

Számítógépes Grafika

Valasek Gábor

valasek@inf.elte.hu

Eötvös Loránd Tudományegyetem
Informatikai Kar

2011/2012. őszi félév

Tartalom

1 Textúrázás

- Bevezetés
- Textúra leképezés
- Paraméterezés
- Textúra szűrés
- Procedurális textúrák
- Nem-szín textúrák

2 Inkrementális képszintézis a GPU-n

- Inkrementális képszintézis
- DirectX
- OpenGL
- Grafikus gyorsítókártya generációk

Tartalom

1 Textúrázás

- Bevezetés
 - Textúra leképezés
 - Paraméterezés
 - Textúra szűrés
 - Procedurális textúrák
 - Nem-szín textúrák

2 Inkrementális képszintézis a GPU-n

- Inkrementális képszintézis
 - DirectX
 - OpenGL
 - Grafikus gyorsítókártya generációk

Bevezetés

Mi az a textúrázás?

- Eddig: egyszínű anyag modellek, azaz a teljes felület azonos színű.

Bevezetés

Mi az a textúrázás?

- Eddig: egyszínű anyag modellek, azaz a teljes felület azonos színű.
 - Kevés ilyen van a valóságban.

Bevezetés

Mi az a textúrázás?

- Eddig: egyszínű anyag modellek, azaz a teljes felület azonos színű.
 - Kevés ilyen van a valóságban.
 - Finom részleteket szeretnénk megadni.

Bevezetés

Mi az a textúrázás?

- Eddig: egyszínű anyag modellek, azaz a teljes felület azonos színű.
 - Kevés ilyen van a valóságban.
 - Finom részleteket szeretnénk megadni.
 - Változó paraméterekre van szükségünk a BRDF-ben.

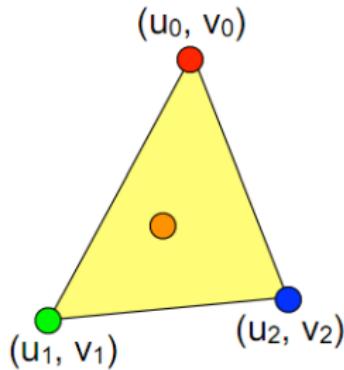
Bevezetés

Mi az a textúrázás?

- Eddig: egyszínű anyag modellek, azaz a teljes felület azonos színű.
 - Kevés ilyen van a valóságban.
 - Finom részleteket szeretnénk megadni.
 - Változó paraméterekre van szükségünk a BRDF-ben.
 - Ezeket a paramétereket, elsősorban színt, adjuk meg a *textúrákban*.

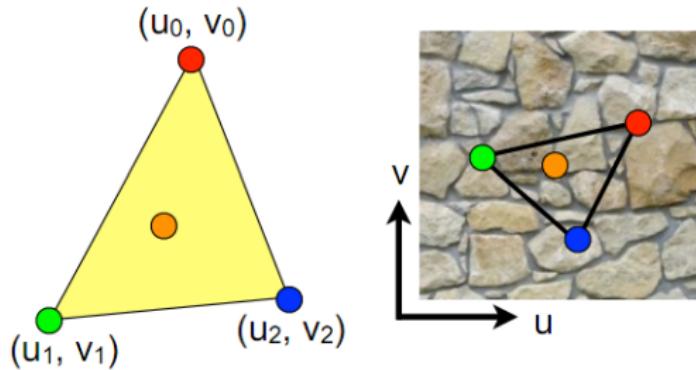
Bevezetés

Textúrázás



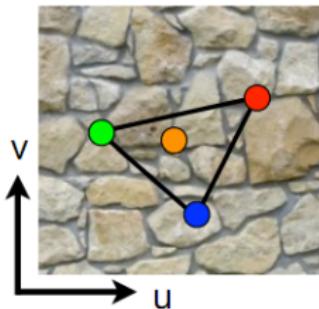
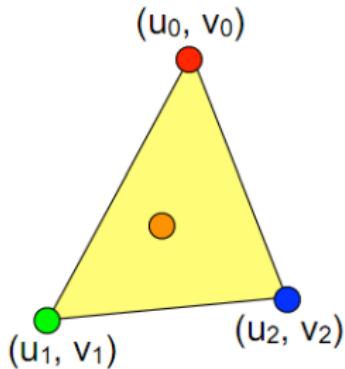
Bevezetés

Textúrázás



Bevezetés

Textúrázás



Bevezetés

Textúra megadási módok

- "Tömbbel":

Bevezetés

Textúra megadási módok

- "Tömbbel":
 - valamelyen 1/2/3 dimenziós tömbből olvasunk, elemei a *texel*-ek

Textúra megadási módok

- "Tömbbel":
 - valamelyen 1/2/3 dimenziós tömbből olvasunk, elemei a *texel*-ek
 - 2D szemléletesen: a geometriánkat a tömbben tárolt "képpel" "fedjük/csomagoljuk be"

Textúra megadási módok

- "Tömbbel":
 - valamelyen 1/2/3 dimenziós tömbből olvasunk, elemei a *texel*-ek
 - 2D szemléletesen: a geometriánkat a tömbben tárolt "képpel" "fedjük/csomagoljuk be"
 - *textúratérben* vett koordinátákkal azonosítjuk a texeleket, a koordináták [0, 1] intervallumon vannak értelmezve

Bevezetés

Textúra megadási módok

- "Tömbbel":
 - valamelyen 1/2/3 dimenziós tömbből olvasunk, elemei a *texel*-ek
 - 2D szemléletesen: a geometriánkat a tömbben tárolt "képpel" "fedjük/csomagoljuk be"
 - *textúratérben* vett koordinátákkal azonosítjuk a texeleket, a koordináták [0, 1] intervallumon vannak értelmezve
 - Függvénynel adjuk meg:

Bevezetés

Textúra megadási módok

- "Tömbbel":
 - valamelyen 1/2/3 dimenziós tömbből olvasunk, elemei a *texel*-ek
 - 2D szemléletesen: a geometriánkat a tömbben tárolt "képpel" "fedjük/csomagoljuk be"
 - *textúratérben* vett koordinátkkal azonosítjuk a texeleket, a koordináták $[0, 1]$ intervallumon vannak értelmezve
 - Függvénynel adjuk meg:
 - valamelyen $f(x, y, z) \rightarrow$ szín függvény segítségével

Bevezetés

Textúra megadási módok

- "Tömbbel":
 - valamelyen 1/2/3 dimenziós tömbből olvasunk, elemei a *texel*-ek
 - 2D szemléletesen: a geometriánkat a tömbben tárolt "képpel" "fedjük/csomagoljuk be"
 - *textúratérben* vett koordinátákkal azonosítjuk a texeleket, a koordináták $[0, 1]$ intervallumon vannak értelmezve
 - Függvénynel adjuk meg:
 - valamelyen $f(x, y, z) \rightarrow$ szín függvény segítségével
 - ezt nevezzük *procedurális textúrázásnak*

Textúra leképezés

Tartalom

1 Textúrázás

- Bevezetés
 - **Textúra leképezés**
 - Paraméterezés
 - Textúra szűrés
 - Procedurális textúrák
 - Nem-szín textúrák

2 Inkrementális képszintézis a GPU-n

- Inkrementális képszintézis
 - DirectX
 - OpenGL
 - Grafikus gyorsítókártya generációk

Textúra leképezés

Leképezési módok

- Két terünk van: *képtér* (képernyő pixelei) és *textúratér* (textúra texelei)

Textúra leképezés

Leképezési módok

- Két terünk van: *képtér* (képernyő pixelei) és *textúratér* (textúra texelei)
- Honnan, hova képezzünk?

Textúra leképezés

Leképezési módok

- Két terünk van: képtér (képernyő pixelei) és *textúratér* (textúra texelei)
- Honnan, hova képezzünk?
- Textúratér → képtér: *textúra alapú leképezés*

Textúra leképezés

Leképezési módok

- Két terünk van: *képtér* (képernyő pixelei) és *textúratér* (textúra texelei)
- Honnan, hova képezzünk?
- Textúratér → képtér: *textúra alapú leképezés*
 - » minden *texel*hez keressünk a neki megfelelő *pixelt*.

Textúra leképezés

Leképezési módok

- Két terünk van: *képtér* (képernyő pixelei) és *textúratér* (textúra texelei)
- Honnan, hova képezzünk?
- Textúratér → képtér: *textúra alapú leképezés*
 - » minden *texel*hez keressünk a neki megfelelő *pixelt*.
 - ⊕ Hatékony.

Textúra leképezés

Leképezési módok

- Két terünk van: *képtér* (képernyő pixelei) és *textúratér* (textúra texelei)
- Honnan, hova képezzünk?
- Textúratér → képtér: *textúra alapú leképezés*
 - » minden *texel*hez keressünk a neki megfelelő *pixelt*.
 - ⊕ Hatékony.
 - ⊖ Nem biztos, hogy minden pixel sorra kerül, és ami mégis, nem biztos, hogy csak egyszer.

Textúra leképezés

Leképezési módok

- Két terünk van: *képtér* (képernyő pixelei) és *textúratér* (textúra texelei)
- Honnan, hova képezzünk?
- Textúratér → képtér: *textúra alapú leképezés*
 - » minden *texel*hez keressünk a neki megfelelő *pixelt*.
 - ⊕ Hatékony.
 - ⊖ Nem biztos, hogy minden pixel sorra kerül, és ami mégis, nem biztos, hogy csak egyszer.
- Képtér → textúratér: *képtér alapú leképezés*

Textúra leképezés

Leképezési módok

- Két terünk van: *képtér* (képernyő pixelei) és *textúratér* (textúra texelei)
- Honnan, hova képezzünk?
- Textúratér → képtér: *textúra alapú leképezés*
 - » minden *texel*hez keressünk a neki megfelelő *pixelt*.
 - ⊕ Hatékony.
 - ⊖ Nem biztos, hogy minden pixel sorra kerül, és ami mégis, nem biztos, hogy csak egyszer.
- Képtér → textúratér: *képtér alapú leképezés*
 - » minden *pixel*hez keresünk a neki megfelelő *texelt*.

Textúra leképezés

Leképezési módok

- Két terünk van: *képtér* (képernyő pixelei) és *textúratér* (textúra texelei)
- Honnan, hova képezzünk?
- Textúratér → képtér: *textúra alapú leképezés*
 - » minden *texel*hez keressünk a neki megfelelő *pixelt*.
 - ⊕ Hatékony.
 - ⊖ Nem biztos, hogy minden pixel sorra kerül, és ami mégis, nem biztos, hogy csak egyszer.
- Képtér → textúratér: *képtér alapú leképezés*
 - » minden *pixel*hez keresünk a neki megfelelő *texelt*.
 - ⊕ Inkrementális képszintézishez jól passzol.

Textúra leképezés

Leképezési módok

- Két terünk van: *képtér* (képernyő pixelei) és *textúratér* (textúra texelei)
- Honnan, hova képezzünk?
- Textúratér → képtér: *textúra alapú leképezés*
 - » minden *texel*hez keressünk a neki megfelelő *pixelt*.
 - ⊕ Hatékony.
 - ⊖ Nem biztos, hogy minden pixel sorra kerül, és ami mégis, nem biztos, hogy csak egyszer.
- Képtér → textúratér: *képtér alapú leképezés*
 - » minden *pixel*hez keresünk a neki megfelelő *texelt*.
 - ⊕ Inkrementális képszintézishez jól passzol.
 - ⊖ Szükség van hozzá a paraméterezési és vetítési transzformációk inverzére.

Paraméterezés

Tartalom

1 Textúrázás

- Bevezetés
- Textúra leképezés
- Paraméterezés**
- Textúra szűrés
- Procedurális textúrák
- Nem-szín textúrák

2 Inkrementális képszintézis a GPU-n

- Inkrementális képszintézis
- DirectX
- OpenGL
- Grafikus gyorsítókártya generációk

Paraméterezés

Paraméterezés

- Hogyan tudjuk eldönten, hogy az egyes felületi pontokhoz a tömb melyik elemét választjuk, vagy a procedurális textúra függvényét milyen paraméterekkel értékeljük ki?

Paraméterezés

Paraméterezés

- Hogyan tudjuk eldönten, hogy az egyes felületi pontokhoz a tömb melyik elemét választjuk, vagy a procedurális textúra függvényét milyen paraméterekkel értékeljük ki?
- A felület minden pontjához *textúra koordinátákat* rendelünk.

Paraméterezés

Paraméterezés

- Hogyan tudjuk eldönten, hogy az egyes felületi pontokhoz a tömb melyik elemét választjuk, vagy a procedurális textúra függvényét milyen paraméterekkel értékeljük ki?
- A felület minden pontjához *textúra koordinátákat* rendelünk.
- A textúra koordináták hozzárendelését a felülethez nevezzük most *paraméterezésnek*.

Paraméterezés

Paraméterezés

- Hogyan tudjuk eldönten, hogy az egyes felületi pontokhoz a tömb melyik elemét választjuk, vagy a procedurális textúra függvényét milyen paraméterekkel értékeljük ki?
- A felület minden pontjához *textúra koordinátákat* rendelünk.
- A textúra koordináták hozzárendelését a felülethez nevezzük most *paraméterezésnek*.
- A továbbiakban 2D textúrákról beszélünk.

Paraméterezés

Parametrikus felületek paraméterezése

- Parametrikus felület:

$$F \in \mathbb{R}^2 \rightarrow \mathbb{R}^3, \quad F(u, v) := (x, y, z)$$

Parametrikus felületek paraméterezése

- Parametrikus felület:

$$F \in \mathbb{R}^2 \rightarrow \mathbb{R}^3, \quad F(u, v) := (x, y, z)$$

- u, v paraméterek természetes módon használhatóak textúra koordinátáknak.

Parametrikus felületek paraméterezése

- Parametrikus felület:

$$F \in \mathbb{R}^2 \rightarrow \mathbb{R}^3, \quad F(u, v) := (x, y, z)$$

- u, v paraméterek természetes módon használhatóak textúra koordinátáknak.
- Ha $\mathcal{D}_F \neq \mathcal{D}_{tex}$, akkor u, v -t transzformálni kell.

Parametrikus felületek paraméterezése

- Parametrikus felület:

$$F \in \mathbb{R}^2 \rightarrow \mathbb{R}^3, \quad F(u, v) := (x, y, z)$$

- u, v paraméterek természetes módon használhatóak textúra koordinátáknak.
- Ha $\mathcal{D}_F \neq \mathcal{D}_{tex}$, akkor u, v -t transzformálni kell.
- Pi:
 - Henger palást
 - $\mathcal{D}_F = [0, 2\pi] \times [0, h]$, $F(u, v) := (\cos u, h, \sin u)$
 - Textúra tér: $(\bar{u}, \bar{v}) \in [0, 1] \times [0, 1]$
 - Transzformáció: $\bar{u} := u/2\pi$, $\bar{v} := v/h$

Háromszögek paraméterezése

- Legyen adott a háromszög három csúcsa:

$p_i = (x_i, y_i, z_i) \in \mathbb{R}^3$, $i \in \{1, 2, 3\}$, valamint az ezeknek megfelelő csúcsok textúra térben:

$$t_i = (u_i, v_i) \in \mathbb{R}^2, i \in \{1, 2, 3\}.$$

Háromszögek paraméterezése

- Legyen adott a háromszög három csúcsa:
 $p_i = (x_i, y_i, z_i) \in \mathbb{R}^3$, $i \in \{1, 2, 3\}$, valamint az ezeknek megfelelő csúcsok textúra térben:
 $t_i = (u_i, v_i) \in \mathbb{R}^2$, $i \in \{1, 2, 3\}$.
- Olyan $\mathbb{R}^3 \rightarrow \mathbb{R}^2$ leképezést keresünk, amire $p_i \mapsto t_i$, és háromszöget háromszögbe visz át.

Paraméterezés

Háromszögek paraméterezése

- Legyen adott a háromszög három csúcsa:
 $p_i = (x_i, y_i, z_i) \in \mathbb{R}^3$, $i \in \{1, 2, 3\}$, valamint az ezeknek megfelelő csúcsok textúra térben:
 $t_i = (u_i, v_i) \in \mathbb{R}^2$, $i \in \{1, 2, 3\}$.
- Olyan $\mathbb{R}^3 \rightarrow \mathbb{R}^2$ leképezést keresünk, amire $p_i \mapsto t_i$, és háromszöget háromszögbe visz át.
- A legegyszerűbb ilyen leképezés a lineáris leképezés, ami megadható egy 3×3 -as mátrixszal.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} A_x & A_y & A_z \\ B_x & B_y & B_z \\ C_x & C_y & C_z \end{bmatrix} \cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{P} \cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

Paraméterezés

Háromszögek paraméterezése

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} A_x & A_y & A_z \\ B_x & B_y & B_z \\ C_x & C_y & C_z \end{bmatrix} \cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{P} \cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

- Kilenc ismeretlen, kilenc egyenlet

Paraméterezés

Háromszögek paraméterezése

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} A_x & A_y & A_z \\ B_x & B_y & B_z \\ C_x & C_y & C_z \end{bmatrix} \cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{P} \cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

- Kilenc ismeretlen, kilenc egyenlet
- Ez textúra alapú leképezés!

Paraméterezés

Háromszögek paraméterezése

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} A_x & A_y & A_z \\ B_x & B_y & B_z \\ C_x & C_y & C_z \end{bmatrix} \cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{P} \cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

- Kilenc ismeretlen, kilenc egyenlet
- Ez textúra alapú leképezés!
- Képtér alapú leképezéshez inverz trafó kell:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{P}^{-1} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Paraméterezés

Textúrázás és sugárkövetés

- Felület-sugár metszés: világ-koordináta rendszerben

Paraméterezés

Textúrázás és sugárkövetés

- Felület-sugár metszés: világ-koordináta rendszerben
- Inverz transzformációk:

Paraméterezés

Textúrázás és sugárkövetés

- Felület-sugár metszés: világ-koordináta rendszerben
- Inverz transzformációk:
 - Világ KR → Model KR

Paraméterezés

Textúrázás és sugárkövetés

- Felület-sugár metszés: világ-koordináta rendszerben
- Inverz transzformációk:
 - Világ KR → Model KR
 - Model KR → textúra tér

Paraméterezés

Model KR → textúra tér

- Parametrikus felületek

A metszéspont számítás során adódik u, v , nem kell külön számítani.

Paraméterezés

Model KR → textúra tér

- Parametrikus felületek

A metszéspont számítás során adódik u, v , nem kell külön számítani.

- Háromszögek

Az előbb levezetett \mathbf{P}^{-1} szükséges.

Gyorsítási lehetőség: ha sem maga a háromszög, sem a textúra koordináták nem változnak a csúcsokban, akkor \mathbf{P}^{-1} állandó.

Paraméterezés

Háromszögek paraméterezése

- Gyakorlatban ez gyorsan számítható inkrementális algoritmussal.

Paraméterezés

Háromszögek paraméterezése

- Gyakorlatban ez gyorsan számítható inkrementális algoritmussal.
- Ha a három csúcsban adottak a textúra koordináták, akkor ezeket a háromszögek kitöltésénél használt algoritmussal kiszámíthatjuk minden pixelre.

Paraméterezés

Háromszögek paraméterezése

- Gyakorlatban ez gyorsan számítható inkrementális algoritmussal.
- Ha a három csúcsban adottak a textúra koordináták, akkor ezeket a háromszögek kitöltésénél használt algoritmussal kiszámíthatjuk minden pixelre.
- minden pontra legyen a felület egy pontja
 $\mathbf{p} = \alpha\mathbf{p}_1 + \beta\mathbf{p}_2 + \gamma\mathbf{p}_3$, az α, β, γ baricentrikus koordinátákkal adott.

Paraméterezés

Háromszögek paraméterezése

- Gyakorlatban ez gyorsan számítható inkrementális algoritmussal.
- Ha a három csúcsban adottak a textúra koordináták, akkor ezeket a háromszögek kitöltésénél használt algoritmussal kiszámíthatjuk minden pixelre.
- minden pontra legyen a felület egy pontja
 $\mathbf{p} = \alpha\mathbf{p}_1 + \beta\mathbf{p}_2 + \gamma\mathbf{p}_3$, az α, β, γ baricentrikus koordinákkal adott.
- Ekkor a \mathbf{p} -hez tartozó textúra koordináta is megkapható
 $t = \alpha t_1 + \beta t_2 + \gamma t_3$ alapján

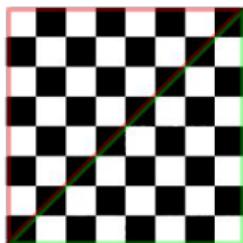
Paraméterezés

Perspektívikusan korrekt textúrázás

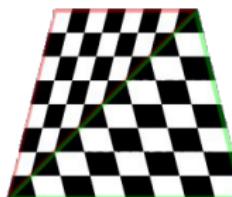
- A textúra koordináták lineáris interpolációja hibás képet ad, ha a megjelenítendő háromszögön nem csak affin transzformációt végzünk.

Perspektívikusan korrekt textúrázás

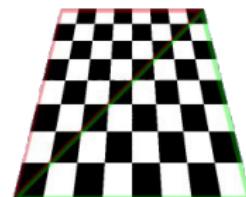
- A textúra koordináták lineáris interpolációja hibás képet ad, ha a megjelenítendő háromszögön nem csak affin transzformációt végzünk.
- Értsd: esetek 99%-a.



Sík



Affin



Helyes

Paraméterezés

Perspektívikusan korrekt textúrázás

- Tehát u, v lineáris interpolációja nem lesz jó nem affin transzformációk esetén - nem lineárisak a képernyőtérben

Perspektívikusan korrekt textúrázás

- Tehát u, v lineáris interpolációja nem lesz jó nem affin transzformációk esetén - nem lineárisak a képernyőtérben
- Transzformálva, homogén osztás előtt a koordinátáink legyenek $[x_t, y_t, z_t, w_t]$

Perspektívikusan korrekt textúrázás

- Tehát u, v lineáris interpolációja nem lesz jó nem affin transzformációk esetén - nem lineárisak a képernyőtérben
- Transzformálva, homogén osztás előtt a koordinátáink legyenek $[x_t, y_t, z_t, w_t]$
- Homogén osztás után: $[x_s, y_s, z_s, 1] = [\frac{x_t}{w_t}, \frac{y_t}{w_t}, \frac{z_t}{w_t}, 1]$

Perspektívikusan korrekt textúrázás

- Tehát u, v lineáris interpolációja nem lesz jó nem affin transzformációk esetén - nem lineárisak a képernyőtérben
- Transzformálva, homogén osztás előtt a koordinátáink legyenek $[x_t, y_t, z_t, w_t]$
- Homogén osztás után: $[x_s, y_s, z_s, 1] = [\frac{x_t}{w_t}, \frac{y_t}{w_t}, \frac{z_t}{w_t}, 1]$
- Ha $[x_s, y_s, z_s, 1]$ szerint interpoláljuk a textúra koordinátákat, akkor az nem lesz jó.

Perspektívikusan korrekt textúrázás

- Tehát u, v lineáris interpolációja nem lesz jó nem affin transzformációk esetén - nem lineárisak a képernyőtérben
- Transzformálva, homogén osztás előtt a koordinátáink legyenek $[x_t, y_t, z_t, w_t]$
- Homogén osztás után: $[x_s, y_s, z_s, 1] = [\frac{x_t}{w_t}, \frac{y_t}{w_t}, \frac{z_t}{w_t}, 1]$
- Ha $[x_s, y_s, z_s, 1]$ szerint interpoláljuk a textúra koordinátákat, akkor az nem lesz jó.
- Ezzel szemben: ha $\frac{1}{w}$ -t interpoláljuk, az helyes marad! Sőt, bármilyen $\frac{q}{w}$ érték jól interpolálódik!

Perspektívikusan korrekt textúrázás

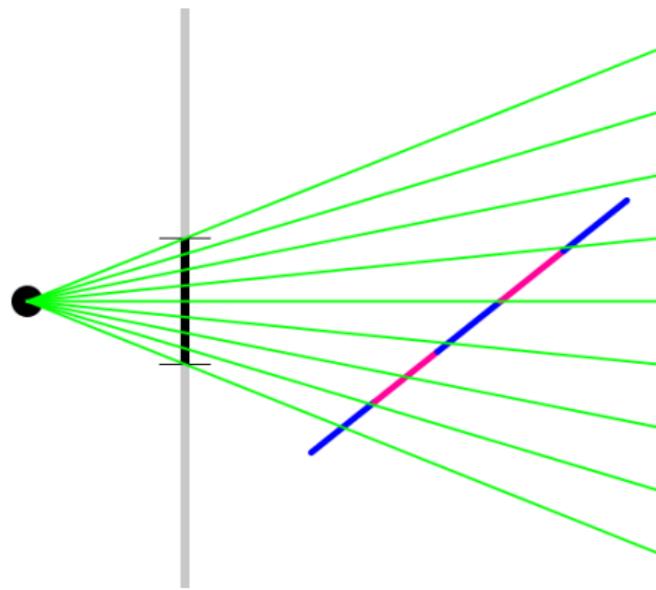
- Tehát u, v lineáris interpolációja nem lesz jó nem affin transzformációk esetén - nem lineárisak a képernyőtérben
- Transzformálva, homogén osztás előtt a koordinátáink legyenek $[x_t, y_t, z_t, w_t]$
- Homogén osztás után: $[x_s, y_s, z_s, 1] = [\frac{x_t}{w_t}, \frac{y_t}{w_t}, \frac{z_t}{w_t}, 1]$
- Ha $[x_s, y_s, z_s, 1]$ szerint interpoláljuk a textúra koordinátákat, akkor az nem lesz jó.
- Ezzel szemben: ha $\frac{1}{w}$ -t interpoláljuk, az helyes marad! Sőt, bármilyen $\frac{q}{w}$ érték jól interpolálódik!
- α, β, γ helyett használjuk a világ koordináta-rendszerbeli $\alpha_w, \beta_w, \gamma_w$ koordinátákat, és

Perspektívikusan korrekt textúrázás

- Tehát u, v lineáris interpolációja nem lesz jó nem affin transzformációk esetén - nem lineárisak a képernyőtérben
- Transzformálva, homogén osztás előtt a koordinátáink legyenek $[x_t, y_t, z_t, w_t]$
- Homogén osztás után: $[x_s, y_s, z_s, 1] = [\frac{x_t}{w_t}, \frac{y_t}{w_t}, \frac{z_t}{w_t}, 1]$
- Ha $[x_s, y_s, z_s, 1]$ szerint interpoláljuk a textúra koordinátákat, akkor az nem lesz jó.
- Ezzel szemben: ha $\frac{1}{w}$ -t interpoláljuk, az helyes marad! Sőt, bármilyen $\frac{q}{w}$ érték jól interpolálódik!
- α, β, γ helyett használjuk a világ koordináta-rendszerbeli $\alpha_w, \beta_w, \gamma_w$ koordinátákat, és
- $\frac{\alpha_w}{w}, \frac{\beta_w}{w}, \frac{\gamma_w}{w}$ interpolálva helyes textúrázást kapunk.

Paraméterezés

Perspektívikusan korrekt textúrázás



Textúra szűrés

Tartalom

1 Textúrázás

- Bevezetés
- Textúra leképezés
- Paraméterezés
- Textúra szűrés**
- Procedurális textúrák
- Nem-szín textúrák

2 Inkrementális képszintézis a GPU-n

- Inkrementális képszintézis
- DirectX
- OpenGL
- Grafikus gyorsítókártya generációk

Textúra szürés

Textúrák szűrése

- Ritkán esik pontosan egy texel egy pixelre.

Textúra szürés

Textúrák szűrése

- Ritkán esik pontosan egy texel egy pixelre.
- Nagyítás: a pixel mérete kisebb a texelénél – több pixel jut egyetlen texelre. OpenGL: GL_TEXTURE_MAG_FILTER

Textúra szűrés

Textúrák szűrése

- Ritkán esik pontosan egy texel egy pixelre.
- Nagyítás: a pixel mérete kisebb a texelénél – több pixel jut egyetlen texelre. OpenGL: GL_TEXTURE_MAG_FILTER
- Kicsinyítés: a pixel mérete nagyobb a texelénél – több texel jut egyetlen pixelre. OpenGL: GL_TEXTURE_MIN_FILTER

Textúra szürés

Textúrák szűrése

- Ritkán esik pontosan egy texel egy pixelre.
- Nagyítás: a pixel mérete kisebb a texelénél – több pixel jut egyetlen texelre. OpenGL: GL_TEXTURE_MAG_FILTER
- Kicsinyítés: a pixel mérete nagyobb a texelénél – több texel jut egyetlen pixelre. OpenGL: GL_TEXTURE_MIN_FILTER
- További probléma: ami a textúra térben lineáris, az nem az a képtérben a perspektíva miatt.

Textúra szűrés

Nagyítás

Egy pixel mérete kisebb egy texelénél – több pixel jut egyetlen texelre

- Szűrés nélkül: a pixel középpontjához legközelebb eső texel értékét használjuk.

Textúra szürés

Nagyítás

Egy pixel mérete kisebb egy texelénél – több pixel jut egyetlen texelre

- Szűrés nélkül: a pixel középpontjához legközelebb eső texel értékét használjuk.
- Bilineáris szűrés: a legközelebbi négy texel súlyozott átlagát vesszük.

Textúra szürés

Kicsinyítés

Egy pixel mérete nagyobb egy texelénél – több texel jut egyetlen pixelre

- Pontos mintavételezés lenne: a pixel négyzetét transzformáljuk el textúra térbe, és az ott kijelölt texelek átlagát vegyük.

Textúra szűrés

Kicsinyítés

Egy pixel mérete nagyobb egy texelénél – több texel jut egyetlen pixelre

- Pontos mintavételezés lenne: a pixel négyzetét transzformáljuk el textúra térbe, és az ott kijelölt texelek átlagát vegyük.
- Helyette: a textúra térben is négyzetet veszünk.

Textúra szűrés

Kicsinyítés

Egy pixel mérete nagyobb egy texelénél – több texel jut egyetlen pixelre

- Pontos mintavételezés lenne: a pixel négyzetét transzformáljuk el textúra térbe, és az ott kijelölt texelek átlagát vegyük.
- Helyette: a textúra térben is négyzetet veszünk.
- Gyakorlatban: a textúra egy tetszőleges négyzetének a leátlagolása túl erőforrás igényes, helyette vagy használunk kevesebb texelt, vagy *MIP-map*-eket

Textúra szürés

Kicsinyítés

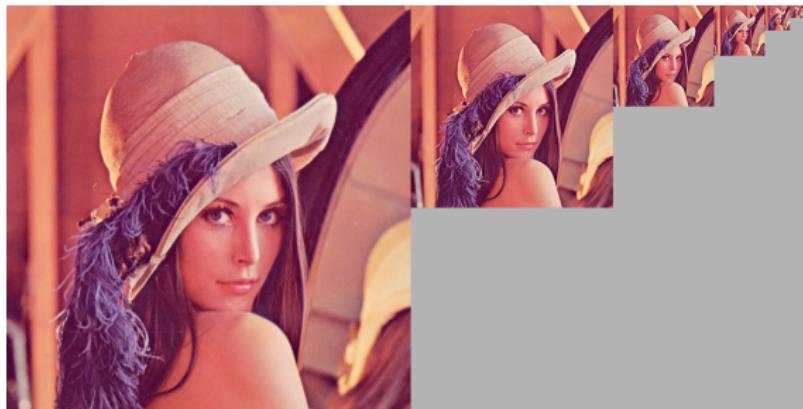
Egy pixel mérete nagyobb egy texelénél – több texel jut egyetlen pixelre

- Pontos mintavételezés lenne: a pixel négyzetét transzformáljuk el textúra térbe, és az ott kijelölt texelek átlagát vegyük.
- Helyette: a textúra térben is négyzetet veszünk.
- Gyakorlatban: a textúra egy tetszőleges négyzetének a leátlagolása túl erőforrás igényes, helyette vagy használunk kevesebb texelt, vagy *MIP-map*-eket
- Kevesebb pixel:
 - Szűrés nélkül: a pixel középpontjához legközelebb eső texel értékét használjuk.
 - Bilineáris szűrés: a legközelebbi négy texel súlyozott átlagát vesszük.

Textúra szürés

MIP-map-ek

- MIP: *multum in parvo* – sok, kis helyen



Textúra szürés

MIP-map-ek

- MIP: *multum in parvo* – sok, kis helyen
- Ú.n. pirasmist generálunk a textúrából ből, minden szinten felezve a méretét



Textúra szürés

MIP-map-ek

- Szűrés során kiválasztjuk a pixel/texel terület aránynak megfelelő szintet és onnan olvasunk.

Textúra szürés

MIP-map-ek

- Szűrés során kiválasztjuk a pixel/texel terület aránynak megfelelő szintet és onnan olvasunk.
- Az adott MIP-map-en belül is lehet az olvasás szűrés nélküli, vagy lehet bilienárisan szűrt.

Textúra szürés

MIP-map-ek

- Szűrés során kiválasztjuk a pixel/texel terület aránynak megfelelő szintet és onnan olvasunk.
- Az adott MIP-map-en belül is lehet az olvasás szűrés nélküli, vagy lehet bilienárisan szűrt.
- *Trilineáris szűrés*: két szomszédos szintet használunk, azokon belül bilineáris szűréssel, és ezeknek vesszük a súlyoztott átlagát.

Procedurális textúrák

Tartalom

1 Textúrázás

- Bevezetés
- Textúra leképezés
- Paraméterezés
- Textúra szűrés
- Procedurális textúrák
- Nem-szín textúrák

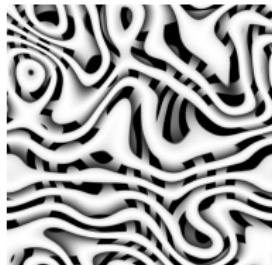
2 Inkrementális képszintézis a GPU-n

- Inkrementális képszintézis
- DirectX
- OpenGL
- Grafikus gyorsítókártya generációk

Procedurális textúrák

Procedurális textúrák

- A textúrákat "tömb" helyett megadhatjuk függvényel is.

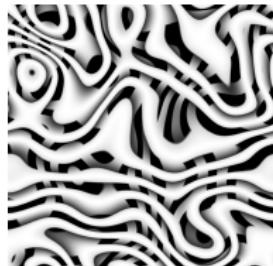
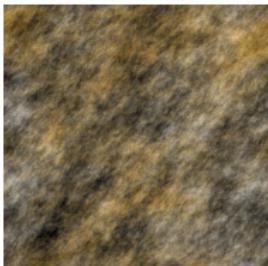


generated with Filter Forge

Procedurális textúrák

Procedurális textúrák

- A textúrákat "tömb" helyett megadhatjuk függvényteljesen is.
- Textúra koordináták → a függvény paramétereit.



generated with Filter Forge

Procedurális textúrák

Tulajdonságai

- Előnyök:

Procedurális textúrák

Tulajdonságai

- Előnyök:
 - Sokkal kevesebb tárhely

Procedurális textúrák

Tulajdonságai

- Előnyök:

- Sokkal kevesebb tárhely
- Tetszőleges felbontás – csak a numerikus pontosság a korlát

Procedurális textúrák

Tulajdonságai

- Előnyök:

- Sokkal kevesebb tárhely
- Tetszőleges felbontás – csak a numerikus pontosság a korlát
⇒ Nagyításnál nem kell szűrés. (Kicsinyítést sajnos nem oldja meg.)

Procedurális textúrák

Tulajdonságai

- Előnyök:
 - Sokkal kevesebb tárhely
 - Tetszőleges felbontás – csak a numerikus pontosság a korlát
 - ⇒ Nagyításnál nem kell szűrés. (Kicsinyítést sajnos nem oldja meg.)
- Hátrányok:

Procedurális textúrák

Tulajdonságai

- Előnyök:
 - Sokkal kevesebb tárhely
 - Tetszőleges felbontás – csak a numerikus pontosság a korlát
 - ⇒ Nagyításnál nem kell szűrés. (Kicsinyítést sajnos nem oldja meg.)
- Hátrányok:
 - Nagy számítás igény.

Procedurális textúrák

Tulajdonságai

- Előnyök:
 - Sokkal kevesebb tárhely
 - Tetszőleges felbontás – csak a numerikus pontosság a korlát
 - ⇒ Nagyításnál nem kell szűrés. (Kicsinyítést sajnos nem oldja meg.)
- Hátrányok:
 - Nagy számítás igény.
 - Nem lehet vele bármit leírni.

Procedurális textúrák

Tulajdonságai

- Előnyök:
 - Sokkal kevesebb tárhely
 - Tetszőleges felbontás – csak a numerikus pontosság a korlát
 - ⇒ Nagyításnál nem kell szűrés. (Kicsinyítést sajnos nem oldja meg.)
- Hátrányok:
 - Nagy számítás igény.
 - Nem lehet vele bármit leírni.
 - Nem módosítható tetszőlegesen.

Nem-szín textúrák

Tartalom

1 Textúrázás

- Bevezetés
- Textúra leképezés
- Paraméterezés
- Textúra szűrés
- Procedurális textúrák
- Nem-szín textúrák

2 Inkrementális képszintézis a GPU-n

- Inkrementális képszintézis
- DirectX
- OpenGL
- Grafikus gyorsítókártya generációk

Nem-szín textúrák

Nem-szín textúrák

- A textúrák segítségével a felületi pont bármilyen tulajdonságát leírhatjuk.

Nem-szín textúrák

Nem-szín textúrák

- A textúrák segítségével a felületi pont bármilyen tulajdonságát leírhatjuk.
- Ezek a tulajdonságok lehetnek pl.:
 - felületi normális – *Bucka ill. normál leképezés*

Nem-szín textúrák

Nem-szín textúrák

- A textúrák segítségével a felületi pont bármilyen tulajdonságát leírhatjuk.
- Ezek a tulajdonságok lehetnek pl.:
 - felületi normális – *Bucka ill. normál leképezés*
 - elmozdulás – *Displacement leképezés*

Nem-szín textúrák

Nem-szín textúrák

- A textúrák segítségével a felületi pont bármilyen tulajdonságát leírhatjuk.
- Ezek a tulajdonságok lehetnek pl.:
 - felületi normális – *Bucka ill. normál leképezés*
 - elmozdulás – *Displacement leképezés*
 - fényforrás láthatósága – *Árnyék térképek*

Nem-szín textúrák

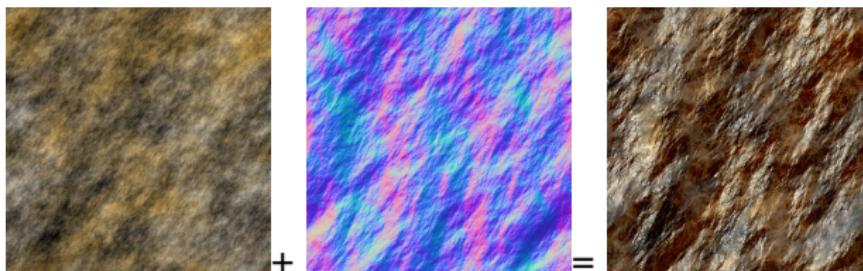
Nem-szín textúrák

- A textúrák segítségével a felületi pont bármilyen tulajdonságát leírhatjuk.
- Ezek a tulajdonságok lehetnek pl.:
 - felületi normális – *Bucka ill. normál leképezés*
 - elmozdulás – *Displacement leképezés*
 - fényforrás láthatósága – *Árnyék térképek*
 - tükröként visszavert fény – *Visszaverődés leképezés ill. Környezet leképezés*

Nem-szín textúrák

Bucka vagy normál leképezés

- A textúrával normál vektorokat adunk meg.

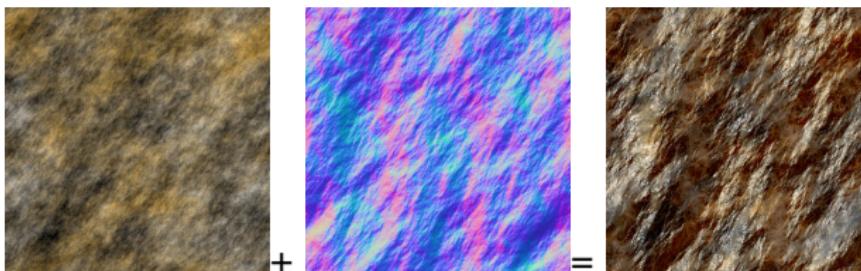


generated with Filter Forge

Nem-szín textúrák

Bucka vagy normál leképezés

- A textúrával normál vektorokat adunk meg.
- A felület eredeti normálisai helyett ezeket használjuk a megvilágítás számításakor.

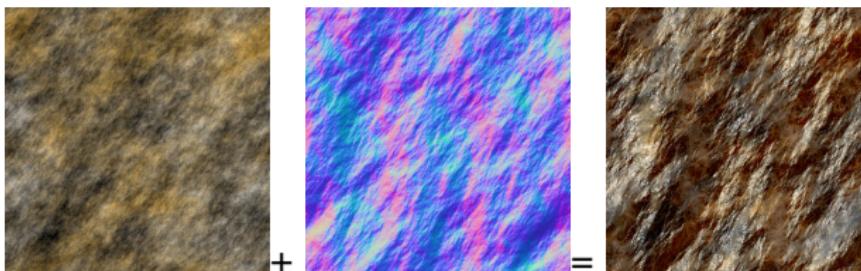


generated with Filter Forge

Nem-szín textúrák

Bucka vagy normál leképezés

- A textúrával normál vektorokat adunk meg.
- A felület eredeti normálisai helyett ezeket használjuk a megvilágítás számításakor.
- Rücskös/érdes felület látszatát adja, amíg nem nézünk rá túl lapos szögben.

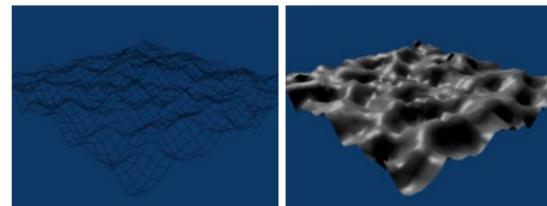
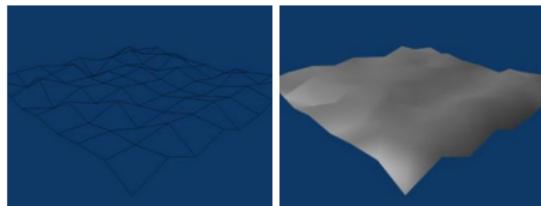


generated with Filter Forge

Nem-szín textúrák

Displacement leképezés

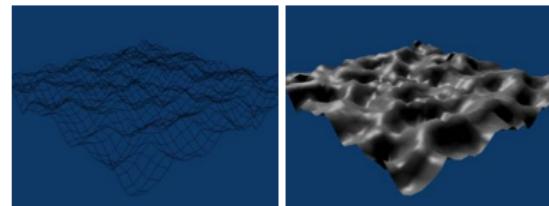
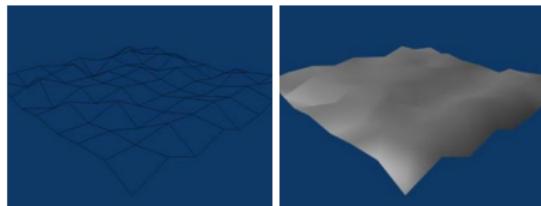
- A feületi pontot tényleges arrébb mozdítjuk.



Nem-szín textúrák

Displacement leképezés

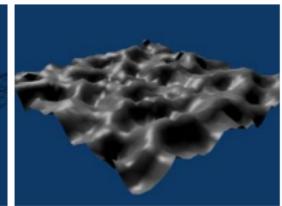
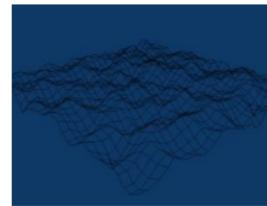
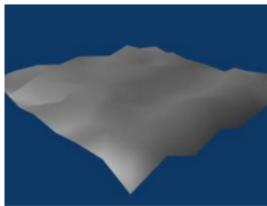
- A feületi pontot tényleges arrébb mozdítjuk.
- A csak háromszögek csúcsait tudjuk elmozdítani \Rightarrow függ a geometria felbontásától.



Nem-szín textúrák

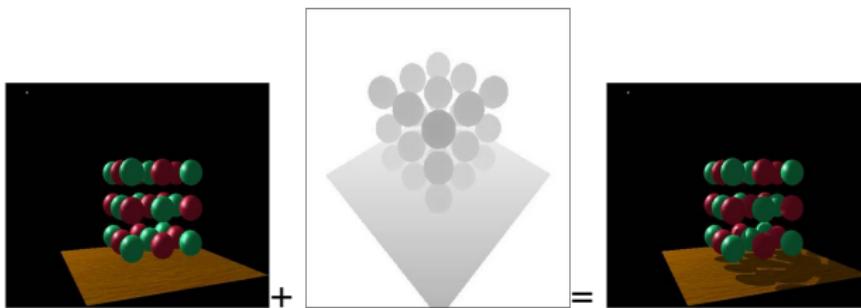
Displacement leképezés

- A feületi pontot tényleges arrébb mozdítjuk.
- A csak háromszögek csúcsait tudjuk elmozdítani \Rightarrow függ a geometria felbontásától.
- Lapos szögben is helyes látványt kapunk.



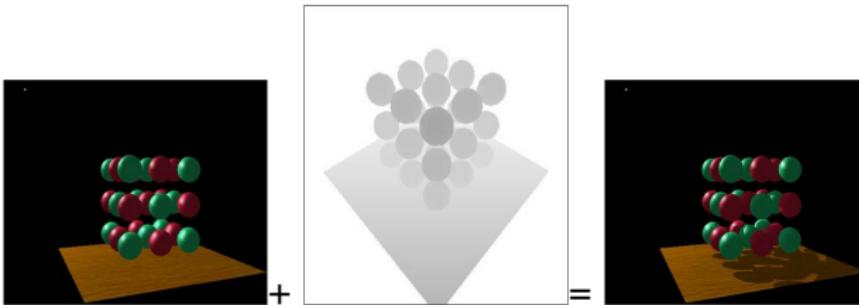
Árnyék térképek

- A fényforrás szemszögéből készítünk egy textúrát, amiben a fényforrástól vett távolságokat tároljuk el.



Árnyék térképek

- A fényforrás szemszögéből készítünk egy textúrát, amiben a fényforrástól vett távolságokat tároljuk el.
- A tényleges rajzolás során ez alapján döntjük el minden pontra, hogy azt éri-e közvetlenül a fény vagy sem.



Nem-szín textúrák

Visszaverődés leképezés

- Sík tükrök esetén használható.

Nem-szín textúrák

Visszaverődés leképezés

- Sík tükrök esetén használható.
- Külön képet készítünk, textúrába mentve, arról, hogy mi látszik tükörirányban.

Nem-szín textúrák

Visszaverődés leképezés

- Sík tükrök esetén használható.
- Külön képet készítünk, textúrába mentve, arról, hogy mi látszik tükrirányban.
- Ezt textúraként ráfeszítjük a felületre a végső kirajzoláskor.

Nem-szín textúrák

Visszaverődés leképezés

- Sík tükrök esetén használható.
- Külön képet készítünk, textúrába mentve, arról, hogy mi látszik tükrirányban.
- Ezt textúraként ráfeszítjük a felületre a végső kirajzoláskor.
- Csak egyszeres tükrözést ad.

Nem-szín textúrák

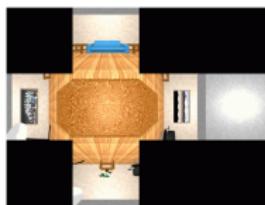
Visszaverődés leképezés

- Sík tükrök esetén használható.
- Külön képet készítünk, textúrába mentve, arról, hogy mi látszik tükrirányban.
- Ezt textúraként ráfeszítjük a felületre a végső kirajzoláskor.
- Csak egyszeres tükrözést ad.
- Tükrönként külön el kell végezni.

Nem-szín textúrák

Környezet leképezés

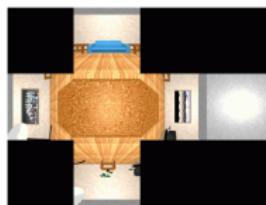
- A színtérünk környezetét végtelenül távolinak tekintjük.



Nem-szín textúrák

Környezet leképezés

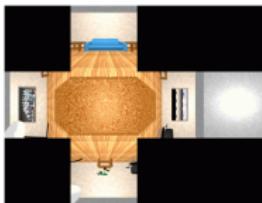
- A színtérünk környezetét végtelenül távolinak tekintjük.
- Speciális textúrában tároljuk.



Nem-szín textúrák

Környezet leképezés

- A színtérünk környezetét végtelenül távolinak tekintjük.
- Speciális textúrában tároljuk.
- Rajzolás a tükröződő felületekhez ebből olvasunk a tükrirány szerint.



Inkrementális képszintézis

Tartalom

1 Textúrázás

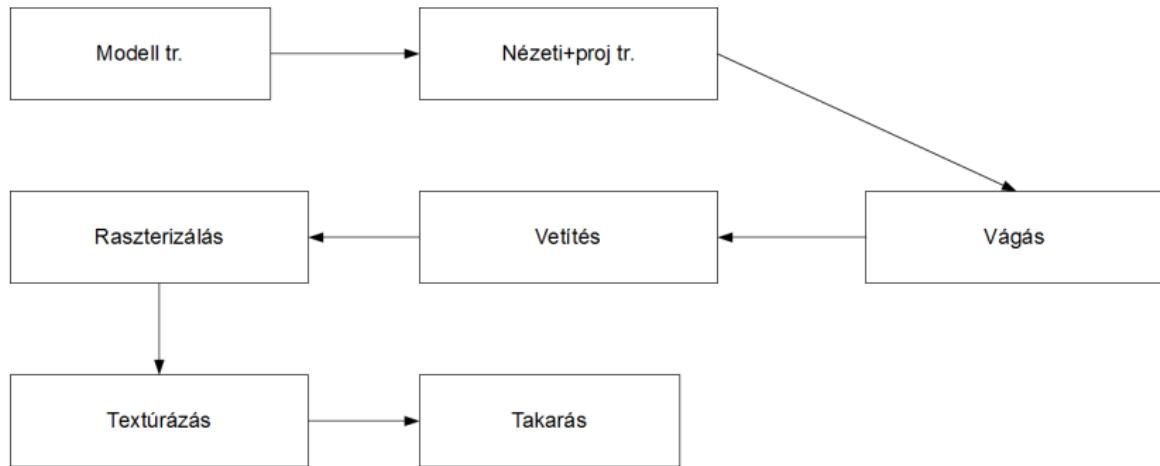
- Bevezetés
- Textúra leképezés
- Paraméterezés
- Textúra szűrés
- Procedurális textúrák
- Nem-szín textúrák

2 Inkrementális képszintézis a GPU-n

- Inkrementális képszintézis
- DirectX
- OpenGL
- Grafikus gyorsítókártya generációk

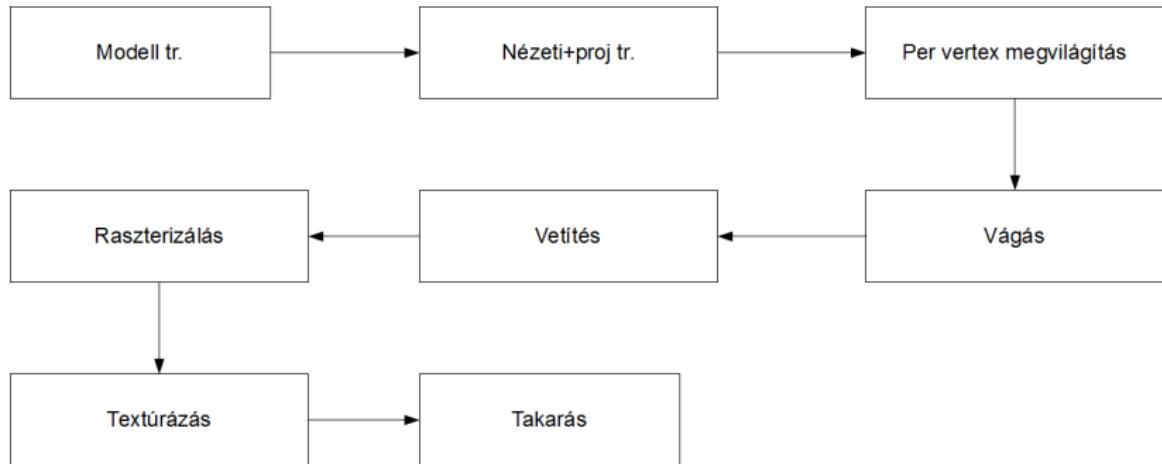
Inkrementális képszintézis

Inkrementális képszintézis



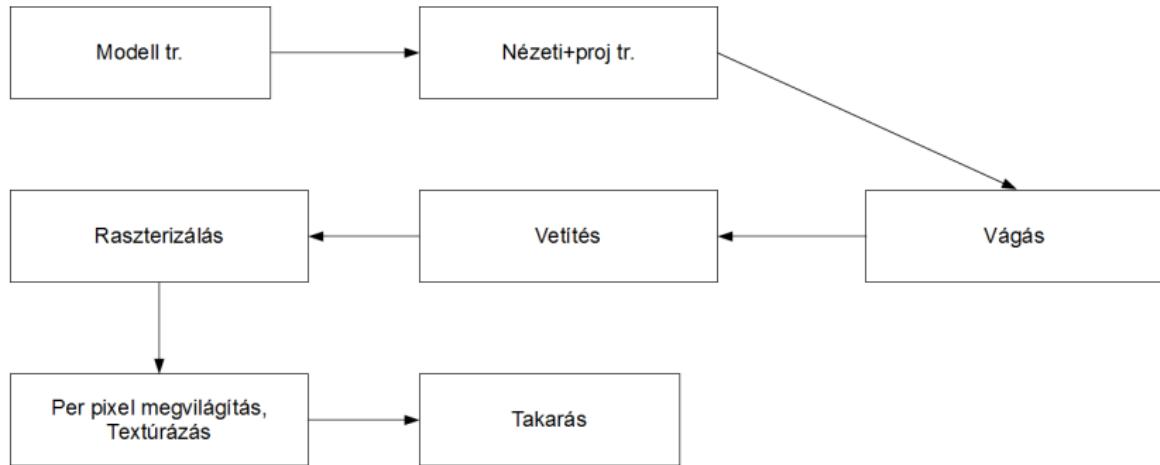
Inkrementális képszintézis

Inkrementális képszintézis



Inkrementális képszintézis

Inkrementális képszintézis



Tartalom

1 Textúrázás

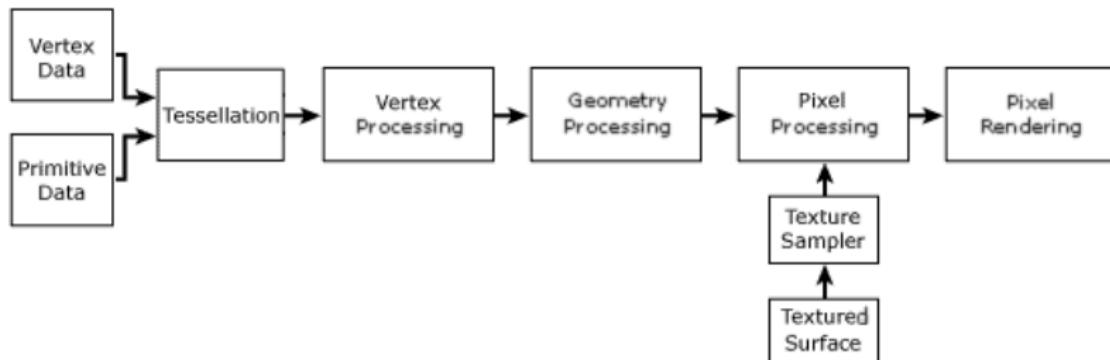
- Bevezetés
- Textúra leképezés
- Paraméterezés
- Textúra szűrés
- Procedurális textúrák
- Nem-szín textúrák

2 Inkrementális képszintézis a GPU-n

- Inkrementális képszintézis
- DirectX
- OpenGL
- Grafikus gyorsítókártya generációk

DirectX

DirectX 9 pipeline



DirectX 9 pipeline

- Vertex data: a modell-koordinátarendszerbeli pozícióadatok

DirectX 9 pipeline

- Vertex data: a modell-koordinátarendszerbeli pozícióadatok
- Primitive data: összekötési szabályok, az előbbiekből hogyan kapunk rajzolási primitíveket

DirectX 9 pipeline

- Vertex data: a modell-koordinátarendszerbeli pozícióadatok
- Primitive data: összekötési szabályok, az előbbiekből hogyan kapunk rajzolási primitíveket
- Tesselation: magasabb rendű primitívek lineáris közelítése, a közelítések csúcspontjainak vertexpufferbe helyezése

DirectX 9 pipeline

- Vertex data: a modell-koordinátarendszerbeli pozícióadatok
- Primitive data: összekötési szabályok, az előbbiekből hogyan kapunk rajzolási primitíveket
- Tesselation: magasabb rendű primitívek lineáris közelítése, a közelítések csúcspontjainak vertexpufferbe helyezése
- Vertex processing: a világ, nézeti és projektív transzformációk elvégzése; a modell-koordinátarendszerből NPKR-be visz ($z \in [0, 1]$)

DirectX 9 pipeline

- Geometry processing: vágás, hátlapeldobás, raszterizáció

DirectX 9 pipeline

- Geometry processing: vágás, hátlapeldobás, raszterizáció
- Textured surface: 1/2/3D-s textúra

DirectX 9 pipeline

- Geometry processing: vágás, hátlapeldobás, raszterizáció
- Textured surface: 1/2/3D-s textúra
- Texture sampler: textúra mintavételező, szűrések stb.

DirectX 9 pipeline

- Geometry processing: vágás, hátlapeldobás, raszterizáció
- Textured surface: 1/2/3D-s textúra
- Texture sampler: textúra mintavételező, szűrések stb.
- Pixel processing: a bejövő fragment színének kiszámítása

DirectX 9 pipeline

- Geometry processing: vágás, hátlapeldobás, raszterizáció
- Textured surface: 1/2/3D-s textúra
- Texture sampler: textúra mintavételező, szűrések stb.
- Pixel processing: a bejövő fragment színének kiszámítása
- Pixel rendering: képernyőre kerülő pixel színének meghatározása (mélységi és egyéb tesztek)

DirectX 9 pipeline

- A fenti fázisok közül kettő programozható, a vertex és a pixel feldolgozás, a többi konfigurálható

DirectX 9 pipeline

- A fenti fázisok közül kettő programozható, a vertex és a pixel feldolgozás, a többi konfigurálható
- Vertex shader:

DirectX 9 pipeline

- A fenti fázisok közül kettő programozható, a vertex és a pixel feldolgozás, a többi konfigurálható
- Vertex shader:
 - bemenete: általában modell-KR-beli pont és egyéb attribútumok

DirectX 9 pipeline

- A fenti fázisok közül kettő programozható, a vertex és a pixel feldolgozás, a többi konfigurálható
- Vertex shader:
 - bemenete: általában modell-KR-beli pont és egyéb attribútumok
 - kimenete: NPKR-beli pont és egyéb attribútumok

DirectX 9 pipeline

- A fenti fázisok közül kettő programozható, a vertex és a pixel feldolgozás, a többi konfigurálható
- Vertex shader:
 - bemenete: általában modell-KR-beli pont és egyéb attribútumok
 - kimenete: NPKR-beli pont és egyéb attribútumok
- Pixel shader:

DirectX 9 pipeline

- A fenti fázisok közül kettő programozható, a vertex és a pixel feldolgozás, a többi konfigurálható
- Vertex shader:
 - bemenete: általában modell-KR-beli pont és egyéb attribútumok
 - kimenete: NPKR-beli pont és egyéb attribútumok
- Pixel shader:
 - bemenete: a raszterizálás végén előálló fragmentek - az összes attribútum elérhető, amit a vertex shaderben előállítottunk

DirectX 9 pipeline

- A fenti fázisok közül kettő programozható, a vertex és a pixel feldolgozás, a többi konfigurálható
- Vertex shader:
 - bemenete: általában modell-KR-beli pont és egyéb attribútumok
 - kimenete: NPKR-beli pont és egyéb attribútumok
- Pixel shader:
 - bemenete: a raszterizálás végén előálló fragmentek - az összes attribútum elérhető, amit a vertex shaderben előállítottunk
 - kimenete: egy színérték

DirectX 9 pipeline

- Régen a vertex és a pixel feldolgozási fázis sem volt programozható

DirectX 9 pipeline

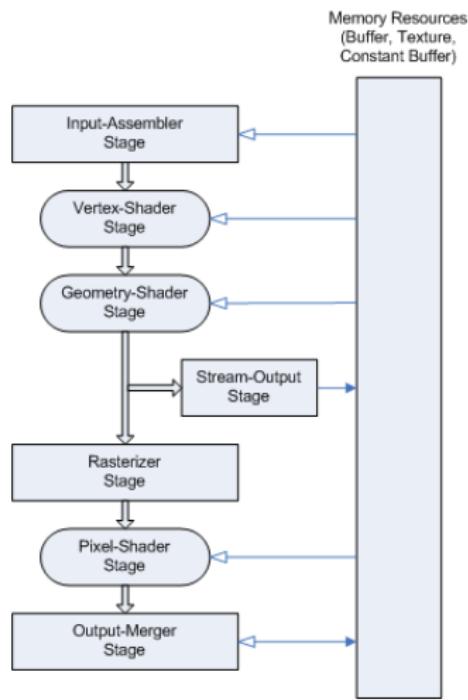
- Régen a vertex és a pixel feldolgozási fázis sem volt programozható
- Csak konfigurálni tudtuk őket → pl. transzformációs mátrixokat átadni, a DX-be bedrótozott megvilágítási modelleket felpáraméterezni stb.

DirectX 9 pipeline

- Régen a vertex és a pixel feldolgozási fázis sem volt programozható
- Csak konfigurálni tudtuk őket → pl. transzformációs mátrixokat átadni, a DX-be bedrótozott megvilágítási modelleket felpáraméterezni stb.
- Ezt hívták Fixed Function Pipeline-nak

DirectX

DirectX 10 pipeline



DirectX 10 pipeline

- Input-Assembler stage: a pipeline bemeneti adatainak előállítását végzi

DirectX 10 pipeline

- Input-Assembler stage: a pipeline bemeneti adatainak előállítását végzi
- Vertex-Shader stage: ld. mint korábban

DirectX 10 pipeline

- Input-Assembler stage: a pipeline bemeneti adatainak előállítását végzi
- Vertex-Shader stage: ld. mint korábban
- Geometry-Shader stage: bemenetként egész primitíveket kap (háromszögek esetén 3, szakaszoknál 2, pontknál 1 pont), eldobhatja a bejövő primitívet vagy egy vagy több új primitívet létrehozhat és a szerelőszalagra helyezheti. Ez egy új, programozható fázis

DirectX 10 pipeline

- Stream-Output Stage: a pipeline-ból a memóriába való adattovábbításra használható, a rasszterizálás előtt történik. Adatokat be- és ki is lehet írni rasszterizáláshoz.

DirectX 10 pipeline

- Stream-Output Stage: a pipeline-ból a memóriába való adattovábbításra használható, a rászterizálás előtt történik. Adatokat be- és ki is lehet írni rászterizáláshoz.
- Rasterizer Stage: rászterizálás

DirectX 10 pipeline

- Stream-Output Stage: a pipeline-ból a memóriába való adattovábbításra használható, a rászterizálás előtt történik. Adatokat be- és ki is lehet írni rászterizáláshoz.
- Rasterizer Stage: rászterizálás
- Pixel-Shader Stage: ld. korábban

DirectX 10 pipeline

- Stream-Output Stage: a pipeline-ból a memóriába való adattovábbításra használható, a rászterizálás előtt történik. Adatokat be- és ki is lehet írni rászterizáláshoz.
- Rasterizer Stage: rászterizálás
- Pixel-Shader Stage: ld. korábban
- Output-Merger Stage: a különböző kimeneti adatok (szín-, mélységi- és egyéb információk) felhasználásával a végső eredmény kiszámítása

Geometry Shader

- Feladata: új geometria generálása, (vagy régi eltüntetése)

Geometry Shader

- Feladata: új geometria generálása, (vagy régi eltüntetése)
- A vertex shader által már transzformált primitívekre fut le.

Geometry Shader

- Feladata: új geometria generálása, (vagy régi eltüntetése)
- A vertex shader által már transzformált primitívekre fut le.
- Bemenete: teljes primitív (szomszédsági infókkal, ha vannak)

Geometry Shader

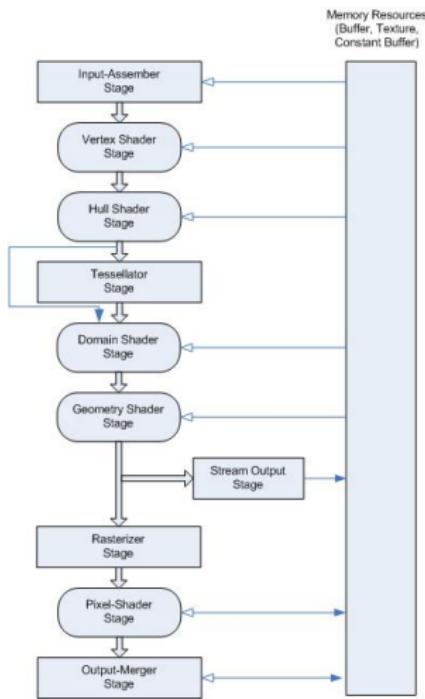
- Feladata: új geometria generálása, (vagy régi eltüntetése)
- A vertex shader által már transzformált primitívekre fut le.
- Bemenete: teljes primitív (szomszédsági infókkal, ha vannak)
- Kimenete: újabb primitív[ek], vagy semmi.

Geometry Shader

- Feladata: új geometria generálása, (vagy régi eltüntetése)
- A vertex shader által már transzformált primitívekre fut le.
- Bemenete: teljes primitív (szomszédsági infókkal, ha vannak)
- Kimenete: újabb primitív[ek], vagy semmi.
- Kimeneti típusok: *PointStream*, *LineStream*, vagy *TriangleStream*

DirectX

DirectX 11 pipeline



DirectX 11 pipeline

- A DX10-es pipeline-nak megfelelő, de itt a tesszeláció már programozható

DirectX 11 pipeline

- A DX10-es pipeline-nak megfelelő, de itt a tesszeláció már programozható
- Emellett a DX11-ben az általános számításokra való felhasználás is előtérbe került → ComputeShader

DirectX 11 pipeline

- A DX10-es pipeline-nak megfelelő, de itt a tesszeláció már programozható
- Emellett a DX11-ben az általános számításokra való felhasználás is előtérbe került → ComputeShader
- Régóta nagy az igény a GPU-kban rejlő számítási kapacitás kihasználására

DirectX 11 pipeline

- A DX10-es pipeline-nak megfelelő, de itt a tesszeláció már programozható
- Emellett a DX11-ben az általános számításokra való felhasználás is előtérbe került → ComputeShader
- Régóta nagy az igény a GPU-kban rejlő számítási kapacitás kihasználására
- Uj. pl.: Intel Core i7 980 XE: 109 GFLOPS, Nvidia GTX 480: 672 GFLOPS, dupla pontosságban

Tesszelációs lépések

- Speciális primitív tipusokat használ

Tesszelációs lépések

- Speciális primitív tipusokat használ
 - négyzetök foltok (*patches*)

Tesszelációs lépések

- Speciális primitív tipusokat használ
 - négyzet foltok (*patches*)
 - háromszög foltok

Tesszelációs lépések

- Speciális primitív tipusokat használ
 - négyzet foltok (*patches*)
 - háromszög foltok
 - iso-vonalak

Tesszelációs lépések

- Speciális primitív tipusokat használ
 - négyzet foltok (*patches*)
 - háromszög foltok
 - iso-vonalak
- Alapötlet: a transzformációkat az alacsony felbontású foltokon végezzük, és tesszelációval, hardveresen bontjuk fel kisebb darabokra és tesszük részletgazdagtávba.

Tesszelációs lépések

- Speciális primitív tipusokat használ
 - négyzet foltok (*patches*)
 - háromszög foltok
 - iso-vonalak
- Alapötlet: a transzformációkat az alacsony felbontású foltokon végezzük, és tesszelációval, hardveresen bontjuk fel kisebb darabokra és tesszük részletgazdagtávba.
- Használató:

Tesszelációs lépések

- Speciális primitív tipusokat használ
 - négyzet foltok (*patches*)
 - háromszög foltok
 - iso-vonalak
- Alapötlet: a transzformációkat az alacsony felbontású foltokon végezzük, és tesszelációval, hardveresen bontjuk fel kisebb darabokra és tesszük részletgazdagtávba.
- Használató:
 - *displacement mapping*

Tesszelációs lépések

- Speciális primitív tipusokat használ
 - négyzetű foltok (*patches*)
 - háromszög foltok
 - iso-vonalak
- Alapötlet: a transzformációkat az alacsony felbontású foltokon végezzük, és tesszelációval, hardveresen bontjuk fel kisebb darabokra és tesszük részletgazdagtávba.
- Használató:
 - *displacement mapping*
 - nézet függő dinamikus LOD (*level of detail*)

Tesszelációs lépések

- Speciális primitív tipusokat használ
 - négyzetű foltok (*patches*)
 - háromszög foltok
 - iso-vonalak
- Alapötlet: a transzformációkat az alacsony felbontású foltokon végezzük, és tesszelációval, hardveresen bontjuk fel kisebb darabokra és tesszük részletgazdagtávba.
- Használató:
 - *displacement mapping*
 - nézet függő dinamikus LOD (*level of detail*)
 - *morph*-olások gyorsítása

Hull Shader Stage

Hull Shader Stage

Hull Shader Stage

- Foltonként dolgozza fel a bementi pontokat.

Hull Shader Stage

- Foltonként dolgozza fel a bementi pontokat.
- Konstansokat rendel a folthoz, amik a tesszelálás módját határozzák meg.

Hull Shader Stage

- Foltonként dolgozza fel a bementi pontokat.
- Konstansokat rendel a folthoz, amik a tesszelálás módját határozzák meg.
- Megadja a tesszeláció mértékét.

Hull Shader Stage

- Foltonként dolgozza fel a bementi pontokat.
- Konstansokat rendel a folthoz, amik a tesszelálás módját határozzák meg.
- Megadja a tesszeláció mértékét.
- Két külön kimenet:

Hull Shader Stage

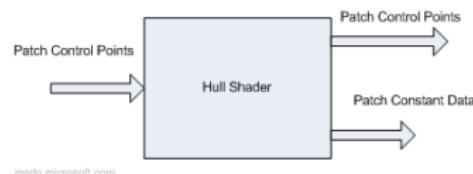
- Foltonként dolgozza fel a bementi pontokat.
- Konstansokat rendel a folthoz, amik a tesszelálás módját határozzák meg.
- Megadja a tesszeláció mértékét.
- Két külön kimenet:
 - Vezérlő pontok a *Domain Shader*-nek

Hull Shader Stage

- Foltonként dolgozza fel a bementi pontokat.
- Konstansokat rendel a folthoz, amik a tesszelálás módját határozzák meg.
- Megadja a tesszeláció mértékét.
- Két külön kimenet:
 - Vezérlő pontok a *Domain Shader*-nek
 - Konstansok a *Tesselator Stage*-nek

Hull Shader Stage

- Foltonként dolgozza fel a bementi pontokat.
- Konstansokat rendel a folthoz, amik a tesszelálás módját határozzák meg.
- Megadja a tesszeláció mértékét.
- Két külön kimenet:
 - Vezérlő pontok a *Domain Shader*-nek
 - Konstansok a *Tesselator Stage*-nek



Tesselator Stage

- Nem programozható.

Tesselator Stage

- Nem programozható.
- Felbontja a tartományt kisebb darabokra (négyszög, háromszög, szakasz).

Tesselator Stage

- Nem programozható.
- Felbontja a tartományt kisebb darabokra (négyszög, háromszög, szakasz).
- u, v (opcionálisan w) koordinátákat állít elő kimenetként.

Tesselator Stage

- Nem programozható.
- Felbontja a tartományt kisebb darabokra (négyszög, háromszög, szakasz).
- u, v (opcionálisan w) koordinátákat állít elő kimenetként.
- "Láthatatlan" kimenet: topológiai információk a *primitive assembly*-nek

Domain Shader Stage

Domain Shader Stage

Domain Shader Stage

- A *Tesselator Stage* során előállított minden egyes pontra lefut.

Domain Shader Stage

- A *Tesselator Stage* során előállított minden egyes pontra lefut.
- Bemenete:

Domain Shader Stage

- A *Tesselator Stage* során előállított minden egyes pontra lefut.
- Bemenete:
 - A *Tesselator Stage*-től:
 $u, v, (w)$ -k.

Domain Shader Stage

- A *Tesselator Stage* során előállított minden egyes pontra lefut.
- Bemenete:
 - A *Tesselator Stage*-től:
 $u, v, (w)$ -k.
 - A *Hull Shader*-től:
vezérlő pontok

Domain Shader Stage

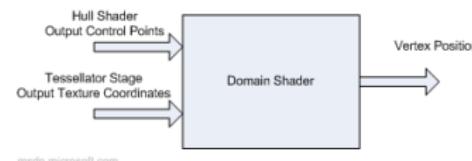
- A *Tesselator Stage* során előállított minden egyes pontra lefut.
- Bemenete:
 - A *Tesselator Stage*-től:
 $u, v, (w)$ -k.
 - A *Hull Shader*-től:
vezérlő pontok
 - A *Hull Shader*-től:
tesszelációs faktorok

Domain Shader Stage

- A *Tesselator Stage* során előállított minden egyes pontra lefut.
- Bemenete:
 - A *Tesselator Stage*-től:
 $u, v, (w)$ -k.
 - A *Hull Shader*-től:
vezérlő pontok
 - A *Hull Shader*-től:
tesszelációs faktorok
- A bemenetekből egyetlen csúcspontot állít elő, és az lesz a kimenet.

Domain Shader Stage

- A *Tesselator Stage* során előállított minden egyes pontra lefut.
- Bemenete:
 - A *Tesselator Stage*-től:
 $u, v, (w)$ -k.
 - A *Hull Shader*-től:
vezérlő pontok
 - A *Hull Shader*-től:
tesszelációs faktorok
- A bemenetekből egyetlen csúcspontot állít elő, és az lesz a kimenet.



msdn.microsoft.com

Compute Shader

- Általános célú számításokat tesz lehetővé a GPU-n.

Compute Shader

- Általános célú számításokat tesz lehetővé a GPU-n.
- Speciális adatszerkezeteket használ.

Compute Shader

- Általános célú számításokat tesz lehetővé a GPU-n.
- Speciális adatszerkezeteket használ.
- Nincsenek megszorítások a ki- és bemenetekre.

Compute Shader

- Általános célú számításokat tesz lehetővé a GPU-n.
- Speciális adatszerkezeteket használ.
- Nincsenek megszorítások a ki- és bemenetekre.
- Felhasználásai:

Compute Shader

- Általános célú számításokat tesz lehetővé a GPU-n.
- Speciális adatszerkezeteket használ.
- Nincsenek megszorítások a ki- és bemenetekre.
- Felhasználásai:
 - Képfeldolgozás/post-processing

Compute Shader

- Általános célú számításokat tesz lehetővé a GPU-n.
- Speciális adatszerkezeteket használ.
- Nincsenek megszorítások a ki- és bemenetekre.
- Felhasználásai:
 - Képfeldolgozás/post-processing
 - Ray-tracing

Compute Shader

- Általános célú számításokat tesz lehetővé a GPU-n.
- Speciális adatszerkezeteket használ.
- Nincsenek megszorítások a ki- és bemenetekre.
- Felhasználásai:
 - Képfeldolgozás/post-processing
 - Ray-tracing
 - Fizika

Compute Shader

- Általános célú számításokat tesz lehetővé a GPU-n.
- Speciális adatszerkezeteket használ.
- Nincsenek megszorítások a ki- és bemenetekre.
- Felhasználásai:
 - Képfeldolgozás/post-processing
 - Ray-tracing
 - Fizika
 - AI

Compute Shader

- Általános célú számításokat tesz lehetővé a GPU-n.
- Speciális adatszerkezeteket használ.
- Nincsenek megszorítások a ki- és bemenetekre.
- Felhasználásai:
 - Képfeldolgozás/post-processing
 - Ray-tracing
 - Fizika
 - AI
- "Testvérei"

Compute Shader

- Általános célú számításokat tesz lehetővé a GPU-n.
- Speciális adatszerkezeteket használ.
- Nincsenek megszorítások a ki- és bemenetekre.
- Felhasználásai:
 - Képfeldolgozás/post-processing
 - Ray-tracing
 - Fizika
 - AI
- "Testvérei"
 - CUDA

Compute Shader

- Általános célú számításokat tesz lehetővé a GPU-n.
- Speciális adatszerkezeteket használ.
- Nincsenek megszorítások a ki- és bemenetekre.
- Felhasználásai:
 - Képfeldolgozás/post-processing
 - Ray-tracing
 - Fizika
 - AI
- "Testvérei"
 - CUDA
 - OpenCL

Tartalom

1 Textúrázás

- Bevezetés
- Textúra leképezés
- Paraméterezés
- Textúra szűrés
- Procedurális textúrák
- Nem-szín textúrák

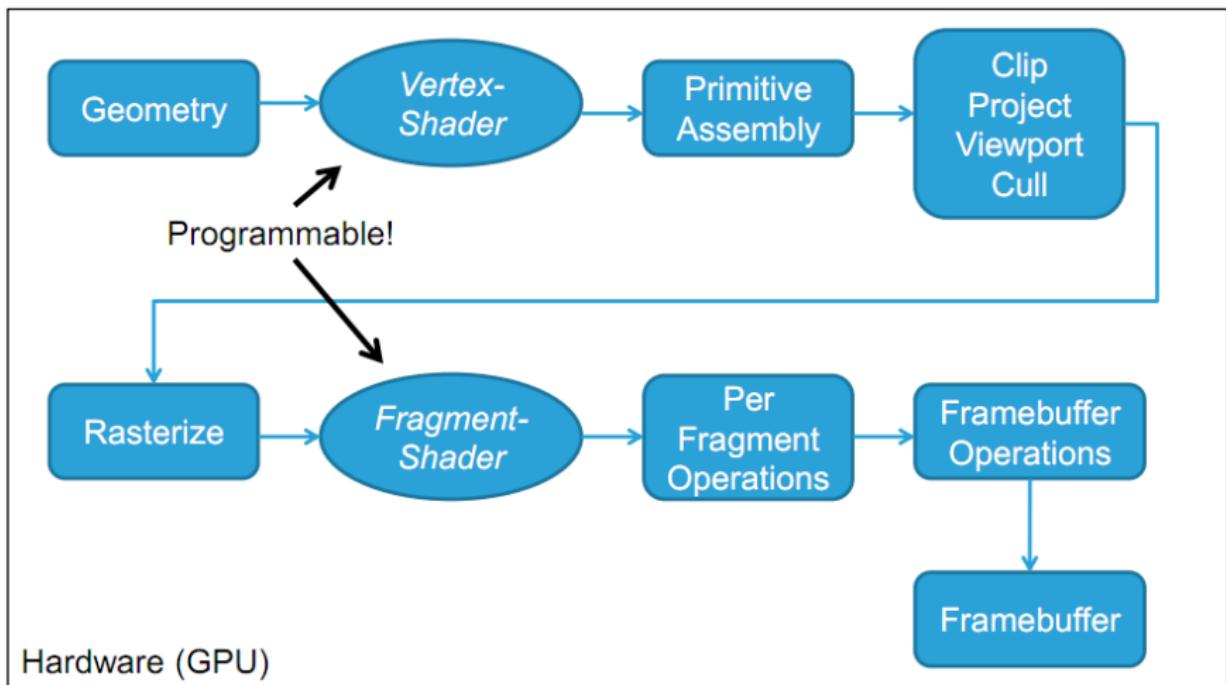
2 Inkrementális képszintézis a GPU-n

- Inkrementális képszintézis
- DirectX
- **OpenGL**
- Grafikus gyorsítókártya generációk

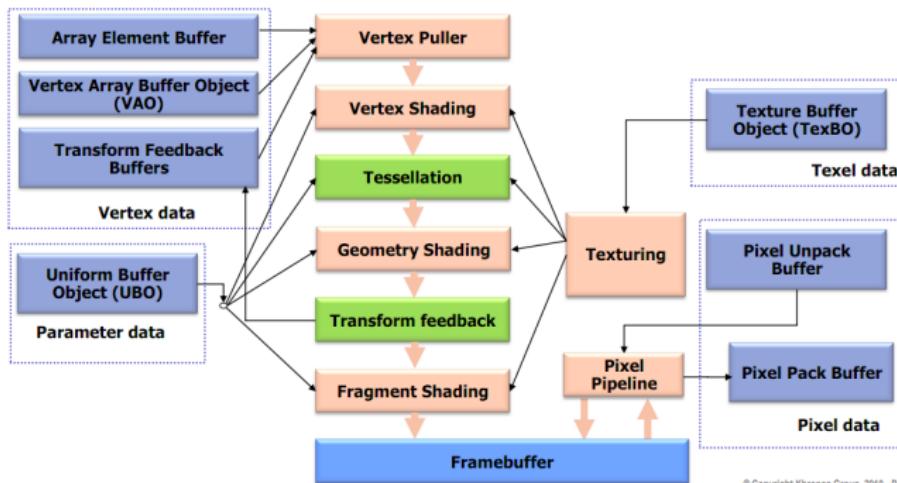
OpenGL pipeline

- Más elnevezésekkel, de nagyjából hasonlóak, mint a DX

OpenGL 3.x pipeline



OpenGL 4.1 pipeline



Grafikus gyorsítókártya generációk

Tartalom

1 Textúrázás

- Bevezetés
- Textúra leképezés
- Paraméterezés
- Textúra szűrés
- Procedurális textúrák
- Nem-szín textúrák

2 Inkrementális képszintézis a GPU-n

- Inkrementális képszintézis
- DirectX
- OpenGL
- **Grafikus gyorsítókártya generációk**

Grafikus gyorsítókártya generációk

1. Generáció

- NVIDIA TNT2, ATI Rage, 3dfx Voodoo3

Grafikus gyorsítókártya generációk

1. Generáció

- NVIDIA TNT2, ATI Rage, 3dfx Voodoo3
- A standard 2d-s videokártyák kiegészítése

Grafikus gyorsítókártya generációk

1. Generáció

- NVIDIA TNT2, ATI Rage, 3dfx Voodoo3
- A standard 2d-s videokártyák kiegészítése
- Csúcspont transzformációkat még a CPU csinálja!

Grafikus gyorsítókártya generációk

1. Generáció

- NVIDIA TNT2, ATI Rage, 3dfx Voodoo3
- A standard 2d-s videokártyák kiegészítése
- Csúcspont transzformációkat még a CPU csinálja!
- A kártya csak a textúrázást, Z-buffer kezelést végezte

Grafikus gyorsítókártya generációk

2. Generáció (1999-2000)

- NVIDIA GeForce 256, GeForce 2, ATI Radeon 7500

Grafikus gyorsítókártya generációk

2. Generáció (1999-2000)

- NVIDIA GeForce 256, GeForce 2, ATI Radeon 7500
- Átveszik a transzformációk és az árnyalás kezelését a CPU-tól.

2. Generáció (1999-2000)

- NVIDIA GeForce 256, GeForce 2, ATI Radeon 7500
- Átveszik a transzformációk és az árnyalás kezelését a CPU-tól.
- Az OpenGL és DirectX 7 is támogatja a hardveres csúcspont transzformációkat

2. Generáció (1999-2000)

- NVIDIA GeForce 256, GeForce 2, ATI Radeon 7500
- Átveszik a transzformációk és az árnyalás kezelését a CPU-tól.
- Az OpenGL és DirectX 7 is támogatja a hardveres csúcspont transzformációkat
- Multi-textúrázás megjelenése: bump map, light map

2. Generáció (1999-2000)

- NVIDIA GeForce 256, GeForce 2, ATI Radeon 7500
- Átveszik a transzformációk és az árnyalás kezelését a CPU-tól.
- Az OpenGL és DirectX 7 is támogatja a hardveres csúcspont transzformációkat
- Multi-textúrázás megjelenése: bump map, light map
- Konfigurálható (driver szinten), de még nem programozható

Grafikus gyorsítókártya generációk

3. Generáció (2001)

- NVIDIA GeForce 3, GeForce 4 Ti, Xbox, ATI Radeon 8500

Grafikus gyorsítókártya generációk

3. Generáció (2001)

- NVIDIA GeForce 3, GeForce 4 Ti, Xbox, ATI Radeon 8500
- A csúcspont pipeline korlátozott programozhatósága

3. Generáció (2001)

- NVIDIA GeForce 3, GeForce 4 Ti, Xbox, ATI Radeon 8500
- A csúcspont pipeline korlátozott programozhatósága
- Fejlettebb pixel szintű konfigurálás, de még nem programozás

3. Generáció (2001)

- NVIDIA GeForce 3, GeForce 4 Ti, Xbox, ATI Radeon 8500
- A csúcspont pipeline korlátozott programozhatósága
- Fejlettebb pixel szintű konfigurálás, de még nem programozás
- 3d-s textúrák, többszörös mintavételezés (antialias-hoz)

Grafikus gyorsítókártya generációk

4. Generáció (2002)

- NVIDIA GeForce FX, ATI Radeon 9700

Grafikus gyorsítókártya generációk

4. Generáció (2002)

- NVIDIA GeForce FX, ATI Radeon 9700
- A csúcspont és pixel pipeline teljesen programozható (erőforrás-korlátok azért még vannak)

Grafikus gyorsítókártya generációk

4. Generáció (2002)

- NVIDIA GeForce FX, ATI Radeon 9700
- A csúcspont és pixel pipeline teljesen programozható (erőforrás-korlátok azért még vannak)
- Magas szintű árnyaló nyelvek (shading languages) megjelenése (NVIDIA Cg, Microsoft HLSL, OpenGL GLSL)

Grafikus gyorsítókártya generációk

4. Generáció (2002)

- NVIDIA GeForce FX, ATI Radeon 9700
- A csúcspont és pixel pipeline teljesen programozható (erőforrás-korlátok azért még vannak)
- Magas szintű árnyaló nyelvek (shading languages) megjelenése (NVIDIA Cg, Microsoft HLSL, OpenGL GLSL)
- Shader Model 2.0 (simple branching)

Grafikus gyorsítókártya generációk

5. Generáció (2004)

- NVIDIA GeForce 6, ATI Radeon X, GeForce 7

Grafikus gyorsítókártya generációk

5. Generáció (2004)

- NVIDIA GeForce 6, ATI Radeon X, GeForce 7
- Több puffer szimultán renderelése

Grafikus gyorsítókártya generációk

5. Generáció (2004)

- NVIDIA GeForce 6, ATI Radeon X, GeForce 7
- Több puffer szimultán renderelése
- 64bites pipeline

Grafikus gyorsítókártya generációk

5. Generáció (2004)

- NVIDIA GeForce 6, ATI Radeon X, GeForce 7
- Több puffer szimultán renderelése
- 64 bites pipeline
- PCIe busz

Grafikus gyorsítókártya generációk

5. Generáció (2004)

- NVIDIA GeForce 6, ATI Radeon X, GeForce 7
- Több puffer szimultán renderelése
- 64 bites pipeline
- PCIe busz
- Több memória, hosszabb csúcspont árnyaló programok

Grafikus gyorsítókártya generációk

5. Generáció (2004)

- NVIDIA GeForce 6, ATI Radeon X, GeForce 7
- Több puffer szimultán renderelése
- 64 bites pipeline
- PCIe busz
- Több memória, hosszabb csúcspont árnyaló programok
- Shader Model 3.0 (branching and looping in the pixel shader (physics))

Grafikus gyorsítókártya generációk

5. Generáció (2004)

- NVIDIA GeForce 6, ATI Radeon X, GeForce 7
- Több puffer szimultán renderelése
- 64 bites pipeline
- PCIe busz
- Több memória, hosszabb csúcspont árnyaló programok
- Shader Model 3.0 (branching and looping in the pixel shader (physics))
- HDRI, SLI, TSAA, TMAA

Grafikus gyorsítókártya generációk

6. Generáció (2007)

- DirectX 10

Grafikus gyorsítókártya generációk

6. Generáció (2007)

- DirectX 10
- Shader Model 4.0 (Unified Shader Model, geometry shader)

Grafikus gyorsítókártya generációk

6. Generáció (2007)

- DirectX 10
- Shader Model 4.0 (Unified Shader Model, geometry shader)
- Unified Shading Architecture

Grafikus gyorsítókártya generációk

6. Generáció (2007)

- DirectX 10
- Shader Model 4.0 (Unified Shader Model, geometry shader)
- Unified Shading Architecture
- Shading performance 2x pixel, 12x vertex above G71

Grafikus gyorsítókártya generációk

6. Generáció (2007)

- DirectX 10
- Shader Model 4.0 (Unified Shader Model, geometry shader)
- Unified Shading Architecture
- Shading performance 2x pixel, 12x vertex above G71
- 700 Mtransistors

Grafikus gyorsítókártya generációk

6. Generáció (2007)

- DirectX 10
- Shader Model 4.0 (Unified Shader Model, geometry shader)
- Unified Shading Architecture
- Shading performance 2x pixel, 12x vertex above G71
- 700 Mtransistors
- 130W to 300W

Grafikus gyorsítókártya generációk

7. Generáció (2009-)

- DirectX 11, OpenGL 4.1

Grafikus gyorsítókártya generációk

7. Generáció (2009-)

- DirectX 11, OpenGL 4.1
- Shader Model 5.0 (Compute Shader, Tesselation Shaders: Hull Shader & Domain Shader)

Grafikus gyorsítókártya generációk

7. Generáció (2009-)

- DirectX 11, OpenGL 4.1
- Shader Model 5.0 (Compute Shader, Tesselation Shaders: Hull Shader & Domain Shader)
- Többszállúság

Grafikus gyorsítókártya generációk

7. Generáció (2009-)

- DirectX 11, OpenGL 4.1
- Shader Model 5.0 (Compute Shader, Tesselation Shaders: Hull Shader & Domain Shader)
- Többszállúság
- *Dynamic Shader Linkage*: OOP jellegű szolgáltatások HLSL-ben

Grafikus gyorsítókártya generációk

7. Generáció (2009-)

- DirectX 11, OpenGL 4.1
- Shader Model 5.0 (Compute Shader, Tesselation Shaders: Hull Shader & Domain Shader)
- Többszállúság
- *Dynamic Shader Linkage*: OOP jellegű szolgáltatások HLSL-ben
- 3000+ Mtransistors

Grafikus gyorsítókártya generációk

7. Generáció (2009-)

- DirectX 11, OpenGL 4.1
- Shader Model 5.0 (Compute Shader, Tesselation Shaders: Hull Shader & Domain Shader)
- Többszállúság
- *Dynamic Shader Linkage*: OOP jellegű szolgáltatások HLSL-ben
- 3000+ Mtransistors
- 1000 GFLOPs