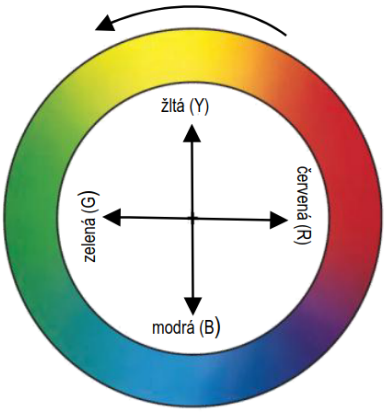
1 Používanie a spracovanie farieb v rámci počítačovej grafiky, základné atribúty svetla, farebný priestor, gamut

- Približne 80% informácií získava človek zrakom, preto zohráva v počítačovej grafike dôležitú úlohu
- Vnímanie svetla:
  1. časticovo (fotóny)
  2. ako vlnenie
- Typy svetla:
  1. achromatické svetlo - biele svetlo (obsahuje všetky farby)
  2. monochromatické svetlo - svetlo len jednej farby
- Atribúty:
  1. farba - závisí od frekvencie (resp. vlnovej dĺžky)
  2. jas - intenzita svetla, jasnosť zdroja svetla
  3. sýtosť - čistota, čím vyššia sýtosť, tým užšie je spektrum frekvencií obsiahnutých vo svetle
  4. svetlosť - veľkosť achromatickej zložky vo svetle s určitou dominantnou frekvenciou
- Farebný priestor:
  - Rozsah od 380nm (fialová) do 780nm (červená)
  - Oblasť farieb pokrytá farebným modelom
- Gamut:
  - Dosiahnuteľná oblasť farieb v danom farebnom priestore
  - Farby mimo gamutu je možné zobraziť len približne

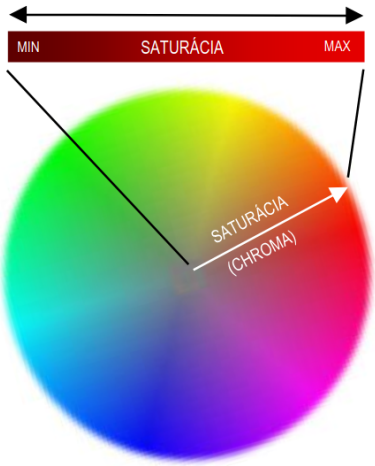
2 Chromatický diagram, typy, odtieň a saturácia

- CIE chromatický diagram:
  - Model vyvinutý na základe štandardného pozorovateľa s normálnym farebným videním
  - Pre určovanie farby sa používa tristimulus (trojzložkový systém)
  - Farba určená trojicou čísel/koordinátov (X, Y, Z), pričom udávaju množstvo každej z troch hypotetických primárnych zložiek
- CIELAB:
  - svetlo absorbované (založené na substraktívnom miešaní)
  - zobrazený ako guľa
- CIELUV:
  - svetlo emitované (založené na aditívnom miešaní)

  - zobrazený ako cylinder
- Odtieň (Hue):
  - popisuje spôsob vnímania farieb a prechod medzi jednotlivými farbami
  - zobrazenie ako uzavretý kruh

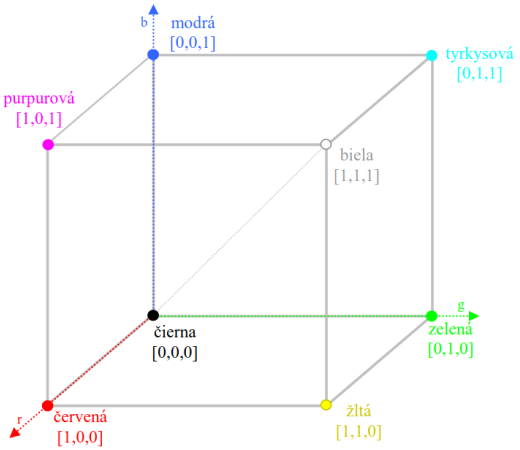


- Saturácia (Chroma):
  - popisuje živosť farby, ako blízko má farba k sivej
  - zobrazenie ako disk, farby v strede sivé, na okrajoch živé



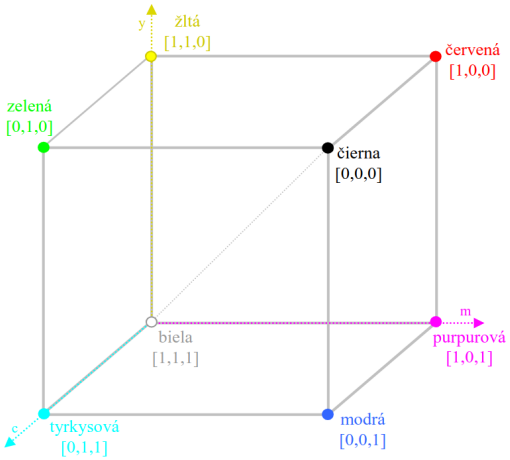
3 Farebné modely RGB a RGBA

- Zložky: **R**ed, **G**reen, **B**lue, **A**lpha (priesvitnosť)
- miešanie je aditívne
- zmena zložky je lineárna
- model reprezentovaný kockou



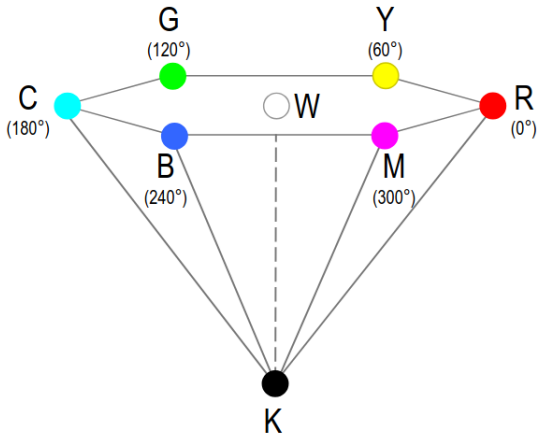
4 Farebné modely CMY a CMYK

- Zložky: **C**yan, **M**agenta, **Y**ellow, Blac**K**
- miešanie je substraktívne
- zmena zložky je lineárna
- model reprezentovaný kockou



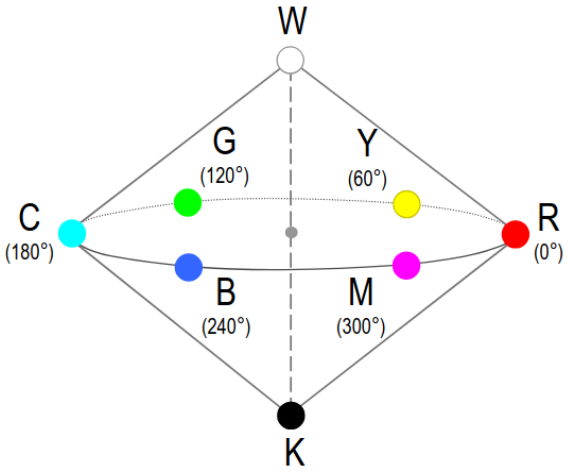
5 Farebný model HSB

- Zložky: **H**ue, **S**aturation, **B**rightness
- miešanie je aditívne
- zmena zložky je uhlová a lineárna
- model reprezentovaný 6-bokým ihlanom



6 Farebný model HLS

- Zložky: **H**ue, **L**ightness, **S**aturation
- miešanie je aditívne
- zmena zložky je uhlová a lineárna
- model reprezentovaný dvojitý kužeľ



7 Gama korekcia a alfa-miešanie

- Gama - koeficient gama, definuje vzťah medzi číselnou hodnotou obrazového bodu a jeho skutočnou svietivosťou
- Gama korekcia:
  - aplikuje sa preto, lebo ľudské oko má vyššiu citlivosť v tieni (L=0,5), kým pri jasnom svetle (L=1) alebo v tme (L=0) je menej citlivé
  - naopak technické prostriedky (senzory, snímače, zobrazovače) vnímajú/emitujú svetlo spravidla lineárne
  - gama korekcia pomáha riešiť rozdiel medzi technickým spracovaním farieb a prirodzeným ľudským vnímaním

- Alfa miešanie:
  - Pri alfa miešaní sa farby dvoch pixelov kombinujú podľa ich alfa hodnôt, pixel s vyššou alfa hodnotou bude dominantnejší a bude viac viditeľný
  - Lineárne alfa miešanie (trajektória prechodu medzi dvoma farbami je úsečka):
    1. uniformné
    2. exponenciálne
  - Nelineárne alfa miešanie (trajektória prechodu medzi dvoma farbami je krivka):
    1. uniformné
    2. logaritmické

8 Problematika ľudského vizuálneho vnemu a jeho spracovania v relácii s počítačovou grafikou

- Hlavným ľudským orgánom pre príjem obrazových informácií je oko
- Ľudské oko vníma farby prostredníctvom:
  - Tyčiniek - pokrývaju sietnicu umožnujú vnímať všeobecné obrazové informácie
  - Čapíkov - v strede sietnice, sú delené podľa citlivosti na farby:
    * červená-zelená
    * modrá-žltá

9 Miešania a rozptyľovania farieb (prevod do šedej škály, halftoning, dithering) v rámci počítačovej grafiky

- Prevod na GrayScale:
  - I - intenzita úrovne šedej
  - Prevod RGB na GRAYSCALE - I = 0.299*R + 0,587*G + 0.114*B
- Halftoning:
  - Poltieňovanie simuluje plynulý prechod medzi odtieňmi využitím bodov a medzier rôznych veľkostí
  - Pri dostatočnej vzdialenosti ľudské oko nedokáže vnímať jednotlivé body a obraz vníma ako celok
- Dithering:
  - Využíva obmedzené rozlíšenie ľudského oka na vytvorenie ilúzie väčšieho počtu farieb
  - Zblízka sú viditeľné jednotlivé pixely, ale z diaľky sa zlúčia do jedného farebného tónu

10 Ergonómia a symbolika farieb, použitie farieb v počítačovej grafike

- Ergonómia farieb:
  - zameriava na to, ako farby ovplyvňujú používateľov

  - farby majú psychologický a vizuálny vplyv
  - správne použitie farieb môže zlepšiť používateľský zážitok, zatiaľ čo nesprávne farebné kombinácie môžu viesť k únavě očí, zmätku alebo zníženej produktivite
- Symbolika farieb:
  - Biela - symbolizuje čistotu, vyvoláva pozitívne pocity, veselá farba
  - Čierna - spojená so smútkom, pasívna farba
  - Modrá - evokuje vernosť a diaľku, pôsobí zdržanlivo, kľudne a vážne, studená a pasívna farba
  - Červená - symbolizuje energiu a vášeň, teplá, aktívna a dominantná farba
  - Zelená - symbolizuje nádej, osviežujúca, ukľudňujúca, pasívna, neutrálna farba
  - Šedá - symbolizuje nerozhodnosť, pasívna, neenergetická, nedominantná farba
  - Žltá - symbolizuje inteligenciu, pôsobí veselo a mocne, aktívna, teplá, dominantná farba
- Pravidlá použitia farieb v počítačovej grafike:
  - Pre veseľosť a vyrovnanosť je vhodné použiť pestré palety farieb, ale s mierou
  - Silné farebné kontrasty zvýrazňujú pozornosť
  - Teplé a studené farby sa používajú na vytvorenie hĺbky a priestoru (teplé farby sa zdajú byť bližšie, studené vzdialenejšie)
  - Pre dominantné prvky sú vhodné kombinácie žiarivých a kalných farieb
  - Použitie monochromatických farieb (odtiene tej istej farby) môže vytvoriť harmonický, decentný vzhľad

11 Dimenzia priestoru a dimenzia objektu, štruktúra dimenzie

- Dimenzia priestoru:
  - Určuje počet nezávislých smerov, v ktorých sa môže objekt pohybovať alebo rotovať
  - poznáme dimenziu celočíselnú a neceločíselnú
- Dimenzia objektu:
  - daná dimenziou priestoru
  - objekt s nižšou dimenziou môže existovať vo vyššom dimenzionálnom priestore, ale nie naopak, bod (0D) môže existovať na priamke (1D), v ploche (2D) aj v priestore (3D)
- Štruktúra dimenzie D(imenzia) = M + N :
  - Homogénna - všetky dimenzie rovnakého typu $3 = 3 + 0$ (geometria + čas)
  - Nehomogénna - dimenzie rôzneho typu $3 = 2 + 1$ (geometria + čas)

12 Priestor a jeho súradnicová sústava, stupeň voľnosti

- Súradnicová sústava - parametrizuje priestor a definuje jeho počiatočný bod (stred) a smery rozvoja
- Súradnice - čísla, ktoré určujú presnú polohu nejakého bodu v koordinačnom priestore
- Stupeň voľnosti:
  - počet rôznych spôsobov, ako môže objekt alebo priestor meniť svoju polohu alebo stav v súradnicovom systéme
  - Translačné pohyby - pohyby v rôznych smeroch
  - Rotačné pohyby - rotácia objektu okolo osi
  - Jednosmerné pohyby – objekt sa môže pohybovať iba v jednom smere v rámci dimenzie
  - Obojsmerné pohyby – objekt sa môže pohybovať v oboch smeroch tej istej dimenzie

13 Vrstvy vizualizačného procesu

1. Definovanie/formalizácia/spracovanie modelu
2. Transformácie nad objektami
3. Riešenie viditeľnosti
4. Tieňovanie
5. Osvetľovanie
6. Realistické zobrazovanie
7. Kompozícia a Vykresľovanie

14 Grafická informácia po objektovej aj typovej stránke

- Objektová stránka:
  - Grafická informácia sa skladá z objektov alebo entít, ktoré môžu byť primitíva (ako čiary, body, polygóny)
  - Objekty môžeme opisovať vlastnosťami ako farba, veľkosť a pozícia
  - Primitíva sa reprezentujú buď vektorovo alebo rastrovo
- Typová stránka:
  - Grafické informácie môžeme rozdeliť na vektorovú (spojitý priestor, rôzne entity) a rastrovú (nespojitý priestor, len body/pixeli)
  - Transformácia je možná obojsmerne, vektor na raster, ako aj raster na vektor
  - Vektorová grafika je škálovateľná bez straty kvality, kým rastrová sa pri zväčšení rozmazáva

15 Základné 2D grafické primitíva a ich atribúty

- Primitíva:

  - Bod
  - Sled bodov
  - Krivka
  - Lomená čiara
  - Grafický text
  - Plocha
  - Vyplnená oblasť
  - Výplňový vzor
  - Všeobecný grafický prvok
- Atribúty:
  - Farba
  - Typ (napr. čiar, písma a pod.)
  - Hrúbka (napr. čiar, písma a pod.)
  - Poloha (napr. písma)
  - Smer vykreslenia (napr. horizontálny, vertikálny atď.)
- Priradenie atribútov:
  - konvenčne - pevne, vedie k nekompatibilite na rôznych zobrazovačoch
  - symbolicky - formou kódu, hovoríme o viazaných (bundled) atribútoch, ktoré sú vzhľadom na zobrazovacie zariadenie nezávislé

16 Spracovanie bodu a sledu bodov v rámci počítačovej grafiky

- Bod:
  - V počítačovej grafike je bod základnou jednotkou, ktorá sa považuje za elementárny (atomárny) objekt
  - Základné atribúty bodu zahŕňajú jeho polohu (v 2D alebo 3D priestore) a farbu, prípadne čas (v prípade heterogénnych štruktúr)
  - Pixel - bod na obrazovke definovaný dvomi súradnicami a farbou, najmenšia jednotka digitálnej rastrovej grafiky
  - Voxel - bod v 3D priestore, definovaný tromi súradnicami a farbou, základná jednotka v objemovej grafike
  - Texel - bod textúry, má definovanú polohu v rámci textúry a vzor, ktorým je textúra vyplnená
  - Na zobrazovacích zariadeniach sú pixely vytvárané prostredníctvom fyzických bodov, ktoré sú rozsvecované, rozlíšenie obrazovky závisí od hustoty týchto bodov, uvádza sa v PPI (Pixels Per Inch) alebo DPI (Dots Per Inch)
- Sled bodov (Polymarker):

  - Množina bodov, definovaná vzťahom medzi atribútmi jednotlivých bodov, operácie ako presúvanie sa vykonávajú nad všetkými bodmi naraz
  - Relácie medzi bodmi polymarkra môžu byť:
    * Homogénne - atribúty bodov polymarkra sú rovnaké
    * Heterogénne - atribúty bodov polymarkra sú rôzne (napr. farba jedného bodu môže závisieť od polohy iného bodu), ak aspoň jedna relácia medzi bodmi je heterogénna, celý polymarker je heterogénny

- Atribúty môžu byť jednotlivým elementom priradené:
  - konvenčne - pevne, vedie k nekompatibilite na rôznych zobrazovačoch
  - symbolicky - formou kódu, hovoríme o viazaných (bundled) atribútoch, ktoré sú vzhľadom na zobrazovacie zariadenie nezávislé

17 DDA algoritmus

- DDA algoritmus (Digital Differential Analyzer):
  - Prírastkový algoritmus používaný na vykresľovanie priamok, založený pripočítavaní konštantných prírastkov k súradniciam bodov priamky (x, y)
  - Efektívny algoritmus, pretože nevyžaduje veľa operácií a je relatívne jednoduchý na implementáciu
- Základné princípy:
  - DDA algoritmus vypočíta všetky body priamky medzi dvoma bodmi $A(x_A, y_A)$ a $B(x_B, y_B)$
  - Vypočíta prírastky pre súradnice x a y na základe rozdielu medzi týmito dvoma bodmi, pričom tieto prírastky sa postupne pripočítavajú k počiatočným hodnotám súradníc

```
def dda_line(xA, yA, xB, yB):
    body_priamky = []  # List pre body priamky
    dx = xB - xA       # Rozdiel medzi x suradnicami
    dy = yB - yA       # Rozdiel medzi y suradnicami

    pocet_krokov = int(max(abs(dx), abs(dy)))  # Pocet krokov je na
    px = dx / pocet_krokov  # Prirastok pre x
    py = dy / pocet_krokov  # Prirastok pre y

    x = xA  # Zacinajuci bod v osi x
    y = yA  # Zacinajuci bod v osi y

    # Iteracia cez pocet krokov
    for i in range(pocet_krokov + 1):
        body_priamky.append((round(x), round(y)))  # Pridanie zaokr
        x += px  # Aktualizacia x
        y += py  # Aktualizacia y

    return body_priamky
```

- Rozlíšenie pre rôzne smernice:
  - Smernica menšia ako 1 ($|dy| < |dx|$):
    - Hlavná zmena prebieha v osi x, prírastky sa vykonávajú prevažne po osi x a menšia zmena nastáva v osi y
  - Smernica väčšia ako 1 ($|dy| > |dx|$):
    - Hlavná zmena prebieha v osi y, prírastky sa vykonávajú prevažne po osi y a menšia zmena nastáva v osi x

18 Bresenhamov algoritmus

- Veľmi efektívny algoritmus na generovanie bodov ležiacich na úsečke, obzvlášť efektívny pri práci s celočíselnými hodnotami
- Na rozdiel od DDA algoritmu Bresenhamov algoritmus využíva predikčný chybový člen, aby určil najbližšie body priamky, pričom minimalizuje potrebu reálnych desatinných výpočtov
- Základné princípy:
  - Bresenhamov algoritmus vykresľuje priamku medzi dvoma bodmi $A(x_A, y_A)$ a $B(x_B, y_B)$ iteratívne, pričom predpovedá, ktorý bod by mal byť najbližší skutočnej úsečke, a aktualizuje hodnoty x a y na základe chybového člena

```
def bresenham_line(xA, yA, xB, yB):
    body_priamky = []  # Zoznam pre body priamky
    dx = abs(xB - xA)  # Rozdiel v x-ovej osi
    dy = abs(yB - yA)  # Rozdiel v y-ovej osi

    # Urcenie krokov podla smernice a kvadrantu
    if xA < xB:
        sx = 1
    else:
        sx = -1

    if yA < yB:
        sy = 1
    else:
        sy = -1

    # Inicializacia chyboveho clena
    errd = dx - dy

    # Iteracia cez body priamky
    while xA != xB or yA != yB:
        body_priamky.append((xA, yA))  # Pridanie aktualneho bodu c

        errd2 = 2 * errd  # Dvojity chybovy clen pre rozhodovanie
        if errd2 > -dy:  # Ak je chyba vacsia ako -dy, aktualizuj x
            errd -= dy
            xA += sx
```

```
            if errd2 < dx:  # Ak je chyba mensia ako dx, aktualizuj y
                errd += dx
                yA += sy

        return body_priamky
```

- Rozlíšenie pre rôzne smernice a kvadranty:
  - Priamky so smernicou menšou ako 1: Algoritmus vykonáva prírastky hlavne v osi x.
  - Priamky so smernicou väčšou ako 1: Algoritmus vykonáva prírastky hlavne v osi y.
  - Kvadranty: Podľa toho, v akom kvadrante sa nachádzajú body, sa upravujú kroky v oboch osiach (smer sx a sy).

19 Spracovanie kružnice a elipsy v rámci počítačovej grafiky a základné metódy ich generovania

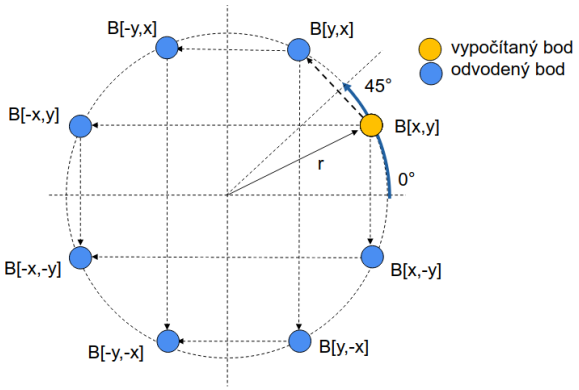
- Kružnica:
  - Množina bodov, ktoré sú rovnaké vzdialené od daného stredu (X_s, Y_s)
  - $(X - X_s)^2 + (Y - Y_s)^2 = r^2$
  - Kružnica môže byť vyjadrená pomocou parametra u (ktorý ide od 0 do 2π alebo 360 stupňov):
    - $X = X_s + r \cdot \cos(u)$
    - $Y = Y_s + r \cdot \sin(u)$
    - (X_s, Y_s) je stred kruhu a r je polomer.
    - Na základe hodnoty parametra u, môžeme vypočítať rôzne body na obvode kruhu
  - Algoritmus kreslenia kruhu podľa predikcie chyby:
    - Algoritmus používa predikciu chyby na iteratívne generovanie bodov kruhu
    - $(X_1 = 0, Y_1 = r)$
    - počiatočna predikcia chyby $E_1 = 1 - r$
      - Ak $E_i < 0$, potom:

$$E_{i+1} = E_i + 2X_i + 3$$
$$X_{i+1} = X_i + 1$$
$$Y_{i+1} = Y_i$$

      - Ak $E_i \geq 0$, potom:

$$E_{i+1} = E_i + 2X_i + 5 - 2Y_i$$
$$X_{i+1} = X_i + 1$$
$$Y_{i+1} = Y_i - 1$$

  - Tento proces pokračuje, až kým sa nevygenerujú všetky body na kruhu



Kreslenie kružnice využitím osovej súmernosti

- Elipsa:
  - Množina bodov, ktoré sú rovnaké vzdialené od dvoch ohnísk $F_1(x_1, y_1)$ a $F_2(x_2, y_2)$
  - $(x - x_s)^2/a^2 + (y - y_s)^2/b^2 = 1$
  - kde (x_s, y_s) je stred elipsy, a je polomer na veľkej osi (polomer v smere x) a b je polomer na malej osi (polomer v smere y)
  - Elipsa môže byť vyjadrená pomocou parametra u (ktorý ide od 0 do 2π alebo 360 stupňov):
    - $x = x_s + a \cdot \cos(u)$
    - $y = y_s + b \cdot \sin(u)$
    - Tento parametrický popis je veľmi podobný ako pri kruhu, ale prispôsobený pre dva rôzne polomery a a b
  - Algoritmus kreslenia elipsy podľa predikcie chyby:
    - Algoritmus vychádza z podobného princípu ako kruhový algoritmus s predikciou chyby
    - Začína sa na hornej časti elipsy ($X_1 = 0, Y_1 = b$) a iteratívne sa generujú body na obvode elipsy
    - počiatočna predikcia chyby $E_1 = b^2 - b \cdot a^2 + a^2/4$
      - Ak $E_i < 0$, potom:

$$E_{i+1} = E_i + b^2 \cdot (2X_i + 1)$$

$$X_{i+1} = X_i + 1$$

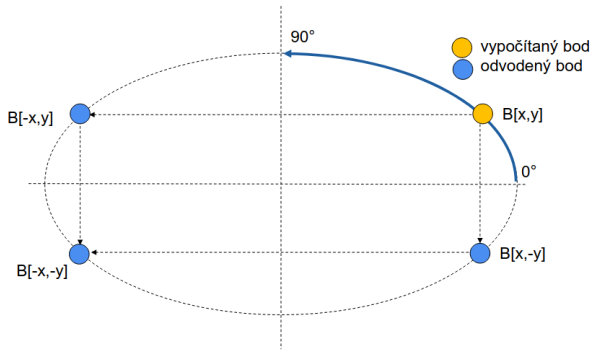
$$Y_{i+1} = Y_i$$

      - Ak $E_i \geq 0$, potom:

$$E_{i+1} = E_i + b^2 \cdot (2X_i + 1) - 2a^2 \cdot Y_i$$

$$X_{i+1} = X_i + 1$$
$$Y_{i+1} = Y_i - 1$$

* Tento proces sa opakuje, kým sa nevygenerujú všetky body na obvode elipsy



Kreslenie elipsy využitím osovej súmernosti

20 Antialiasing

- Technika používaná na zníženie alebo odstránenie viditeľných schodov a zubatých okrajov pri zobrazovaní priamych čiar a hraníc objektov na rastrových displejoch
- Problémy, ktoré rieši:
  - Schodovité okraje - pri zobrazení môžu vzniknúť schody (aliasing), pretože rastrový displej nemá dostatočné rozlíšenie na vyjadrenie hladkých čiar
  - Malé objekty a tenké čiary - pri zobrazení objektov menších ako veľkosť pixelu alebo veľmi tenkých čiar, tieto objekty môžu byť nezreteľné alebo zobrazené ako nepravidelná sekvencia bodov
  - Zložité scény - pri generovaní detailných scén, napríklad pomocou raytracingu, sa malé detaily môžu stať nejasnými alebo skreslenými

21 Filtrovacie a rozptyľovacie metódy, medián filter, poltónovanie, dithering, distribúcia chyby pri rozptyľovaní

- Filtrovacie metódy:
  - Slúžia na určenie farby bodov v obraze
  - Náhodne - farba sa určí náhodne
  - Početnosť farieb - Vyberie sa farba, ktorá sa vyskytuje najčastejšie
  - Spriemernenie - farba sa vypočíta ako priemer hodnôt v oblasti (4-susedstvo, 8-susedstvo)
  - Mediánová funkcia - farba sa vyberie podľa mediánu hodnoty v oblasti (4-susedstvo, 8-susedstvo)
- Rozptyľovacie metódy:
  - Využíva vlastnosť ľudského oka, ktoré zmieša farby z viacerých bodov a vytvorí dojem plynulého prechodu

  - Tieto metódy sa používajú na simuláciu vyšších farebných odtieňov na zariadeniach s obmedzeným farebným rozlíšením
  - Poltónovanie - Halftoning:
    * Poltieňovanie simuluje plynulý prechod medzi odtieňmi využitím bodov a medzier rôznych veľkostí
    * Pri dostatočnej vzdialenosti ľudské oko nedokáže vnímať jednotlivé body a obraz vníma ako celok
  - Dithering:
    * Využíva obmedzené rozlíšenie ľudského oka na vytvorenie ilúzie väčšieho počtu farieb
    * Zblízka sú viditeľné jednotlivé pixely, ale z diaľky sa zlúčia do jedného farebného tónu
- Distribúcia chyby pri rozptyľovaní:
  - Floyd-Steinberg: Koeficient/16
  - Stucki: Koeficient/42
  - Burkes: Koeficient/32
  - Sierra: Koeficient/32
  - Jarvis, Judice, Ninke: Koeficient/48
  - Stevenson, Arce: Koeficient/200
- Algoritmus pre Floyd-Steinberg:
  1. Pre každý bod obrazu nájdeme najbližšiu hodnotu z palety
  2. Po zaokrúhlení hodnoty určíme zanedbanú hodnotu a túto hodnotu rozdelíme medzi susedné body
  3. Tento proces sa opakuje pre každý bod v obraze

22 Popis a reprezentácia objektov v počítačovej grafike, priestor a jeho parametre

- Priestor a jeho parametre:
  - Charakter priestoru:
    * Translačný priestor (T) - určuje pohyb objektov v priestore bez rotácie
    * Rotačný priestor (R) - určuje rotáciu objektov v priestore
    * Kombinovaný priestor (C) - kombinuje translačné aj rotačné operácie
  - Typ dimenzie:
    * Celočíselné (I) - topologické priestory, kde sa používajú celočíselné hodnoty
    * Neceločíselné (F) - nepoužívajú celočíselné hodnoty (napr. reálne čísla)
  - Štruktúra dimenzií:

        - Homogénna štruktúra - uniformný priestor s rovnakým počtom dimenzií
        - Heterogénna štruktúra - rôzne dimenzie v priestore
- Objekty v počítačovej grafike:
    - 0-rozmerný objekt: Bod - základný objekt bez rozmerov
    - 1-rozmerný objekt: Priamka, Úsečka - objekt s jedným rozmerom (dlžka)
    - 2-rozmerný objekt: Plocha - objekt s dvoma rozmermi (šírka a výška)
    - 3-rozmerný objekt: Teleso - objekt s troma rozmermi (dĺžka, šírka, výška)
- Popis a reprezentácia objektov:
    - Pri modelovaní geometrických objektov existujú rôzne spôsoby, ako objekty popísať a reprezentovať
    - Hraničná reprezentácia (B-rep) - objekt reprezentovaný svojimi hranicami
    - Konštruktívna geometria telies (CSG) - objekty modelované pomocou kombinácie základných geometrických tvarov
    - Reprezentácia pomocou bodov - mračná bodov (Point Clouds) používajú množinu bodov na definovanie objektu
    - Drôtový model (Wire Frame Model) - reprezentácia objektu pomocou drôtových rámov, ktoré definujú jeho hrany
    - Povrchový model (Surface Model) - objekt je reprezentovaný svojím povrchom
    - Objemový model (Solid Model) - komplexnejšia reprezentácia, zahŕňa objem objektu

23 Hraničná reprezentácia

- Vychádza z predstavy, že najdôležitejšou časťou telesa sú hraničné elementy (hrany, povrch)
- Na najzákladnejšej úrovni sa hraničná reprezentácia zjednodušuje na hrany a vrcholy (drôtový model)
- Hraničná reprezentácia môže byť nejednoznačná, najmä ak sa objekt s n-rozmermi zobrazuje v n-1-rozmernom priestore (3D objekt v 2D priestore)
- B-REP (Boundary Representation) Metóda:
    - Definuje objekt povrchom, povrch je tvorený stenami, ktoré sa vzájomne dotýkajú iba na spoločných hranách
    - Každá hrana orientovaná, to umožňuje jednoznačne určiť, či ide o vnútornú alebo vonkajšiu hranu
    - B-REP využíva Winged Edge Structure, tvorenú štyrmi druhmi uzlov:
        - Vrchol (Vertex)
        - Hrana (Edge)
        - Stena (Face)

        * Teleso (Solid)

24 Konštruktívna geometria telies

- Metodika pre modelovanie 3D modelov využívajúca ADT strom
- Základné komponenty konštruktívnej geometrie telies CSG:
  - Listy stromu - základné geometrické tvary (kocky, valce, gule), používané na tvorbu komplexných objektov
  - Uzly stromu - operácie základnými tvarmi (zjednotenie, rozdiel, prienik)
  - Hrany stromu - určujú transformácie základných tvarov (rotácia, translácia, škálovanie)
  - Koreň stromu - v koreni je už definovaný celý objekt

25 Súradnicová sústava priestoru, súradnicové sústavy používané v počítačovej grafike, súradnicový reťazec

- Základné parametre súradnicovej sústavy:
  - Počiatok - stred súradnicovej sústavy (origin)
  - Os - smer rozvoja fyzikálnej veličiny v príslušnej dimenzii priestoru
  - Súradnice - parametre, ktoré určujú polohu bodu v priestore
  - Smer rozvoja zoradenia súradníc - Určuje točivosť, zvyčajne ľavo alebo pravotočivá
- Typy súradnicových sústav:
  - Podľa linearity:
    * Lineárne sústavy - osová distribúcia veličín je priamo úmerná
    * Nelineárne sústavy - veličiny sa vyvíjajú nelineárne v závislosti od osí
  - Podľa vzťahu osí:
    * Pravouhlé - os je kolmá
    * Nepravouhlé - os nie je kolmá
- 2D súradnicové systémy:
  - Karteziánska 2D pravouhlá súradnicová sústava - bežný systém so základnými osami X a Y
  - Polárna súradnicová sústava - určuje bod pomocou vzdialenosti a uhla od počiatku (origin)
- 3D súradnicové systémy:
  - Karteziánska 3D pravouhlá súradnicová sústava - bežný 3D systém s osami X, Y a Z
  - Sférická súradnicová sústava - Určuje polohu bodu pomocou vzdialenosti, azimutu a výšky

- Štandardné pracovné priestory:
  - USS (Univerzálna Súradnicová Sústava) - globálna, pre celý priestor
  - SSO (Súradnicová Sústava Objektu) - špecifická pre konkrétny objekt
  - SST (Súradnicová Sústava Textúry) - súradnice textúry v rámci objektu
  - NSS (Normalizovaná Súradnicová Sústava) - sústava, kde sú hodnoty normalizované do jednotkového rozsahu
  - SSC (Súradnicová Sústava Kamery) - súradnice určené vzhľadom na kameru (pohľad na scénu)
  - SSC2 (Dvojkamerová Stereoskopická Súradnicová Sústava) - používa sa v stereoskopickom zobrazení
  - SSZ (Súradnicová Sústava Zariadenia) - logická a fyzická súradnicová sústava zariadenia

26 Transformácie a transformačné zobrazovacie reťazce v rámci počítačovej grafiky

- Typy transformácií podľa schopnosti meniť charakter objektu:
  - Lineárne transformácie:
    - nemení počet a typ parametrov objektu
    - posunutie (translácia), otočenie (rotácia), škálovanie, skosenie, zrkadlenie
  - Nelineárne transformácie:
    - mení počet a typ parametrov objektu
    - distorzia obrazu, rybie oko, panoráma, zošikmenie, face warp
- Typy transformácií podľa schopnosti zachovať sémantiku objektu:
  - Zachovanie sémantiky:
    - Väčšinou lineárne transformácie, kde charakter objektu (napr. tvar) zostáva nezmenený
  - Preklad sémantiky:
    - Väčšinou nelineárne transformácie, kde sa mení aj samotná podstata objektu (napr. vzhľad alebo štruktúra)
- Typy transformácií podľa priestoru, v ktorom sa aplikujú:
  - Filtrovanie (Filtering):
    - Aplikuje sa v priestore farieb, pri aplikácii na pixel sa mení jeho farba
  - Deformovanie (Warping):
    - Aplikuje sa v geometrickom priestore, pri aplikácii na pixel sa mení jeho poloha

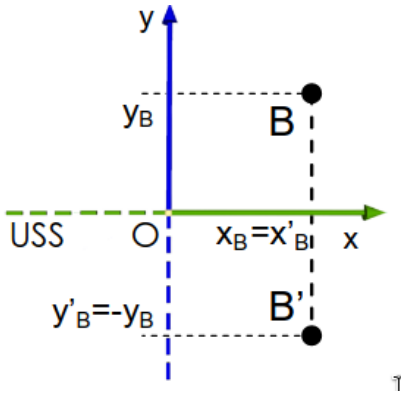
27 Transformácie v rámci zobrazovacích reťazcov (medzisústavové transformácie)), transformačné matice a homogénne súradnice, afinné transformácie

- Medzisústavové transformácie:
  - Transformácie, ktoré sa vykonávajú medzi rôznymi súradnicovými sústavami
  - Globálna (geometrická) transformácia (GT) - aplikovaná na objekt v globálnom priestore
  - Pohľadová transformácia (PT) - upravuje pozíciu objektov vzhľadom na kameru
  - Zobrazovacia (projekčná) transformácia (ZT) - zobrazovanie 3D objektov na 2D plochu
  - Orezávacia transformácia (OT) – určuje, ktoré časti objektu sa zobrazia na obrazovke
- Implementácia transformácií a transformačné matice:
  - Implementácia analytickým spôsobom - priamy výpočet transformovaných hodnôt podľa vzorcov
  - Implementácia pomocou maticového počtu - využíva matice na aplikovanie transformácií,
- Homogénne súradnice:
  - Významným obmedzením 3x3 matic (v 3D) je nemožnosť vyjadriť posunutie (transláciu), homogénne súradnice umožňujú reprezentovať posunutie pomocou matíc
  - V homogénnom priestore sa pridáva ďalšia dimenzia w, ktorá umožňuje prácu so všetkými typmi transformácií vrátane translačných
- Afinné transformácie:
  - Špecifický typ lineárnej geometrickej transformácie, zachováva základné geometrické vlastnosti objektu (rovinnosť, rovnobežnosť a pomer vzdialeností), nezachováva hodnoty atribútov ako uhly alebo dĺžky
  - Zrkadlenie - premietanie objektu cez zrkadlovú os
  - Zmena mierky (škálovanie) - zmena veľkosti objektu
  - Posunutie (translácia) - presunutie objektu z jedného miesta na druhé
  - Skosenie - deformácia objektu, mení uhol medzi osami
  - Otočenie (rotácia) – otáčanie objektu okolo bodu alebo osi

28 Transformácia zrkadlenia

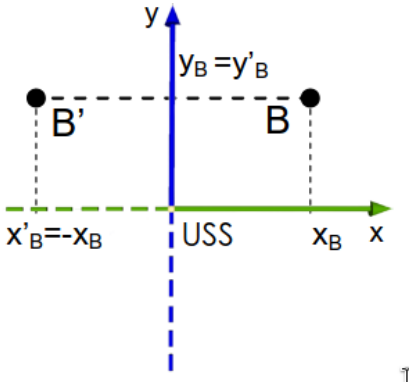
- Zrkadlenie (2D) - analytické vyjadrenie
  - vzhľadom na os x

$$x'_B = x_B$$
$$y'_B = -y_B$$



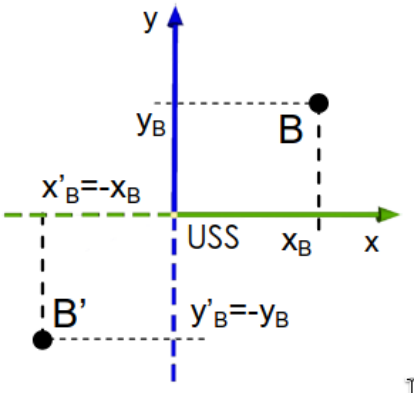
– vzhľadom na os y

$$x'_B = -x_B$$
$$y'_B = y_B$$



– vzhľadom na stred

$$x'_B = -x_B$$
$$y'_B = -y_B$$



- Zrkadlenie (3D) - homogénne matice
  - vzhľadom na os x

$$\mathbf{T}_{Zx} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

– vzhľadom na os xy

$$\mathbf{T}_{Zxy} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

– vzhľadom na stred

$$\mathbf{T}_{Zs} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

29 Transformácia posunutia

- Posunutie (2D)

$$x'_B = x_B + px$$
$$y'_B = y_B + py$$

$$\mathbf{T}_P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ px & py & 1 \end{bmatrix}$$

- Posunutie (3D)

$$x'_B = x_B + px$$
$$y'_B = y_B + py$$
$$z'_B = z_B + pz$$

$$\mathbf{T}_P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ px & py & pz & 1 \end{bmatrix}$$

30 Transformácia zmeny mierky, zväčšenie/zmenšenie rastrového objektu

- Zmena mierky (2D)

$$x'_B = M_x \times x_B$$
$$y'_B = M_y \times y_B$$

$$\mathbf{T}_M = \begin{bmatrix} M_x & 0 & 0 \\ 0 & M_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Zmena mierky (3D)

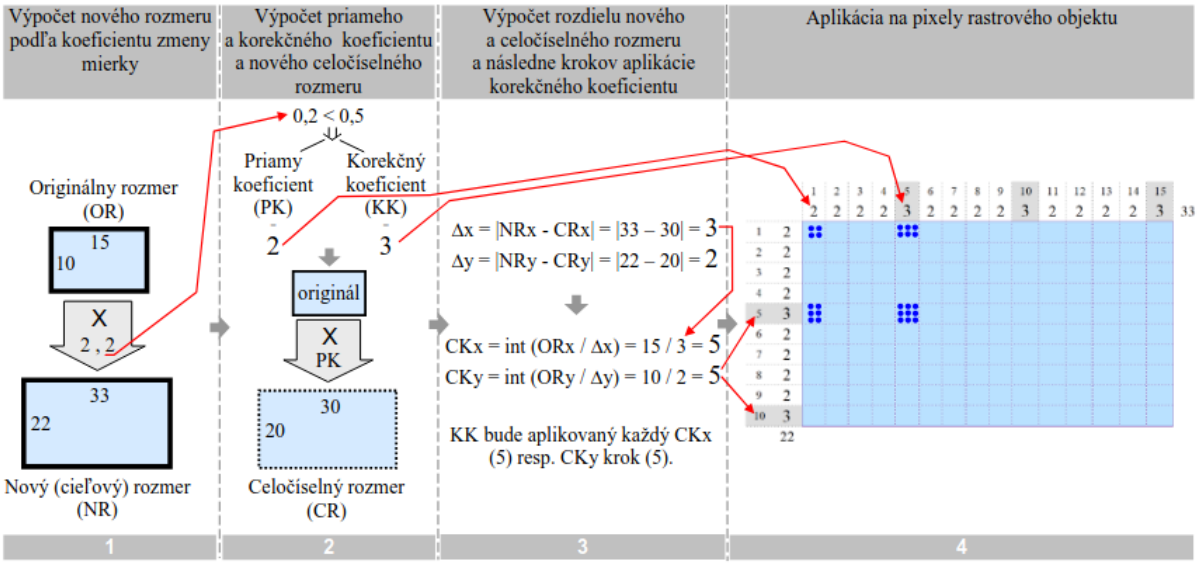
$$x'_B = M_x \times x_B$$
$$y'_B = M_y \times y_B$$
$$z'_B = M_z \times z_B$$

$$\mathbf{T}_M = \begin{bmatrix} M_x & 0 & 0 & 0 \\ 0 & M_y & 0 & 0 \\ 0 & 0 & M_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

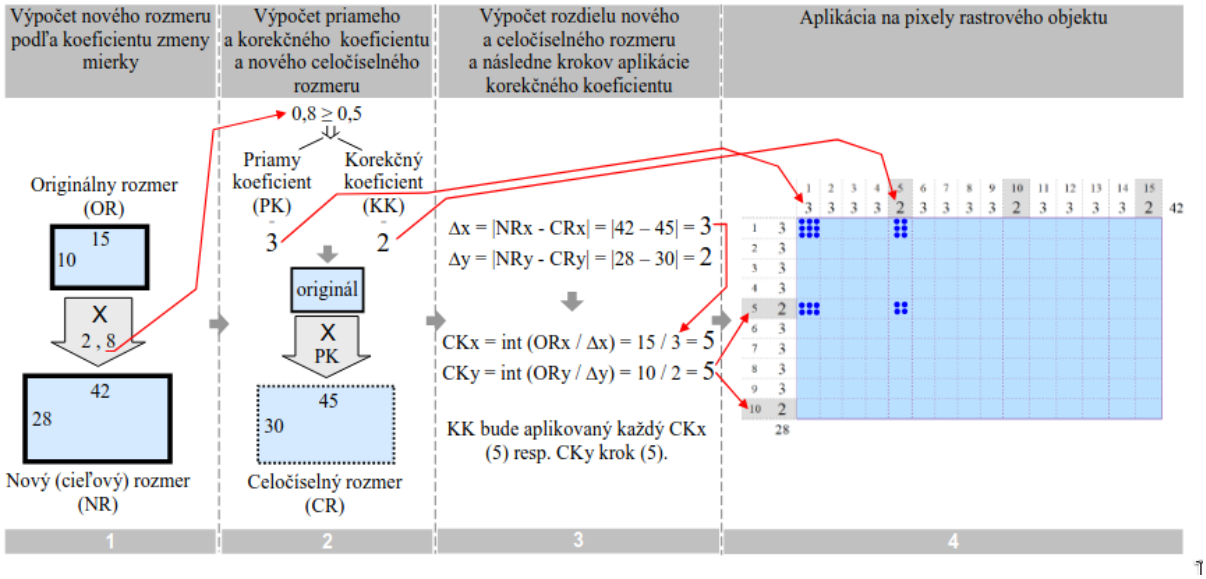
- Zväčšenie rastrového objektu:
  1. Výpočet predpokladaného nového rozmeru (NR) - rozmer rastrového objektu po zväčšení
  2. Určenie koeficientu zmeny mierky - koeficient rozdelený na celočíselnú a desatinnú časť
  3. Test desatinného koeficientu - test, či desatinná časť koeficientu zmeny mierky je väčšia ako 0.5
  4. Výpočet pomocného koeficientu (PK) - PK buď totožný s celočíselnou časťou koeficientu, ak desatinná časť koeficientu M bola $<= 0.5$, alebo o 1 väčší, ak desatinná časť koeficientu M > 0.5. Potom nový korekčný koeficient (KK) je určený presne opačne ako priamy koeficient.
  5. Výpočet rozmeru rastrového objektu (CR) - vypočíta sa nový rozmer objektu, ak by bol koeficient zmeny mierky len celočíselný

6. Zistenie celočíselného rozdielu (Δ) - určí sa, koľko bodov je potrebné doplniť do obrázka na dosiahnutie presného rozmeru.
7. Výpočet korekčného kroku (CK) - korekčný krok určuje kedy sa zmení koeficient zmeny mierky o 1
8. Priradenie koeficientov - konečné priradenie celočíselného koeficientu zmeny mierky a korekčného koeficientu v jednotlivých krokoch

ZVÄČŠENIE, DESATINNÁ ČASŤ KOEFICIENTU <0.5



ZVÄČŠENIE, DESATINNÁ ČASŤ KOEFICIENTU ≥ 0.5



- Zmenšenie rastrového objektu:

1. Výpočet zmeny mierky v opačnom smere - inverzná hodnota koeficientu zväčšenia, tzn. M=1/M, to umožní zistiť, aká matica pixelov (napr. 2x2, 3x3) bude tvoriť subpixel
2. Určenie farby subpixelu - Farba subpixelu sa určuje niekoľkými spôsobmi:
   - Spriemerovanie farieb pixelov - farba sa vypočíta ako priemer farieb pixelov
   - Mediánová funkcia - farby sa vyberie pomocou mediánu hodnôt farieb v matici pixelov
   - Výber najčastejšej farby - vyberie sa farba, ktorá sa v matici pixelov vyskytuje najčastejšie
   - Výber najmenej častej farby - v prípade rovnakého počtu farieb sa vyberie farba, ktorá sa vyskytuje najmenej
   - Výber farby ľavého horného bodu - ak je potrebné, vyberie sa farba ľavého horného pixelu v matici

31 Transformácia skosenia, skosenie rastrového objektu

- Skosenie (2D)
  - v smere osi x

$$x'_B = x_B + S_x \times y_B$$
$$y'_B = y_B$$

$$\mathbf{T}_{Sx} = \begin{bmatrix} 1 & 0 & 0 \\ S_x & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

  - v smere osi y

$$x'_B = x_B$$
$$y'_B = y_B + S_y \times x_B$$

$$\mathbf{T}_{Sy} = \begin{bmatrix} 1 & S_y & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Skosenie (3D)

– v smere osi x vzhľadom na z

$$x'_B = x_B + S_{xoz} \times z_B$$
$$y'_B = y_B$$
$$z'_B = z_B$$

$$\mathbf{T}_{Sxoz} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ S_{xoz} & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

– v smere osi x vzhľadom na y

$$x'_B = x_B + S_{xoy} \times y_B$$
$$y'_B = y_B$$
$$z'_B = z_B$$

$$\mathbf{T}_{Sxoz} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ S_{xoy} & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

– v smere roviny xy

$$x'_B = x_B + S_{xoz} \times z_B$$
$$y'_B = y_B + S_{yoz} \times z_B$$
$$z'_B = z_B$$

$$\mathbf{T}_{Sxy} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ S_{xoz} & S_{yoz} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Skosenie rastrových objektov:
  - Ak je potrebné skosenie v oboch smeroch, musí sa vykonať postupne, najprv v prvom a potom v druhom smere
  - Pri skosení vzniká aliasing, riešený napr. pomocou Bresenhamovho algoritmu

32 Transformácia otočenia

- Otáčanie (2D)

$$x'_B = x_B \times cos\alpha - y_B \times sin(\alpha)$$
$$y'_B = x_B \times sin\alpha + y_B \times cos(\alpha)$$

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

- Otáčanie (3D)
  - na osi x

$$x'_B = x_B$$
$$y'_B = y_B \times cos\alpha - z_B \times sin(\alpha)$$
$$z'_B = y_B \times sin\alpha + z_B \times cos(\alpha)$$

$$\mathbf{T}_{Ox} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha & 0 \\ 0 & -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

  - na osi y

$$x'_B = x_B \times cos\alpha - z_B \times sin(\alpha)$$
$$y'_B = y_B$$
$$z'_B = x_B \times sin\alpha + z_B \times cos(\alpha)$$

$$\mathbf{T}_{Oy} = \begin{bmatrix} \cos\alpha & 0 & \sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

  - na osi z

$$x'_B = x_B \times cos\alpha - y_B \times sin(\alpha)$$
$$y'_B = x_B \times sin\alpha + y_B \times cos(\alpha)$$
$$z'_B = z_B$$

$$\mathbf{T}_{Oz} = \begin{bmatrix} \cos\alpha & \sin\alpha & 0 & 0 \\ -\sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

33 Otáčanie okolo všeobecnej priamky využitím Eulerových uhlov, gimbal lock, quaternióny

- Otáčanie definované Eulerovými uhlami:
  - Rotácia rozložená na tri zložky, každá okolo jednej z osí karteziánskej súradnicovej sústavy
  - Tieto zložky sa potom kombinujú, aby sa dosiahla požadovaná rotácia v priestore
  - Otáčanie reprezentované pomocou všeobecných transformačných matíc, ktoré sa aplikujú postupne na objekt
- Otáčanie okolo všeobecnej priamky:

$$T_P \times T_{Oa} \times T_{Ob} \times T_O \times T_{Ob}^{-1} \times T_{Oa}^{-1} \times TP^{-1}$$

  - T_P je transformácia, ktorá reprezentuje otáčanie okolo priamky
  - T_{Oa} a T_{Ob} sú transformačné matice pre body na priamke
  - T_O je maticová transformácia pre samotný objekt
- Gimbal Lock (Strata stupňa voľnosti):
  - Pri používaní Eulerových uhlov sa môže vyskytnúť problém známy ako gimbal lock, tento problém vzniká, keď sa osi otáčania zhodujú a jedna z osí stráca svoju nezávislosť, čím sa rotácia stáva obmedzená
  - Tento problém je známy z leteckého a vesmírneho priemyslu, kde sa používajú gyroskopy na vytváranie umelého horizontu a stabilizovanie orientácie objektov
- Rotácia pomocou Quaterniónov:
  - Quaternióny sú pokročilým spôsobom reprezentácie rotácií v 3D grafike
  - Na rozdiel od Eulerových uhlov, quaternióny eliminujú problémy ako gimbal lock
  - Quaternión je definovaný ako komplexné číslo:

$$q = \omega + xi + yj + zk$$

  - ω je reálna časť
  - x,y,z sú imaginárne komponenty
  - i,j,k sú imaginárne jednotky, ktoré splňajú pravidlá násobenia:
    - $i * j = -j * i = k$
    - $j * k = -k * j = i$
    - $k * i = -i * k = j$

34 Transformácia otočenia rastrového objektu

- **Princíp otočenia rastrového objektu:**
  - Transformácia otočenia spočíva v otočení každého pixelu obrázka okolo bodu alebo osi rotácie.
  - V praxi sa však manipuluje s celým obrázkom, nie s jednotlivými pixelmi. To znamená, že sa vykonáva rotácia celej matice pixelov, pričom sa musí zabezpečiť správna interpolácia medzi pixelmi, aby sa zachovala kvalita obrazu.
- **Typy otáčania rastrového objektu:**
  - **Priame otáčanie:**
    - Priame otáčanie je matematicky vypočítané pre každý pixel. Keď sa obrázok otočí, môže dôjsť k situácii, že niektoré pixely sa po rotácii dostanú na pozíciu, ktorá neexistuje v mriežke (tj. medzi dvoma existujúcimi pixelmi). Tento problém sa rieši pomocou interpolácie medziľahlých bodov.
    - Rotácia sa zaokrúhľuje, pretože pozície pixelov sú definované v celočíselných hodnotách (pozície sú [Int; Int]), čo môže spôsobiť vznik dier v obraze.
  - **Spätné otáčanie:**
    - Spätné otáčanie zahŕňa definovanie hraníc rastrového objektu a vyplnenie týchto hraníc farbou.
    - Pri tomto type rotácie sa interpolácia vykonáva pre všetky body v obraze, čo znamená, že operácia používa reálnu (float) aproximáciu. Tento spôsob je presnejší, ale môže viesť k problémom pri určovaní farby pixelu, pretože farba je určená na základe najbližšieho pixela v okolí (4-susedovská interpolácia).
- **Interpolácia pri otáčaní:**
  - Pri oboch typoch otáčania je dôležité správne vykonať interpoláciu medzi pixelmi, aby sa zabránilo vzniku "schodovania" (aliasingu) alebo dier v obraze.
  - Pri priamom otáčaní je často použitá bilineárna interpolácia, ktorá vypočíta hodnotu pixelu ako vážený priemer okolitých pixelov.
  - Pri spätnom otáčaní sa pri interpolácii zohľadňuje najbližší pixel alebo priemerná hodnota viacerých pixelov, aby sa vytvoril hladší prechod.

35 Morphing, warping

- Warping:
  - Statická deformácia (transformácia) objektu, aby sa zmenil jeho tvar, nemení však identitu objektu
  - Hodnoty parametrov je možné získať z iného objektu
  - Používa sa napríklad na korekciu perspektívy či mapovanie textúr na 3D objekty
- Morphing:
  - Metóda používaná na plynulú (dynamickú) premenu jedného objektu na druhý

  - Mení identitu vstupného objektu na výstupný
  - Používa sa napríklad pri animáciách alebo vizuálnych efektoch

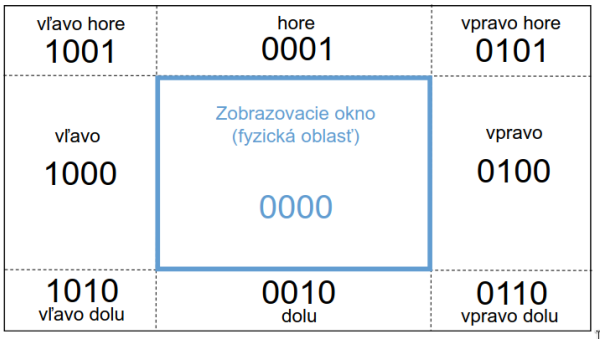
36 2D premietacie transformácie používané v počítačovej grafike, logická a fyzická pracovná oblasť, otáčanie kamery na báze Eulerových uhlov

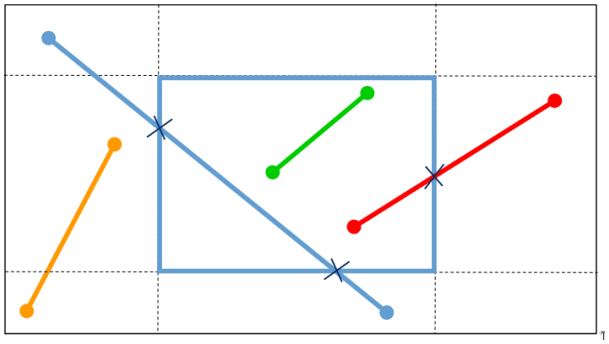
- 2D premietacie transformácie:
  - Ortografická (kolmá) projekcia (rovnobežné lúče, rozmery objektu sa nezmenšujú s hĺbkou):
    * Mongeova projekcia – nárys, pôdorys, bokorys
    * Axonometria:
      · Izometria (všetky osi pod rovnakým uhlom, mierka rovnaká)
      · Dimetria (dve osi majú rovnaké skreslenie)
      · Trimetria (každá os má iné skreslenie)
      · Technická axonometria (projekcia) (šikmá, jedna os v skutočnej mierke)
  - Perspektíva (zbiehavé lúče, rozmery objektu sa zmenšujú s hĺbkou):
    * Jednobodová perspektíva (1 úbežník (bod, kam sa zbiehajú lúče))
    * Dvojbodová perspektíva (2 úbežníky)
    * Trojbodová perspektíva (3 úbežníky)
    * Panoramatická perspektíva (pre veľké uhly pohľadu)
  - Špeciálne projekcie:
    * Sférická projekcia (mapy, XR (VR+AR+MR))
    * Cylindrická projekcia (mapy, panorámy)
    * Konická projekcia (mapy)
- Logická oblasť - virtuálna oblasť (USS)
- Fyzická oblasť - časť logickej oblasti, ktorá je reálne zobrazená zariadením (SSZ)
- Otáčanie kamery na báze Eulerových uhlov:
  - Yaw (Otáčanie okolo zvislej osi) - rotácia v rovine X-Y
  - Pitch (Sklonenie alebo zdvihnutie pohľadu kamery) - otáčanie kamery okolo horizontálnej osi, v rovine Y-Z
  - Roll (Otáčanie okolo vektora pohľadu kamery) - rotácia kamery okolo vlastného pohľadu, v rovine X-Z
  - Celkovú transformáciu rotácie kamery môžeme vyjadriť ako kompozíciu troch rotácií:

$$T_{cam} = T_{yaw} \times T_{pitch} \times T_{roll}$$

37 Orezávacie algoritmy, Cohen-Sutherlandovho algoritmu

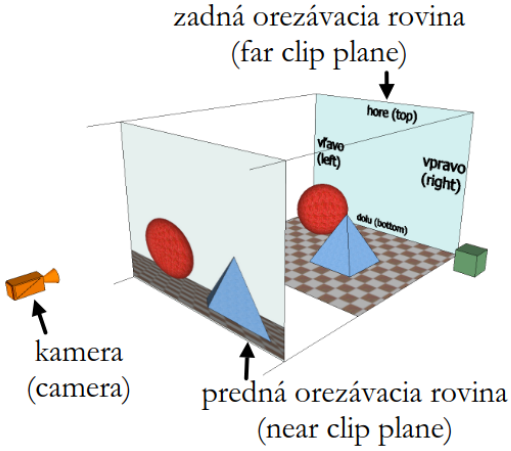
- Najpoužívanejšie orezávacie algoritmy:
  - Cohen–Sutherlandov algoritmus:
    - slúži na orezávanie úsečiek (čiar)
    - funguje na princípe bitových masiek (kódov)
    - používa sa väčšinou v softvérových riešeniach v pravouhlej oblasti
  - Sutherland–Hodgmanov algoritmus:
    - slúži na orezávanie konvexných polygónov
    - funguje na princípe postupného (iteratívneho) testovania každého vrcholu polygónu voči hraniciam orezávacieho okna
    - používa sa väčšinou v softvérových riešeniach v pravouhlej oblasti ale je možné sa stretnúť aj s hardvérovou implementáciou
  - Liang–Barského algoritmus:
    - podobný ako Cohen-Sutherlandov algoritmus na orezanie čiar, používa však menej výpočtov a teda je rýchlejší
    - funguje na princípe použitia parametrického tvaru čiar
    - používa sa väčšinou v softvérových riešeniach ale je možné sa stretnúť aj s hardvérovou implementáciou
  - Weiler–Athertonov algoritmus:
    - podobný ako Sutherland-Hodgmanov algoritmus, ale umožňuje aj spracovanie nekonvexných polygónov
    - používa sa menej, kvôli svojej zložitosti
- Cohen-sutherlandov algoritmus:
  - kódy oboch koncových bodov sú nulové (prázdne) potom oba koncové body ležia vo vnútri zobrazovacieho okna a je možné úsečku vykresliť bez orezania
  - jeden z kódov nenulový, potom je nutné orezanie, pretože časť úsečky určite leží mimo zobrazovacieho okna
  - obidva kódy sú nenulové. Potom sú možné dva prípady:
    - celá úsečka je mimo zobrazovacieho okna a nevykreslí sa. Toto sa deje najmä ak obidva kódy dva príslušné bity rovnaké (napr. 1000 a 1010)
    - časť úsečky je v okne. Toto sa môže stať, ak obidva kódy nemajú dva príslušné bity rovnaké (napr. 1001 a 0010), vtedy je nutné orezanie





38 Kolmá (ortografická) projekcia, Mongeova projekcia

- Kolmá (ortografická) projekcia:
  - Zobrazuje 3D objekty na 2D plochu bez perspektívy, pričom rozmery objektov sa nezmenšujú s hĺbkou, lúče sú rovnomerné a paralelné
  - príklady kolmej projekcie sú Mongeova projekcia a Axonometria



- Mongeova projekcia:
  - Zobrazuje objekt pomocou troch pohľadov: nárys (z prednej strany), pôdorys (zhora) a bokorys (z boku)

39 Axonometria

- Ortografická (kolmá) projekcia (rovnobežné lúče, rozmery objektu sa nezmenšujú s hĺbkou)

- Izometria (všetky osi pod rovnakým uhlom, mierka rovnaká)

$$J^X = J^Y = J^Z$$

$$\alpha = \beta$$

- Dimetria (dve osi majú rovnaké skreslenie)

$$J^X = J^Y$$

$$\alpha = \beta$$

- Trimetria (každá os má iné skreslenie)

$$J^X \neq J^Y \neq J^Z$$

$$\alpha \neq \beta$$

- Technická axonometria (projekcia) (šikmá, jedna os v skutočnej mierke)

$$J^X = J^Y$$

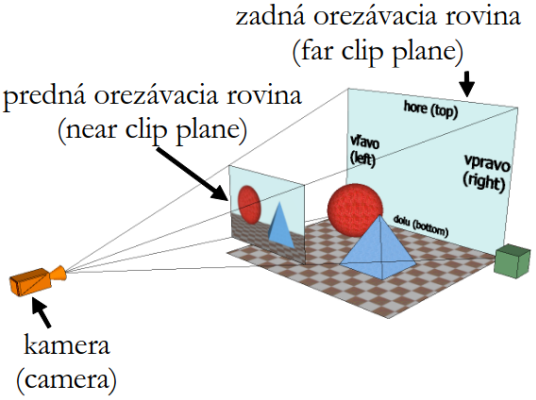
$$J^Z = 0.5 * J^X$$

$$\alpha = 45$$

$$\beta = 0$$

40 Perspektíva

- Simuluje, ako objekty vyzerajú v reálnom svete, keď sa pozorujú z určitého bodu
- Objekty, ktoré sú ďalej od pozorovateľa, sa javia menšie
- Typy:
  - Jednobodová perspektíva - jeden úbežník, všetky rovnobežné línie sa zbiehajú do jedného bodu na horizonte (pohľad priamo na objekt)
  - Dvojbodová perspektíva - dva úbežníky, rovnobežné línie v horizontálnom smere sa zbiehajú do dvoch rôznych bodov (pohľad pod uhlom)
  - Trojbodová perspektíva - tri úbežníky, roviny zbiehajúce sa do troch bodov (pohľad zhora alebo zospodu)

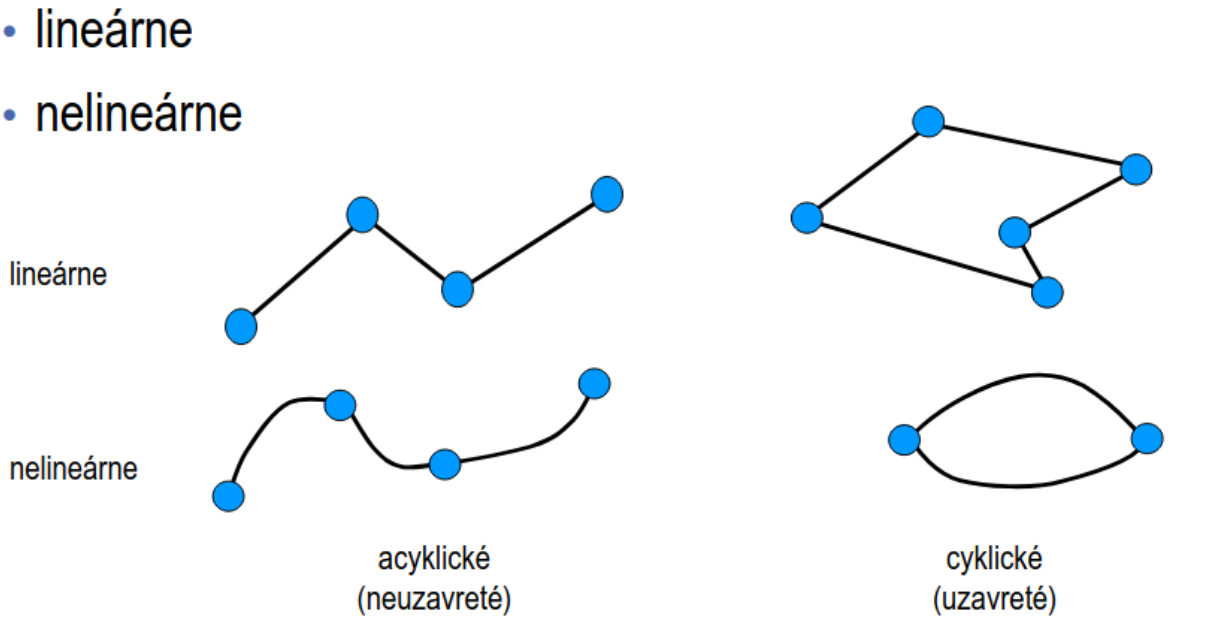


41 Nelineárne premietacie transformácie, distorzia obrazu, rybie oko

- Nelineárne premietacie transformácie spôsobujú optické skreslenia, kde zväčšenie obrazu nie je rovnaké po celej ploche
- Distorzia obrazu (skreslenie obrazu):
  - Distorzia vzniká, keď priečne zväčšenie nie je rovnaké na celej ploche obrazu, závisí aj od konštrukcie šošovky alebo rozptylky, Skreslenie je viditeľné najmä v rôznych vzdialenostiach od optickej osi
  - Barrel distorsion - pri zväčšovaní vzdialenosti od osi kamery vznikajú konvexné krivky
  - Pincushion distorsion - pri zmenšovaní vzdialenosti od osi kamery vznikajú konkávne krivky
- Rybie oko (fisheye):
  - vytvára výrazné skreslenie obrazu, okraje značne rozšírené, objekty sa v strede obrazu javia správne, ale v okrajoch sú veľmi skreslené

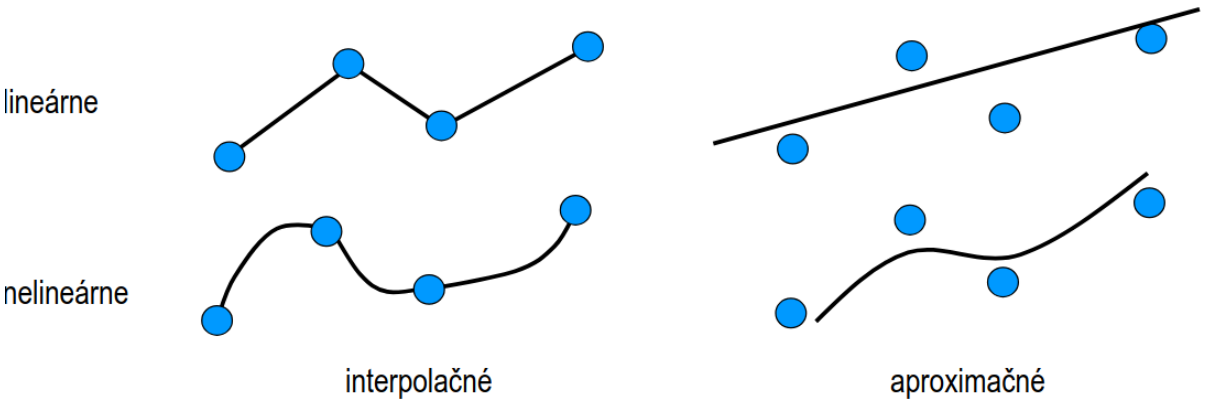
42 Krivky používané v počítačovej grafike, spôsob využitia, 1D krivkové útvary

- Typy 1D krivkových útvarov:
  - Neuzavretá (acyklická) - krivka, ktorá nemá spätné prepojenie, je otvorená
  - Uzavretá (cyklická) - krivka, ktorá je uzavretá a vytvára cyklus
  - Lineárne - krivky s rovnými segmentmi
  - Nelineárne - krivky s plynulými ohybmi, ako Bézierove alebo B-spline krivky



  - Interpolačné - vrcholy sú súčasťou krivky

  - Aproximačné - vrcholy nemusí byť priamo na krivke, ale ovplyvňujú jej tvar



- Používané krivky:
  - Lineárna interpolácia, lomená čiara - jednoduchá krivka, segmenty sú priamky spájajúce vrcholy
  - Bézierove krivky - používané na definovanie plynulých kriviek pomocou riadiacich bodov, môžu byť rôznych stupňov (n)
  - Racionálne Bézierove krivky - špeciálny typ Bézierových kriviek, kde každý riadiaci bod má priradenú váhu
  - B-spline krivky - krivky definované pomocou kontrolných bodov, ktoré môžu byť uniformné alebo neuniformné, B-spline poskytuje väčšiu flexibilitu pri vytváraní hladkých kriviek
  - Kubické B-spline krivky - špeciálny typ B-spline s využitím kubických funkcií, ktoré zabezpečujú hladkú interpoláciu medzi bodmi
- Modifikovateľnosť kriviek:
  - Zmena polohy riadiacich vrcholov - pohybovanie bodov, ktoré určujú tvar krivky
  - Zmena váh riadiacich vrcholov - vplyv váh na tvar racionálnych Bézierových alebo B-spline kriviek
  - Modifikácia uzlového vektora - úprava vplyvu, akým sa uzlový vektor (pre B-spline) mení na celkový tvar krivky

43 Fergusonova krivka

- Nelineárna interpolácia používaná v počítačovej grafike na modelovanie plynulých kriviek
- Definuje sa štyrmi bodmi, ktoré určujú orientáciu krivky medzi bodmi:
  - dvoma riadiacimi bodmi
  - dvoma smerovými vektormi
- Krivka zaručuje plynulý prechod s hladkými prvými a druhými deriváciami
- Vhodná na vytváranie plynulých trajektórií

44 Bézierove krivky

- Aproximačné krivky vyvinuté Pierre Bézierom v 60. rokoch pre Renault
- Vlastnosti:
  - aproximačná
  - interpoluje koncové vrcholy
  - definuje ju polynomiálna funkcia stupňa n, kde n je počet riadiacich bodov mínus 1
  - leží v konvexnom obale riadiacich vrcholov
  - pseudolokálna kontrola - pohyb jedného riadiaceho bodu ovplyvní tvar krivky iba v lokálnych oblastiach
  - afinná invariancia - po aplikácii afinných transformácií (napr. posunutie, škálovanie) tvar krivky zostáva zachovaný
  - pre Bézierovu krivku stupňa n platí, že ide o polynóm n-tého stupňa

45 Spline, Catmull-Rom spline a B-spline krivky

- **Spline krivka**
  - Spline krivka je funkcia stupňa m pre $n+1$ bodov $X_i = (x_i, y_i)$ (kde $i = 0 \dots n$ a $x_0 < x_1 < \dots < x_n$)
  - Krivka je definovaná ako:
    - $f(x) = f_k(x)$ na intervale $< x_k, x_{k+1} >$, kde f_k je polynóm stupňa m
    - Krivka má spojité derivácie až do stupňa $m - 1$
  - Najčastejšie sa používajú **kubické spline funkcie** (s $m = 3$)
- **Catmull-Rom spline**
  - Interpolačná krivka - prechádza cez všetky riadiace body P_i
  - C1 kontinuita - nemá diskontinuity v smere a veľkosti dotyčnice, ale druhá derivácia je **lineárne interpolovaná**, zakrivenie sa mení lineárne
  - Body segmentu môžu ležať mimo domény medzi bodmi P_1 a P_2
  - Aj keď segment je definovaný pomocou 4 riadiacich bodov, môže mať **ľubovoľný počet ďalších riadiacich bodov**
- **B-spline krivka**
  - B-spline je generalizácia Bézierovej krivky
  - používa jednoduchšie Coonsove polynómy
  - **Vlastnosti B-spline krivky**:
    - **Aproximačná a uniformná**
    - Krivka začína v bode P_0 a končí v bode P_{L+n-1}, kde n je počet riadiacich bodov a L je počet násobností riadiacich bodov

    - **Pseudolokálna kontrola a segmentovateľnosť**.
    - **Afinná invariancia** - tvar krivky zostáva zachovaný po aplikácii afinných transformácií

- **Porovnanie Bézierovej a B-spline krivky**
  - Bézierová krivka je stupňa n, zatiaľ čo B-spline krivka je zložená z $n-1$ segmentov, pričom každý segment je popísaný polynómom tretieho stupňa.
  - **Nevýhoda B-spline kriviek**: Krivka nezačína ani nekončí v pôvodných vrcholoch polygónu, čo môže byť upravené zmenou násobnosti uzlového vektora.

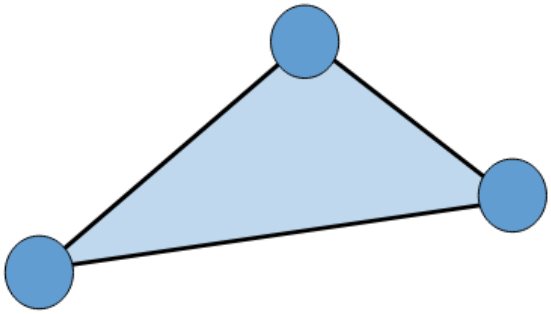
46 NURBS krivky

- **Uniformnosť a neuniformnosť**
  - **Uniformná uzlová postupnosť** - ak platí $u_{i+1} - u_i = kontanta$ pre všetky i
  - **Neuniformná uzlová postupnosť** - ľubovoľná neklesajúca postupnosť uzlov
- **NURBS - Nonuniform Rational B-Spline:**
  - Skratka pre **neuniformný racionálny B-spline**.
  - Zahrňuje vlastnosti B-spline s pridaním neuniformnosti a racionálnosti
  - **Neuniformnosť** - intervaly medzi jednotlivými uzlami nemusia byť rovnaké
  - **Racionálnosť** - každému riadiacemu bodu je priradená váha, ktorá určuje, aký vplyv bude mať daný bod na krivku
- **Vlastnosti NURBS kriviek:**
  - **Zovšeobecnenie** - zovšeobecnenie nižších kriviek, ako napríklad Bézierovej krivky alebo neracionálnej B-spline krivky, ktoré sú len špeciálnymi prípadmi NURBS.
  - **Lokálnosť** - zmena váhy alebo polohy riadiaceho bodu ovplyvní iba úsek krivky odpovedajúci príslušným úsekom uzlového vektora
  - **Diferencovateľnosť** - racionálna bázická funkcia je vo vnútri svojho uzlového intervalu nekonečne veľa krát spojito diferencovateľná, ak je menovateľ rôzny od nuly
  - **Konvexný obal** - krivky ležia v konvexnom obale bodov $P_{i-p}, \ldots, P_i$.

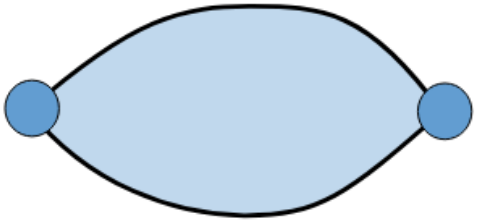
47 Plochy používané v počítačovej grafike, 2D plošné útvary, modifikovateľnosť plôch

- **Plochy používané v počítačovej grafike**
  - Plochy dané analytickým popisom
  - Interpolačné plochy
  - Aproximačné plochy
- **Plošné 2D útvary - typy**
  - 2D útvar definovaný vrcholmi a hranami – polygón:

    * Lineárny



    * Nelineárny



- **Používané plochy**
  - **Rovinné (pravítkové) plochy** (lineárne)
  - **Bézierove plochy** (nelineárne)
  - **B-spline plochy** (nelineárne)
  - **Racionálne B-spline plochy** (nelineárne):
    * Uniformné
    * Neuniformné
- **Modifikovateľnosť plôch**
  - Zmena polohy riadiacich vrcholov
  - Zmena váh riadiacich vrcholov
  - Modifikácia uzlových vektorov

48 Coonsova bilineárna plocha

- **Bilineárna Coonsova plocha**
  - Ak protiľahlé strany sú úsečky, dostaneme tzv. priamkovú (pravítkovú) plochu
  - Coonsova bilineárna plocha je teda všeobecnejšia ako plocha priamková

49 Bézierová bikubická plocha

- **Bézierová bikubická plocha**
  - Základná Bézierova plocha, nazývaná aj Bézierova bikubická plocha, je daná maticou 4x4 bodov, teda 16-imi uzlami $B_{ij},\ i,j = 0,1,2,3$.
  - Plocha je definovaná explicitnou rovnicou:

$$z = [E(u), F(u), G(u), H(u)] \cdot B \cdot [E(w), F(w), G(w), H(w)]$$

- **Kubické Bernsteinove polynómy**
  - $E(t) = (1-t)^3$
  - $F(t) = 3t(1-t)^2$
  - $G(t) = 3t^2(1-t)$
  - $H(t) = t^3$

50 Problém riešenia viditeľnosti a jeho kategorizácia v rámci počítačovej grafiky

- **Riešenie viditeľnosti (Visibility)**
  - Spočíva v odstránení (resp. odlíšení) tých častí trojrozmerných objektov, ktoré pri danom premietaní do 2D nie sú z miesta pozorovateľa viditeľné.
  - Týmto spôsobom sú tieto časti "zakryté" a dostávame jednoznačné priemety želaných telies.
- **Riešenie viditeľnosti - Kategorizácia**
  - **Podľa priestoru, kde je viditeľnosť riešená:**
    - riešenie v 3D
    - riešenie v 2D priemete
  - **Podľa reprezentácie objektov, ktorých viditeľnosť riešime:**
    - objektovo orientované algoritmy - kde sa rieši, ktorá časť príslušného objektu je viditeľná
    - obrazovo orientované algoritmy - kde sa rieši spätne pre každý obrazový bod, ktorý objekt je v ňom vidieť
  - **Podľa toho, či sa uvažuje aj osvetlenie telesa:**
    - riešenie bez osvetlenia - vyhodnotenie farieb je aplikované lokálne na každý objekt
    - riešenie s osvetlením - sem patrí aj metóda sledovania lúča (raytracing) alebo vyžarovacia metóda (radiosity), ktoré sa častokrát označujú ako globálne metódy riešenia viditeľnosti s globálnou aplikáciou farieb v rámci scény (napr. aj odrazy)
  - **Podľa vplyvu možnej chyby pri vykonávaní:**
    - s lokálnym vplyvom chyby na výsledok

    - s globálnym vplyvom chyby na výsledok
  - **Podľa času potrebného na riešenie viditeľnosti:**
    - riešenie mimo reálneho času
    - riešenie v reálnom čase

51 Algoritmus plávajúceho horizontu

- **Popis:** Algoritmus plávajúceho horizontu je technika používaná v počítačovej grafike na riešenie problému viditeľnosti v 3D grafike, najmä pri renderovaní scén. Tento algoritmus sa využíva najmä v kontexte vizualizácie terénov alebo scény s viacerými objektmi, kde je potrebné efektívne zobraziť len tie časti scény, ktoré sú viditeľné zo stanoviska kamery.
- **Stručný popis:** Algoritmus plávajúceho horizontu sa zameriava na dynamické určovanie viditeľných oblastí vo 3D priestore pomocou horizontálnej priesečnej plochy, ktorá sa pohybuje alebo "pláva" v závislosti na výškach objektov alebo terénov. Tento priesečný horizont definovaný určitou výškou sa môže posúvať po scéne a určuje, ktoré objekty sú viditeľné z aktuálneho pohľadu kamery.
- **Hlavný princíp:** Plávajúci horizont postupne "otvára" alebo "uzatvára" zobrazenie objektov tým, že s každým posunom vyhodnocuje, ktoré časti scény sú skryté alebo odhalené. Tento algoritmus zvyčajne využíva metódu skenovania, kde sa postupne prechádza cez objekty a vyhodnocuje sa, ktoré sú na danom horizontálnom "priesečníku" viditeľné.
- **Výhody:** Tento algoritmus je efektívny, pretože umožňuje zobrazenie len tých častí terénu alebo objektov, ktoré sú relevantné, čím šetrí výpočtové zdroje a zvyšuje výkon renderovania.
- **Použitie:** Algoritmus sa často používa v aplikáciách, ako sú simulácie, hry, alebo iné aplikácie, kde sa renderujú veľké 3D scény a je potrebné optimalizovať výkon pri zobrazovaní terénov alebo rozsiahlych scén.

52 Freeman-Lotrelov algoritmus riešenia viditeľnosti

- **Algoritmus pracuje v 3D** a je založený na rozdelení stien na:
  - neviditeľné,
  - potenciálne viditeľné.
- Rozhodovanie sa zakladá na uhlu φ medzi vektorom pohľadu kamery k a normálou steny n.
- Kreslenie sa vykonáva zozadu dopredu len z množiny potenciálne viditeľných stien.
- Je potrebné zabezpečiť jednotnú orientáciu všetkých stien v scéne (v smere alebo proti smeru hodinových ručičiek).

53 Freeman-Lotrelov algoritmus riešenia viditeľnosti

- **Algoritmus pracuje v 3D** a je založený na rozdelení stien na:
  - neviditeľné,
  - potenciálne viditeľné.

- Rozhodovanie sa zakladá na uhlu φ medzi vektorom pohľadu kamery k a normálou steny n.
- Kreslenie sa vykonáva zozadu dopredu len z množiny potenciálne viditeľných stien.
- Je potrebné zabezpečiť jednotnú orientáciu všetkých stien v scéne (v smere alebo proti smeru hodinových ručičiek).

54 Algoritmus pamäte hĺbky (Z-buffer)

- **Výhody:**
  - Korektne rieši viditeľnosť.
  - Polygóny (mnohouholníky) sa môžu navzájom pretínať.
  - Polygóny je možné kresliť v ľubovoľnom poradí (nevyžaduje žiadne predspracovanie ani triedenie polygónov).
  - Vhodný pre objekty s množstvom malých polygónov.
- **Nevýhody:**
  - Vysoké nároky na pamäť, minimálne $2 \times maxX \times maxY$ bajtov.
  - Prekresľovanie.
  - Je pomalý (málo efektívny) pri veľkých scénach (veľa veľkých polygónov), ale je ho možné urýchliť použitím orezania na zorný ihlan.

55 Metóda BSP stromov pri riešení viditeľnosti

- **BSP Stromy (Binary Space Partitioning):**
  - **Podľa rozloženia objektov:**
    - Vyvážené
    - Nevyvážené
  - **Podľa rozloženia deliacich rovín:**
    - Statické
    - Dynamické
    - Adaptívne
    - Stratové
- **Postup tvorby BSP stromu:**
  - Zvoľ ľubovoľnú úsečku/plochu zo vstupnej množiny ako deliacu.
  - Rozdeľ ostatné úsečky/plochy podľa deliacej na dve množiny L (ľavé) a R (pravé):
    - Množina L obsahuje úsečky/plochy, ktoré ležia celé naľavo od deliacej.
    - Množina R obsahuje úsečky/plochy, ktoré ležia celé napravo od deliacej.
  - Ak niektorá úsečka/plocha pretína deliacu, rozdeľ ju tak, aby jej časti bolo možné priradiť do jednej z množín L alebo R.

  - Rekurzívne opakuj pre množiny L a R, kým nie sú prázdne.
  - Ak sú množiny prázdne, pokračuj alebo skonči.
- **Prechod BSP stromom:**
  - **Prechod zozadu dopredu:**
    * Začni prechádzať strom od koreňa.
    * Na ktorej strane deliacej priamky/plochy uzla je pozorovateľ?
      · Ak vľavo - prejdi pravý podstrom, vykresli úsečku/plochu v uzle, prejdi ľavý podstrom
      · Ak vpravo - prejdi ľavý podstrom, vykresli úsečku/plochu v uzle, prejdi pravý podstrom
  - **Prechod spredu dozadu:**
    * Začni prechádzať strom od koreňa.
    * Na ktorej strane deliacej priamky/plochy uzla je pozorovateľ?
      · Ak vľavo - prejdi ľavý podstrom, vykresli úsečku/plochu v uzle, prejdi pravý podstrom
      · Ak vpravo - prejdi pravý podstrom, vykresli úsečku/plochu v uzle, prejdi ľavý podstrom

56 Metóda oktantových stromov pri riešení viditeľnosti

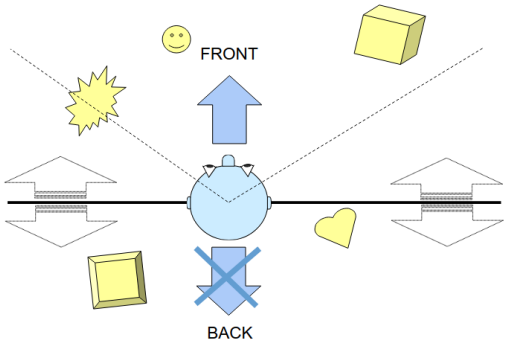
- **Oktantové stromy:** Oktantové stromy sú metóda používaná na rozdelenie 3D priestoru do menších oblastí, nazývaných oktanty, a sú využívané na riešenie problému viditeľnosti v počítačovej grafike. Každý uzol v oktantovom strome reprezentuje časť 3D priestoru, a každý listový uzol predstavuje malú časť tohto priestoru, kde sa vykonáva analýza viditeľnosti.
- **Delenie priestoru:** Pri použití oktantových stromov sa 3D priestor delí na 8 menších častí alebo oktantov. Tento proces delenia pokračuje rekurzívne, čím sa vytvára strom, ktorý efektívne reprezentuje celý 3D priestor.
- **Adaptívne delenie:** V prípade adaptívneho delenia sa priestor delí na menšie časti len tam, kde je to potrebné. To znamená, že nie všetky časti priestoru sú delené rovnako, ale delenie závisí od hustoty objektov alebo od miesta, kde je potrebné riešiť viditeľnosť detailnejšie.
- **Výhody a nevýhody:**
  - **Výhody:** Oktantové stromy umožňujú efektívne zúženie priestoru, čo urýchľuje výpočty týkajúce sa viditeľnosti. Adaptívne delenie znižuje nároky na pamäť tým, že sa sústredí na relevantné oblasti scény.
  - **Nevýhody:** Adaptívne delenie môže byť náročné na implementáciu a môže vyžadovať dodatočné výpočty na optimalizáciu delenia priestoru.
- **Použitie v počítačovej grafike:** Oktantové stromy sú veľmi užitočné pri riešení viditeľnosti v komplexných 3D scénach, kde sa musí analyzovať, ktoré objekty sú

viditeľné z určitého bodu pohľadu. Pomáhajú pri optimalizácii týchto výpočtov, čím umožňujú rýchlejšie vykresľovanie scény.

- **Príklad delenia priestoru:**
  - Predstavte si 3D priestor rozdelený na 8 oktantov, pričom každý oktant môže byť ďalej rozdelený, ak je to potrebné. Tento proces pokračuje, až kým sa priestor nedelí na dostatočne malé časti, ktoré sú vhodné na vykonanie analýzy viditeľnosti.
  - Adaptívne delenie zabezpečuje, že oblasti s vysokou hustotou objektov budú rozdelené na menšie časti, zatiaľ čo oblasti s nízkou hustotou môžu zostať väčšie, čím sa šetrí čas a pamäť.

57 Urýchľovacie metódy pre riešenie viditeľnosti v počítačovej grafike

- **FV (Front View) / BC (Back Cut) algoritmus:**
  - Algoritmus FV/BC sa zameriava na urýchlenie výpočtov viditeľnosti tým, že predbežne identifikuje a orezáva časti objektov, ktoré nie sú viditeľné z pohľadu pozorovateľa. Tento prístup využíva informácie o prednej (FV) a zadnej (BC) strane objektov.
  - **FV (Front View)**: Predstavuje pohľad na objekty, ktorý zobrazuje len prednú časť objektu, pričom sa zohľadňuje, ktoré objekty sú viditeľné z pozorovateľovho bodu pohľadu.
  - **BC (Back Cut)**: Tento krok slúži na odstránenie zadných častí objektov, ktoré sú zakryté inými objektmi a teda nie sú viditeľné pre pozorovateľa. Tento krok urýchľuje výpočty, pretože sa znižuje počet objektov, ktoré musia byť zobrazené.
  - Cieľom je minimalizovať počet objektov, ktoré je potrebné vykresliť, a tým výrazne znížiť výpočtovú náročnosť.

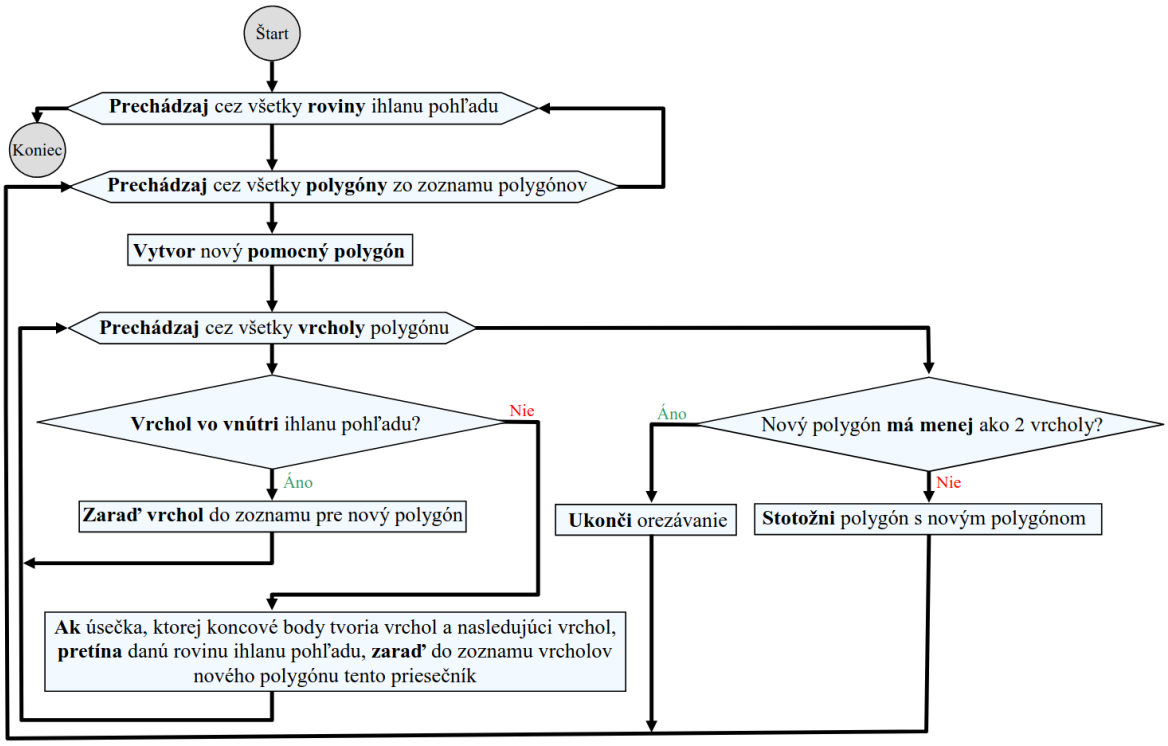


- **Orezávanie na zorný ihlan:** Táto metóda spočíva v orezaní objektov, ktoré sa nachádzajú mimo zorného poľa pozorovateľa, definovaného zorným ihlanom (view frustum). Pomáha eliminovať objekty, ktoré sa nachádzajú mimo pohľadu a nemusia byť renderované.
- **Ohraničujúce objekty:** Používa sa na zjednodušenie analýzy viditeľnosti tým, že objekty v scéne sú ohraničené virtuálnymi objektmi (bounding boxes), ktoré zjednodušujú výpočty tým, že sa pracuje s jednoduchšími geometrickými tvarmi.

- **Sektorovanie:** Pri sektorovaní sa scéna rozdelí na sektory, pričom každý sektor obsahuje objekty, ktoré sú medzi sebou vzájomne viditeľné. To umožňuje rýchlejšie zistenie viditeľnosti, pretože sa analyzujú len objekty v rovnakom sektore.
- **Potenciál viditeľnosti (PVS):** Potenciál viditeľnosti je technika, ktorá predpočítava, ktoré objekty budú pravdepodobne viditeľné z rôznych pohľadov alebo miest v scéne. Tým sa predíde zbytočným výpočtom viditeľnosti v reálnom čase.
- **S-buffer:** S-buffer je technika na uchovávanie informácií o viditeľnosti v pamäti, ktorá umožňuje rýchlejšie zisťovanie viditeľnosti objektov. Pomáha urýchliť renderovanie tým, že eliminuje potrebu opakovaného výpočtu viditeľnosti pre každý pixel.

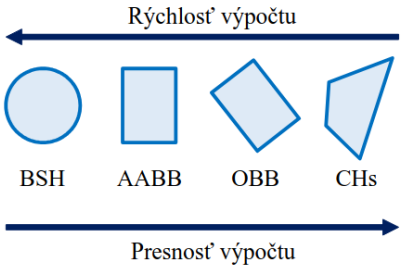
58 Orezávanie na zorný ihlan pri riešení viditeľnosti v počítačovej grafike

- **Zorný ihlan:**
  - Zorný ihlan je trojrozmerná oblasť definovaná zorným poľom (Field of View, FOV) a pomerom strán obrazovky (H/V FOV). Tento priestor obsahuje všetky objekty, ktoré sú v zornom poli pozorovateľa. Po aplikácii projekčných a zobrazovacích transformácií (USS $\rightarrow$ SSC $\rightarrow$ SSZ) sa vytvorí obdĺžnik na obrazovke, ktorý reprezentuje časť priestoru, ktorá bude viditeľná na zobrazenom obraze.
  - Orezávanie na zorný ihlan sa robí s cieľom eliminovať objekty, ktoré sú mimo tohto priestoru, čím sa optimalizuje výkon renderovania.
- **Orezávanie na zorný ihlan - algoritmus:**
  - Algoritmus pre orezávanie na zorný ihlan prechádza cez všetky roviny ihlanu pohľadu a na základe priesečníkov s polygónmi určuje, ktoré časti objektov sú viditeľné a ktoré nie.
  - **Postup:**
    1. **Štart:** Začneme s prehľadávaním všetkých rovín zorného ihlanu pohľadu.
    2. Prechádzame všetkými rovnými plochami (rovinami) zorného ihlanu pohľadu.
    3. Ak nie sú žiadne ďalšie roviny na spracovanie, algoritmus končí.
    4. Ak sú ešte ďalšie roviny, prechádzame zoznamom polygónov a analyzujeme ich vrcholy.
    5. Pre každý polygón vytvárame nový pomocný polygón a prechádzame jeho vrcholy.
    6. Ak vrchol leží vo vnútri zorného ihlanu, zaradíme ho do nového polygónu.
    7. Ak úsečka medzi dvoma vrcholmi pretína rovinu ihlanu, zaradíme bod priesečníka do nového polygónu.
    8. Pokračujeme v prechádzaní všetkými vrcholmi polygónu.
    9. Ak nový polygón má menej ako dva vrcholy, orezávanie sa ukončuje.
    10. Inak, nový polygón sa stotožní s pôvodným a pokračujeme na ďalší polygón.

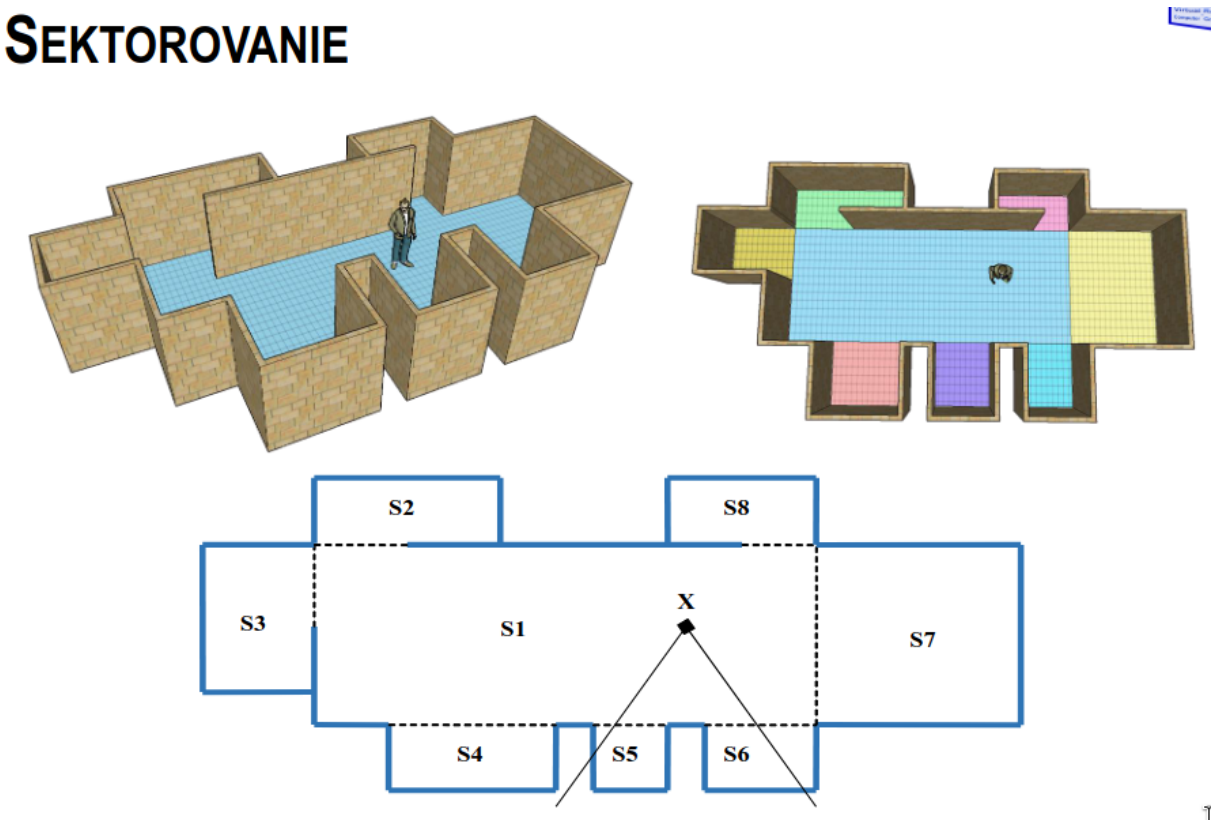


59 Ohraničujúce objekty, sektorovanie a potenciál viditeľnosti pri urýchľovaní riešenia viditeľnosti v počítačovej grafike

- **Ohraničujúce objekty:**
  - Metóda ohraničujúcich objektov sa zameriava na zrýchlenie výpočtov viditeľnosti tým, že namiesto výpočtu viditeľnosti celého zložitého objektu sa použije geometricky jednoduchší objekt, ktorý ohraničuje pôvodný objekt.
  - Ak je ohraničujúci objekt viditeľný, môžeme s istotou tvrdiť, že aj pôvodný objekt je viditeľný. Táto technika sa tiež využíva pri detekcii kolízií v 3D prostredí.
  - Rôzne typy ohraničujúcich objektov:
    - **Hierarchia ohraničujúcich gulí (BSH)**: Používa hierarchickú štruktúru gulí na ohraničenie objektov, pričom každá guľa obsahuje viacero objektov alebo podobjektov.
    - **Osovo-orientovaný kváder (AABB)**: Jednoduchý kváder (pravoúhly box), ktorý je orientovaný podľa osí koordinátovej sústavy a obklopuje objekt.
    - **Objektovo-orientovaný kváder (OBB)**: Kvadr, ktorý sa orientuje podľa orientácie samotného objektu, čo poskytuje presnejšie ohraničenie než AABB.
    - **Konvexná obálka (Convex Hull)**: Je to najmenší konvexný objekt, ktorý obsahuje celý daný objekt. Tento prístup je veľmi efektívny pri určení základného ohraničenia objektu.

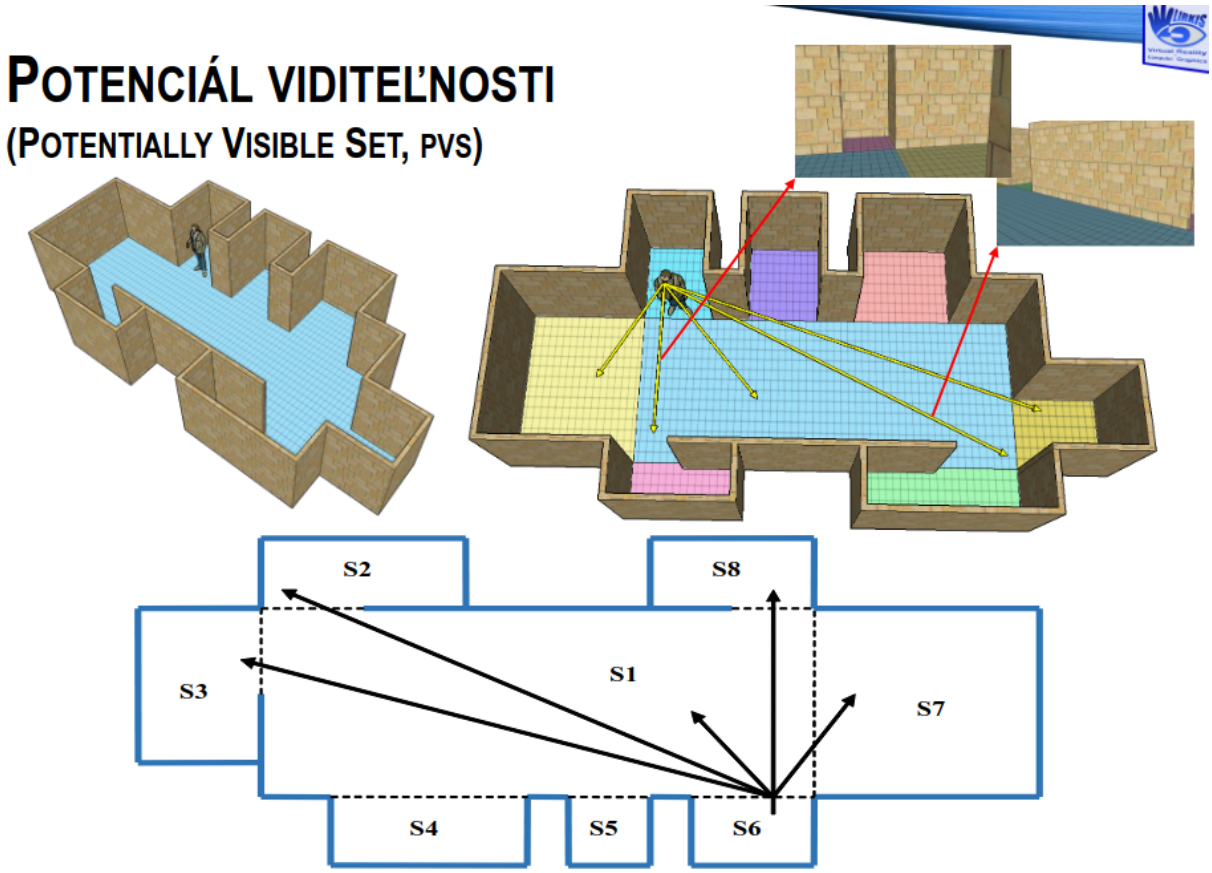


- **Sektorovanie:**
  - Sektorovanie rozdeľuje scénu do samostatných oblastí alebo sektorov, pričom každý sektor obsahuje objekty, ktoré sú navzájom viditeľné. Tento prístup umožňuje efektívnejšie zisťovanie viditeľnosti, pretože sa zameriavame len na objekty v rovnakom sektore.
  - Tento prístup minimalizuje množstvo objektov, ktoré je potrebné skontrolovať, čo vedie k zlepšeniu výkonu pri renderovaní.



- **Potenciál viditeľnosti (PVS):**
  - Potenciál viditeľnosti (PVS) je technika, ktorá sa zameriava na predpočítanie viditeľných objektov z rôznych pohľadov alebo oblastí v scéne.
  - PVS pomáha predchádzať zbytočným výpočtom viditeľnosti v reálnom čase, pretože už máme vopred pripravený zoznam objektov, ktoré sú pravdepodobne viditeľné z určitého pohľadu.

– Týmto spôsobom je možné znížiť výpočtovú náročnosť pri renderovaní komplexných scén.



60 S-buffer pri riešení viditeľnosti v počítačovej grafike

- **S-buffer (Span buffer):**
  - S-buffer je technika používaná na zabezpečenie neprekresľovania už vykreslených polygónov v rámci scény, ktorá sa zobrazuje spredu dozadu. Algoritmus S-buffer prioritne pracuje v 2D, a v 3D prostredí sa ukladá hĺbka iba na začiatku a na konci spanu, pričom medzi týmito hodnotami sa interpoluje.
  - S-buffer je efektívny pri vykresľovaní scenérií, kde je potrebné rýchlo zistiť, ktoré časti už boli vykreslené, a neprekresľovať ich zbytočne.
- **Údajové štruktúry pre S-buffer:**
  - Údajové štruktúry používané pri S-buffer sú rôzne, ale najčastejšie sa využíva spojkový zoznam (linked list). Tento zoznam obsahuje časti/úseky riadkov, ktoré už boli vykreslené.
  - Zoznamy môžu byť zostavené kontinuálne alebo osobitne pre každý riadok obrazovky. Smerníky na prvý span v každom riadku sú uložené v hašovacej tabuľke, čo zefektívňuje prístup k jednotlivým spanom.
- **Spracovanie pri S-buffer:**
  - Pri kreslení polygónu (mnohouholníka) sa každý span podrobuje testu, či nie

je prekrytý iným už vykresleným spanom. Ak je span prekrytý, t.j. leží celý v rozmedzí x0 a x1 nejakého spanu v danom riadku, nebude sa znova kresliť.

- Ak je span neprekrytý, vypočíta sa prienik medzi spanmi a vykreslia sa len tie časti, ktoré ešte neboli vykreslené (tým sa zaplnia "diery" v S-buffri).
- Rutina na vkladanie spanu nie je jednoduchá a jej efektívnosť závisí na výkone vizualizačného systému.

- **S-buffer vs. Z-buffer:**
  - S-buffer a Z-buffer sú alternatívne techniky na riešenie problému viditeľnosti v počítačovej grafike.
  - **S-buffer:**
    - Namiesto ukladania hĺbky pre každý pixel, S-buffer ukladá viditeľné horizontálne úseky (spany), pričom hĺbka sa ukladá iba na začiatku a na konci spanu, a medzi týmito hodnotami sa interpoluje.
    - Pri vykresľovaní sa kontroluje, či nový span prekrýva (podľa hĺbky) existujúci span, a podľa toho sa aktualizuje.
    - Potrebuje menej pamäte, pretože pracuje s úsekmi a nie s každým pixelom, čo je efektívnejšie pre scény s veľkými polygónmi.
    - Má zložitejšiu implementáciu a nie je ideálny pre veľmi fragmentované scény.
    - Rieši viditeľnosť po častiach (úsekoch), čo znamená nižší počet operácií a menšiu pamäťovú náročnosť.
  - **Z-buffer:**
    - Z-buffer ukladá hĺbku (Z hodnotu) pre každý pixel na obrazovke.
    - Pri vykresľovaní každého bodu kontroluje, či je bližšie k pozorovateľovi než aktuálna hodnota v Z-bufferi, a podľa toho aktualizuje hodnoty.
    - Má väčšiu pamäťovú náročnosť, pretože musí spracovať každý pixel.
    - Má jednoduchšiu implementáciu a funguje pre ľubovoľné scény.
    - Rieši viditeľnosť pixel po pixeli, čo znamená väčší počet operácií a vyššiu pamäťovú náročnosť.

61 Vyplňovanie oblastí používané v počítačovej grafike

- **Typy vyplňovania:**
  - **Vyplnenie oblasti jedinou farbou:** Jednoduchý spôsob, kde celá vyplňovaná oblasť dostane jednu rovnakú farbu. Tento spôsob je najbežnejší pri jednoduchých objektoch alebo základných výpočtoch.
  - **Vyšrafovanie oblasti:** Pri vyšrafovaní sa používa vzor čiar alebo šrafovanie, aby sa oblasť vyplnila s rôznymi štruktúrami, čo môže pomôcť vizuálne odlíšiť rôzne časti obrazu alebo oblasti. Tento prístup je často využívaný v technických alebo inžinierskych aplikáciách.

  - **Vyplnenie farebným vzorom (textúrovanie):** Tento spôsob využíva textúry alebo vzory, ktoré sa aplikujú na plochu vyplňovanej oblasti. Textúrovanie je bežne používané v 3D grafike na zvýraznenie detaily povrchu objektu, ako sú štruktúry, materiály alebo vzory.

- **Rozdelenie algoritmov vyplňovania podľa typu hranice:**
  - **Hranica definovaná geometricky:** Tento typ vyplňovania predpokladá, že hranica oblasti je zadaná geometricky, napríklad pomocou rovnín, kriviek alebo polygónov. Algoritmy ako "scanline filling" alebo "boundary fill" sa používajú na výplň oblastí definovaných týmito geometrickými tvarmi.
  - **Hranica nakreslená na zobrazovači:** V tomto prípade je hranica oblasti zadaná priamo na obrazovke, zvyčajne pomocou kreslenia priesečníkov alebo zložitejších techník, ktoré identifikujú, ktoré pixely ležia vo vnútri a ktoré vonku definovanej hranice.

62 Algoritmus riadkového rozkladu pri vyplňovaní oblastí

- **Postup metódy Scanline:**
  1. Postupuje sa od najvyššieho vrcholu oblasti k najnižšiemu, smerom zľava doprava v každom riadku. Tento proces je vykonávaný po jednotlivých riadkoch, pričom každý riadok je analyzovaný nezávisle.
  2. Pre každý rozkladový riadok, ktorý je rovnobežný s osou X a má konštantnú hodnotu Y, sa hľadajú priesečníky s hranicami vyplňovanej oblasti. Hodnota Y sa znižuje o 1 pri prechode na ďalší riadok.
  3. Vo výslednom zozname priesečníkov sú tieto priesečníky usporiadané zľava doprava (podobne ako v S-bufferi), a potom sa vyfarbujú úseky medzi nepárnymi a párnymi priesečníkmi. Preto je nevyhnutné, aby bol počet priesečníkov vždy párny, aby bolo možné správne definovať vyplnené úseky medzi priesečníkmi.

63 Vyplňovanie spektrom

- Vyplňovanie spektrom - 2 spôsoby:
  - **Prvý spôsob:**
    - Tento spôsob vyplňovania spektrom využíva metódu riadkového rozkladu, ktorá bola popísaná skôr.
    - Na začiatku sa musí mnohouholník otočiť tak, aby smer vykresľovania priamok bol rovnobežný s osou X, teda o uhol β.
    - Po otočení mnohouholníka sa metódou riadkového rozkladu nájdu priesečníky.
    - Pri vykresľovaní úsekov medzi priesečníkmi je potrebné úseky opäť otočiť o uhol β.
    - Rozdiel oproti bežnej riadkovej výplni spočíva v tom, že oblasť nie je vyplnená jednou farbou, ale farba sa plynule mení, prechádzajúc postupne z prvej farby do druhej.
  - **Druhý spôsob:**

    - Tento spôsob využíva nastavenie orezávacej oblasti, ktoré poskytujú grafické rozhrania, ako napríklad MS Windows.
    - Na začiatku sa nastaví orezávacia oblasť na celú oblasť mnohouholníka a vypočíta sa minimálna a maximálna hodnota súradníc (X_{min}, X_{max}, Y_{min}, Y_{max}).
    - Podľa veľkosti uhla β sa vyplní príslušný rovnobežník a orezávacia oblasť zabezpečí vyplnenie len v oblasti mnohouholníka.

- Plynulý prechod farieb:
  - Pri vyplňovaní spektrom je dôležité vykonať plynulý prechod farieb medzi dvoma farbami. Tento prechod sa robí nasledovne:
    1. Zistí sa rozdiel medzi jednotlivými zložkami farieb (napr. RGB) prvej a druhej farby.
    2. Vypočíta sa počet úsečiek v rovnobežníku, ktorý bude vyplňovaný.
    3. Vypočíta sa prírastok každej zložky farby (R, G, B) ako pomer rozdielu medzi zložkami prvej a druhej farby a počtu úsečiek. Tento prírastok určuje, o koľko sa každá zložka farby zmení medzi jednotlivými úsečkami.
    4. Príklad: Ak pre zložku R je rozdiel medzi prvu a druhou farbou 50 a počet úsečiek je 10, prírastok pre R je 5.

- Vyplňovanie spektrom (prechod v RGB modeli):
  - V rámci vyplňovania spektrom môže byť prechod realizovaný v RGB modeli, kde farby plynule prechádzajú medzi dvoma hodnotami pomocou alfa miešania.
  - **Alfa miešanie:** Alfa miešanie môže byť lineárne alebo nelineárne, a umožňuje vytvárať plynulý prechod medzi dvoma farbami v RGB priestore.
  - Týmto spôsobom je možné dosiahnuť realistický efekt prechodu farieb, čo sa často využíva v grafických aplikáciách na vykresľovanie svetelných efektov alebo gradientov.

64 Inverzné a plotové vyplňovanie

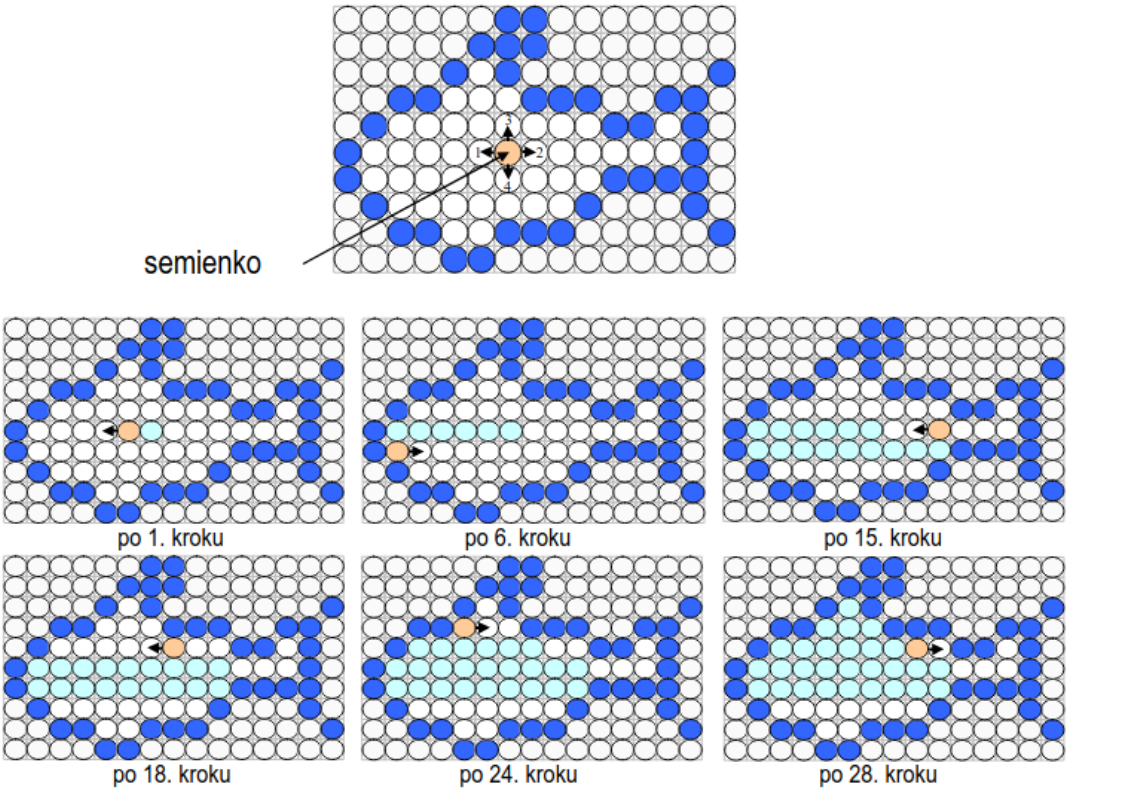
- Inverzné vyplňovanie využíva metódu XOR (exkluzívny OR), ktorá je definovaná nasledovne:
$$0 \oplus 0 = 0, \quad 1 \oplus 1 = 0, \quad 0 \oplus 1 = 1, \quad 1 \oplus 0 = 1$$
- Pri inverznom vyplňovaní sa vykonáva "inverzia" farieb alebo hodnôt v závislosti od predchádzajúceho stavu pixelov.
- Táto technika je veľmi užitočná na vykresľovanie oblastí s alternujúcimi farbami alebo v prípadoch, keď je potrebné "vypnúť" už vykreslené oblasti.
- Dva hlavné typy inverzného vyplňovania:
  - **Vodorovné (riadkové) vyplňovanie:** Vyplňovanie sa vykonáva po jednotlivých riadkoch, kde sa XOR operácia vykonáva v horizontálnom smere.
  - **Zvislé (plotové) vyplňovanie:** Vyplňovanie sa vykonáva v zvislom smere, t.j. po jednotlivých stĺpcoch, kde sa XOR operácia vykonáva vertikálne.

65 Rekurzívne a nerekurzívne semienkové vyplňovanie

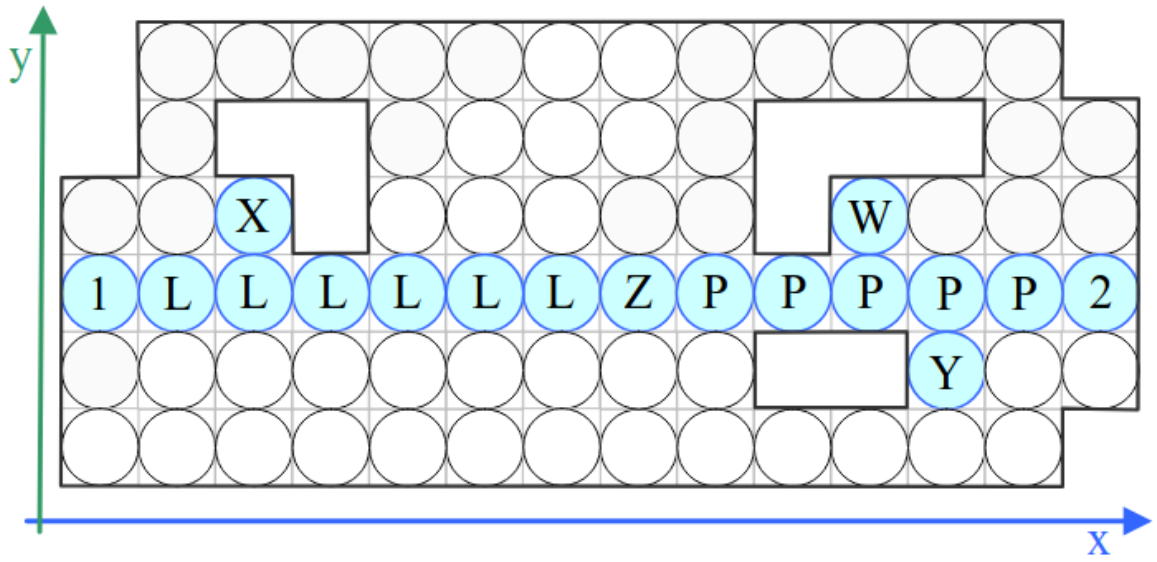
- Semienkové vyplňovanie (Seed Fill)
  - Semienkové vyplňovanie je algoritmus používaný na vyplňovanie uzavretých oblastí, kde sa začína od určitého bodu (semienka) a vyplňuje sa oblasť okolo tohto bodu, až kým sa nenarazí na hranicu alebo už vyplnenú oblasť. Algoritmus je založený na opakovaní určitej operácie (vyplnenie) pre susedné pixely
  - Algoritmus `SeedFill(x, y)` začína na pozícii (sx, sy), kde sa najprv nastaví pixel na modrú farbu (v tomto prípade označujúcu vyplnenie).
  - Potom algoritmus rekurzívne kontroluje a vyplňuje susedné pixely (ľavý, pravý, horný, dolný), ktoré nie sú farby hraničnej oblasti (`HR_FARBA`) a nie sú už vyplnené (nie sú modré).
  - Tento proces pokračuje rekurzívne, kým nebudú všetky priesečníky vyplnené.

```
SeedFill(x, y)
{
    SetPixel(sx, sy, BLUE);
    if(((GetPixel(sx-1, sy)) != HR_FARBA) && ((GetPixel(sx-1, sy))
        SeedFill(sx-1, sy);
    if(((GetPixel(sx+1, sy)) != HR_FARBA) && ((GetPixel(sx+1, sy))
        SeedFill(sx+1, sy);
    if(((GetPixel(sx, sy-1)) != HR_FARBA) && ((GetPixel(sx, sy-1))
        SeedFill(sx, sy-1);
    if(((GetPixel(sx, sy+1)) != HR_FARBA) && ((GetPixel(sx, sy+1))
        SeedFill(sx, sy+1);
}
```

- Rekurzívne semienkové vyplňovanie:
  - Rekurzívna verzia semienkového vyplňovania používa rekurzívne volania na vyplňovanie oblastí v štyroch smeroch (hore, dole, vľavo, vpravo). Tento prístup je jednoduchý, ale môže spôsobiť problémy pri veľmi veľkých oblastiach, keď sa prekročí maximálna hĺbka rekurzívneho volania.



- Nerekurzívne semienkové vyplňovanie:
  - Nerekurzívne semienkové vyplňovanie používa zásobník na simuláciu rekurzívnych volaní, čo umožňuje kontrolovať hĺbku a zabraňuje prekročeniu limitov rekurzívnych volaní. Tento prístup je efektívnejší pre veľké oblasti, kde rekurzia nie je ideálna.
  - Namiesto použitia rekurzie sa vytvorí zásobník, do ktorého sa ukladajú pixely, ktoré je potrebné spracovať.
  - Kým zásobník nie je prázdny, algoritmus vyberie pixel, skontroluje jeho susedov a ak sú platní, pridá ich na zásobník.
  - Tento prístup je efektívnejší, pretože sa vyhýba problémom s rekurzívnym pretečením zásobníka pri veľkých oblastiach.



66 Textúrovanie a jeho vzťah k zobrazovacím reťazcom

- Proces aplikácie obrazových vzoriek, ako sú textúry alebo tapety, na povrch 3D objektov za účelom dosiahnutia vizuálneho dojmu, že objekt je z určitého materiálu, napríklad dreva, kameňa, kovu a pod
- Tento proces výrazne zlepšuje vizuálnu komplexnosť a realistickosť počítačových scén
- **Typy textúr podľa topologického rozmeru:**
  - **1D textúry:** Textúry, ktoré sú definované len pozdĺž jednej osi, často používané pre textúrovanie povrchov s pravidelným opakovaním (napríklad textúra pások alebo čiar).
  - **2D textúry:** Najbežnejší typ textúr, ktoré sú definované na dvojrozmernej ploche. Využívajú sa na textúrovanie plochých povrchov alebo rovinných plôch objektov.
  - **3D textúry:** Textúry, ktoré sú aplikované na trojrozmerné objekty a obsahujú hĺbku, teda môžu byť použité na textúrovanie 3D objektov s komplexnými povrchmi, ako sú okolo zakrivených objektov.
- **Typy textúr podľa spôsobu nanášania:**
  - **Statické textúry:** Textúry, ktoré sa nemenia počas vykresľovania, teda sú fixné a nemenné.
  - **Dynamické textúry:** Textúry, ktoré sa môžu meniť počas vykresľovania, a to buď:
    - **Procedurálne textúry:** Textúry generované matematicky na základe algoritmov. Môžu byť dynamicky vytvárané v reálnom čase a nevyžadujú veľké množstvo pamäte.
    - **Animačné textúry:** Textúry, ktoré sa animujú, napríklad pre simuláciu pohybu alebo zmeny (ako vodné efekty, pohyb dymu alebo plameňov).

67 Bilineárne textúrovanie, bump-map textúrovací proces a technológia „pečenia“ textúr

- **Textúrovanie (Bilineárne, UV):**
  - Pri procese nanášania obrazového formátu na plochy trojrozmerného objektu sa najčastejšie používa textúra vo forme obrázka.
  - Textúra je aplikovaná na 3D objekt pomocou dvojrozmernej súradnicovej sústavy SST [u, v], kde:
    - **Súradnica „u“:** Zodpovedá osi x obrázka (textúry).
    - **Súradnica „v“:** Zodpovedá osi y obrázka (textúry).
  - **Jedna jednotka:** Jedna jednotka (u/v) je definovaná ako dĺžka alebo šírka obrázku textúry, teda ide o normalizované hodnoty medzi 0 a 1. Tento proces mapovania zabezpečuje, že každý bod na povrchu objektu bude zodpovedať konkrétnemu bodu na textúre.
- **Bump-map textúrovací proces:**
  - Bump mapping je technika, ktorá simuluje detaily povrchu (ako sú vrásky, drsnosti, rysy), ktoré sa zvyčajne vyskytujú na textúrovaných objektoch bez potreby skutočného modelovania geometrie týchto detailov.
  - Používa sa šedotónová mapa (tzv. bump map), ktorá obsahuje informácie o výškach bodov na povrchu objektu. Tieto hodnoty sa používajú na simuláciu odrazov svetla, čím sa vytvára efekt detaily bez zvyšovania geometrie modelu.
  - Hlavnou výhodou bump mappingu je, že výrazne zlepšuje vzhľad povrchu, keď je spracovávaný svetelný efekt, pričom sa nezvyšuje počet polygónov.
  - Bump-mapping je často kombinovaný s inými technikami, ako je normal mapping, na ešte presnejšie modelovanie svetelných interakcií na povrchoch.
- **„Pečenie“ textúry (Texture Baking):**
  - „Pečenie“ textúry je proces, ktorý sa používa na prenos údajov o textúre (materiály) zo zložitejšieho 3D modelu na jednoduchší 3D model, ktorý má menej detailov a nižší počet polygónov.
  - Tento proces umožňuje preniesť detaily, ako sú textúry, farby, osvetlenie, tiene, odrazy, a iné vizuálne efekty z high-poly modelu na low-poly model.
  - **Hlavné dôvody použitia „pečenia“ textúr:**
    - Zníženie veľkosti súboru pri použití low-poly modelu, ktorý je menší a efektívnejší na spracovanie.
    - Umožňuje použitie zjednodušených modelov s vysokými vizuálnymi detailmi v reálnom čase, čo je výhodné pre aplikácie ako video hry alebo virtuálna realita.
    - Zjednodušený model umožňuje efektívnejšie spracovanie v reálnom čase, pretože má menej polygónov a nevyžaduje náročné výpočty.

– „Pečenie“ textúry sa často používa v súvislosti s LOD (Level of Detail) technikami, kde sa detailné textúry a efekty prenášajú na menej detailné modely, čím sa znižuje počet potrebných výpočtov pri zobrazovaní scény.

68 LOD technológie v rámci počítačovej grafiky

- **Level of Detail (LOD):**
  - LOD je technika, ktorá optimalizuje výkon počítačovej grafiky znižovaním komplexnosti objektov a ich parametrov v závislosti od vzdialenosti od kamery alebo uhla pohľadu.
  - LOD sa implementuje:
    * **Diskrétne** – v skokových intervaloch, napríklad zamenením objektu alebo materiálu v závislosti od vzdialenosti alebo uhla.
    * **Kontinuálne** – dynamicky (priebežne) prepočítavané podľa vzdialenosti alebo uhla, vrátane prepočítavania zložitosti objektov.
- **Základné techniky LOD v počítačovej grafike:**
  - **Geometrický LOD (Geometric LOD, Tessellation LOD):** Mení geometrickú zložitosť objektu v závislosti od vzdialenosti alebo uhla pohľadu. Bližšie objekty majú viac detailov, vzdialené menej.
  - **Substitučný LOD (Substitution LOD, Impostor LOD):** Vzdialené objekty sa nahrádzajú 2D billboardmi, ktoré zobrazujú objekt z rôznych uhlov. Používa sa na zníženie počtu trojuholníkov pri zachovaní vizuálnej ilúzie.
  - **Obrazovkový LOD (Screen-space LOD, Resolution LOD):** Určuje úroveň detailu objektu podľa toho, koľko pixelov objekt zaberá v zornom poli kamery.
  - **Hierarchický LOD (Hierarchical LOD, Group LOD):** Objekty v diaľke sa zhlukujú do väčších skupín a zobrazujú ako jeden jednoduchý objekt (napr. skupina stromov alebo vtákov).
  - **Terénny LOD (Terrain LOD, Geomipmapping):** Optimalizácia terénnej grafiky delením terénu na dlaždice, z ktorých každá má svoju úroveň LOD (HLOD). Vzdialené dlaždice sú zobrazené s menším detailom.
  - **Materiálový/textúrovací LOD (Material LOD):** Kvalita textúry alebo materiálu sa volí podľa vzdialenosti. Používa sa napríklad texture baking alebo mipmapping na rôzne veľkosti textúr.
  - **Osvetľovací LOD (Lighting LOD):** Zložitosť osvetlenia (napr. Lambertov alebo Phongov model) sa upravuje podľa vzdialenosti. Môžu sa použiť osvetľovacie mapy (lightmaps) alebo obmedziť vrhanie tieňov.
  - **Stereoskopický LOD (Stereo alebo Dual-eye rendering LOD):** Určuje použitie jednej alebo dvoch kamier, ak je vzdialenosť väčšia ako stereoskopické rozlíšenie oka (cca 220m).
  - **Tieňovací LOD (Shading LOD):** Úroveň tieňovania (a tým počet počítaných normál) sa mení podľa vzdialenosti objektu.

  - **LOD zorného poľa (Foveated Eye-tracked LOD):** Vykresľovanie je optimalizované tak, že sa zvyšuje detail v strede obrazu (fovea) a znižuje sa detail smerom k periférnemu zornému poľu oka.

- **Rozšírené a urýchľovacie techniky LOD:**
  - **Zjednodušovanie siete (Mesh simplification, Mesh decimation):** Znižovanie počtu vrcholov a polygónov pomocou redukcie alebo zoskupovania veľmi blízkych vrcholov. Používa sa minimalizácia kvadratickej chyby, kde sa vyberá nová pozícia vrcholu tak, aby sa minimalizoval súčet chýb.
  - **Multiresolution meshes:** Použitie rôznych úrovní detailov v rámci jednej scény, ktoré sa prispôsobujú pohľadu v reálnom čase a blízkosti ku kamere.
  - **Occlusion culling:** Spojenie LOD techník s detekciou viditeľnosti, kde sa nevykresľujú objekty, ktoré sú zakryté inými objektmi z pohľadu kamery. Pomáha to znižovať počet objektov na spracovanie a zvyšovať výkon.
  - **Neurónové siete pre generovanie LOD (Neural Geometric LOD):** Použitie neurónových sietí na generovanie alebo optimalizáciu geometrického detailu objektov v reálnom čase. S pomocou Signed Distance Function (SDF) a adaptívnych oktantových stromov sa určuje optimálna úroveň LOD pre každý objekt.

69 Konštantné (flat) tieňovanie v rámci počítačovej grafiky

- **Konštantné tieňovanie (Flat Shading):**
  - Tento spôsob tieňovania predpokladá rovnakú orientáciu všetkých stien v scéne, čo znamená, že pre každú plochu objektu sa použije rovnaký tieň.
  - Všetky vrcholy na danej ploche majú rovnaký normálový vektor, a preto sa na celú plochu aplikuje rovnaký tieň.

- **Podmienka tieňovania:**
  - Tieňová intenzita je závislá od uhla φ medzi vektorom dopadu lúča l zo svetelného zdroja a normálou steny n.
  - Uhol φ sa používa na výpočet intenzity svetla na konkrétnom mieste, pričom čím je tento uhol menší (teda svetelný lúč je viac kolmo na plochu), tým je intenzita svetla vyššia a tým je tieň svetlejší.

- **Výhody konštantného tieňovania:**
  - Vysoký výkon, pretože výpočty sú jednoducho implementované na základnom geometrickom priestore (stačí počítať normály a svetelné lúče).
  - Rýchle na výpočty v reálnom čase, čo je dôležité pre aplikácie, ktoré vyžadujú vysokú snímkovú frekvenciu, ako napríklad videohry.

- **Nevýhody konštantného tieňovania:**
  - Vytvára tvrdé, zjednodušené tiene a nedokáže zachytiť jemné prechody tieňov, ktoré sa vyskytujú v skutočnom svete.

  - Neposkytuje realistický vzhľad, pretože pri tomto type tieňovania nie sú zohľadnené detaily ako zmäkčenie tieňov alebo odraz svetla medzi plochami.

70 Tieňovanie interpoláciou farby (Gouraud) v rámci počítačovej grafiky

- **Gouraudovo tieňovanie:**
  - Ide o metódu tieňovania, ktorá interpoluje farby medzi vrcholmi (vertices) trojuholníkového objektu. Na každom vrchole je vypočítaná farba na základe svetelného modelu, a následne sa tieto farby interpolujú pozdĺž hraníc trojuholníka.
  - Táto metóda je rýchla, pretože nevyžaduje výpočty pre každý pixel, ale iba pre vrcholy objektu.
- **Popis procesu:**
  - Na začiatku sa vypočíta intenzita svetla na vrcholoch trojuholníka. Tento výpočet zahŕňa faktory ako normály, svetelné lúče a ďalšie parametre.
  - Následne sa tieto farby interpolujú medzi vrcholmi, čím sa vytvorí plynulý prechod farieb na povrchu trojuholníka.
- **Výhody Gouraudovho tieňovania:**
  - Rýchle a efektívne, pretože sa nevyžaduje výpočet svetla pre každý pixel.
  - Ideálne pre aplikácie, kde je potrebný rýchly výpočet farieb s prijateľnou úrovňou detailu.
- **Nevýhody Gouraudovho tieňovania:**
  - Môže viesť k chybám v zobrazení tienistých oblastí, pretože interpolácia medzi vrcholmi nezohľadňuje detaily medzi vrcholmi.
  - V prípade veľmi zakrivených povrchov alebo malých plôch môže dôjsť k vizuálnym artefaktom (napríklad „svetelným pásom“).

71 Phongovo tieňovanie (Interpolácia normály) v rámci počítačovej grafiky

- **Phongovo tieňovanie (Interpolácia normály):**
  - Phongovo tieňovanie je pokročilá metóda tieňovania, ktorá interpoluje normály medzi vrcholmi objektu (na rozdiel od Gouraudovho tieňovania, ktoré interpoluje priamo farby).
  - Tento prístup poskytuje realistickejšie výsledky, pretože sa počíta svetelná intenzita na základe normál medzi jednotlivými pixelmi, čím sa dosahuje hladší prechod svetla.
- **Určenie normál:**
  - Normály sú vektory, ktoré sú kolmé na povrch objektu a hrajú kľúčovú úlohu pri výpočte svetelných odrazov.
  - Pri Phongovom tieňovaní sa normály určujú na základných vrcholoch trojuholníka, ktoré sa následne interpolujú medzi pixelmi na povrchu objektu.
- **Určenie normál po interpolácii (príklad):**

  - Po interpolácii normál medzi vrcholmi trojuholníka sa vypočíta intenzita svetla v každom pixeli na základe uhla medzi normálou a svetelným lúčom.
  - Tento proces vedie k realistickému tieňovaniu povrchu a lepšiemu vykresleniu svetelných efektov ako je lesk a odrazy.

72 Osvetľovanie a osvetľovacie modely, svetelné zdroje

- **Osvetľovanie:**
  - Osvetľovanie je proces, ktorý opisuje vplyv svetelného zdroja, materiálu a ostatných objektov na svoje okolie. Zohľadňuje vrhanie tieňov, odrazy svetla a rôzne interakcie svetla s povrchmi.
  - Osvetľovanie sa často delí na dva typy:
    * **Statické osvetlenie:** Osvetľovanie, ktoré sa nemeni počas vykresľovania scény, teda svetelný zdroj a jeho vplyv sú predpočítané a fixné.
    * **Dynamické osvetlenie:** Osvetľovanie, ktoré sa mení počas vykresľovania, napríklad kvôli pohybu objektov alebo svetelných zdrojov.
- **Osvetľovacie modely:**
  - Osvetľovací model je matematický model, ktorý opisuje vlastnosti povrchov, ako sú farba, lesklosť, matnosť a drsnosť, a ich interakciu so svetlom.
  - Základom osvetľovacieho modelu je **odrazová funkcia (reflection function)**. Táto funkcia určuje, ako svetlo odráža od povrchu v závislosti od smeru dopadajúceho svetla a smeru pozorovania. Čím realistickejšie je odrazová funkcia, tým presvedčivejší je vizuálny dojem generovaného objektu v počítačovej grafike.
- **Typy svetelných zdrojov podľa farby:**
  - **Monochromatické svetlo:** Svetlo jednej farby alebo vlnovej dĺžky (napríklad svetlo červenej farby).
  - **Achromatické svetlo:** Svetlo, ktoré obsahuje všetky vlnové dĺžky a neprodukuje farebný tón, teda je neutrálne (napríklad biele svetlo).
- **Typy svetelných zdrojov podľa kinematiky:**
  - **Statické svetlo:** Svetelný zdroj, ktorý je fixný a jeho umiestnenie alebo intenzita sa nemenia počas vykresľovania.
  - **Dynamické svetlo:** Svetelný zdroj, ktorý môže meniť svoju polohu alebo intenzitu počas vykresľovania.
- **Základné typy svetelných zdrojov:**
  - **Okolité (ambientné) svetlo:** Svetlo, ktoré osvetľuje scénu rovnomerne, bez ohľadu na smer. Nevyžaruje od žiadneho konkrétneho zdroja a slúži na osvetlenie celého prostredia.
  - **Bodové svetlo (Point light):** Svetelný zdroj, ktorý vyžaruje svetlo rovnomerne do všetkých smerov z určitého bodu v priestore.

  - **Reflektorové svetlo (Spot light):** Svetelný zdroj, ktorý vyžaruje svetlo v kužeľovom tvare, so špecifickým smerom a intenzitou.
  - **Smerové svetlo (Directional light):** Svetelný zdroj, ktorý vyžaruje svetlo v jednom smere, pričom všetky lúče sú paralelné. Tento typ svetla sa používa na simuláciu slnečného svetla.

73 Lambertov osvetľovací model

- **Lambertov osvetľovací model** je model používaný na simuláciu difúzneho osvetlenia, ktorý predpokladá, že svetlo sa rozptyľuje rovnomerne po povrchu objektu bez ohľadu na uhol pohľadu.
- Tento model je založený na predpoklade, že intenzita svetla závisí od uhla medzi normálou povrchu a smerom dopadajúceho svetla.
- **Zložky Lambertovho modelu:**
  - **Difúzna zložka:** Táto zložka reprezentuje osvetlenie povrchu, ktoré je rovnomerne rozptýlené vo všetkých smeroch. Je závislá na uhlu medzi normálou povrchu a smerom svetelného zdroja.
  - **Ambientná zložka:** Táto zložka predstavuje základné osvetlenie scény, ktoré nezávisí od svetelných zdrojov. Je to osvetlenie, ktoré je rovnomerne rozptýlené po celej scéne a je nezávislé na pozícii svetelných zdrojov.
- **Výsledok:**
  - Kombinácia difúznej a ambientnej zložky vedie k výslednému osvetleniu povrchu, ktoré je realisticky osvetlené bez ohľadu na orientáciu povrchu, ale s vplyvom svetelného zdroja a okolitého osvetlenia.

74 Phongov osvetľovací model

- **Phongov osvetľovací model** je rozšírený osvetľovací model, ktorý popisuje tri hlavné zložky svetla: ambientnú, difúznu a zrkadlovú.
- **Zložky Phongovho modelu:**
  - **Ambientná zložka (ambient light):**
    - Predstavuje odraz nešpecifikovaného, okolitého svetla, ktoré prichádza zo všetkých smerov.
    - Okolité svetlo vzniká viacnásobnými odrazmi od iných objektov a rozptylom svetla spôsobeným molekulami vzduchu.
    - Táto zložka je väčšinou biela (achromatická) a nezávisí na smere pohľadu.
  - **Difúzna zložka (diffuse):**
    - Táto zložka sa riadi pravidlom, že odrazené svetlo je rovnomerne rozptýlené vo všetkých smeroch.
    - Intenzita tejto zložky závisí len na uhle dopadu svetelného lúča (úmerná kosínusu uhla medzi normálou povrchu a smerom svetelného lúča).
    - Táto zložka nám poskytuje informácie o farbe povrchu objektu.

  - **Zrkadlová zložka (specular):**
    - Zrkadlová zložka závisí na smere pohľadu a zrkadlových vlastnostiach povrchu.
    - Svetelný lúč, ktorý je odrazený od povrchu, sa riadi zákonmi štatistického rozdelenia, čo môže viesť k utlmeniu odrazu v závislosti na vlastnostiach povrchu.
    - Veľkosť tejto zložky závisí na uhle dopadu a optických vlastnostiach povrchu, ako sú lesk a hladkosť.
- Tento model umožňuje realistické simulovanie svetla a odrazu na povrchu objektov, pričom zohľadňuje nielen intenzitu svetla, ale aj jeho interakciu s materiálom objektu.

75 Osvetľovacie mapy a zrkadlá

- **Osvetľovacie mapy (Lightmaps)** predstavujú predpočítaný celkový vplyv svetelných zdrojov na scénu.
- Slúžia na ukladanie informácií o osvetlení priamo do textúr, čím sa znižuje výpočtová náročnosť počas vykresľovania.
- **Proces osvetľovania scény:**
  1. Výpočet osvetľovacej mapy (globálne osvetlenie, tiene).
  2. Dodanie textúr objektom.
  3. Dodanie dynamických svetelných zdrojov.
- **Implementácia zrkadiel** rieši realistické zobrazenie odrazov okolitých objektov a svetelných zdrojov na povrchoch.
- **Typy zrkadiel:**
  - **Environmentálne zrkadlá:**
    - Využívajú mapovanie okolia (napr. cubemap).
    - Odraz je aproximovaný na základe okolitej scény.
    - Sú výpočtovo menej náročné, ale menej presné.
  - **Geometrické zrkadlá:**
    - Využívajú presný geometrický výpočet odrazu.
    - Scéna sa renderuje z pohľadu zrkadla.
    - Poskytujú realistické odrazy, ale sú výpočtovo náročnejšie.

76 Realistické zobrazovanie a globálne osvetľovacie techniky v rámci počítačovej grafiky, metódy vychádzajúce od pozorovateľa a od svetelného zdroja

- **Globálne osvetľovacie techniky** slúžia na riešenie zobrazovacej rovnice a simulujú realistické šírenie svetla v scéne vrátane viacnásobných odrazov, lomov a tieňov.
- Ich cieľom je:
  - výpočet osvetlenia všetkých plôch v scéne (pohľadovo nezávislé),

  - výpočet osvetlenia pre konkrétny smer k pozorovateľovi (pohľadovo závislé).
- **Rozdelenie globálnych osvetľovacích techník:**
  - metódy vychádzajúce od pozorovateľa,
  - metódy vychádzajúce od svetelného zdroja,
  - obojsmerné metódy,
  - vyžarovacia metóda (Radiosity).
- **Metódy vychádzajúce od pozorovateľa (gathering methods):**
  - Sú pohľadovo závislé a sledujú dráhu lúča od oka do scény.
  - Zhromažďujú svetelnú energiu počas spätnej trajektórie lúča.
  - **Ray tracing:**
    * Podporuje zrkadlové odrazy a ostré tiene.
    * Nezohľadňuje úplné globálne osvetlenie (napr. kaustiky, difúzne odrazy).
  - **Distributed ray tracing:**
    * Rozširuje ray tracing o náhodné difúzne odrazy.
    * Umožňuje realistickejšie simulácie osvetlenia.
- **Metódy vychádzajúce od svetelného zdroja (shooting methods):**
  - Sledujú lúče vyžarované zo svetelných zdrojov do scény.
  - Lepšie riešia javy ako kaustiky a skryté svetelné zdroje, ale trpia šumom.
  - **Photon tracing:**
    * Duálna metóda k ray tracingu.
    * Sleduje dráhu fotónov zo svetla až k difúznemu odrazu.
  - **Light tracing:**
    * Duálna metóda k path tracingu.
    * Príspevok svetla sa započíta, ak je bod odrazu viditeľný pozorovateľom.

77 Algoritmy fotorealistických metód a metóda raytracing

- **Fotorealistické metódy** slúžia na realistické zobrazovanie scén na základe fyzikálne korektnej simulácie šírenia svetla.
- Podľa spôsobu spracovania dát sa delia na:
  - algoritmy zobrazujúce povrchy,
  - objemové algoritmy,
  - algoritmy pracujúce v obrazovom priestore,
  - algoritmy pracujúce v objektovom priestore,
  - hybridné algoritmy.

- **Algoritmy zobrazujúce povrchy (surface-based techniques):**
  - Z objemových dát vytvárajú pomocnú geometrickú reprezentáciu povrchu.
  - Povrch je aproximovaný pomocou geometrických primitív.
  - Informácie o vnútri objektu sa strácajú.
  - Príklady:
    * contour tracking,
    * marching cubes,
    * marching tetrahedra,
    * dividing cubes,
    * opaque cubes.
- **Objemové algoritmy (volume rendering):**
  - Využívajú kompletnú trojrozmernú mriežku voxelov.
  - Sú nezávislé od geometrickej zložitosti scény.
  - Sú pamäťovo a výpočtovo náročné, ale zachovávajú vnútornú štruktúru objektu.
  - Podľa klasifikácie údajov:
    * **Binárne** – voxel je buď úplne obsadený, alebo prázdny.
    * **Pravdepodobnostné** – voxel má priradený čiastočný podiel objektu.
- **Algoritmy pracujúce v obrazovom priestore:**
  - Pre každý pixel výsledného obrazu sa hľadajú prispievajúce voxely.
  - Hodnoty voxelov sa určujú interpoláciou.
  - Príklady:
    * ray tracing / ray casting,
    * Sábelova metóda.
- **Algoritmy pracujúce v objektovom priestore:**
  - Pre každý voxel sa určujú pixely, ktoré ovplyvňuje.
  - Príspevky voxelov sa akumulujú do obrazového priestoru.
  - Príklady:
    * V-buffer,
    * splatting.
- **Hybridné algoritmy:**
  - Kombinujú prístup obrazového a objektového priestoru.
  - Typickým príkladom je **Shear-Warp algoritmus**.

  - Je veľmi rýchly, ale produkuje artefakty a nižšiu kvalitu obrazu.
- **Metóda Ray tracing:**
  - Je pohľadovo závislá fotorealistická metóda.
  - Pre každý pixel sa vysiela lúč z pozorovateľa do scény.
  - Po náraze lúča na objekt sa počítajú:
    * odrazené lúče,
    * lomené lúče,
    * tieňové lúče.
  - Podporuje zrkadlové odrazy, lom svetla a ostré tiene.
  - Nezabezpečuje úplné riešenie globálneho osvetlenia (napr. difúzne odrazy, kaustiky).

78 Fraktály a časticové systémy

- **Fraktály** sú geometrické objekty, ktoré sa vyznačujú sebepodobnosťou a detailnou štruktúrou v rôznych mierkach.
- Podľa Mandelbrota je fraktál množina, ktorej Hausdorffova dimenzia je väčšia než jej topologická dimenzia.
- Názov fraktál pochádza z latinského slova *frangere* (lámať).
- **Sebepodobnosť:**
  - Základná vlastnosť fraktálov.
  - Časti objektu sú podobné celku, líšia sa iba mierkou.
  - Príkladom sú prírodné objekty, napr. karfiol alebo papraď.
- **Typy fraktálov:**
  - **L-systémy (Lindenmayerove systémy):**
    * Jednoduché fraktály definované pravidlami prepisu.
    * Vhodné na modelovanie rastlín a prírodných tvarov.
    * Definované stavom a tabuľkou akcií.
  - **IFS – Iterated Function Systems:**
    * Fraktály generované iterovanou aplikáciou afinných transformácií.
    * Využívajú sa napr. pri fraktálnej kompresii obrazu.
  - **Dynamické systémy (dynamické množiny):**
    * Vznikajú iteráciou komplexných funkcií.
    * Tretí rozmer je znázornený farebne.
    * Príkladom sú Mandelbrotova a Juliova množina.

- **L-systémy – grafická interpretácia:**
  - Používajú tzv. korytnačiu grafiku.
  - Pohyb je riadený jednoduchými príkazmi:
    - F – pohyb vpred s kreslením,
    - f – pohyb vpred bez kreslenia,
    - + – otočenie vľavo o uhol δ,
    - – – otočenie vpravo o uhol δ.
- **Časticové systémy (Particle systems):**
  - Slúžia na simuláciu prírodných javov pomocou veľkého množstva častíc.
  - Každá častica má vlastný životný cyklus – vznik, vývoj a zánik.
  - Správanie častíc je definované programátorom.
  - Používajú sa na simuláciu dymu, ohňa, dažďa, snehu alebo explózií.
  - V niektorých prípadoch sú vhodnejšie ako fraktály.

79 Virtuálna realita – základné pojmy, kategorizácia a podsystémy

- **Virtuálna realita (VR)** je interaktívny počítačový systém, ktorý vytvára ilúziu neexistujúceho alebo syntetického priestoru a umožňuje tesné spojenie človek–výpočtový systém.
- Cieľom VR je vytvoriť čo najdokonalejšiu simuláciu prostredia a pocit prítomnosti používateľa v tomto prostredí.
- **Základné atribúty virtuálnej reality:**
  - **Imerzia** – pocit pohltenia používateľa virtuálnym prostredím.
  - **Percepcia** – vnímanie priestoru a prítomnosti v prostredí.
  - **Interakcia** – možnosť komunikovať a ovplyvňovať virtuálne prostredie.
- **Používané pojmy v oblasti VR:**
  - **Virtual Reality (VR)** – plne syntetické, počítačom generované prostredie.
  - **Mixed Reality (MR)** – prepojenie reálneho a virtuálneho sveta s ich vzájomnou interakciou.
  - **Augmented Reality (AR)** – rozšírenie reálneho sveta o virtuálne objekty.
  - **Augmented Virtuality (AV)** – prevažne virtuálne prostredie obohatené o reálne prvky.
  - **Enriched Reality (ER)** – obohatenie reality o dodatočné digitálne informácie.
  - **Virtualised Reality (VdR)** – virtualizovaná podoba reálneho sveta založená na dátach z reality.
  - **eXtended Reality (XR)** – zastrešujúci pojem pre VR, AR a MR.

  - **Hyper Reality** – technologická kombinácia VR, AR a MR vrátane fyzických prvkov.
  - **Metaverzum (Metaverse)** – zdieľaný, trvalo existujúci virtuálny priestor.
  - **Cyberspace** – virtuálny priestor vznikajúci prepojením počítačových sietí.
  - **Avatar** – digitálna reprezentácia používateľa vo virtuálnom prostredí.
  - **HCI (Human Computer Interaction)** – oblasť skúmajúca interakciu človeka s počítačom.
- **Kategorizácia VR systémov podľa úrovne prvkov:**
  - Entry VR,
  - Basic VR,
  - Medium VR,
  - Immersive VR.
- **Kategorizácia VR systémov podľa dynamiky:**
  - **SESO** – Static Environment Static Observer
  - **DESO** – Dynamic Environment Static Observer
  - **SEDO** – Static Environment Dynamic Observer
  - **DEDO** – Dynamic Environment Dynamic Observer
- **Podsystémy VR systému:**
  - **Vizualizačný podsystém** – generovanie obrazu a grafiky.
  - **Akustický podsystém** – priestorový zvuk.
  - **Statokinetický a kinematický podsystém** – sledovanie pohybu a polohy používateľa.
  - **Hmatový podsystém** – spätná haptická odozva.
  - **Čuchový podsystém** – simulácia pachov.
  - Ostatné podsystémy podľa aplikácie.

80 XR (eXtended Reality), kolaboratívna VR a systémy interakcie

- **XR (eXtended Reality)** - spája technológie VR+AR+MR
- **Zložky XR systému:**
  - **Virtuálna realita (VR)** – plne syntetické, počítačom generované prostredie
  - **Rozšírená realita (AR)** – doplnenie reálneho sveta o virtuálne objekty
  - **Zmiešaná realita (MR)** – prepojenie reálneho a virtuálneho sveta so vzájomnou interakciou
- **Kolaboratívna VR / XR (CVR, CXR):**

  - Umožňuje viacerým používateľom zdieľať spoločné virtuálne alebo zmiešané prostredie v reálnom čase

- **Systémy interakcie v XR:**
  - Jednopoužívateľské a viacpoužívateľské systémy
  - Zber vstupov pomocou gest, pohybu tela, ovládačov alebo hlasu