Počítačová grafika 2

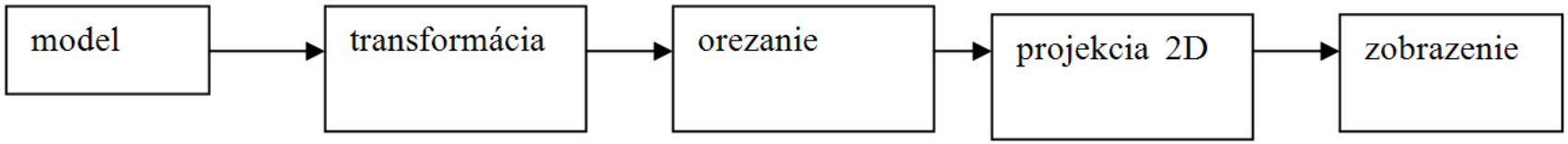
pipeline, kamera, orezávanie, viditeľnosť

Martin Florek
florek@sccg.sk

FMFI UK

10. marca 2009

Zobrazovací „pipeline“



Kamera v 3D scéne

- čo potrebujeme?

Kamera v 3D scéne

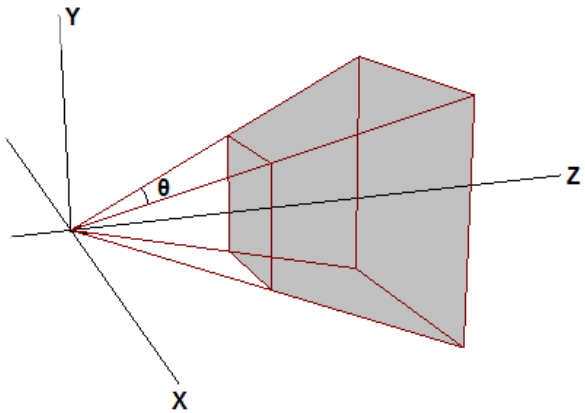
- čo potrebujeme?
- pozíciu
- orientáciu

Kamera v 3D scéne

- čo potrebujeme?
- pozíciu
- orientáciu
- uhoľ pohľadu
- náklon – (vektor „up“)

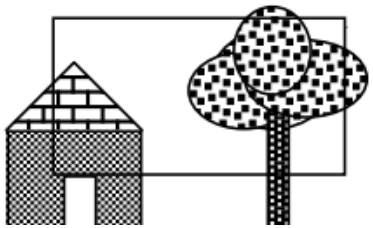
Pohľadový objem

- vidí všetko od svojej pozície v danom smere
- kanonický pohľadový objem
  - aproximuje pohľad oka
  - náročné výpočty
- aproximovať zrezaným ihlanom („frustum“)
  - vhodné pre obdĺžnikový monitor
  - ľahké lineárne rovnice
- predná („near“) a zadná („far“) orezávacia rovina



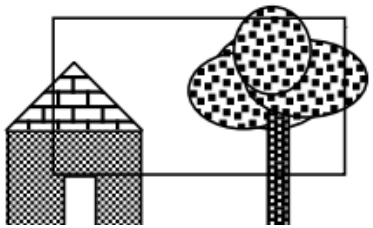
Orezávanie

- 2D – do okna (minulý semester)

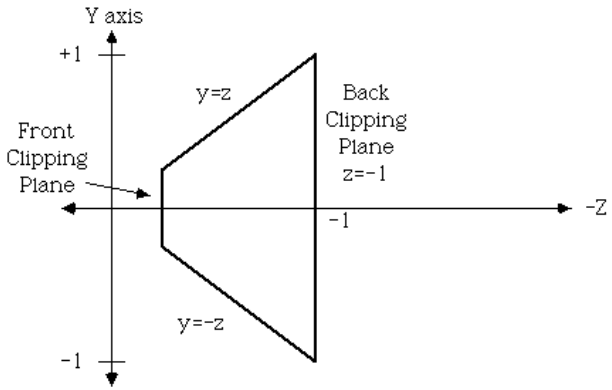


Orezávanie

- 2D – do okna (minulý semester)

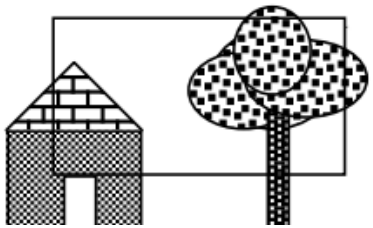


- 3D – do pohľadového objemu

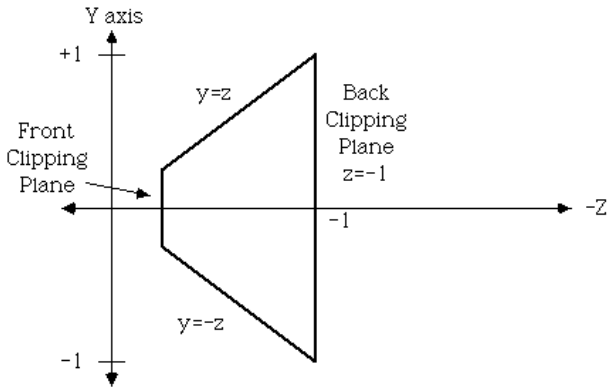


Orezávanie

- 2D – do okna (minulý semester)



- 3D – do pohľadového objemu



- odvrátené steny

Pohľadové orezávanie

- frustum culling
- nerenderovať objekty mimo pohľad
- testovať, či sa daný objekt nachádza v pohľade
  - testovanie všetkých polygónov objektu je náročné

Pohľadové orezávanie

- frustum culling
- nerenderovať objekty mimo pohľad
- testovať, či sa daný objekt nachádza v pohľade
  - testovanie všetkých polygónov objektu je náročné
  - zabaliť objekty do obálok

Pohľadové orezávanie 2

- keď polygón pretína okno. . .

Pohľadové orezávanie 2

- keď polygón pretína okno...
- rozšírenie Cohen-Sutherlandovho algoritmu
  - 6 bitov namiesto 4
  - vľavo, vpravo, nad, pod, pred a za
  - triviálne prijatia a zamietnutia?
  - v iných prípadoch rátať priesečníky...
  - homogénne súradnice to uľahčujú

Odvrátené steny

- ďalšia optimalizácia
- netreba vykreslovať strany, ktoré sú odvrátene
- ako?

Odvrátené steny

- ďalšia optimalizácia
- netreba vykreslovať strany, ktoré sú odvrátene
- ako?
- uhol medzi normálou a pohľadovým vektorom
  - $N \cdot V \geq 0$

Viditeľnosť

- na čo?

Viditeľnosť

- na čo?
- scéna je viac realistická
- konvexné teleso je ľahké
- viac telies, nekonvexné teleso $\longrightarrow$ vzájomné prekrývanie
- prvé algoritmy na prelome 60. a 70. rokov

Delenie algoritmov

- podľa výstupu
- rastrové – HSR
  - hidden surface removal
  - výstupom je obraz, ktorého pixle obsahujú farbu zodpovedajúcu viditeľným plochám
  - zavisí na rozlíšení
- líniové – HLR
  - hidden line removal
  - výstup je súbor viditeľných úsečiek
  - nezávisí na rozlíšení

Delenie algoritmov 2

- podľa výpočtovej zložitosti
- objektovo orientované
  - testovanie každého objektu s každým
  - pre každý objekt zistí, ktorá jeho časť je vidieť
- obrazovo orientované
  - pre každý pixel zistí, ktorý objekt je v ňom
- nejednoznačné delenie – metódy v sebe požívaju aj iné metódy

Robertsonov algoritmus

- rozdelíme hrany na predné, obrysové a zadné
- zadné zahodíme
- postupne testujeme všetky zaujímávé hrany, čí ich neprekrýva nejaký mnohosten
  - ak prekrýva úplne, tak hranu zahodíme
  - ak prekrýva čiastočne, tak hranu rozdelíme a ďalej testujeme viditeľné časti
- na konci nám ostane zoznam viditeľných hrán, ktoré vykreslíme

Appelov algoritmus

- pracuje v priestore objektov
- vychádza z toho, že každá hrana je prienikom dvoch stien
- rozdelíme hrany ako u Robertsonovho algoritmu
- potenciálne viditeľné hrany testujeme voči obrysovým
- priesečníky, kde testovaná hrana „vstupuje“ pod obrysovú hranu, sa ukladajú do zoznamu a ohodnocujú sa ako vstupné a výstupné
- nie pre telesá, ktoré pretínaju samé seba

Rastrové algoritmy

- aké?

Rastrové algoritmy

- aké?
- Maliarov algoritmus
- Warnockov algoritmus
- zobrazovanie priestorových grafov
- Z-Buffer
  - riadkový „scan-line“ z-buffer

Maliarov algoritmus

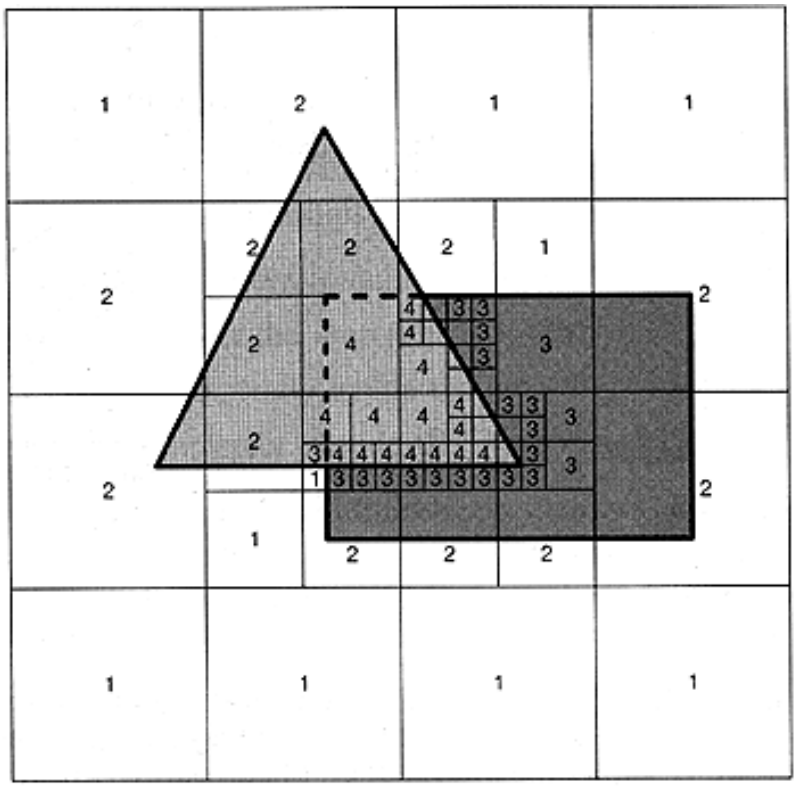
- depth-sorting algoritmus
- utriedime polygóny podľa vzdialenosti od kamery
- vykreslíme od zadu, do predu
- blízke plochy sú vykreslované neskôr takže prekreslia vzdialenejšie
- obálky na objekty

Maliarov algoritmus

- depth-sorting algoritmus
- utriedime polygóny podľa vzdialenosti od kamery
- vykreslíme od zadu, do predu
- blízke plochy sú vykreslované neskôr takže prekreslia vzdialenejšie
- obálky na objekty
- čo sa nedá vykresliť?

Warnockov algoritmus

- zložitý problém rozložíme na menšie
  - divide and conquer

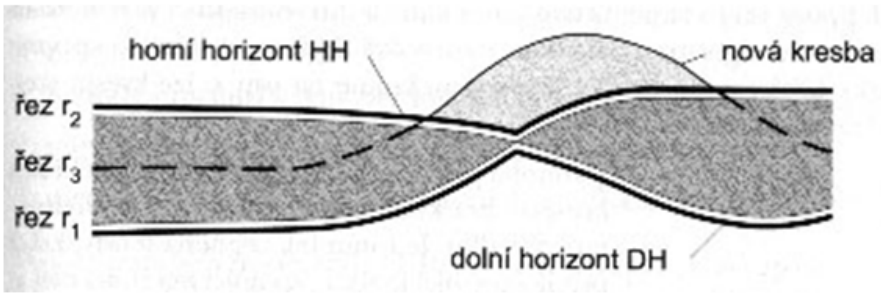


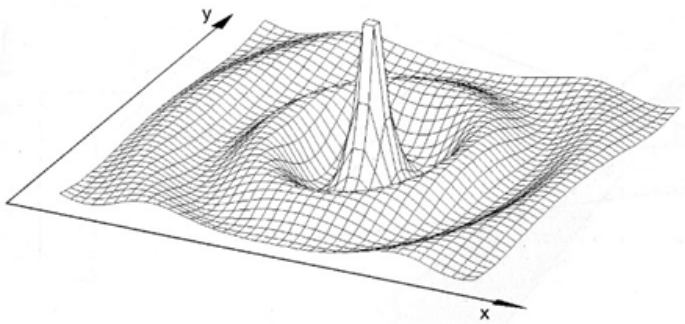
Warnockov algoritmus 2

1. ak okno neobsahuje žiadny objekt, je vyplnené farbou pozadia
2. do okna zasahuje práve jedna plocha, potom bude táto plocha vo vnútri okna vyplnená a zvyšok okna získa farbu pozadia
3. do okna zasahuje viacej plôch, ale plocha najbližšia k pozorovateľovi ich všetky prekrýva, preto je celé okno vyplnené farbou tejto plochy
4. okno obsahuje komplikovanejšiu časť scény, preto sa rozdelí na štyri menšie okná a algoritmus sa opakuje pre každé okno zvlášť

zobrazovanie priestorových grafov

- kreslíme jednotlivé rezy od pozorovateľa smerom dozadu
- viditeľnosť určujeme metódou *plávajúceho horizontu*
- využitie dvoch polí – horný a dolný horizont





Z-Buffer

- depth-buffer
- najznámejšia a najefektívnejšia metóda
- veľké nároky na pamäť
- vysoká rýchlosť
- udržiava sa najmenšia z-súradnica jednotlivých pixelov a tiež ich farba
- stenu transformujeme a rasterizujeme a porovnaváme z-tovú súradnicu so z-bufferom
- ak je hodnota menšia, tak je bod bližšie a prepíšeme farbu a z-buffer

riadkový Z-Buffer

- scanline z-buffer
- používa menej pamäti
- viditeľnosť je spracovaná vždy len v jednom riadku
- každá plocha je spracovaná niekoľko krát
- dá sa zlepšiť utriedením hrán podľa y-súradnice

Ďakujem za pozornosť

florek@sccg.sk
www.sccg.sk/~florek