

# PB001: Úvod do informačních technologií

Luděk Matyska

Fakulta informatiky Masarykovy univerzity

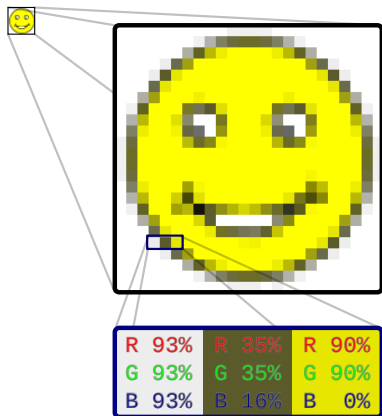
podzim 2014

# Obsah přednášky

- 1 Rastrové displeje a algoritmy
- 2 Modely a modelování
- 3 Renderování
- 4 Renderování na GPU

# Rastrový obraz

- obraz je 2D pole pixelů = obrazových bodů
- barva každého pixelu je definována **b** bity, tzv. barevná hloubka
  - 1 bit: černobílý obraz
  - barevný obraz: 8, 15, 16, 24 (True Color), až 96 bitů



# Technologie displejů

## CRT

- tři svazky elektronů jsou urychlovány a cíleny na lumiforovou vrstvu s RGB oblastmi

## LCD

- organické molekuly uložené mezi dvěma polarizačními filtry s kolmými osami polarizace
- v klidové poloze polarizují světlo o  $90^\circ$  a umožňují jeho průchod.
- v excitované poloze nepolarizují a pixel se jeví jako nerozsvícený
- nevydává světlo: vyžaduje podsvícení, či reflexní vrstvu

# Technologie displejů

## Plazmové displeje

- plyn uzavřený v malých buňkách (3 na pixel) je excitován el. polem a vydává UV záření
- UV záření dopadá na fosfor uvnitř buňky a ten vydá viditelné světlo.

## OLED

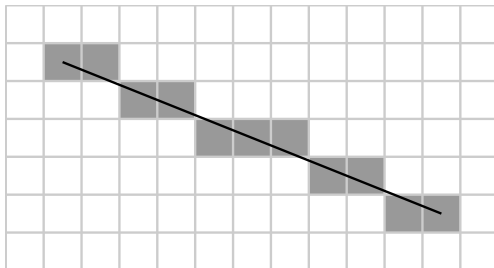
- několik vrstev organického materiálu uložených mezi anodou a katodou
- při průchodu el. proudu organickým materiálem dochází k emisi viditelného světla
- aktivní zdroj světla (nepotřebuje podsvícení), ohebné

## Dotykové displeje

- spojení obrazového výstupu a hmatového vstupu

# Rastrová konverze úseček

Cíl: převedení spojitě úsečky do rastrové reprezentace.



Podél dané úsečky se v krocích po ose  $x$  počítá nejbližší pixel v ose  $y$ .

- výpočet v pomoci `round()` v každém kroku je neefektivní
- inkrementální výpočet: Bresenhamův algoritmus

# Výplň ploch

Cíl: obarvení všech pixelů v dané oblasti.

Možné definice oblastí:

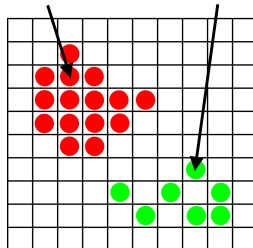
- všechny pixely dané barvy
- všechny pixely v dané vzdálenosti od pixelu
- oblast definovaná polygonem

Definice sousedního pixelu:

- 4-směrná: společná hrana
- 8-směrná: společná hrana, či vrchol

Pixelově definované oblasti:

4 směrná      8 směrná



# Výplň polygonální oblasti

Záplavové vyplňování:

- zvol jeden pixel uvnitř oblasti
- rekurzivně obarvuj sousedy

Řádkové vyplňování:

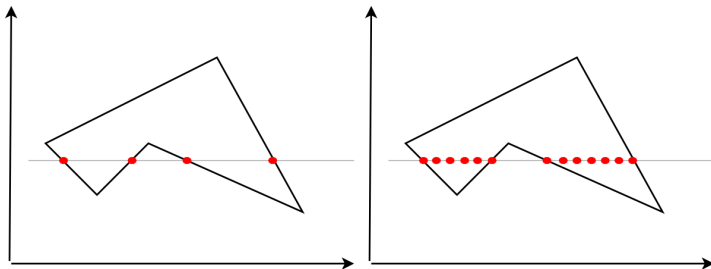
- rekurze probíhá po sousedících řádcích, ne pixelech
- výrazně efektivnější



# Výplň polygonální oblasti

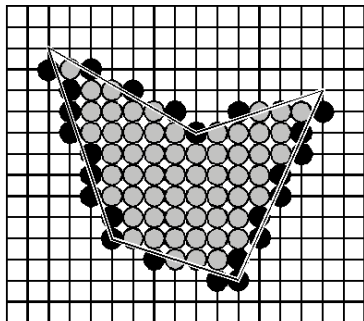
Paritní vyplňování:

- najdi průsečíky řádky s polygonem
- seřaď podle polohy na ose x
- vybarvi sudé úseky

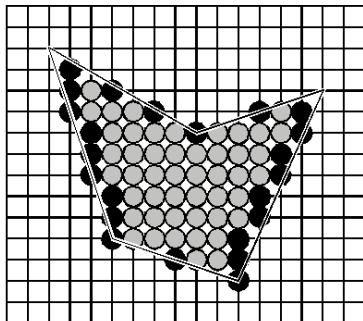


# Výplň polygonální oblasti

Nejednoznačnost hranice a tedy i výplně:



(a)



(b)

● Span extrema    ● Other pixels in the span

# Antialiasing

Převodem spojitého obrazu na diskrétní rastrovou reprezentaci vznikají chyby:

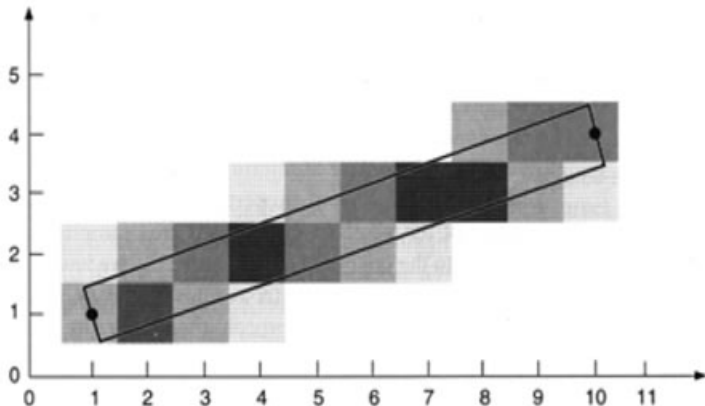
- ztráta detailu
- vznik nežádoucích artefaktů
- rozpad tvaru

Řešení:

- zvýšené rozlišení
- předfiltrování
- postfiltrování

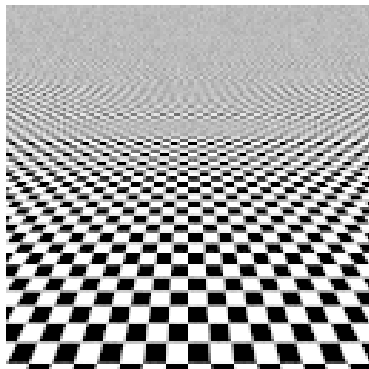
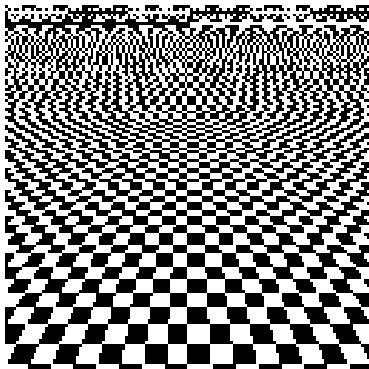
# Antialiasing: předfiltrování

- aplikuje se během rasterizace
- každému pixelu je nastavena intenzita poměrně k velikosti plochy, kterou je zakrýván rasterovaným objektem



# Alias: rozpad tvaru

Zvýšené rozlišení (supersampling): obraz je vykreslen ve větším rozlišení a následně zmenšen.



# Rasterizace písma

Bez antialiasingu:

sample



Antialiasing:

sample



Antialiasing a hinting = předpočítané parametry pro daný font a rozlišení:

sample



# Modely a modelování

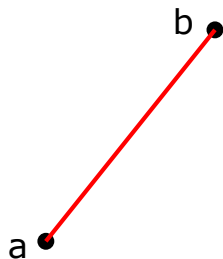
Cíl: popsat „co je na obraze“

- ze základních primitiv se skládají komplexní tvary
- 2D – vektorová grafika
  - úsečka, křivka, elipsa/kužnice, mnohoúhelník. . .
- 3D – popis povrchů
  - 2D objekty s obsahem, parametrické plochy, spojování plátů. . .

# Parametrické křivky

úsečka

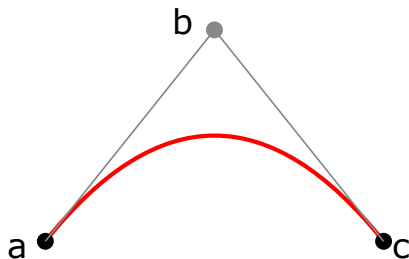
- koncové body **a, b**



$$p(t) = (1 - t)a + tb$$

Bezierova křivka

- koncové body **a, c**
- řídicí bod **b**



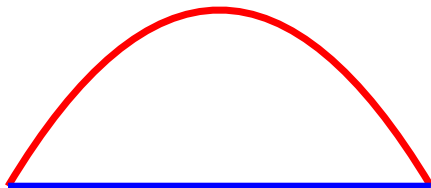
$$p(t) = (1 - t)^2a + 2t(1 - t)b + t^2c$$



# Parametrické křivky – příklad

## Scalable Vector Graphics (SVG)

```
<path d="M50,300 Q200,50 350,300" fill="none"  
stroke="red" stroke-width="5" />  
<path d="M50,300 L350,300" fill="none"  
stroke="blue" stroke-width="5" />
```

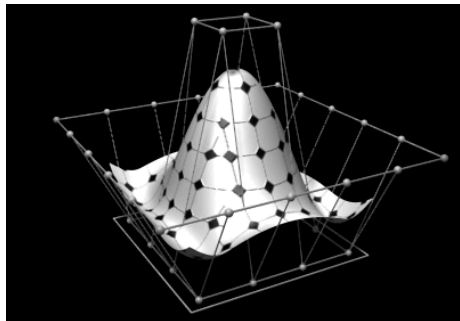


# Parametrické plochy

- umožňují popis hladkých zakřivených povrchů
- vhodné pro průmyslový design

Možnosti definice:

- okrajovými křivkami
- polygonovou sítí



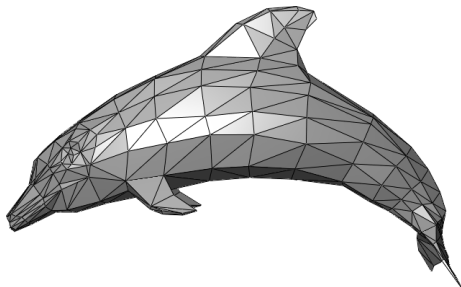
# Polygonové modely

Polygonový model:

- tvar je složen z konvexních 2D primitiv
- dvoj-, troj- mnohoúhelníky (polygony)
- snadné vykreslení

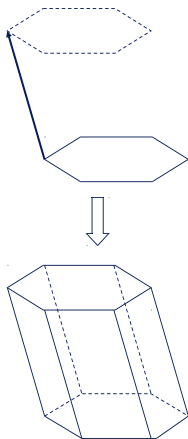
Techniky úprav povrchové sítě:

- tažení (extrudování) povrchu
- rotace profilu kolem osy
- zjemnění a deformace sítě
- konstruktivní geometrie těles

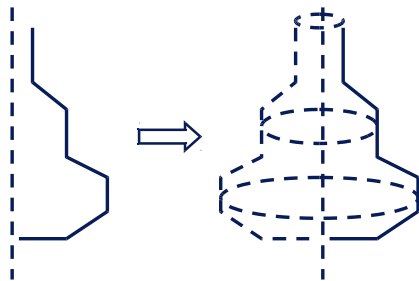


# Vytváření polygonových modelů

## Tažení povrchu



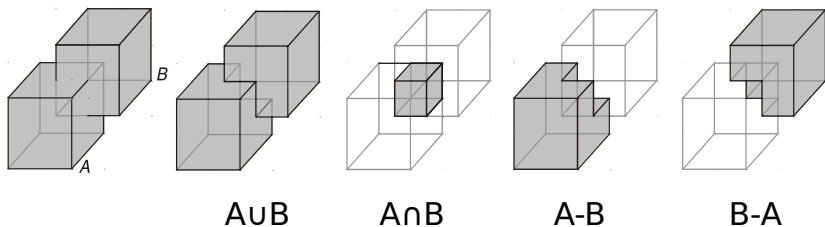
## Rotace



# Konstruktivní geometrie těles

Komplexní objekty jsou z jednodušších vytvářeny pomocí boolských operací:

- sjednocení
- průnik
- rozdíl

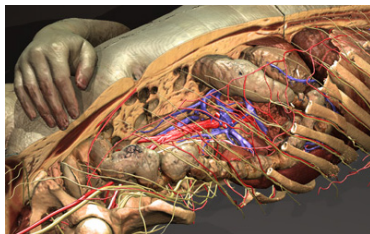
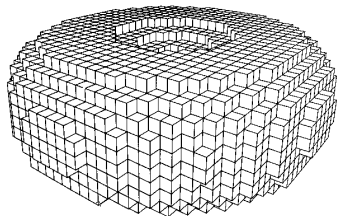


# Objemové modelování

- prostor uniformně rozdělen na voxely
- voxely mají různou barvu, průsvitnost. . .

## Aplikace:

- zobrazení medicínských dat
- objem zadáván po řezech
- možnost selektivního zobrazení



# Renderování

Cíl: vytvoření obrazu na základě modelu.

Popis scény:

- geometrie objektů
- osvětlení
- textury
- směr pohledu
- stínování

# Popis scény

## Geometrie objektů

- polygonové/parametrické modely
- úroveň detailu a počet objektů ovlivňují výpočetní náročnost

## Osvětlení

- popis zdrojů světla a jejich vlastností
- různé modely šíření světla

## Stínování

- úprava úrovně jasu povrchu v závislosti na osvětlení
- pomocí stínů vnímáme hloubku, tvary. . .



# Textury

Technika přidání detailu na povrch modelu.

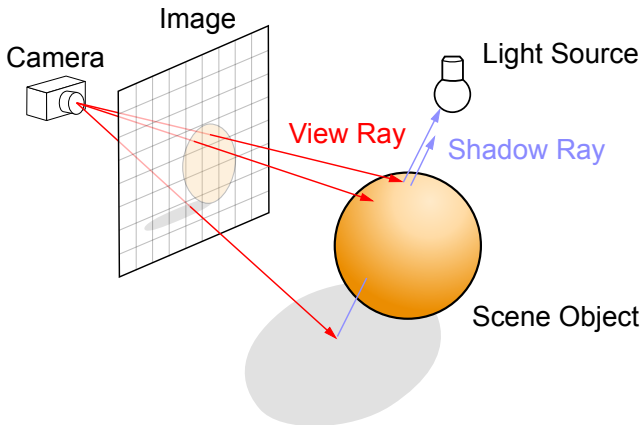
- určuje optické vlastnosti materiálu objektu
  - barva, průsvitnost, lom světla, ...
- přidává detailní změny geometrie
  - normálové mapy (výstupky)
- na jeden povrch je možné aplikovat více textur

Textury

- rasterové
  - rasterový obraz je mapován („natažen“) na povrch
  - výsledek je ovlivněn rozlišením textury a použitou interpolací
- procedurální
  - vlastnosti pixelů povrchu jsou zadány funkcí
  - vyžaduje programovatelný HW
  - dobře škáluje na výsledné rozlišení obrazu

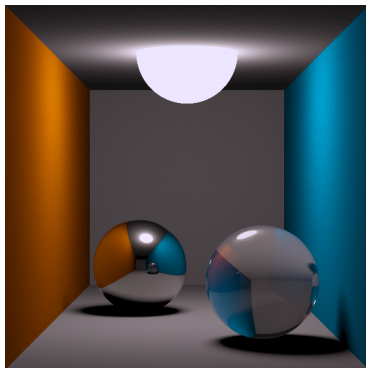
# Zpětné sledování paprsku (Raytracing)

- sleduje cesty paprsků z oka do zdrojů světla
- daný stupeň odrazů
- umožňuje stínování, lesklé povrchy...



# Distribuované sledování paprsku

- každý paprsek nahrazen svazkem paprsků
- výsledkem je průměr získaných hodnot
- umožňuje hloubku ostrosti, měkké stíny...



# Renderování na GPU

Renderování v reálném čase:

- nižší nároky na kvalitu, důležitý výkon (25 fps)
- rasterizace místo raytracingu
  - geometrie transformována do 2D
  - určeny viditelné trojúhelníky
  - převedení na pole pixelů

Programovatelné GPU

- novější generace GPU umožňují obecnější výpočty
- komplexnější per-pixel efekty (bump mapy, shadery)
- možné využít GPU i mimo jednoduchou rasterizaci (raytracing, konverze videa, vědecké výpočty. . . )

# Vývoj práce s GPU

## Statické API (application programming interface)

- pevně dané množina funkcí
- abstrakce od konkrétního HW
- OpenGL 1.0, DirectX do verze 7

## Programovatelné shadery

- umožňují vytváření jednoduchých procedur vykonávaných na GPU
- OpenGL 2.0, DirectX 8 a výše

## GPGPU (General-purpose computing on graphics processing units)

- další rozšíření programovatelnosti GPU
- grafická karta jako stream procesor
- OpenCL, CUDA,...