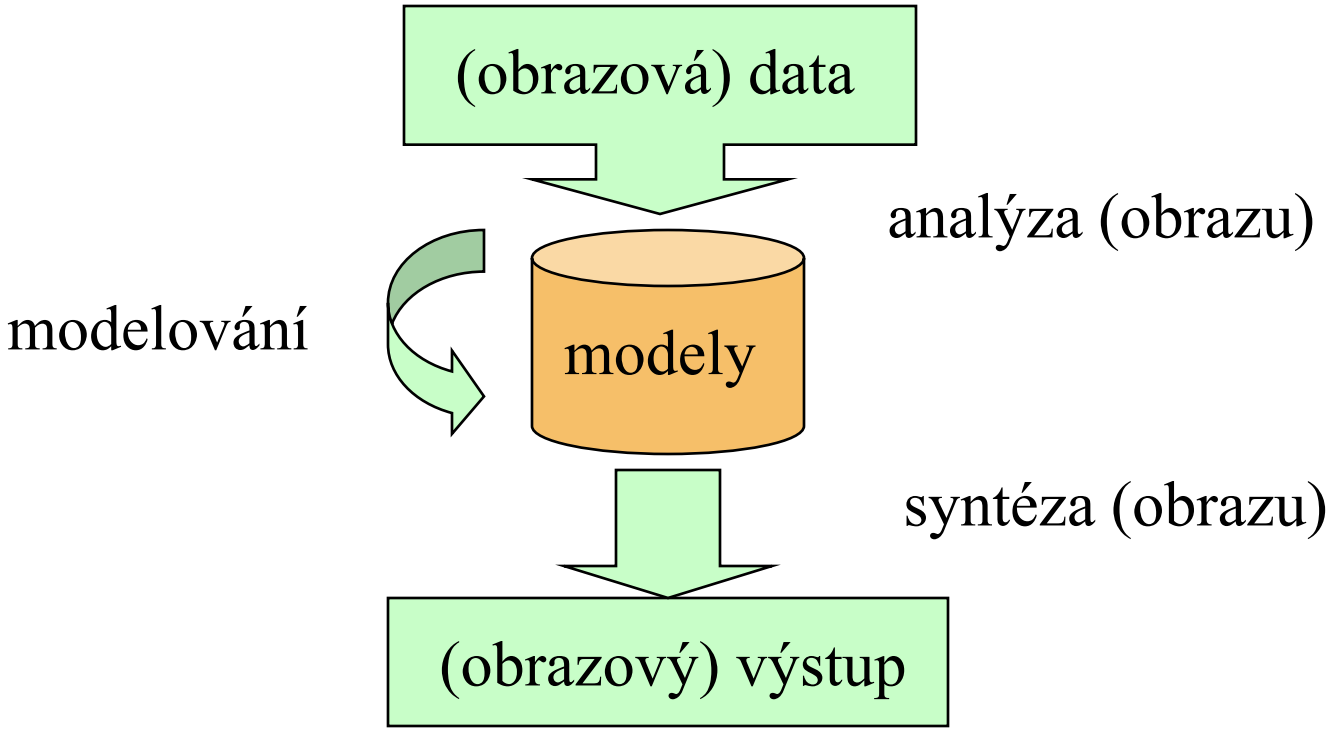
Počítačová grafika letem světem

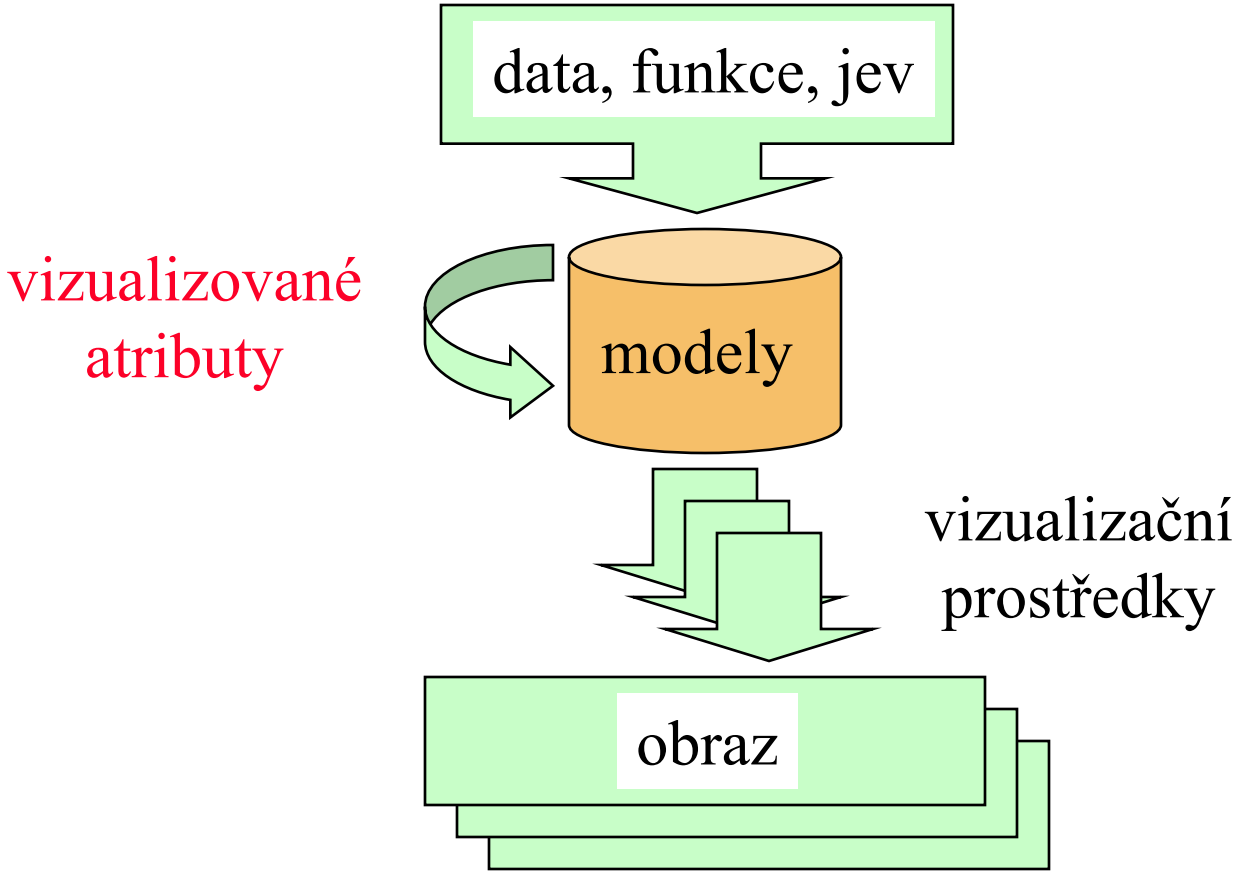
© 2002 Jiří Sochor
FI MU Brno

sochor@fi.muni.cz
http://www.fi.muni.cz/usr/sochor/

Analýza a syntéza obrazu



Vizualizace



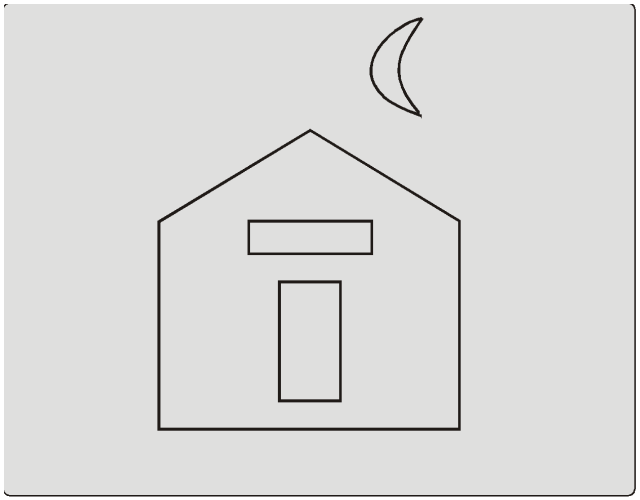
Problémové okruhy

- co a jak nakreslit
- jak to vytvořit
- jak se na to podívat
- jak to osvětlit
- jak modelovat „realitu“
- jak to rychle vypočítat

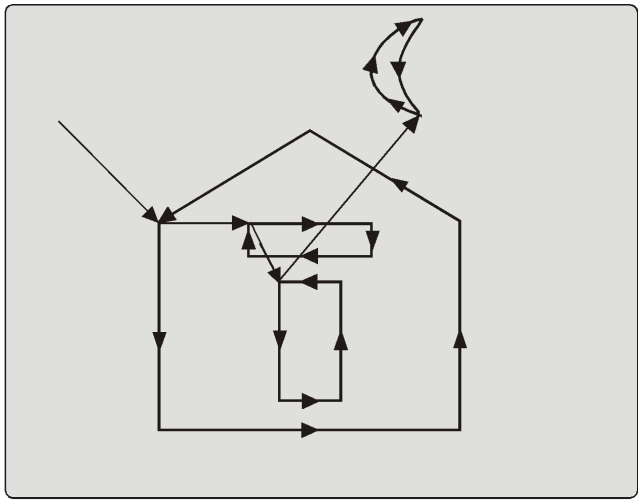
Něco o tvrdém vybavení

Displeje

- první displeje byly *vektorové* displeje
  - Electronový paprsek sledoval čáry
  - obraz definován sekvencí koncových bodů
  - drátové zobrazení, bez vypňování



(a) Ideal line drawing



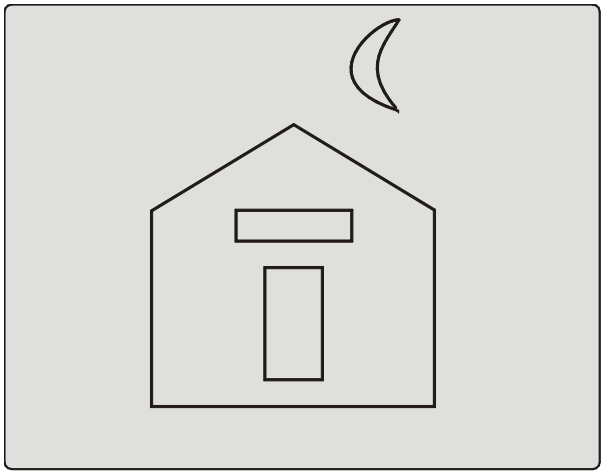
(b) Random scan

Displeje

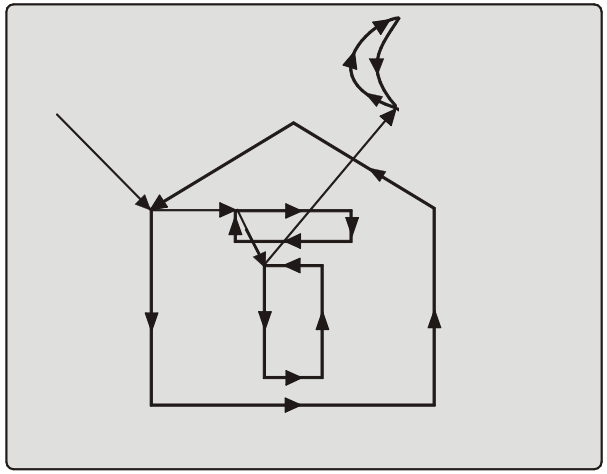
- Rastrové displeje
  - elektronový paprsek prochází po pravidelné dráze
  - obraz je 2D pole pixelů
  - rychlé, chyby vzorkování

- Každý pixel má b bitů pro barvu
  - B&W: 1 bit
  - Základní barvy: 8, 15, 16, 24 bits
  - špičkové: 96 bits

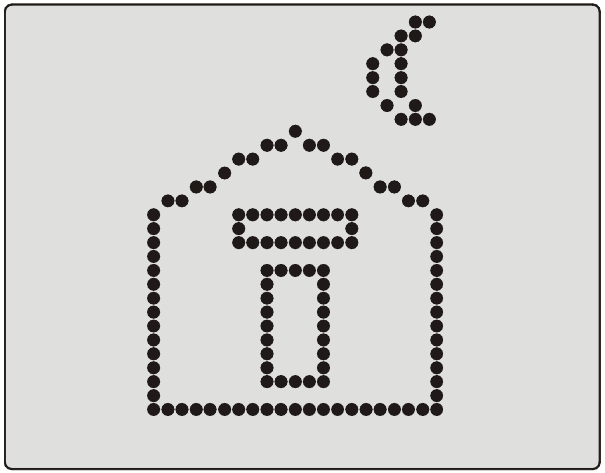
Displeje



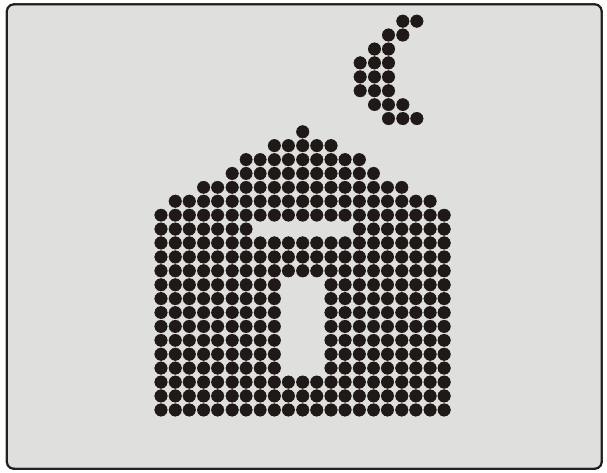
(a) Ideal line drawing



(b) Random scan



(c) Raster scan with outline primitives

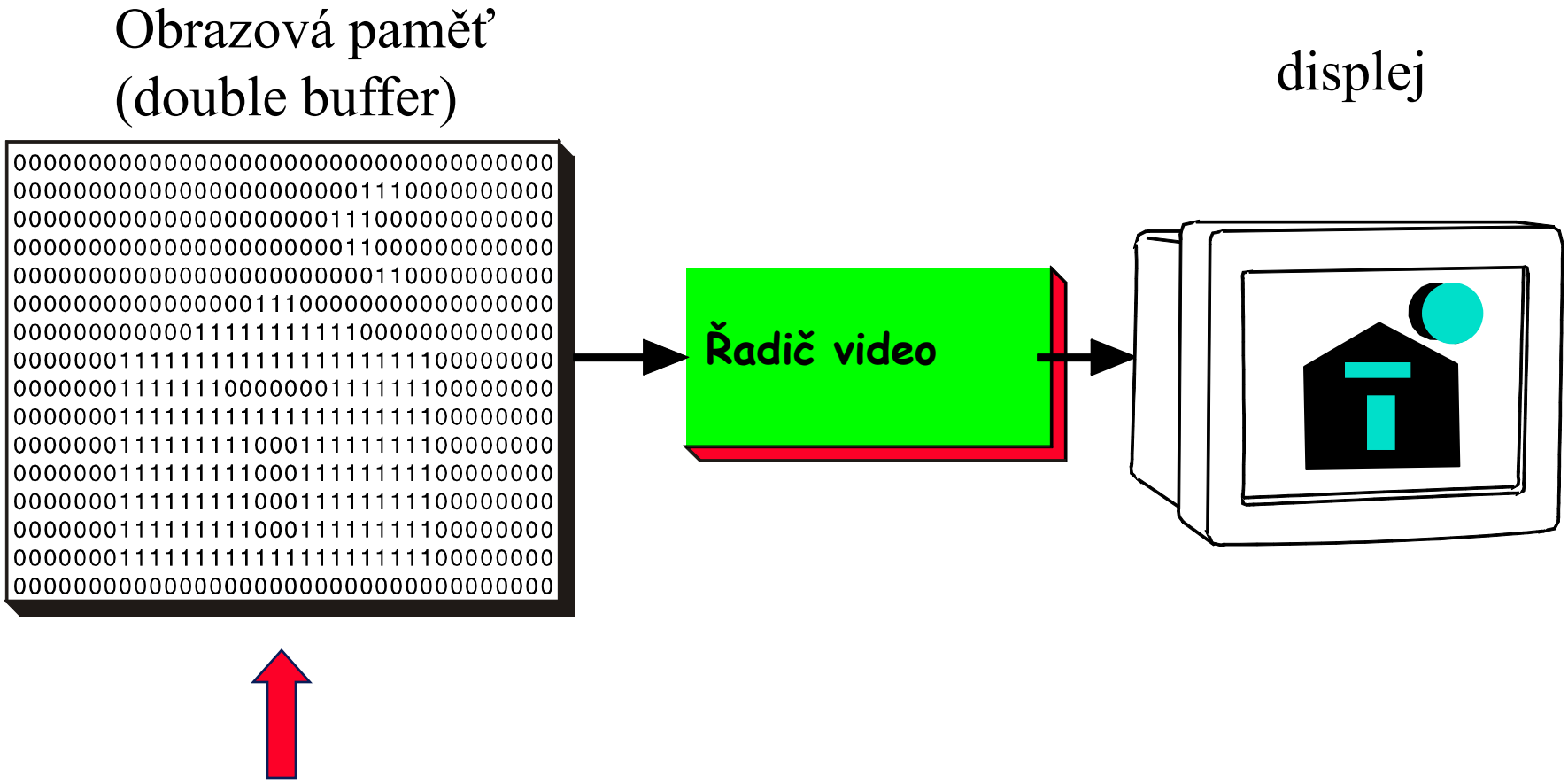


(d) Raster scan with filled primitives

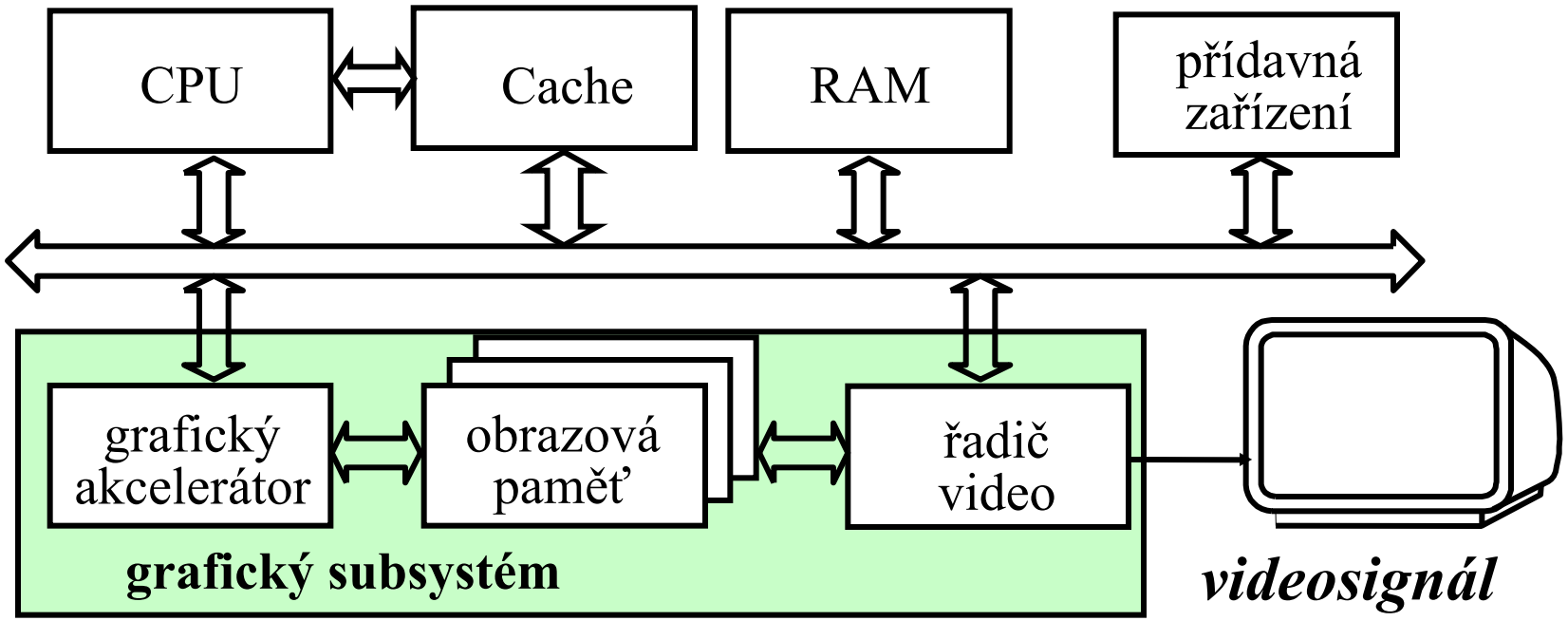
Displeje a obrazové paměti

- rastrový obraz je uložen v paměti jako 2D pole pixelů – obrazová paměť
- barva každého pixelu určuje intenzitu paprsku
- Video hardware čte obrazovou paměť 60+ Hz
  - změny v obrazové paměti se ukazují na obrazovce => dvojitá paměť
  - přepnutí pamětí po dokončení kresby snímku

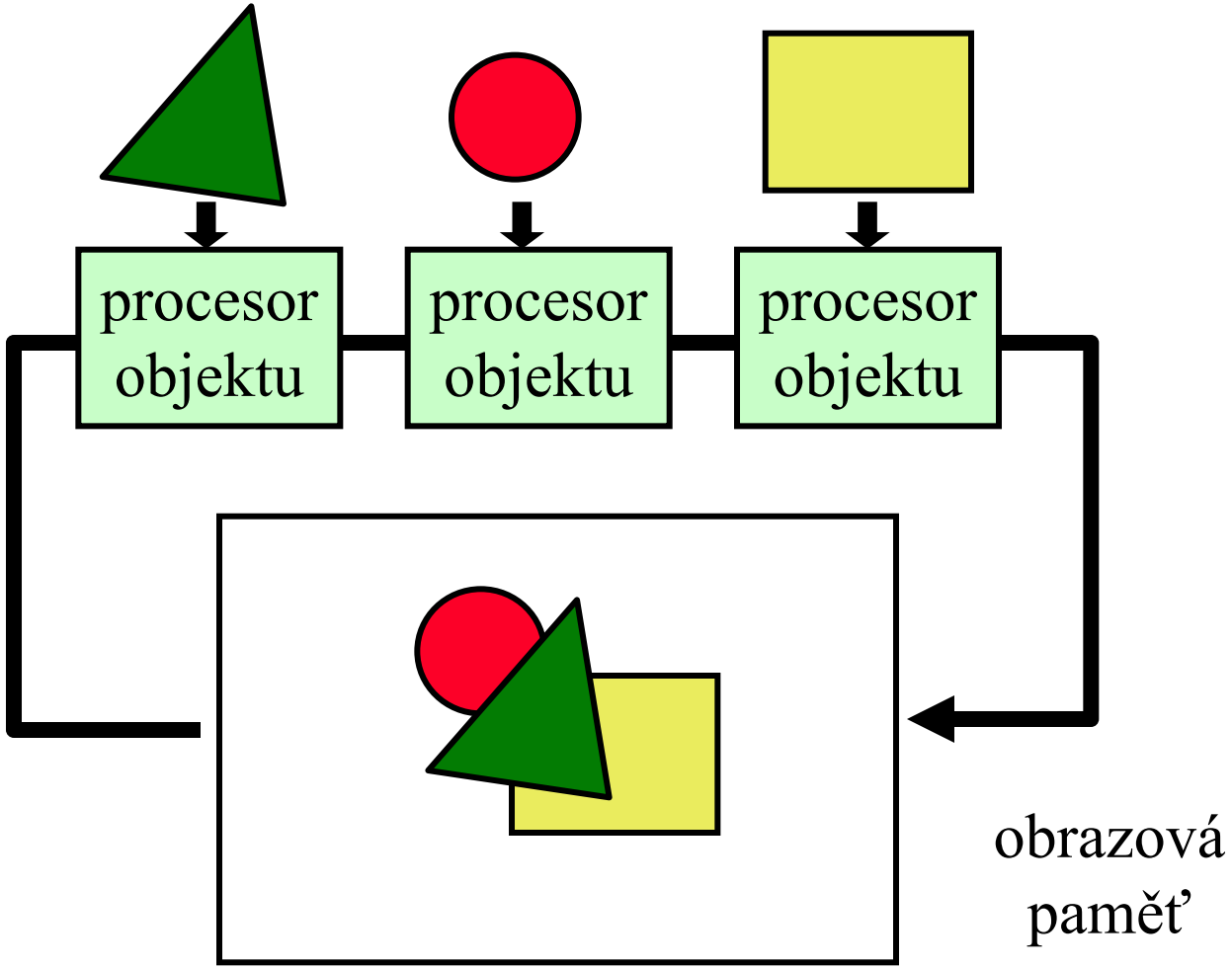
Displeje a obrazové paměti



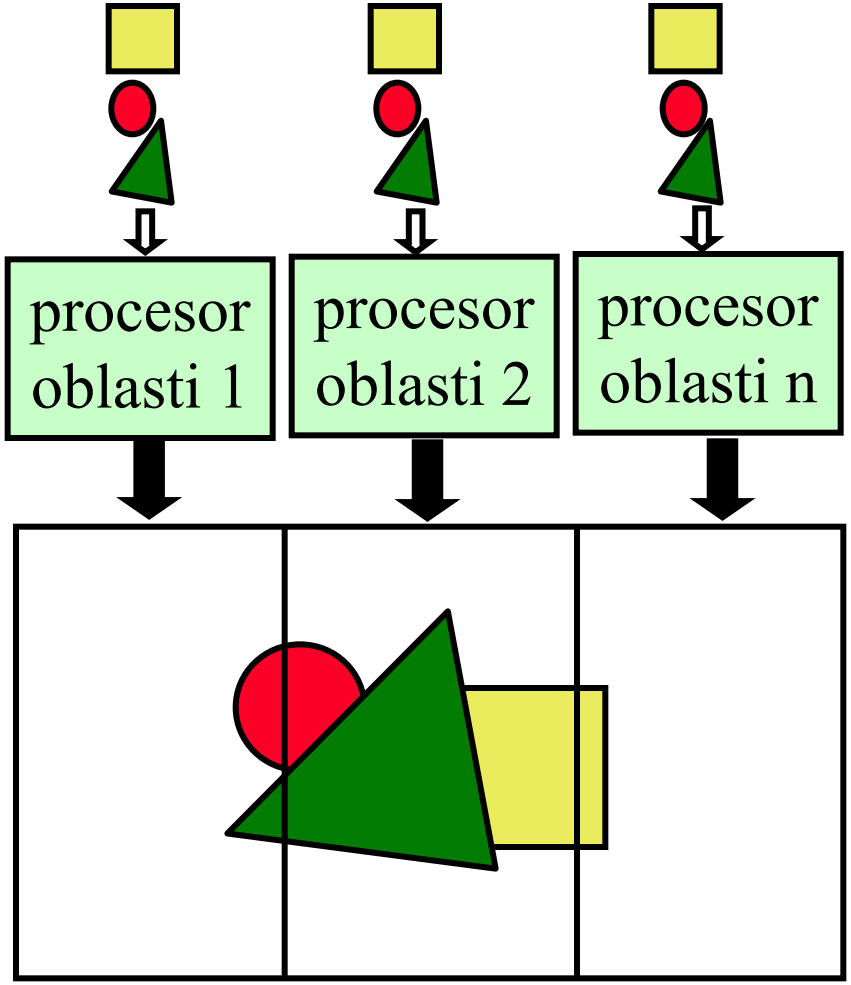
Běžná pracovní stanice



Nové architektury



Nové architektury



Rastrové algoritmy

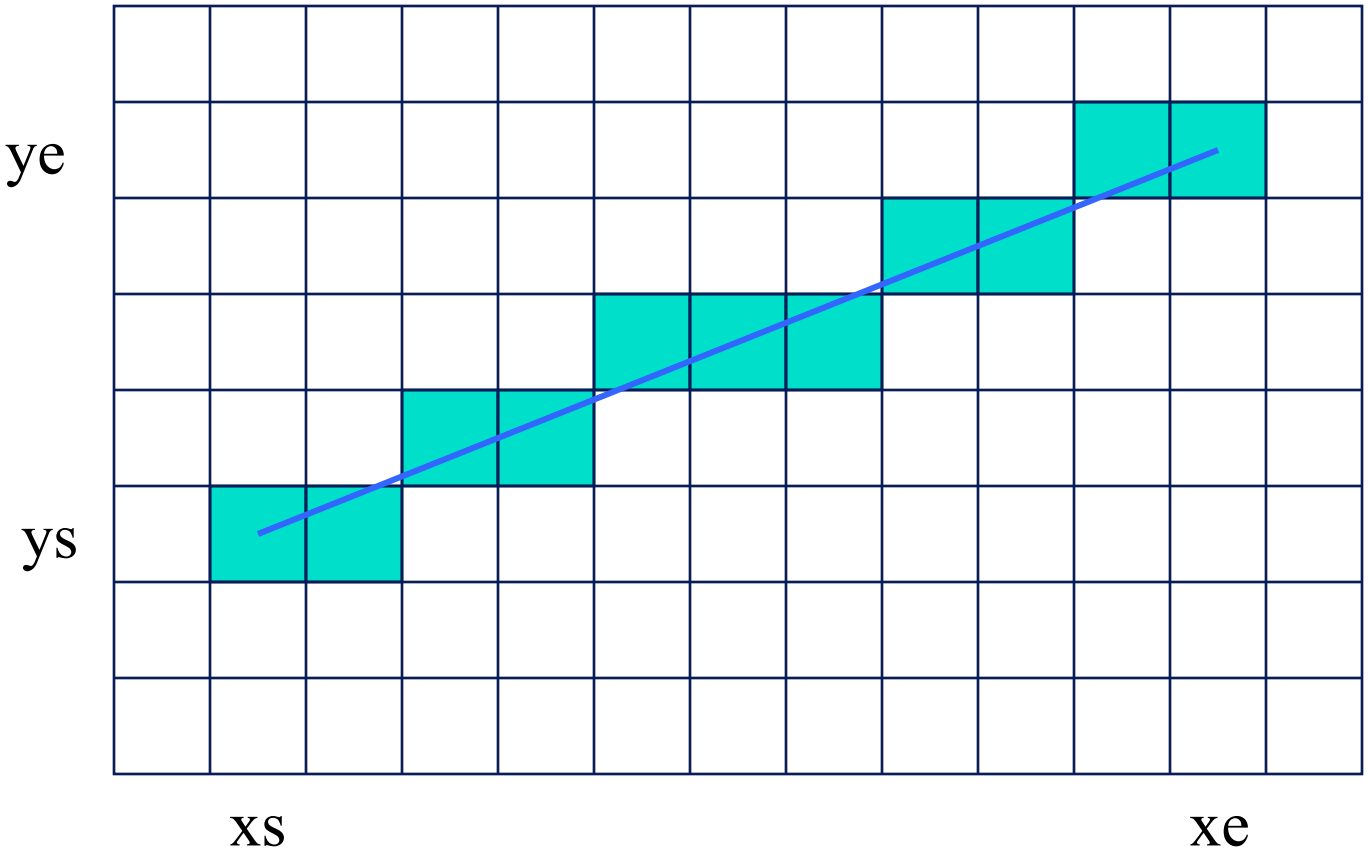
Rastrová konverze úseček

- nalezni pixely nejblíže k ideální přímce

$$y_i = m \cdot x_i + b$$

- předpoklad $|m| \leq 1$:
  vybarvi 1 pixel ve sloupci,
  zpracuj inkrementálně
- if $|m| > 1$: $x \Leftrightarrow y$.

Rastrová konverze úseček



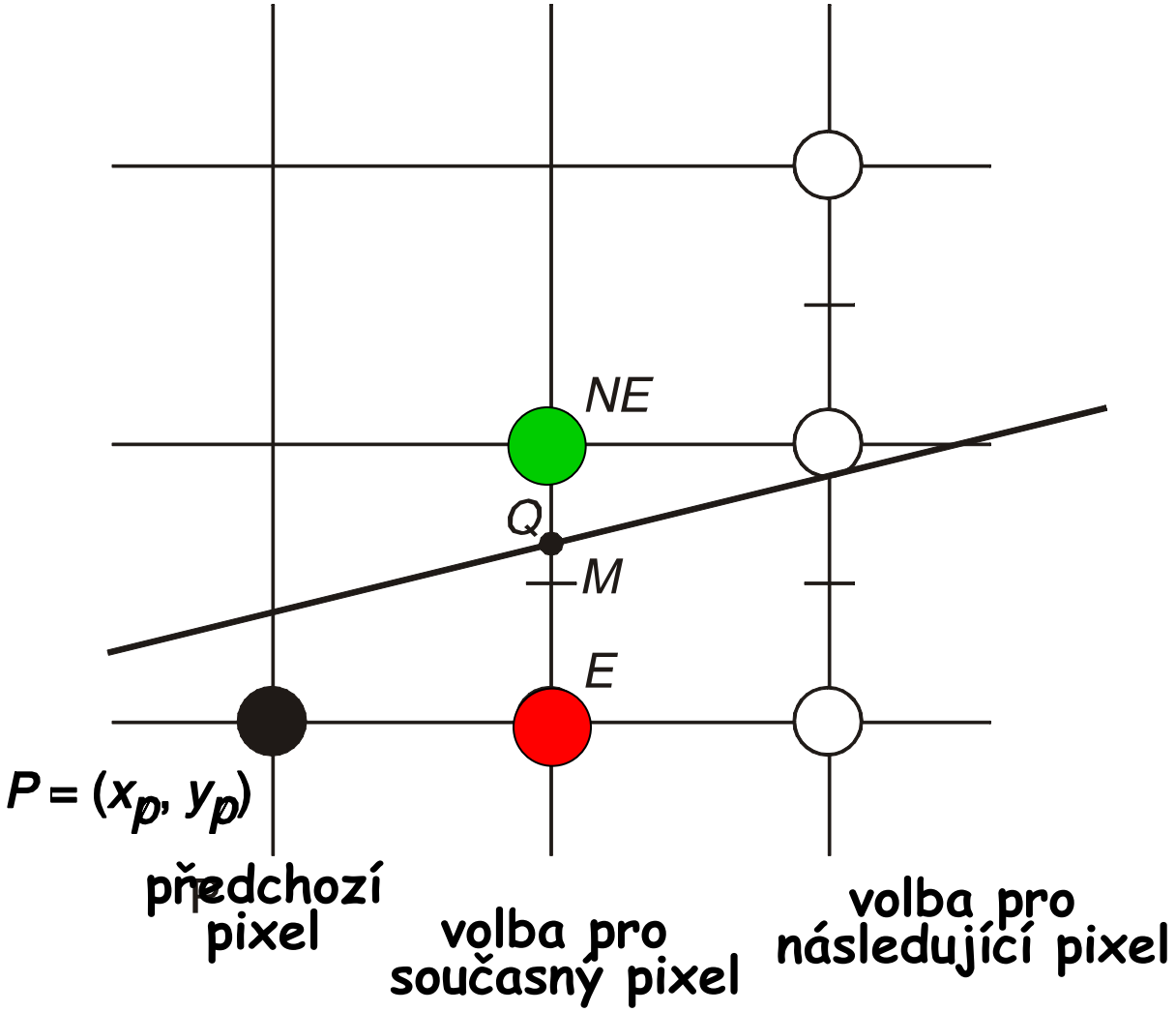
Rastrová konverze úseček

- neefektivní metoda: výpočet round(y) pro každé celé x
- inkrementální výpočet:

$$y_i = mx_i + B$$
$$y_{i+1} = mx_{i+1} + B = m(x_i + 1) + B = y_i + m$$

- pouze celočíselná aritmetika: Bresenhamův algoritmus

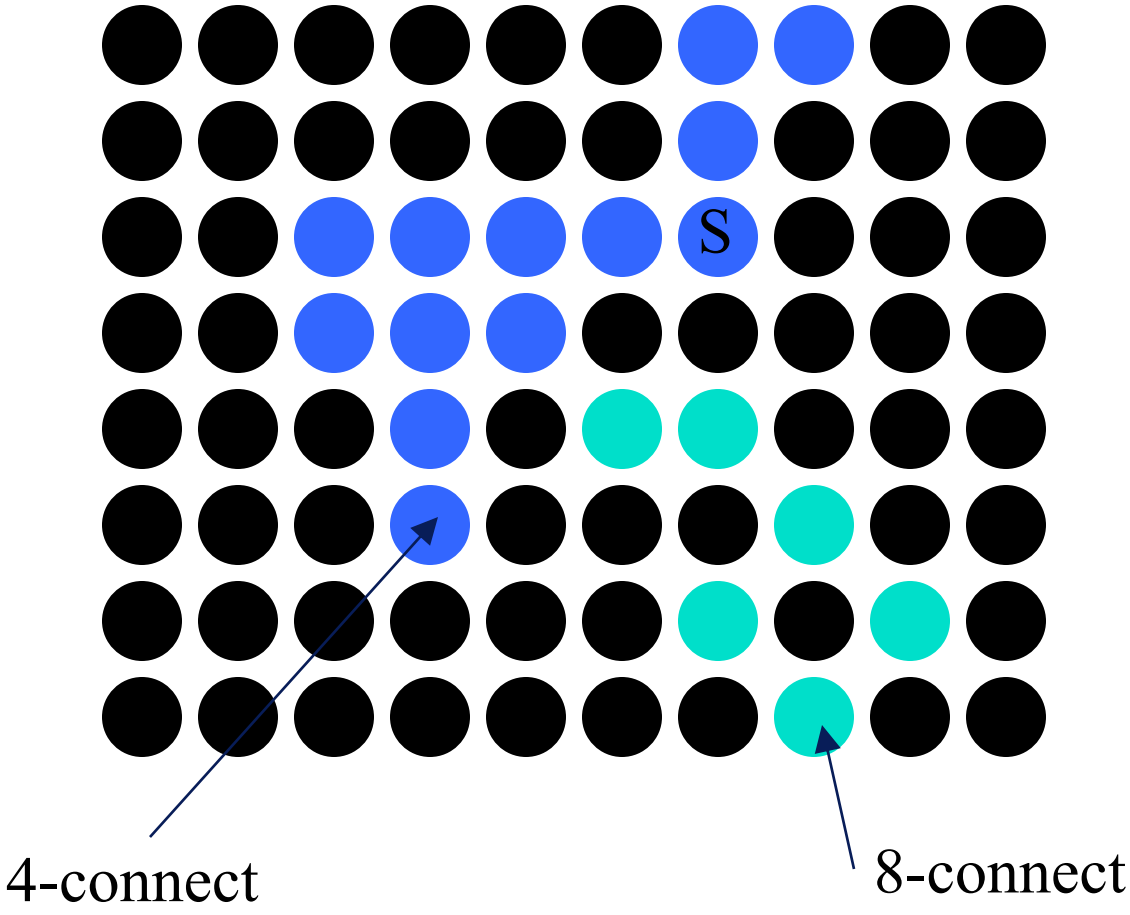
Rastrová konverze úseček



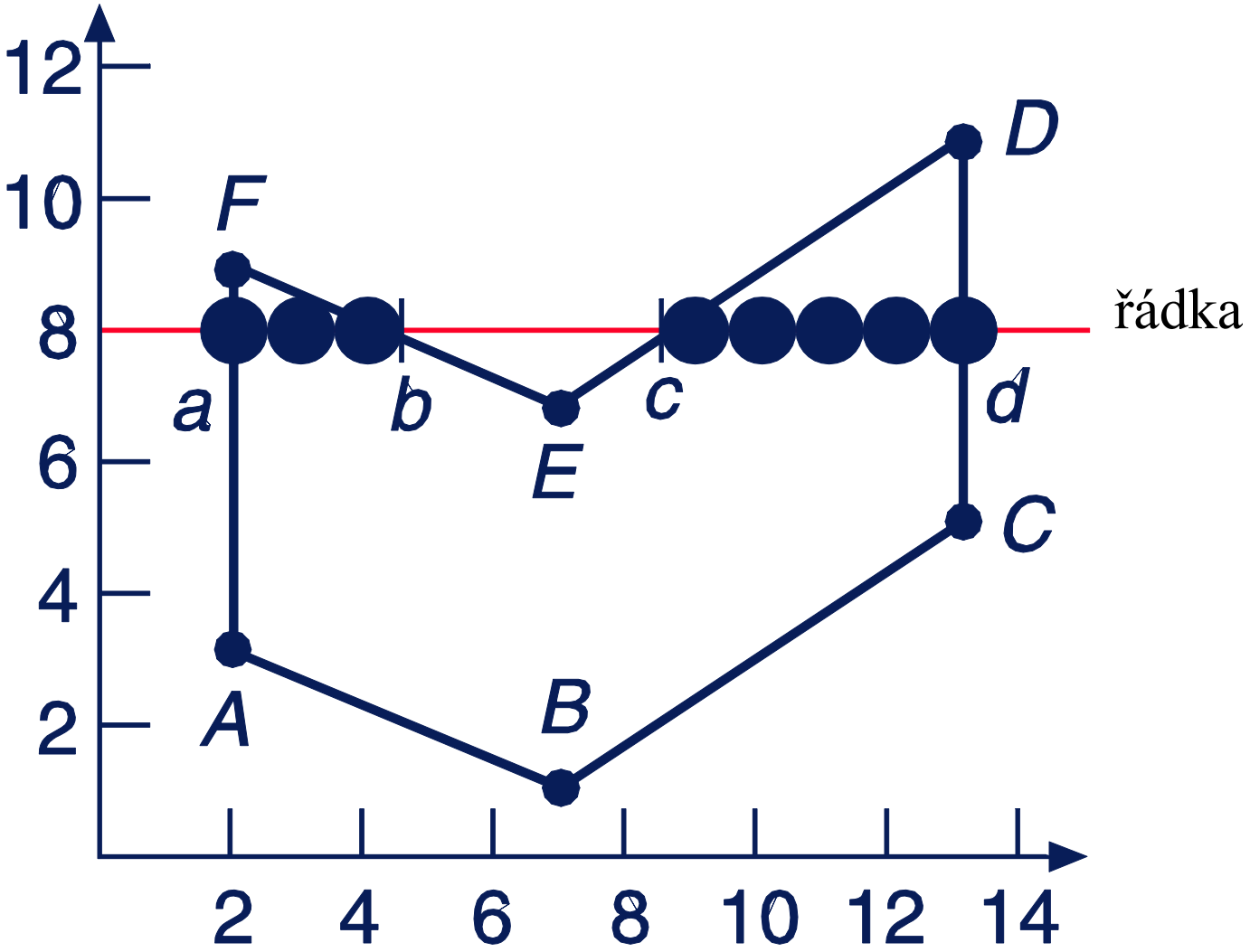
Výplň ploch

- obarvení všech pixelů v dané oblasti
- oblast =
  - všechny pixely určité barvy (pixelově definovaná oblast)
  - všechny pixely v určité vzdálenosti od pixelu
  - všechny pixely uvnitř daného polygonu (oblast definovaná polygonem)

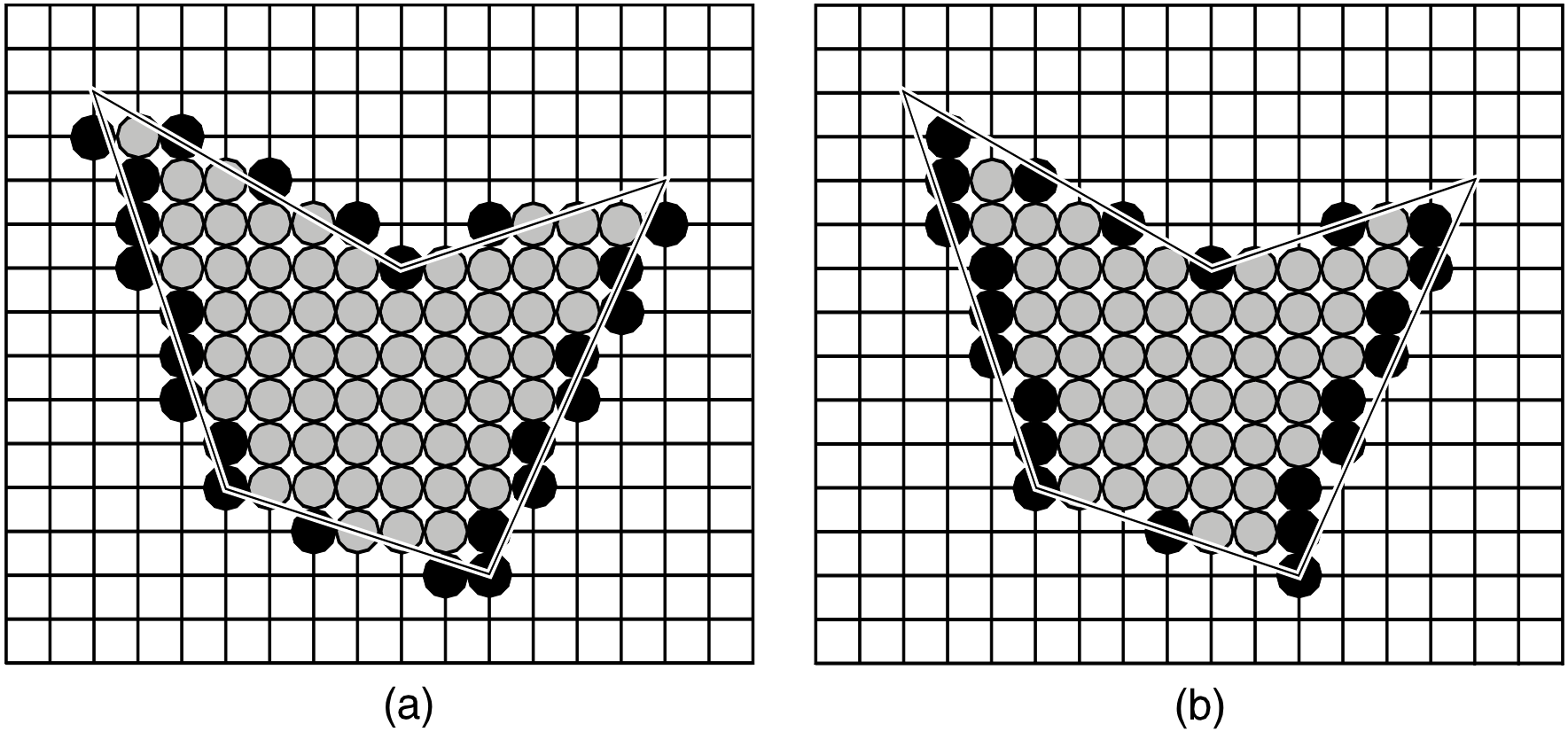
Pixelově definované oblasti



Výplň polygonální oblasti

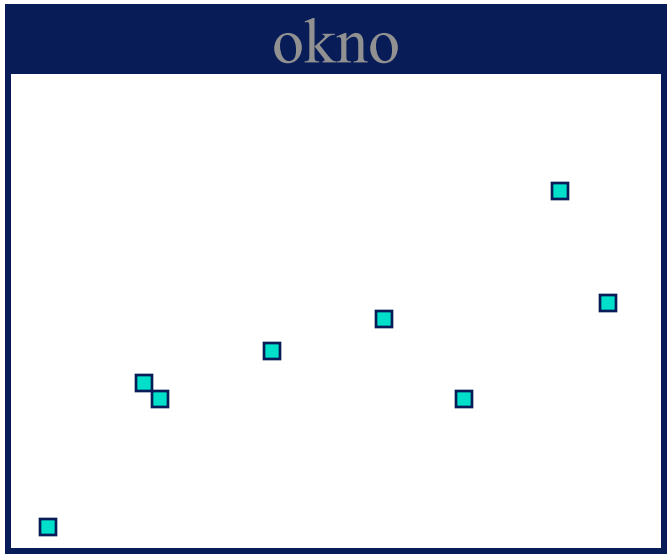


Výplň polygonální oblasti



Souřadné systémy a transformace

Jak něco nakreslit ?

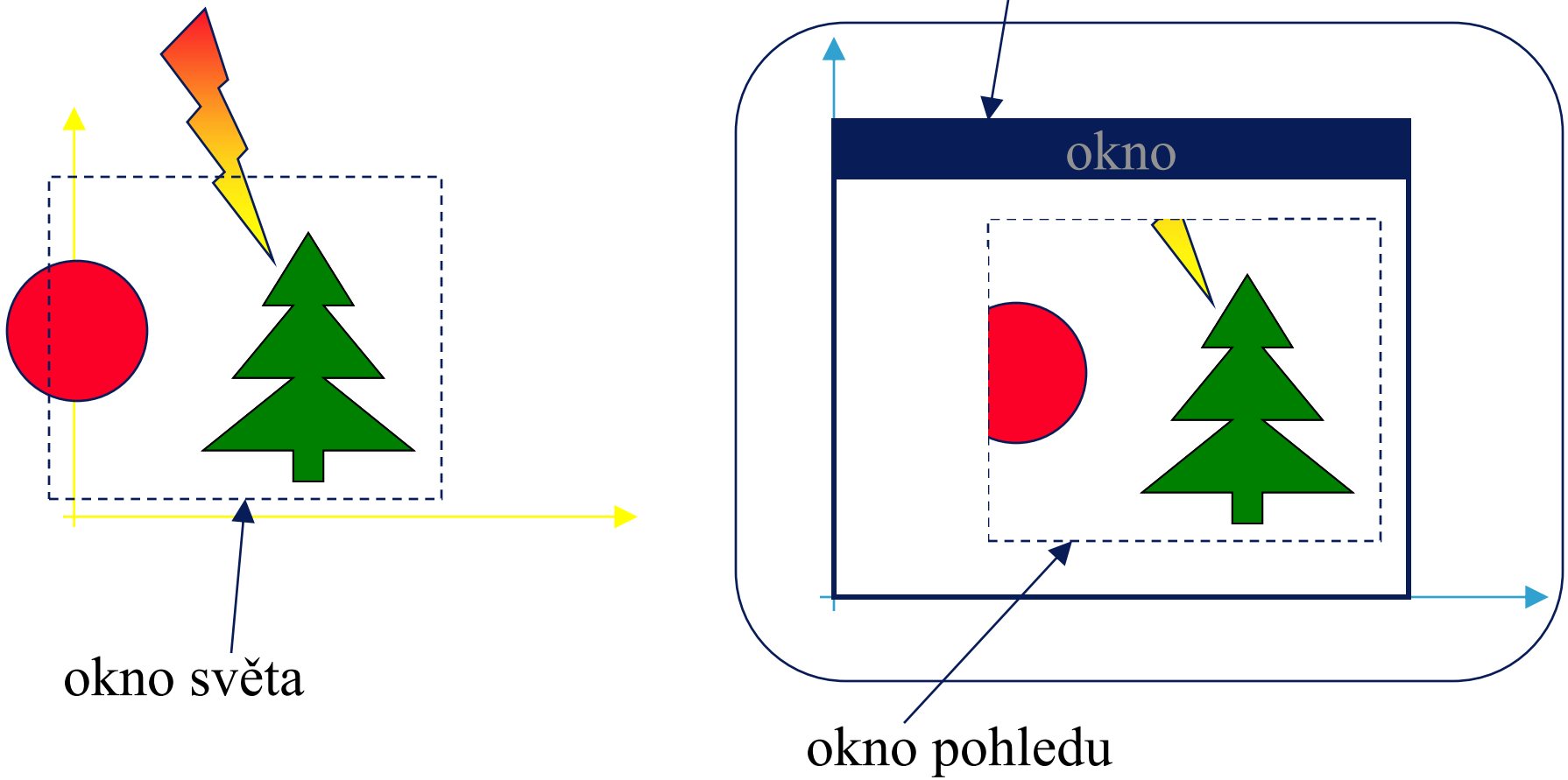


- plotPixel(289,190)
- plotPixel(320,128)
- plotPixel(239,67)
- plotPixel(194,101)
- plotPixel(129,83)
- plotPixel(75,73)
- plotPixel(74,74)
- plotPixel(20,10)

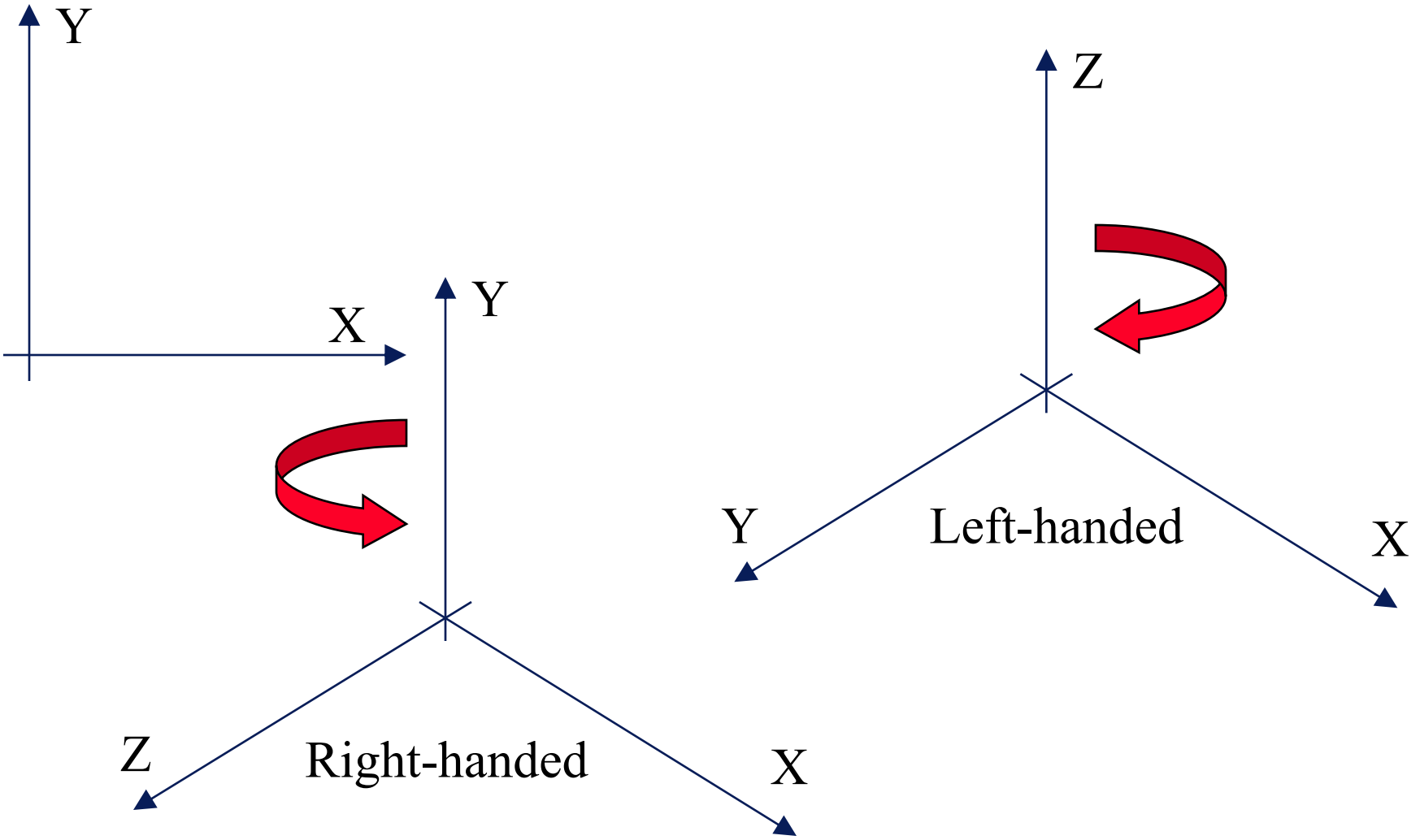
Proč je to nepraktické ?

- Souřadnice jsou vyjádřeny v *prostoru obrazovky*, ale objekty žijí v (3D) ve světovém prostoru
- Při změně velikosti okna musíme změnit souřadnice kreslených objektů
- Chceme rozlišit mezi:
  - hodnotami, které popisují geometrické objekty
  - hodnotami potřebnými pro nakreslení těchto objektů na obrazovku

Okno světa & okno pohledu

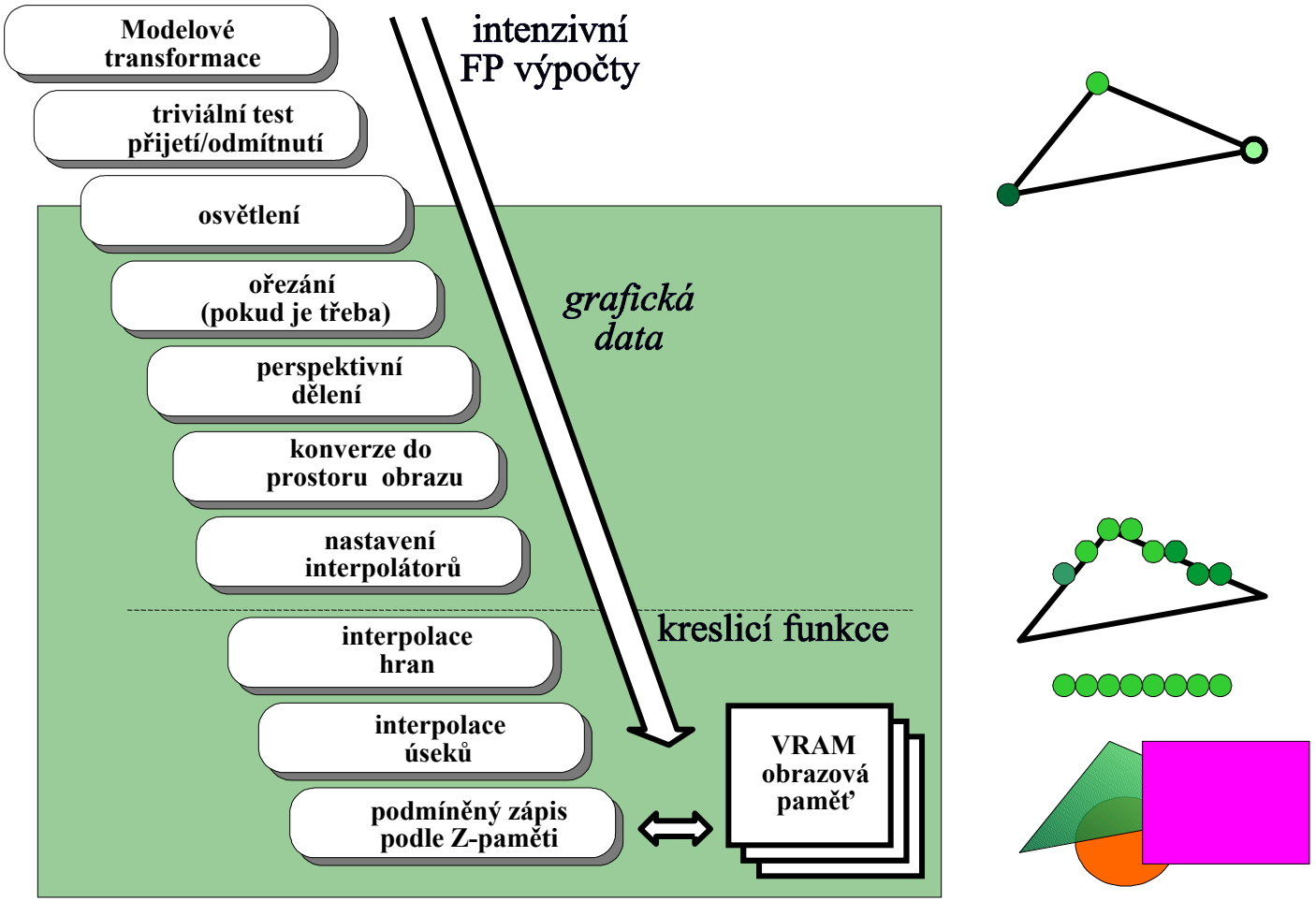


Souřadné systémy: 2D & 3D



OpenGL

3D proudová architektura SW+HW



OpenGL

Co je OpenGL grafický systém?

- je to API (application programming interface)
- softwarový interface pro grafický hardware
- vrstva mezi programátorem a grafickým hardware
- je to “Graphics Assembly Language”

Požadavky na hardware

- musí obsahovat frame buffer, tj. musí být pixelově orientován

Rysy

- nezávisí na hardware
- nemá příkazy pro práci s okny
- OpenGL není pixelově přesné

→ shodná posloupnost příkazů může vytvořit trochu rozdílné obrazy na různých platformách

→ můžeme použít různé algoritmy (float, int)

→ opět, je hardwarově nezávislé

Co dělá GL?

- vykreslování trojúhelníků, čar, polygonů (3D)
- manipulace s rastrovým obrazem (2D)
- texturování
- osvětlování
- stínování
- mlha
- výpočet viditelnosti
- alpha míchání
- transformace
- akumulační paměť
- šablonová paměť (stencil b.)

Co GL nedělá?

- úlohy s okny
- definice objektů
- NURBS (parametrické křivky, plochy)
- stíny
- odrazy
- voxely
- reprezentace scény

Další pohled na OpenGL

- OpenGL je stavový automat
- tj., můžeme nastavit stav, nebo se na něj dotázat

- nastavení stavu: **`glEnable(), glDisable(), etc.`**
- zjištění stavu: **`glGetSomething()`**

Jednoduchý program

```
void main(){
  OpenWindow();
  InitOpenGL();              // nastav GL stav
  glDisable(GL_LIGHTING); // nastav GL stav
  glBegin(GL_TRIANGLES);  // nastav GL stav
  glShadeMode(GL_SMOOTH); // nastav GL stav
  glColor3f(1.0,0.0,0.0);glVertex3f(0.0, 0.0, 0.0);
  glColor3f(0.0,1.0,0.0);glVertex3f(1.0, 0.0, 0.0);
  glColor3f(0.0,0.0,1.0);glVertex3f(1.0, 1.0, 0.0);
  glEnd();                   // nastav GL stav
  Wait4Key();// čekej na vstup z  klávesnice
  CloseWindow();}
```

Knihovny související s OpenGL

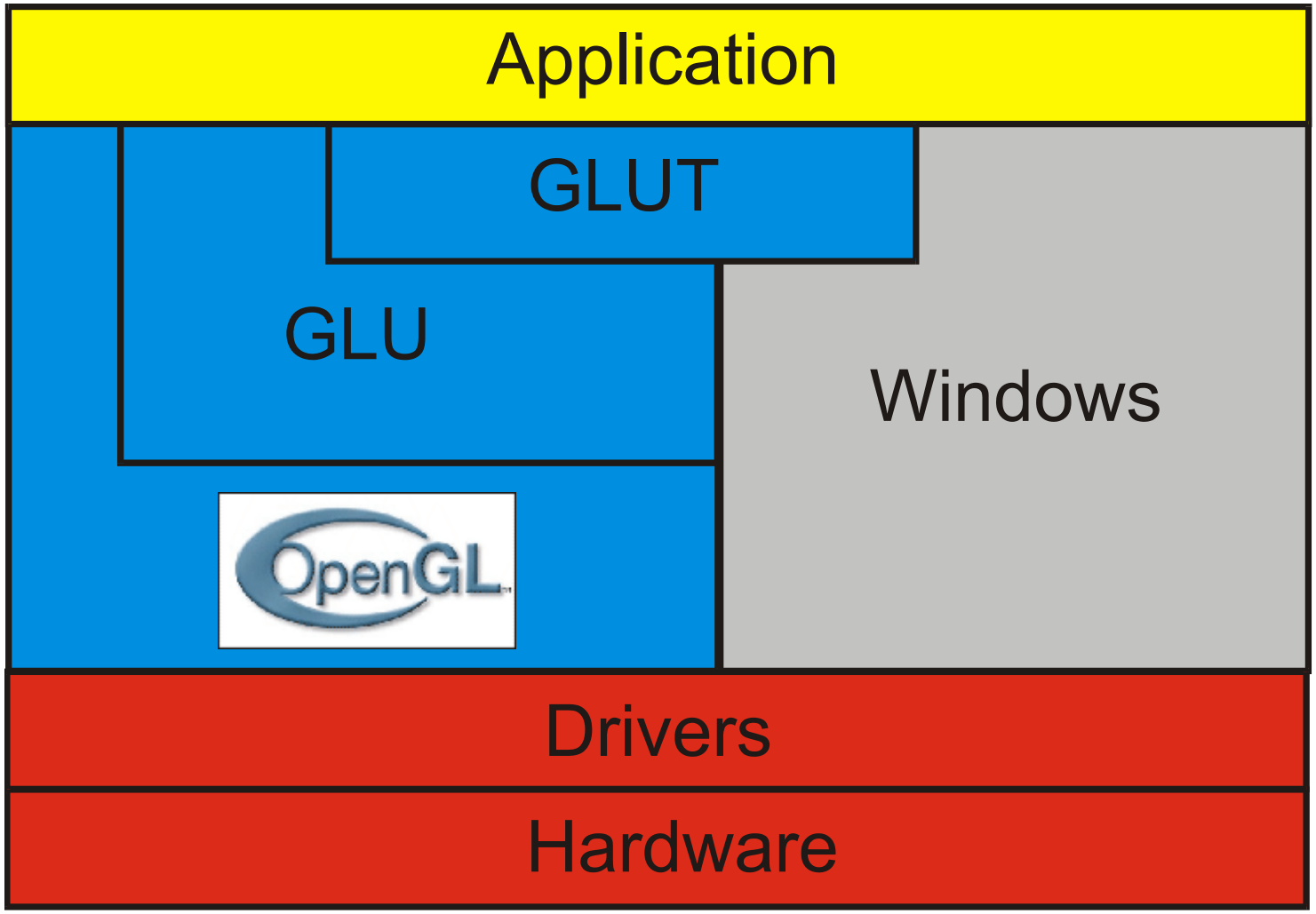
- aux.h pomocná knihovna (SGI)
  - basic window manipulation
  - simple objects (box, sphere, disc)
  - obsolete
- glu.h OpenGL Utility Library (SGI)
  - velmi užitečná!
  - práce s obrazem (měřítko, mip-mapování)
  - transformace souřadnic
  - základní objekty
  - Non Uniform Rational B-Splines (NURBS)
  - jednoduché (jednodušší) operace

    `gluLookAt(),gluOrtho2D()`

Knihovny související s OpenGL

- glut.h OpenGL Utility Toolkit (Mark Kilgard)
  - velmi užitečná!
  - manipulace s okny nezávislá na platformě
  - jednoduché menu
  - vyhlazování (antialiasing)
  - stereo zobrazování (pokud je podpořeno hw)
  - zobecněné válce
  - vytažené objekty
  - atd.

OpenGL a související API



Paměti

OpenGL používá několik pamětí

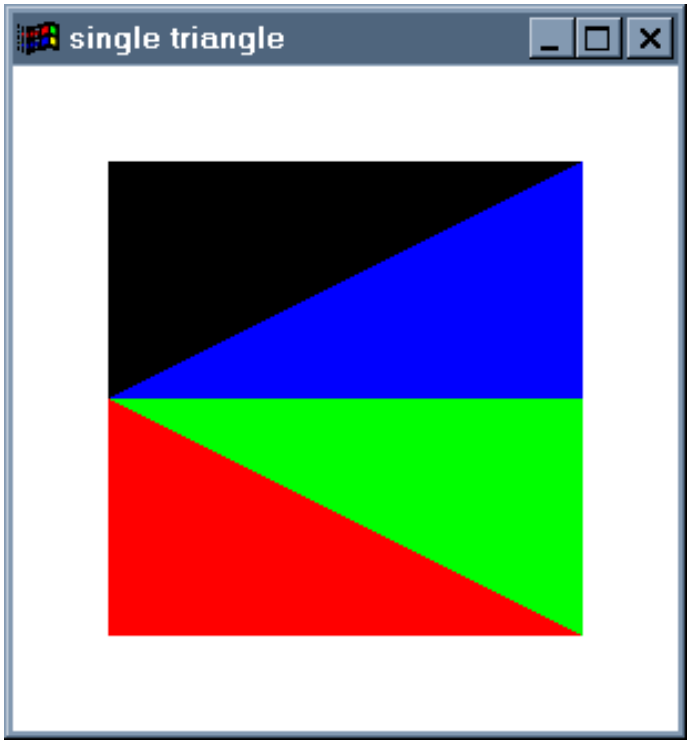
- obrazová paměť	složená z pixelů (obrazovka)
- z-paměť (paměť hloubky)	hloubka pixelu
- alfa paměť	průhlednost pixelu
- akumulační paměť	pohybové rozmazání, aliasing
- indexová paměť	barevný mód
- paměť šablony	popisuje šablonu

Některé hodnoty lze nastavit přímo.

Některé se nastaví při kreslení.

Kresba trojúhelníků

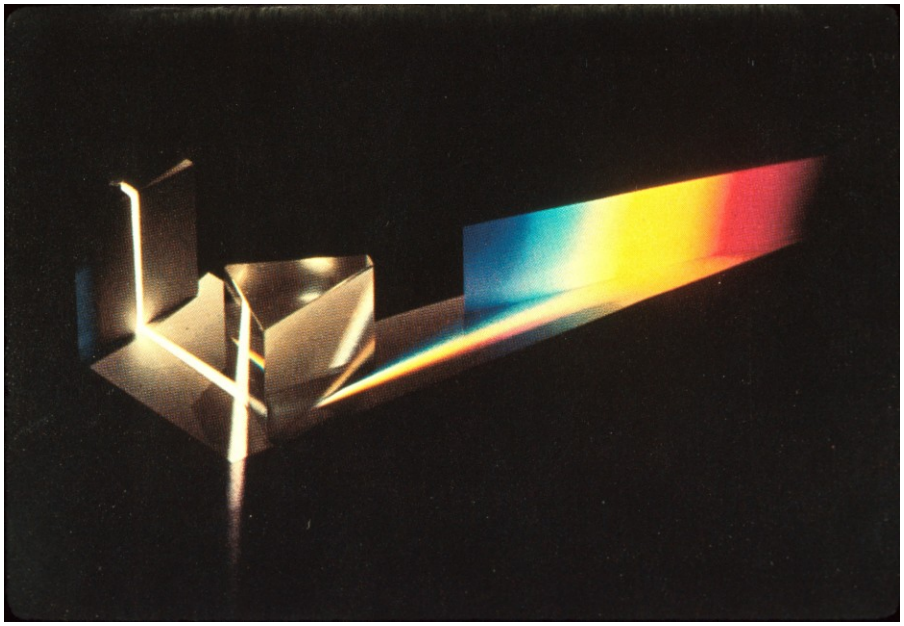
```
glBegin(GL_TRIANGLE_FAN);
glColor3f(1.0,0.0,0.0);
glVertex2i(0, 5);
glVertex2i(0, 0);
glVertex2i(10, 0);
glColor3f(0.0,1.0,0.0);
glVertex2i(10, 5);
glColor3f(0.0,0.0,1.0);
glVertex2i(10, 10);
glColor3f(0.0,0.0,0.0);
glVertex2i(0, 10);
glEnd();
```

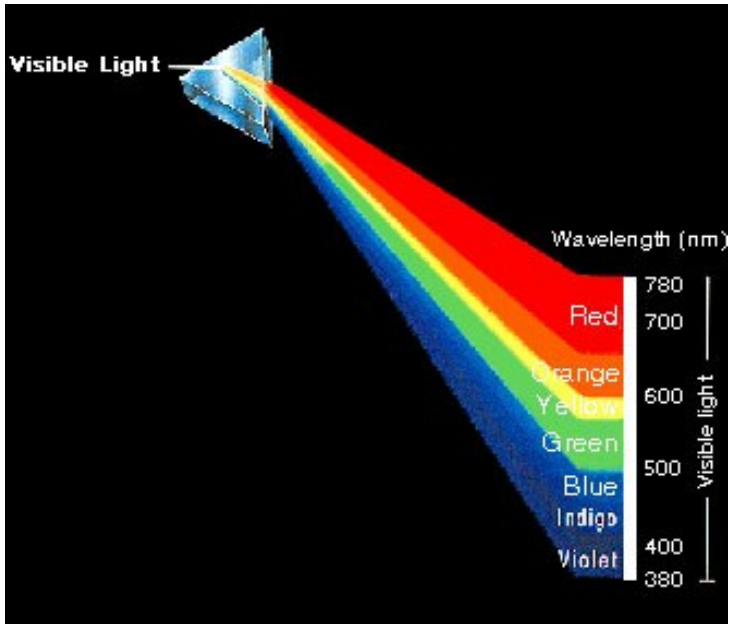


Barva

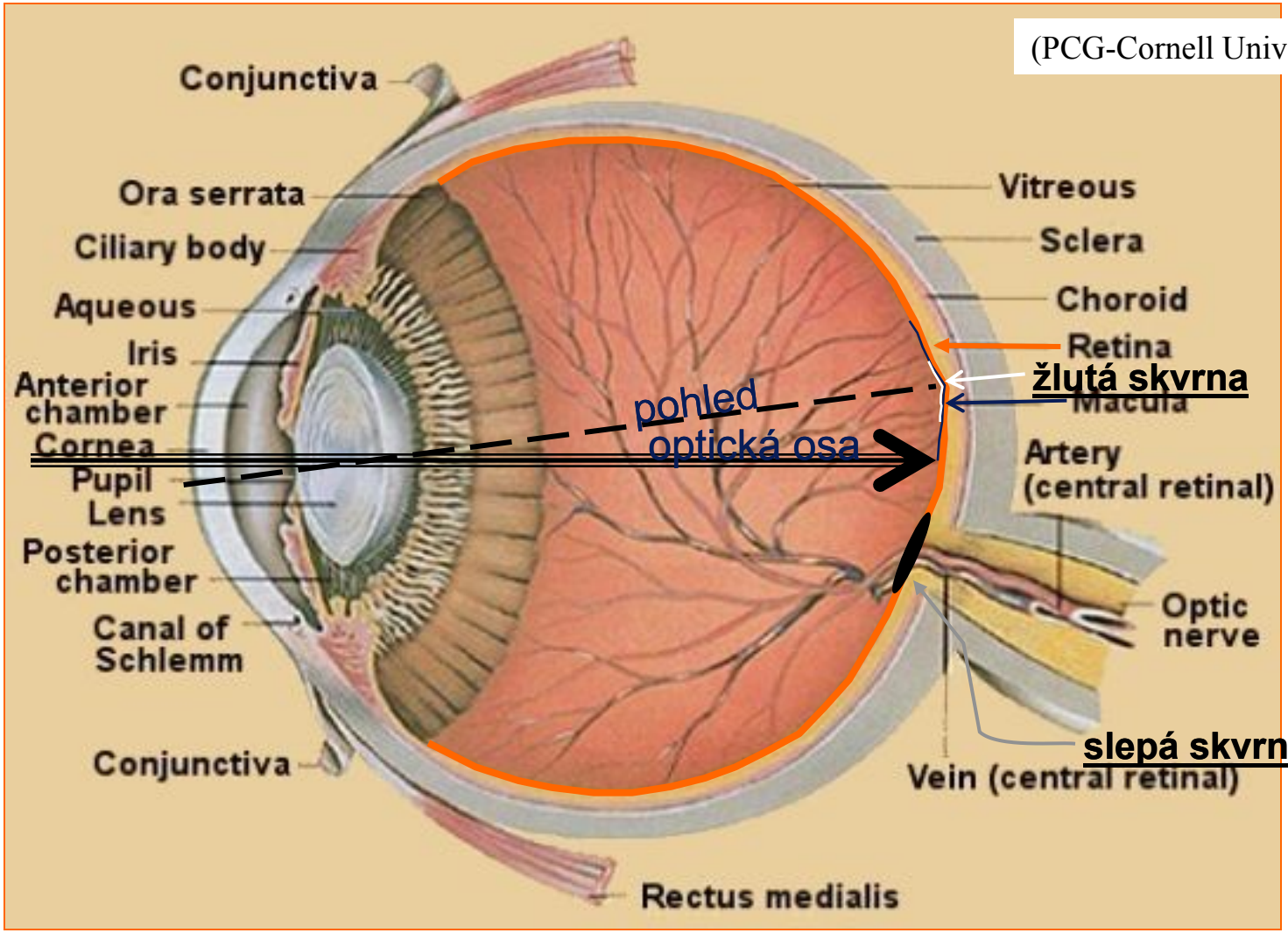
Co je barva?

- Světlo = elektromagnetické vlny
- viditelné spektrum
  - 400 nm (fialová) → 700 nm (červená)

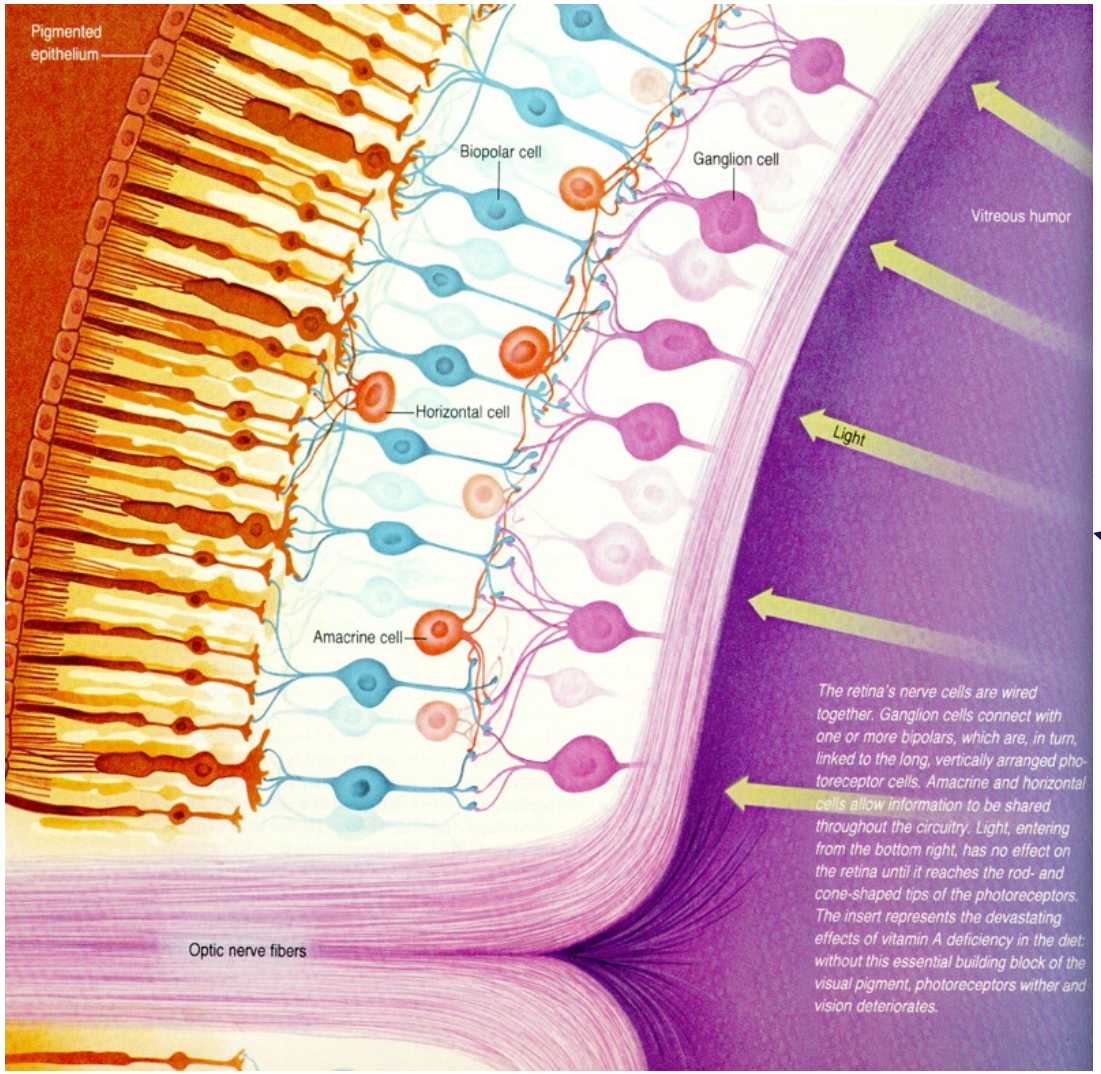




Lidské oko



Lidské oko: sítnice

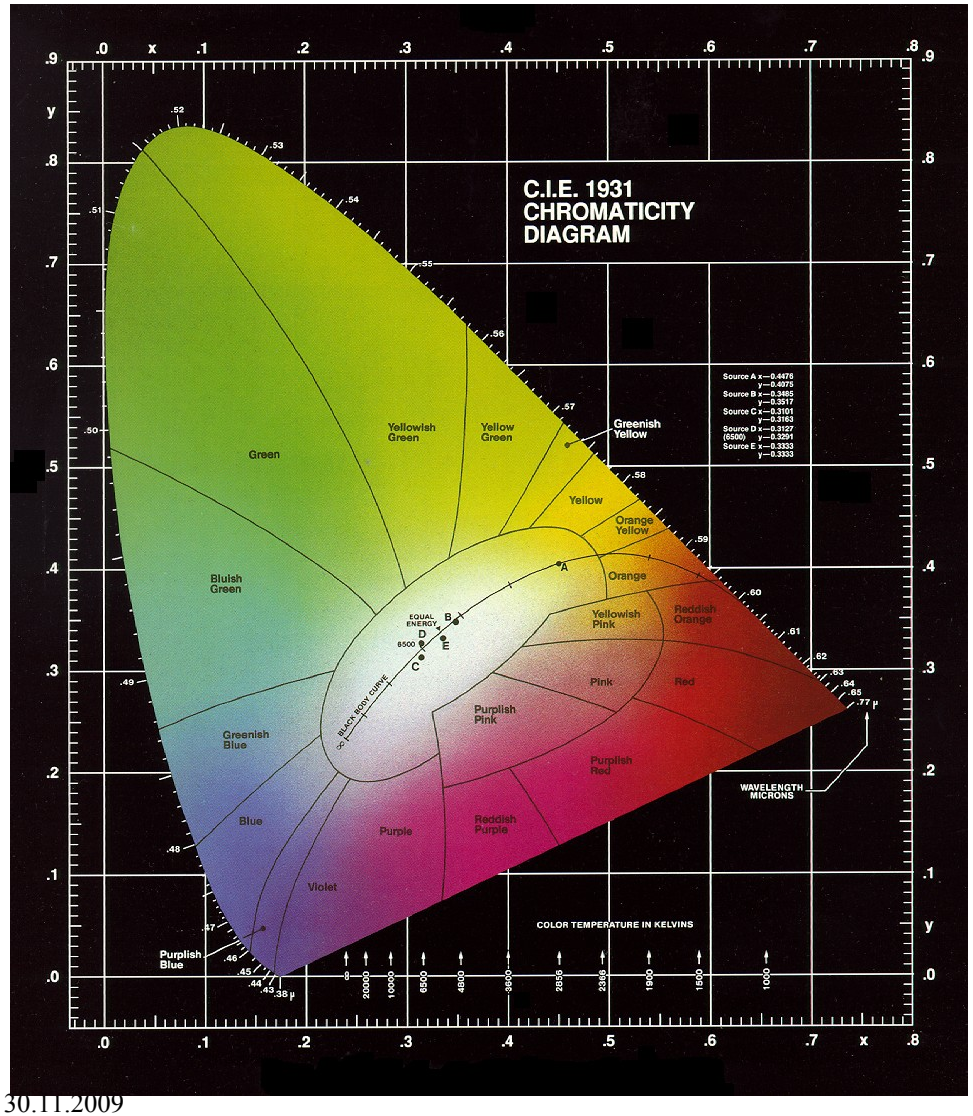


(PCG-Cornell University)

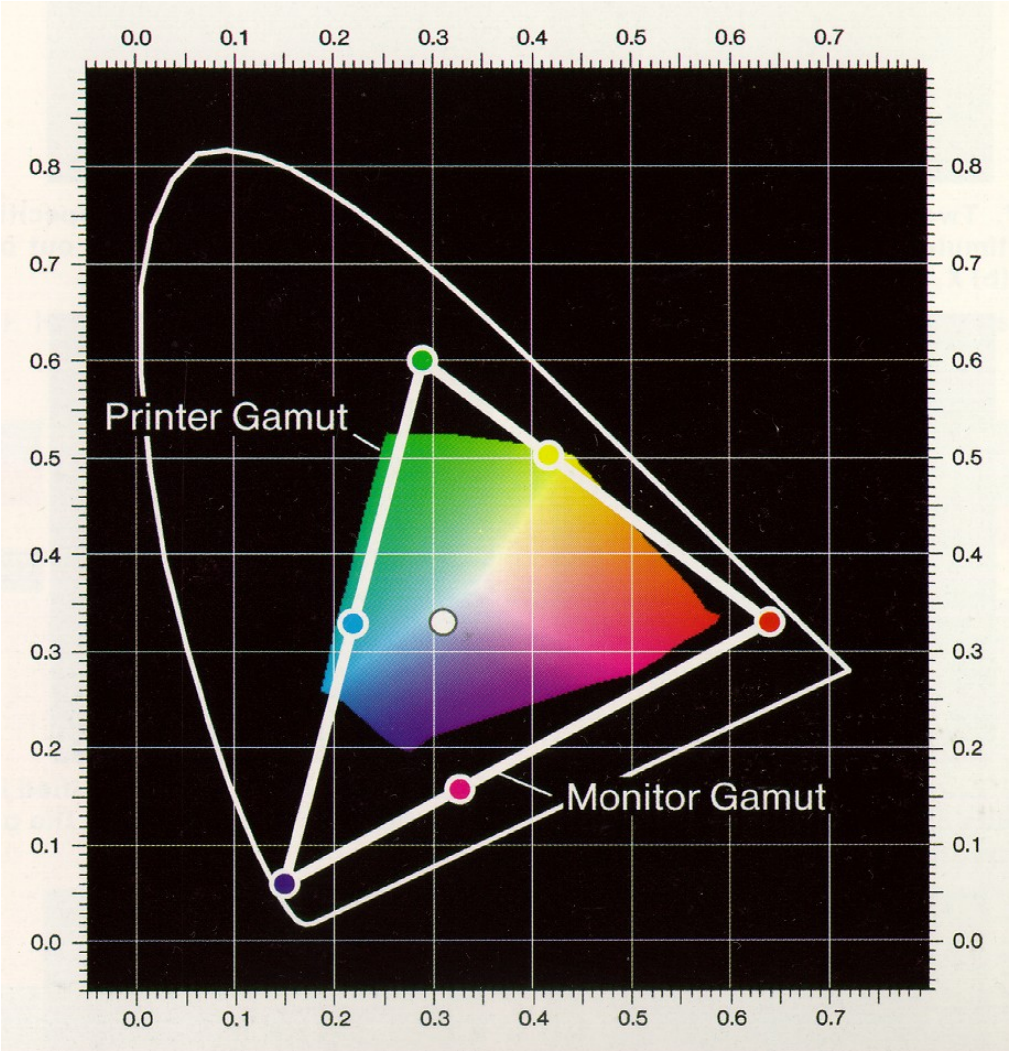
světlo

prochází krevním řečištěm a vrstvami sítnice před dosažením čípků a tyčinek

CIE diagram barev



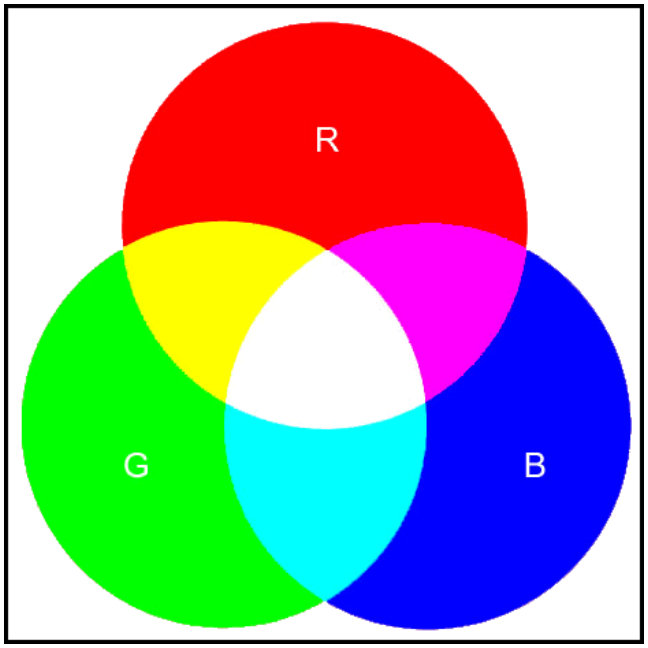
Gamut (rozsah) barevného monitoru



- Ne všechny barvy lze zobrazit na CRT monitoru
- barevné souřadnice každého luminoforu se u jednotlivých monitorů liší

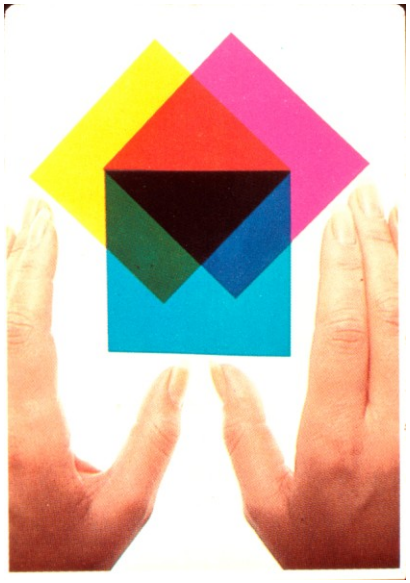
RGB

- nejběžnější v grafice: přímé mapování na CRT
- aditivní barevný systém



CMY (cyan-magenta-yellow)

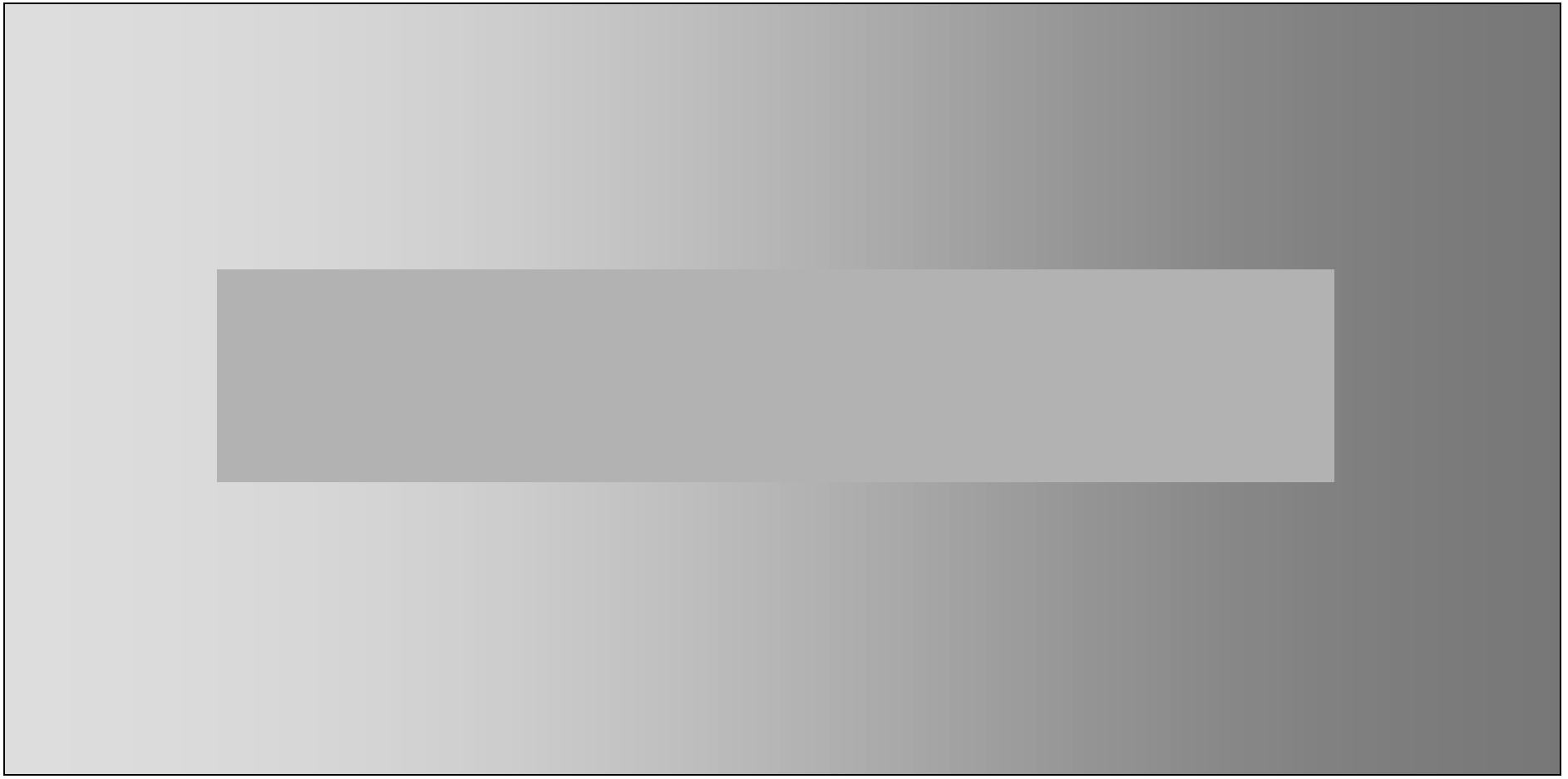
- subtraktivní barevný systém
  - pro získání výsledné barvy odečítá barvy od bílé
  - (r g b) = (1 1 1) – (c m y)



v systému CMY

v systému RGB

Barevné iluze

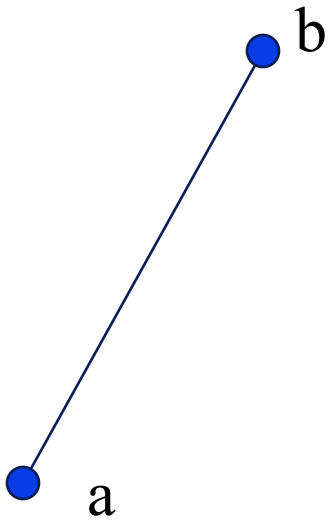


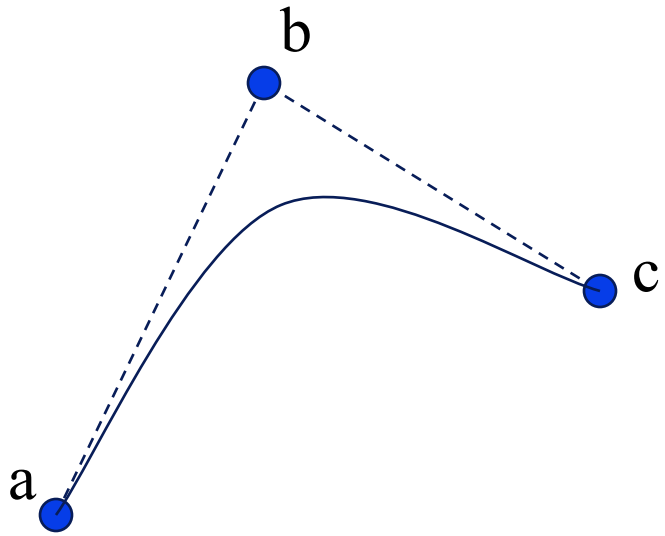
Barevné iluze

Modely a modelování

Parametrické křivky a plochy

- Bezierova křivka



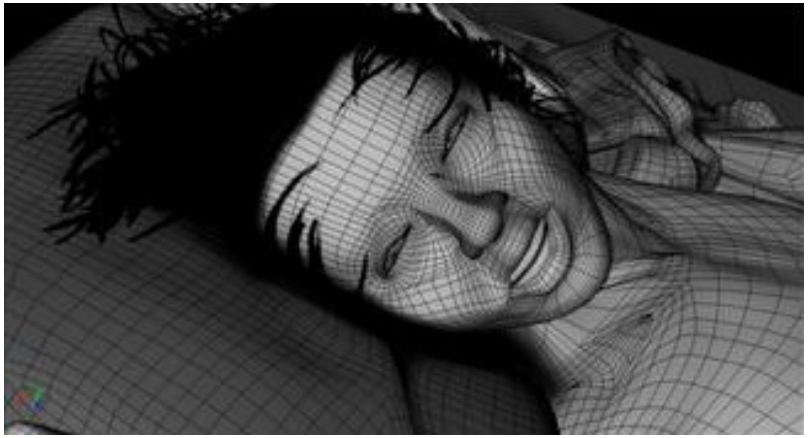


$$p(t) = (1-t)\mathbf{a} + t\mathbf{b}$$

$$p(t) = (1-t)^2\mathbf{a} + 2t(1-t)\mathbf{b} + t^2\mathbf{c}$$

Polygony

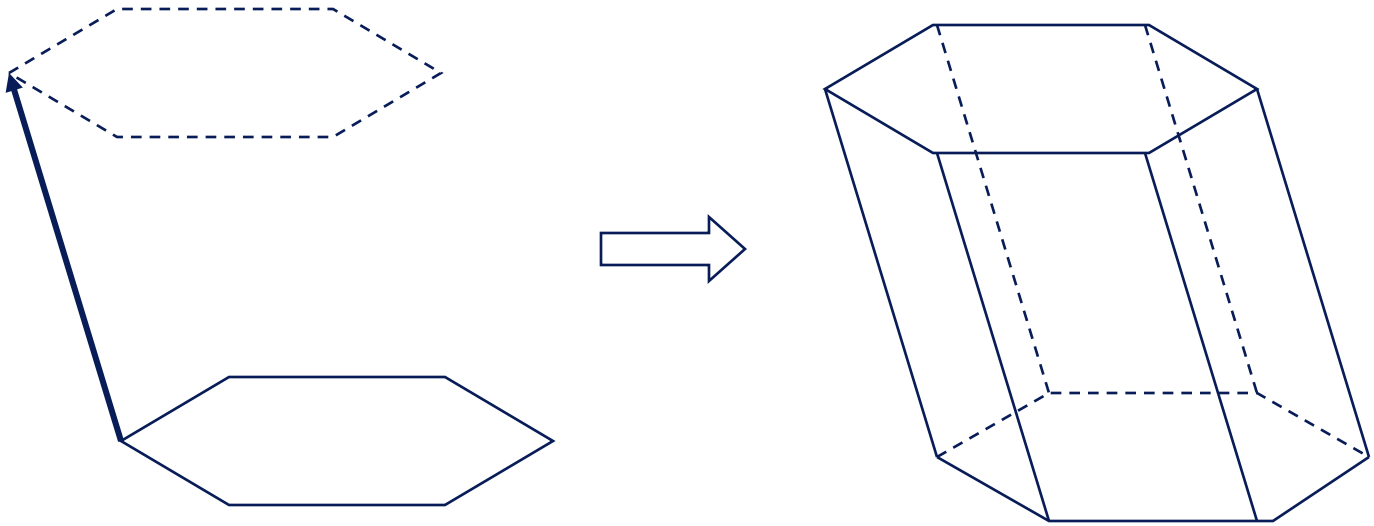
- Objekty ve 3D jsou zhotoveny z polygonů



- Polygony jsou základním stavebním blokem v grafice !

Tažené (extrudované) povrchy

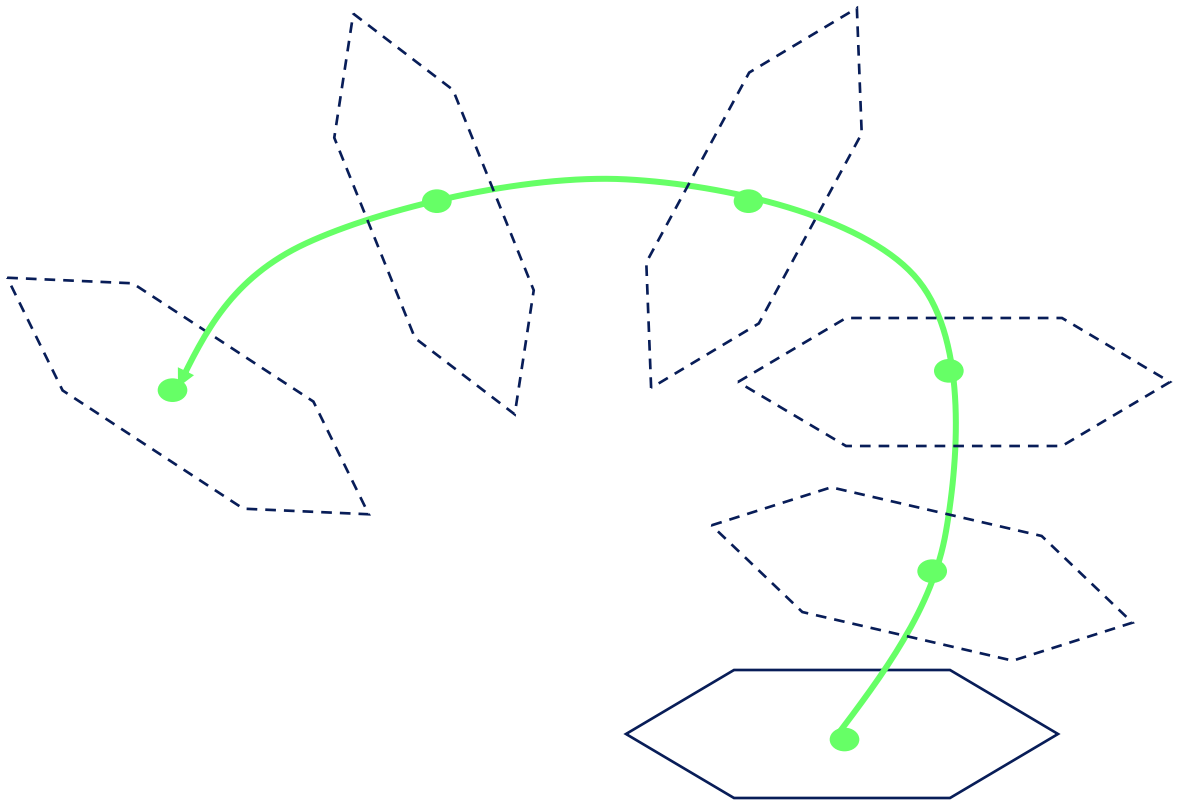
- Hranol



- Povrchová síť trojúhelníků sestrojena automaticky

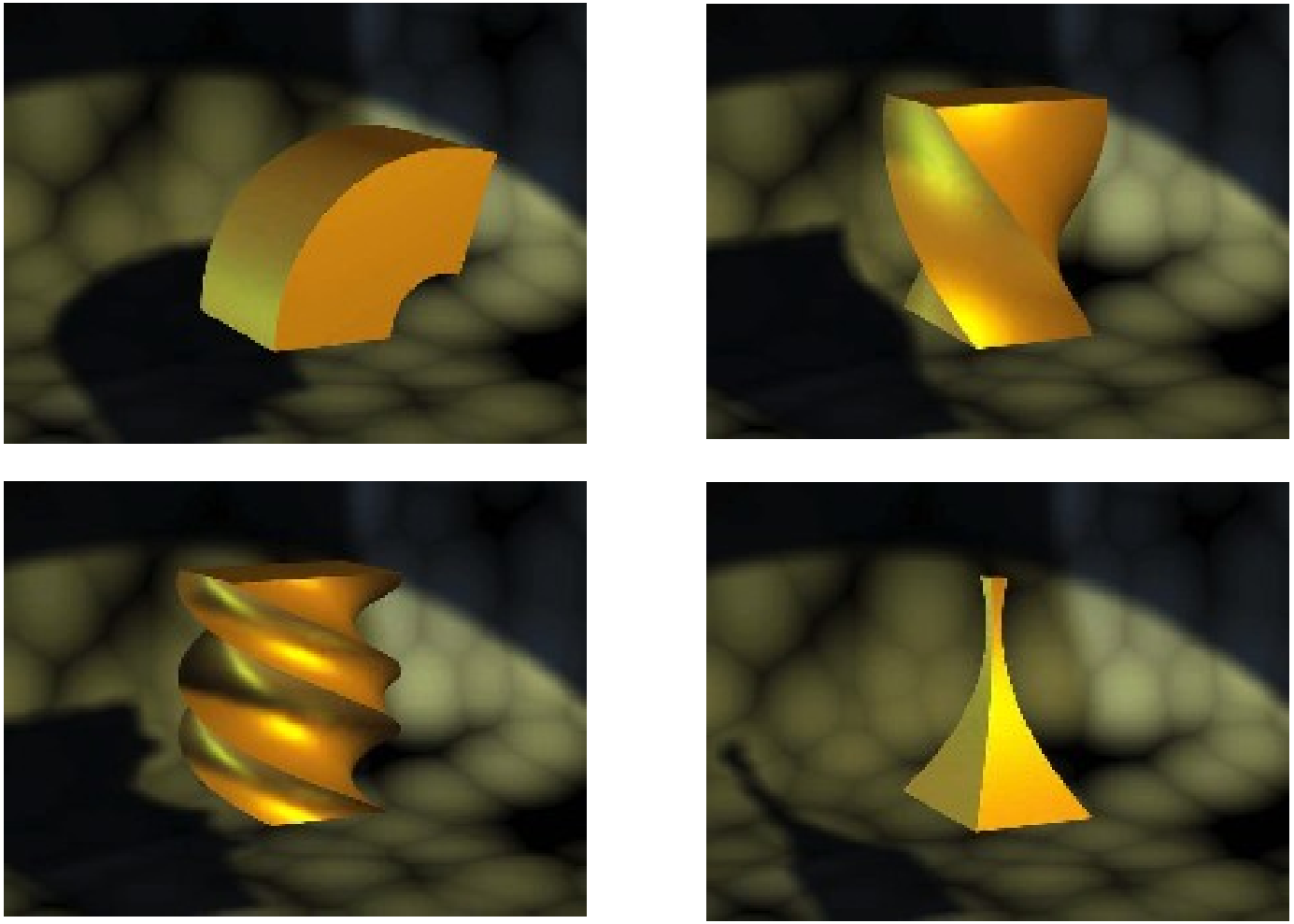
Tažené (extrudované) povrchy

- obecnější: pohyb profilu podél křivky



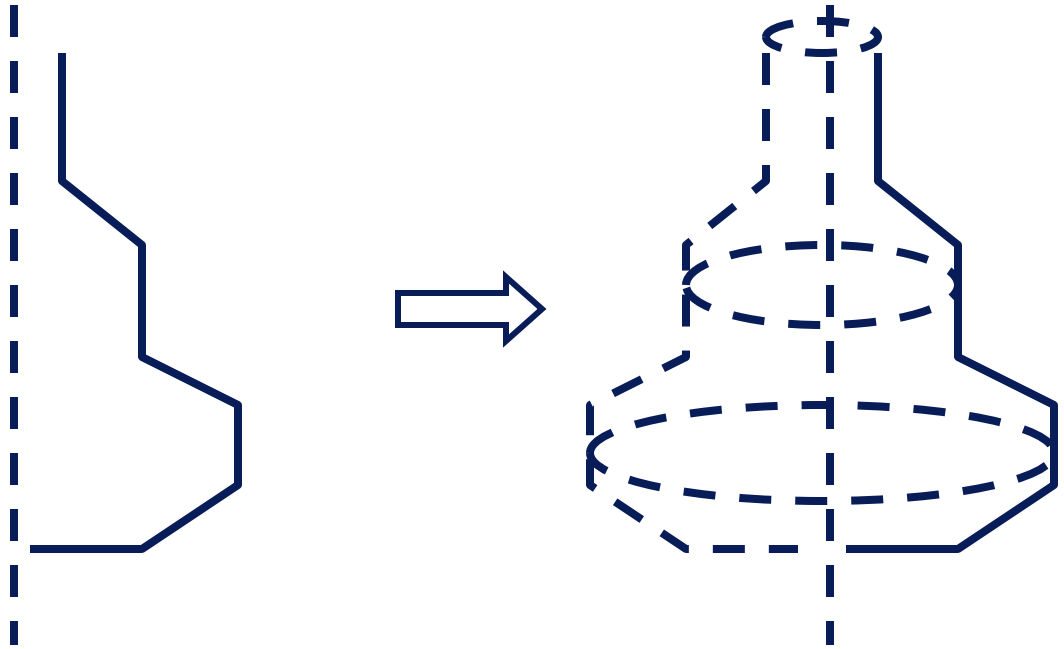
konstrukce polygonu mezi klíčovými polohami

Tažené (extrudované) povrchy



Rotační povrchy

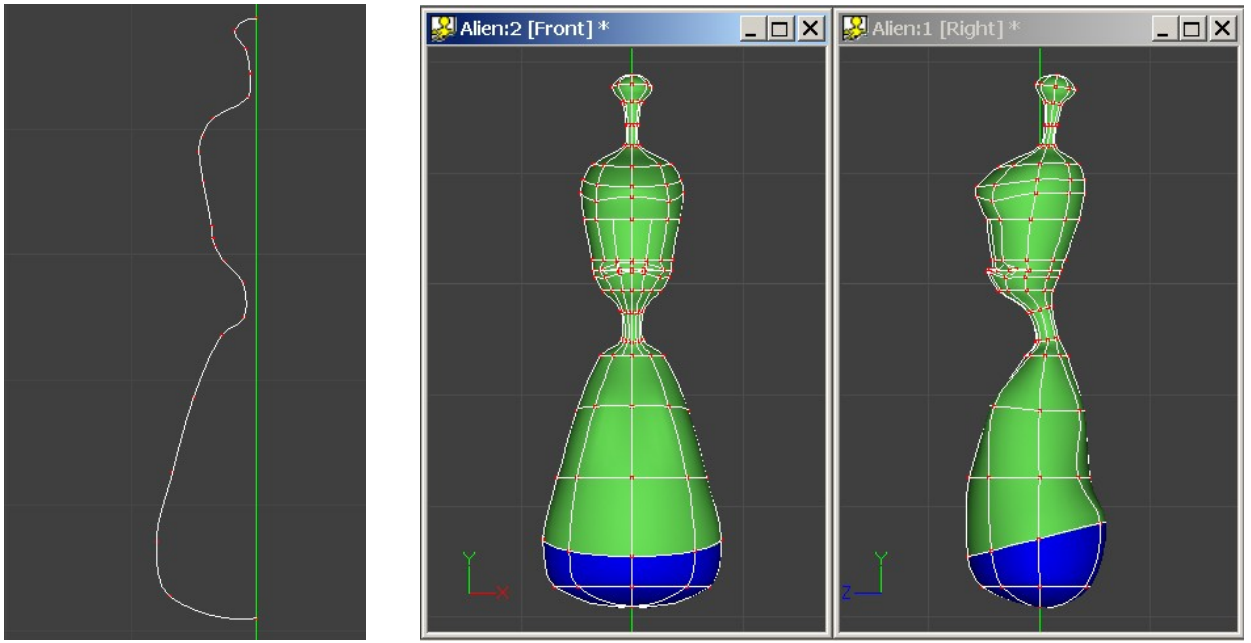
- definice profilu
- rotace profilu okolo osy



Rotační povrchy

- obdoba tažených povrchů

RHINO ....



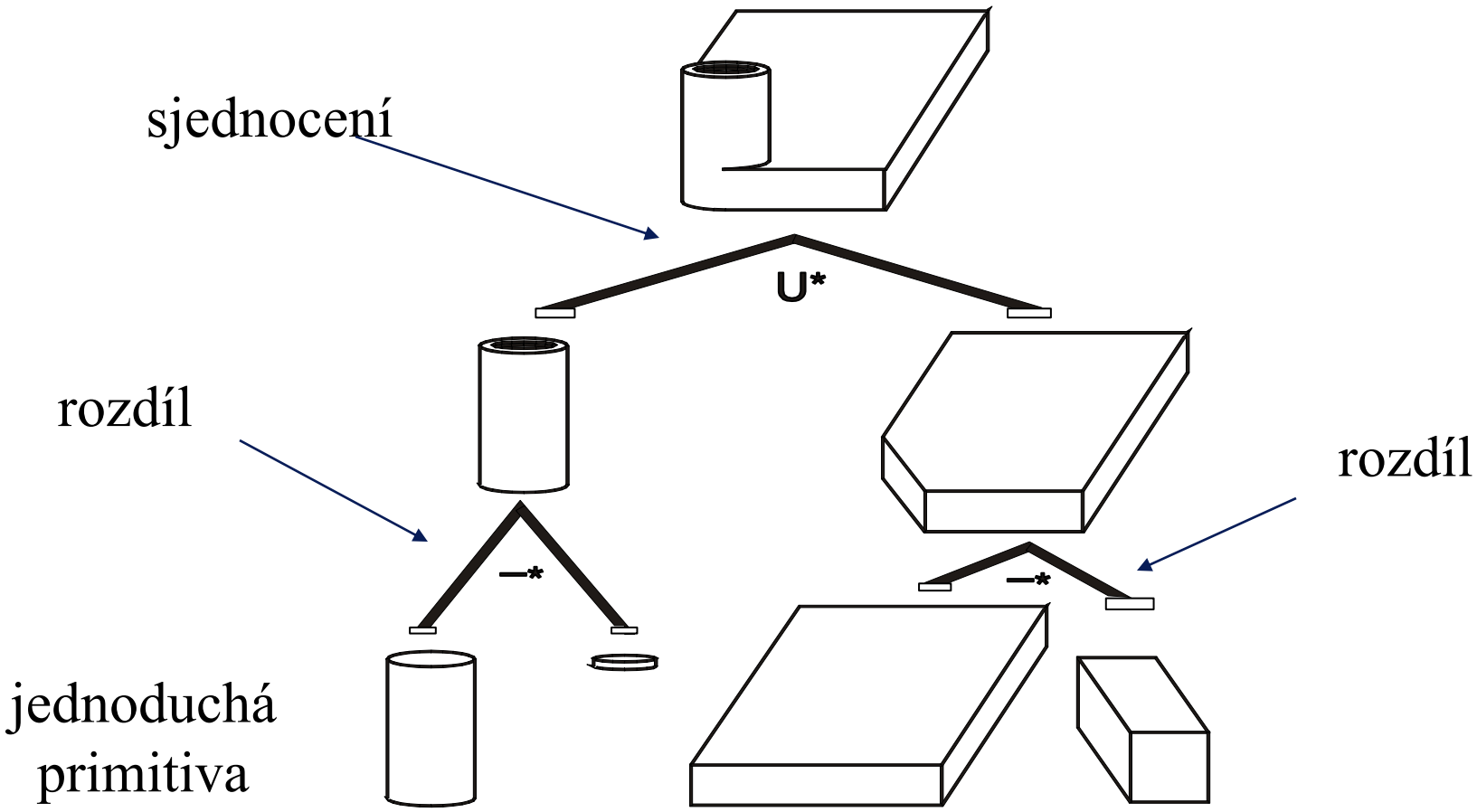
(rotační povrch + modifikace)

Konstruktivní geometrie těles - CSG

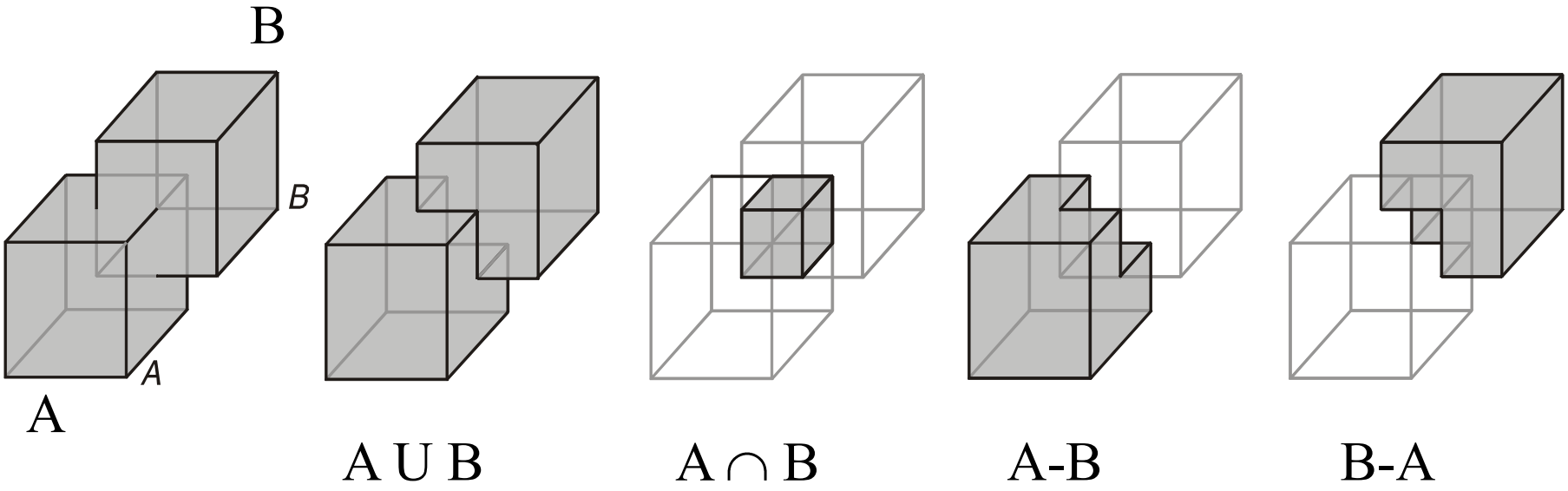
CSG – Constructive Solid Geometry

- Máme objekty popsané pomocí polygonů
- Jak zkombinujeme objekty ?
- Boolské operace:
  - součet, sjednocení
  - průnik
  - rozdíl

CSG

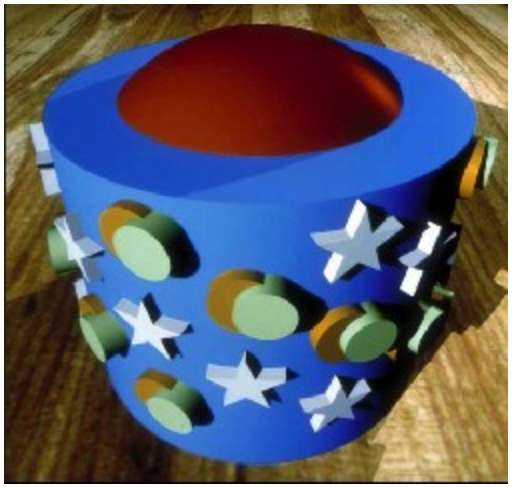


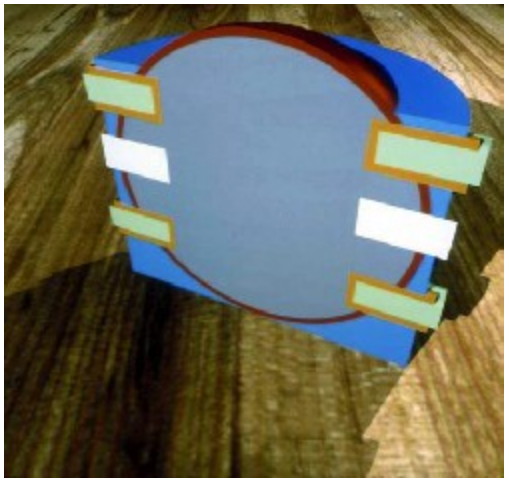
CSG



- řeší se pomocí ořezávání polygonů
- netriviální řešení okrajových případů

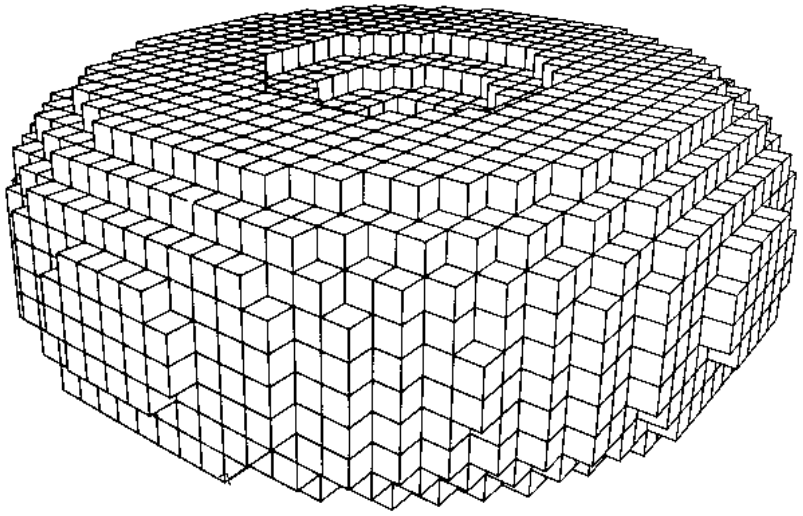
CSG





Objemové modelování

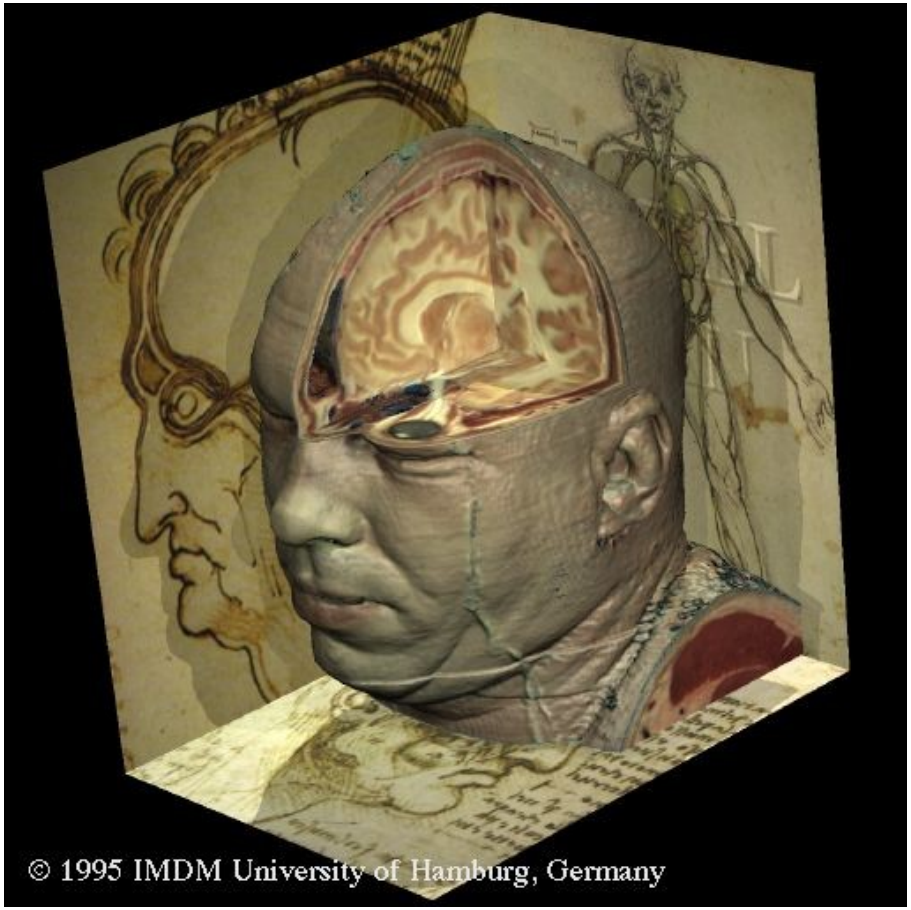
- prostor rozdělen na “voxely”
- označení každého voxelu:
  - Patří k objektu ?
  - Barva ?



Objemové modelování

- Aplikace: projekt „viditelný člověk“
- nasnímané řezy mrtvého těla
  - každý řez je polem voxelů
  - celé tělo je popsáno souborem objemových dat

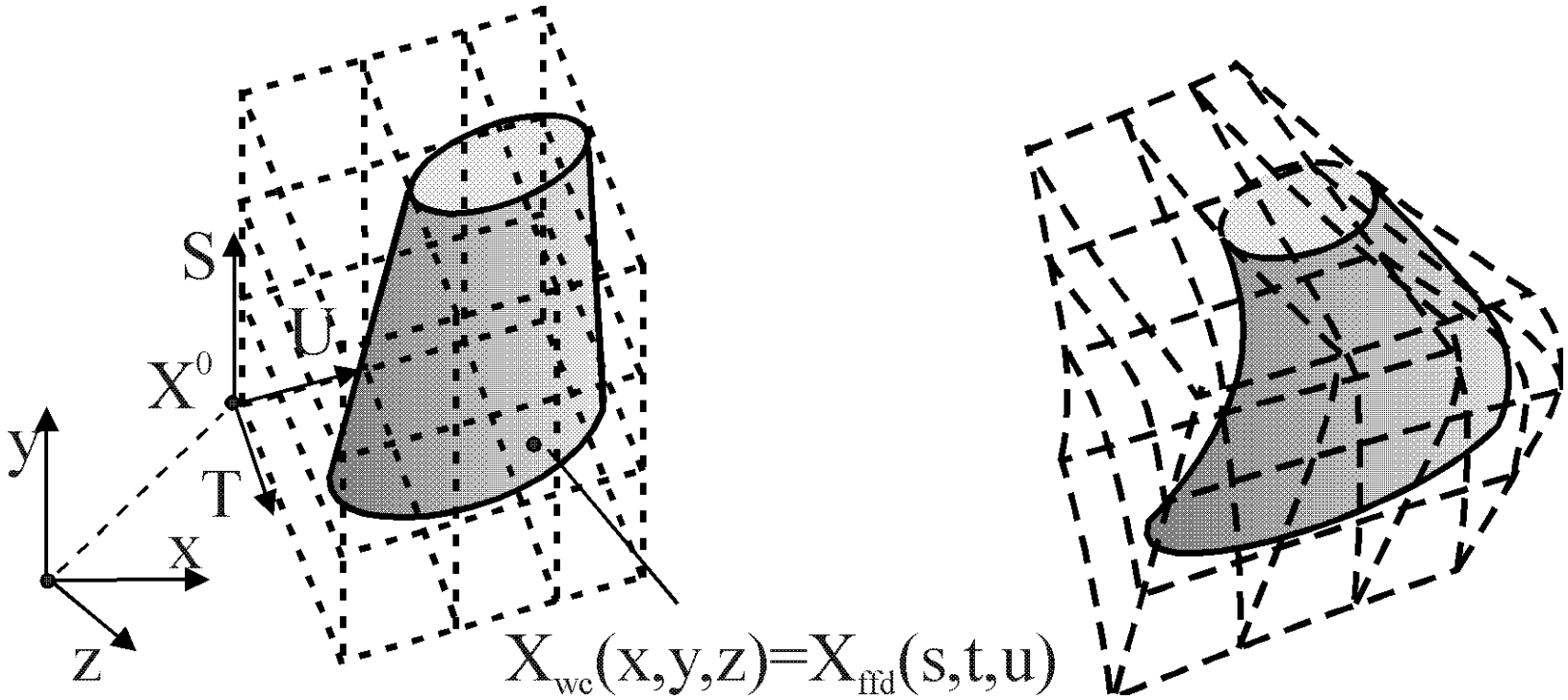
Projekt „Viditelný člověk“



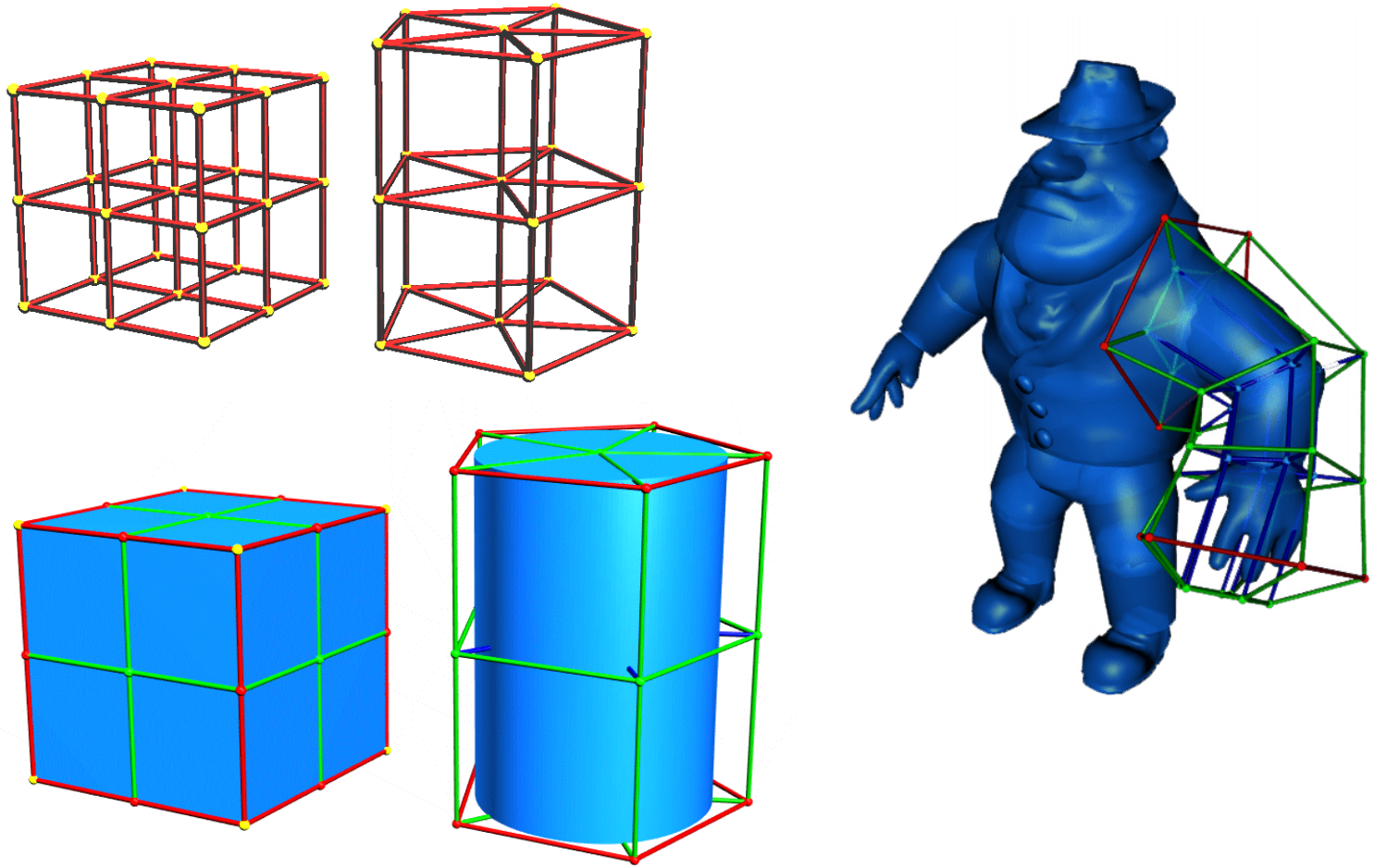
Volné tvarování

Volné deformace-Sederberg&Parry

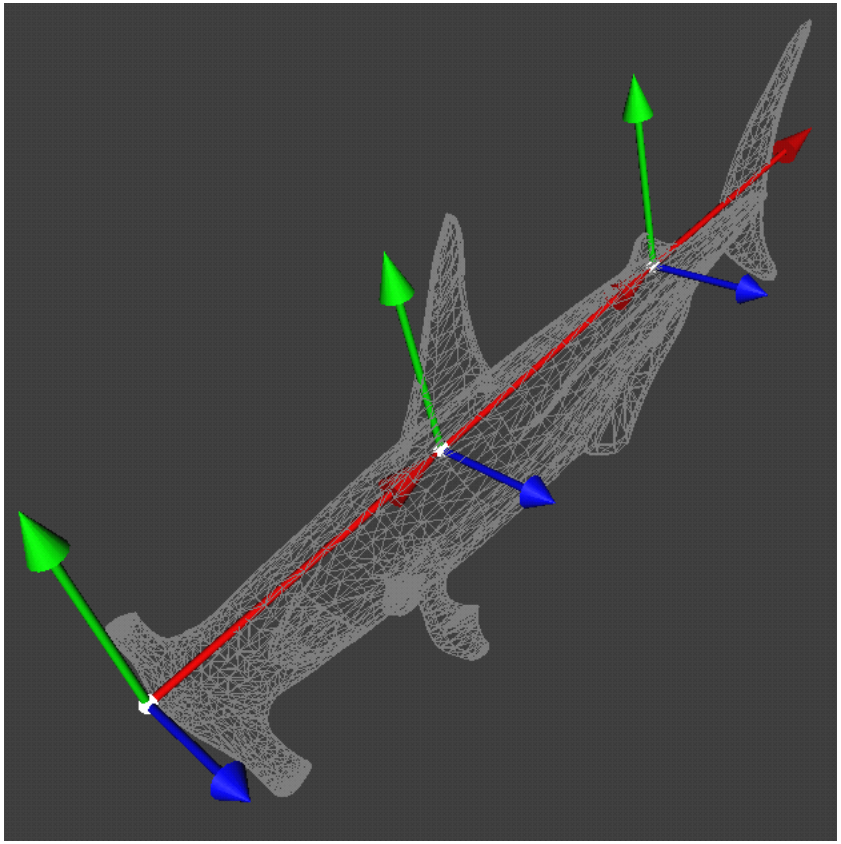
„volné deformace“ - Free Form Deformation - FFD

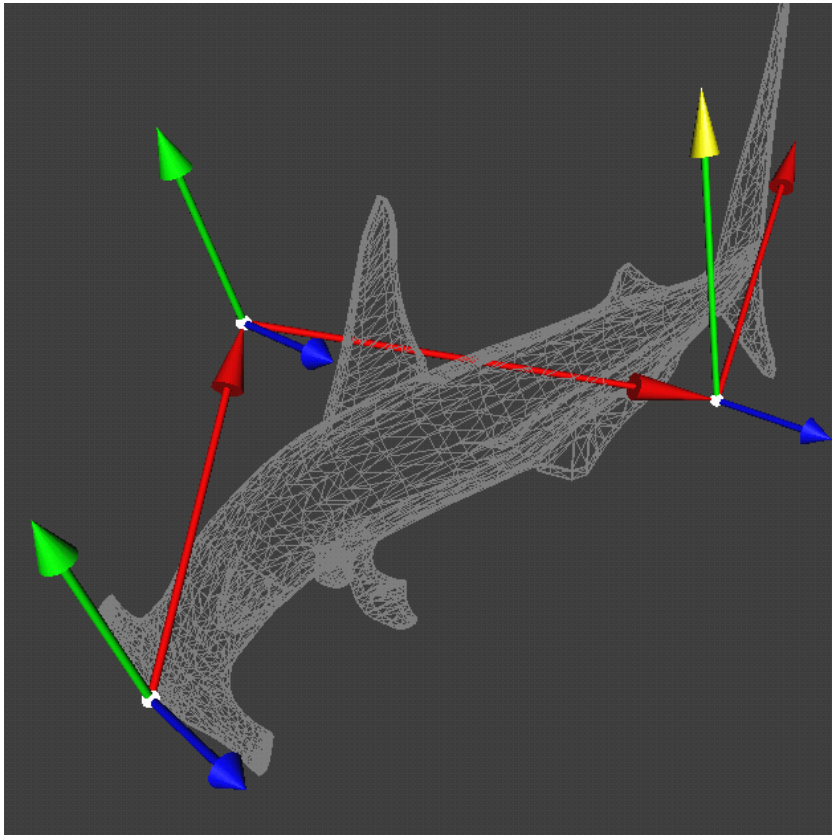


FFD - Cracken&Joy



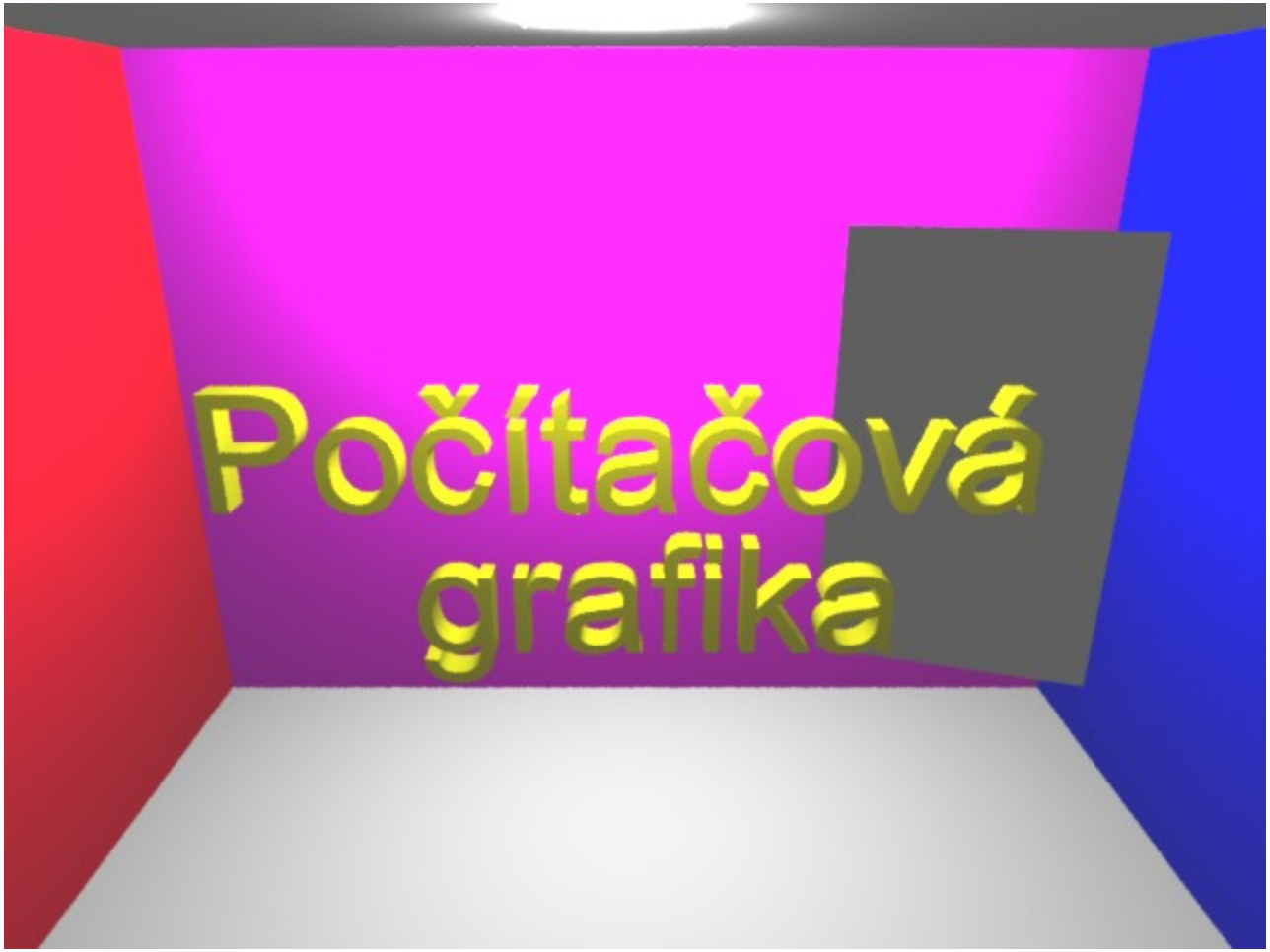
Deformace deCasteljau



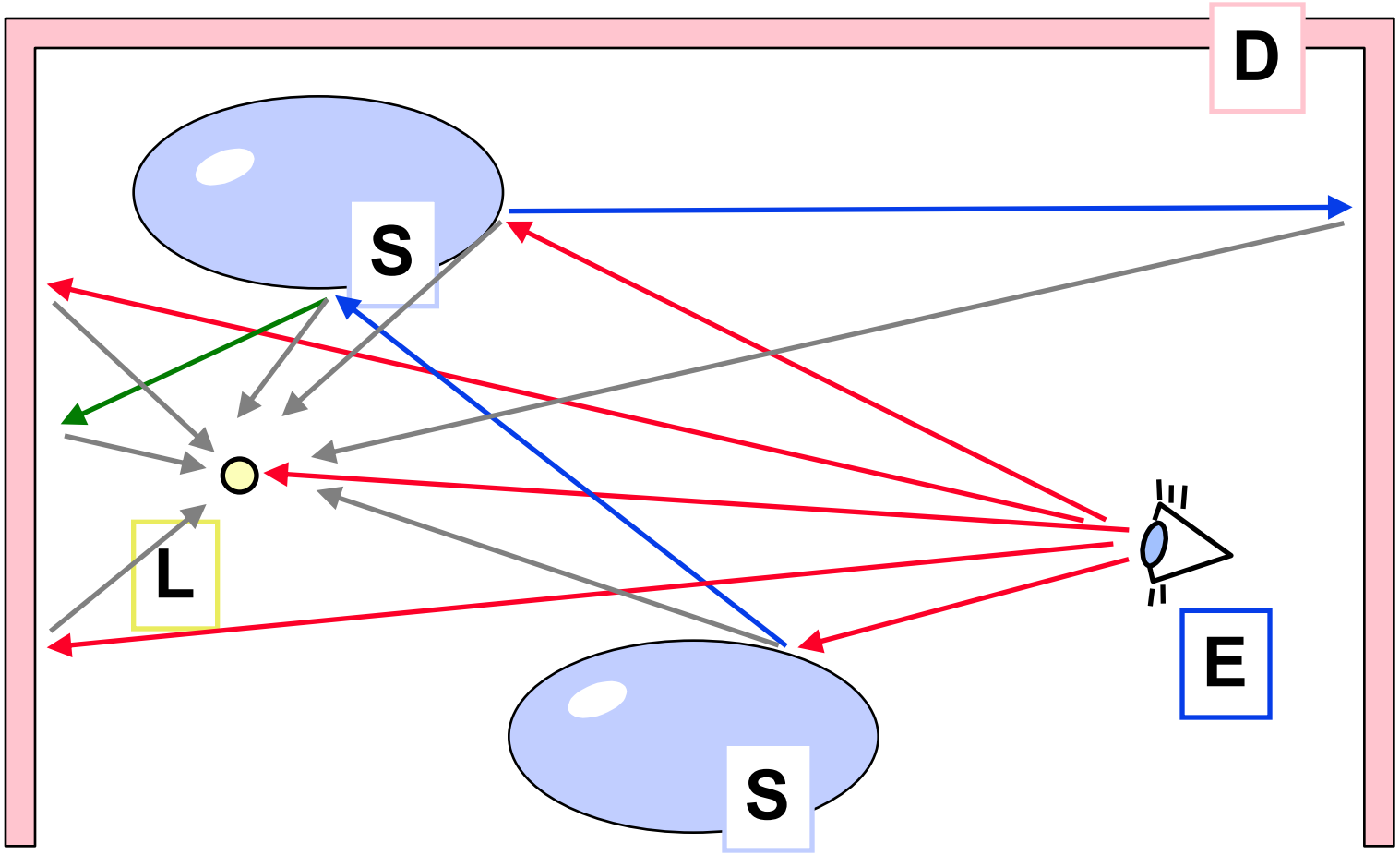


Světla a stíny

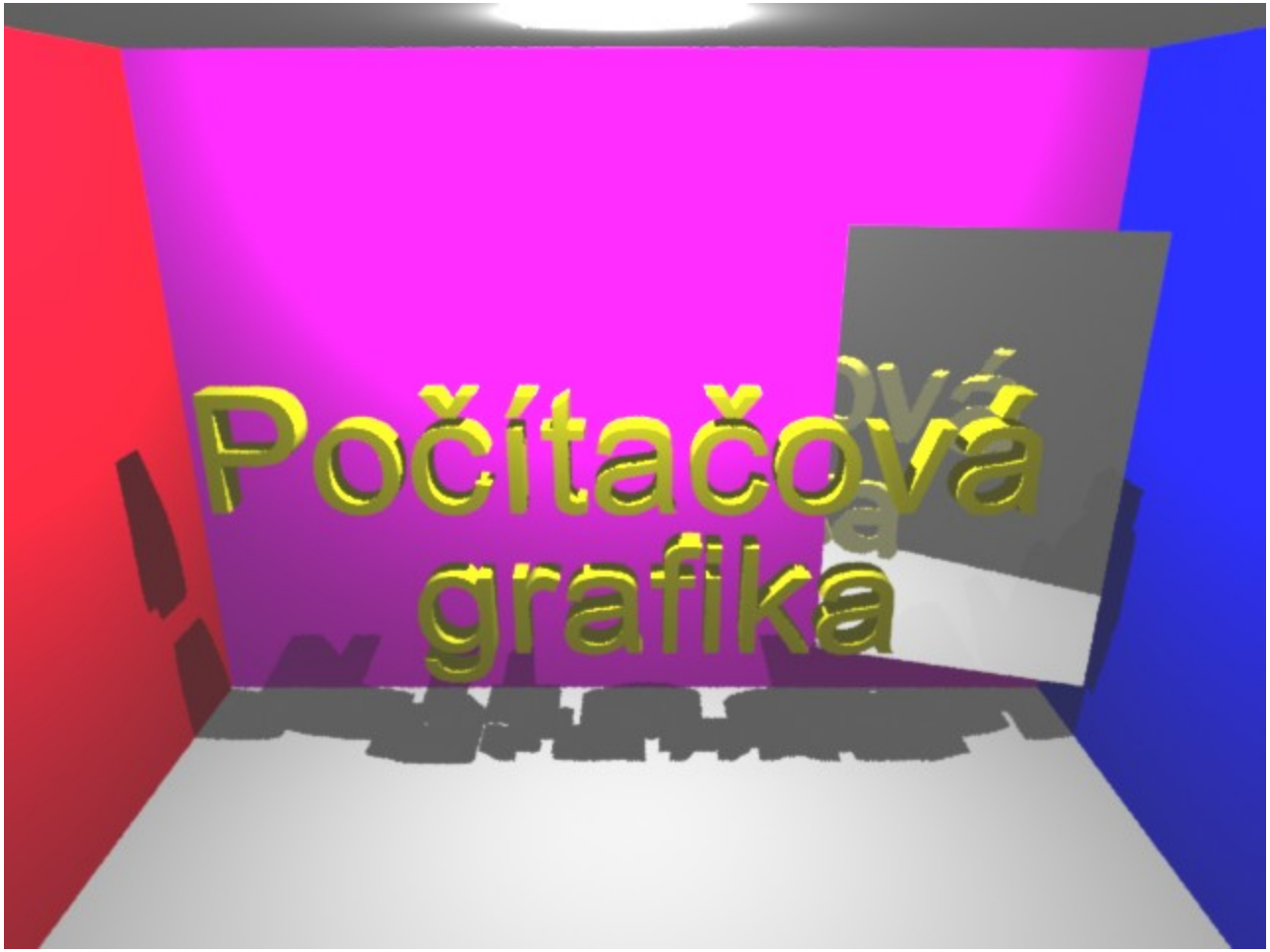
Klasická zobrazovací metoda



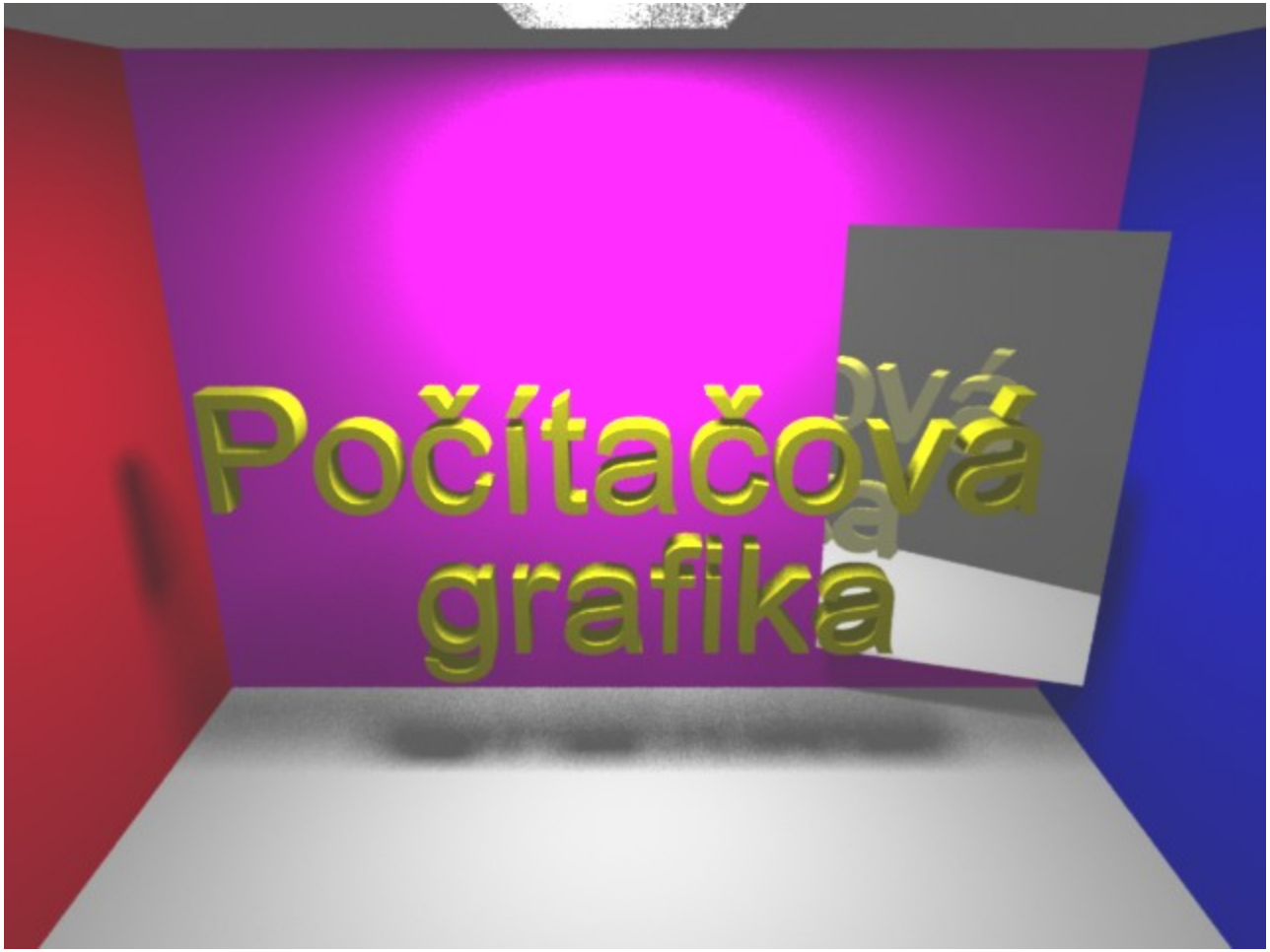
Zpětné sledování paprsku (eye ray-tracing)



Zpětné sledování paprsku



Distribuované sledování paprsku



Jednoduchá dvoukroková metoda

Tříkrokova metoda (zahrnutí **LS*D** cest)

Vizualizace

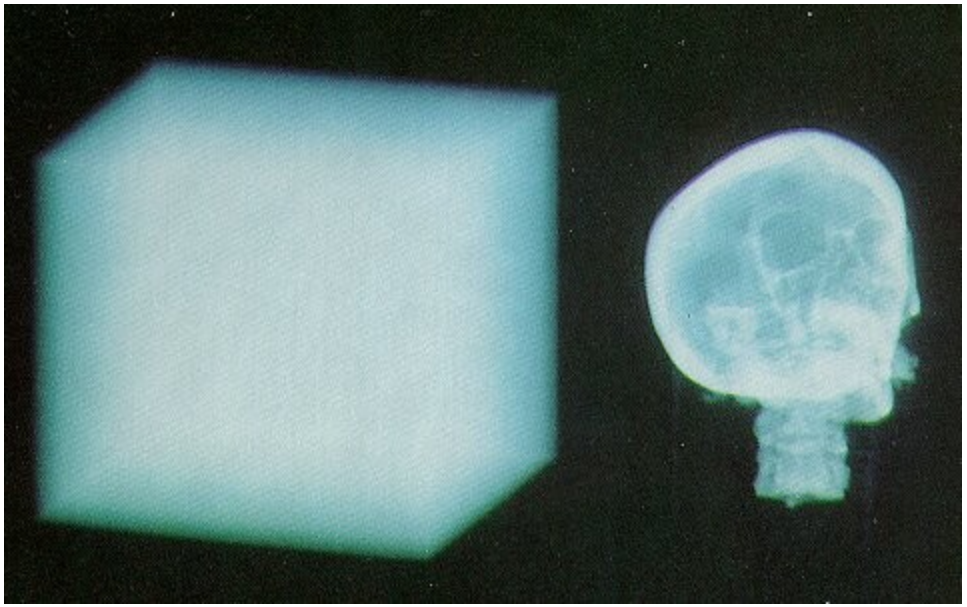
„průmyslové aplikace“

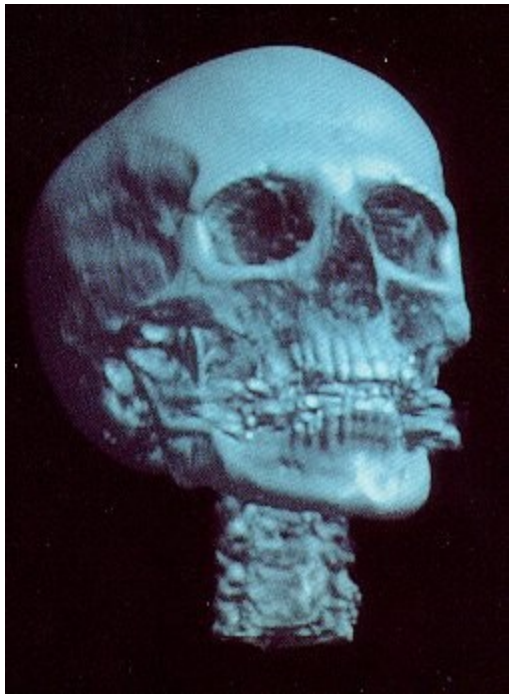
- **medicína**
  - (rentgenová) počítačová tomografie (CAT)
  - nukleární magnetická rezonance (NMR, MRI)
  - pozitronová emisní tomografie (PET)
  - “single photon emission computer tomography” (SPECT)

  + kombinace různých technologií (např. CAT+NMR)

- průmyslová **defektoskopie**
  - sonogramy, rentgenové přístroje, ..

Vizualizace v lékařství

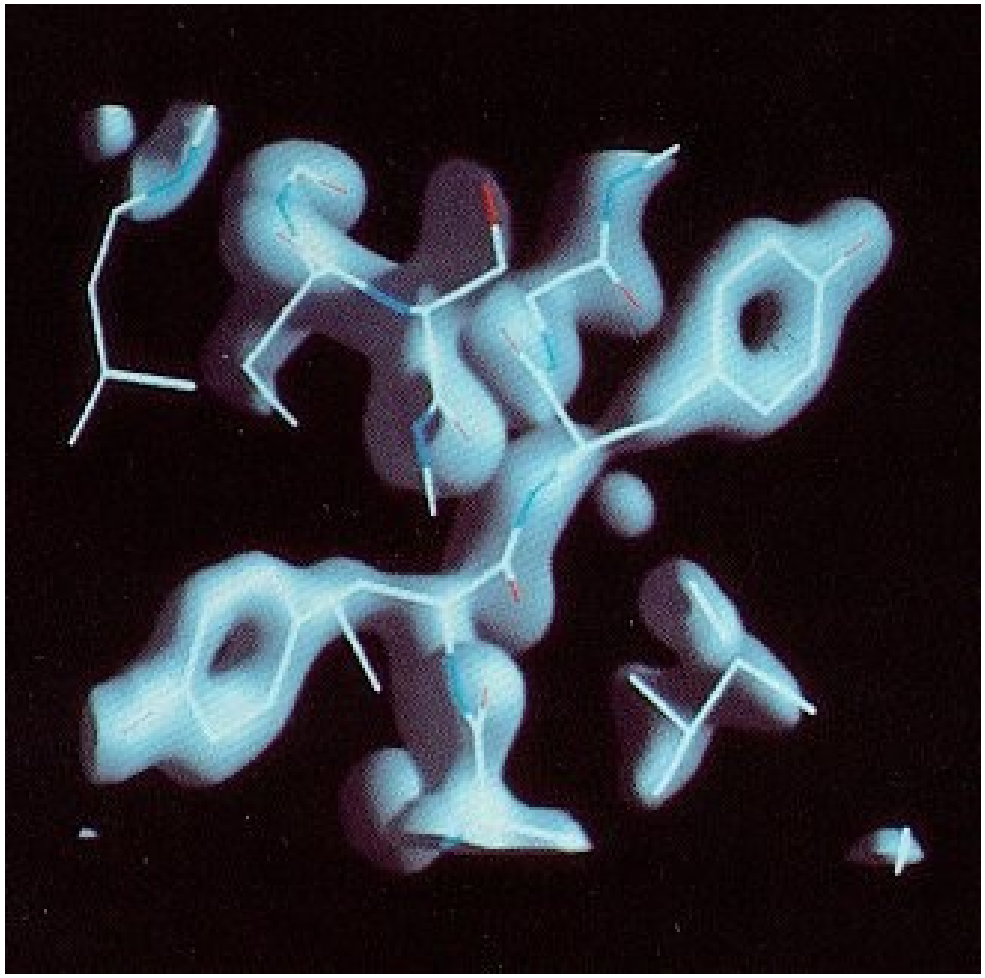




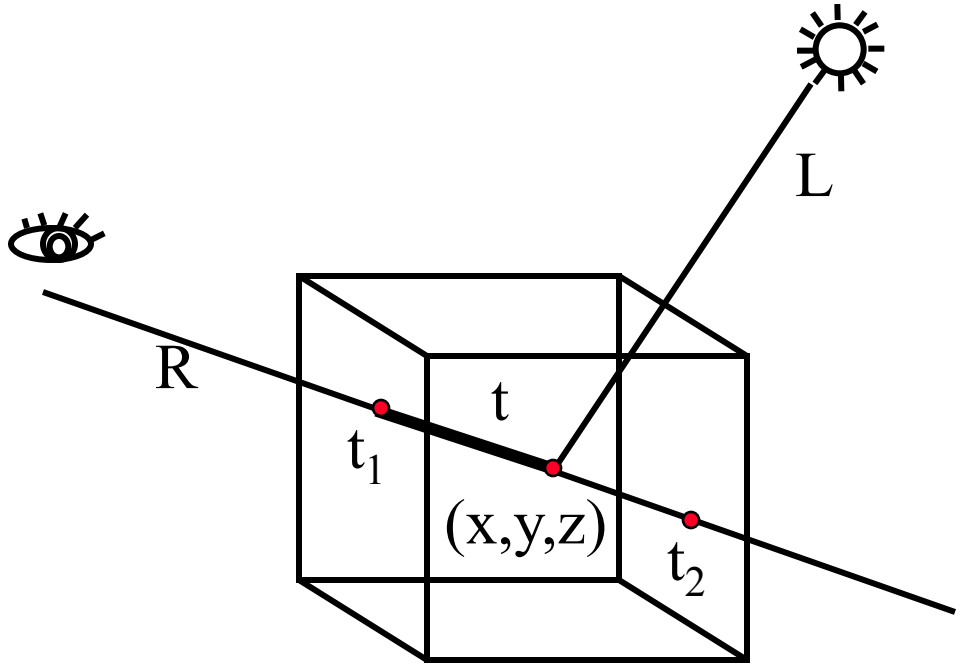
Vědecké aplikace

- zobrazení **naměřených dat**
  - geologie, seismologie
  - meteorologie
  - molekulární chemie a biologie
- zobrazení **matematické simulace**
  - **(dynamická) vektorová pole**: průmyslová konstrukce, aerodynamika, meteorologie, ..
  - astronomie a astrofyzika
  - zobrazení implicitně definovaných ploch

Vizualizace v chemii



Model rozptylu světla



osvětlení

$$I(t) = L(t) H(cc_{\theta})$$

hustota

odrazová funkce

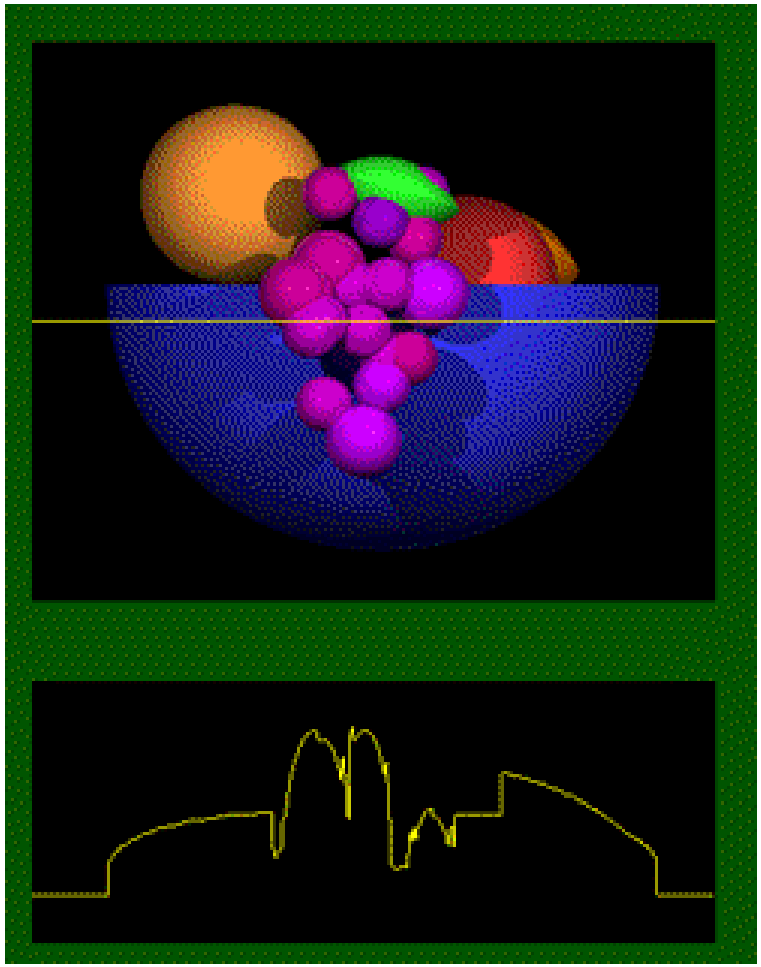
paprsek R prochází prostorem skalární funkce 3 proměnných *x,y,z*

Antialiasing

Antialiasing

- Alias vzniká díky diskrétní povaze obrazovky (rastrových zařízení):
- aliasové jevy mohou být redukovány pomocí
  - zvýšeného rozlišení
  - předfiltrováním
  - postfiltrováním

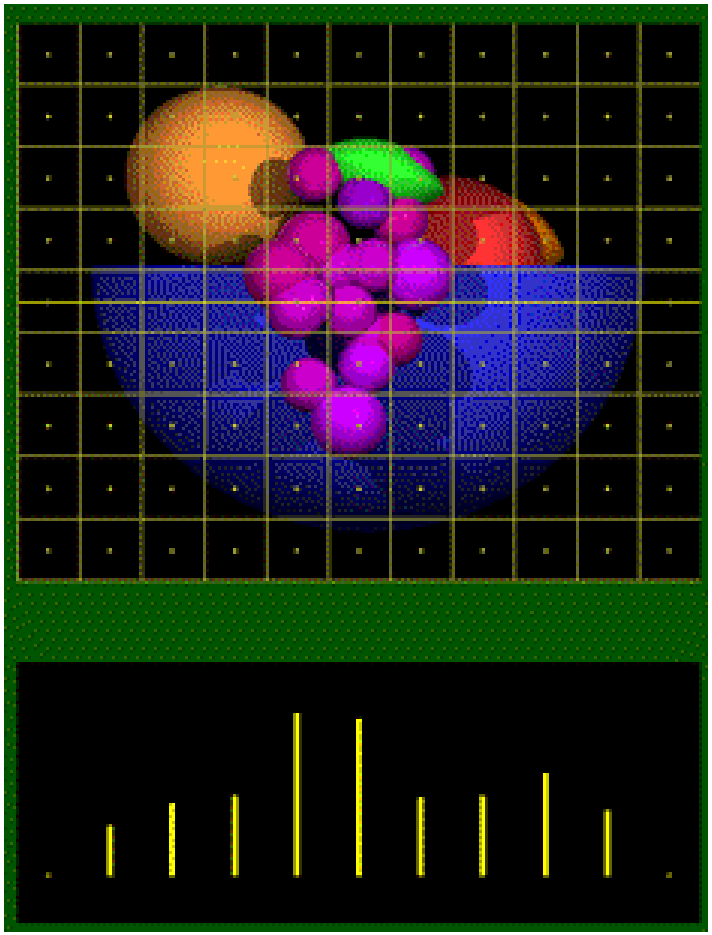
Antialiasing



původní scéna

průběh jasu

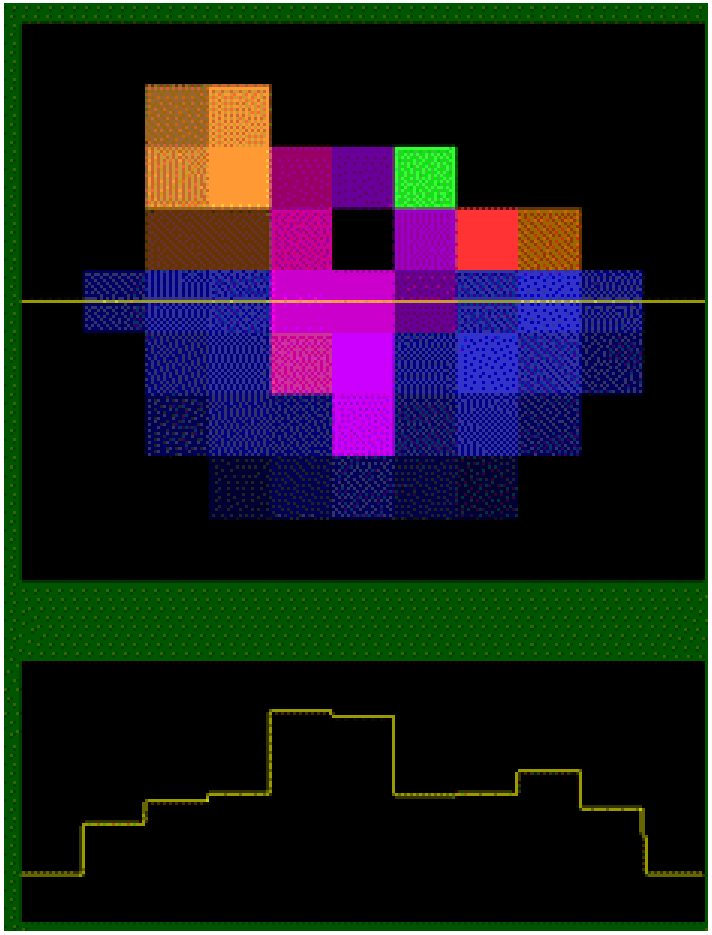
Vzorkování



vzorkování ve středech pixelů

vzorky jasového signálu

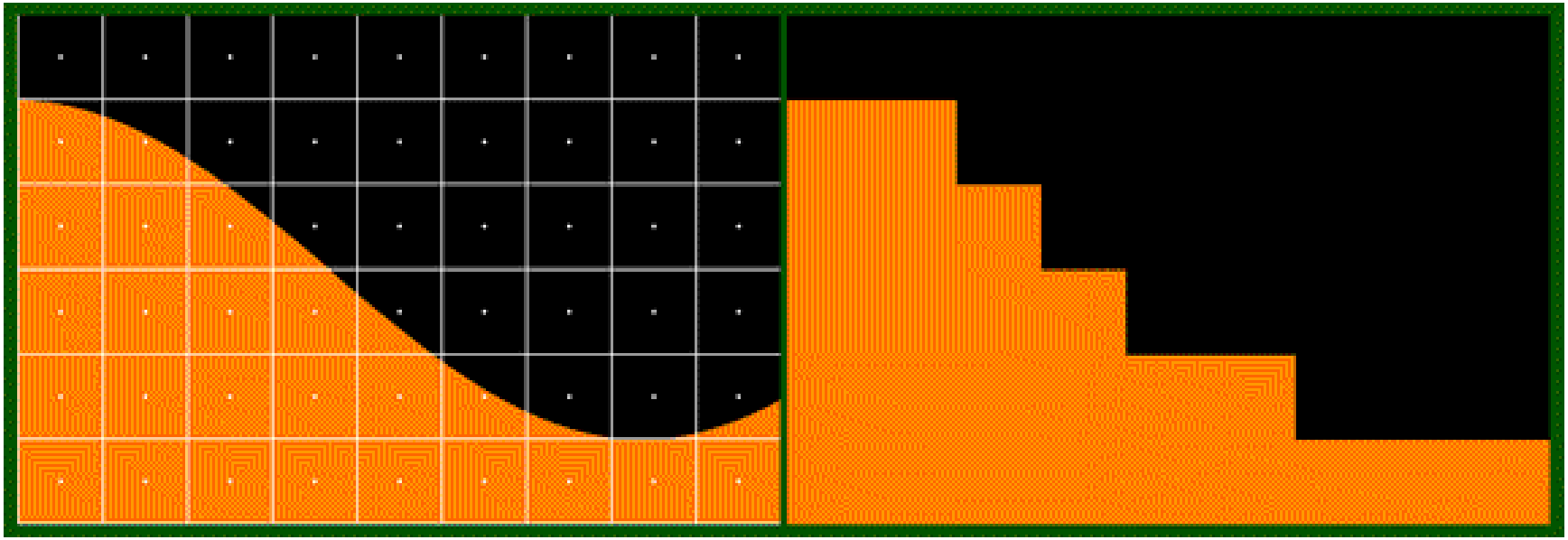
Alias - zobrazení



zobrazení

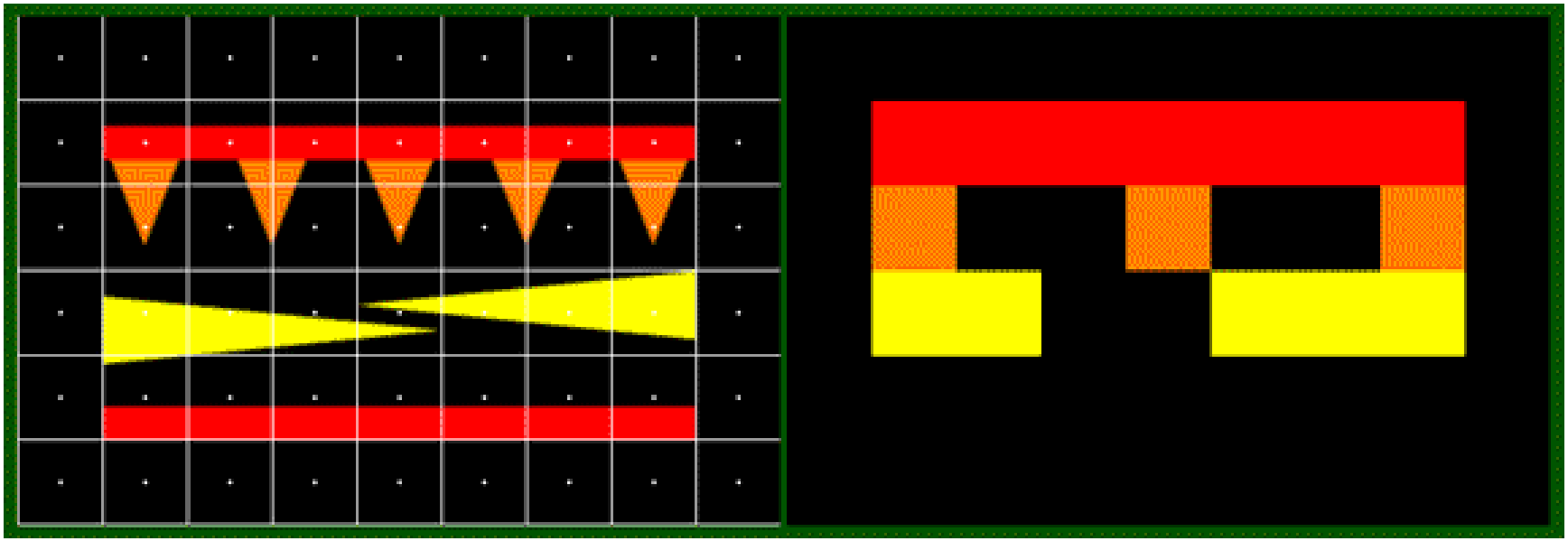
průběh jasu

Alias – zubaté profily



originál kresba

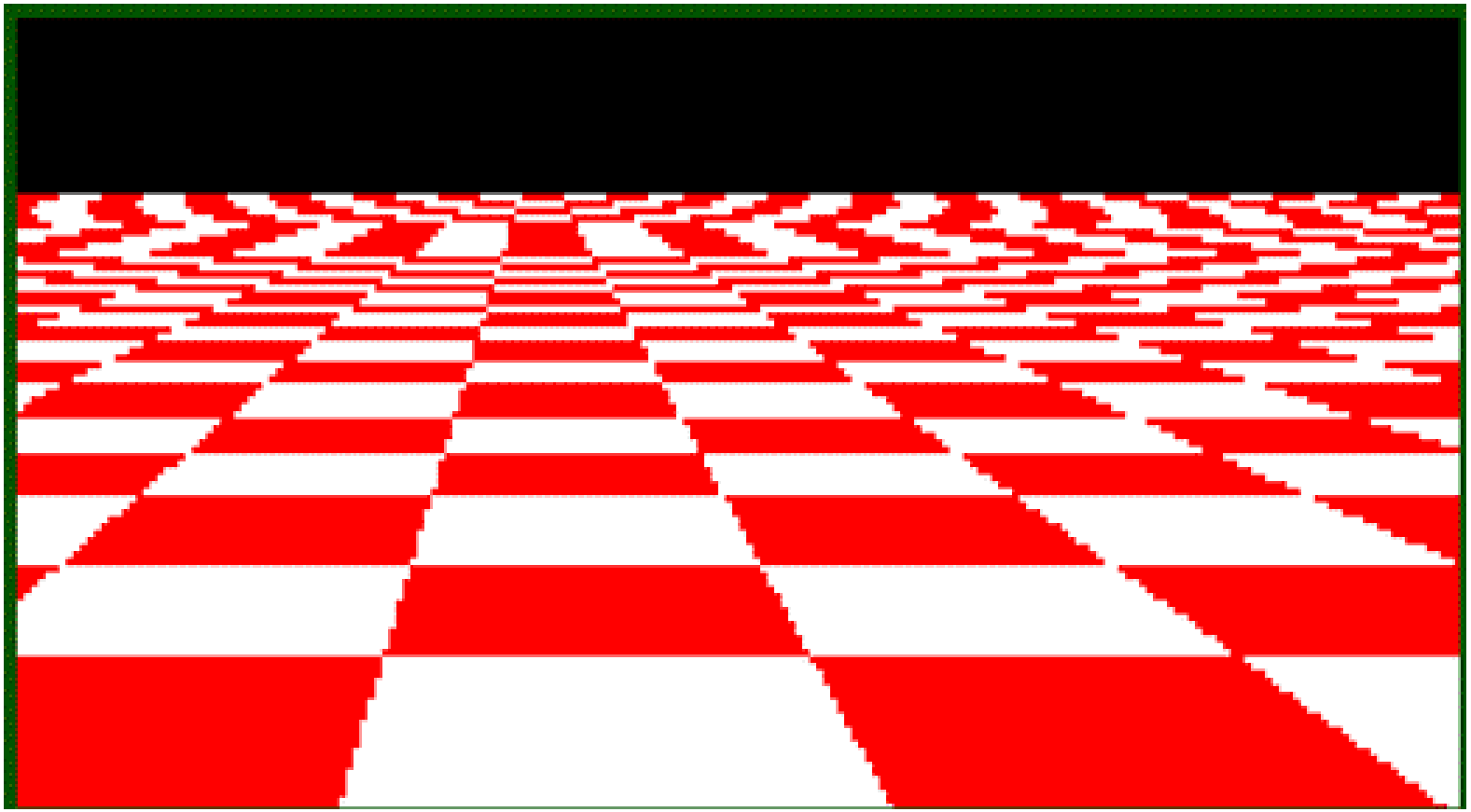
Alias – ztráta detailu



originál

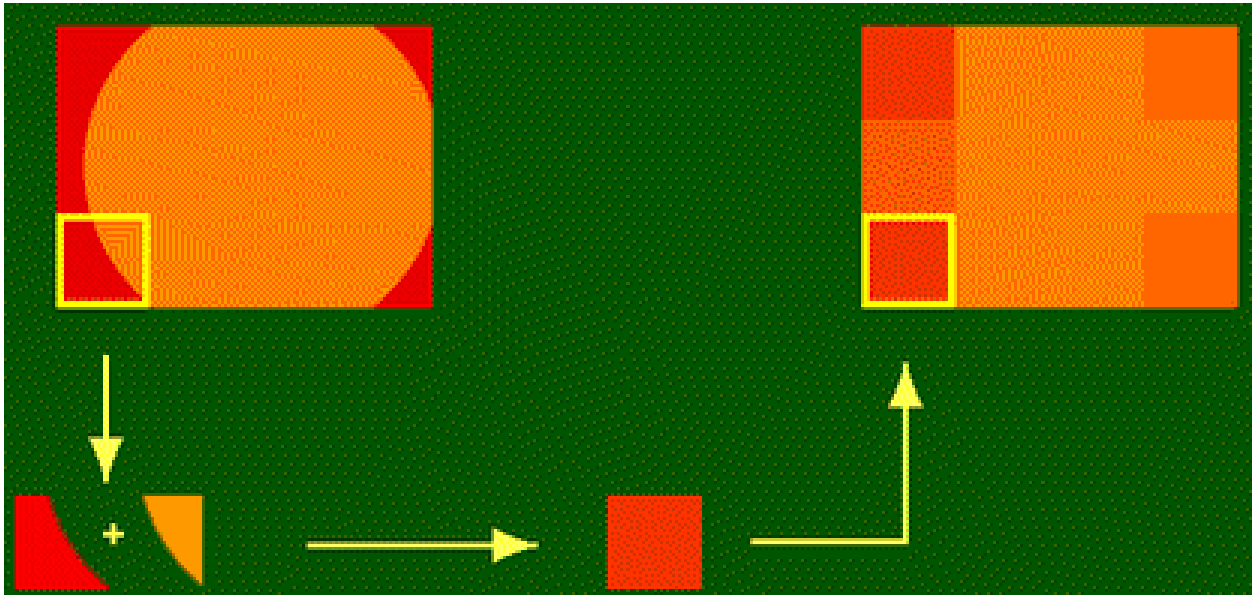
kresba

Alias – rozpad tvaru



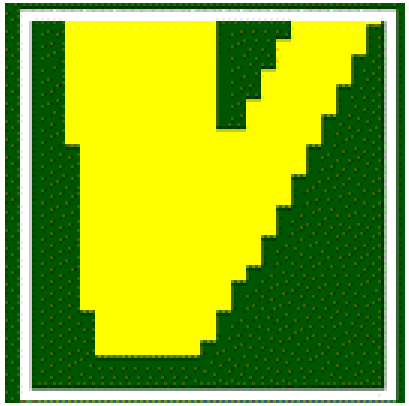
pravidelná šachovnice v perspektivě

Filtrování

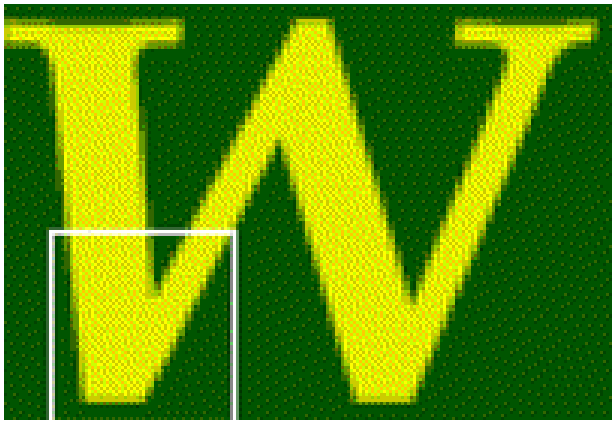


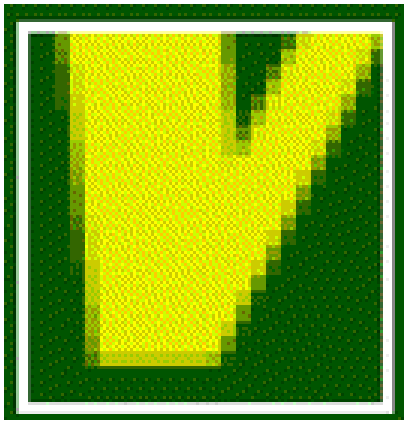
Předfiltrování zjišťuje barevné plochy uvnitř pixelu

Ukázka – bez antialiasing

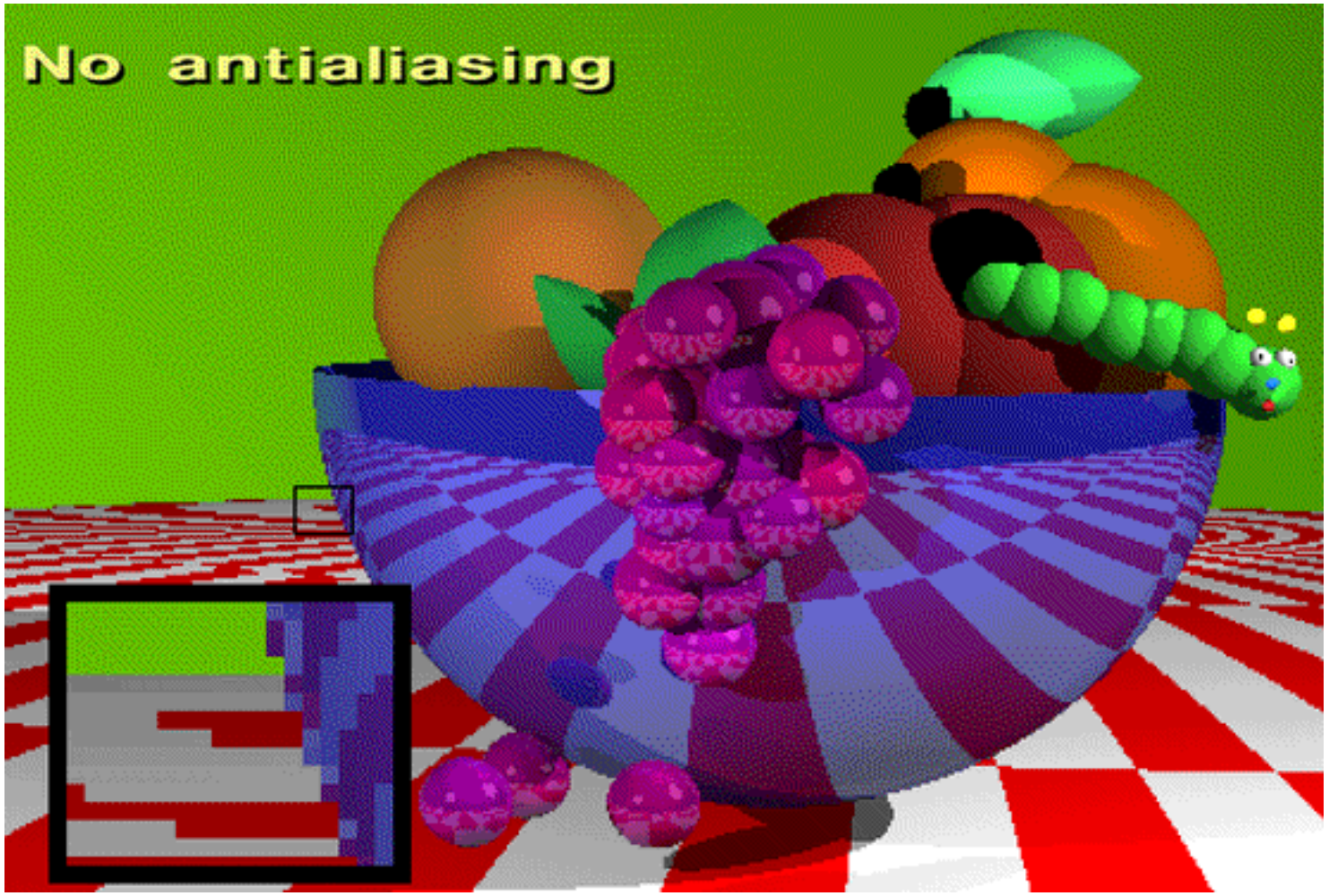


Ukázka – předfiltrování





Ukázka – bez antialiasing



Ukázka – předfiltrování