 5)$ (Kamera a Z-tengelyen), $C = (0, 0, 0)$ (Origóba néz), $U = (0, 1, 0)$.

1. $n = (0, 0, 5) - (0, 0, 0) = (0, 0, 5) \rightarrow$ Normálva: $(0, 0, 1)$. 2. $u = (0, 1, 0) \times (0, 0, 1) = (1, 0, 0)$. 3. $v = (0, 0, 1) \times (1, 0, 0) = (0, 1, 0)$.

Mátrix összeállítása:

$$-u \cdot E = -(1 \cdot 0 + 0 \cdot 0 + 0 \cdot 5) = 0$$

$$-v \cdot E = -(0 \cdot 0 + 1 \cdot 0 + 0 \cdot 5) = 0$$

$$-n \cdot E = -(0 \cdot 0 + 0 \cdot 0 + 1 \cdot 5) = -5$$

$$M_{view} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Ez megfelel annak, hogy a világot 5 egységgel eltoljuk a $-Z$ irányba (a kamera elé).

11.2 Vetítések (Projection)

11.2.1 Párhuzamos (Ortografikus) vetítés

A vetítési sugarak párhuzamosak. Megőrzi a méreteket, függetlenül a távolságtól (mérnöki ábrázolás). Mátrixa egyszerűen skálázza és eltolja a koordinátákat a $[-1, 1]$ kockába (NDC).

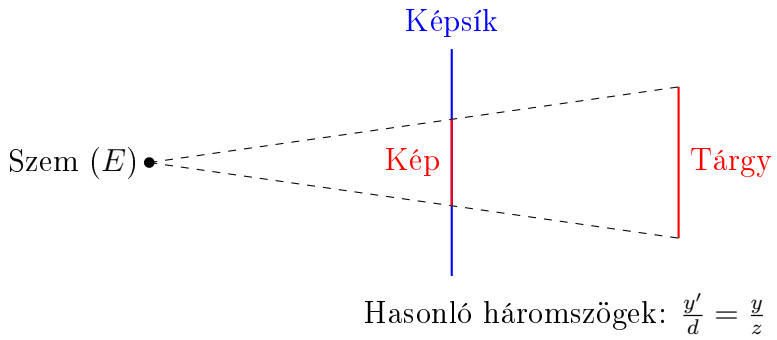
11.2.2 Perspektív vetítés

A vetítési sugarak egy közös pontba (szem/kamera) futnak össze. A távolabbi tárgyak kisebbnek látszanak. Ez a w koordinátával történő osztással érhető el (perspektivikus osztás).

A vetítési mátrix egyszerűsített alakja (ahol d a fókusztávolság):

$$M_{persp} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix}$$

Ekkor a homogén koordináta: $P(x, y, z, z/d)$. Az osztás után: $P'(\frac{x}{z/d}, \frac{y}{z/d}, d)$. Tehát az x' és y' koordináták fordítottan arányosak z-vel.



12 Felületreprezentációs módszerek

A 3D-s objektumok felületének leírására többféle matematikai megközelítés létezik, amelyek eltérő alkalmazásokban előnyösek.

12.1 Explicit reprezentáció

A felületet egy függvényként adjuk meg, ahol az egyik koordináta a másik kettő függvénye.

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

- **Normálvektor számítása:** A felület érintősíkjának normálisa a parciális deriváltakból számolható.
$$n = \left(-\frac{\partial f}{\partial x}, 1, -\frac{\partial f}{\partial z} \right)$$
- **Hátránya:** Nem képes zárt alakzatokat (pl. gömb) leírni.

12.2 Implicit reprezentáció

A felületet egy egyenlet megoldáshalmazaként definiáljuk: $F(x, y, z) = 0$. Példa (Gömb): $x^2 + y^2 + z^2 - R^2 = 0$.

12.2.1 Matematikai háttér: A Gradiens

Az implicit felületek legfontosabb tulajdonsága, hogy a felület normálvektora (N) bármely $P(x, y, z)$ pontban megegyezik a függvény gradiensvektorával (∇F) abban a pontban.

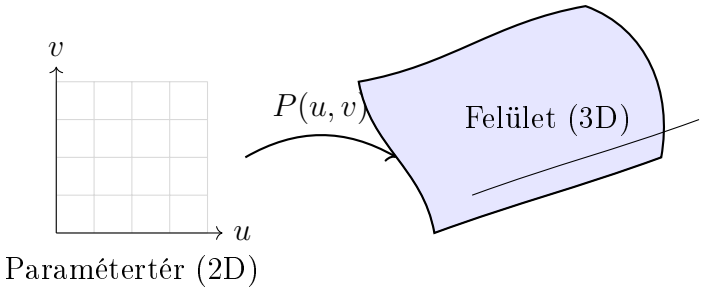
$$N(x, y, z) = \nabla F = \begin{pmatrix} \frac{\partial F}{\partial x} \\ \frac{\partial F}{\partial y} \\ \frac{\partial F}{\partial z} \end{pmatrix}$$

Példa: Gömb esetén $N = (2x, 2y, 2z)$, ami (normalizálva) valóban a középpontból kifelé mutató vektor.

12.3 Paraméteres felületek

A felület pontjait két paraméter (u, v) függvényében adjuk meg:

$$P(u, v) = \begin{pmatrix} x(u, v) \\ y(u, v) \\ z(u, v) \end{pmatrix}, \quad u, v \in [0, 1]$$



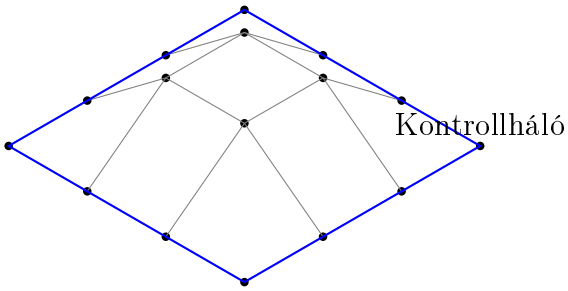
12.3.1 Differenciálgeometria: Érintők és Normális

A felületi normálvektor a két érintővektor vektoriális szorzata:

$$N(u,v) = \frac{T_u \times T_v}{||T_u \times T_v||}, \quad \text{ahol } T_u = \frac{\partial P}{\partial u}, T_v = \frac{\partial P}{\partial v}$$

12.4 Bikubikus felületek (Bézier-felületek)

A Bézier-felületet 16 darab kontrollpont ($P_{00} \dots P_{33}$) határozza meg. A felület "kifeszül" ezen pontok közé.



A mátrixos egyenlet:

$$Q(u,v) = U \cdot M \cdot G \cdot M^T \cdot V^T$$

12.4.1 Bézier-felület normálvektora

A fényeléshez szükségünk van a normálvektorra. Ehhez deriválnunk kell a $Q(u,v)$ függvényt u és v szerint.

$$N = \frac{\partial Q}{\partial u} \times \frac{\partial Q}{\partial v}$$

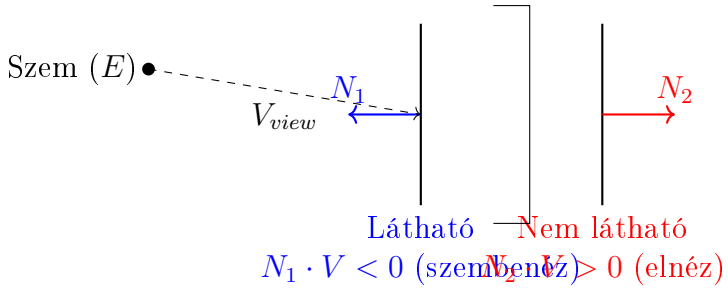
13 Felület leíró adatstruktúrák

(A korábbi leírás változatlan: Csúcslista, Laplista, Winged-Edge).

14 Láthatósági algoritmusok

14.1 Hátsó lap eldobás (Backface Culling)

Zárt testeknél a hátsó lapok eldobhatók.



$$\cos\theta = \frac{N \cdot V}{||N|| \cdot ||V||}$$

14.2 Z-buffer (Mélységi puffer)

A perspektivikus vetítés során a z koordináták nem lineárisan képeződnek le. A Z-bufferben tárolt érték (z') és a valós távolság (z_{eye}) közötti kapcsolat:

$$z' = \frac{a}{z_{eye}} + b$$

15 Fény- és anyagtulajdonságok

(A korábbi leírás változatlan: Ambient, Diffuse, Specular komponensek).

16 Megvilágítási és árnyalási modellek - Matematikai Mélységek

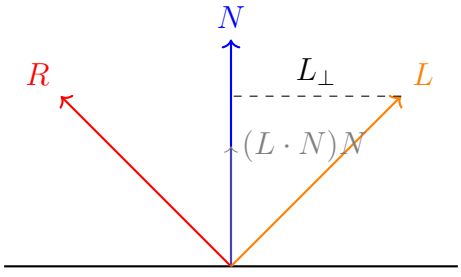
16.1 Lambert-féle koszinusz törvény (Diffuse)

A matt felületek fényvisszaverése a beesési szög koszinuszával arányos.

$$I_d = I_{light} \cdot k_d \cdot (L \cdot N)$$

16.2 Reflexiós vektor levezetése (Specular)

A Phong-modellhez szükség van a tökéletes tükrözési irányra (R).



Képlet: $R = 2(N \cdot L)N - L$

16.3 Phong-modell kitevője (n)

A $(R \cdot V)^n$ tagban az n a felület simaságát (shininess) jellemzi.

16.4 Árnyalási Interpoláció (Gouraud vs Phong)

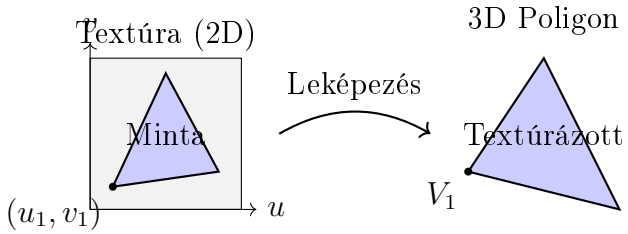
- **Gouraud:** A színeket (C) interpoláljuk.

$$C_P = \alpha C_A + \beta C_B + \gamma C_C$$

- **Phong:** A normálvektorokat (N) interpoláljuk, majd minden pixelben újra normalizáljuk.

17 Textúrázás

A textúrázás során egy 2D-s képet (textúrát) feszítünk rá a 3D-s modellre.



17.1 Perspektivikus korrekció

Lineáris interpoláció a képernyő síkjában nem helyes. Helyes eljárás: $u/w, v/w$ és $1/w$ interpolálása, majd osztás.