počítačová Grafika – ALgoritmy a PRincípy

Alexej Kolcun

Autor: Alexej Kolcun

počítačová Grafika - ALgoritmy a PRincípy

Obsah

Kapitola 1

Úvod

Položme si nasledujúce dve otázky.

1. Čo je to počítačová grafika?
2. Prečo vlastne počítačová grafika existuje?

V [23] sa počítačovou grafikou rozumie *oblasť výpočtovej techniky, ktorá sa zaoberá znázorňovaním dat v grafickej podobe.* [7] vymedzuje počítačovú grafiku ako *metódy a techniky konštrukcie, manipulácie, ukladania a zobrazenia obrazov pomocou počítača.* A mohli by sme určite nájsť ďalšie definície toho, čo to je počítačová grafika. Naše chápanie tohto pojmu sa pokúsime vymedziť nasledujúcimi riadkami.

Vonkajší svet vnímame na základe piatich zmyslov: *zrak, sluch, hmat, čuch, chuť.* Pritom prvé dva informačné kanály - obrazový a zvukový majú pre nás dominantné postavenie.

Keď sa zamyslíme nad tým, aké technológie používal človek v minulosti na uchovanie informácie, zistíme že s „konzervovaním“ vizuálnej informácie si poradil podstatne skôr ako so zvukom. Napr. verné uchovanie obrazového vnemu zvládal už od pradávna. Odhaduje sa, že napr. nástenné maľby v jaskyniach Lascaux sú zhruba 15-17 tisíc rokov staré. Samotné písmo, skonštruovanie ktorého malo zásadný význam v kultúrnom vývoji človeka, je vlastne transformácia informácie zvukovej do vizuálnej podoby. Technológia, ktorá zaznamenáva obraz *nezávisle na ľudskom faktore* – fotografia – vzniká v prvej polovici 19. storočia (J. N. Niépce – 1826, L. Daguerre – 1837). A v krátkom čase na to je zvládnutá i farebná fotografia (J. C. Maxwell – 1861). Pritom podstatný kus práce pre formalizáciu farebných vnemov spravil H. Grassmann už pred rokom 1853. V tejto súvislosti je dôležité si uvedomiť, že verné zachovanie zvuku – vynález fonografu – sa pripisuje T. A. Edisonovi až v r. 1877.

Je dôležité si uvedomiť, že zvukový vnem tesne súvisí s časom a s jeho nevratným jednosmerným plynutím. Formálne mu odpovedá sekvenčný informačný tok. Vizuálny vnem nie je s plynúcim časom tak pevne spojený. Pre spracovanie statickej vizuálnej informácie sa

dá v tomto kontexte použiť mechanizmus práce s externou pamätou s priamym prístupom. Pri určitej miere zjednodušenia tak môžeme povedať, že

- zrakový kanál využívame hlavne pre príjem a spracovanie informácií z obrovskej vonkajšej pamäti s priamym prístupom,
- zvukový kanál je náš dominantný prostriedok pre spracovanie sekvenčného informačného toku.

Okrem reálneho sveta do nášho života zasahuje čím ďalej tým viac svet *virtuálny*: Svet, ktorý je postavený na realizácii algoritmov, svet ktorý sa začína a končí zapnutím a vypnutím počítača. Prvotná komunikácia s ním bola obojstranne sekvenčne symbolická: vstup do virtuálneho sveta počítačov bol daný postupnosťou symbolov, výstup z neho tvorili taktiež postupnosti symbolov. No už prvé, z dnešného pohľadu veľmi nevýkonné, počítače boli schopné generovať veľmi veľké množstvo symbolov. Preto vznikla potreba predstaviť *výstupnú informáciu* v podobe, ktorá je človeku bližšia, tj. v podobe grafickej. A táto potreba bola podstatne dôležitejšia ako potreba prirodzeného *interaktívneho vstupu*. Naviac, prirodzenú interaktívnu komunikáciu (hlasovú) môžeme v mnohých situáciách celkom uspokojivo nahradiť podstatne jednoduchšou formou – písaným textom, tj. opäť s použitím vizuálneho informačného kanálu. Preto, aj keď dnes už existujú veľmi dômyselné a vysoko sofistikované spôsoby komunikácie s počítačom, možno povedať, že dodnes je dostatočným a uspokojivým základným vstupným zariadením pre interaktívnu komunikáciu s virtuálnym svetom počítačov klávesnica. A oknom, ktorým sa do tohoto sveta pozeráme, tj. výstupným zariadením z virtuálneho sveta, je pre nás stále grafická obrazovka.[1] (Viac o komunikácii s virtuálnou realitou [1][2]).

Počítačovou grafikou preto chápeme HW a SW prostriedky pre komunikáciu s virtuálnym svetom na základe vizuálneho informačného kanálu, tj. zraku.

Dôležitým momentom pre ďalší vývoj počítačovej grafiky je i zaujímavosť a netriviálnosť riešených problémov.

- Zvukový kanál tak, ako ho používame pre bežné potreby komunikácie, je prirodzene jednorozmerný: symbolu (slovu) môžeme priradiť index odpovedajúci poradiu, v akom je generovaný. Naproti tomu vizuálny kanál pracuje v dvoj resp. trojrozmernom priestore.
- Veľkou výzvou pre vývoj je i to, že v skutočnom svete sú naše vizuálne vnemy výsledkom fyzikálnych zákonitostí, ktorými sa svet riadi. V prípade virtuálneho sveta musíme tieto zákonitosti sami vytvárať tak, aby boli v súlade s našou skúsenosťou.

[1] Ináč povedané, počítač s klávesnicou a grafickou obrazovkou je pre nás viac akceptovateľný ako počítač s myšou, prípadne joystickom ale bez grafickej obrazovky.

[2] Aj keď je táto kniha v kontexte vývoja nových technológií „veľmi stará“, myslím, že osloví i dnešného čitateľa. Naviac, dáva čitateľovi možnosť priamo vidieť rýchlosť pokroku v tejto oblasti.

- Formalizmus postupov počítačovej grafiky je inšpiratívny i pre vývoj technológií pre ďalšie komunikačné kanály pre virtuálnu realitu. Napr. hmatový vnem môžeme pri určitom zjednodušení chápať ako výsledok riešenia priestorovej kolízie geometrických útvarov.

Určitú klasifikáciu úloh, ktoré súvisia s počítačovou grafikou, dávajú nasledujúce vzťahy [?].

vstup\výstup	symbol	obraz
symbol	1. spracovanie symbolov	2. generovanie obrazov
obraz	3. rozpoznávanie obrazov	4. spracovanie obrazov

V širšom slova zmysle do počítačovej grafiky patria oblasti 2–4. V užšom kontexte budeme chápať pod počítačovou grafikou oblasť 2., tj. generovanie obrazovej informácie. Tento text je venovaný hlavne oblastiam 2. a 4. V konkrétnych prípadoch je ale niekedy ťažké určiť hranice medzi uvedenými disciplínami.

Obmedzíme sa na formuláciu základných algoritmických postupov, ktoré sa v počítačovej grafike používajú. Snahou je popísať ich zrozumitelne (niekedy možno na úkor exaktnosti), a hlavne, *dať ich do vzájomného kontextu.*

Samozrejme musíme rátať s tým, že mechanizmus priameho grafického výstupu počítača na zrakové centrum mozgu, tj. vynechanie „zbytočných článkov“ (výstupná obrazovka - vstupné oko) asi zásadným spôsobom zmení v budúcnosti koncepciu počítačovej grafiky. Výskum v tejto oblasti dnes intenzívne prebieha. No zdá sa, že súčasná koncepcia počítačovej grafiky nám ešte dlho vydrží.

V počítačovom svete pracujeme s grafickou informáciou dvojakým spôsobom, preto sa používajú dva pojmy:

- vektorová grafika,
- rastrová grafika.

1.1 Vektorová grafika

Vo vektorovom vyjadrení je objekt *určený definovanou konštrukciou nad množinou zadaných bodov.* Napr. bod je daný svojimi súradnicami, úsečka svojimi krajnými bodmi, lineárne lomená čiara je daná usporiadanou množinou bodov, n-uholníková plocha (stena) je určená svojou orientovanou hranicou, tj. uzavretou lineárne lomenou čiarou s vybraným smerom usporiadania, mnohosten je daný množinou svojich stien atď.

Nezastupiteľné miesto má vektorový prístup v konštrukčných systémoch CAD/CAM (Computer Aided Design/Manufacturing) napr. v strojníctve, architektúre, pri riadení manipulátorov a robotov, kde trajektórie – krivky určujeme príslušnou postupnosťou bodov. Podobne v geografických informačných systémoch (GIS) veľa geografických objektov prírodného charakteru (výškové izolínie, vodné toky, ...) alebo človekom vytvorené (cesty, inžinierske siete, ...) majú charakter kriviek.

Správnejšie by bolo asi používať názov *procedurálna reprezentácia*.[3] Napr. úsečku charakterizujeme jej krajnými bodmi a atribútmi (typ, hrúbka a farba čiary, prípadne ešte spôsob, akým sa realizujú konce úsečky). Vo svojej podstate je geometria daná len koncovými bodmi, všetko ostatné je vec určitých činností - procedúr: krajné body *spoj* rovnou čiarou, *použij* pri tom zadanú farbu, konce úsečky *označ* predpísaným spôsobom ...

I zložitejšie objekty sa dajú charakterizovať podobne. Napr. často používaný CSG-model (Constructive Solid Geometry) vychádza z

1. definovaných *primitívnych telies* (napr. kváder, valec, rotačný elipsoid),
2. parametrov, určujúcich *tvar primitívneho telesa* (napr. dĺžky hrán kvádra, polomer a výška valca, veľkosti hlavných poloosí elipsoidu),
3. parametrov, určujúcich *polohu a orientáciu telesa* v danom súradnicovom systéme (napr. skutočná poloha vybraných bodov telesa),
4. *atribútov* (napr. spôsob vyfarbenia hraničných plôch),
5. *množinových operácií* nad primitívnymi telesami (výsledné teleso budujeme pomocou zjednotenia, prieniku a rozdielu nad primitívnymi telesami).

Keďže množinové operácie sú binárne, táto reprezentácia vedie k použitiu formalizmu - *binárny strom*. Editácia telies je v tejto reprezentácii veľmi jednoduchá a intuitívne pochopiteľná. Naviac, výsledný binárny strom nesie v sebe celú históriu tvorby daného telesa.

1.2 Rastrová grafika

Je veľa grafických aplikácií, kde sa na obraz pozeráme ako na systém malých, farebne homogénnych plôšok. Takto sú vytvárané mozaiky, takto registruje grafickú informáciu oko. Tento mechanizmus využívame pri technológiách snímania obrazu[4].

V počítačovej grafike pre tieto účely vytvárame pravidelnú štruktúru - pravouhlú mriežku zvanú *raster* s ďalej nedeliteľnými bunkami, ktoré nazývame *pixely* (PIcture ELement).

[3] Ovšem pojem procedurálne modelovanie je v dnešnej dobe používaný v iných súvislostiach (fraktálna geometria, dynamická simulácia) [29].

[4] Toto samozrejme determinuje i následné spracovanie, analýzu a rozpoznávanie v obraze, tj. oblasť počítačového videnia.

V tejto reprezentácii každú zobrazovanú situáciu vyjadrujeme celou rastrovou mriežkou. (Napríklad pri reprezentácii jedného bodu na farebne homogénnom pozadí sú všetky pixely, až na jeden, rovnakej farby). Rastrová reprezentácia je preto pamäťovo veľmi náročná. Pamäť, kde sa táto informácia ukladá, sa nazýva VideoRAM (VRAM).

Rozlíšenie, tj. počet bodov (v megapixeloch Mpix) rastrovej mriežky, ktoré dané zariadenie registruje (resp. zobrazuje) je jednou zo základných charakteristík grafického HW.

Často (hlavne v prípade tlačiarní) sa stretneme s charakteristikou rozlíšenia udávaného v jednotkách *dpi* (Dots Per Inch) tj. počet bodov na jeden palec (cca 2,5cm). Napr. dnes používané tlačiarne mávajú rozlíšenie od 300 do 1200 dpi.

Rozvoj príslušných technológií (pamäte a rýchly prístup do nich) spôsobili, že základné výstupné grafické zariadenia (obrazovka, tlačiareň) sú dnes realizované práve rastrovým spôsobom.[5] K tomu prispieva i fakt, že mnohé algoritmy sú pre rastrovú reprezentáciu oveľa jednoduchšie ako pre reprezentáciu vektorovú. Priblížime si to na jednoduchom príklade.

V prípade vektorovej grafiky pre nájdenie prieniku dvoch úsečiek musíme riešiť sústavu rovníc. Výsledkom sú body, ktoré sú pre nás „nové“, tj. na vstupe východiskových úsečiek tieto body explicitne uvedené nemáme.

Z druhej strany, prienikom objektov zadaných v rasterizovanej podobe je objekt, opäť vyjadrený v rasterizovanej podobe (v tej istej rastrovej mriežke). Hľadané body preto nedostávame riešením rovníc, ale výberom z konečnej množiny bodov rastrovej mriežky. Toto je síce pamäťovo oveľa náročnejšie, ale výpočtovo veľmi jednoduché riešenie.

Samozrejme sa ponúkajú rôzne zovšeobecnenia rastrového prístupu.

- Trojrozmerný ekvivalent rastrovej mriežky je rozklad na *voxely* (VOlume ELement).
- Namiesto pravidelnej štvorcovej mriežky môžeme uvažovať pravidelnú 6-uholníkovú mriežku (zjednodušený variant tohto prístupu je používaný napr. pri zobrazovaní TV-signálu) [18].
- Oblasť sa rozkladá nie pravidelnou mriežkou, ale všeobecnejšou – v praxi sa často používajú triangulácie v 2D a rozklad na štvorsteny v 3D.

V žiadnom prípade však nemožno povedať, že rastrová grafika nahradí grafiku vektorovú. Vekorová grafika ponúka v mnohých prípadoch úspornejší zápis. Hlavne však dovoľuje väčšie možnosti grafickej editácie *celých objektov* a presnejšie geometrické vyjadrenie tvaru modelovaných objektov.

[5] V prvopočiatkoch tomu tak nebolo. Grafické obrazovky boli *vektorové*. Obraz sa v tomto prípade nekreslil po pixeloch, ale podobne ako v detskej hračke „magická tabuľka“. Pohyb elektronového lúča bol ovládaný nezávisle v horizontálnom a vertikálnom smere. Po dopade na luminofor obrazovky sa tak vytvárala čiarová kresba [13].

1.3 Kontrolné otázky

1. Koľko pamäti potrebujeme v prípade rastrovej mriežky 1000x1000 pixelov v prípade, že pre každý pixel potrebujeme 1 byte pamäti?

2. Stotožnite pojmy dot a pixel a vyjadrite rozlíšenie v dpi (resp. ppi = pixels per inch) pre 21 palcový monitor (pri klasickom pomere strán 4:3), ktorý má rozlíšenie 1024x768 pixelov.

3. Vyjadrite veľkosť rastrovej mriežky pre formát A4 pri hustote bodov 300, 600, 1200 dpi.

4. Skúste sa zamyslieť nad tým, ako bude vyzerať zväčšený detail obrázku, ktorý je zadaný vektorovo a obrázku, ktorý je zadaný rastrovo.

1.4 Sprievodca textom

Predkladaný text ukazuje ako riešiť základné problémy počítačovej grafiky. Úvodné kapitoly sú venované rastrovej grafike.

- Keďže základom rastrovej grafiky je pixel, a pixel má jediný atribút - farbu, tak v nasledujúcich dvoch kapitolách ukážeme prístupy pre formalizáciu farebných vnemov.

- Štvrtá kapitola sa venuje základným prístupom pri spracovaní rastrového obrazu. Táto problematika sa s rozširovaním dostupnosti prístrojov, ako sú napr. scanner a digitálny fotoaparát, stáva aktuálnou pre čoraz širší okruh ľudí.

- V piatej kapitole vysvetľujeme základné princípy komprimovania rastrových obrázkov.

Nasledujúce tri kapitoly pojednávajú o rasterizácii obrazu, tj. o prechode od vektorovej grafiky k rastrovej. Dôležitosť tohto procesu je v tom, že základné výstupné zariadenia sú rastrového charakteru, a preto sa rasterizácii vektorových dát nevyhneme.

- Šiesta kapitola demonštruje, akým spôsobom sa prevádza vektorový obraz do rastrovej podoby v najjednoduchšej forme, tj. pre prípad úsečky. Dôležitosť úsečky v počítačovej grafike je dvojaký. Z jednej strany je úsečka veľmi často používaný geometrický objekt. Z druhej strany, úsečka je z algebraického pohľadu *lineárna interpolácia*. Jej použitie je preto veľmi časté. Z tohto dôvodu kladieme dôraz na rýchlosť programovej realizácie. Zmienime sa i o metódach, ktoré vykresľujú úsečku vo vyššej kvalite. Celočíselná rasterizácia je zovšeobecnená na niektoré krivky druhého stupňa a je ukázané obmedzenie použitej idey.

- Siedma kapitola ukazuje ako zvládnuť záludnosti počítačového sveta v situácii, keď je vykresľovaný objekt, alebo jeho časť, mimo zobrazovanú oblasť. Analyzujeme metódy orezávania (pravdaže opäť na jednoduchých príkladoch - úsečkách). Keďže každý vykresľovaný objekt musíme podrobiť analýze, či je alebo nie je nutné ho orezať, i tu nás bude zaujímať časová efektivita algoritmov.

- V ôsmej kapitole sa venujeme vyplňovacím algoritmom. Pritom sa na problém vypĺňovania pozrieme z dvoch strán: 1. problém rasterizácie, 2. problém rastrovej grafiky.

Záverečná časť textu je venovaná 3-rozmernej vektorovej grafike.

- Deviata kapitola vysvetľuje základné operácie pri práci s vektorovo orientovanými dátami, s tuhými transformáciami. Tie nám dovoľujú s objektom hýbať, tj. rotovať, posúvať, približovať ho. Vysvetlíme si i pojem projekcie, tj. spôsob, ako zobraziť 3D objekt na 2D plochu.

- V nasledujúcej kapitole popisujeme veľmi rozšírený mechanizmus, ktorý sa používa pre generovanie kriviek - parametrické vyjadrenie. Formulujeme tu základné vlastnosti Bézierovych kriviek, ktoré sa v dnešnej dobe veľmi často používajú, a ktoré sú dobrým východiskom pre pochopenie všeobecnejšieho nástroja parametrického modelovania kriviek a plôch – NURBS.

- Jedenásta kapitola ukazuje princípy, na ktorých sú konštruované 3D modely. Najväčší priestor je pritom venovaný zovšeobecneniu princípov parametrizácie z predošlej kapitoly.

- Posledná kapitola pojednáva o metódach, ktoré sú nutné k tomu, aby výsledné zobrazenie telies v 3D scéne pôsobilo realistickým dojmom. Vysvetlené je riešenie problému viditeľnosti. Sú tu taktiež ukázané techniky lokálneho a globálneho osvetľovania scény. Na záver sú zmienene princípy výstupných zariadení, ktoré dovoľujú 3-rozmerné vnímanie počítačových modelov.

Nie je cieľom tohto textu podať vyčerpávajúce informácie z oblasti počítačovej grafiky. Záujemcom o detailnejšie prenikutie do problematiky nech poslúži zoznam literatúry.

Kapitola 2

Formalizácia vnímania farieb – čiernobiele videnie

Všetky naše vizuálne vnemy majú jedného spoločného menovateľa - svetlo. Vo svojej fyzikálnej podstate je to elektromagnetické žiarenie. Jeho prirodzeným zdrojom je Slnko. Na rozhraní rôznych prostredí sa svetlo môže lámať, odrážať, pohlcovať, prípadne ohýbať. Napr. na rozhraní sklo-vzduch nastáva lom svetla, pričom uhol lomu vzrastá so zmenšujúcou sa vlnovou dĺžkou. Takto sa dá ukázať, že prirodzené svetlo je zmesou svetiel s rôznymi vlnovými dĺžkami. (V prírode tento jav, keď lom nastáva na rozhraní kvapiek vody a vzduchu, pozorujeme ako dúhu.)

Svetlo s konkrétnou vlnovou dĺžkou nazývame *monochromatické*. Rôzne monochromatické svetlá registrujeme ako rôzne farby. Zdravé ľudské oko je schopné vnímať svetlá s vlnovou dĺžkou λ v rozpätí cca 380 - 780 nm.[1] Prirodzené slnečné svetlo nazývame achromatické, resp. biele, či „bezfarebné“. Možno to interpretovať tak, že v achromatickej zmesi nie sme schopní rozlíšiť, ktorá z farieb je dominantná, tj. všetky sú vo „vzájomnej rovnováhe“. Rozklad zmesi svetiel rôznych farieb na monochromatické svetlá nazývame spektrum.

U svetiel prirodzeným spôsobom rozlišujeme tri vlastnosti:

1. *intenzitu* - tj. mieru energie žiarenia,
2. *farebný tón* - ktorý súvisí s vlnovou dĺžkou,
3. *sýtosť* - tj. mieru primiešavania achromatického svetla k svetlu monochromatickému.

Druhej a tretej vlastnosti svetla je venovaná ďalšia kapitola. V tejto kapitole budeme analyzovať prvú z uvedených vlastností.

[1] Človek vníma elektromagnetické žiarenie väčšieho rozsahu – ale nie zrakom: infračervené žiarenie ($\lambda > 780$ nm) vnímame ako teplo, na UV-žiarenie ($\lambda < 380$ nm) reaguje naša pokožka zhnednutím.

U monochromatického svetla (a taktiež u achromatického) vnímame len jeden atribút *intenzitu*. Je to miera energie daného žiarenia. V prípade bieleho svetla intenzitu popisujeme ako rôzne úrovne šedej farby. Bežne sme schopní rozlíšiť približne 100 úrovní.[2] Preto je rozumné škálu intenzít *kvantovať*, tj. previesť ju zo vstupnej – spojitej škály hodnôt do výstupnej – diskrétnej škály.

Napr. pre fungovanie tlačiarní, kde jediný čierny pigment zabezpečí tlač širokej škály šedých odtieňov, je veľmi významná *výstupná dvojstupňová škála*. V tejto kapitole si ukážeme mechanizmy, ako nahradiť spojitú škálu intenzít achromatického svetla konečným počtom intenzít tak, aby vizuálny vnem zostal podľa možností nezmenený.

2.1 Metóda konštantného prahu

Označme vstupnú resp. výstupnú hodnotu pixelu `(ix,iy)` `IN[ix][iy]` resp. `OUT[ix][iy]`. Predpokladajme že

- $\langle 0, L)$ je interval vstupných hodnôt intenzity pixelov,
- $0 = h[0] < h[1] < \cdots < h[n] = L$ sú vopred zvolené prahové hodnoty,
- $\{0, 1 \ldots, n-1\}$ je množina výstupných hodnôt.

V prípade tlače za 0-tý stupeň považujeme biely (čistý) papier a postupným pridávaním intenzít vyrábame tmavšie šedé odtiene. Maximálnej hodnote bude odpovedať čierna farba. V prípade výstupu na obrazovku je to naopak, tj. 0-tý stupeň je čierny a maximálnej hodnote odpovedá biela farba.

Diskretizácia vstupnej hodnoty `IN[ix][iy]` pixelu `(ix,iy)` sa dá vyjadriť napr. takto:

```
i=1; while (IN[ix][iy]>h[i]) i++;
OUT[ix][iy]=i-1;
```

Vo svojom najjednoduchšiom variante - dvojstupňovej škále - dostávame:

```
if (IN[ix][iy]>h[1]) OUT[ix][iy]=1;
else                 OUT[ix][iy]=0;
```

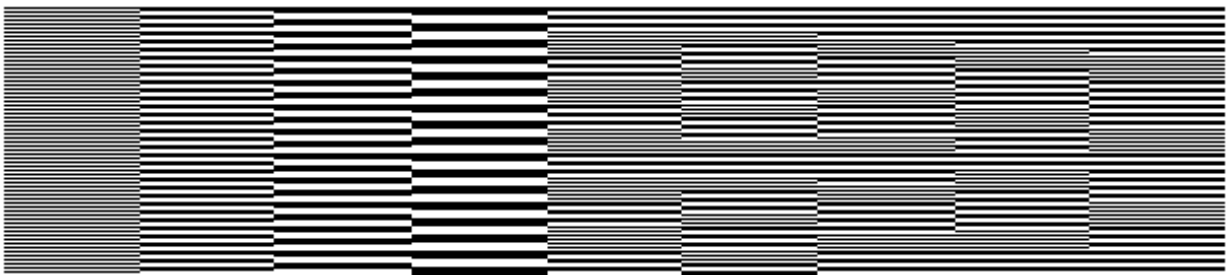
kde spravidla je zvolená prahová hodnota $h[1] = L/2$.

Tento spôsob kvantovania nazývame *metóda konštantného prahu*. Typické HW zariadenie, ktoré pracuje len s 2-stupňovou výstupnou škálou farieb, je čierno-biela tlačiareň. Používa síce len jeden - čierny toner, no napriek tomu dokáže vyrobiť celú paletu odtieňov šedej farby. Nie je ťažké vidieť, že vyššieopísaná metóda konštantného prahu nedáva v tomto prípade uspokojivý výsledok.

[2] Z dôvodov používaných číselných reprezentácií sa spravidla používajú škály 2-stupňová (1 bit), 16-stupňová (4 bity), 256 stupňová (8b = 1 Byte). Často je využívaná 64-stupňová škála (6 bitov).

2.2 Podstata emulácie

Ako každý iný receptor, aj oko má obmedzenia geometrického i energetického vnímania. Tj. objekty, ktoré sú vizuálne menšie ako nejaká medzná hodnota, nevidíme. Podobne objekty, ktorých celková vyžarovaná energia je menšia, ako nejaká prahová hodnota, sú pre nás čierne. No tieto podprahové hodnoty oko neignoruje, ale je schopné ich sčítať. Tento jav sa nazýva *integračná schopnosť oka*. Napr. na nočnej oblohe voľným okom vidíme Mliečnu cestu - svetlý pás, tiahnúci sa celou oblohou (najlepšie vidieť v lete). V skutočnosti je to obrovské množstvo hviezd, ktoré sú z hľadiska nášho vnímania podprahové (možno sa presvedčiť pohľadom cez ďalekohľad). Podobne obr.2.1 demonštruje integračnú schopnosť oka pre rôzne vzory. Každý stĺpec obsahuje rovnaký počet bielych a čiernych čiar. Kód vzoru m(a)n(b) znamená, že m-krát sa opakuje postupnosť a-bielych a a-čiernych pixelov a potom n-krát sa opakuje b-bielych a b-čiernych pixelov. Napr. 3(2)6(1) je kód piateho (zľava) stĺpca. Pri zväčšujúcej sa vzdialenosti od obrázku prestávame vnímať štruktúru vzorov – príslušné plochy sa budú javiť ako šedé. Všimnite si netriviálny jav – štruktúru tretieho a štvrtého stĺpca prestávame vnímať z menšej vzdialenosti ako štruktúru stĺpcov vpravo.



Obr. 2.1: Integračná schopnosť oka pre rôzne typy vzorov: (1), (2), (3), (4), 3(2)6(1), 3(2)3(1), 4(2)4(1), 5(2)5(1), 6(2)6(1).

Hlavná myšlienka emulovania šedých odtieňov vychádza z predpokladu, že pixel má podprahovú veľkosť. Pritom sa berie do úvahy integračná schopnosť oka. Zrakom vnímanú najmenšiu oblasť budeme nazývať *elementárna oblasť*. V našich úvahách budeme postupovať tak, že pôvodnú šedú štvorcovú elementárnu oblasť nahradíme štvorcom $n \times n$ čiernych a bielych pixelov. Stupeň šedi budeme vnímať ako pomer počtu bielych a čiernych pixelov v danom štvorci. Štvorec so stranou n pixelov sme tak schopní vnímať v $n^2 + 1$ odtieňoch šedej farby.

Metódy, ktoré toto realizujú, tj. vyrábajú nami zadanú škálu odtieňov pri obmedzenej farebnej palete, nazývame *poltónovacie metódy* (halftoning), resp. *rozptyľovacie metódy* (dithering).

Je dôležité si uvedomiť, že dnešné monitory hardverovo zabezpečujú dostatočnú škálu intenzít pre jednotlivé pixely. Preto v prípade výstupu na monitor emulačné techniky nepoužívame. Z druhej strany, tlačiarenské technológie využívajú technológie emulácie škály

odtieňov jednej farby, čo zvyšuje nároky na použité rozlíšenie. To je taktiež dôvod pre rozlišovanie pojmov *pixel* a *dot*.

2.3 Náhodné rozptyľovanie

Uvažujme nasledujúci experiment. Zoberme čísla $0 \leq a \leq b$ a budeme „náhodne strieľať“ do intervalu $\langle 0, b\rangle$. Bude nás pritom zaujímať, koľkokrát z toho sa trafíme do podintervalu $\langle 0, a\rangle$. Predpokladáme pritom, že každá z hodnôt intervalu $\langle 0, b\rangle$ je rovnako pravdepodobná. Pravdepodobnosť úspešného zásahu (tj. do intervalu $\langle 0, a\rangle$) je daná podielom $\frac{a}{b}$.

Nech náhodný nástrel do intervalu $\langle 0, b\rangle$ realizuje funkcia `random(b)`. Počet úspešných zásahov OK pre dostatočne veľký počet opakovaní n sa bude blížiť hodnote $n\frac{a}{b}$ a proces výpočtu dá sa vyjadriť takto:

```
OK=0;
for(i=0; i<n; i++)
  if (a>random(b)) OK++;
```

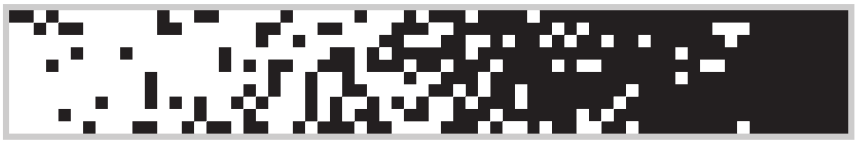
Teraz to aplikujme na prípad emulovania šedých odtieňov na plochu $nx \times ny$ pixelov:

1. Hodnotu b nahradíme maximálnou možnou intenzitou L.
2. Hodnotu a nahradíme vstupnou intenzitou pixelu `IN[ix][iy]`.
3. Úspešný zásah znamená, že výstupný pixel bude mať hodnotu 1.

```
for(ix=0; ix<nx; ix++)
for(iy=0; iy<ny; iy++)
  if (IN[ix][iy] > random(L) OUT[ix][iy]=1;
  else                       OUT[ix][iy]=0;
```

Uvažujme pre jednoduchosť celú plochu $n = nx \times ny$ pixelov s konštantným stupňom vstupnej intenzity c. Počet „zásahov“, tj. počet pixelov s výstupnou hodnotou 1 bude pre dostatočne veľkú plochu približne $n\frac{c}{L}$, tj. v priemere na jeden pixel dostávame $\frac{c}{L}$ jednotiek, čo odpovedá pomernej hodnote vstupnej intenzity c. To znamená, že metóda sa správa podľa nášho očakávania.

Nevýhodou tejto metódy je, že náhodné rozmiestnenie čiernych a bielych pixelov spôsobuje, že vznikajú väčšie čierne, resp. biele súvislé plôšky. Majú rôzne tvary a rozmery, ktoré sú v niektorých prípadoch nad prahom nášho priestorového rozlíšenia (tj. väčšie ako elementárna plocha). Dôsledkom toho vnímame výsledný obraz ako silne zrnitý. Preto sa budeme snažiť riadiť vzájomnú polohu rozmiestňovania čiernych a bielych pixelov podľa nejakej pravidelnej schémy, napr. tak, aby boli podľa možností rozmiestnené rovnomerne.



Obr. 2.2: Výsledok emulácie škály šedých odtieňov náhodným rozptýlením.

2.4 Maticové rozptýlenie

V nižšieopísanej metóde nahradíme vstupný pixel štvorcom $n \times n$ čiernych resp. bielych pixelov. To z jednej strany zväčšuje pôvodný obrázok, no z druhej strany nám to dovoľuje emulovať $n^2 + 1$ stupňovú škálu šedých odtieňov $\{0, 1, \ldots, n^2\}$.

Budeme využívať taký spôsob generovania vzorov, kde dvojici susedných vstupných intenzít odpovedajú vzory, ktoré sa líšia len v jednej pozícii. Tento mechanizmus dovoľuje všetky vzory vyjadriť jedinou kódovacou maticou – *maskou* $M = (m_{ij})$, $1 \leq i, j \leq n$, kde všetky hodnoty m_{ij} sú vzájomne rôzne. Pre danú vstupnú intenzitu h bude výsledný vzor obsahovať všetky tie prvky, pre ktoré $m_{ij} \leq h$.

Pre elementárnu oblasť 2×2 pixely, tj. pre 5 stupňov šedej farby, máme tri možné masky (s presnosťou na symetrie): diagonálnu M_d, rotačnú M_r a líniovú M_l.

$$M_d = \begin{pmatrix} 1 & 3 \\ 4 & 2 \end{pmatrix}, \qquad M_r = \begin{pmatrix} 1 & 2 \\ 4 & 3 \end{pmatrix}, \qquad M_l = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}.$$

Napr. pre masku M_d dostávame vzory z obr. 2.3.

Obr. 2.3: Emulácia šedej farby maticou M_d.

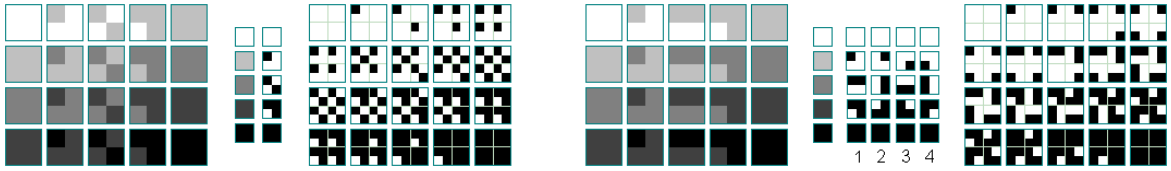
2.4.1 Viacúrovňová emulácia

Päť úrovní intenzity je pre praktické účely málo, preto sa v praxi používajú väčšie matice. Napr. 8×8 matica dovoľuje emulovať 65 úrovní čo už býva spravidla dostatočné. My sa pre jednoduchosť obmedzíme na matice 4×4.

$$M_D = \begin{pmatrix} 1 & 9 & 3 & 11 \\ 13 & 5 & 15 & 7 \\ 4 & 12 & 2 & 10 \\ 16 & 8 & 14 & 6 \end{pmatrix}, \qquad M_R = \begin{pmatrix} 1 & 5 & 14 & 2 \\ 13 & 9 & 10 & 6 \\ 8 & 12 & 11 & 15 \\ 4 & 16 & 7 & 3 \end{pmatrix}. \tag{2.1}$$

Maska M_D vzniká jednostupňovým vnorením M_d do seba – obr. 2.4. Môžeme to interpretovať tak, že

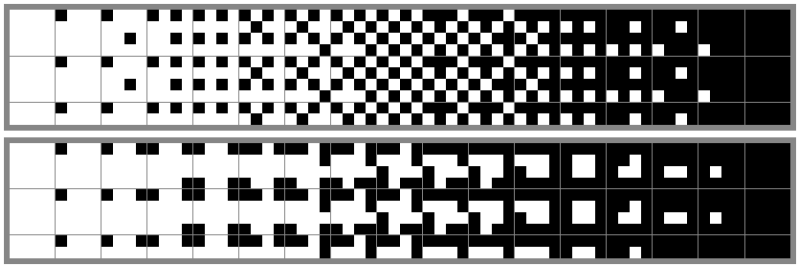
- aplikujeme masku M_d postupne pre dvojice intenzít 0-1, 1-2, 2-3, 3-4 – obr. 2.4a),
- každý stupeň šedi 0 - 4 nahradíme príslušným vzorom, definovaným maskou M_d – obr. 2.4b),
- výsledok dvojúrovňovej emulácie vidíme na obr. 2.4c).



Obr. 2.4: Konštrukcia dvojúrovňového vzoru na základe matice M_d a)-c) a na základe matice M_r d)-f) .

Maska M_R vzniká na podobnom princípe. Masku M_r aplikujeme postupne pre dvojice intenzít 0-1, 1-2, 2-3, 3-4 – obr. 2.4d). Okrem vnorenia však vkladané matice M_r otáčame o 90^o v smere hodinových ručičiek. Pre každý stupeň tak dostávame štyri rôzne pozície – obr. 2.4e). Tie pri vkladaní postupne prestriedavame – obr. 2.4f).

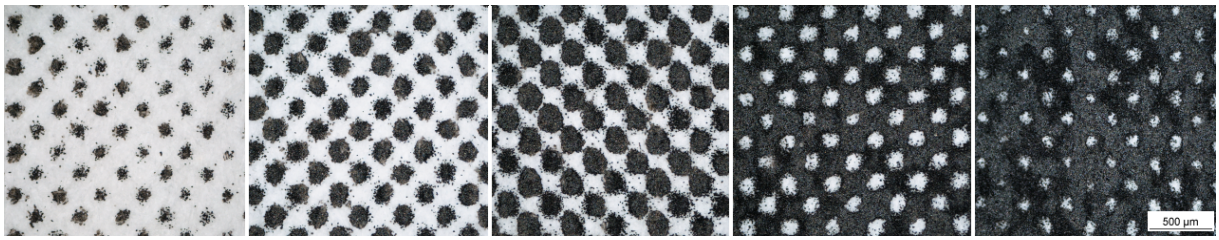
Na obr.2.5 vidíme, že maska M_D rozmiestňuje čierne a biele pixely „čo najrovnomernejším spôsobom“, kým maska M_R čierne plôšky postupne zväčšuje.



Obr. 2.5: Masky typu M_D sa používali pri emulácií jemných odtieňov farieb v starších typoch monitorov. Ako bolo zmienené vyššie, dnešné monitory zvládajú zobraziť dostatočnú škálu odtieňov rôznych farieb a spravidla emuláciu nepotrebujú.Emulácia šedých odtieňov maskou M_D(hore) a maskou M_R(dole).

Tlačiarenské tecMasky typu M_D sa používali pri emulácií jemných odtieňov farieb v starších typoch monitorov. Ako bolo zmienené vyššie, dnešné monitory zvládajú zobraziť dostatočnú škálu odtieňov rôznych farieb a spravidla emuláciu nepotrebujú.hnológie využivajú masky typu M_R: aby sa dosiahla kvalitná tlač čiernej súvislej plochy, vytlačené čierne body

rastru sa musia trochu prekrývať tj. sú väčšie, ako biele body rastru. Maska M_R je na rôznu veľkosť čiernych a bielych bodov rastru očividne menej citlivá ako maska M_D[3].



Obr. 2.6: Rôzne stupne šedej – detail výstupu z laserovej tlačiarne.

2.4.2 Korekcia veľkosti obrázku

Ako už bolo uvedené, priama aplikácia vyššieuvedených maticových metód zväčšuje veľkosť výsledného obrázku. Pre zachovanie pôvodného rozmeru postupujeme nasledujúco:

emulácia – hodnotu každého pixelu nahradíme príslušným vzorom, tj. maticou pixelov,

redukcia – z príslušnej matice pixelov odpovedajúcich vzoru vyberieme len jeden pixel.

Pre jednoduchosť vykresľovania si to ukážeme na jednorozmernom príklade. Majme vstupný jednoriadkový obrázok $IN[6] = [0\ 0\ 3\ 0\ 0\ 3]$ a jednorozmernú masku $M = (1\ 3\ 2)$. Nahradenie vstupných hodnôt odpovedajúcim maticovým vzorom[4] zväčší pôvodný obrázok trikrát (v horizontálnom smere). Preto pre zachovanie pôvodnej veľkosti obrázku musíme ho v horizontálnom smere redukovať na 1/3.

index	1			2			3			4			5			6		
IN[`index]	0			0			3			0			0			3		
emulácia	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**
redukcia	**0**			**0**			**1**			**0**			**0**			**1**		

Obr. 2.7: Dva kroky emulácie.

V uvedenom príklade je jedno, akým spôsobom obrázok redukujeme, tj. ktorú z hodnôt masky vyberáme. Všeobecne to však neplatí. Ak by sme napr. vždy vyberali pixel z jednej a tej istej pozície masky, výsledok nebude odpovedať našim očakávaniam viď obr. 2.8 – redukcia 1.

[3]Masky typu M_D sa používali pri emulácií jemných odtieňov farieb v starších typoch monitorov. Ako bolo zmienené vyššie, dnešné monitory zvládajú zobraziť dostatočnú škálu odtieňov rôznych farieb a spravidla emuláciu nepotrebujú.

[4]Namiesto čiernych a bielych pixelov budeme používať **0** a **1**.

index	1	2	3	4	5	6
IN[index]	2	2	2	2	2	2
emulácia	**1** **0** **1**	**1** **0** **1**	**1** **0** **1**	**1** **0** **1**	**1** **0** **1**	**1** **0** **1**
redukcia 1	**1**	**1**	**1**	**1**	**1**	**1**
redukcia 2	**1**	**0**	**1**	**1**	**0**	**1**

Obr. 2.8: Redukcia 1 s výberom hodnoty z konštantnej pozície v maske (prvá pozícia), redukcia 2 s výberom hodnoty v maske postupne zľava do prava.

Pri redukcii je nutné prestriedať všetky možné pozície vzoru. Najjednoduchšie je vyberať pozície postupne, viď obr. 2.8 – redukcia 2.

Postupný výber pozície v maske veľkosti n (v závislosti na indexe spracovávaného pixelu i) sa dá vyjadriť ako zvyšková trieda pri celočíselnom delení

$$i \bmod n = i - n(i \operatorname{div} n).$$

V prípade štvorcových masiek $n \times n$ postupujeme v oboch smeroch rovnako. Preto výsledné formálne vyjadrenie maticovej emulácie škály šedých odtieňov sa dá vyjadriť schémou

```
for(ix=0; ix<nx; ix++)
for(iy=0; iy<ny; iy++)
  if (IN[ix][iy]>M[ix mod n][iy mod n]) OUT[ix][iy]=1;
  else                                  OUT[ix][iy]=0;
```

kde `M[n][n]` je maska použitého vzoru veľkosti $n \times n$.

2.5 Distribúcia chyby

Vráťme sa na chvíľu k metóde konštantného prahu pre 2-stupňovú výstupnú škálu. Nahradenie vstupnej hodnoty výstupnou vedie k strate farebnej informácie. Pritom najväčšej chyby sa dopúšťame pri hodnotách z okolia prahu.

Napr. pri 64 stupňovej vstupnej škále a prahu $h = 31$, sa vstupná hodnota $IN = 31$ zmení na $OUT = 0$, no jej blízka vstupná hodnota $IN = 32$ sa zmení na $OUT = 1$.

Myšlienka, ako túto chybu potlačiť, spočíva v prenose informácie, ktorá sa prahovaním stratí, do ešte neanalyzovaného okolia. Tj. keď pri použití konštantného prahu pre vstupnú hodnotu $IN \leq h$ všetkých IN jasových jednotiek nenávratne stratíme, teraz sa budeme snažiť vzniknutý deficit eliminovať tak, že práve o túto stratenú hodnotu zvýšime hodnotu vstupu v ešte neanalyzovanom okolí. Naopak, v prípade, že pôvodná hodnota vstupnej intenzity je väčšia ako prahová, dosadením maximálnej možnej výstupnej hodnoty neúmerne

zvýšime výstupnú hodnotu daného pixelu. Preto teraz chybu eliminujeme tak, že v okolí znížimeme hodnotu vstupu. Týmto postupom sa snažíme zachovať bilanciu vyžarovaného svetla v okolí daného pixelu, čo dobre korešponduje s integračnou schopnosťou oka.

Proces spracovania prebieha spravidla zhora dole, zľava do prava. Preto okolie, kam distribuujeme chybu prahovania, je pravé dolné. Keďže susednosť dvoch pixelov môže byť daná buď celou spoločnou hranou, alebo len vrcholom, chybu rozložíme nerovnomerne - hranovo susedným pixelom viac ako susedom cez vrchol. Napr. Floyd-Steinbergova metóda [23] používa hodnoty podľa obr.2.9. Hodnoty príslušných častí distribuovanej chyby do susedných pixelov môžu byť podľa rôznych autorov rôzne [29].

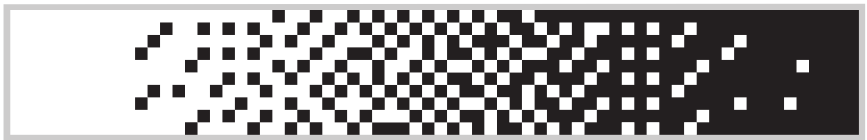
0	0	0
0	*	3/8
0	3/8	1/4

Obr. 2.9: Podiel distribuovanej chyby do okolia analyzovaného pixelu (označený *).

Schéma, ktorá distribúciu chyby pre vnútorný pixel `(ix,iy)` realizuje, sa dá vyjadriť napr. takto[5]:

```
e = IN[ix][iy]; h=L/2;
if (e >= h) {OUT[ix][iy]=1; e -= L;}
else         OUT[ix][iy]=0;
eh=3*e/8; ev=e-2*eh;
IN[ix+1][iy]   += eh;
IN[ix][iy+1]   += eh;
IN[ix+1][iy+1] += ev;
```

Všimnite si výpočet hodnoty `ev`. Vzhľadom k nutnej konverzii z reálnej do celočíselnej aritmetiky je tento spôsob zaťažený menšou chybou ako priamy výpočet ev=e/4 . Výsledkom pre škálu šedých odtieňov je obr. 2.10.

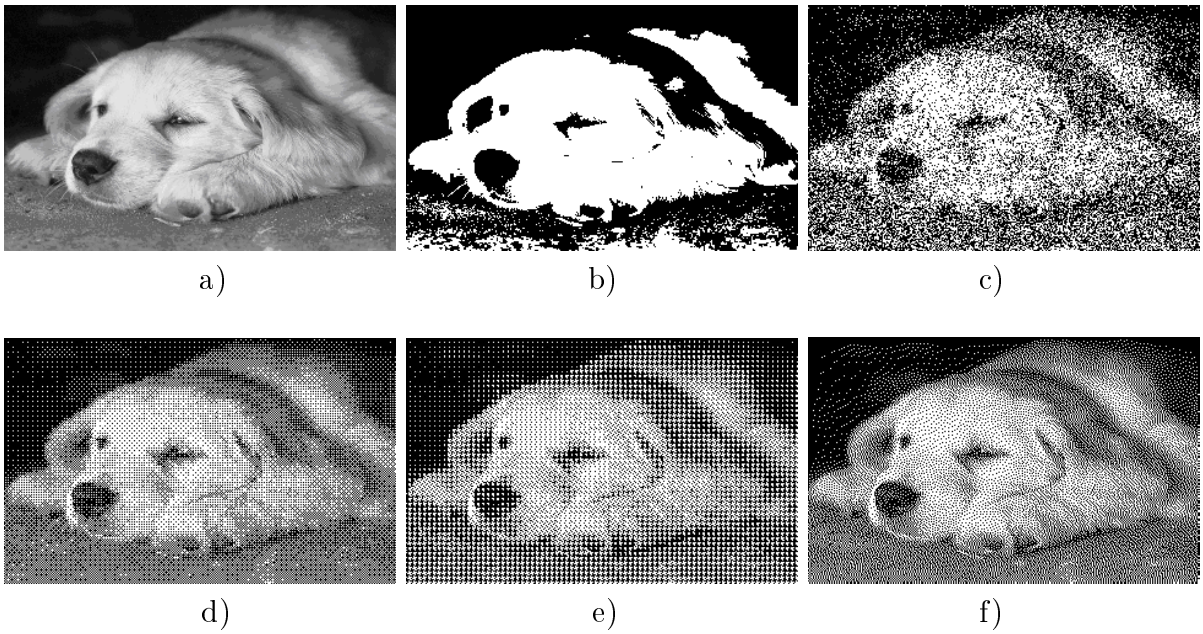


Obr. 2.10: Distribúcia chyby – Floyd-Steinbergova metóda.

Na rozdiel od maticovej emulácie, ktorej pravidelná štruktúra môže pôsobiť rušivo, tento prístup, vďaka zaokrúhľovaniu distribuovanej časti chyby, dáva výsledku trochu náhodný charakter. V porovnaní s náhodným rozptyľovaním je však vnem podstatne kvalitnejší.

Obr. 2.11 ukazuje rôzne metódy emulácie šedých odtieňov. V maticových rozptyľovacích metódach je najprv škála redukovaná na 16 stupňov a potom sú aplikované masky (2.1).

[5]Necháva sa na čitateľovi správne ošetriť konverzie medzi reálnou a celočíselnou aritmetikou.



Obr. 2.11: Porovnanie rôznych metód emulácie šedých odtieňov. a) Pôvodný obraz v 256 stupňovej škále. b) Metóda konštantného prahu. c) Náhodné rozptýlenie. d) Rozptýlenie maticou M_D. e) Rozptýlenie maticou M_R. f) Metóda distribúcie chyby.

2.6 Gamma korekcia

Vo všetkých vyššieuvedených metódach sme predpokladali, že stupeň šedého odtieňa je určený počtom bielych resp. čiernych pixelov v uvažovanej elementárnej ploche. To však znamená, že pre vzťah vnímanej intenzity I (tj. polohy v škále šedých odtieňov) a svetelnej energie W, ktorá toto spôsobí, predpokladáme lineárnu závislosť

$$I \sim W$$

V skutočnosti je vzťah medzi vnímanou intenzitou svetla a jeho energetickou bilanciou (Lambert-Beerov zákon) logaritmický,

$$I \sim \log(W).$$

Znamená to, že keď chceme dosiahnuť aby sa intenzita skutočných vnemov postupne rovnomerne zväčšovala, dodanú energiu musíme znásobovať:

$$I_2 = 2I_1 \Leftrightarrow W_2 = W_1^2.$$

To je dané fyziológiou vizuálneho vnímania. Preto škála šedých odtieňov, konštruovaná na základe vyššieuvedených metód, je z hľadiska zrakového vnemu nerovnomerná. Podobne pri snímaní obrázkov (scannerom, či digitálnym fotoaparátom) a taktiež pri zobrazovaní na monitor, je každý krok poznamenaný nelinearitou vzťahu intenzita - energia.

Výsledný nesúlad rieši *gamma korekcia*, ktorá býva implementovaná v softvéroch na úpravu obrazu - pozri kapitolu 4.4.

Z dostupných internetových zdrojov o gamma korekcii doporučujem napr.[32],[33],[34].

2.7 Zhrnutie a úlohy

- Popísali sme základné princípy emulovania škály šedých odtieňov pri použití obmedzenej palety farieb. Všetky metódy využívajú integračnú schopnosť oka.
- Kontrolné otázky:
  1. Ktorý z typov emulácie odtieňov šedej je použitý v tlačiarenských technológiách?
  2. Ako sa uplatňuje integračná schopnosť oka v popísaných metódach?
  3. Prečo potrebujeme u tlačiarní podstatne jemnejšie rozlíšenie (bežne nad 300 dpi) v porovnaní s monitorami (pod 100 ppi[6])?
- Samostatné úlohy:
  1. Realizujte emuláciu šedých odtieňov na základe maticového rozptýlenia.
  2. Emulujte šedé odtiene s pomocou distribúcie chyby.

[6] ppi = Pixels Per Inch

Kapitola 3

Formalizácia vnímania farieb – farebné videnie

Farebné videnie berieme ako životnú samozrejmosť, no jeho jeho presný popis presahuje možnosť výkladu v tomto texte. Na sietnici oka sa nachádzajú dva druhy receptorov. *Tyčinky (rods)* slúžia pre nočné - čiernobiele videnie, *čapíky (cones)* zabezpečujú denné - farebné videnie. Rozlišujeme tri druhy čapíkov podľa toho, na aké vlnové dĺžky sú citlivé. Závislosť výsledného vnemu na spektre vstupného žiarenia je netriviálna. My sa budeme venovať len základným otázkam formalizácie farebných vnemov. Záujemci o detailnejšie znalosti z oblasti farebného vnímania nech siahnu napríklad po zdrojoch [12], [8], [20], [24], [29], [14].

3.1 Grassmannove zákony

Ako sme sa zmienili v úvode predchádzajúcej kapitoly, významnou vlastnosťou farebného vnímania je skutočnosť, že monochromatické svetlá rôznych vlnových dĺžiek vnímame ako rôzne farby.

λ [nm]	380-435	435-500	500-520	520-565	565-590	590-625	625-780
farba	fialová	modrá	tyrkysová	zelená	žltá	oranžová	červená

Tabuľka 3.1: Vzťah vlnovej dĺžky λ [nm] a farebného vnemu.

Veľmi netriviálny je *vnem zmesi monochromatických svetiel*: jednotlivé monochromatické zložky svetiel nie sme schopní rozlíšiť a takýto podnet taktiež vnímame ako jedinú farbu. Napr. na jednej strane existuje žlté monochromatické svetlo, tj. žiarenie konkrétnej vlnovej

dĺžky (napr. 580 nm), no na druhej strane, vnem žltého svetla môžeme získať aj ako zmes červeného a zeleného monochromatického svetla. Oba tieto prípady sú z hľadiska fyzikálnej podstaty rôzne, no z hľadiska nášho vnímania rovnaké.[1] Podstatné vlastnosti nášho farebného vnímania sú sformulované v troch Grassmannových zákonoch [12].

1. Ľubovoľná štvorica farieb je závislá (tj. ľubovoľný farebný podnet možno nahradiť aditívnou zmesou troch nezávislých farebných podnetov).

2. Pri spojitej zmene spektrálneho zloženia svetla sa spojite mení aj náš vnem – farba (samozrejme to platí len pre viditeľnú časť spektra – viď pozn. 1 na začiatku predošlej kapitoly).

3. Farba zmesi je určená len farbou miešaných zložiek a nezávisí od ich spektrálneho zloženia (tj. napríklad vo vyššieuvedenom príklade je jedno, či budeme k nejakej farbe pridávať monochromatickú žltú, alebo žltú, ktorá je zmesou červenej a zelenej – výsledný farebný vnem bude rovnaký).

3.2 Aditívny a subtraktívny model miešania farieb

Je všeobecne známe, že žltá farba na monitoroch je výsledkom zmiešavania červeného a zeleného svetla. Môžeme sa o tom presvedčiť napr. pohľadom na obrazovku s použitím lupy. Naproti tomu, každý z nás má skúsenosť, že zmiešaním červenej a zelenej vodovej farby nikdy nedostaneme farbu žltú. Toto je dôsledok rôznych mechanizmov, ktoré sa vo vyššieuvedených situáciách uplatňujú.

Keď na biely papier svietime červeným reflektorom, papier toto svetlo odráža a „svieti červene“. Analogicky, pri použití zeleného reflektoru bude papier „svietiť zelene“. To znamená, že keď súčasne svietime oboma reflektormi, budú sa odrážať obe svetlá a my vnímame ich zmes. No a takto vytvorenú červeno-zelenú zmes nazývame žltá farba. Toto je mechanizmus, označovaný ako *aditívne miešanie farieb*. Na tomto princípe funguje napr. farebný výstup na monitor, kde zmes farebných vnemov je dôsledkom nám už známej integračnej schopnosti oka, uplatnenej na trojicu základných farebných bodov, z ktorých je pixel na obrazovke tvorený.

V prípade miešania vodových farieb je situácia úplne iná. Pigmentové zrnká, ktoré farbu tvoria, sú vlastne malé farebné filtre. Uvažujme napr. červený filter, ktorý prepustí zo zmesi svetiel len červené svetlá – ostatné zložky sa pohltia. Keď ešte pred červený umiestnime zelený filter, ktorý prepúšťa len zelené svetlo a ostatné pohlcuje, takáto sústava filtrov neprepustí žiadne svetlo.[2] Tento mechanizmus nazývame *subtraktívne miešanie farieb*. Toto

[1] Úplne iná situácia je u vnemov zvukových, ktoré majú tiež charakter vlnenia: z akordu, tj. zmesi tónov, vždy vieme vyseparovať jednotlivé tóny, ktoré daný akord tvoria, ale žiadny z akordov nemožno nahradiť jediným tónom.

[2] Pre korektonosť vyššieuvedeného mechanizmu je nutné dodať, že v tejto úvahe predpokladáme monochromatické filtre, tj. také, ktoré prepúšťajú len svetlo vybranej vlnovej dĺžky.

je podstata fungovania farebných tlačiarní. Preto výsledným vnemom miešania červenej a zelenej vodovej farby by mala byť farba čierna. Že tomu tak nie je, je dôsledkom toho, že nie sme schopní zaistiť, aby jednotlivé pigmentové zrnká boli rovnakého tvaru a veľkosti. Naviac, nie sme schopní docieliť ich presné prekrytie pri nanášaní na papier.[3] Vyššieuvedené tiež objasňuje, prečo sa okrem farebných pigmentov používa aj pigment čierny. (Nehľadiac na to, že i v prípade „dokonalých pigmentov“ by „dvojfarebné“ generovanie čiernej farby bolo zbytočne zložité a drahé.) Ďalej v celej tejto kapitole budeme uvažovať aditívny farebný model.

3.3 Kolorimetrický priestor

Kvantitatívne vyjadrenie prvého Grassmannovho zákona je dané kolorimetrickými vzťahmi a je výsledkom nasledujúceho experimentu.

Zoberieme tri reflektory rôznych farieb[4]. Tieto svetlá nazveme etalónové. Svietime nimi na jedno miesto. Pritom intenzity jednotlivých reflektorov môžeme rôzne meniť – dostaneme tak rôzne svetelné zmesi.

Na druhej strane, zoberme monochromatické svetlo vybranej vlnovej dĺžky (označme ho X). Budeme sa snažiť namiešať svetelnú zmes tak, aby sme ju farebne vnímali rovnako, ako monochromatické svetlo X. Formálne vyjadrené dostaneme

$$X = rR + gG + bB,$$

kde r, g, b sú reálne nezáporné čísla vyjadrujúce intenzitu príslušných etalónových svetiel. Môžeme sa obmedziť na $0 \leq r, g, b \leq 1$.

Ukazuje sa však, že nie pre všetky monochromatické svetlá môžeme takto vyrobiť zhodné farebné vnemy. V niektorých prípadoch je nutné experiment modifikovať tak, že sa hľadá farebná rovnosť zmesi skúmaného a jedného z etalónových svetiel, so zmesou ostatných dvoch etalónových svetiel. Najmarkantnejšie sa to prejavuje v oblasti vlnových dĺžok 450-550 nm, kde ku skúmanému monochromatickému svetlu X je treba pridať červené svetlo. Tentokrát teda hľadáme rovnosť

$$X + rR = gG + bB.$$

Pri prepise do tvaru

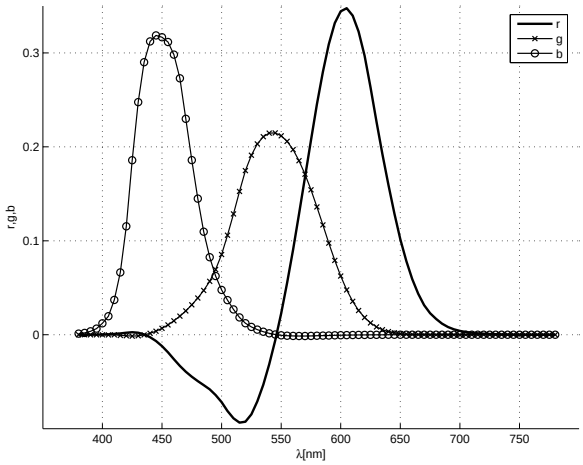
$$X = -rR + gG + bB$$

[3] A samozrejme nie sme schopni zabezpečiť presnú monochromatickosť pigmentov.

[4] Za základ kolorimetrickej sústavy boli Medzinárodnou komisiou pre osvetlenie – CIE (Commission Internationale de l' Éclairage) v r.1931 vzaté žiarenia o vlnových dĺžkach 700 nm (R-červená), 546,1 nm (G-zelená), 435,8 nm (B-modrá).

tak dostávame zápornú hodnotu – koeficient $-r$. Treba si uvedomiť, že r, g, b sú intenzity, a preto samy nemôžu byť záporné.

Výsledné hodnoty koeficientov r, g, b pre jednotlivé vlnové dĺžky sú uvedené v grafe - obr. 3.1, resp. v tab. 3.2, [12], [24].



Obr. 3.1: Farebné koeficienty.

3.4 Normovaný kolorimetrický priestor

Všimnime si farebné vnemy odpovedajúce vlnovej dĺžke $\lambda > 690$nm. Všetky sa dajú vyjadriť len na základe červeného podnetu rôznej intenzity. Ináč povedané, všetky tieto farby môžeme vyjadriť ako výsledok vzájomného pomeru *r:g:b = 1:0:0*, no pri rôznej intenzite samotného vnemu. Podobne aj u ostatných vlnových dĺžok je vo vyjadrení z tab. 3.2 zahrnutá okrem vzájomného pomeru aj celková intenzita jednotlivých zložiek.

Aby sme sa priblížili popisu farebného vnímania v zmysle úvodu kap. 2, je treba oddeliť vzájomné pomery základných vnemov a intenzitu výsledného vnemu.

Preto sa namiesto hodnôt r, g, b používajú pomerné hodnoty

$$\rho = \frac{r}{r+g+b}, \qquad \gamma = \frac{g}{r+g+b}, \qquad \beta = \frac{b}{r+g+b}. \tag{3.1}$$

Vzhľadom na to, že $\rho + \gamma + \beta = 1$, tieto veličiny vyjadrujú *pomerné zastúpenie etalónovej farby* pre danú konkrétnu farbu, tj. *váhu*[5] danej farby. Nazývame ich *normované farebné koeficienty*.

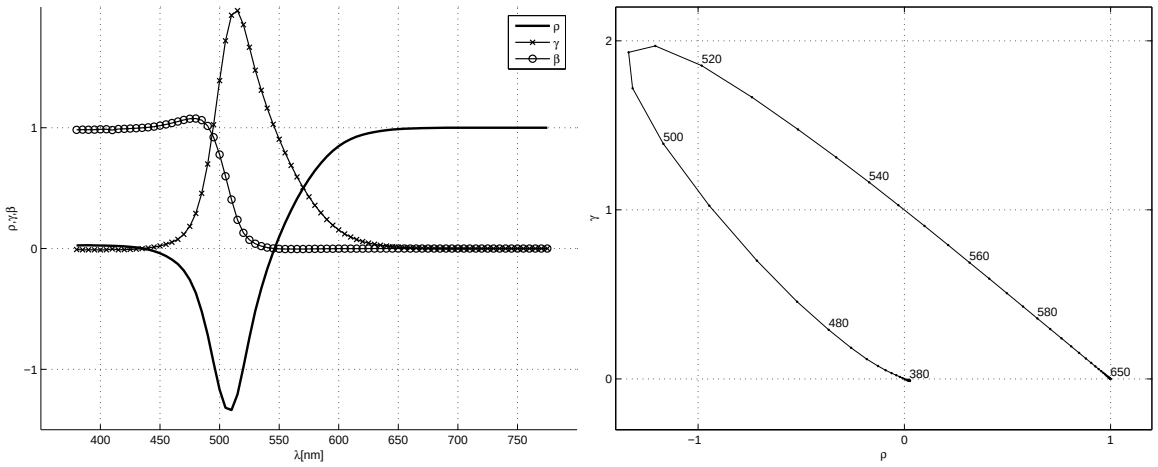
[5]Môžeme tak používať terminológiu, aká je bežná pri definovaní zmesí.

λ	r	g	b	λ	r	g	b
380	0.00003	-0.00001	0.00117	**580**	0.24526	0.1361	-0.00108
385	0.00005	-0.00002	0.00189	**585**	0.27989	0.11686	-0.00093
390	0.0001	-0.00004	0.00359	**590**	0.30928	0.09754	-0.00079
395	0.00017	-0.00007	0.00647	**595**	0.33184	0.07909	-0.00063
400	0.0003	-0.00014	0.01214	**600**	0.34429	0.06246	-0.00049
405	0.00047	-0.00022	0.01969	**605**	0.34756	0.04776	-0.00038
410	0.00084	-0.00014	0.03707	**610**	0.33971	0.03557	-0.0003
415	0.00139	-0.0007	0.06637	**615**	0.32265	0.02583	-0.00022
420	0.00211	-0.0011	0.11541	**620**	0.29708	0.01828	-0.00015
425	0.00266	-0.00143	0.18575	**625**	0.26348	0.01253	-0.00011
430	0.00218	-0.00119	0.24769	**630**	0.22677	0.00833	-0.00008
435	0.00036	-0.00021	0.29012	**635**	0.19233	0.00537	-0.00005
440	-0.00261	0.00149	0.31228	**640**	0.15968	0.00334	-0.00003
445	-0.00673	0.00379	0.3186	**645**	0.12905	0.00199	-0.00002
450	-0.01213	0.00678	0.3167	**650**	0.10167	0.00116	-0.00001
455	-0.01874	0.01046	0.31166	**655**	0.07857	0.00066	-0.00001
460	-0.02608	0.01485	0.29821	**660**	0.05932	0.00037	0
465	-0.03324	0.01977	0.27295	**665**	0.04366	0.00021	0
470	-0.03933	0.02538	0.22991	**670**	0.03149	0.00011	0
475	-0.04471	0.03183	0.18592	**675**	0.02294	0.00006	0
480	-0.04939	0.03914	0.14494	**680**	0.01687	0.00003	0
485	-0.05364	0.04713	0.10968	**685**	0.01187	0.00001	0
490	-0.05814	0.05689	0.08257	**690**	0.00819	0	0
495	-0.06414	0.06948	0.06246	**695**	0.00572	0	0
500	-0.07173	0.08536	0.04776	**700**	0.0041	0	0
505	-0.0812	0.10593	0.03688	**705**	0.00291	0	0
510	-0.08901	0.1286	0.02698	**710**	0.0021	0	0
515	-0.09356	0.15262	0.01842	**715**	0.00148	0	0
520	-0.09264	0.17468	0.01221	**720**	0.00105	0	0
525	-0.08473	0.19113	0.0083	**725**	0.00074	0	0
530	-0.07101	0.20317	0.00549	**730**	0.00052	0	0
535	-0.05316	0.21083	0.0032	**735**	0.00036	0	0
540	-0.03152	0.21466	.00146	**740**	0.00025	0	0
545	-0.00613	0.21487	0.00023	**745**	0.00017	0	0
550	0.02279	0.21178	-0.00058	**750**	0.00012	0	0
555	0.05514	0.20588	-0.00105	**755**	0.00008	0	0
560	0.0906	0.19702	-0.0013	**760**	0.00006	0	0
565	0.1284	0.18522	-0.00138	**765**	0.00004	0	0
570	0.16768	0.17087	-0.00135	**770**	0.00003	0	0
575	0.20715	0.15429	-0.00123	**775**	0.00001	0	0

Tabuľka 3.2: Numerické hodnoty farebných koeficientov r, g, b v závislosti na vlnovej dĺžke λ [nm]. Prevzaté z [24].

Z obr. 3.2 a) vidíme, že naše oko je rôzne citlivé na rôzne farby. Napr. citlivosť na modrú farbu (koeficient β) je v porovnaní s červenou (koeficient γ) veľmi malá: k modrej farbe stačí pridať len veľmi málo červenej a výsledný vnem bude fialový.

Normované koeficienty majú i ďalšiu veľmi užitočnú vlastnosť. Keďže $\rho + \gamma + \beta = 1$, na vyjadrenie každej farby spektra nám stačia len dva koeficienty, napr. ρ a γ – obr. 3.2 b). Koeficient β jednoznačne vyjadríme ako $\beta = 1 - \rho - \gamma$.



a) b)

Obr. 3.2: Normované farebné koeficienty. a) Vzťah medzi vlnovou dĺžkou a normovanými koeficientami. b) Vzájomná závislosť normovaných farebných koeficientov ρ a γ. Spektrálne farby dávajú otvorenú krivku (vyznačené body na krivke odpovedajú hodnotám z tab. 3.2).

3.5 Geometria trojuholníka

Pre ďalšie vysvetlenie si potrebujeme uvedomiť jednu veľmi dôležitú algebraickú vlastnosť úsečky: *úsečka je váženým priemerom svojich krajných bodov.*

Vskutku, vychádzajúc z parametrického tvaru úsečky, ktorá je daná vrcholami P a Q

$$X = P + t(Q - P), \qquad 0 \leq t \leq 1,$$

jednoduchou úpravou dostaneme

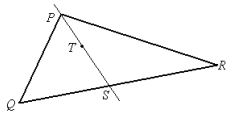
$$X = (1 - t)P + tQ, \qquad 0 \leq t \leq 1,$$

alebo trochu ináč

$$X = pP + qQ, \qquad 0 \leq p, q \leq 1, \quad p + q = 1. \tag{3.2}$$

Znamená to, že vnútorné body úsečky reprezentujú rôzne zmiešavacie pomery krajných bodov. Pritom rôznym bodom odpovedajú rôzne váhy p, q a váhy musia byť zviazané vzťahom $p + q = 1$. Stred úsečky je zároveň jej ťažiskom. Jeho súradnice sú aritmetickým priemerom súradníc krajných bodov úsečky.

Zovšeobecníme tento vzťah pre trojuholník – obr. 3.3. Uvažujme trojuholník určený trojicou jeho vrcholov P, Q, R a uvažujme vnútorný pod trojuholníka T. Bodmi P, T vedieme priamku a označíme S jej prienik so stranou QR.



Obr. 3.3: Bod trojuholníka ako vážený priemer svojich vrcholov.

Evidentne platí – viď (3.2),

$$\begin{array}{rcll} S & = & aQ + bR, & 0 \leq a, b \leq 1, \quad a + b = 1, \\ T & = & cP + dS, & 0 \leq c, d \leq 1, \quad c + d = 1. \end{array} \tag{3.3}$$

Dosadením za S v druhom vzťahu dostávame

$$T = cP + dS = cP + d\,(aQ + bR) = cP + adQ + bdR.$$

Označme $p = c$, $q = ad$, $r = bd$. Vzhľadom k (3.3) dostaneme

$$p + q + r = c + ad + bd = c + d\,(a + b) = c + d = 1,$$

čo vedie k výslednej formulácii

$$T = pP + qQ + rR. \qquad 0 \leq p, q, r \leq 1, \quad p + q + r = 1. \tag{3.4}$$

To znamená, že ku každému bodu T trojuholníka PQR vieme nájsť trojicu nezáporných váh p, q, r tak, že T *je váženým priemerom vrcholov* P, Q, R. Napr. ťažisku odpovedá prostý priemer $p = q = r = \frac{1}{3}$.

Uvedenú konštrukciu môžeme zovšeobecňovať ďalej: zoberme množinu bodov $P_0, P_1, ...P_n$ a budeme vytvárať ich všetky možné vážené priemery

$$P = p_0P_0 + p_1P_1 + ... + p_nP_n,$$

čo znamená, že od koeficientov požadujeme, aby $0 \leq p_0, p_1, \ldots, p_n \leq 1$ a zároveň $p_0 + p_1 + \ldots p_n = 1$. Tieto koeficienty nazývame *barycentrické súradnice* bodu P.

Dostávame tak *konvexný obal* danej množiny bodov $P_0, P_1, \ldots, P_n$. Ťažiskom tejto množiny bodov je bod, súradnice ktorého sú aritmetické priemery súradníc uvedených bodov, tj . $p_0 = p_1 = \cdots = p_n = \frac{1}{n+1}$.

Tento aparát použijeme pri formalizácii farebných vnemov, a stretneme sa s ním tiež v kapitole venovanej generovaniu kriviek.

Vráťme sa naspäť k trojuholníku a vytvorme navonok veľmi podobnú konštrukciu:

$$X = pP + qQ + rR, \qquad 0 \leq p, q, r \leq 1. \tag{3.5}$$

Teraz však nekladieme podmienku $p + q + r = 1$. Takáto konštrukcia má tiež veľmi názornú geometrickú interpretáciu: keď body P, Q, R interpretujeme ako bázové vektory 3D priestoru,

$$P = (1, 0, 0), \quad Q = (0, 1, 0), \quad R = (0, 0, 1),$$

potom pri meniacich sa koeficientoch p, q, r vzťah (3.5) určuje všetky body jednotkového kvádra, čiže kocky.

3.6 Chromatický diagram

Keď zoberieme dve farby z krivky na obr. 3.2 b), tak na úsečke, ktorá ich spája, dostávame všetky možné farebné zmesi týchto farieb. Napr. úsečka spájajúca oba konce krivky (červený a fialový) vyjadruje rôzne zmesi červeno-fialových, tj. purpurových farieb. Na rozdiel od farieb z krivky, pre ktoré existujú vlnové dĺžky, t.j. existujú pre ne monochromatické svetlá, *purpurové farby nemajú svoju vlnovú dĺžku – sú to vždy len zmesi monochromatických svetiel.*

Preto farby z krivky na obr. 3.2 b) nazývame spektrálne a purpurové farby nazývame nespektrálne. Všetky tieto farby, tj. spektrálne a purpurové, sa označujú ako čisté (sýte) farby a hovoríme, že poloha bodu na uzavretej krivke z obr. 3.2 b) určuje *farebný tón* analyzovaného farebného vnemu. Celú plochu, ktorá je ohraničená práve analyzovanou uzavretou krivkou, nazývame *chromatický diagram.*

Vzniká zákonite otázka: *Dá sa nejakým spôsobom v chromatickom diagrame vyjadriť biela farba?* V úvode kapitoly 2 sme uviedli, že achromatické svetlo, tj. svetlo, ktoré sa prejavuje „bielou farbou“, je zmesou všetkých monochromatických svetiel, pričom žiadne z nich neprevláda, čiže všetky sú v tejto zmesi rovnocenné. Prevedené do formálneho vyjadrenia to znamená, že achromatické svetlo je aritmetickým priemerom všetkých uvažovaných monochromatických svetiel. Preto jeho normovaný farebný koeficient ρ_a získame ako aritmetický priemer hodnôt normovaných koeficientov ρ_λ všetkých uvažovaných monochromatických svetiel s vlnovou dĺžkou λ. Podobným spôsobom dostaneme aj γ_a a β_a

$$\rho_a = \frac{1}{\|L\|} \sum_{\lambda \in L} \rho_\lambda, \quad \gamma_a = \frac{1}{\|L\|} \sum_{\lambda \in L} \gamma_\lambda, \quad \beta_a = \frac{1}{\|L\|} \sum_{\lambda \in L} \beta_\lambda.$$

Pritom L sú uvažované monochromatické svetlá a $\|L\|$ ich počet.

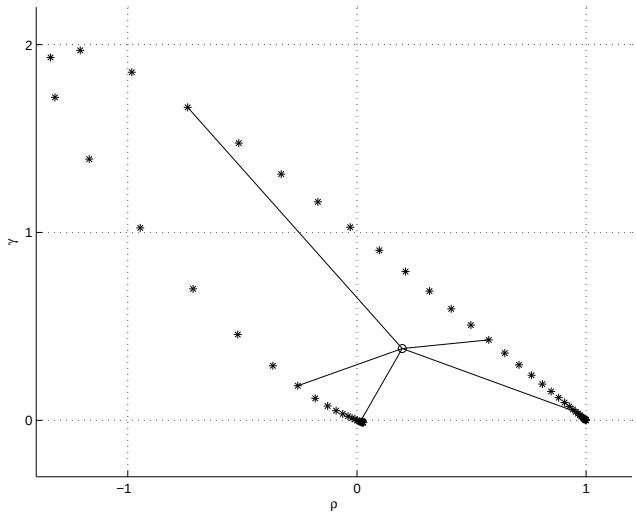
V našom prípade (tab. 3.2) $\|L\| = 80$ (vlnové dĺžky 380 –775 nm s krokom po 5 nm) dostávame hodnoty normovaných koeficientov achromatického svetla

$$\rho_a = 0,368, \quad \gamma_a = 0,301, \quad \beta_a = 0,331.$$

Keďže z geometrického (a taktiež mechanického) hľadiska je aritmetický priemer množiny bodov ťažiskom uvažovanej množiny bodov, *bielej farbe odpovedá v chromatickom diagrame ťažisko bodov, na základe ktorých sme tento diagram skonštruovali.*

Napadne nás ďalšia otázka: *keď sme našli v chromatickom diagrame bielu farbu, kde sa nachádza šedá farba?* Pri hľadaní odpovede si musíme uvedomiť, že u šedej farby, podobne ako u bielej, nie je žiadna farebná zložka prevládajúca. Takže povedané obrazne, „šedá je vo svojej podstate biela, ale nižšej intenzity“. Pritom pri konštrukcii chromatického diagramu sme sa intenzity zámerne zbavili prechodom od farebných koeficientov k normovaným farebným koeficientom. Preto *v chromatickom diagrame nie sme schopní rozlíšiť farebné vnemy líšiace sa len svojou intenzitou.* Je tak správnejšie hovoriť, že *ťažisku v chromatickom diagrame odpovedá achromatické svetlo* (achromatické farby) a nie biela farba.

A ešte jedna otázka – a*ko je to s farbou čiernou?* Celý experiment vedúci k chromatickému diagramu vychádzal z predpokladu existencie svetla. Keďže čierna farba znamená neexistenciu žiadneho svetla, principiálne *nemožno čiernu farbu v danom kontexte, tj. pomocou chromatického diagramu, vyjadriť.*



Obr. 3.4: Body určujúce chromatický diagram (*) a ťažisko (o), ktorému odpovedá biela farba. Každá z úsečiek reprezentuje farby s rovnakým farebným tónom.

Z vyššieuvedenej konštrukcie vyplýva, že na úsečke, ktorá spája ťažisko a hraničný bod chromatického diagramu (obr.3.4) dostávame farby, ktoré sa vzájomne líšia iba mierou achromatickej zložky. Majú teda jeden farebný tón.

Chromatický diagram tak dovoľuje formalizovať dva atribúty všetkých farebných vnemov prirodzeným spôsobom:

- *farebný tón* – je daný polohou bodu na hranici chromatického diagramu,
- *čistota (sýtosť)* farby – je daná polohou na spojnici ťažiska a hraničného bodu (čím bližšie k hranici, tým sýtejšia farba).

V počítačovej literatúre, sa stretneme s trochu iným tvarom chromatického diagramu, kde koeficienty sú vybrané tak, aby nadobúdali len nezáporné hodnoty. Toto sa dá dosiahnuť napr. lineárnou transformáciou, ktorá je stanovená CIE normou [24]

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 2,769 & 1,752 & 1,13 \\ 1,0 & 4,591 & 0,06 \\ 0 & 0,057 & 5,594 \end{pmatrix} \begin{pmatrix} r \\ g \\ b \end{pmatrix} \tag{3.6}$$

a následným normovaním, podobne ako v (3.1)

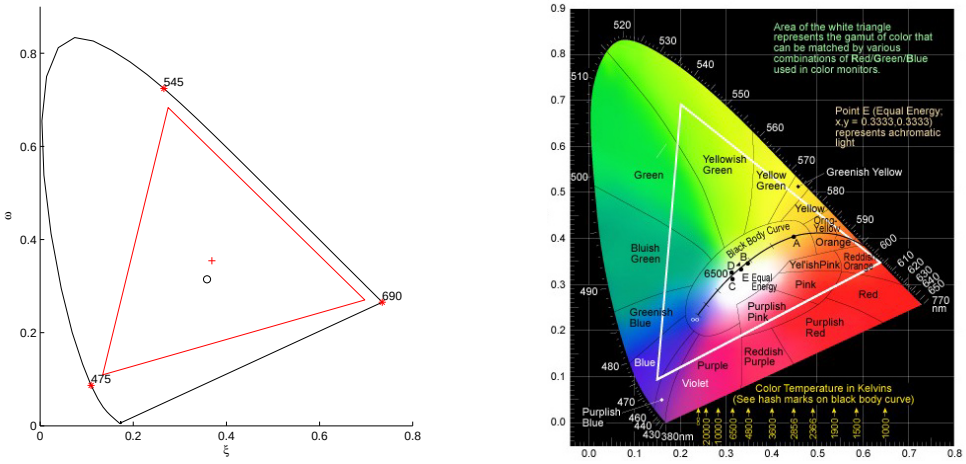
$$\xi = \frac{x}{x+y+z}, \qquad \omega = \frac{y}{x+y+z}.$$

Vyjadrenie farieb pomocou hodnôt ξ, ω môžeme nájsť v literatúre pod názvom CIE kolorimetrická sústava – obr. 3.5. Podrobné odôvodnenie toho, prečo je zvolena práve taká transformácia, nájdu záujemci napr. v [28].

3.7 Chromatický diagram a výstupné zariadenia – gamut

Monitor (a podobne i všetky ostatné výstupné zariadenia) môže vyrábať rôzne farebné odtiene len ako vážené priemery vybraných základných farieb. Vyberme tri základné farby – nech im v chromatickom diagrame odpovedajú body A, B, C. (Keďže chromatický diagram vyjadruje všetky možné farebné odtiene vyskytujúce sa v reálnom svete, body A, B, C nemôžu byť mimo chromatický diagram.) Všetky farebné odtiene, ktoré môžeme daným výstupným zariadením vyrobiť, tj. *gamut* výstupného zariadenia, sa nachádzajú vo vnútri trojuholníka ABC (*Maxwellov trojuholník*) obr. 3.5a). Kvalita farebného výstupu, tj. množstvo reprodukovateľných farebných odtieňov, závisí od voľby základných farieb. Pri ich výbere sú preto prirodzené dve podmienky. Snažíme sa, aby:

1. plocha trojuholníka ABC bola čo najväčšia,
2. ťažisko chromatického diagramu (achromatické svetlo) bolo vo vnútri trojuholníka ABC.



a) b)

Obr. 3.5: Chromatický diagram v CIE kolorimetrickej sústave.
a) Maxwellov trojuholník pre vybrané etalónové farby 475 nm, 545 nm 690 nm. Ťažisko Maxwellovho trojuholníka (+) neodpovedá ťažisku chromatického diagramu (o) – kap. 3.9.2.
b) Často používaná farebná reprezentácia chromatického diagramu.

Často sa môžeme stretnúť s metodicky nesprávnou interpretáciou chromatického diagramu: farebne sa vypĺňa celá jeho plocha, obr. 3.5b). Principiálne však toto nie je možné. Ako bolo zmienené vyššie, farebne správne sa môže vyplniť len vnútro trojuholníka tvoreného vrcholmi, ktoré odpovedajú základným farbám (pigmentom) daného zariadenia. Pritom napr. v prípade monitoru jeho vrcholy by mali mať „najviac červenú“, „najviac zelenú“ a „najviac modrú“ farbu. Farby, ktoré sú v chromatickom diagrame mimo trojuholník nie sme schopní v skutočnosti zobraziť, hoci v reálnom svete ich vnímame[6]. To znamená, že pre dané konkrétne výstupné zariadenie sú čisté farby iba tie, ktoré tvoria hranicu jeho Maxwellovho trojuholníka.

Pravdaže čitateľa nahlodá pochybnosť – z bežného života máme takú skúsenosť, že dnešné monitory, ako i ostatné výstupné zariadenia, sú schopné celkom verne zobrazovať farby. To je práve preto, že základné farby tvoria trojuholník, ktorý pokrýva veľkú časť chromatického diagramu. Naviac, geometrická vzdialenosť bodov v chromatickom diagrame nekorešponduje presne s rozdielom odpovedajúcich farebných vnemov – obr. 3.5, 3.8. Pre bežný život nám vlastne nezobraziteľné farby ani veľmi nechýbajú.[7]

[6]Na pochopenie problému je možné vykonať nasledujúci experiment z oblasti chuťových vnemov: predstavme si, že sladkú chuť budeme vyjadrovať pomocou bežne dostupného cukru, jeho rôznou koncentráciou (toto je analóg základného farebného pigmentu výstupného zariadenia). Na krabičke sacharínu sa dočítate, že je zhruba 100-krát sladší ako náš etalón – cukor. Tým síce získame predstavu, ako asi je sacharín sladký (tj. môžete to nejako, dokonca i graficky zobraziť – toto je v prípade farieb chromatický diagram), ale skutočnú jeho sladkosť môžete vnímať len jeho priamym ochutnaním. Ochutnaním cukru v žiadnom prípade nie!

[7]Použijúc príklad s chuťovými vnemami – Ako často potrebujeme k životu sladkosť sacharínu?

3.8 Farebné modely

Teraz, keď už vieme, aký formalizmus sa používa pre vyjadrenie farebnosti (farebný odtieň a sýtosť, resp. čistota farby) pozrime sa na mechanizmus, ktorý nám dovolí zahrnúť aj tretí atribút – intenzitu.

3.8.1 RGB model

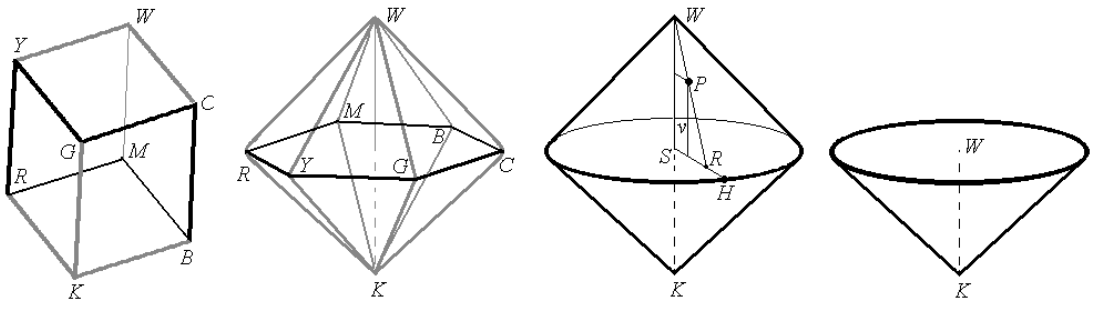
Vyššieuvedené základné farby A,B,C v (3.5) interpretujme ako bázové vektory 3D priestoru.

$$A = (1, 0, 0), \quad B = (0, 1, 0), \quad C = (0, 0, 1).$$

Všetky dosiahnuteľné farby sa dajú vyjadriť v tvare

$$X = aA + bB + cC, \qquad 0 \leq a, b, c \leq 1,$$

kde a,b,c sú intenzity základných farieb. Ako sme uviedli v kapitole 3.5, takto definovaná oblasť tvorí jednotkový kváder (3.5). Keďže za bázové farby sa vyberajú červená, zelená a modrá, namiesto všeobecného označenia A,B,C, a,b,c sa zaužívalo R,G,B, r,g,b (R=Red, G=Green, B=Blue). *RGB* model je základným modelom pre formálne spracovávanie farieb počítačom. Každý z farebných kanálov sa uvažuje ako nezávislý. Vrcholy výsledného kvádru sa zvyknú označovať ako K=blacK, R=Red, Y=Yellow, G=Green, C=Cyan, B=Blue, M=Magenta, W=White, obr. 3.6a). Na telesovej uhlopriečke KW sú odtiene šedej. Farby na uzavretej lineárne lomenej čiare $RYGCBMR$, obr. 3.6a) (všimnite si, že je priestorová) sú sýte (čisté) farby pre dané výstupné zariadenie. *Sú to práve tie farby, ktoré sú v Maxwellovom trojuholníku výstupného zariadenia na jeho hranici.*



a) b) c) d)

Obr. 3.6: Schématické porovnanie modelov a) *RGB*, b), c) *HSL*, d) *HSV*.

Intuitívne pochopiteľný mechanizmus formalizácie farieb by mal byť jednoduchým zovšeobecnením Maxwellovho trojuholníka. Keďže *RGB* model zobrazí hranicu Maxwellovo trojuholníka do *priestorovej* lomenej čiary, budeme hľadať modely, kde je tento vzťah jednoduchší.

3.8.2 HSL model[8]

Poďme „deformovať“ jednotkový kváder RGB modelu tak, že vrcholy tvoriace priestorový šesťuholník presunieme do jednej roviny a vrcholy W a K umiestnime symetricky podľa tejto roviny. Každý zo štvorcov povrchu pôvodného kvádra sa transformuje do dvojice trojuholníkov. Výsledkom je 6-boký dvojihlan, obr. 3.6b). Záverečnou „kozmetickou deformáciou“ zmeníme tento dvojihlan na dvojkužeľ, obr. 3.6c). V strede S spoločnej podstavy kužeľov je 50% odtieň šedej (tj. achromatickej) farby, preto dve farebné zložky môžeme formalizovať podobne, ako tomu bolo u chromatického diagramu (resp. v Maxwellovom trojuholníku).

Pre každý vnútorný bod P kužeľov nájdeme jeho charakteristiky nasledujúcim spôsobom. Demonštrujeme pre bod z kužeľa s vrcholom W – obr. 3.6c) Pre body z druhého kužeľa je konštrukcia podobná.

1. Nájdeme prienik R priamky WP a podstavy kužeľa.
2. Nájdeme prienik H priamky RS a hraničnej kružnice podstavy kužeľa -- bod H charakterizuje farebný odtieň (Hue).
3. Podiel dĺžiek usečiek $|RS|/|HS|$ určuje čistotu farby (Saturation).
4. Vzdialenosť bodu P od podstavy kužeľa v nesie achromatickú informáciu (Lightness).

L - lightness vyjadruje symetriu bielej a čiernej pri miešaní farebných pigmentov. Napr. z červenej vyrobíme ružovú pridaním bieleho pigmentu. Tmavočervenú naopak, pridaním čierneho pigmentu. Preto z hľadiska technológií miešania farieb je model HSL prirodzený. Z „fyzikálneho“ hľadiska je však takto definovaná achromatická zložka ťažko interpretovateľná. Nasledujúci model tento nedostatok odstraňuje.

3.8.3 HSV model

Vyššievytvorený dvojkužeľ budeme ďalej deformovať tak, že biely vrchol W „vtlačíme“ do roviny vrcholov R, Y, G, C, B, M do stredu podstavy S, obr. 3.6d). Pre farebné charakteristiky H a S môžeme použiť tú istú konštrukciu ako v prípade modelu HSL.

Achromatická charakteristika V - value je určená horizontálnym rezom tohto kužeľa. Je to vlastne *gamut zariadenia, tj. Maxwellov trojuholník, pre fixovaný stupeň intenzity žiarenia.* (Pravdaže „deformovaním“ sme zmenili trojuholník na kružnicu.) Toto je pre bežné chápanie zrozumiteľnejšie ako L - lightness v prípade HSL modelu.

[8] Používa sa i alternatívne označenie HLS.

3.8.4 Modely s jasovou zložkou

Z ďalších farebných modelov stoja za zmienku systémy, ktoré pracujú zároveň vo farebnom i čierno-bielom režime. Pre prevod farieb na adekvátnu škálu šedých odtieňov tak, aby výsledok zostal prirodzený, je používaný empirický vzťah

$$Y = 0,299r + 0,587g + 0,114b \tag{3.7}$$

Pozor, y zo vzťahu (3.6) a Y zo vzťahu (3.7) nesúvisia!

Napr. pri prenose analógového TV signálu z dôvodu spätnej kompatibility ČB príjmačov na farebný signál, nebolo priame používanie *RGB* modelu vhodné. Preto jedna z prenášaných zložiek bola totožná s uvedenou zložkou Y v (3.7). Napr. podľa normy PAL sa farebný signál prenáša v zložkách Y, U, V:

$$\begin{pmatrix} Y \\ U \\ V \end{pmatrix} = \begin{pmatrix} 0,299 & 0,587 & 0,114 \\ -0,141 & -0,289 & 0,437 \\ 0,615 & -0,515 & -0,10 \end{pmatrix} \begin{pmatrix} r \\ g \\ b \end{pmatrix} \tag{3.8}$$

Existujú i ďalšie modely pre podobné účely (napr. model *YCBCR* používa norma *SECAM*, model *YIQ* je použitý v norme *NTSC*). Spoločný majú práve prenos zložky Y.

3.9 Doplňujúce poznámky

3.9.1 Lineárne – nelineárne modely

RGB model skryte predpokladá lineárne zmiešavacie pomery – obr. 3.7a).

Vzhľadom na to, že citlivosť nášho zraku pri zmiešavaní nie je lineárna (obr. 3.2), taký spôsob transformácie nedáva vždy dobré výsledky. Napr. na obr. 3.7b) vidíme, že prechod medzi žltou a zelenou je podstatne strmší, ako prechod medzi modrou a fialovou.

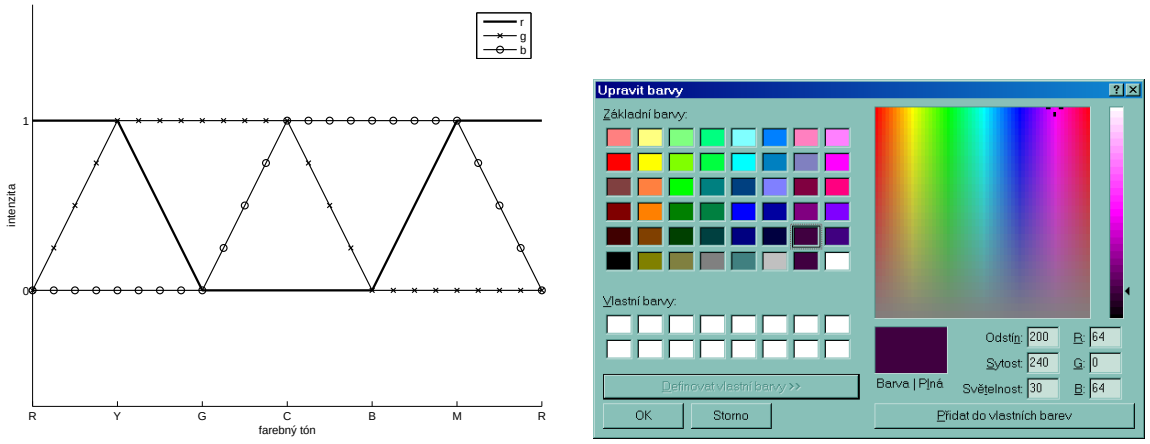
Pre vernejšie vyjadrenie farieb preto treba nelineárne modely. Nelinearitu zmiešavacích pomerov základných farieb ilustruje obr. 3.8. Dvojice bodov spojených úsečkou vnímame ako vzájomne rovnako vzdialené. Vidíme tak, že geometrická vzdialenosť bodov chromatického diagramu *CIE* nemusí zodpovedať vizuálnemu rozdielu príslušných farieb.

Preto sa v praxi namiesto *CIE* kolorimetrickej sústavy používajú i iné. Napr.

$$u = \frac{2\xi}{6\omega - \xi + 1.5}, \qquad v = \frac{4.5\omega}{6\omega - \xi + 1.5}, \tag{3.9}$$

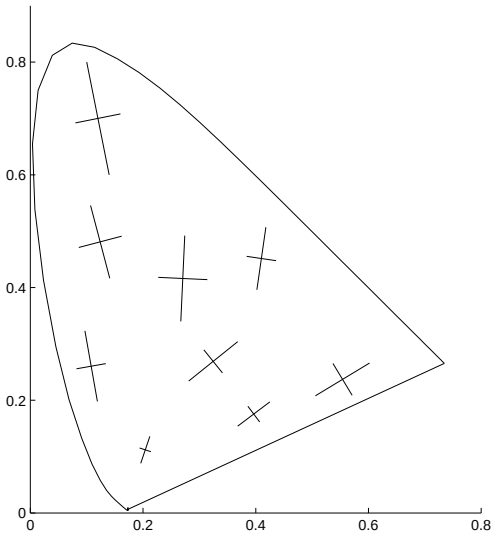
označovaná ako *CIE 1976 Luv*, rozmiestňuje farebné vnemy v chromatickom diagrame rovnomernejšie.[9]

[9] u,v z (3.9) a U, V z (3.8) spolu nesúvisia. Označenie je zvolené kvôli zaužívanej notácii.



a) b)

Obr. 3.7: a) Priebeh intenzít trojice základných farieb R, G, B na spojnici farieb R-Y-G-C-B-M-R pri použití modelu *RGB*. b) Ovládací panel pre prácu s farebným modelom *HSL* v aplikáciách MS-windows..



Obr. 3.8: Farebný rozdiel spojených dvojíc bodov vnímame ako konštantný. Prevzaté z [31].

Iný spôsob riešenia rôznej citlivosti zraku na rôzne farby spočíva v použití zložitejších spôsoboch prechodu k normovaným farebným koeficientom. Namiesto normovania prostou sumou (3.1) sú používané buď vážené priemery, prípadne zložitejšie konštrukcie (*CIEL*u*v**, *CIELAB* – [24],[29]).

3.9.2 Problém bieleho svetla

Ďalším dôvodom používania zložitejších modelov je správna reprezentácia bielej farby. Pri definovaní achromatického svetla sme vychádzali z toho, že všetky vlnové dĺžky prispievajú k jeho tvorbe rovnakým dielom. V slnečnom svetle, ktoré považujeme za achromatické, však nie sú zastúpené všetky vlnové dĺžky úplne rovnomerne. Ako bolo ukázané vyššie (obr. 3.5a)), ťažisko Maxwellovho trojuholníka a ťažisko chromatického diagramu sú rôzne body. Skutočnosťou taktiež je, že umelé zdroje svetla majú svoje farebné spektrum odlišné od slnečného. Naviac, fotosenzory vstupných zariadení (digitálny fotoaparát, scanner) a elementárne pigmenty farebných tlačiarní vykazujú v porovnaní s našim zrakom inú citlivosť na jednotlivé svetlá. Preto pre riešenie tohto problému treba uvažovať fakt, že achromatické svetlo nemusí byť presne v ťažisku chromatického diagramu, resp. že achromatickému svetlu odpovedá v chromatickom diagrame určitá oblasť, nie len jeden bod. Tento spôsob farebnej korekcie obrazu nazývame *vyváženie bielej farby.*

3.9.3 Farebné palety

Keď počet použitých farieb v obrázku nie je príliš veľký, má zmysel použité farby indexovať, čo dovolí vyjadriť farbu daného pixelu jediným číslom. Toto indexovanie nazývame *farebnou paletou.* Často sa používajú 1-bajtové palety **rxgxb** kde **r**,**g**,**b** označuje počet intenzít pre príslušné farebné kanály *RGB* modelu.

8x8x4 táto paleta sa častejšie označuje ako **3-3-2** paleta, tj. pre červenú a zelenú zložku sú vyhradené 3 bity, pre modrú len 2 bity kvôli menšej citlivosti oka na modrú farbu. Výhodou je jednoduchosť prevodovej tabuľky pre generovanie indexu farby v palete.

6x7x6 paleta obsahuje síce menší počet farieb (252), no každá zo zložiek je rovnomerne zastúpená.

7x12x3 paleta zohľadňuje rôznu citlivosť oka na základné farebné zložky.

Používajú sa taktiež adaptívne palety, ktoré sú „šité na mieru“ pre daný farebný obraz. O princípoch ich konštrukcie sa zmienime podrobnejšie v nasledujúcej kapitole.

3.10 Zhrnutie a úlohy

- Opísali sme princípy, na ktorých sú založené farebné modely v počítačovej grafike.
  - Existujú dva základné mechanizmy pre miešanie farieb – aditívny a subtraktívny.
  - Formalizmus farebného videnia popisujú Grassmannove zákony.
  - Na jednoznačné určenie farby potrebujeme tri hodnoty.
  - Chromatický diagram formalizuje dve zložky farebného vnímania ľudského oka – čistotu farby a farebný tón.
  - Farebné modely zjednodušujú mechanizmus miešania farieb.
- Veľa podrobnejších informácií nájdete v [24], [31].
- Kontrolné otázky.
  1. Čo znamená lineárna závislosť farieb (ukážte v chromatickom diagrame trojicu takých farieb, z ktorých sa nedá namiešať ľubovoľná farba).
  2. Vyznačte v Maxwellovom trojuholníku najviac fialovú farbu.
  3. Akú hodnotu y v (3.7) má žltá farba maximálnej intenzity (jasu)?
  4. Aký musí byť vzájomný pomer intenzít červenej a modrej farby, aby sa v Č/B zobrazení javili rovnako?
  5. V aplikáciách MS-windows nájdete pre prácu s farbami grafický nástroj z obr. 3.7b) prezentovaný ako HSL model. Ako by ste našli vzťah medzi ním a HSL modelom, vysvetleným v texte?
  6. Uvažujme RGB model s rozlíšením 1B na každý farebný kanál. Uvažujme rastrový obraz veľkosti $nx.ny$ pixelov. Pri akom počte farieb je pamäťovo výhodnejšie použiť paletu ako priamy zápis r,g,b hodnôt pre každý pixel?
- Samostatné úlohy:
  1. Pomocou bežne dostupných prostriedkov (napr. tabuľkový procesor Excel) transformujte hodnoty farebných koeficientov z tab. 3.1 do kolorimetrických sústav *CIE*, *CIE 1976 Luv* a pre takto získané hodnoty vytvorte chromatický diagram.
  2. Pre otestovanie faktu, že zrak vníma rôzne farby rôzne, aplikujte mechanizmus emulácie šedých odtieňov pre rôzne dvojice susedných vrcholov RGB farebného modelu.

Kapitola 4

Spracovanie rastrového obrazu

Rastrový obraz sa reprezentuje maticou, kde prvku matice zodpovedá elementárna plôška – pixel, a hodnote tohto prvku prislúcha výsledná intenzita pixelu. Toto budeme nazývať základnou reprezentáciou rastrového obrazu.

Medzi typické úlohy spracovania rastrového obrazu patria

- zmena rozlíšenia rastrového obrazu,
- zmena intenzít jednotlivých farebných zložiek,
- potlačenie nežiadúcej „schodovitosti“ použitého vzorkovania,
- potlačenie náhodných chýb - šumu.

S takými úlohami sa stretávame čoraz častejšie, keďže zariadenia, produkujúce rastrové obrazy (napr. digitálne fotoaparáty, scannery) sú čím ďalej tým viac dostupnejšie. Cieľom kapitoly je ozrejmiť, na akých princípoch sa tieto a ďalšie úlohy riešia.

Z predchádzajúcej kapitoly vieme, že na jednoznačné určenie farby treba tri hodnoty. Keď budeme uvažovať napr. model RGB, obraz môžeme rozložiť na tri monochromatické obrazy a každý farebný kanál spracovávať zvlášť. *Preto úplne stačí, keď idey metód na spracovanie obrazu vysvetlíme na achromatických obrázkoch, tj. takých, ktoré používajú len škálu odtieňov šedej.*

4.1 Digitalizácia obrazu

Primárnou úlohou pri spracovaní rastrového obrazu je jeho získanie z originálnej predlohy, tj. *digitalizácia*. Predpokladáme, že v achromatickej predlohe vieme rozlíšiť ľubovoľne blízke body a každému bodu môžeme priradiť hodnotu intenzity. Tá môže nadobúdať ľubovoľnú hodnotu z vopred stanoveného intervalu. Proces digitalizácie si môžeme predstaviť tak, že

1. zvolíme rastrovú mriežku, tj. konečný počet bodov v predlohe,
2. zvolíme konečný počet možných intenzít,
3. každému bodu rastru priradíme jednu z intenzít diskretizovaných v 2.

Vidíme, že výsledný digitalizovaný obraz obsahuje dvojakú diskretizáciu:

vzorkovanie – *priestorová diskretizácia*, tj. rozklad plochy rastrovou mriežkou,

kvantovanie – jasová diskretzácia, tj. diskretizácia oboru hodnôt obrazovej funkcie.

Rastrový obraz (obrazovú funkciu) môžeme interpretovať buď ako funkciu definovanú na množine izolovaných bodov, alebo ako po častiach konštantnú funkciu dvoch premenných, pričom oblasti konštantnej hodnoty generujeme na základe rastrovej mriežky.

Vyššie uvedená schéma nekladie žiadne nároky na rastrovú mriežku. V ďalšom texte budeme predpokladať špeciálny prípad, keď rastrovú mriežku tvoria uzly pravidelnej štvorcovej siete. Toto dovoľuje používať maticovú reprezentáciu rastrového obrazu.

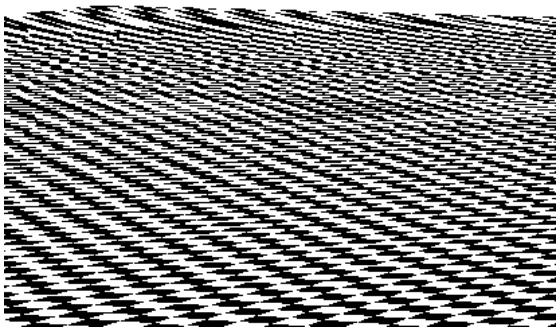
4.2 Diskretizačná chyba rastrovej reprezentácie obrazu

Diskretizácia môže niekedy viesť k veľkému nesúladu reálnej predlohy a výsledného obrázku – *diskretizačnej chybe*. Chybu diskretizácie spravidla nemožno odstrániť úplne, možno ju len čiastočne eliminovať.

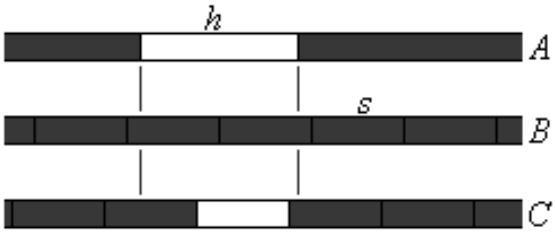
4.2.1 Vzorkovanie

Chybu vzorkovania nazývame aliasing. Prejavuje sa „schodovitosťou“ či „kostrbatosťou“ pôvodne hladkých línií. Iný typický prejav ilustruje obr. 4.1a) – tj. šachovnica v ktorej sa štvorce s nárastom vzdialenosti zmenšujú. Vidíme, že od určitej vzdialenosti sa pôvodná štruktúra obrazu úplne naruší.

Príčinu tohto javu vysvetlíme na jednoduchom 1-rozmernom príklade. Majme vzor pozostávajúci z bielej úsečky dĺžky h na čiernom pozadí - obr. 4.1 b)A. Uvažujme diskretizáciu - delenie intervalu na elementárne segmenty dĺžky s. Elementárny segment bude mať čiernu farbu práve vtedy, keď aspoň čiastočne pokrýva čiernu časť vzoru. Korektnou diskretizáciou budeme rozumieť také delenie, pri ktorom nestratíme informáciu o žiadnom detaile, tj. v našom prípade v diskretizovanom obraze bude aspoň jeden biely elementárny segment. Otázka znie: aká môže byť maximálna dĺžka elementárneho segmentu s, aby diskretizácia bola korektná? Ľahko vidno, že v prípade $s \geq h/2$ vždy môžeme zvoliť delenie tak, že biela časť vzoru bude úplne pokrytá čiernymi elementárnymi segmentami - stačí, aby elementárny segment začínal uprostred bielej časti vzoru obr. 4.1 b)B. Samozrejme, že i pre $s \geq h/2$ môžeme dostať korektný rozklad, kde sa biela časť vzoru nestratí - obr. 4.1 b)C. V



a)



b)

Obr. 4.1: a) Šachovnica v stredovom zobrazení. b) Vzorkovanie vzoru dĺžky h elementárnym segmentom dĺžky s.

našich úvahách ale hľadáme „najpesimistickejší scenár“, pretože chceme dostať korektnú diskretizáciu nezávisle na tom, ako je diskretizačná mriežka vzhľadom ku vzoru umiestená. Preto

- *pre korektnú diskretizáciu musíme zvoliť dĺžku elementárneho segmentu* $s < h/2$.

Táto podmienka je známa ako *Shannonova vzorkovacia veta.* V literatúre (napr. [29] str. 46, [7] str. 79) sa spravidla vyjadruje frekvenčným spôsobom:

- *vzorkovacia frekvencia musí byť viac ako dvakrát väčšia ako najvyššia frekvencia spracovávaného signálu.*

Frekvenčnú formuláciu ilustruje nasledujúci príklad. Uvažujme hodinový ciferník a sledujme na ňom pohyb sekundovej ručičky. Frekvencia jej rotácie je 1 otočka za minútu. Preto pri zaznamenávaní polohy ručičky raz za minútu neregistrujeme žiaden jej pohyb. Pri polminútovom intervale sledovania (tj. pri dvojnásobnej frekvencii snímania) vidíme ručičku v protiľahlých polohách, no nevieme určiť smer jej rotácie. Pri častejšom sledovaní, tj. pri vyššej frekvencii snímania (napr. raz za 21 sekúnd) sme schopni správne rekonštruovať smer rotácie ručičky, pri nižšej frekvencii snímania, (napr. raz za 33 sekúnd) ručička rotuje zdanlivo opačným smerom. *Aby sme registrovali správny smer rotácie ručičky, musíme ju snímať minimálne* raz za 29 sekúnd, tj. *s frekvenciou vyššou, ako je dvojnásobok frekvencie rotácie ručičky.*

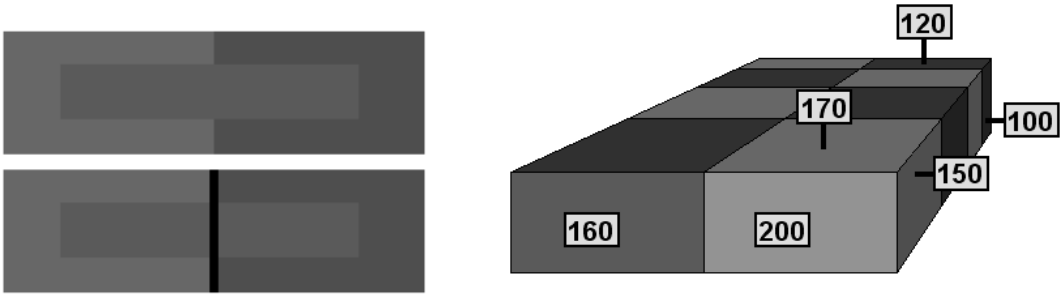
4.2.2 Kvantovanie

Ľudské oko rozlíši zhruba 100 rôznych stupňov šedých odtieňov. Veľakrát i menšia škála stačí k tomu, aby sme prechody medzi jednotlivými stupňami vnímali spojite. Preto 1-bajtová škála intenzít pre potreby počítačovej grafiky vyhovuje. Paradoxne, veľmi jemná

Obr. 4.2: Časť 64-stupňovej (hore) a 32-stupňovej (dole) škály šedých odtieňov pre rôzne veľké oblasti konštantnej farby.

škála môže niekedy viesť k nesprávnej interpretácii odtieňov. Na obr. 4.2 vidíme, že diskretizačná chyba sa v tomto prípade prejavuje vo forme *Machovych pruhov*.

Všimnime si, že pravé okraje farebne konštantných plôšok na obr. 4.2 sa zdajú byť tmavšie, ako ľavé okraje. Toto je dôsledok faktu, že *intenzitu farby danej plochy nevnímame v absolútnej hodnote, ale pomerne k intenzite okolia*[1]. Kvantovanie sa prejaví paradoxne tým výraznejšie, čím menšia je priestorová zmena (gradient) intenzít, tj. pre väčšie farebne homogénne oblasti je diskrétnosť škály výraznejšia. Lokálnosť vnímania môže viesť k vizuálnym klamom – obr. 4.3.



a) b)

Obr. 4.3: a) Obrázky sa líšia iba zvislou čiernou úsečkou. Jednotlivé plochy majú inteznitu 150, 160, 170 z 256-stupňovej škály šedých odtieňov. b) Ľavý predný štvorec je v skutočnosti svetlejší ako najprednejší bočný. Hodnoty v tabuľkách označujú intenzity príslušných štvorcov.

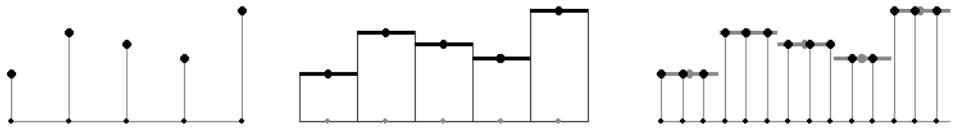
4.3 Prevzorkovanie rastrového obrazu

Dôležitou úlohou rastrovej grafiky je zmena veľkosti výsledného rastrového obrazu. Ide vlastne o prevedenie obrazu z jednej diskretizačnej mriežky na inú. Obmedzíme sa na základnú metódu – interpoláciu najbližším susedom (nearest-neighbor interpolation). Metóda ukazuje zmysel oboch reprezentácií (bodová vs. po častiach konštantná) obrazovej funkcie.

[1]Lokálnosť vnímania a spracovania obrazu v našej mysli sa netýka len kvantovania. Všimnime si, že na obr. 4.3b) tabuľku s hodnotou 120 interpretujeme ako väčšiu v porovnaní s tabuľkou 170 aj keď sú rovnako veľké, pretože ich vnímame v kontexte veľkosti šachovnicových políčok.

1. Majme obrazovú funkciu v bodovej reprezentácii.
2. Prejdeme k interpretácii po častiach spojitej funkcie.
3. Zvoľme nové body rastrovej mriežy.
4. Nové body rastrovej mriežky lokalizujeme v oblastiach pôvodnej rastrovej mriežky.
5. Novým bodom priradíme hodnoty oblastí pôvodnej rastrovej mriežky.

Metódu demonštruje 1-rozmerný príklad z obr. 4.4.



Obr. 4.4: Prevzorkovanie rastrového obrazu. a) Bodová reprezentácia rastrového obrazu. b) Po častiach konštantná reprezentácia. c) Prevzorkovanie – bodová reprezentácia.

Viac sofistikované metódy sa líšia od uvedenej v poslednom kroku, keď hodnoty v nových bodoch vypočítavame vhodne zvolenou interpoláciou [2].

Ďalej sa budeme venovať kvantizačným metódam.

4.4 Priama zmena škály intenzity

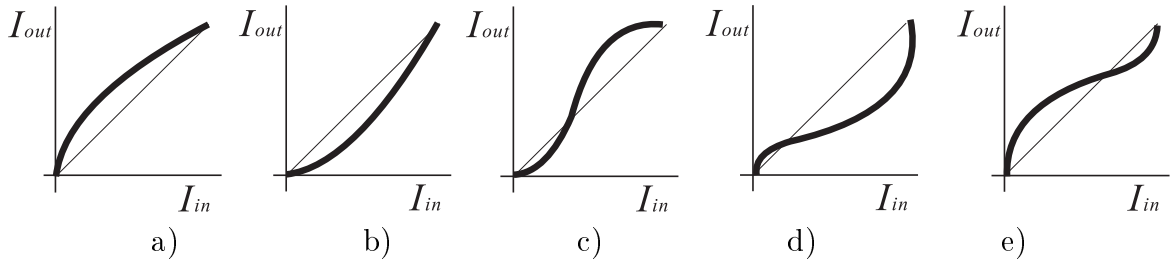
Veľmi často používanou úpravou obrazu je priama zmena oboru hodnôt obrazovej funkcie, (zmena kvantovania). To znamená, že hodnotu intenzity každého pixelu zameníme novou hodnotou podľa vopred zvoleného pravidla. Je to vlastne *bezkontextové spracovanie informácie* – výstupná hodnota pixelu závisí výlučne len na jej vstupnej hodnote a na ničom inom. Táto metóda býva v určitej forme implementovaná spravidla v každom bežnom SW pre spracovanie obrázkov. Funkcie na zmenu intenzity bývajú preddefinované, prípadne interaktívne modifikovateľné.

Pri úpravách obrazu sa často využíva režim zmeny intenzity, kombinovaný s histogramom intenzít. Histogram intenzít nám dáva informáciu o tom, ako často sa jednotlivé intenzity v obrázku vyskytujú.

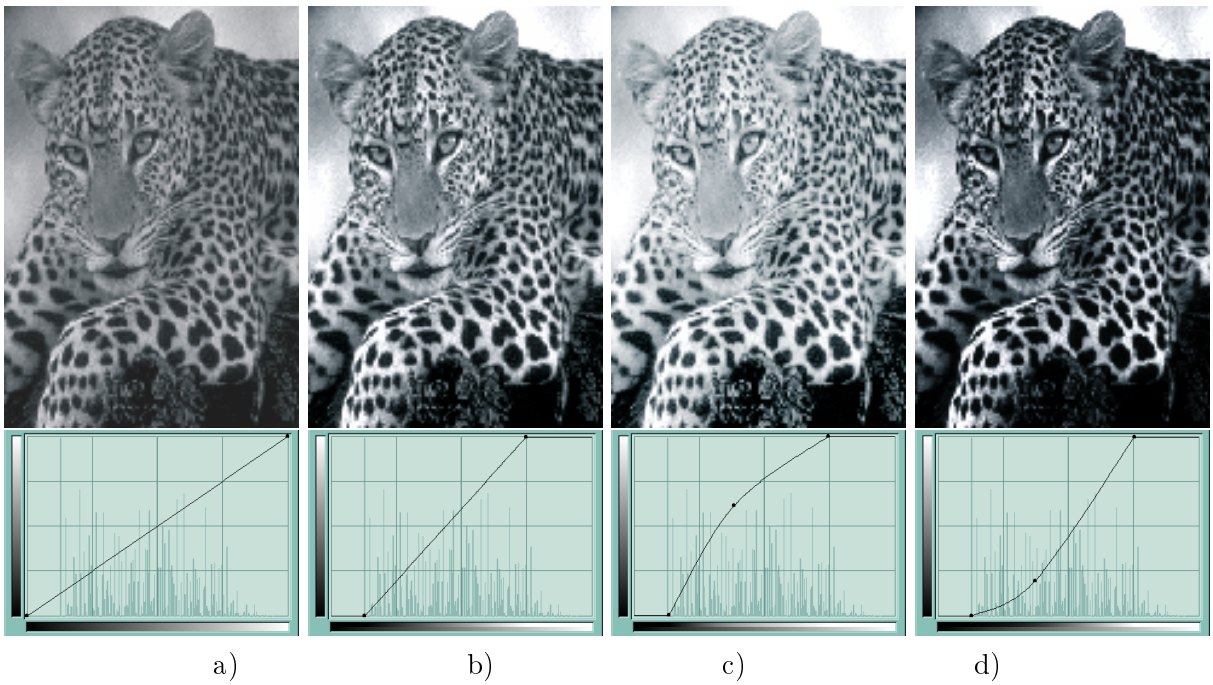
V grafoch na obr. 4.6 sa na horizontálnej osi vynáša pôvodná hodnota intenzity, na vertikálnej osi nová hodnota intenzity (krivka zmeny intenzity) a zároveň početnosť výskytu danej intenzity (histogram)[3].

[2] Interpolačnými metódami sa budeme podrobne zaoberať v kap. 10. a 11.

[3] Vidíme, že táto modifikácia už nie je bezkontextová – *výstupná hodnota pixelu je globálne závislá na vstupných hodnotách pixelov celého obrazu.*



Obr. 4.5: Príklady zmeny škály intenzity: a) zosvetlenie, b) stmavenie, c) zvýraznenie kontrastu, d) zmenšenie kontrastu a súčasne zníženie jasu, e) zmenšenie kontrastu a súčasne zvýšenie jasu.



Obr. 4.6: Zmena intenzity: a) originál, b) zvýšenie kontrastu podľa histogramu: pôvodnú škálu zredukujeme len na odtiene vyskytujúce sa v obraze, c) zosvetlenie obrazu, d) stmavenie obrazu. (Spracované pomocou GIMP 2.2.7).

K metódam priamej zmeny škály zaraďujeme i *gamma korekciu* (kap 2.6), ktorá potlačuje nežiadúci efekt linearizácie intenzity, čo je spôsobené rozdielom logaritmickej škály vnímania a lineárnej škály emulácie intenzity. V grafickom vyjadrení má gamma korekcia pre $\gamma > 1$ charakter transformácie z obr.4.5 a)¸ pre $0 < \gamma < 1$ je to transformácia typu obr.4.5b).

4.5 Automatické kvantovanie a palety

Okrem *interaktívnosti* transformácie vstupnej škály na výstupnú je niekedy vhodné tento proces *automatizovať*. Najjednoduchším príkladom je napr. *rovnomerné* hrubšie kvantovanie, tj. prechod na menší počet možných výstupných hodnôt, keď interval vstupných intenzít rozdelíme rovnomerne na požadovaný počet výstupných podintervalov.

Všeobecne možno formulovať úlohu automatického škálovania nasledujúco.

1. Škálu vstupných hodnôt rozdeľ na vzájomne dizjunktné intervaly.
2. Každému intervalu priraď výstupnú intenzitu.

Vytvorené podintervaly tvoria paletu použitých odtieňov. Dôvodom pre také manipulácie býva napr. snaha o jasové vyváženie obrazu a snaha o komprimáciu obrazu. Uvedený prístup je možné použiť nielen v 1-rozmernom prípade (achromatické, resp. monochromatické svetlo) ale i v prípade všeobecných farebných modelov.

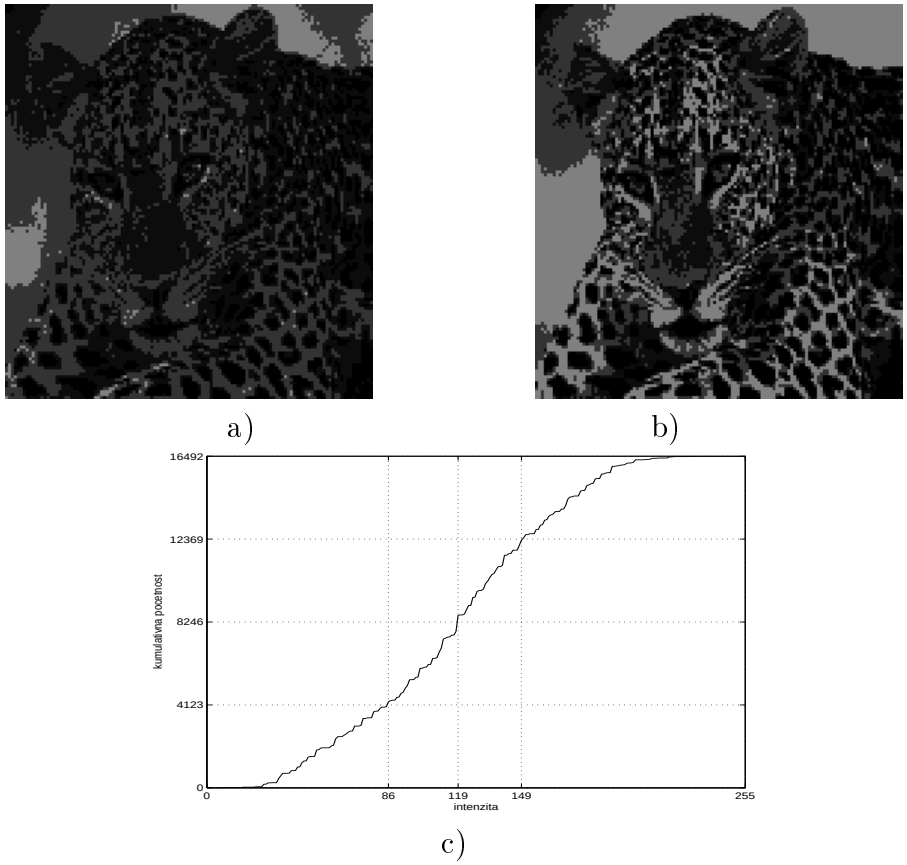
1. Pre analyzovaný obraz vyjadri všetky použité farby
   ako body uvažovaného farebného modelu.
2. Získanú bodovú množinu pokry systémom vzájomne
   disjunktných množín.
3. Každej množine priraď jednu charakteristickú farbu.

V kroku 2 uvedenej schémy sa často používajú metódy zhlukovej analýzy [19].

Obr. 4.7 demonštruje dva prístupy automatického kvantovania. V prvom prípade – obr. 4.7a) je použité rovnomerné preškálované na štyri hodnoty. V druhom prípade – obr. 4.7b) je ukázaný jednoduchý a účinný spôsob *jasového vyváženia obrazu*. Intenzity vyberáme tak, aby každá z nich bola vo výstupnom obraze zastúpená rovnako často. Algoritmus možno sformulovať nasledujúcim spôsobom:

1. Pre hodnoty vstupných intenzít skonštruuj distribučnú
   funkciu $f(x)$: na osi x sú intenzity pixelov pôvodného obrázku,
   na osi y je kumulatívny počet pixelov danej intenzity (tj. počet

pixelov ktoré majú intenzitu menšiu alebo rovnú danej intenzite).
2. Os y rozdeľ rovnomerne na požadovaný počet odtieňov.
3. Pre každú takto získanú hodnotu y_i nájdi také x_i, že $y_i = f(x_i)$.
4. Hodnoty x_i tvoria hranice intervalov hľadaných výstupných intenzít.



Obr. 4.7: a) Rovnomerné preškálovanie 256 hodnotovej palety na 4-hodnotovú. b) Adaptívna 4-hodnotová paleta. c) Kumulatívna početnosť (tj. distribučná funkcia) pôvodných vstupných intenzít s vyznačením hraníc adaptívnej palety na horizontálnej osi.

4.6 Konvolučné metódy

Konvolúcia patrí medzi veľmi dôležité nástroje, používané pri spracovaní obrazu. Ukážeme si jej použitie na 1. zmiernenie schodovitosti rastrovej reprezentácie, tj. anti-aliasing, 2. potlačenie šumu v rastrových obrazoch, 3. detekciu hranice oblastí v úlohách rozpoznávania.

Konvolúciu si môžeme predstaviť ako pohľad cez okienko, ktorým sa pozeráme na rastrovú mriežku. Pritom pixelu, ktorý je uprostred okienka, priradíme hodnotu, ktorá závisí

na hodnotách v okolí. Znamená to, že ide o *kontextové spracovanie informácie*. Príkladom je aritmetický priemer hodnôt všetkých susedov z okolia analyzovaného pixelu. Pre najjednoduchší prípad - okno veľkosti 3x3 pixely, to formálne zapisujeme takto:

$$F(x,y) = \frac{1}{9}\sum_{j=-1}^{1}\sum_{i=-1}^{1} f(x-i, y-j).$$

kde *x,y* sú súradnice pixelu, *f(x,y)* je pôvodná a *F(x,y)* výsledná hodnota *pixelu.*

Používa sa i viac „skratkovitá“ a zároveň všeobecnejšia symbolika $F = [a \otimes f]$,

$$F(x,y) \equiv [a \otimes f](x,y) = \sum_{j=-1}^{1}\sum_{i=-1}^{1} a_{ij} f(x-i, y-j),$$

kde matica a sa nazývaná konvolučné jadro (maska). Pre vyššieuvedený prípad $a = a_1$.

$$a_1 = \frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}, \qquad a_2 = \frac{1}{16}\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}, \qquad a_{2\,00} = \frac{1}{9}\begin{pmatrix} 0 & 0 & 0 \\ 0 & 4 & 2 \\ 0 & 2 & 1 \end{pmatrix}.$$

Aritmetický priemer znamená, že všetky pixely v maske považujeme za rovnocenné. V skutočnosti však niektoré pixely majú spoločnú celú hranu, niektoré len jeden vrchol. Preto miera ich vzájomnej susednosti nie je rovnaká. Pre vyjadrenie rôznej miery vplyvu okolia na hodnotu obrazovej funkcie v danom bode (x,y) je z toho tdôvodu výstižnejšie konvolučné jadro a_2, ktoré namiesto prostého priemeru používa vážený priemer hodnôt z okolia.

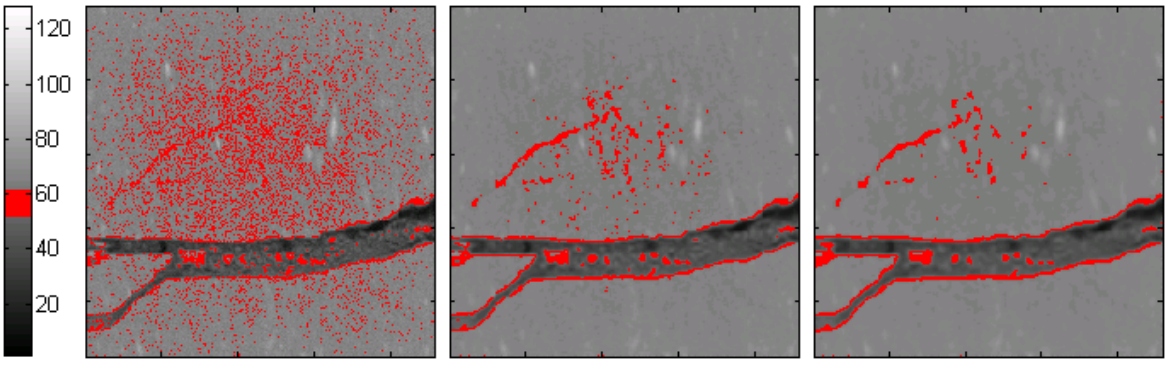
Vyššieuvedené vzťahy pre konvolúciu platia pre vnútorné pixely obrázku. Pre hraničné pixely uvažujeme len príslušné podokolia. Napr. pre ľavý horný roh a vážené priemerovanie máme jadro $a_{2\,00}$. (Taktiež možno fiktívne zväčšiť obrázok z každej strany o jeden riadok a stĺpec, a v takom obrázku aplikovať konvolúciu len pre jeho vnútorné pixely.)

S potlačením chyby vzorkovania tieto metódy zároveň potlačujú i náhodný šum v obraze. Vskutku, napr. na čiernom pozadí $f(x,y) = 0$ bude mať izolovaný biely pixel s hodnotou $f(x,y) = 9$ po aplikovaní prostého priemerovania hodnotu $F(x,y) = 1$. Je ale pravdou, že plocha rastrového obrazu izolovaného bodu sa po priemerovaní zväčší - hodnotu 1 budú mať i bezprostrední susedia pixelu (x,y).

Obr. 4.9 ukazuje potlačenie šumu v digitálnom obraze konvolučným vyhladzovaním s použitím masky a_1. Priemerovanie spôsobí, že veľké rozdiely hodnôt susedných pixelov sa zmenšujú, čím sa zmenšuje i efekt schodovitosti rastrového obrazu. V prípade, že v obraze je skutočne ostrá hranica (horizontálna, či vertikálna), konvolúcia ju rozmaže, čo pôsobí

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 1 1 1 0 0 1 1 1 0 0 0 0 0 0 0	0 1 1 1 0 0 1 1 1 0 0 0 0 0 0 0
0 0 9 0 0 0 0 9 0 0 0 0 0 0 0 0	0 1 1 1 0 1 3 3 2 0 0 0 0 0 0 0	0 1 2 1 0 1 3 4 2 0 0 0 0 0 0 0
0 0 0 0 0 0 9 9 0 0 0 0 0 0 0 0	0 1 1 1 1 3 6 6 5 2 1 0 0 0 0 0	0 1 1 1 1 3 6 7 4 2 1 0 0 0 0 0
0 0 0 0 0 9 9 9 9 9 0 0 0 0 0 0	0 0 0 1 3 6 8 8 7 4 2 0 0 0 0 0	0 0 0 1 3 6 8 8 7 5 2 0 0 0 0 0
0 0 0 0 9 9 9 9 9 9 0 0 0 0 0 0	0 0 1 3 6 8 9 9 9 7 5 2 1 0 0 0	0 0 1 3 6 8 9 9 9 7 4 2 1 0 0 0
0 0 0 9 9 9 9 9 9 9 9 9 0 0 0 0	0 1 3 6 8 9 9 9 9 8 7 4 2 0 0 0	0 1 3 6 8 9 9 9 9 8 7 5 2 0 0 0
0 0 9 9 9 9 9 9 9 9 9 9 9 0 0 0	2 3 6 8 9 9 9 9 9 9 9 7 5 3 0 0	1 3 6 8 9 9 9 9 9 9 9 7 4 2 0 0
0 9 9 9 9 9 9 9 9 9 9 9 9 9 0 0	2 5 8 9 9 9 9 9 9 9 9 8 6 5 0 0	2 5 8 9 9 9 9 9 9 9 9 8 7 6 0 0
a)	b)	c)

Obr. 4.8: a) Predloha s izolovaným pixelom (vľavo hore) a rôznou mierou schodovitosti (vľavo jednoduchá, vpravo zdvojená). b) Konvolúcia s jadrom a_1. c) Konvolúcia s jadrom a_2.



a) b) c)

Obr. 4.9: Rez CT snímku horninovej vzorky s výraznou diskontinuitou v škále 128 šedých odtieňov so zvýraznením hodnôt 50–60. a) Pôvodný obraz $f(x, y)$ b) Konvolúcia $[a_1 \otimes f](x, y)$. c) Dvojnásobná konvolúcia $[a_1 \otimes [a_1 \otimes f]](x, y)$.

vizuálne rušivo. Zníženie rozmazania skutočnej hranice sa dá zabezpečiť konvolúciou s prahom, kde prahovú hodnotu T musíme vhodne zvoliť:

$$F(x,y) = \begin{cases} [a \otimes f](x,y) & |[a \otimes f](x,y) - f(x,y)| < T \\ f(x,y) & |[a \otimes f](x,y) - f(x,y)| \geq T \end{cases}$$

Prevedené do „ľudskej reči“ to znamená, že keď sa konvolúcia pixela veľmi líši od originálu, tak predpokladáme, že to je spôsobené nie diskretizačnou chybou, ale tým, že v obraze je skutočná hranica. Preto v tomto prípade pôvodnú hodnotu intenzity ponecháme.

Používajú sa i iné typy konvolučných jadier. Napr. pri použití *mediánovej masky* sa výsledná hodnota pixelu definuje ako medián hodnôt v použitom okne.

Viac o tejto problematike čitateľ nájde napr. v [7],[?].

4.7 Detekcia hrán

V kontexte konvolučných metód stojí za zmienku i metóda, ktorá vo svojej podstate patrí do inej, veľmi dôležitej oblasti – *rozpoznávania obrazu*. S rozširovaním použitia scannerov, digitálnych kamier a digitálnych fotoaparátov, tj. rastrových vstupných zariadení, úloha rozpoznávania obrazu sa stáva veľmi častou v rôznych oblastiach. Rozpoznať v rastrovom obraze určitý objekt sme schopni práve nájdením jeho hranice.

Predpokladajme, že oblasti reprezentujúce objekt sa vyznačujú temer konštantnou hodnotou svojej farby, tj. rozdiel hodnôt susedných pixelov bude malý[4]. Naopak, rozdiel hodnot susedných pixelov na hranici rôznych oblastí bude veľký. Vidíme tak, že *rozdiel dobre indikuje hranicu oblasti*.

Pre výpočet rozdielu je vhodné uvažovať všetky možné smery. Najjednoduchšia forma rozdielových konvolučných masiek R_h, R_v, R_{d_1}, R_{d_2}, vedie k charakteristike, zvanej *Robertsov* operátor, ktorý má tvar:

$$R(x,y) = v\,|R_v \otimes f(x,y)| + h\,|R_h \otimes f(x,y)| + d_1\,|R_{d_1} \otimes f(x,y)| + d_2\,|R_{d_2} \otimes f(x,y)|\,,$$

kde $v, h, d_1, d_2 > 0$, sú váhy preferencie jednotlivých smerov pre detekciu hrany, pričom $v + h + d_1 + d_2 = 1$, a

$$R_v = \begin{pmatrix} -1 & 0 \\ 1 & 0 \end{pmatrix}, \quad R_h = \begin{pmatrix} -1 & 1 \\ 0 & 0 \end{pmatrix}, \quad R_{d_1} = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}, \quad R_{d_2} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$

[4] Zdôrazňujeme, že máme na mysli len monochromatické, resp. achromatické obrazy. V prípade farebných obrazov musíme zaviesť mieru rozdielnosti pixelov na základe farebnej rozdielnosti, tj. rozdielnosti 3D vektorov.

sú konvolučné masky pravého dolného podokolia pixelu (x,y).

Podobne funguje *Sobelov* operátor

$$S(x,y) = v\,|S_v \otimes f(x,y)| + h\,|S_h \otimes f(x,y)| + d_1\,|S_{d_1} \otimes f(x,y)| + d_2\,|S_{d_2} \otimes f(x,y)|,$$

s konvolučnými maskami

$$S_v = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}, \quad S_h = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix},$$
$$S_{d_1} = \begin{pmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{pmatrix}, \quad S_{d_2} = \begin{pmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{pmatrix}.$$

Vidíme, že v tomto prípade uvažované symetrické okolie pixelu.

V praxi sa často používajú *Laplaceove* operátory

$$L(x,y) = |h \otimes f(x,y)|,$$

s konvolučným jadrom

$$h_1 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}, \text{ resp. } h_2 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{pmatrix}.$$

Na lepšie pochopenie toho, čo vlastne uvedené Laplaceove operátory robia, zabudnime na chvíľku, že obraz je rasterizovaný (tj. uvažujeme *spojitú* funkciu intenzity) a uvažujme len jednorozmerný prípad. Funkciu $f(x)$ z obr. 4.10 môžeme interpretovať ako jeden riadok analyzovaného obrázku. Použijúc aparát diferenciálneho počtu vidíme, že hranica je tam, kde prvá derivácia nadobúda veľkú hodnotu, tj. dosahuje lokálny extrém, resp. tam, kde nenulová druhá derivácia prudko klesne k nule a je z oboch strán výrazne ohraničená nenulovými hodnotami.

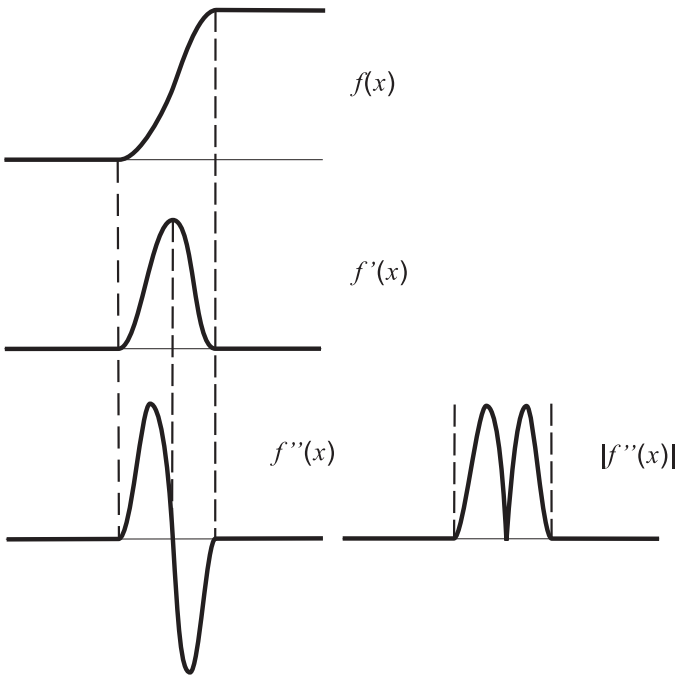
Vráťme sa teraz opäť do nášho diskrétneho sveta. Prvú deriváciu

$$f'(x) = \lim_{\Delta \to 0} \frac{f(x+\Delta) - f(x)}{\Delta}$$

v diskrétnom tvare priblížime rozdielom $f(x+1) - f(x)$, pretože minimálne možné kladné a nenulové Δ v diskrétnom prípade je $\Delta = 1$.

Druhú deriváciu

$$f''(x) = \lim_{\Delta \to 0} \frac{f'(x) - f'(x-\Delta)}{\Delta}$$



Obr. 4.10: Jednorozmerný príklad spojitého rozhrania a jeho derivácie.

nahradíme z toho istého dôvodu výrazom

$$f(x+1) - f(x) - (f(x) - f(x-1)) = f(x+1) - 2f(x) + f(x-1).$$

Z vyššie uvedeného vyplýva, že Robertsov a Sobelov operátor aproximujú 1. deriváciu obrazovej funkcie vo všetkých smeroch, Laplaceov operátor aproximuje 2. deriváciu: uvažovanie horizontálneho a vertikálneho smeru dáva konvolučné jadro h_1, keď uvažujeme aj diagonálne smery, dostávame jadro h_2.

Viac o problematike spracovania obrazu čitateľ nájde napr. v knihách [7], [10], [13].

4.8 Zhrnutie a úlohy

- Ukázali sme prejavy diskretizačnej chyby vzorkovania a kvantovania,
- demonštrovali sme metódy spracovania obrazu:
  - priama zmena škály vstupných a výstupných intenzít,
  - metódy využívajúce histogram hodnôt vstupnej škály,
  - konvolučné metódy.
- Opísali sme vplyv použitých metód na:

  - potlačenie chyby vzorkovania,
  - elimináciu náhodného šumu.

- Zmienili sme možnosť použitia konvolučných jadier prvej a druhej derivácie na detekciu hranice oblasti.

Kontrolné otázky:

1. S akou pravdepodobnosťou dostaneme korektnú diskretizáciu na obr. **??** pre $h > s > h/2$? Predpokladáme pritom, že každá možná pozícia elementárneho segmentu voči vzoru je rovnako pravdepodobná. (Použijte prístup podobný ako v prípade náhodného rozptylu.)

2. Ako bude vyzerať funkcia priamej zmeny vstupnej a výstupnej škály, ktorá realizuje inverziu obrázku?

3. V texte sa opisuje vplyv konvolučného jadra prostého priemerovania na elimináciu náhodného šumu. Ako budú na náhodný šum reagovať ďalšie zmienené konvolučné jadrá (vážené priemerovanie, konvolúcia s prahom)?

Samostatné úlohy:

1. Aplikujte konvolúciu mediánom pre okolie 3x3.

2. Porovnajte konvolúciu prostého priemerovania pre okolie 3x3 a 5x5. Ako budú vyzerať histogramy inenzít?

3. Aplikujte konvolúciu s prahom pre rôzne zvolené prahové hodnoty. Použite konvolučné jadrá a_1, a_2.

4. Realizujte metódu detekcie hrán niektorou z uvedených metód.

Kapitola 5

Kompresia rastrového obrazu

5.1 Rastrový obraz ako sekvencia symbolov

V minulej kapitole sme zaviedli základnú reprezentáciu rastrového obrazu, tj. sekvenčný zápis hodnôt jednotlivých pixelov. Pritom buď uchovávame každú farebnú zložku pre každý pixel osobitne (tj. pre každý pixel uchovávame tri číselné hodnoty), alebo vytvoríme paletu farieb, tj. každej použitej farbe priradíme jej index a pre každý pixel uchovávame len tento index. V tomto prípade treba pravdaže pre každý obraz uchovávať niekde informáciu o použitej palete. V ďalších úvahách pre jednoduchosť predpokladáme, že obraz je reprezentovaný pomocou palety, tj. jednou číselnou hodnotou pre každý pixel.

Keďže nároky na pamäť pri takej reprezentácii sú pomerne vysoké, (napr. pri použití 1 Byte na pixel potrebujeme pre obrázok s rozlíšením 1000x1000 pixelov 1 MB pamäti) je snaha o používanie úspornejších spôsobov ako je základná reprezentácia. Na to slúžia *komprimačné metódy*.

Komprimovať obrazovú informáciu môžeme podobne ako každú inú informáciu, uchovávanú v digitálnej podobe. Napr. *Huffmanovo kódovanie* využíva nerovnomernú dĺžku bitového kódu pre každý znak, pričom znaky s najväčím výskytom majú najkratší bitový kód a naopak. Dá sa pritom zostrojiť také kódovanie, že bitový kód žiadneho znaku nie je zároveň predponou bitového kódu iného znaku, čo dovoľuje jednoznačné dekódovanie i bez použitia špeciálneho oddeľovača. (Táto vlastnosť sa volá prefixnosť kódu.) Výstupný kód sa generuje postupným prideľovaním hodnôt 0,1 vstupným znakom, ktoré majú najnižšiu početnosť výskytu, a ich následným zlúčením. Proces generovania kódu možno vyjadriť binárnym stromom.

```
1. Pokiaľ existuje viac ako jeden znak, potom
  a) preusporiadaj znaky vzostupne podľa početnosti výskytu,
  b) prvé dva znaky zlúč do nového znaku,
  c) aktualizuj početnosť výskytov.
```

2. Generuj binárny strom.
   Vetvám vľavo priraď hodnotu 0, vetvám vpravo hodnotu 1.
3. Každému vstupnému znaku odpovedá kód jeho cesty v strome.

Napr. pre sekvenciu

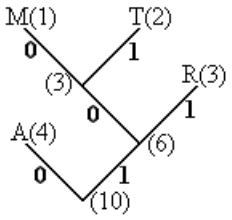
TRAMTARARA

dostávame postupne kódy (v zátvorke je počet výskytov daného znaku).

M(1)T(2)R(3)A(4) a(3)R(3)A(4) A(4)b(6) c(10)

Malé písmená značia zlúčené znaky a=MT, b=aR, c=Ab.

Výsledný ohodnotený binárny strom je na obr. 5.1.



Obr. 5.1: Binárny strom Huffmanovho kódovania.

Kódy symbolov (od koreňa k listom) sú c(A)=0, c(R)=11, c(T)=101, c(M)=100. V prípade štandardného kódovania znakov je dĺžka vstupného kódu 20 bitov (4 vstupné znakov môžeme kódovať 2-bitovým kódom, všetkých znakov v sekvencii je 10). Huffmanovo kodovanie dáva dĺžku výsledného kódu 18 bitov.

Hlavná myšlienka *kódovania LZW* (Lempel-Ziv-Welch) spočíva v tom, že každé opakovanie slova v datovej sekvencii sa nahradí identifikátorom daného slova. Pritom prvý výskyt je vo výstupnej sekvencii nekomprimovaný - vtedy mu priradzujeme identifikátor. Takto dynamicky generovaný slovník identifikátorov nie je nutné uchovávať. Slovník sa generuje podobne aj pri dekompresii. Vyššieuvedená sekvencia bude po skomprimovaní vyzerať takto:

TRAM1A22

(Použili sme nasledujúci slovník: T=1, RA=2, M=3, A=4. Opis toho, ako sa slovník generuje, viď napr. [25].) Takto sú komprimované napr. súbory typu zip.

Bližšie si opíšeme vyslovene graficky orientované komprimačné metódy. Tie môžu byť podobne, ako vyššiespomínané, *bezstratové*, čo znamená, že pri komprimácii odstraňujeme len tú časť informácie, ktorá je v kódovanej sekvencii nejakým spôsobom viacnásobne uložená. Pri spätnej rekonštrukcii dostávame dáta, identické s originálom. V grafike je možné používať (a často sa tak deje) *stratové komprimačné metódy*, ktoré uchovávajú len „podstatnú informáciu“.

5.2 Graficky orientovaná bezstratová komprimácia

Základnou myšlienkou bezstratových graficky orientovaných metód je využitie *koherencie*. Koherenciou rozumieme určitú spojitosť, tj. fakt, že pixel a jeho susedia sú väčšinou identické, alebo aspoň veľmi blízke.

5.2.1 Kódovanie dĺžky behu – RLE (Run Length Encoding)

Je to bezstratová metóda, ktorá vychádza z predpokladu, že v obrázku sú spravidla vždy väčšie plochy ako jeden pixel, ktoré sú jednej farby. Potom napr. namiesto postupnosti hodnôt pixelov

```
... 2 2 2 10 10 10 10 10 13 13 13 ...
```

je výhodnejšie použiť zápis

```
... 3 2 5 10 3 13 ...,
```

tj. obrázok sa kóduje dvojicami čísel: počet, farba.

Kompresný pomer (čiže pomer veľkostí súboru pred a po komprimácii) závisí na samotnom obrázku, ale taktiež aj na smere načítavania dát: najčastejšie sa používa načítavanie po riadkoch, resp. po stĺpcoch[1].

Avšak nie vždy musí viesť táto metóda k požadovanej komprimácii. Napr. postupnosť pixelov

```
... 2 2 2 9 10 11 10 9 11 10 10 10 ...
```

vedie ku kódu

```
... 3 2 1 9 1 10 1 11 1 10 1 9 1 11 3 10 10 10 ...,
```

ktorý je zjavne dlhší ako pôvodný kód. Taká situácia sa rieši tak, že časť kódu, ktorú RLE nevie skomprimovať, sa uchová bez zmeny, s uvedením špeciálneho príznaku a dĺžky tejto sekvencie. Vyššieuvedený príklad sa potom komprimuje takto:

```
... 3 2 0 6 9 10 11 10 9 11 3 10 ...
```

Hodnota 0 na 3. pozícii plní úlohu špeciálneho príznaku toho, že nasleduje nekomprimovaná časť. Hodnota 6 na 4. pozícii špecifikuje dĺžku tejto nekomprimovanej sekvencie.

RLE kompresia sa uplatňuje napr. pri formátoch GIF, TIFF, BMP[2].

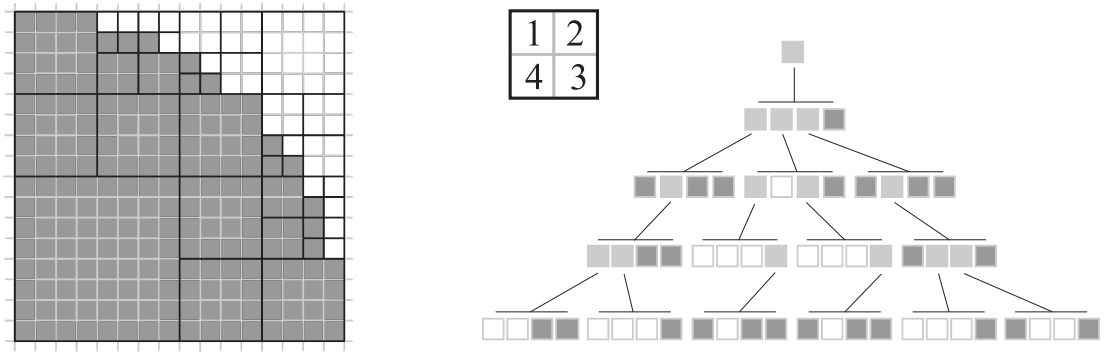
[1] Neskôr uvidíme použitie smeru „po uhlopriečke“.

[2] Formát BMP vo svojej pôvodnej podobe nepoužíva žiadnu kompresiu.

5.2.2 Kvadrantový strom

Na rozdiel od predošlej metódy, ktorá využíva lineárnu koherenciu, teraz využijeme koherenciu rovinnú. Podstatou metódy je dichotomické delenie, zovšeobecnené na prípad 2-rozmerného priestoru. V prípade dvojfarebného (čierno-bieleho) obrazu to vyjadríme nasledujúcou rekurzívnou schémou.

```
PROCEDURA quadtree(obraz)
1. { if (všetky pixely biele) zapíš 1;
2.   else if (všetky pixely čierne) zapíš 0;
3.   else
4.    { Zapíš *;
5.      Rozdeľ obraz na štvrtiny o1,o2,o3,o4;
6.      quadtree(o1);
7.      quadtree(o2);
8.      quadtree(o3);
9.      quadtree(o4);
10.   }
11.}
```



Obr. 5.2: a) Adaptívne delenie oblasti na štvrtiny. b) Kvadrantový strom s uvedeným číslovaním kvadrantov. Oblasť, ktorej odpovedá svetlošedá farba, sa delí ďalej.

Výsledkom procedúry na príklade z obr.5.2a) je nasledujúca postupnosť:

```
**0**1100*11100000**111*01001*111*01000*0*0*1110*01100000
```

Viac zrozumiteľný je ekvivalentný zápis, keď znaku * je priradená dvojica zátvoriek ( ), ktorá určuje preferencie rozkladu:

```
( (0((1100)(1110)00)00) ((111(0100))1(111(0100))0) (0(0(1110)(0110)0)00) 0)
```

V zátvorke je pritom vždy štvorica prvkov 0,1,( ).

Najviac názorná je reprezentácia v tvare kvadrantového stromu obr.5.2b). Jeho lineárny zápis (tj. postupne jednotlivé úrovne zľava doprava) má tvar:

```
zzzz00z00z1z00z00zz00111z111z0zz0110011100100010011100110
```

Znak `z` znamená, že daná úroveň sa ďalej zjemňuje (svetlošedý uzol v kvadrantovom strome).

Posledná forma zápisu dovoľuje rýchly náhľad na obraz s postupným vykresľovaním detailov: časť kódu medzi dvojicou zvislých čiar `||` dáva jednu úroveň detailu obrazu (stupeň vnorenia v kvadrantovom strome):

```
|z|zzz0|0z00 z1z0 0z00|zz00 111z 111z 0zz0|1100 1110 0100 0100 1110 0110|
```

Symboly `|` a medzera sú pomocné a v samotnom kóde nie sú nutné. (Medzera oddeľuje štvoricu delenia jednej oblasti na podoblasti, `|` znamená prechod na jemnejšie delenie.) Informácia o tom, kde sa prechádza na novú úroveň detailu vyplýva z už spracovanej časti obrázku.

5.3 Komprimácia so stratou

V praxi sa ukazuje, že napr. zníženie kvality obrazu na 75% je vo väčine prípadov nepozorovateľné. Preto pre grafické aplikácie je možné používať stratové kompresie, tj. uchovávať neúplnu informáciu o obraze. Pritom je možné dosiahnúť vysoký kompresný pomer.

5.3.1 Hrubšie kvantovanie

Zredukovaním použitej palety môžeme výrazne skomprimovať i základnú reprezentáciu. Napr. ak namiesto 256 farebnej škály použijeme iba 16 farebnú, pre kódovanie farby jedného pixelu stačia namiesto jedného byte iba štyri bity, čo skomprimuje výsledný súbor na polovičnú veľkosť.

Priamym dôsledkom redukcie farebnej palety býva fakt, že oblasti konštantnej farby sa oproti základnej reprezentácii zväčšia. Znamená to, že rozumným zredukovaním použitej palety môžeme napr. výrazne zlepšiť RLE kompresiu.

5.3.2 Hrubšie vzorkovanie

- Jednoduché varianty stratovej kompresie dostaneme priamym použitím hrubšej diskretizácie ako v pôvodnom obrázku.
- Odstránenie najnižších úrovní kvadrantového stromu vedie k stratovej kompresii.

- Metóda, ktorá je základom dnes veľmi často používaného formátu JPEG, taktiež využíva hrubšie vzorkovanie – ovšem nie priamou aplikáciou na obraz. Predtým, než si vysvetlíme, ako to funguje, musíme aspoň trochu ozrejmiť nutný matematický aparát.

5.3.3 Matematické minimum pre spektrálnu kompresiu

Predstavme si jeden riadok rastrového obrazu, kde na osi x je index príslušného pixelu a na os y vynášame hodnotu intenzity jasu daného pixelu. Dostaneme tak po častiach konštantnú funkciu.

To, o čo sa budeme snažiť, je vyjadrenie jednej funkcie pomocou hľadiska jednoduchších funkcií.

V pozadí týchto metód je prístup, známy z lineárnej algebry: každý n+1 -rozmerný vektor sa dá vyjadriť ako lineárna kombinácia bázových prvkov (vektorov)

$$\vec{x} = a_0\vec{e_0} + a_1\vec{e_1} + \cdots + a_n\vec{e_n}, \tag{5.1}$$

kde $a_0, a_1 \ldots, a_n$ sú projekcie vektora $\vec{x}$ na príslušné bázové vektory $\vec{e_0}, \vec{e_1}, \ldots, \vec{e_n}$.

Podobným spôsobom sa dá pracovať aj s funkciami. Napr. hodnoty funkcie $\sin x$ sa dajú vypočítať pomocou mocninového radu (Taylorov rozvoj):

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \ldots \ . \tag{5.2}$$

Z hľadiska lineárnej algebry to môžeme interpretovať tak, že máme bázové prvky - funkcie

$$e_0(x) = 1, \quad e_1(x) = x, \quad e_2(x) = x^2, \quad \ldots \quad e_k(x) = x^k, \quad \ldots \tag{5.3}$$

a zložky $a_0 = 0, \quad a_1 = 1, \quad a_2 = 0, \quad a_3 = -\frac{1}{3!}, \cdots$ obecne vyjadrené

$$a_{2k} = 0, \quad a_{4k+1} = \frac{1}{(4k+1)!}, \quad a_{4k+3} = -\frac{1}{(4k+3)!}. \tag{5.4}$$

Vzťah (5.2) je vlastne vyjadrením funkcie $\sin x$ pomocou bázy (5.3). Je to podobné, ako (5.1), len namiesto konečnej sumy používame sumu nekonečnú[3]:

$$f(x) = \sum_{k=0}^{\infty} a_k e_k(x) = \lim_{n\to\infty} \sum_{k=0}^{n} a_k e_k(x) \ . \tag{5.5}$$

[3]Je užitočné si uvedomiť, že takým spôsobom sa vypočítavajú elementárne funkcie (trigonometrické, exponenciálne, logaritmické) v kalkulačkách a počítačoch, pričom sa využíva vlastnosť konvergencie radu (5.5). Namiesto nekonečného radu sa spočíta pre vhodne zvolené n konečný súčet. Stačí tak pamätať si len niekoľko čísel $a_0, a_1 \ldots, a_n$ (n býva pomerne malé) a z nich aproximujeme hodnotu funkcie pre ľubovoľnú hodnotu argumentu.

Všeobecná formulácia podmienok, ktoré musia splňovať funkcie $e_k(x)$ i funkcia $f(x)$, aby limita (5.5) vôbec existovala, je netriviálna a ďaleko presahuje rámec tohto textu. Záujemci nech siahnu napr. po knihe [16]. My sa obmedzíme na konštatovanie, že pre náš prípad po častiach konštantných funkcií tento postup aplikovať môžeme.

Systém funkcií (5.3) nie je jediná možná sada bázových funkcií. Pri spracovaní a analýze signálu sa často sa používa *Fourierova kosínusová báza*

$$e_0(x) = 1, \quad e_1(x) = \cos \pi x, \quad e_2(x) = \cos 2\pi x, \ldots e_k(x) = \cos k\pi x, \ldots \tag{5.6}$$

alebo jej podobné bázy. Tieto funkcie majú charakter vĺn rôznych frekvencií, preto zápis (5.5) s bázou (5.6) sa niekedy nazýva *rozklad na spektrum* a čísla a_0, a_1, ..., a_n, ... nazývame *spektrálne koeficienty,* resp. *spektrum* funkcie.

Presná odpoveď na otázku, prečo sú vhodné práve také bázy (5.6), opäť presahuje rámec tohto textu. Predstaviť si to aspoň čiastočne môžeme, vrátiac sa ku konečnorozmerným vektorovým priestorom[4]. Keď skalárne prenásobime obe strany rovnice (5.1) bázovym vektorom $\vec{e_k}$, dostávame

$$\begin{aligned} (\vec{x}, \vec{e_k}) &= (a_0\vec{e_0} + a_1\vec{e_1} + \cdots + a_n\vec{e_n}\,,\, \vec{e_k}) \\ &= a_0\,(\vec{e_0}, \vec{e_k}) + a_1\,(\vec{e_1}, \vec{e_k}) + \cdots + a_n\,(\vec{e_n}, \vec{e_k}) \end{aligned}\,. \tag{5.7}$$

Pri úprave sme využili linearitu skalárneho súčinu.

Keď použijeme bázu vzájomne kolmých vektorov jednotkovej dĺžky (ortonormálna báza), čo znamená, že

$$(\vec{e_i}, \vec{e_j}) = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}\,, \tag{5.8}$$

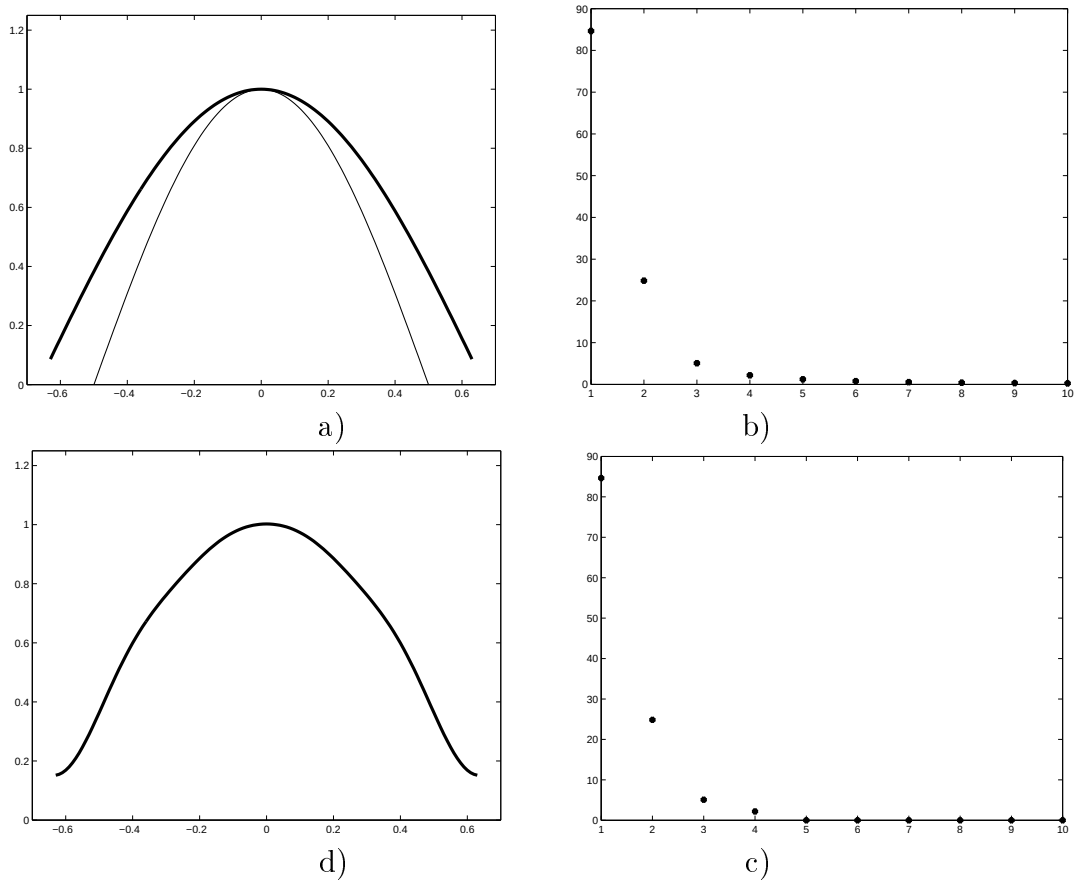
vzťah (5.7) sa zjednoduší do tvaru

$$a_k = (\vec{x}, \vec{e_k})\,. \tag{5.9}$$

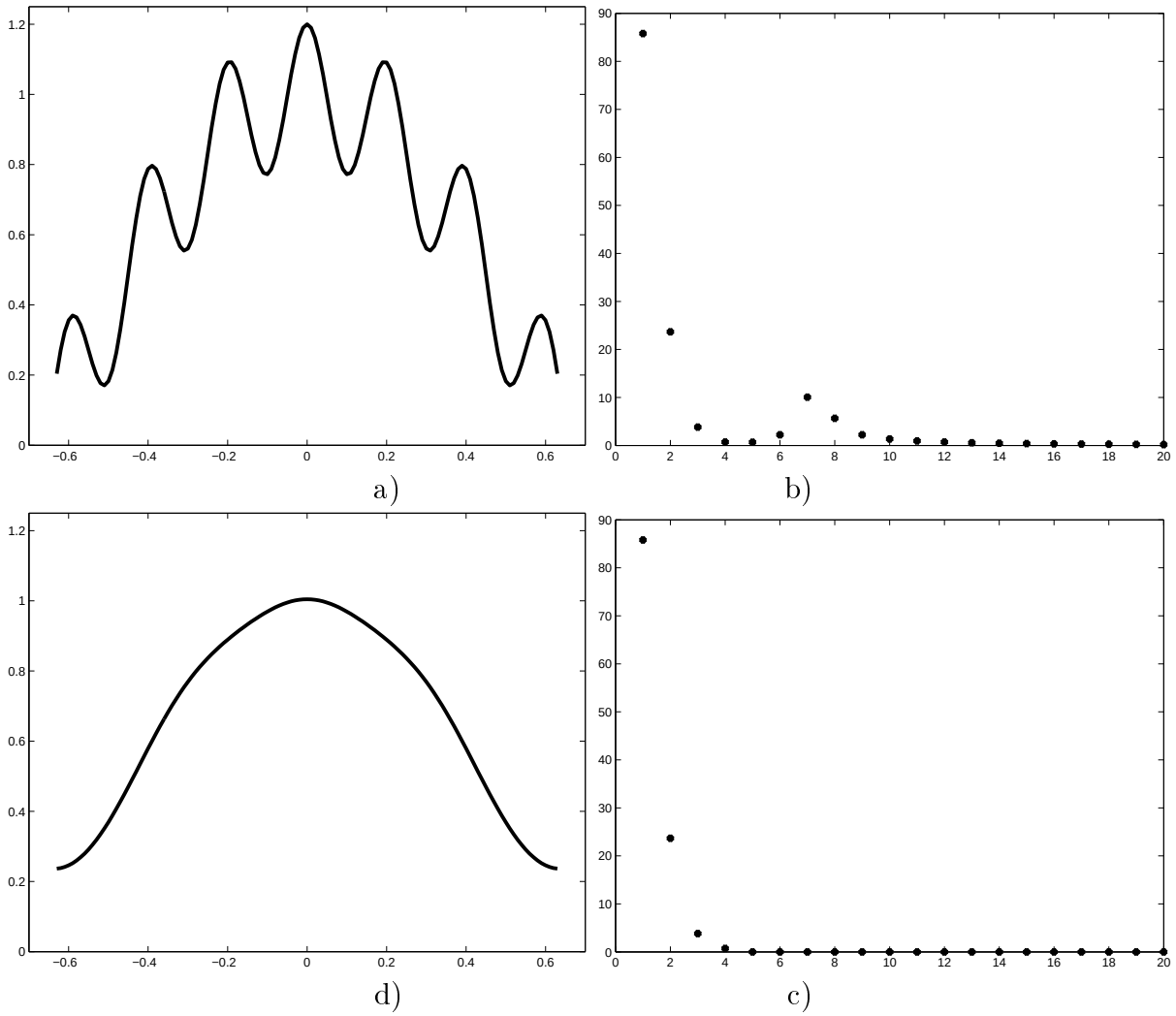
Toto je spôsob, ako nájsť projekcie vektoru do bázových vektorov. Podobne pri vhodnom zavedení skalárneho súčinu pre prípad funkcionálnych priestorov, sú bázové funkcie (5.6) vzájomne kolmé (pre bázu (5.3) to neplatí). Preto je v tomto prípade výpočet koeficientov, ako uvidíme nižšie – (5.11), jednoduchý.

Nasledujúce príklady demonštrujú vzťah medzi jednorozmernou funkciou a jej spektrom v báze (5.6). Na každom z obrázkov je a) pôvodná funkcia, b) jej spektrum, c) upravené spektrum, tj. z pôvodného spektra sú ponechané len prvé štyri frekvencie a všetky vyššie sú odstránené, d) funkcia zrekonštruovaná z upraveného spektra.

[4]Predpokladáme, že čitateľ ovláda základy lineárnej algebry, tj. vie, čo je to vektor, báza vektorového priestoru, skalárny súčin dvoch vektorov, vzájomná ortogonálnosť a ortonormálnosť dvojice vektorov.

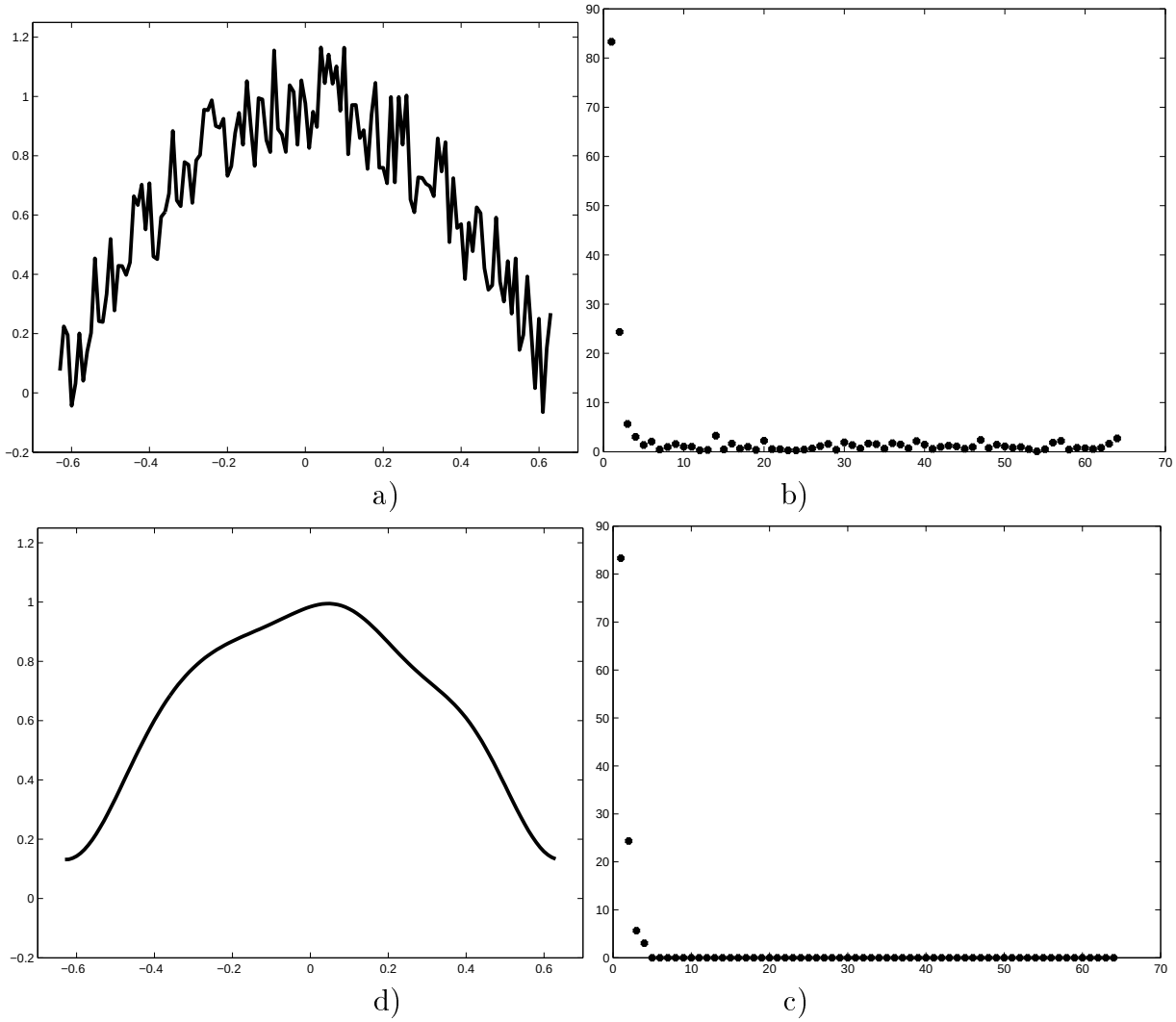


Obr. 5.3: Funkcia s jednou frekvenciou $f(x) = cos\frac{3}{4}\pi x$. (Základná bázová funkcia $e_1(x) = cos\pi x$ je vyznačena tenkou čiarou.) Vidíme, obr. (d), že na dostatočne dobré priblíženie pôvodnej funkcie $f(x)$ postačujú štyri koeficienty obr. (c).



Obr. 5.4: Superpozícia funkcie z obr. 5.3 a funkcie s vyššou frekvenciou – $f(x) = cos\frac{3}{4}\pi x + cos 10\pi x$ a). Vidíme, že v spektre funkcie sa vyššie frekvencie prejavili väčšími hodnotami koeficientov $a_5 - a_9$ – obr. b). I v tomto prípade z prvých štyroch koeficientov – obr. c) celkom verne zrekonštruujeme – obr. d) pôvodnú funkciu z obr. 5.3 a) .

Z obr. 5.4 sa dá predpokladať, že odstránením vysokých frekvencií v spektre funkcie by sa mal dať odstrániť prípadný šum. Na vyjadrenie izolovaného pixla (čo je spravidla nežiadúci šum) potrebujeme totiž bázové funkcie vysokej frekvencie. Preto ignorovaním vysokých frekvencií šum eliminujeme (ale bohužiaľ môžeme stratiť i jemné detaily, ktoré sú „za šum považované“). Že tomu tak skutočne je, ukazuje nasledujúci obrázok.



Obr. 5.5: Príklad funkcie z obr. 5.3 zaťaženej šumom. Pre rekonštrukciu – obr. (d) sú opäť použité iba šyri koeficienty – obr. (c).

Demonštrovaný spôsob práce so spektrom funkcie ukazuje, že tento mechanizmus dovoľuje obrazovú informáciu veľmi efektívne komprimovať. Namiesto zápisu obrazovej funkcie bod po bode, vyjadrujeme ju približne, pomocou konečného počtu bázových funkcií. Tj. stačí si pamätať len niekoľko koeficientov $a_0, a_1, \ldots, a_n$ a miera kompresie je daná práve ich počtom n. Pritom s rastúcim n sme schopní zachytiť jemnejšie detaily.

5.4 Algoritmus formátu JPEG

Medzi najvýznamnejšie dnes používané stratové metódy patrí metóda štandardu ISO/IEC IS 10918-1 navrhnutá skupinou JPEG (Joint Photographic Expert Group). Hlavná myšlienka spočíva vo

- využití vyššieopísanej diskrétnej Fourierovej transformácie s kosínusovou bázou (dvojrozmerný variant),
- stratovej úprave spektra,
- bezstratovej komprimácii upraveného spektra.

Obraz sa nespracováva celý naraz, ale sa rozdelí na štvorce 8x8 pixelov a nezávisle na každom z nich sa spektrum analyzuje.

Demonštrujeme to na jednoduchom jednorozmernom príklade, tj. na 8 hodnotách funkcie $f(x)$, $x = 0, 1, \ldots, 7$. Uvažujme funkcie vychádzajúce z funkcií (5.6)

$$e_0(x) = \frac{1}{2\sqrt{2}}, \quad e_k(x) = \frac{1}{2}\cos\frac{2x+1}{16}\pi k \quad k = 1, \ldots 7 \tag{5.10}$$

Hodnoty týchto funkcií sú nasledujúce:

x	0	1	2	3	4	5	6	7
$e_0(x)$	0.3536	0.3536	0.3536	0.3536	0.3536	0.3536	0.3536	0.3536
$e_1(x)$	0.4904	0.4157	0.2778	0.0975	-0.0975	-0.2778	-0.4157	-0.4904
$e_2(x)$	0.4619	0.1913	-0.1913	-0.4619	-0.4619	-0.1913	0.1913	0.4619
$e_3(x)$	0.4157	-0.0975	-0.4904	-0.2778	0.2778	0.4904	0.0975	-0.4157
$e_4(x)$	0.3536	-0.3536	-0.3536	0.3536	0.3536	-0.3536	-0.3536	0.3536
$e_5(x)$	0.2778	-0.4904	0.0975	0.4157	-0.4157	-0.0975	0.4904	-0.2778
$e_6(x)$	0.1913	-0.4619	0.4619	-0.1913	-0.1913	0.4619	-0.4619	0.1913
$e_7(x)$	0.0975	-0.2778	0.4157	-0.4904	0.4904	-0.4157	0.2278	-0.0975

Všimnime si, že funkcie (5.10) môžeme interpretovať ako vektory v 8-rozmernom vektorovom priestore[5]. Je jednoduché overiť, že tieto vektory sú vzájomne ortonormálne.

Preto (viď (5.9)) pre výpočet spektra (dopredná Fourierova transformácia) platia vzťahy

$$a_k = \sum_{x=0}^{7} f(x)e_k(x) \qquad k = 0, \ldots, 7 \tag{5.11}$$

Pre spätnú rekonštrukciu funkcie (spätná Fourierova transformácia) nás bude zaujímať postupnosť čiastočných súčtov

$$S_0(x) = a_0e_0(x), \quad S_1(x) = a_0e_0(x) + a_1e_1(x), \ \ldots, \ S_7(x) = \sum_{k=0}^{7} a_ke_k(x) \quad x \in \{0, \ldots, 7\}\,. \tag{5.12}$$

Napríklad pre funkciu

[5] Vďaka tomu, že obrazová funkcia je po častiach konštantná, každú takú funkciu môžeme interpretovať ako vektor v 8-rozmernom priestore $f = (f_0, f_1, \cdots, f_7)$ kde $f_i = f(i)$.

x	0	1	2	3	4	5	6	7
$f(x)$	0	10	20	30	20	10	0	0

dostávame spektrum

k	0	1	2	3	4	5	6	7
a_k	31,82	7,911	-26,924	-8,657	3,536	0,229	-0,328	-3,524

a čiastočné súčty (5.12) sú nasledujúce

x	0	1	2	3	4	5	6	7
$S_0(x)$	11	11	11	11	11	11	11	11
$S_1(x)$	15	15	13	12	10	9	8	7
$S_2(x)$	3	9	19	24	23	14	3	-4
$S_3(x)$	0	10	23	27	21	10	2	0
$S_4(x)$	0	9	22	28	22	9	1	0
$S_5(x)$	0	9	22	28	22	9	1	0
$S_6(x)$	0	9	21	28	22	9	1	0
$S_7(x)$	0	10	20	30	20	10	0	0

Vidíme, že $S_7(x)$ presne rekonštruuje pôvodnú funkciu, a vidíme taktiež, že už čiastočná suma $S_3(x)$ (na jej generovanie potrebujeme vedieť iba štyri koeficienty a_0, a_1, a_2, a_3) dobre (s maximálnou odchýlkou 3) približuje pôvodnú funkciu.

Dvojrozmerný variant funguje podobne: obraz sa rozdelí na podobrazy 8x8 pixelov a na každom z nich sa realizuje 2D diskrétna kosínusová transformácia s bázou (5.10). Výsledkom sú matice spektier $A = (a_{ij})$, $i, j = 0, \ldots, 7$ veľkosti 8x8, kde

$$a_{ij} = \sum_{x=0}^{7} \sum_{y=0}^{7} f(x,y) e_i(x) e_j(y) \tag{5.13}$$

a $f(x,y)$ je dvojrozmerná obrazová funkcia, tj. hodnote pixelu so súradnicami (x,y). Podľa požadovanej kvality sa každé spektrum A upravuje s pomocou vopred definovanej matice $H = (h_{ij})$ na výsledné celočíselné spektrum $B = (b_{ij})$, $i, j = 0, \ldots, 7$, kde

$$b_{ij} = int(\frac{a_{ij}}{h_{ij}}). \tag{5.14}$$

Matica H je jedna pre celý obraz. Napr. podľa [29] pre 75% kvalitu obrazu je matica

a)

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	20	31	31	31	31	10	0
0	31	63	63	63	63	10	0
0	31	63	63	63	21	0	0
0	31	63	63	21	0	0	0
0	31	63	21	0	0	0	0
0	10	10	0	0	0	0	0

b)

125.1	39.4	-84.3	-41.7	-20.4	-7.8	9.9	2.1
-20.9	-26.9	8.8	33.9	10.4	-1.6	-1.9	-5.1
-98.6	-13.4	78.2	10.4	-0.6	7.3	-5.4	1.7
0.1	19.5	-4.8	-29.3	9.0	7.9	-5.1	4.1
13.4	-10.9	-16.7	19.4	7.9	-10.1	1.1	1.0
24.8	4.4	-9.3	-4.5	-19.0	-0.0	11.8	-2.9
0.7	-2.2	-2.2	0.6	6.1	5.1	-7.2	-6.4
-1.9	3.5	2.8	-2.5	-0.4	-5.6	-2.5	9.2

c)

8	4	-8	-3	-1	0	0	0
-2	-2	1	2	0	0	0	0
-7	-1	5	0	0	0	0	0
0	1	0	-1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

d)

128	44	-80	-48	-24	0	0	0
-24	-24	14	38	0	0	0	0
-98	-13	80	0	0	0	0	0
0	17	0	-29	0	0	0	0
18	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

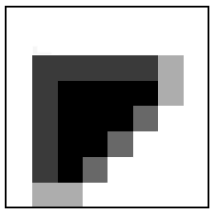
e)

3	9	7	-5	-12	-5	3	7
-4	1	3	0	0	4	1	-7
2	15	26	32	36	32	14	-6
9	34	59	66	61	46	21	0
0	37	69	70	49	25	7	-3
-3	34	64	56	26	4	-1	-1
0	28	46	33	8	-2	0	6
0	16	22	8	-7	-8	0	5

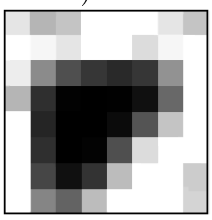
f)

3	9	7	-5	-12	-5	3	7
-4	1	3	0	0	4	1	-7
2	-5	-5	1	5	1	4	-6
9	3	-4	3	-2	-17	11	0
0	6	6	7	-14	4	7	-3
-3	3	1	-7	5	4	-1	-1
0	-3	-17	12	8	-2	0	6
0	6	12	8	-7	-8	0	5

g)



h)



Obr. 5.6: Príklad kompresie JPEG. a) Pôvodný obrázok $f(x, y)$. b) Jeho spektrum A. c) Upravené spektrum B. Je použitá matica 5.15 – tu nastáva strata informácie. Matica B sa ďalej bezstratovo komprimuje (napr. RLE metódou v smere vedľajšej diagonály). Práve takto skomprimovaná matica B sa uchováva. d) Rekonštruované spektrum $A' = (a'_{ij})$, kde $a'_{ij} = h_{ij}b_{ij}$. e) Rekonštruovaný obraz. f) Rozdiel pôvodného a rekonštruovaného obrazu. g) Pôvodný obraz – graficky. h) Rekonštruovaný obraz – graficky (záporné hodnoty sú nahradené nulou, hodnoty väčšie ako 63 sú nahradené hodnotou 63).

$$H = \begin{pmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 61 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{pmatrix}. \tag{5.15}$$

Takto upravené spektrum B obsahuje nenulové členy len v okolí nízkych frekvencií (okolo ľavého horného prvku), preto pre jeho uchovanie sa dá dobre využiť napr. RLE kódovanie so smerom kódovania po uhlopriečke.

Spätná transformácia sa opäť realizuje podobne ako 1D prípad:

$$f(x,y) = \sum_{i=0}^{7} \sum_{j=0}^{7} a_{ij} e_i(x) e_j(y) \tag{5.16}$$

Na obr. 5.6 je ukážka postupnej komprimácie a dekomprimácie obrazu.

5.5 Zovšeobecňujúca poznámka – wavelety

Ukážeme si, čo má spoločné metóda kvadrantového stromu a metóda, založená na Fourierovej transformácii.

Uvažujme, podobne ako v podkapitole 5.3.3, pre jednoduchosť len 1D prípad, tj. jeden riadok rastrového obrazu. Metóda kvadrantového stromu v tomto prípade degeneruje na dichotomické delenie, pomocou ktorého vyjadríme intervaly konštantnej hodnoty. Binárny strom dichotomického delenia môžeme interpretovať aj ako systém funkcií – viď obr. 5.7.

Je dôležité si uvedomiť, že všetky funkcie sú si v určitej miere podobné. Keď zvolíme funkciu[6]

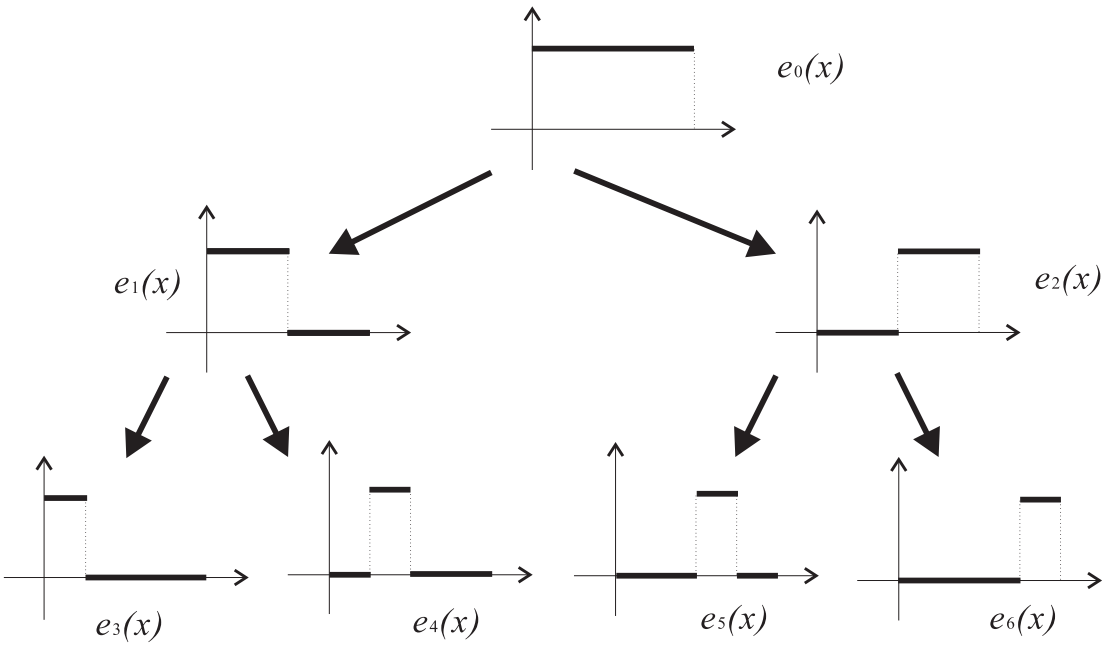
$$\phi(x) = \begin{cases} 1 & x \in \langle x_{min}, x_{max} \rangle \\ 0 & x \notin \langle x_{min}, x_{max} \rangle \end{cases},$$

všetky funkcie $e_k(x)$ sa dajú vyjadriť ako

$$e_k(x) = \phi(a_k x - b_k) \tag{5.17}$$

pre vhodne zvolené koeficienty a_k, b_k. Koeficient a_k definuje šírku intervalu (*frekvenciu*), na ktorom je funkcia nenulová, koeficient b_k zas jeho *posun* na osi x vzhľadom k hodnote 0.

[6] Takáto funkcia sa niekedy nazýva nosič intervalu $\langle x_{min}, x_{max} \rangle$. Hovoríme, že funkcia ϕ generuje interval $\langle x_{min}, x_{max} \rangle$.



Obr. 5.7: Dichotomické delenie, reprezentované systémom funkcií.

Z obr. 5.7 vidíme, že systém funkcií, resp. systém koeficientov a, b je dichotomickým delením jednoznačne určený. Napr. pre jednotkový interval, tj. $x_{min} = 0$, $x_{max} = 1$ dostaneme pre funkcie $e_0(x)$, ... $e_6(x)$ nasledujúce hodnoty koeficientov:

i	0	1	2	3	4	5	6
a_i	1	2	2	4	4	4	4
b_i	0	0	1	0	1	2	3

Pre každú hodnotu obrazovej funkcie c vyjadríme jej definičný obor pomocou príslušných funkcií $e_k(x)$, $k \in I_c$. Povedané trochu ináč (a možno zrozumitelnejšie) I_c sú indexy takých bázových funkcií, pre ktoré obrazová funkcia dosahuje hodnotu c. Pritom uvažujeme len také funkcie, že pre ľubovoľnú dvojicu platí $e_i(x)\, e_j(x) = 0$. Napr. pre piaty riadok zhora na obr. 5.2 dostávame vyjadrenie

$$f(x) = c_0 e_1(x) + c_0 e_5(x) + c_1 e_6(x),$$

kde c_0, c_1 sú hodnoty tmavošedej a bielej farby. Máme tak opäť tvar identický s (5.1) resp. (5.5). Naviac, použité bázové funkcie sú podobne ako v prípade (5.10) vzájomne ortonormálne.

Keď sa pozrieme na (5.6) v kontexte schémy (5.17) vidíme, že táto báza je generovaná z funkcie $\phi(x) = \cos \pi x$ len zmenou koeficientu a_k

$$e_k(x) = \phi(kx) = \cos k\pi x.$$

Z tohto pohľadu *je generovanie bázových funkcií dichotomického delenia* (5.17) *všeobecnejšie ako generovanie bázových funkcií Fourierovho rozkladu.* (V oboch prípadoch bázové funkcie vznikajú z jednej vopred zvolenej funkcie na základe určitej „samopodobnosti“[7]. V prípade Fourierovej bázy sa mení len frekvencia východziej nosnej funkcie, v prípade dichotomicej bázy sa okrem meniacej sa frekvencie uvažuje i meniaci sa posun). Vyššieopísaná schéma tvorby bázových funkcií na základe rôzneho posunu a rôznej frekvencie (5.17) sa používa u *waveletovej transformácie.*

5.6 Zhrnutie a úlohy

Opísali sme princípy komprimačných metód.

- Bezstratová kompresia uchováva celú vstupnú informáciu. Komprimácia sa dosahuje tým, že sa odstráni viacnásobné uchovanie tej istej informácie.
- Stratová kompresia uchováva neúplnú informáciu.
- Často používaný mechanizmus v stratovej komprimácii spočíva v tom, že informácia sa neodstraňuje priamo z obrazu, ale vytvorí sa charakteristika zvaná spektrum (Fourierova transformácia). Menej významná informácia sa odstraňuje práve zo spektra.
- Okrem kompresie tento prístup dovoľuje veľmi účinne eliminovať šum.

Kontrolné otázky:

1. Kedy je pamäťovo výhodnejšie používať paletu farieb a kedy priame kódovanie farby pre každý pixel?
2. Aký je rozdiel medzi stratovou a bezstratovou kompresiou?
3. Aký význam majú vysoké frekvencie v spektre obrázku?
4. Kde nastáva stratovosť pri kompresii JPEG?

Samostatné úlohy:

1. Realizujte RLE kompresiu s rôznymi smerami postupu v obraze.
2. Realizujte stratovú kompresiu na základe Fourierovej transformácie s využitím vzťahov (5.10), (5.13), (5.16) pre rastrový obrázok 8x8 pixelov.

[7] V pomerne nedávnej minulosti ca často hovorilo o tzv. fraktálnej kompresii, založenej na samopodobnosti. Je zarážajúce, že v tomto kontexte nikdy neboli zmieňované Fourierovské a waveletovské prístupy.

Kapitola 6

Rýchle algoritmy rasterizácie kriviek

Po zvládnutí problémov spracovania rastrového obrazu prejdeme teraz k problematike prevodu vektorovej grafickej informácie na rastrovú. Nutnosť toho kroku je daná tým, že základné výstupné zariadenia sú rastrového typu, preto aj pri práci s vektorovou grafikou v každom prípade musíme proces rasterizácie podstúpiť.

Úsečka je asi najčastejšie sa vyskytujúci objekt v počítačovej grafike. Je tomu tak preto, že postupnosťou úsečiek

$$(P_0, P_1), (P_1, P_2), \ldots, (P_{n-1}, P_n)$$

kde $P_i,\ i = 0, \cdots, n$ sú body, tj. lineárne lomenou čiarou, môžeme pri vhodnom výbere bodov dostatočne dobre aproximovať ľubovoľnú spojitú krivku.

Úsečka sa dá interpretovať nielen ako geometrický objekt, ale aj ako najjednoduchšií spojitý spôsob prenosu hodnôt z izolovaných uzlov do ich okolia. V kap. 3.5 sme sa stretli s ďalšou interpretáciou úsečky, tj. ako s množinou všetkých vážených priemerov dvojice krajných bodov.

Vzhľadom na vyššie uvedenú dôležitosť úsečky, ukážeme tento jednoduchý geometrický objekt i z pohľadu informatického, tj. z pohľadu efektivity algoritmickej realizácie.

Pri riešení otázky efektivity si musíme uvedomiť, ktoré operácie sú „lacné“ a ktoré nie. Zložitosť algoritmu budeme merať počtom operácií, pritom predpokladáme, že rôzne operácie sú rôzne náročné, napr.

1. operácie v celočíselnej aritmetike sú jednoduchšie ako operácie v reálnej aritmetike,
2. z aritmetických operácií je najnáročnejšie delenie,
3. konverzia medzi reálnym a celočíseým formátom je časovo náročná,
4. operácia porovnania v celočíselnej aritmetike je jednoduchá.

Pri rastrovej reprezentácii potrebujeme nájsť tie pixely rastrovej mriežky, ktoré sú najbližšie skutočným bodom zobrazovanej úsečky. Vychádzame z predpokladu, že úsečka je daná svojimi krajnými bodmi $P_Z = (x_Z, y_Z)$, $P_K = (x_K, y_K)$ a že *hodnoty súradníc sú celočíselné*. Použijeme vyjadrenie úsečky v tvare $y = ax + b$, kde

$$a = \frac{y_K - y_Z}{x_K - x_Z} = \frac{p}{r} \quad b = y_Z - ax_Z = \frac{x_K y_Z - x_Z y_K}{x_K - x_Z} = \frac{q}{r}, \quad x_Z \leq x \leq x_K \tag{6.1}$$

tj. p, q, r sú celé čísla. Je dostatočné obmedziť sa na prípady

$$x_Z < x_K,\ 0 < |a| < 1 \tag{6.2}$$

pretože

1. singulárne prípady, $x_Z = x_K$, tj. horizontálna úsečka, resp. $y_Z = y_K$, tj. vertikálna úsečka, resp. $|y_K - y_Z| = |x_K - x_Z|$, tj. úsečka so sklonom 45°, sa riešia triviálne,
2. prípad $x_Z > x_K$ prevedieme na požadovaný prehodením krajných bodov,
3. v prípade $|a| > 1$ použijeme zápis v tvare $x = \frac{1}{a}y - \frac{b}{a} = cy + d$, kde $|c| < 1$ a vo výslednom algoritme vzájomne zameníme x a y.

6.1 Rasterizácia úsečky s reálnou aritmetikou

Predpokladajme, že máme k dispozícií procedúru `PutPixel(x,y)`, ktorá vykresľuje pixel na pozícii `(x,y)`. Budeme priamo vypočítavať tie pixely, ktoré reprezentujú úsečku.

```
a=(float)(yk-yz)/(float)(xk-xz); b=(float)(yz-a*xz);
PutPixel(xz,yz);
for (x=xz+1; x<xk; x=x++) PutPixel(x,int(a*x+b));
PutPixel(xk,yk);
```

Našou snahou je zredukovať počet operácií, ktoré sa vykonávajú v cykle. K tomu využijeme fakt, že v priebehu cyklu sa hodnota premennej x postupne zväčšuje o jednotku

$$x_0 = x_Z, \quad x_{i+1} = x_i + 1, \quad x_n = x_K.$$

Rekurentný výpočet hodnoty y dáva

$$y_0 = y_Z, \quad y_{i+1} = ax_{i+1} + b = a(x_i + 1) + b = ax_i + b + a = y_i + a.$$

Rekurencia významným spôsobom redukuje náročnosť výpočtu. Namiesto priameho výpočtu hodnoty funkcie v nasledujúcom kroku, k aktuálnej hodnote funkcie pripočítame len prírastok, čo je spravidla jednoduchšie.

```
a=(float)(yk-yz)/(float)(xk-xz); y=(float)yz;
PutPixel(xz,yz);
for (x=xz+1; x<xk; x=x++){ y+=a; PutPixel(x,int(y));}
PutPixel(xk,yk);
```

Tento algoritmus sa v literatúre (napr. [9]) uvádza pod názvom DDA (Digital Differential Analyzer). Vidíme, že namiesto násobenia a sčítania v každom kroku cyklu vystačíme len s jedným inkrementovaním. Keďže a nie je celočíselné (viď (6.1)), bohužiaľ sa nevyhneme aritmetickým opreáciám v pohyblivej rádovej čiarke. Také opreácie sú však v porovnaní s aritmetikou celočíselnou spravidla podstatne pomalšie. Preto algoritmy využívajúce len celočíselnú aritmetiku (tam, kde je to pravdaže možné), môžeme považovať za kvalitnejšie. Takže aj keď z pohľadu formulovania je uvedený DDA algoritmus jednoduchý, z pohľadu výpočtovej zložitosti to tak nie je.

Ukážeme, že existuje algoritmus, ktorý využíva iba celočíselnú aritmetiku, a z hľadiska náročnosti realizovaných výpočtov je podstatne lepší.

6.2 Celočíselná rasterizácia úsečky (Bresenham)

Hlavná myšlienka toho, ako sa zbaviť aritmetiky v pohyblivej rádovej čiarke, je nasledujúca: keď máme priamku v tvare

$$y = ax + b$$

znamená to, že výraz

$$\phi(P) = \phi(x, y) = ax + b - y = \frac{p}{r}x + \frac{q}{r} - y \tag{6.3}$$

klasifikuje body roviny $P = (x, y)$ do troch tried:

$$\begin{array}{lcl} \phi(P) < 0 & \Leftrightarrow & P \text{ je nad priamkou} \\ \phi(P) = 0 & \Leftrightarrow & P \text{ je na priamke} \\ \phi(P) > 0 & \Leftrightarrow & P \text{ je pod priamkou} \end{array} \tag{6.4}$$

Pre zvolenú hodnotu x, $x_Z \le x \le x_K$, hľadáme takú dvojicu bodov rastrovej mriežky $P = (x, y)$, $P' = (x, y+1)$, že $\phi(P) \ge 0$ a zároveň $\phi(P') < 0$. Takto nájdený bod P budeme považovať za bod rastrovej reprezentácie úsečky.

Namiesto reálneho výrazu $\phi(P)$, *u ktorého nás zaujíma len znamienko*, stačí vyhodnocovať celočíselný výraz[1]

$$\begin{array}{rcl} \psi(P) & = & r\phi(P) = px + q - ry = \\ & = & (y_K - y_Z)\,x + (x_K y_Z - x_Z y_K) - (x_K - x_Z)y \end{array} . \tag{6.5}$$

Nasleduje detailné odvodenie algoritmu, v ktorom

[1] Predpokladáme (6.2), tj. že $x_Z < x_K$, a preto $r > 0$.

1. používame len celočíselnú aritmetiku,
2. minimalizujeme počet aritmetických operácií – toto dosiahneme podobne ako v prípade DDA tak, že vyhodnocovaný výraz $\psi(P)$ budeme vypočítavť rekurentne.

Predpokladajme, že máme pixel $P_i = (x_i, y_i)$, ktorý patrí do rastrovej reprezentácie analyzovanej úsečky. Takým bodom je napr. $P_Z = P_0 = (x_0, y_0)$.

Pre bod P_{i+1} dostávame nasledujúce dve možnosti[2]:

$$P_{i+1} = (x_{i+1}, y_{i+1}) \in \left\{P'_{i+1},\ P''_{i+1}\right\} = \{(x_i + 1, y_i),\ (x_i + 1, y_i + 1)\} \tag{6.6}$$

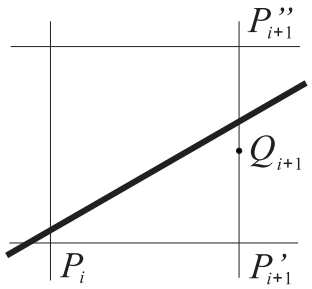
Ináč povedané, platí:

$$x_{i+1} = x_i + 1, \quad y_{i+1} \in \{y_i, y_i + 1\}\,.$$

Pre rozhodnutie o tom, ktorá z uvedených alternatív v (6.6) nastane, budeme sa snažiť nájsť čo najjednoduchšie kritérium.

Vezmime stred Q_{i+1} úsečky (P'_{i+1}, P''_{i+1}), $Q_{i+1} = \frac{P'_{i+1}+P''_{i+1}}{2} = (x_i + 1, y_i + \frac{1}{2})$

a analyzujme, do ktorej z tried (6.4) bod Q_{i+1} patrí.



Obr. 6.1: Príklad i-teho kroku analýzy. Vzhľadom na (6.4) $q_i = \phi(Q_{i+1}) > 0$, pretože bod Q_{i+1} je pod priamkou. Bod P''_{i+1} je k priamke bližšie, ako bod P'_{i+1}. Preto v tomto prípade určíme $y_{i+1} = y_i + 1$.

Na základe obr 6.1 má zmysel nasledujúce rozhodovacie kritérium[3]:

$$\begin{array}{lll} h_i = 2\psi(Q_{i+1}) \leq 0 & \Rightarrow & y_{i+1} = y_i \\ h_i = 2\psi(Q_{i+1}) > 0 & \Rightarrow & y_{i+1} = y_i + 1 \end{array} \tag{6.7}$$

Pre výpočet charakteristiky $h_i = 2\psi(Q_{i+1})$ odvodíme rekurentný vzťah.

Keďže P_0 leží na priamke, tj. $\psi(P_0) = 0$, a $x_1 = x_0 + 1$, preto dostávame

[2]Pripomíname, že uvažujeme úsečku so sklonom do 45°– (6.2).

[3]Dôvod, prečo je vhodné použiť dvojnásobok výrazu $\psi\,(Q)$ bude zrejmý z nižšie uvedeného odvodenia – vzťah (6.8). Zachováva sa celočíselnosť analyzovaného výrazu i v prípade, že v argumente je neceločíselná hodnota $\frac{1}{2}$.

$$\begin{aligned} h_0 &= 2\psi(Q_1) = 2\psi\left(x_1, y_0 + \tfrac{1}{2}\right) = \\ &= 2px_1 - 2r\left(y_0 + \tfrac{1}{2}\right) + 2q = 2px_0 + 2p - 2ry_0 - r + 2q = \\ &= 2px_0 - 2ry_0 + 2q + 2p - r = 2\psi(P_0) + 2p - r = 2p - r \end{aligned}$$

Podobne pre Q_{i+1} dostávame

$$\begin{aligned} h_i &= 2\psi(Q_{i+1}) = 2\psi\left(x_{i+1}, y_i + \tfrac{1}{2}\right) = 2px_{i+1} - 2r\left(y_i + \tfrac{1}{2}\right) + 2q \\ &= 2px_i - 2ry_i + 2p - r + 2q \end{aligned} . \tag{6.8}$$

Formálnym prepísaním vzťahu (6.8) pre $(i+1)$-vý krok s máme

$$h_{i+1} = 2px_{i+1} - 2ry_{i+1} + 2p - r + 2q = 2px_i - 2ry_{i+1} + 4p - r + 2q.$$

Odčítaním posledných dvoch vzťahov dostaneme

$$h_{i+1} - h_i = 2p - 2r(y_{i+1} - y_i).$$

Charakteristiky h_i tak môžeme vypočítať postupne:

$$h_0 = 2p - r, \quad h_{i+1} = h_i + 2p - 2r(y_{i+1} - y_i). \tag{6.9}$$

Využitím vzťahu (6.7) v (6.9) dostaneme

$$\begin{array}{lcllll} h_0 & = 2p - r, & & & & \\ h_i & \leq & 0, & \Rightarrow & y_{i+1} = y_i, & h_{i+1} = h_i + 2p \\ h_i & > & 0, & \Rightarrow & y_{i+1} = y_i + 1, & h_{i+1} = h_i + 2p - 2r \end{array} . \tag{6.10}$$

Toto vedie k výslednému zdrojovému textu (pre úsečku rastúcu i klesajúcu so sklonom do 45°, s predpokladom `xz<xk`).

```
if (yz>yk) d=-1; else d=1;
h1=2*abs(yk-yz); h2=h1-2*(xk-xz); h=h1-(xk-xz);
for (x=xz; x<xk; x=x++)
{ PutPixel(x,yz);
  if (h>0) {yz+=d; h+=h2;}
  else      h+=h1;
}
PutPixel(xk,yk);
```

Prípad úsečky s väčším sklonom ako 45° sa rieši jednoduchou modifikáciou zdrojového textu – „prehodením“ `x` a `y`:

```
if (xz>xk) d=-1; else d=1;
h1=2*abs(xk-xz); h2=h1-2*(yk-yz); h=h1-(yk-yz);
for (y=yz; y<yk; y=y++)
{ PutPixel(xz,y);
  if (h>0) {xz+=d; h+=h2;}
  else      h+=h1;
}
PutPixel(xk,yk);
```

Túto metódu v literatúre nájdete pod názvom *midpoint algorithm* a je to variant *Bresenhamovo algoritmu* [2]. Pre svoju jednoduchosť je implementovaná hardverovo v grafických kartách, čo ešte zvyšuje výslednú rýchlosť realizácie.

6.3 Zovšeobecnenie celočíselnej rasterizácie

Vzniká zákonite otázka, či neexistuje priama analógia Bresenhamovho algoritmu i pre zložitejšie krivky. Odpoveď je kladná. Podobný prístup, ako je opísaný vyššie, bol použitý pre rýchlu rasterizáciu kružnice [4]. Pre eliptický oblúk

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

má táto schéma tvar:

$$\begin{array}{lll} x_0 = 0 & y_0 = b, & h_0 = a^2 + 4b^2 - 4a^2 b \\ x_{i+1} = x_i + 1 & & \\ h_i \leq 0 \quad \Rightarrow & y_{i+1} = y_i, & h_{i+1} = h_i + 4b^2(2x_i + 3), \\ h_i > 0 \quad \Rightarrow & y_{i+1} = y_i + 1, & h_{i+1} = h_i + 4b^2(2x_i + 3) - 8a^2(y_i - 1) \end{array} \tag{6.11}$$

Skúste ju odvodiť sami. Pritom postupujte takisto, ako v prípade úsečky, akurát

1. tentokrát vyhodnocujeme výraz $\phi(P) = \phi(x, y) = b^2x^2 + a^2y^2 - a^2b^2$,
2. generovanie začíname v bode $P_Z = (0, b)$,
3. podmienka obmedzeného sklonu, má tvar $x_{i+1} = x_i + 1, \quad y_{i+1} \in \{y_i, y_i - 1\}$,
4. pre rozhodovacie kritérium $q_i = \phi(Q_{i+1})$ uvažujeme stred $Q_{i+1} = \left(x_i + 1, y_i - \frac{1}{2}\right)$ úsečky $\left((x_{i+1}, y_i), (x_{i+1}, y_i - 1)\right)$.

Analýza toho, kde používame schému typu

$$x_{i+1} = x_i + 1, \quad y_{i+1} \in \{y_i, y_i + d\}, \quad d \in \{-1, 1\}$$

a kde typu

$$y_{i+1} = y_i + 1, \quad x_{i+1} \in \{x_i, x_i + d\}, \quad d \in \{-1, 1\}$$

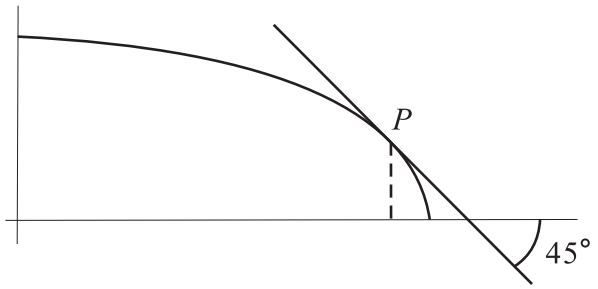
je pre úsečky jednoduchá – spravíme ju raz, pred samotným vykresľovaním, pretože úsečka má konštantný sklon. V prípade elipsy je schéma (6.11) použiteľná len pre tú jej časť, ktorá má sklon dotyčnice menší ako 45°, tj. pre

$$0 \leq x \leq \frac{a^2}{\sqrt{(a^2 + b^2)}}.$$

Na úseku

$$\frac{a^2}{\sqrt{(a^2 + b^2)}} \leq x \leq a$$

musíme použiť symetrickú schému (podobne ako u úsečky vzťah medzi schémami (**??**) a (**??**)). U všeobecnejších kriviek môže byť táto analýza natoľko zložitá, že celková efektivita takto koncipovaného algoritmu sa stráca.



Obr. 6.2: Bod, P v ktorom musíme prejsť od schémy (6.11) k symetrickej.

Preto sa všeobecnejšia krivka generuje nie priamou rasterizáciou, ale tak, ako bolo uvedené na začiatku kapitoly – ako lineárne lomená čiara, určená dostatočne veľkou množinou svojich bodov.

6.4 Vyššia kvalita vykresľovania úsečky

Uvažujeme dve reprezentácie úsečky s krajnymi bodami $P_Z P_K$.

1. *Pixelová reprezentácia*, tj. výsledok rasterizačného algoritmu, v ktorej je práve N pixelov.

2. Obdĺžniková reprezentácia, kde sa na úsečku pozeráme ako na obdĺžnik s rozmerami L a H. V prípade $H = 1$ hovoríme o *základnej obdĺžnikovej reprezentácii.*

Použijeme značenie

$\triangle x = |x_K - x_Z|\,,\ \triangle y = |y_K - y_Z|\,,\ N = \max\{\triangle x, \triangle y\}\,,\ L = \sqrt{\triangle x^2 + \triangle y^2}$.

Na rasterizovanej úsečke sa prejaví diskretizačná chyba niekoľkorakým spôsobom.

6.4.1 Schodovitosť

S touto formou diskretizačnej chyby sme sa stretli v kap. 4.6. Aby sme v tomto prípade eliminovali diskretizačnú chybu, je rozumné dať jednotlivým pixelom intenzitu, úmernú veľkosti plochy, ktorú základná obdlžníková reprezentácia v rastrovej mriežke vytyčuje. Priamy výpočet však príliš spomalí chod algoritmu. A ako ukazuje obr. 6.3 d), výsledok nie je ideálny – intenzita takto rasterizovanej úsečky sa periodicky mení. Dobrý výsledok dáva zjednodušený prístup, ktorý spočíva v nasledujúcom:

1. Každý horizontálny (resp. vertikálny v prípade sklonu väčšieho ako $45°$) úsek pixelovej reprezentácie vyplníme maximálnou intenzitou.
2. Intenzitu úseku nad a pod (resp. vpravo a vľavo) lineárne interpolujeme.

Keďže lineárnu interpoláciu vieme realizovať v celočíselnej aritmetike (Bresenhamov algoritmus) vykresľovanie úsečky s elimináciou schodovitosti môžeme spraviť taktiež bez použitia reálnej aritmetiky – obr. 6.3 c).

6.4.2 Korekcia intenzity

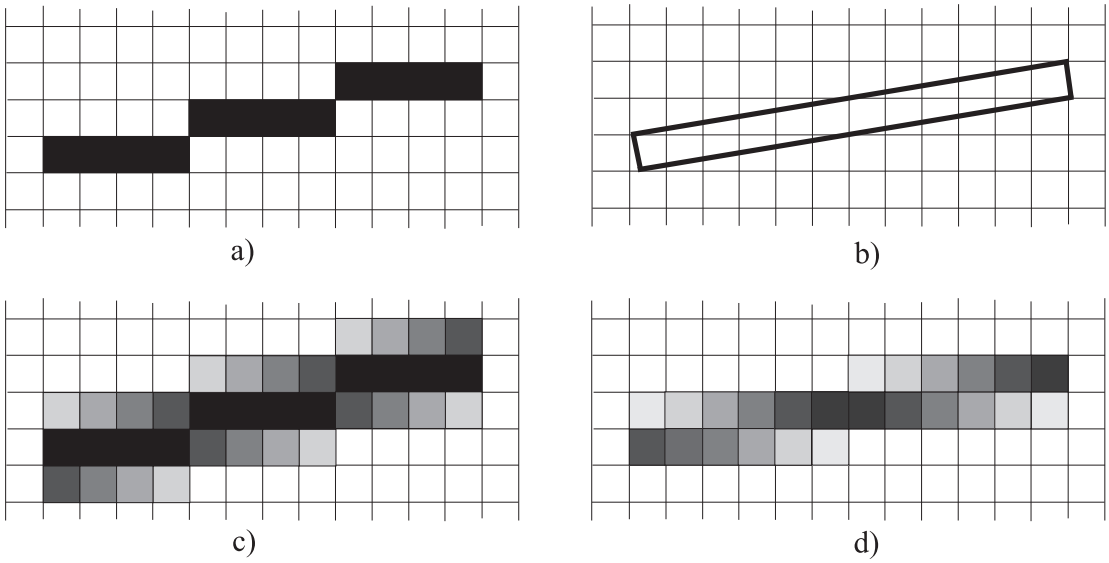
Stupeň intenzity danej farby je vnímaný ako pomer celkovej vyžarovanej energie a veľkosti objektu (kap. 2.2).

Uvažujme základnú obdĺžnikovú reprezentáciu úsečky, tj. obdĺžnik s $L = \sqrt{\triangle x^2 + \triangle y^2}$, $H = 1$. Jeho plocha je $S = L$.

Keďže počet pixelov N predstavuje celkovú vyžiarenú energiu, dostávame tak intenzitu farby úsečky (tj. vyžarovanie jednotkového štvorca)

$$C = \frac{N}{S} = \frac{N}{L} = \frac{\max\{\triangle x, \triangle y\}}{\sqrt{\triangle x^2 + \triangle y^2}}. \tag{6.12}$$

Znamená to, že použitie štandardnej pravouhlej mriežky pre diskretizáciu vedie k tomu, že vnímaná intenzita farby úsečiek s rôznym sklonom bude rôzna. Napr. pre úsečku u_1 s krajnými bodami (1,1)-(n,n) je $C_1 = \frac{n}{n\sqrt{2}} = \frac{1}{\sqrt{2}}$. Pre úsečku u_2 s krajnými bodami (1,1)–(n,1) je intenzita $C_2 = \frac{n}{n} = 1$. Aby sme dostali úsečku, farba ktorej nezávisí na jej sklone, musíme intenzitu korigovať – prenásobiť hodnotou $1/C$.



Obr. 6.3: Detailný pohľad na rasterizovanú úsečku: a) pixelová reprezentácia, b) obdlžníková reprezentácia, c) lineárna interpolácia intenzít, d) intenzity z pomeru plôch obdlžníkovej reprezentácie.

6.4.3 Korekcia hrúbky

Podobne ako na intenzitu, má rastrová mriežka vplyv i na vnímanie hrúbky vyklresľovanej čiary. Pre korekciu hrúbky môžeme použiť ten istý mechanizmus ako pre korekciu intenzity, akurát s inou interpretáciou.

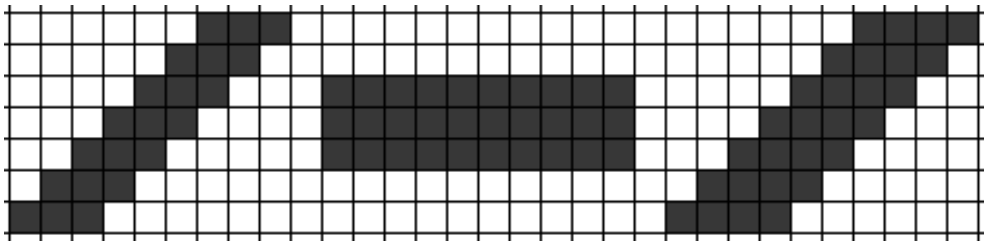
Uvažujme obdlžníkovú reprezentáciu úsečky s $L = \sqrt{\triangle x^2 + \triangle y^2}$, $H = C = \frac{N}{L}$. Jeho plocha je v tomto prípade $S = H \times L = N$. Tj. jedná sa o úsečku s požadovanou maximálnou intenzitou farby, ale s menšiu hrúbku vykresľovanej čiary. Preto ak požadujeme úsečku s obdlžníkovou reprezentáciou hrúbky K, jej pixelová reprezentácia musí pozostávať z

$$KC^{-1} = K\frac{L}{N} = K\frac{\sqrt{\triangle x^2 + \triangle y^2}}{\max\{\triangle x, \triangle y\}}$$

pixelových reprezentácií pôvodnej úsečky. Napr. rasterizovanej úsečke so sklonom 45^o hrúbky 3 pixely lepšie odpovedá 4-násobná pixelová reprezentácia – obr. 6.4 c) ako 3-násobná ($3\sqrt{2} \doteq 3 \cdot 1.4 = 4.2$) – obr. 6.4 a).

6.4.4 Korekcia vzoru

Okrem bežného vykreslenia úsečky plnou čiarou v aplikáciách vyžadujeme možnosti kresliť čiary rôznym vzorom, napr. prerušovane, kde sa striedajú vykresľované a nevykresľované elementárne segmenty. Prirodzenou požiadavkou je, aby oba konce úsečky boli vykreslené podľa možnosti rovnakou vykreslenou časťou použitého vzoru – obr. 6.5.



Obr. 6.4: Úsečky a) diagonálna a b) horizontálna hrúbky 3 pixely, c) hrúbky 4 pixely.

Obr. 6.5: Úsečka požadovanej dĺžky vykreslená štandardne, prerušovane nesprávne (pravý koncový bod je v oblasti nevykresenej časti vzoru, čím nastáva skrátenie vykreslenej úsečky) a prerušovane správne (oba konce úsečky sú vykreslené).

6.4.5 Zakončenie úsečky

Vychádzajme z obdĺžnikovej reprezentácie úsečky. Pri generovaní lineárne lomenej čiary, ktorá má hrúbku väčšiu ako jednotkovú, vzniká na spoji úsečiek nežiadúci jav – obr. 6.6 a). Relatívne jednoduché riešenie je kresliť úsečku ako ovál, tj. zakončenia realizovať ako polkruhy – obr. 6.6 b).

Obr. 6.6: Úsečky reprezentované ako a) obdlžníky a ako b) ovály.

6.5 Zhrnutie a úlohy

Objasnili sme metódy rýchlej rasterizácie kriviek.

- Úsečku je možné rasterizovať s využitím len celočíselnej aritmetiky.
- Celočíselné algoritmy je možné použiť i pre zložitejšie krivky – obmedzenie použitia celočíselného algoritmu rasterizácie súvisí s meniacim sa sklonom krivky.

- Pre vyššiu kvalitu zobrazovania kriviek je nutné eliminovať diskretizačnú chybu.

Kontrolné otázky:

1. Na základe čoho je možné pri rasterizácii kriviek nahradiť aritmetiku v pohyblivej rádovej čiarke výhradne celočíselnou?
2. Pri nahradení aritmetiky v pohyblivej rádovej čiarke celočíselnou sa využíva predpoklad, že smernica priamky je vždy racionálne číslo – (6.5). Ako si Bresenhamov algoritmus poradí s úsečkou, ktorá má sklon iracionálny? Napr. úsečka, $y = \sqrt{3}x + 1$, $0 \leq x \leq 1$?
3. Aký sklon úsečky vyžaduje najväčšie korekcie jasu, hrúbky a dĺžky?
4. Aké vzájomné proporcie prerušovanej čiary nám zaručujú korektné vykreslenie (tj. že oba konce úsečky náležia vykreslenej časti vzoru)?

Samostatné úlohy:

1. Realizujte algoritmy DDA a Bresenham pre kreslenie úsečky a porovnajte ich z hľadiska rýchlosti.
2. Porovnajte elimináciu schodovitosti metódou lineárnej interpolácie jasu a metódou konvolučného vyhladzovania.

Kapitola 7

Algoritmy orezávania

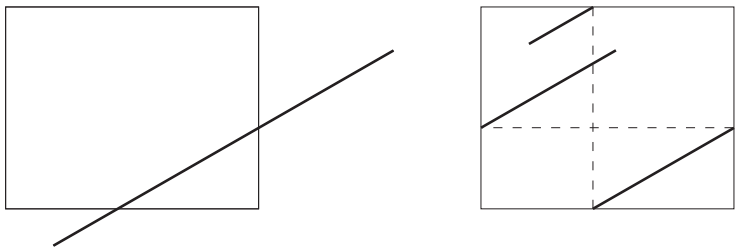
Nevyhnutnosť riešiť problém orezávania je daná spôsobom výpočtu adresy pixelov pri rasterizácii objektu. Podobne, ako pri prekročení maximálnej povolenej hodnoty napr. pre čísla

```
char i=127, j=-128;
i++; j--;
```

budú výsledné hodnoty `i=-128,j=127`, tak v prípade, že sa napr. x-ová súradnica požadovaného pixelu x' dostane mimo dovolený interval hodnôt $x_{min} \leq x \leq x_{max}$, určí sa jeho skutočná hodnota x spravidla ako

$$x = x_{min} + (x' - x_{min}) \mod (x_{max} - x_{min}),$$

čo spôsobí „navinutie obrazu“ do obrazovej pamäti.



a) b)

Obr. 7.1: Navinutie úsečky na obrazovku. a) Skutočnosť. b) Výsledné vykreslenie.

Budeme sa venovať hlavne *orezávaniu úsečky* pravouhlým oknom. Všimneme si však i zovšeobecnenia tohto problému – orezávanie úsečky m-uholníkovým konvexným oknom. Ako sa ukáže, dá sa tento prístup zovšeobecniť aj na orezávanie úsečky 3D konvexným oknom. V závere kapitoly je opísaný jednoduchý algoritmus orezávania mnohouholníkov.

Okrem riešenia problému ktorý je demonštrovaný na obr. 7.1, úlohy orezávania vo všeobecnejšej formulácii vznikajú napríklad pri snahe urýchliť proces vykresľovania: procedúry zobrazovacieho reťazca predchádzajúce zápisu do obrazovej pamäti nerobíme so všetkými objektami, ale len s tými, ktoré sa nachádzajú v pre nás zaujímavej časti zobrazovanej scény (ktorá je definovaná ako konvexné 3D okno). 3D orezávanie využijeme napr. i v prípade vykreslenia tieňa, vrhnutého vybraným objektom na ostatné objekty v scéne. (Toto vedie ku konštrukcii „tieňového telesa“ a hľadaniu prieniku ostatných objektov scény s ním.)

Ukážeme, ako jednotlivé algoritmy spolu vzájomne súvisia, a zároveň to, akým spôsobom zmenené predpoklady zmenia výsledné riešenie.

7.1 Orezávanie izolovaných bodov

Uvažujme pravouhlé okno Ω zadané hodnotami x_{min}, x_{max}, y_{min}, y_{max}. Najjednoduchší spôsob orezávania spočíva v tom, že tesne pred zápisom do videopamäti testujeme, či pixel je v požadovanom okne:

$$\texttt{if } ((x \geq x_{min}) \wedge (x \leq x_{max}) \wedge (y \geq y_{min}) \wedge (y \leq y_{max})) \texttt{ PutPixel}(x, y);$$

Pre izolované body je toto jediné riešenie. V prípade zložitejších objektov sa budeme snažiť využiť ich určité geometrické vlastnosti, resp. budeme chcieť používať takú triedu objektov, ktorá dovoľuje jednoduchšie riešenie orezávania. Pre tieto účely sa ukazujú byť vhodné konvexné objekty. Pritom pre riešenie orezávania sa intenzívne využíva priamo vlastnosť, ktorá definuje konvexitu:

$$\mathcal{A} \text{ je konvexná } \iff (\text{ body } P, Q \in \mathcal{A} \Rightarrow \text{ úsečka } PQ \subset \mathcal{A}).$$

Využijeme i nasledujúcu vlastnosť konvexných množín,

$$\mathcal{A}, \mathcal{B} \text{ - konvexné množiny} \Longrightarrow \mathcal{A} \cap \mathcal{B} \text{ - konvexná množina},$$

komutativitu a asociativitu operácie prieniku,

$$\mathcal{A} \cap \mathcal{B} = \mathcal{B} \cap \mathcal{A}, \quad (\mathcal{A} \cap \mathcal{B}) \cap \mathcal{C} = \mathcal{A} \cap (\mathcal{B} \cap \mathcal{C}),$$

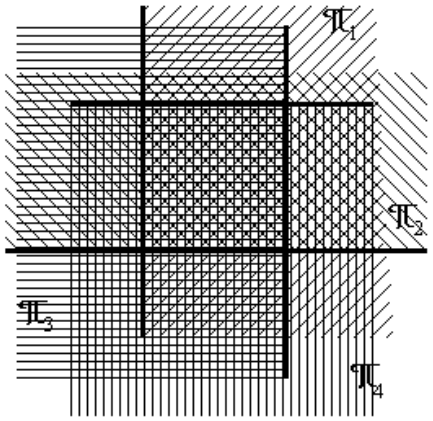
ako i triviálny fakt, že pre ľubovoľné množiny $\mathcal{A}, \mathcal{B}$ platí

$$\mathcal{A} \cap \mathcal{B} \subseteq \mathcal{A}, \quad \mathcal{A} \cap \mathcal{B} \subseteq \mathcal{B}.$$

Uvažujme pravouhlé okno Ω ako prienik polrovín, $\Omega = \pi_1 \cap \pi_2 \cap \pi_3 \cap \pi_4$, kde

$$\pi_1 = \{(x, y) : x \geq x_{min},\, y \in R\}, \quad \pi_2 = \{(x, y) : y \geq y_{min},\, x \in R\},$$
$$\pi_3 = \{(x, y) : x \leq x_{max},\, y \in R\}, \quad \pi_4 = \{(x, y) : y \leq y_{max},\, x \in R\}.$$

Hraničné priamky polrovín π_1, π_2, π_3, π_4 označíme p_1, p_2, p_3, p_4.



Obr. 7.2: Okno ako prienik polrovín.

7.2 Orezávanie úsečky

Uvažujme úsečku s krajnými bodami $A = (x_A, y_A)$, $B = (x_B, y_B)$. Predpokladajme, že aspoň jeden z nich leží mimo Ω. Ďalej predpokladajme, že $x_A \neq x_B$, $y_A \neq y_B$[1]. V takom prípade musíme nájsť priesečníky $AB \cap p_i$ s niektorou z hraničných priamok p_i. Pre $P_i = AB \cap p_i$ dostávame

$$\begin{aligned} P_1 &= (x_1, y_1) &&= \left(x_{min}\,,\; y_A + \tfrac{y_B - y_A}{x_B - x_A}(x_{min} - x_A)\right), \\ P_2 &= (x_2, y_2) &&= \left(x_A + \tfrac{x_B - x_A}{y_B - y_A}(y_{min} - y_A)\,,\; y_{min}\right), \\ P_3 &= (x_3, y_3) &&= \left(x_{max}\,,\; y_A + \tfrac{y_B - y_A}{x_B - x_A}(x_{max} - x_A)\right), \\ P_4 &= (x_4, y_4) &&= \left(x_A + \tfrac{x_B - x_A}{y_B - y_A}(y_{max} - y_A)\,,\; y_{max}\right). \end{aligned} \tag{7.1}$$

7.2.1 Základný algoritmus

Keďže oblasť je ohraničena priamkami p_1, p_2, p_3, p_4, hľadaná orezaná úsečka, **1.** musí byť časťou úsečky AB, **2.** jej krajné body sú niektoré z bodov A, B, P_1, P_2, P_3, P_4.

Ponúka sa nasledujúci algoritmus orezávania úsečky.

1. `Ak` $A \in \Omega \wedge B \in \Omega$ `potom vykresli úsečku` AB.
   `Koniec.`
2. `Spočítaj body` P_1, P_2, P_3, P_4.
3. `Usporiadaj body` A, B, P_1, P_2, P_3, P_4, `vzostupne podľa x-ovej súradnice (usporiadanie označíme` $\preceq$`).`
4. `Ak žiaden z bodov` A, B, P_1, P_2, P_3, P_4, `neleží v` Ω,

[1] Horizontálne a vertikálne úsečky sa analyzujú jednoducho – nechávame toto na čitateľovi.

potom úsečka s oknom Ω prienik nemá.
Koniec.

5. Ak $A \in \Omega \wedge A \preceq B$, potom kresli úsečku AP, $P \in \{P_1, P_2, P_3, P_4\}$, kde $A \preceq P$ a P je bezprostredný pravý sused bodu A.
Koniec.
6. Ak $B \in \Omega \wedge A \preceq B$, potom kresli úsečku PB, $P \in \{P_1, P_2, P_3, P_4\}$, kde $P \preceq B$ a P je bezprostredný ľavý sused bodu B.
Koniec.
7. Ak $A \notin \Omega \wedge B \notin \Omega$, potom nájdi dvojicu bodov $P, Q \in \{P_1, P_2, P_3, P_4\}$takú, že $P, Q \in \Omega$.
Kresli úsečku PQ.
8. Koniec.

Pre korektnosť uvedeného algoritmu je treba

a) ošetriť prípady $x_A = x_B$, resp. $y_A = y_B$ v kroku 2,
b) zabezpečiť existenciu bodu P v krokoch 5 resp. 6,
c) zabezpečiť existenciu dvojice bodov P, Q v kroku 7,
d) nájsť jednoduchý spôsob vyhodnotenia vzťahu $P \in \Omega$.

Podmienka a) znamená úsečku rovnobežnú s osou x resp. y. Nad prípadmi b), c), d) nech sa zamyslí čitateľ.

7.2.2 Poznámky k zefektívneniu orezávania úsečky

Dôležité je si uvedomiť, že na rozdiel od generovania úsečky, kde sme si vystačili s celočíselnou aritmetikou, tu sa nevyhneme číselným formátom s plávajúcou desatinnou čiarkou. V tomto prípade je test dvojíc čísel na ich rovnosť nestabilný – toto vyžaduje opatrnosť a dôslednú analýzu. Podstatným rysom všetkých orezávacích algoritmov je fakt, že v priebehu analýzy hľadáme prieniky dvojíc priamok (priamka, na ktorej leží analyzovaná úsečka a priamka, tvoriaca hranicu okna). Toto môže nemať riešenie v prípade, že priamky sú rovnobežné (to ale znamená test na rovnosť dvoch čísel – viď poznámka vyššie) resp. mať nestabilné riešenie (priamky sú skoro rovnobežné). Ďalším kameňom úrazu býva situácia, keď orezávaná priamka prechádza vrcholom orezávacieho okna (prienik trojice priamok). Nepresnosť spôsobená zaokrúhľovacou chybou v aritmetike s plávajúcou desatinnou čiarkou môže viesť k nesprávnym výsledkom.

Pri popise nižšieuvedených algoritmov na tieto skutočnosti nebudeme upozorňovať, a venujeme sa *hlavne vysvetleniu hlavných myšlienok uvedených algoritmov*. Pri implementácii však čitateľ musí na tieto úskalia myslieť.

Podstatné črty základného algoritmu, popísaného v predošlej časti, spočívajú v tom, že

- *najprv spočítame všetky priesečiky P_i (s výnimkou prípadu $A, B \in \Omega$) a až potom, na základe usporiadania potenciálnych krajných bodov A, B, P_1, P_2, P_3, P_4, rozhodneme o výsledku orezávania.*

Pre zefektívnenie orezávania budeme postupovať tak, že

- *hneď po spočítaní priesečníka P_i sa ho budeme snažiť umiestniť v požadovanom usporiadaní na správne miesto.*

Formálne vyjadrené, prienik

$$AB \cap \Omega = AB \cap (\pi_1 \cap \pi_2 \cap \pi_3 \cap \pi_4)$$

konštruujeme v základnom algoritme tak, že hľadáme *všetky* prieniky *priamky a polorovín*

$$AB \cap \pi_1, \quad AB \cap \pi_2, \quad AB \cap \pi_3, \quad AB \cap \pi_4.$$

Nosná konštrukcia pre zefektívnenie orezávania spočíva v použití inej elementárnej operácie, ako je prienik priamky s polrovinou. Vhodným sa ukazuje byť nasledujúce:

$$\begin{aligned} AB \cap \Omega &= AB \cap (\pi_1 \cap \pi_2 \cap \pi_3 \cap \pi_4) = \left(\vec{AB} \cap \vec{BA}\right) \cap (\pi_1 \cap \pi_2 \cap \pi_3 \cap \pi_4) = \\ &= \left(\vec{AB} \cap \pi_i\right) \cap \left(\vec{BA} \cap \pi_j\right), \end{aligned} \tag{7.2}$$

kde π_i, π_j sú niektoré z polrovín, vymedzujúcich okno. Tj. namiesto *prieniku AB s každou z polrovín budeme hľadať vhodné dvojice polpriamka – polrovina.* Pritom „vhodná“ dvojica je taká, že

(e1) začiatok A polpriamky je mimo polrovinu,

(e2) polpriamka má s polrovinou neprázdny prienik medzi bodami A, B – ináč povedané bod B je v polrovine.

Operáciu $\vec{AB} \cap \pi = \vec{PB}$ nazveme: P je projekcia polpriamky A na polrovinu π.

7.2.3 Algoritmus Cohen–Sutherland (C-S)

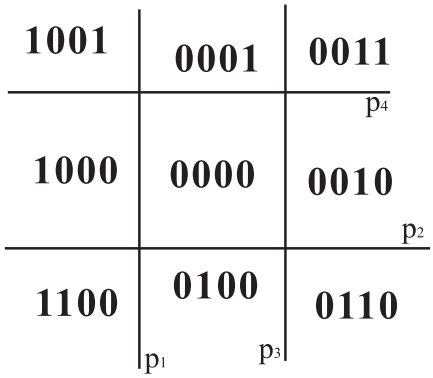
Každému bodu $P = (x, y)$ priradíme 4-bitový kód $c^P = c_1^P c_2^P c_3^P c_4^P$ nasledujúcim spôsobom:

$$\begin{array}{llll} c_1^P = 1 & \iff & P \notin \pi_1 & \qquad c_2^P = 1 \iff P \notin \pi_2 \\ c_3^P = 1 & \iff & P \notin \pi_3 & \qquad c_4^P = 1 \iff P \notin \pi_4 \end{array} \tag{7.3}$$

V ostatných prípadoch je $c_i^P = 0$. Pre pravouhlé okno je výpočet veľmi jednoduchý:

$$\begin{array}{llll} c_1^P = 1 & \iff & x < x_{mi} & \qquad c_2^P = 1 \iff y < y_{mi} \\ c_3^P = 1 & \iff & x > x_{ma} & \qquad c_4^P = 1 \iff y > y_{ma} \end{array} \tag{7.4}$$

Dostávame tak rozklad roviny na 9 oblastí s rôznymi kódami. Kódy charakterizujú polohu bodu vzhľadom k oknu, ktorú môžeme vyjadriť ako „vľavo“, „pod“, „vpravo“, „nad“.



Obr. 7.3: Kódy oblastí, ktoré vytyčujú hraničné priamky.

Na vylúčenie úsečky z orezávania sa využíva konvexita polroviny a úsečky: napr. ak obidva krajné body úsečky sú „pod“ oknom, potom celá úsečka je „pod“ oknom. Prepísaním tejto konštrukcie do všeobecnej formálnej podoby dostávame

$$(c^A \& c^B \neq 0000) \implies (AB \cap \Omega = \emptyset). \tag{7.5}$$

kde $c^A \& c^B$ je bitový súčin kódov, tj. $(c^A \& c^B)_i = c_i^A \wedge c_i^B$.

Z druhej strany, keď sú obidva konce úsečky v okne, tak celá usečka je v okne

$$\big((c^A = 0000) \wedge (c^B = 0000)\big) \implies (AB \cap \Omega = AB). \tag{7.6}$$

Máme tak veľmi jednoduché kritérium pre úsečky, ktoré

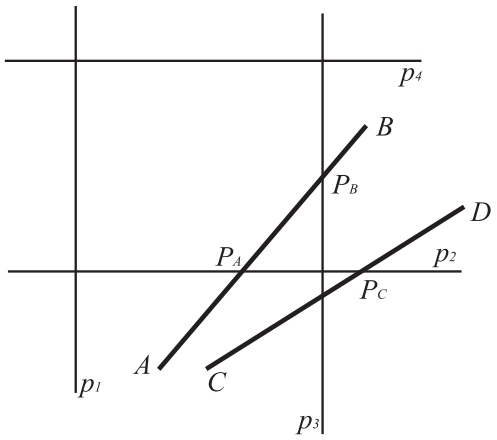
1. môžeme úplne vylúčiť z vykresľovania – (7.5),
2. bez úpravy celé vykresľujeme – (7.6).

V ostatných prípadoch podmienky (e1), (e2) s pomocou kódov môžeme sformulovať takto:

- Bod A spĺňa (e1) $\Leftrightarrow c^A \neq 0000$.
- Polpriamka $\vec{AB}$ spĺňa (e2) $\Leftrightarrow \big(c^A \neq 0000\big) \wedge (c^A \& c^B = 0000)$.

Priebeh analýzy orezávania demonštrujeme na úsečkách AB a CD z obr. 7.4. V tejto situácii sa nevyhneme hľadaniu priesečikov P_i. Použijeme pritom vyššiespomínanú konštrukciu (7.2).

Analyzujme najprv úsečku AB. Z kódu koncového bodu A, $c^A = 0100$ vidíme, že má zmysel uvažovať polpriamku $\vec{AB}$ a polrovinu π_2. Presunieme preto bod A do jeho projekcie $P_2 = P_A$. Dostávame kód $c^{P_A} = 0000$, tj. našli sme úspešne požadovaný bod. Pre dokončenie orezávania uvažujme teraz polpriamku $\vec{BA}$. Keďže $c^B = 0010$, budeme hľadať projekciu P_3



Obr. 7.4: Úsečky, ktorých vzájomne si odpovedajúce krajné body majú rovnaké kódy, pričom AB oknom prechádza a CD neprechádza.

bodu B na polrovinu π_3, tj. $P_3 = P_B$. Vzhľadom na to, že $c^{P_B} = 0000$, hľadaným celkovým riešením je úsečka $P_A P_B$.

Analyzujme teraz úsečku CD. Keďže $c^C = 0100$, bod C podobne ako bod A projektujeme na π_2, $P_2 = P_C$. Tentokrát $c^{P_C} = 0010$. Prejdeme teraz k projekcii polpriamky $\vec{DC}$. Pretože $c^D = 0010$, nie je splnená podmienka (e2) (D síce je vonkajší vzhľadom na π_3, no P_C nie je vzhľadom na π_3 vnútorný) – preto úsečka okno nepretína.

I v prípade úsečky AB, i v prípade úsečky CD sme projektovali krajné body každej z uvažovaných polpriamok len raz, pretože kódy krajných bodov obsahovali práve jednu hodnotu 1 (čiže počiatok polpriamky bol vždy vonkajší vzhľadom k práve jednej polrovine). Keďže kód ľubovoľného bodu roviny obsahuje maximálne dve jednotky, vo všeobecnom prípade budeme každý krajný bod projektovať maximálne dvakrát. Preto

- *prípad, že úsečka oknom neprechádza, zistíme najneskôr po druhej projekcii krajného bodu.*

Keď si uvedomíme, že pri vykreslení detailu pôvodného obrázku je spravidla veľká časť objektov mimo vykresľovacie okno, vyššieuvedený vzťah ukazuje, že tento prístup podstatným spôsobom základný orezávací algoritmus urýchľuje.

Bohužiaľ, v situácii, keď úsečka okno pretína, vždy existujú také prípady, že projekciu koncového bodu musíme realizovať 4-krát (2-krát projekcia každého krajného bodu pôvodnej úsečky). Necháva sa čitateľovi také prípady ukázať.

Na základe vyššieuvedeného je sformulovaný nasledujúci algoritmus (Cohen–Sutherland) [26].

1. Spočítaj kód c^B krajného bodu B úsečky AB.
2. Spočítaj kód c^A krajného bodu A úsečky AB.

3. Ak $(c^A = 0000) \wedge (c^B = 0000)$, potom kresli úsečku AB.
   Koniec.
4. Ak bitový súčin $c^A \& c^B \neq 0000$, úsečka nepretína okno.
   Koniec.
5. Keď $c^A = 0000$ prehoď krajné body.
6. Nájdi projekciu P bodu A, $P = \vec{AB} \cap p$.
7. Priraď $A = P$ a pokračuj krokom 2.
8. Koniec.

Niekoľko poznámok k zovšeobecneniu a efektivite

Optimalizácia počtu projekcií Uvažujme orezávanú úsečku, ktorá oknom neprechádza. V prípade, že v bitovom kóde koncového bodu je len jedna nenulová hodnota, máme determinované, ktorou hraničnou priamkou musíme tento koncový bod projektovať. To znamená, že zistenie, že úsečka oknom neprechádza, dosiahneme už po jedinej projekcii.

V prípade, že v bitovom kóde sú dve nenulové hodnoty, je možné, že k vylúčeniu úsečky z vykresľovania dospejeme až po druhej projekcii. Znamená to, že ak oba koncové body majú nenulový bitový kód, projekciu je dobré začať tým krajným bodom, ktorý má v kóde jediný nenulový bit (ak taký existuje).

Redukcia počtu delení Z operácií, ktoré sú nutné v procese orezávania, je najdrahšou delenie v reálnej aritmetike. Pre každú projekciu (krok 6. algoritmu C-S je nutné jedno delenie – (7.1). Maximálny počet delení – 4 však môžeme zredukovať na polovicu vzhľadom na to, že orezávacie priamky sú navzájom rovnobežné, $p_1 || p_3$, $p_2 || p_4$. Výpočty v (7.1) zorganizujeme nasledujúco:

$$\begin{aligned} &D = (x_B - x_A)/(y_B - y_A), \\ &x_2 = x_A + D(y_{mi} - y_A),\ x_4 = x_A + D(y_{ma} - y_A), \\ &D = 1/D, \\ &y_1 = y_A + D(x_{mi} - x_A),\ y_3 = y_A + D(x_{ma} - x_A). \end{aligned}$$

Okrem toho, ďalšie urýchlenie sa dá získať nezávislou analýzou všetkých, do úvahy prichádzajúcich možností dvojíc kódov koncových bodov orezávanej úsečky. Experimentálne sa dosiahlo oproti C-S algoritmu až dvojnásobné zrýchlenie [5].

Zovšeobecnenie orezávacieho okna Keďže sme v C-S algoritme využívali len vzťah *incidencie bodu k polrovine* a operáciu *prienik poloroviny a polpriamky*, otázkou je, nakoľko je podmienka pravouhlého okna pre algoritmus podstatná.

Pri pozornej revízii čitateľ zistí, že algoritmus sa dá sformulovať i pre obecné konvexné m-uholníkové okno. V tomto prípade každému vrcholu bude odpovedať m-bitový kód a zložitosť realizácie sa zvýši ešte tým, že test príslušnosti bodu polrovine je v tomto prípade

zložitejší ako (7.4). Samozrejme v prípade vzájomne rôznobežných hraničných priamok nemožno aplikovať redukciu počtu delení.

Zovšeobecnenie do 3D Algoritmus C-S zostáva platným i pri prechode do 3D. Musíme si ale uvedomiť, že teraz prepokladáme okno ako prienik polpriestorov, a preto polpriamky $\vec{AB}$, $\vec{BA}$ neprojektujeme na hraničné polroviny, ale na hraničné polpriestory.

7.2.4 Algoritmus Cyrus–Beck (C-B)

V algoritme C-S je podstatné si uvedomiť, že jeden krajný bod úsečky projektujeme na hraničné priamky „pokiaľ to je možné“ – a až potom prejdeme k projekciám druhému krajnému bodu – tj. postup analýzy *je vzhľadom na krajné body úsečky.*

Pozrieme sa teraz na inú organizáciu projekcií, keď cyklus bude prebiehať *vzhľadom na hraničné priamky orezávacieho okna.*

Algoritmus sformulujeme hneď pre prípad všeobecného konvexného m-uholníkového okna v 2D, ktoré je zadané postupnosťou bodov $O_1, O_2, \ldots, O_m, \quad O_i = (x_i, y_i)$. Pritom uvažujeme orientáciu hranice proti smeru pohybu hodinových ručičiek. Pre zjednodušenie zápisu uvažujme ešte jeden bod $O_{m+1} = O_1$. Každá z polrovín π_i, ktorá toto okno určuje, $(\Omega = \pi_1 \cap \pi_2 \cap \cdots \cap \pi_m)$ je daná hraničnou priamkou

$$p_i = \overline{O_i O_{i+1}}, \quad i = 1, 2, \ldots, m$$

a vonkajšou normálou

$$\vec{n}_i = (y_{i+1} - y_i, x_i - x_{i+1}), \quad i = 1, 2, \ldots, m.$$

(Presnejšie – na určenie polroviny π_i stačí jeden bod O_i a vonkajšia normála v ňom $\vec{n}_i$.)

Uvažujme priamku $\overline{AB} = \{P : P = A + t(B - A), t \in R\}$. Pre vonkajšiu normálu $\vec{n}_i$ polroviny π_i a smerový vektor $\vec{v} = B - A$ priamky $\overline{AB}$ platí práve jeden zo vzťahov:

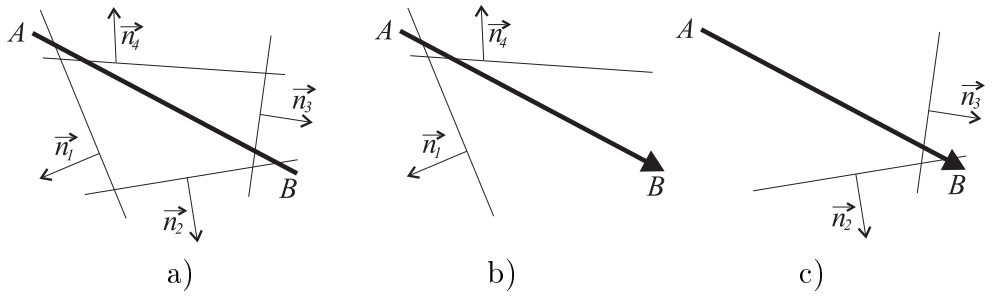
$$(\vec{v}, \vec{n}_i) < 0, \qquad (\vec{v}, \vec{n}_i) > 0, \qquad (\vec{v}, \vec{n}_i) = 0, \tag{7.7}$$

kde $(\vec{u}, \vec{v}) = |\vec{u}| \,.\, |\vec{u}| \,.\, \cos\alpha$ je skalárny súčin vektorov $\vec{u}, \vec{v}$ a α je uhol medzi vektormi $\vec{u}, \vec{v}$.

Budeme sledovať pohyb bodu P po priamke od bodu A k bodu B s rastúcim t.

Prvý prípad v (7.7) nastáva, keď s rastúcim t v niektorom momente t_i priamka *vstupuje* do polroviny π_i – obr. 7.5 b), druhý, naopak, keď v niektorom momente t_i priamka *vystupuje* z polroviny π_i – obr. 7.5 c). V treťom prípade celá priamka je buď vovnútri alebo mimo polrovinu π_i.[2]

[2] V kontexte konštrukcie z 7.2.2, (tj. hľadanie prieniku polpriamky s polrovinou) prvý prípad zodpovedá projekcii $\vec{AB} \cap \pi_i$, druhý prípad zodpovedá projekcii $\vec{BA} \cap \pi_i$.



Obr. 7.5: Orezávanie priamky AB. b) S rastúcim parametrom t priamka „vstupuje“ do polrovín π_1, π_4 a c) „vystupuje“ z polrovín π_2, π_3.

Prienikom polroviny π_i a priamky je tak v prvom prípade polpriamka

$$\overline{AB} \cap \pi_i = \{P \,:\, P = A + t(B - A),\, t \geq t_i\}.$$

Keďže zo všetkých týchto prípadov hľadáme prienik, vo všetkých týchto prípadoch nás bude zaujímať najväčšia hodnota $t_m = \max\{t_i\}$ – obr. 7.5 b).

V druhom prípade je prienikom polroviny a priamky polpriamka

$$\overline{AB} \cap \pi_i = \{P \,:\, P = A + t(B - A),\, t \leq t_i\}.$$

Opäť nás zaujíma prienik týchto prípadov, čo tenokrát vedie k nájdeniu najmenšej hodnoty $t_M = \min\{t_i\}$ – obr. 7.5 c).

V treťom prípade, $(\vec{v}, \vec{n}_i) = 0$ je priamka AB rovnobežná s hraničnou priamkou, preto spoločný prienik nemajú[3].

Zjednodušene povedané,

```
Pre každú hraničnú priamku orezávacieho okna vykonaj kroky 1-4:
1. Spočítame prienik priamky AB a hraničnej priamky.
2. Keď polpriamka AB vstupuje prienikom do vnútra polroviny,
   aktualizujeme ľavý koniec výslednej orezanej úsečky.
3. Keď polpriamka AB vystupuje prienikom von z polroviny,
   aktualizujeme pravý koniec výslednej orezanej úsečky.
4. Keď sa ľavý koniec dostane za pravý, resp. pravý pred ľavý,
   priamka AB oknom neprechádza.
```

Nájdenie prieniku priamok $\overline{AB}$, $\overline{O_i O_{i+1}}$ odpovedá nájdeniu parametra t_i takého, že

$$\begin{aligned} X = &\; A + t_i \vec{v}, \\ X = &\; O_i + s\,(O_{i+1} - O_i)\,. \end{aligned}$$

[3] V praxi tento prípad býva problematický. Test na nulovú hodnotu v aritmetike s plávajúcou desatinnou čiarkou býva vzhľadom na zaokrúhľovaciu chybu nestabilný.

Skalárnym prenásobením oboch rovníc normálovým vektorom $\vec{n}_i$ dostávame

$$(A, \vec{n}_i) + t_i\,(\vec{v}, \vec{n}_i) = (O_i, \vec{n}_i) + s\,(O_{i+1} - O_i, \vec{n}_i)\,.$$

Vzhľadom na ortogonalitu $\vec{n}_i \perp (O_{i+1} - O_i)$ je skalárny súčin $(O_{i+1} - O_i, \vec{n}_i) = 0$, a preto

$$t_i = \frac{(O_i - A, \vec{n}_i)}{(\vec{v}, \vec{n}_i)}$$

(samozrejme za predpokladu, že priamka $\overline{AB}$ nie je rovnobežná s hraničnou priamkou p_i).

Algoritmus je možné sformulovať exaktnejšie nasledujúcim spôsobom [6].

```
1.  Inicializuj t_m = 0, t_M = 1;
2.  Pre každý hraničný segment p_i
3.  { Spočítaj  s = (O_i − A, n_i), u = (B − A, n_i);
4.    if (u = 0)
5.    { if (s < 0) Koniec;}
6.    else
7.    { t = s/u;
8.      if (u < 0)  t_m = max{t, t_m};
9.      else        t_M = min{t, t_M};
10.   }
11.   if (t_m > t_M) Koniec;
12. }
13. Krajné body úsečky sú  A' = A + t_m (B − A), B' = A + t_M (B − A);
14. Koniec;
```

Krok 11. uvedeného algoritmu indikuje, že úsečka AB oknom neprechádza. Necháva sa čitateľovi nájsť geometrickú interpretáciu tohto faktu, a taktiež dôkladne analyzovať prípad $u = 0$ (kroky 4-5).

Pre algoritmus je podstatné, že

1. hraničné priamky sú dané bodom a vonkajšou normálou,
2. úsečka je daná parametricky,
3. pre rozhodovanie je použitý skalárny súčin.

Použitý formálny aparát dovoľuje veľmi jednoduchým spôsobom zovšeobecniť tento algoritmus i na 3D situácie, tj. orezanie úsečky konvexným mnohostenom. Okno je v tomto prípade určené ako prienik polpriestorov. Bod O_i a vonkajšia normála v ňom $\vec{n}_i$ definujú príslušnú hraničnú stenu.

7.2.5 Vzťah C-S a C-B algoritmov

Spoločným rysom algoritmov C-S a C-B je postupné projektovanie koncových bodov úsečky na hraničné priamky okna a súčasne realizovaná analýza výsledného prieniku. Pritom

- C-S projektuje najprv jeden a potom druhý koncový bod (cyklus prebieha po koncových bodoch úsečky),
- z druhej strany – C-B v priebehu orezávania aktualizuje polohu oboch koncových bodov (cyklus prebieha po elementoch hranice orezávacieho okna),
- C-B je formulovaný pre všeobecnú formu orezávacieho konvexného m-uholníkového okna. Pre C-S je toto taktiež možné, aj keď bežne sa C-S formuluje len pre ortogonálne okno.
- Oba algoritmy možno formulovať i v 3D, tj. orezávanie úsečky konvexným mnohostenom.

Niekedy sa uvádza, že C-B algoritmus realizuje orezávanie priamky. Podľa nášho názoru toto nie je správne. V kroku 1. totiž začíname krajnými bodmi úsečky, a i pri iných nastaveniach inicializačných hodnôt t_m, t_M *vždy orezávame úsečku.*

Pozrieme sa na problematiku orezávania ešte iným spôsobom, a ukážeme algoritmus, ktorý *skutočne orezáva priamku.*

7.2.6 Rýchle orezávanie priamky

Chceme nájsť efektívny algoritmus na orezávanie priamky m-uholníkovým oknom. Použijeme podobnú myšlienku, ako v prípade algoritmu C-S, kde sme „lacno“ eliminovali tie úsečky, ktoré boli „z jednej strany okna“.

Teraz budeme toto aplikovať duálnym spôsobom, tj. budeme sa pozerať, či okno ktorým orezávame, je celé na jednej strane orezávanej priamky $\overline{AB}$. V kladnom prípade prienik priamky a okna neexistuje, v opačnom prípade hľadáme priesečník na tej hraničnej úsečke, ktorej koncové body sú v opačných polrovinách vytýčenách priamkou $\overline{AB}$.

Uvažujme implicitné vyjadrenie orezávanej priamky

$$\phi(P) = \phi(x, y) = ax + by + c = 0.$$

Použijeme klasifikáciu

$$\begin{aligned} \phi(P) = 0 &\Leftrightarrow P \text{ je na priamke,} \\ \phi(P) < 0 &\Leftrightarrow P \text{ je v jednej polrovine mimo priamku,} \\ \phi(P) > 0 &\Leftrightarrow P \text{ je v druhej polrovine.} \end{aligned}$$

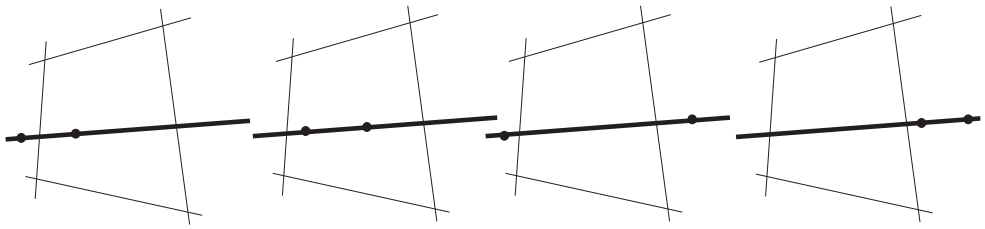
Priesečník priamky s oknom hľadáme na len tých častiach hranice O_i, O_{i+1}, kde platí

$$\phi\left(O_i\right).\phi\left(O_{i+1}\right) < 0, \tag{7.8}$$

tj. tam. kde susedné vrcholy okna sú na rôznych stranách orezávanej priamky.

1. Nájdi dvojice bodov $O_i, O_{i+1}, \quad O_j, O_{j+1}$ kde priamka $\overline{AB}$ pretína okno. Ak také dvojice neexistujú, koniec.
2. Nájdi prienik priamky $P_i = \overline{AB} \cap p_i$.
3. Nájdi prienik priamky $P_j = \overline{AB} \cap p_j$.
4. Kresli úsečku $P_i P_j$. Koniec.
5. Koniec.

Takto koncipovaný algoritmu je rýchly, pretože **1.** test (7.8) je jednoduchý a **2.** nezávisle na počte vrcholov hraničného mnohouholníka algoritmus hľadá maximálne dva priesečníky. Tento prístup dovoľuje orezávať priamku i nekonvexným m-uholníkovým oknom – [22]. No bohužiaľ v 3D nie sme schopni vyjadriť priamku implicitne jedným vzťahom $\phi\left(P\right) = 0$ (čo potrebujeme v 1. kroku algoritmu pri analýze vzťahu (7.8)). Preto zovšeobecnenie tohto postupu do 3D nie je také priamočiare ako v prípade algoritmov C-S a C-B.



Obr. 7.6: Orezanie priamky a orezanie úsečky. Jednému orezaniu priamky odpovedajú rôzne výsledky orezania úsečky v závislosti na vzájomnej polohe koncových bodov.

7.2.7 Ďalší spôsob orezania úsečky

Obr. 7.6 demonštruje, že orezanie priamky je principiálne jednoduchšie, ako orezanie úsečky. Nižšie uvedený algoritmus, vychádzajúci z orezania priamky, orezáva úsečky podstatne efektívnejšie ako C-S, alebo C-B.

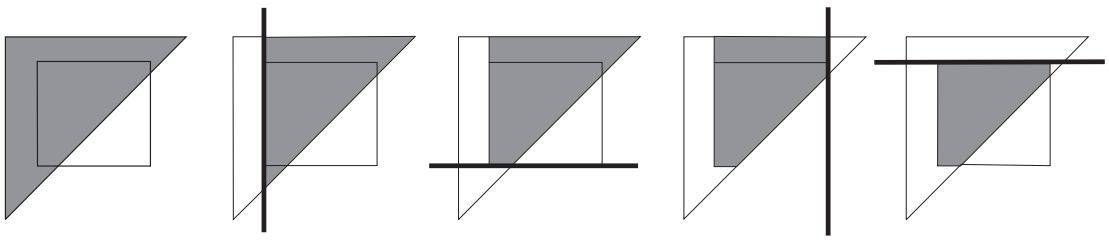
1. Nájdi dvojice bodov $O_i, O_{i+1}, \quad O_j, O_{j+1}$
   (tj. hraničné priamky $p_i, \quad p_j$, ktoré určujú,
   kde priamka $\overline{AB}$ pretína okno).
   Ak také dvojice neexistujú, koniec.
2. Urči pozíciu krajných bodov A, B k polrovinám π_i, π_j.

3. Ak $A, B \notin \pi_i$ koniec.
4. Ak $A, B \notin \pi_j$ koniec.
5. Ak $A \notin \pi_i$, najdi projekciu $A = \vec{AB} \cap \pi_i$.
6. Ak $A \notin \pi_j$, najdi projekciu $A = \vec{AB} \cap \pi_j$.
7. Ak $B \notin \pi_i$, najdi projekciu $B = \vec{BA} \cap \pi_i$.
8. Ak $B \notin \pi_j$, najdi projekciu $B = \vec{BA} \cap \pi_j$.
7. Kresli AB.
8. Koniec.

Kroky 1. a 2. su rýchle, v prípade celočíselných súradníc bodov vyžadujú len celočíselné operácie. Z dvojíce krokov 5.-6. sa realizuje maximálne jeden. Podobne pre kroky 7.-8. Celkovo sa teda realizujú maximálne dve operácie projekcie. Naviac, v prípade, že úsečka oknom neprechádza, nevypočítava sa žiadna projekcia.

7.3 Orezávanie m-uholníka

Pri orezávaní m-uholníka konvexným oknom môžeme organizovať algoritmus taktiež dvojakým spôsobom. Tj. analýza môže prebiehať buď po elementoch hranice orezávaného objektu, alebo po elementoch hranice orezávacieho okna (viď C-S vs. C-B). Jednoduchšie je v tomto prípade orezávať tak, že *cyklus prebieha postupne po hraničných polrovinách* (podobne ako C-B), ktoré určujú orezávacie okno.



Obr. 7.7: Postupné orezávanie trojuholníka polorovinami, určujúcimi okno.

Pritom v každom kroku hľadáme prienik mnohouholníka s hraničnou priamkou. Preto sa ponúka použiť postup rýchleho orezávania priamky (pre každú hraničnú priamku analyzujeme polohu susedných vrcholov orezávaného mnohouholníka). Tento algoritmus je známy pod názvom Sutherland-Hodgmanov algoritmus [27].

Algoritmus nevyžaduje, aby orezávaný m-uholník bol konvexný.

7.4 Zhrnutie a úlohy

Popísali sme metódy orezávania úsečky a mnohouholníka konvexným oknom.

- Sformulovali sme algoritmus orezávania izolovaného bodu.
- Sformulovali sme triviálny algoritmus orezávania úsečky.
- Popísali sme princípy, na základe ktorých je možné algoritmy zefektívniť.
- Podrobne sme odvodili dva základné algoritmy orezávania úsečky: Co-Su, Cy-Be a ukázali ich vzájomnú súvislosť.
- Sformulovali sme rýchly algoritmus orezávania priamky a ukázali jeho využitie pre urýchlenie orezávania úsečky.
- Demonštrovali sme algoritmus orezávania mnohouholníka konvexným oknom.

Kontrolné otázky:

1. Kde sa v algoritmoch využíva podmienka konvexity orezávacieho okna?
2. Koľko operácií delenia v pohyblivej rádovej čiarke vyžadujú použité algoritmy?
3. Na akých úsečkách sa prejaví väčšia efektivita algoritmu Co-Su oproti triviálnemu orezávaniu?
4. V čom je algoritmus Co-Su výhodnejší oproti algoritmu Cy-Be?
5. Pri orezávaní n-uholníka konvexným oknom sú (nie sú) kladené podmienky konvexity na n-uholník? Prečo?
6. Orezávaním n-uholníka m-uholníkovým konvexným oknom je výsledkom k-uholník. Akú maximálnu hodnotu može nadobúdať k?

Samostatné úlohy:

1. Realizujte aspoň jeden algoritmus orezávania úsečky. Vykreslite neorezanú úsečku a orezanú úsečku (inou farbou). Výsledok komentujte.

Kapitola 8

Vyplňovacie algoritmy

Okrem kreslenia čiar vyžadujeme od grafického systému schopnosť vyplniť požadovanú oblasť vopred určenou farbou. Predpokladajme, že v obdĺžniku máme zadanú uzavretú spojitú krivku, ktorá sama seba nepretína. Taká krivka delí obdĺžnik na dve časti. Každú z nich preto jednoznačne určuje *hranica* a ešte aspoň *jeden vnútorný bod* (inicializačný bod vyplňovania). Namiesto inicializačného bodu sa môže využiť fakt, že priamka ktorá prechádza oblasťou, pretína jej hranicu párny počet krát (s výnimkou prípadov, keď sa priamka oblasti len dotýka).

Problém vyplňovania oblasti zahrnuje dva rôzne typy úloh:

1. zároveň s rasterizáciou hranice oblasti vyplňujeme aj vnútorné pixely oblasti,
2. vyplňujeme všetky vnútorné pixely oblasti pri vopred rasterizovanej hranici.

V oboch prípadoch šrafujeme danú oblasť horizontálnymi čiarami hrúbky 1 pixel, ktoré sú susedné. To zabezpečí, že pri rasterizácii tieto čiary súvisle vyplnia oblasť.

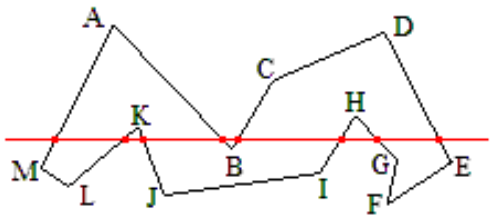
8.1 Vyplňovanie ako rasterizačný proces

Budeme predpokladať, že oblasť je určena uzavretou lineárne lomenou čiarou. Využijeme fakt, že priamka ktorá prechádza oblasťou, pretne hranicu párny počet krát (s výnimkou prípadov, keď sa priamka hranice len dotýka).

8.1.1 Scan-Line algoritmus

Základná myšlienka je veľmi jednoduchá: použijeme horizontálne čiary (scan-lines), ktoré pretínajú hranicu oblasti[1]. Pri každom nepárnom priesečniku sa dostávame do vnútra oblasti a pri každom párnom vychádzame z oblasti von.

[1] Vo výpočtovej geometrii sa táto priamka nazýva „zametacia“ (sweepline).

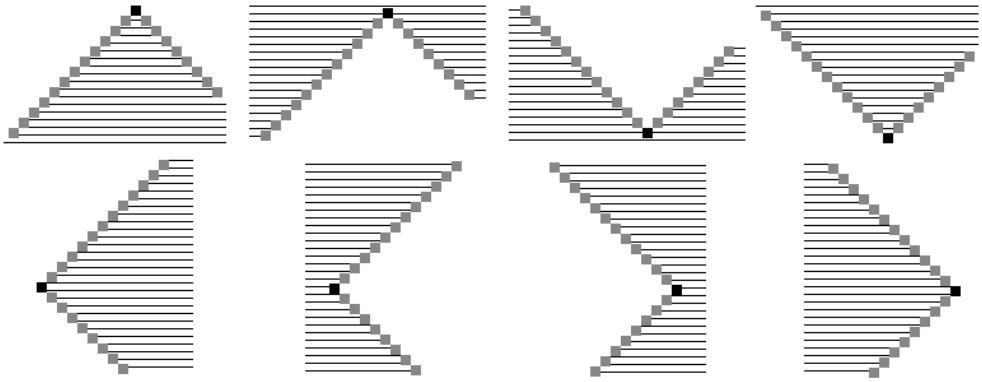


Obr. 8.1: Hlavná myšlienka algoritmu Scan-Line. Analyzovaná oblasť je ohraničená lineárne lomenou čiarou (čiernou), zametacia priamka a jej priesečníky s hranicou sú červené.

Algoritmus môžeme formulovať nasledujúco:

```
Pre všetky možné zametacie priamky vykonaj 1 - 3:
1. nájdi priesečníky zametacej priamky a hraničných úsečiek oblasti,
2. usporiadaj priesečníky z ľava do prava,
3. vyplňuj medzi nepárnym a párnym priesečníkom.
```

Takýto algoritmus ale nebude korektne pracovať na tých polohách zametacej priamky, ktoré prechádzajú vrcholami vyplňovaného polygónu. Konfigurácie všetkých možných polôh susedných úsečiek vzhľadom na horizontálny smer zametania sú na na obr. 8.2.



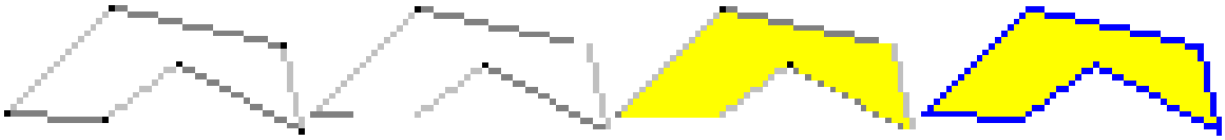
Obr. 8.2: Konfigurácie susedných hraničných úsečiek vzhľadom na horizontálne zametanie. Dvojnásobný priesečník zametacej priamky a hraničnej úsečky je čierny. Prvé štyri konfigurácie (horný riadok) algoritmus spracuje správne, no na konfiguráciách z dolného riadku algoritmus nebude fungovať .

Vychádzajúc z toho, že dve susedné zametacie priamky sú v podstate dotýkajúce sa obdĺžniky, riešenie tohto problému je jednoduché:

```
a) Pred aplikáciou algoritmu skrátime pri rasterizácii
   každú hraničnú líniu o poslednú úroveň[2].
b) Po skončení algoritmu znovu vykreslíme celú hranicu.
```

[2]Tým eliminujeme všetky nepriaznivé dvojnásobné body.

Proces vyplňovania ilustruje príklad z obr. 8.3



a) b) c) d)

Obr. 8.3: Scan-line vyplňovanie. a) Hranica s vyznačením dvojnásobných hraničných bodov (čierne). b) Skrátenie každej hraničnej úsečky o spodnú líniu. Vidíme, že zostali len prvé dve konfigurácie (obr. 8.2) dvojnásobných vrcholov. c) Vyplnenie vnútra oblasti. Hraničné pixely nemeníme. Za hraničný pixel úsečky na danej scan-line považujeme vždy ten pixel, ktorý je najviac vľavo. d) Záverečné vykreslenie hranice. (Pre názornosť sme zvolili farbu hranice rôznu od farby vnútra.

8.1.2 Optimalizácia Scan-Line algoritmu

Kroky 1. a 2. uvedeného algoritmu sa dajú realizovať efektívne, vychádzajúc z toho, že

- zametacia priamka pretína len niektoré hraničné úsečky (nazveme ich aktívne hrany),
- presunom zametacej priamky na nasledujúcu úroveň sa priesečik $P = (x, y)$ s aktívnou hranou zmení, tj. aktualizuje na $Q = (x + 1/a, y + 1)$, kde a je smernica danej aktívnej hrany,
- aktualizácia existujúcich priesečíkov nezmení ich vzájomné usporiadanie.

Tomuto javu hovoríme *koherencia*. Problém sa tak skoncentruje na analýzu len tých prípadov, keď zmenou zametacej priamky sa zmení množina aktívnych hrán tj. tam, kde sa koherencia poruší. Ináč povedané,

```
1*. na začiatku procesu vyplňovania urči množinu aktívnych hrán,
2*. generuj zametaciu priamku,
3*. aplikuj kroky 1. a 2. Scan-Line algoritmu,
4*. aplikuj krok 3. Scan-Line algoritmu,
5*. ak nie si na konci, prejdi na nasledujúcu zametaciu priamku,
6*. kontroluj koherenciu:
      keď je zachovaná - aktualizuj priesečníky a pokračuj krokom 4*.,
      keď nie je zachovaná - aktualizuj množinu aktívnych hrán
                             a pokračuj krokom 3*.
```

Nájdenie priesečníka zametacej priamky a hraničnej priamky vyžaduje prvoplánovo použitie „reálnej" aritmetiky. Môžeme však použiť tú istú myšlienku ako pri generovaní úsečky

s použitím len celočíselnej aritmetiky: označme vrcholy, definujúce hraničný n-uholník $A_1, A_2, \cdots, A_n$ a definujme naviac vrchol $A_{n+1} = A_1$, kde $A_i = (x_i, y_i)$. Predpokladajme pritom, že uvedené body určujú orientovanú hranicu v smere pohybu hodinových ručičiek. i-tu hraničnú úsečku môžeme vyjadriť v explicitnom tvare

$$\varphi_i(P) = \varphi_i(x, y) = (y_{i+1} - y_i)\, x - (x_{i+1} - x_i)\, y + x_{i+1}y_i - x_{i+1}y_i = 0, \tag{8.1}$$

kde hodnoty x sú medzi x_i a x_{i+1}, a hodnoty y medzi y_i a y_{i+1}. Vzhľadom na orientáciu hranice, body vľavo od hraničnej priamky sú body mimo vyplňovanú oblasť – v tom prípade $\varphi_i(x, y) > 0$. Body vpravo od hraničnej priamky sú vo vnútri vyplňovanej oblasti – v tom prípade $\varphi_i(x, y) < 0$. Keď máme spočítanú hodnotu príznaku $\varphi_i(P_k) = \varphi_i(x, y)$, tak výpočet príznaku pre nasledujúci (susedný) pixel zametacej priamky je veľmi jednoduchý (využijeme rekurenciu):

$$\begin{aligned}
\varphi_i(P_{k+1}) &= \varphi_i(x+1, y) = (y_{i+1} - y_i)(x+1) - (x_{i+1} - x_i)\, y + x_{i+1}y_i - x_{i+1}y_i = \\
&= (y_{i+1} - y_i)\, x - (x_{i+1} - x_i)\, y + x_{i+1}y_i - x_{i+1}y_i + (y_{i+1} - y_i) = \\
&= \varphi_i(P_k) + (y_{i+1} - y_i)
\end{aligned}$$

Dostávame tak celočíselný príznak, ktorý sa jednoducho vypočítava a určuje, či daný pixel je vzhľadom k hraničnej úsečke vnútorný (tj. ten. ktorý potrebujeme vyplniť požadovanou farbou). Toto je podstata algoritmu, ktorý v r. 1988 publikoval J. Pineda.

8.1.3 Inverzné vyplňovanie (Inverse-Fill)

Pri definovaní farby pre daný pixel sa často využíva režim `XORPut(a,b)`, kde `a` je hodnota pôvodnej farby, `b` je vkladaná farba. Tento režim pracuje tak, že odpovedajúce si bity oboch čísel sčítava `mod 2`. Napr. `XORPut(7,5)`$= 111 \oplus 110 = 001$. Nie je ťažké sa presvedčiť, že $a \oplus b \oplus b = a$, čo znamená, že dvojnásobné prekreslenie pixelu tou istou farbou b na pozadie farby a, zmení výslednú farbu na pôvodnú farbu pozadia.

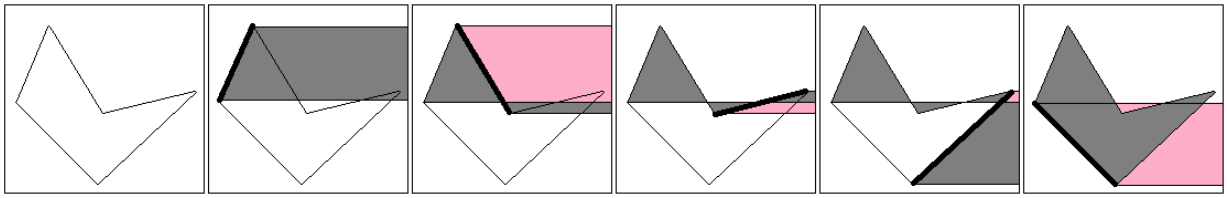
Významnou skutočnosťou je, že pri vykresľovaní izolovaného pixelu je časovo najnáročnejšia samotná alokácia pixelu. Z toho pohľadu je napr. vyplnenie jedného pixelu so súradnicami (x, y) zrovnateľné s vyplnením horizontálnej úsečky $(x, y) - (x_{MAX}, y)$.

Preto, z pohľadu rýchlosti, je nakreslenie úsečky AB zrovnateľné s vyplnením lichobežníka $ABCD$, kde

$$A = (x_A, y_A)\,,\ B = (x_B, y_B)\ C = (x_{MAX}, y_B)\,,\ D = (x_{MAX}, y_A)\,. \tag{8.2}$$

Toto dovoľuje realizovať vyplňovanie polygonálnej oblasti tak, že namiesto každej hraničnej úsečky AB kreslíme lichobežník $ABCD$ (8.2) v režime `XORPut`. Hraničné úsečky musíme, podobne ako u Scan-Line algoritmu, pred vyplňovaním skrátiť o spodnú úroveň.

To najpodstatnejšie, čo oba algoritmy odlišuje, je organizácia cyklov. Kým Scan-Line je vo forme



Obr. 8.4: Ilustrácia vyplňovania nmohouholníka metódou Inverse-fill. Zvýraznená hraničná úsečka indikuje aktuálne vyplňovaný pravý lichobežník. Ružová vyjadruje vynulovanie farby pre danú úsečku na základe dvojnásobného vyplnenia.

```
for (horizontalne_ciary)
{ for (hranicne_usecky)
  { fill_1}
}
```

Inverse-Fill má cykly prehodené:

```
for (hranicne_usecky)
{ for (horizontalne_ciary)
  { fill_2}
}
```

Nasledujúci rozdiel je v zložitosti tela cyklov: kým v prvom prípade `fill_1` musí nájsť priesečníky hranice s vyplňovacími líniami a vyplňovať medzi párnym a nepárnym priesečníkom, v druhom prípade `fill_2` je toto automaticky zabezpečené použitím režimu `XORPut`.

Významnou vlastnosťou algoritmu Inverse-Fill je jeho jednoduchá paralelizácia, vyplývajúca z toho, že vyplnenie pravého lichobežníka jednej hraničnej úsečky nezávisí na žiadnej ďalšej hraničnej úsečke.

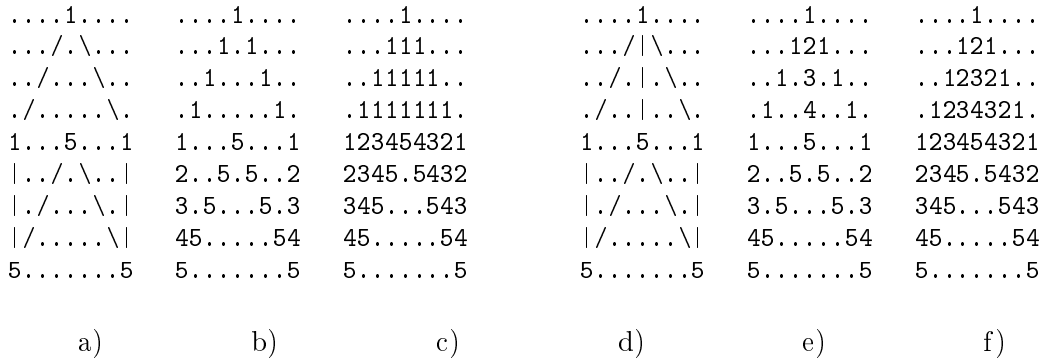
8.1.4 Zovšeobecnené vyplňovanie

Okrem vyššieopísaného problému vyplniť oblasť konštantnou hodnotou, často vzniká úloha interpolovať hodnoty z hranice do vnútra oblasti.

Nech okrem vrcholov vyplňovaného n-uholníka je v každom vrchole zadaná i číselná hodnota. Potom pre interpoláciu hodnôt do vnútra možno Scan-Line modifikovať:

```
(i)  Pri hľadaní polohy priesečníka hraničnej úsečky
     a scan-line, interpoluj súčasne aj číselnú
     hodnotu krajných bodov danej hraničnej úsečky.
(ii) Každý vyplňovaný úsek vyplň hodnotami, ktoré
     sú interpoláciou hodnôt príslušných priesečníkov.
```

Štandardný Scan-Line funguje na ľubovoľnej, tj. i nekonvexnej oblasti bez problémov. Modifikovaný algoritmus pre nekonvexné oblasti sa ale môže správať nie úplne podľa našich predstáv – obr. 8.5a)–c). Problém je možné riešiť rozkladom oblasti na konvexné časti a ich následným vyplnením – obr. 8.5d)–f).



Obr. 8.5: Zovšeobecnené vyplňovanie nekonvexného 6-uholníka. a) Oblasť s vyznačenými hodnotami vo vrcholoch. b) Interpolácia hodnôt na hranici. c) Interpolácia vo vnútri oblasti. Vidíme, že na 4. a 5. riadku výsledok nezodpovedá nášnu očakávaniu – nastáva tu narušenie spojitosti. d) Rozklad oblasti na konvexné podoblasti. Dodaný hraničný segment je dvojnásobný (pre pravý a pre ľavý 4-uholník). e) Interpolácia na hranici. f) Interpolácia vo vnútri oblastí.

Zovšeobecnené vyplňovanie ukazuje podstatný rozdiel medzi Scan-Line a Inverse-Fill: Inverse-Fill sa pre tento prípad použiť nedá: v lichobežníku $ABCD$ (8.2) nemôžeme odvodiť skalárnu hodnotu vo vrcholoch C, D len z hodnôt vo vrcholoch A, B.

8.1.5 Vyplňovanie konvexných oblastí

Uvedený príklad zovšeobecneného vyplňovania ukazuje dôležitú triedu vyplňovaných oblastí – konvexné mnohouholníky. V tomto prípade sa Scan-Line algoritmus podstatne zjednoduší: pre každú zametaciu priamku existujú práve dve aktívne hrany, a preto druhý krok Scan-Line algoritmu – preusporiadanie hraníc vyplňovania je triviálny a posledný – tretí krok zdegeneruje do prostého vyplnenia medzi dvojicou hraničných pixelov.

Keď sa obmedzíme na základnú triedu vyplňovaných oblastí – trojuholníky, algoritmus sa zjednoduší ešte viac. V procese vyplňovania nastáva zmena koherencie len jedenkrát: práve jedna aktívna hrana sa mení na inú.

Efektívny algoritmus pri znalosti orientovanej hranice je Pinedov algoritmus:

```
(i)   Pre každú hraničnú priamku (P1,P2) testujeme, či
      vyplňovaný bod P(x,y) je vpravo od hranice.
(ii)  Pre tento test použijeme výpočet vektorového súčinu
        V(x,y)=(P2-P1)X(P-P1)=dx*(y-y1)-dy*(x-x1)
```

```
(iii) Pre susedný bod je výpočet vektorového súčinu
      inkrementálny:
        V(x+1,y)=V(x,y)-dy
```

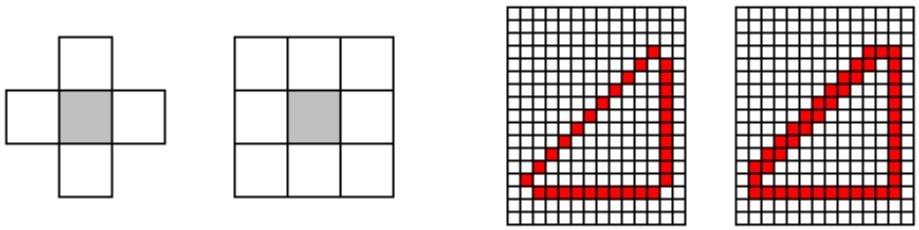
Špeciálne prípady, ktoré stojí za to riešiť samostatne, sú napr. rasterizácia pravouholníka, rasterizácia trojuholníka.

8.2 Rastrové vyplňovanie

Teraz predpokladajme, že hranica oblasti je už rasterizovaná a že máme k dispozícii aspoň jeden vnútorný pixel oblasti.

Vyplňovanie začneme inicializačným pixelom a ďalej postupujeme na základe analýzy okolia už vyplnenej oblasti.

Dôležitú úlohu má práve *okolie* pixelu. V závislosti na tom, aké okolie uvažujeme, tj. ktoré pixely považujeme za susedné, vyplýva, čo je a čo nie je elementárna hranica. Pre bežne používanú rastrovú mriežku využívame 4-okolie a 8-okolie pixelu – obr. 8.6 a), b). V prípade rastrovej mriežky má vnútorný pixel oblasti vo svojom okolí len vnútorné a hraničné pixely; v žiadnom prípade nemôže mať vo svojom okolí vonkajší pixel oblasti – obr. 8.6 c), d).



Obr. 8.6: Okolie pixlu v štandardnom rastri. a) 4-okolie, b) 8-okolie a ukážka najtenšej možnej hranice trojuholníka pre c) 4-okolie, d) 8-okolie.

8.2.1 Semienkove vyplňovanie (seed fill)

Riešenie problému sa dá jednoducho sformulovať pomocou rekurzie. Napr. pre 4-okolie, vyplňovaciu farbu `c_i` a farbu hranice `c_b` dostávame:

```
PROCEDURA Vypln(x, y, c_i, c_b)
1. { Ak ((P(x,y)≠c_i) a zároveň (P(x,y)≠c_b))
2.   { P(x,y) = c_i;
```

```
3.      Vypln(x-1,y, c_i, c_b);
4.      Vypln(x+1,y, c_i, c_b);
5.      Vypln(x,y-1, c_i, c_b);
6.      Vypln(x,y+1, c_i, c_b);
7.    }
8. }
```

V každom rekurzívnom vnorení si pamätáme všetky možné smery pokračovania, preto už pri pomerne malých oblastiach je procedúra je prakticky nepoužiteľná. Preto musíme zredukovať informáciu o možných smeroch pokračovania.

8.2.2 Riadkovo semienkové vyplňovanie

Postup sa dá zapísať pre uvažované 4-okolie nasledujúco:

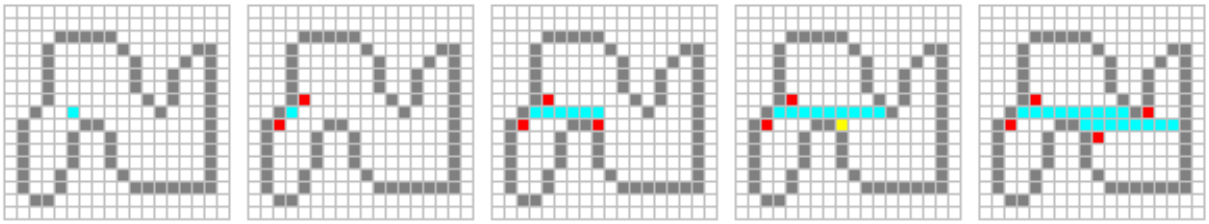
1. Pre zvolený pixel postupujeme na tom istom riadku smerom vľavo, kým nenarazíme na hranicu; pravý susedný pixel je tak prvý vnútorný pixel segmentu.
2. Od prvého vnútorného pixelu vyplňujeme všetky pixely smerom do prava, až kým nenarazíme na pravú hranicu segmentu.
3. Pri vyplňovaní každého pixelu daného segmentu analyzujeme susedné pixely pod a nad vyplňovaným segmentom s cieľom nájsť kritické pixely.
4. Kritické pixely uchováme.
5. Po dosiahnutí hraničného pixelu na riadku zvolíme pre ďalší postup niektorý z kritických pixelov (napr. posledne vytvorený).

Pritom vkladaný kritický pixel je taký, ktorý súčasne spĺňa nasledujúce tri podmienky:

a) je vnútorný pixel oblasti,
b) je nevyplnený,
c) je na riadku, ktorý je "nad" alebo "pod" aktuálne vyplňovaným segmentom a vľavo od neho je hraničný pixel.

Proces začneme inicializačným pixelom a kroky 1.– 5. vykonávame dovtedy, kým existujú kritické pixely. Obr. 8.7 demonštruje proces vyplňovania.

Vidíme, že v tomto prípade potrebujeme v priebehu celého vyplňovania uchovávať len maximálne 4 kritické pixely.



a) b) c) d) e)

Obr. 8.7: Riadkovo semienkové vyplňovanie. a) Inicializačný pixel. b) 1. krok. algoritmu – prechod na začiatok segmentu a označenie kritických pixelov (červené). c) Vyplňovanie segmentu s nájdením ďalšieho kritického pixelu. d) Dokončenie vyplňovania segmentu a výber pokračovacieho kritického pixelu (žltý). e) Pokračovanie vyplňovania ďalšieho segmentu s nájdením kritických pixelov.

8.2.3 Rastrové vyplňovanie konvexných oblastí

Keď vopred vieme, že vyplňovaná oblasť je konvexná, vyplňovací algoritmus sa výrazne zjednoduší. V celom procese vyplňovania si vystačíme maximálne s dvoma kritickými pixelami. Keď v kroku 3. algoritmu nájdeme nad resp. pod vyplňovaným segmentom kritický pixel, môžeme vyplňovať do konca riadku bez hľadania ďalších kritických pixelov.

8.3 Zhrnutie a úlohy

Ukázali sme dva prístupy pri riešení úlohy vyplňovania oblastí

- vyplňovanie ako súčasť rasterizačného procesu,
- vyplňovanie ako úloha rastrovej grafiky.

Kontrolné otázky:

1. Prečo nie je možné použiť algoritmus inverzného vyplňovania pre zovšeobecnené vyplňovanie?
2. Je možné uplatniť princíp zovšeobecneného vyplňovania pre algoritmy semienkového vyplňovania a riadkovo semienkového vyplňovania?
3. Načrtnite oblasť, pri vyplňovaní ktorej metódou riadkovo semienkového vyplňovania, vznikne 5 kritických pixelov.

Samostatné úlohy:

1. Modifikujte algoritmus kreslenia úsečky na inverzné vyplňovanie.

Kapitola 9

Transformácie v rovine a priestore

Pri zobrazovaní objektov zadaných vektorovo vzniká prirodzená potreba ich posúvať, otáčať, zväčšovať resp. zmenšovať. Samotný tvar objektu sa pri tom nemení, preto takéto transformácie nazývame *tuhé*. V prípade 3D telies musíme ešte vedieť spraviť projekciu telesa na projekčnú rovinu. Tieto transformácie musíme vykonať so všetkými vrcholami, ktoré daný objekt určujú.

I teraz nás samozrejme zaujíma rýchlosť operácií. No v tomto prípade sa reálnej aritmetike nevyhneme. Rýchlosť sa dosahuje iným spôsobom – ukážeme, že tuhé transformácie a projekcie sa dajú sformulovať výhradne na základe násobenia matíc. A násobenie matíc je hardverovo podporované. Pritom sa pri výpočtoch využíva jednoduchý paralelizmus.

9.1 Súradnicové systémy a súradnice

Pre určenie vzájomnej polohy bodov postupujeme spravidla tak, že v uvažovanom priestore zvolíme 1. referenčný bod O, tj. *začiatok súradníc*, 2. *vybrané smery*, 3. *jednotkový etalón*.

V prípade *kartézskych súradníc*, zvolíme vzájomne kolmé smery (v rovine dva, v priestore tri) a súradnice predstavujú kolmé projekcie[1] na tieto zvolené smery vzhľadom k začiatku súradníc O.

V prípade *polárnych súradníc* (v rovine) zvolíme referenčný smer (polpriamka vychádzajúca z O) a polohu bodu P definujeme usporiadanou dvojicou (r, φ) kde r je vzdialenosť bodu P od O a φ je uhol, ktorý zviera polpriamka $\overrightarrow{OP}$ s referenčným smerom.

V trojrozmernom priestore sa často používajú dva rôzne analógy polárnych súradníc. *Cylindrické súradnice* sú kombináciou kartézskych a polárnych súradníc. Bod je v tomto prípade určený usporiadanou trojicou (r, φ, z) kde (r, φ) sú polárne súradnice priemetu bodu P do roviny $z = 0$ a z je kartézska súradnica v smere kolmom na rovinu $z = 0$.

[1] Pripomeňme si, že pre vzájomnú projekciu vektorov $\vec{u}\,\vec{v}$ používame ich skalárny súčin, tj. číslo $(\vec{u}, \vec{v}) = u_x v_x + u_y v_y + u_z v_z$. Pritom platí, že $(\vec{u}, \vec{v}) = |\vec{u}| \,.\, |\vec{v}| \,.\, \cos\omega$, kde ω je uhol, ktorý vektory zvierajú.

Sférické súradnice (r, φ, ϕ) sú priamym analógom súradníc polárnych. (r, φ) sú, rovnako ako v prípade cylindrických súradníc, polárne súradnice projekcie bodu do roviny $z = 0$ a ϕ je uhol medzi polpriamkou $\overrightarrow{OP}$ a normálou k rovine $z = 0$.

Pre konverziu z polárnych a cylindrických súradníc do kartézskych platí

$$x = r\cos\varphi, \quad y = r\sin\varphi,$$

sférické súradnice konvertujeme do kartézskych pomocou vzťahov

$$x = r\cos\varphi\sin\phi, \quad y = r\sin\varphi\sin\phi, \quad z = r\cos\phi.$$

Pre korektné definovanie súradníc je nutné zaviesť *orientáciu*. V jednorozmernom prípade, tj. na priamke to znamená výber kladného smeru, resp. výber polpriamky kladných smerov. V dvojrozmernom prípade je orientácia daná tým, či uhol (v polárnych súradniciach) medzi kladnými polpriamkami meriame v smere, alebo proti smeru hodinových ručičiek.

Pre orientáciu v trojrozmernom priestore nám slúži konštrukcia zvaná *vektorový súčin* vektorov $\overrightarrow{u}$, $\overrightarrow{v}$,

$$\begin{aligned}\overrightarrow{w} &= \overrightarrow{u} \times \overrightarrow{v} = \begin{vmatrix} \overrightarrow{e_x} & \overrightarrow{e_y} & \overrightarrow{e_z} \\ u_x & u_y & u_z \\ v_x & v_y & v_z \end{vmatrix} = \overrightarrow{e_x}\begin{vmatrix} u_y & u_z \\ v_y & v_z \end{vmatrix} - \overrightarrow{e_y}\begin{vmatrix} u_x & u_z \\ v_x & v_z \end{vmatrix} + \overrightarrow{e_z}\begin{vmatrix} u_x & u_y \\ v_x & v_y \end{vmatrix} = \\ &= (u_y v_z - u_z v_y,\ u_z v_x - u_x v_z,\ u_x v_y - u_y v_x)\,.\end{aligned}$$

Vidíme, že je to konštrukcia pre výpočet determinantu s tým rozdielom, že prvý riadok matice neobsahuje čísla, ale jednotkové bázové vektory trojrozmerného priestoru.

Pre vopred zvolené poradie bázových vektorov, tj. pre usporiadanú trojicu $(\overrightarrow{e_x}, \overrightarrow{e_y}, \overrightarrow{e_z})$ platí

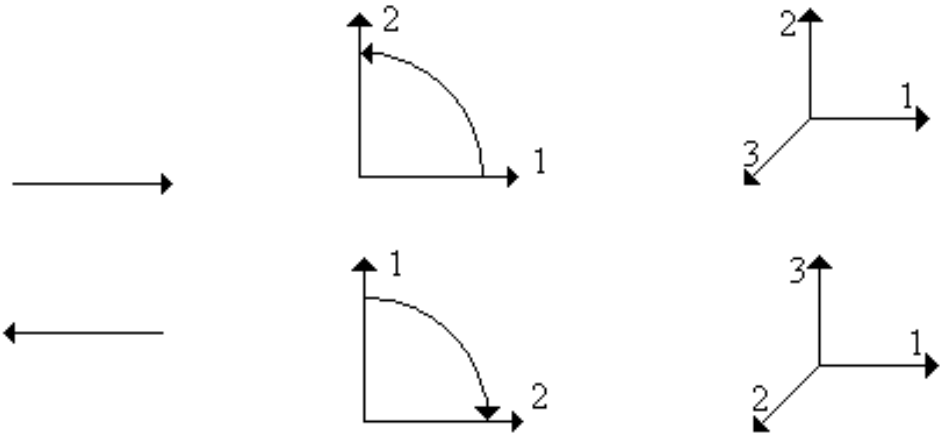
$$\overrightarrow{e_x} \times \overrightarrow{e_y} = \overrightarrow{e_z}, \quad \overrightarrow{e_y} \times \overrightarrow{e_z} = \overrightarrow{e_x}, \quad \overrightarrow{e_z} \times \overrightarrow{e_x} = \overrightarrow{e_y}.$$

Všimnime si, že $\overrightarrow{u} \times \overrightarrow{v} = -\overrightarrow{v} \times \overrightarrow{u}$, z čoho dostávame, že

$$\overrightarrow{e_y} \times \overrightarrow{e_x} = -\overrightarrow{e_z}, \quad \overrightarrow{e_z} \times \overrightarrow{e_y} = -\overrightarrow{e_x}, \quad \overrightarrow{e_x} \times \overrightarrow{e_z} = -\overrightarrow{e_y}.$$

Práve táto závislosť na poradí nám dovoľuje definovať orientáciu: usporiadanú trojicu vektorov $(\overrightarrow{u}, \overrightarrow{v}, \overrightarrow{u} \times \overrightarrow{v})$ nazývame pravotočivá trojica.

V trojrozmernom prípade môžeme pravotočívú trojicu vektorov demonštrovať pomocou palca, ukazováka a prostredníka pravej ruky tak, ako ukazuje obr. 9.1.



Obr. 9.1: Kladná (hore) a záporná (dole) orientácia a) na priamke b) v rovine, c) v priestore.

9.2 Transformácie v 2D

Uvažujme bod P so súradnicami (x, y). Označme bod, ktorý vznikne jeho transformáciou ako P' a jeho súradnice (x', y').

Posunutie o vektor $\vec{t} = (t_x, t_y)$ vyjadríme

$$\begin{array}{rcl} x' & = & x + t_x \\ y' & = & y + t_y \end{array} . \tag{9.1}$$

Pre *rotáciu* bodu P okolo začiatku súradnic o uhol α použijeme vyjadrenie bodu v polárnych súradniciach:

$$x = r\cos\varphi, \quad y = r\sin\varphi.$$

Pootočením o uhol α dostaneme

$$x' = r\cos(\varphi + \alpha), \quad y' = r\sin(\varphi + \alpha).$$

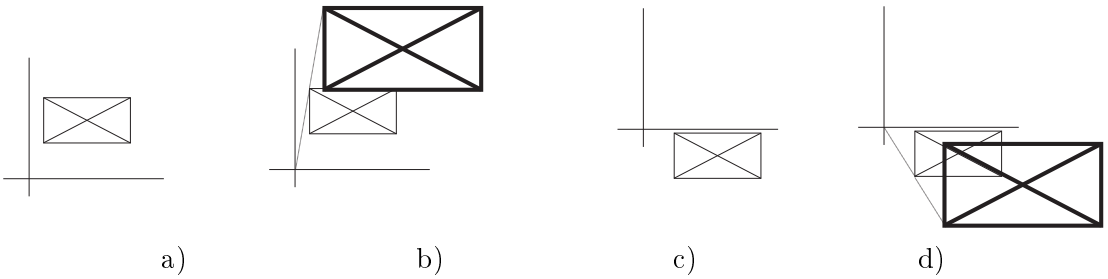
Po úpravách máme:

$$\begin{array}{rcccccl} x' & = & r.\cos(\varphi+\alpha) & = & r\cos\varphi.\cos\alpha - r\sin\varphi.\sin\alpha & = & x\cos\alpha - y\sin\alpha \\ y' & = & r.\sin(\varphi+\alpha) & = & r\cos\varphi.\sin\alpha + r\sin\varphi.\cos\alpha & = & x\sin\alpha + y\cos\alpha \end{array} . \tag{9.2}$$

Zmena mierky vzhľadom k začiatku súradníc vedie k jednoduchým vzťahom

$$\begin{array}{rcl} x' & = & s.x \\ y' & = & s.y \end{array} . \tag{9.3}$$

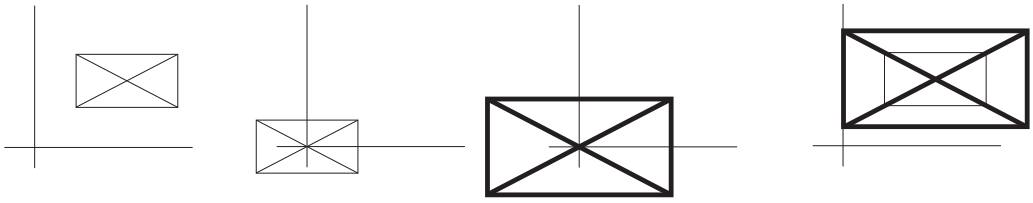
Ako vidno z obr. 9.2, pri preškálovaní objektu vzhľadom na začiatok súradníc (9.3) zároveň so zmenou jeho veľkosti nastáva i posun, ktorý je závislý na súradnicovom systéme.



Obr. 9.2: Zmena mierky vzhľadom k rôznym začiatkom súradníc: a) pôvodný stav, b) dvojnásobné zväčšenie vzhľadom k začiatku súradníc, c) pôvodný stav v inej sústave súradníc, d) dvojnásobné zväčšenie vzhľadom k začiatku súradníc.

Čím ďalej je objekt od začiatku súradníc, tým väčší posun vzniká. Jediný bod, ktorý pri preškálovaní nezmení svoju pozíciu (tj. *stacionárny bod*), je začiatok súradníc.

Aby sa objekt požadovaným spôsobom zmenšil (zväčšil) a pritom „zostal na mieste“[2], musíme ho najprv presunúť do blízkosti začiatku súradníc, potom objekt preškálujeme a takto zväčšený (zmenšený) objekt presunieme na pôvodné miesto – obr. 9.3.



Obr. 9.3: Kompozícia transformácií: a) východzí stav, b) posun stredu do začiatku súradníc, c) škálovanie vzhľadom k začiatku súradníc, d) posun stredu na pôvodné miesto.

Podobne pri rotácii (9.2) je stacionárnym bodom začiatok súradníc. Preto keď chceme otáčať teleso okolo jeho stredu, alebo iného zvoleného bodu a nie okolo začiatku súradníc, musíme postupovať tak, ako bolo vysvetlené vyššie: najprv posunieme teleso tak, aby sa stred otáčania dostal do začiatku súradníc, potom realizujeme rotáciu a na záver teleso presunieme na pôvodné miesto.

Vidíme, že v praxi je nutné realizovať spravidla vždy niekoľko transformácií. Preto je žiadúce použiť pre základné transformácie jednotné vyjadrenie.

9.2.1 Maticová reprezentácia transformácií

Rotáciu (9.2) a zmenu mierky (9.3) môžeme vyjadriť v maticovom tvare:

$$P' = \mathcal{R}(\alpha)\, P, \text{ resp. } P' = \mathcal{S}(s)\, P,$$

[2] V kontexte predošlého odseku na mieste zostane samozrejme iba jediný bod. Väčšinou chceme, aby to bol vnútorný bod objektu. Spravidla je to jeho ťažisko.

kde

$$P = \begin{pmatrix} x \\ y \end{pmatrix},\ P' = \begin{pmatrix} x' \\ y' \end{pmatrix}, \quad \mathcal{R}(\alpha) = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix},\ \mathcal{S}(s) = \begin{pmatrix} s & 0 \\ 0 & s \end{pmatrix}.$$

$\mathcal{R}(\alpha)$ a $\mathcal{S}(s)$ sú matice rotácie a škálovania[3]. Bohužiaľ posunutie (9.1) sa takto zapísať nedá. Aby sme to mohli spraviť, pomôžeme si nasledujúcou formálnou konštrukciou:

$$x' = x + t_x = (1 \quad t_x)\begin{pmatrix} x \\ 1 \end{pmatrix}, \quad y' = y + t_y = (1 \quad t_y)\begin{pmatrix} y \\ 1 \end{pmatrix}.$$

Toto môžeme združiť pre obe súradnice súčasne

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{pmatrix}\begin{pmatrix} x \\ y \\ 1 \end{pmatrix}.$$

Je rozumné mať rovnako vyjadrený pôvodný i transformovaný bod, preto výsledná forma translácie bude mať tvar

$$P' = \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \mathcal{T}\left(\vec{t}\right) P.$$

Rotačná a škálovacia matice majú v tejto reprezentácii tvar

$$\mathcal{R}(\alpha) = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \mathcal{S}(s) = \begin{pmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

9.2.2 Homogénne súradnice

Pre vyjadrenie bodu P použijeme ešte trochu všeobecnejší zápis – vyjadrenie v *homogénnych súradniciach.*

$$P = \begin{pmatrix} x \\ y \end{pmatrix} \longleftrightarrow w\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} wx \\ wy \\ w \end{pmatrix}, \quad w > 0. \tag{9.4}$$

Znamená to, že dvojrozmernému bodu s kartézskymi súradnicami (x, y) odpovedajú trojrozmerné body (wx, wy, w), $w > 0$, tj. polpriamka definovaná začiatkom súradníc a bodom

[3]Môžeme uvažovať i vzájomne nezávislé preškálovanie súradníc. V tom prípade je matica škálovania

$$\mathcal{S}(s_x, s_y) = \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix}.$$

$(x, y, 1)$. Naopak, bodu so súradnicami (X, Y, Z), $Z > 0$, vieme priradiť bod s kartézskymi súradnicami $\left(\frac{X}{Z}, \frac{Y}{Z}\right)$. Maticová reprezentácia uvedených tuhých transformácií zostáva očividne platná i pri použití homogénnych súradníc pre ľubovoľné $w > 0$.

Geometrická interpretácia vzťahu kartézskych a homogénnych súradníc je nasledujúca:

Uvažujme trojrozmerný priestor, v ňom rovinu $\rho: \ z = 1$ a bod R so súradnicami (X, Y, Z). Vedieme polpriamku p so začiatkom v začiatku súradníc $O = (0, 0, 0)$ a bodom R. Nájdeme priesečník $p \cap \rho$ so súradnicami $(x, y, 1)$. Vidíme, že medzi *bodom* P (kartézske súradnice) a *polpriamkou* p (homogénne súradnice) je vzájomne jednoznačné zobrazenie.

9.2.3 Skladanie transformácií

Pri skladaní niekoľkých transformácií násobíme každý bod P (tj. vektor) postupne príslušnou transformačnou maticou. Napr. pre transformácie s maticami $\mathcal{M}_1$, $\mathcal{M}_2$, $\mathcal{M}_3, \ldots \mathcal{M}_n$

$$P^{(1)} = \mathcal{M}_1 P, \quad P^{(2)} = \mathcal{M}_2 P^{(1)}, \quad P^{(3)} = \mathcal{M}_3 P^{(2)}, \ldots, P^{(n)} = \mathcal{M}_n P^{(n-1)}$$

po dosadení dostaneme

$$P^{(n)} = \mathcal{M}_n \left(\ldots \left(\mathcal{M}_2 \left(\mathcal{M}_1 P \right) \right) \ldots \right)$$

Využívajúc asociativitu násobenia matíc, vyššieuvedený vzťah môžeme prepísať do tvaru

$$P^{(n)} = \left(\mathcal{M}_n \ldots \left(\mathcal{M}_3 \left(\mathcal{M}_2 \mathcal{M}_1 \right) \right) \ldots \right) P. \tag{9.5}$$

Takto postupným prenásobením matíc elementárnych transformácií generujeme maticu celkovej transformácie.

Napr. matica rotácie $\mathcal{Q}$ o uhol α okolo bodu A so súradnicami (x_A, y_A) musí byť, ako bolo uvedené vyššie, realizovaná v troch krokoch:

1. posun bodu A do začiatku súradníc, $\quad \mathcal{M}_1 = \mathcal{T}(A)$,
2. rotácia okolo začiatku o uhol α, $\quad \mathcal{M}_2 = \mathcal{R}(\alpha)$,
3. spätný posun začiatku súradníc do bodu A, čo je ekvivalentné posunu bodu $-A$ do začiatku súradníc, $\quad \mathcal{M}_3 = \mathcal{T}(-A)$.

Matica $\mathcal{M}$ výslednej transformácie je

$$\mathcal{M} = \left(\mathcal{M}_3 \left(\mathcal{M}_2 \mathcal{M}_1 \right) \right) = \mathcal{T}(-A) . \mathcal{R}(\alpha) . \mathcal{T}(A)$$

9.3 Transformácie v 3D

Posunutie a zmena mierky majú podobné vyjadrenie ako v 2D:

$$\mathcal{T}\left(\vec{t}\right) = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}, \qquad \mathcal{S}\left(s\right) = \begin{pmatrix} s & 0 & 0 & 0 \\ 0 & s & 0 & 0 \\ 0 & 0 & s & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Všeobecná 3-rozmerná rotácia sa dá vyjadriť ako superpozícia troch elementárnych rotácií: v rovine xy, v rovine yz a v rovine zx.

$$\mathcal{R}\left(\alpha\right)_{xy} = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad \mathcal{R}\left(\alpha\right)_{yz} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

$$\mathcal{R}\left(\alpha\right)_{zx} = \begin{pmatrix} \cos\alpha & 0 & \sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

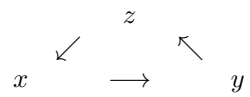
Zdanlivo nesprávne umiestnenie člena v poslednom vzťahu je dôsledkom nutnosti jednotnej orientácie rotácie v príslušných súradnicových rovinách – viď kap. 9.1[4].

9.4 Projekcia

Keďže výsledný obraz zobrazujeme na 2D výstupných zariadeniach, pri modelovaní a zobrazovaní 3D objektov sa objavuje nevyhnutnosť ich projekcie do 2D.

Plocha, na ktorú teleso premietame, sa nazýva *priemetňa*. *Premietací lúč* z bodu P je krivka, prechádzajúca týmto bodom a priemetňou. Bod, v ktorom premietací lúč pretne priemetňu nazývame *priemet* bodu P.

[4] Orientácia má tvar



(preto označenie $\mathcal{R}\left(\alpha\right)_{zx}$ a nie $\mathcal{R}\left(\alpha\right)_{xz}$). Keď v označení rotačných matíc nahradíme indexy x,y,z postupne indexami 1,2,3, vidíme, že prvý index matice odpovedá indexu riadku, na ktorom je $\sin\alpha$ so záporným znamienkom.

Obmedzíme sa len na najjednoduchšie prípady projekcií, keď priemetňou je rovina a premietací lúč je priamka. Pritom stačí uvažovať len jednu projekčnú rovinu, napr. $z = 0$ a tuhé transformácie projektovaného telesa. Pre vzájomnú polohu projektovaného telesa a priemetne je totiž jedno, či meníme polohu priemetne, alebo polohu projektovaného objektu. (Práve v tom je veľký rozdiel oproti tradičnému poňatiu projekcie v deskriptívnej geometrii, kde i pri jednom type projekčnej plochy a premietacích lúčov je výsledný postup závislý na vzájomnej polohe priemetne a projektovaného telesa.)

9.4.1 Kolmá projekcia

Najjednoduchší spôsob projekcie bodu P je zanedbanie jednej z jeho súradníc. Napr. pri zanedbaní z-súradnice dostávame pôdorys telesa

$$x' = x, \quad y' = y, \quad z' = 0.$$

Táto projekcia má transformačnú maticu

$$\mathcal{O}_{xy} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{9.6}$$

Často sa s ňou stretávame v technickej praxi. Pre svoju jednoduchosť je implementovaná ako základná metóda v grafických systémoch.

9.4.2 Kosouhlá projekcia

Zovšeobecnením kolmej projekcie je kosouhlá projekcia. Jej maticové vyjadrenie je

$$\mathcal{P}_{xy} = \begin{pmatrix} 1 & 0 & a & 0 \\ 0 & 1 & b & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{9.7}$$

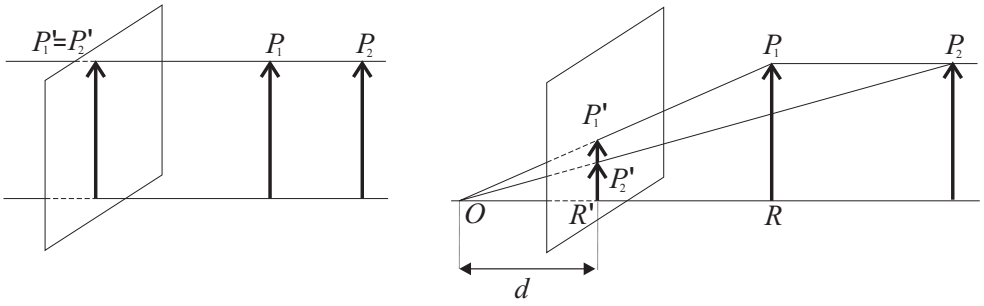
Uvažujme priamku

$$p(t) = A + t\vec{u}. \tag{9.8}$$

Použitím kosouhlej projekcie (9.7) dostávame priamku $p'(t) = A' + t\vec{u'}$, kde

$$\begin{array}{rcl} A' & = & (x_A + z_A a,\ y_A + z_A b,\ 0) \\ \vec{u'} & = & (u_x + u_z a,\ u_y + u_z b,\ 0) \end{array}.$$

Keďže projekcia vektora $\vec{u}$ nezávisí na bode A, projekcie rovnobežných priamok zostanú taktiež rovnobežné. V dôsledku toho veľkosť priemetu nezávisí na vzdialenosti predmetu od priemetne obr. 9.4a). Toto je však v rozpore s našim vnímaním priestorových objektov (napr. zdanlivé zbiehanie sa rovnobežných koľajníc, čo súvisí so zmenšovaním projekcie objektu s jeho zväčšujúcou sa vzdialenosťou).



a) b)

Obr. 9.4: a) Kolmá projekcia. b) Stredová projekcia.

9.4.3 Stredová projekcia

Tento spôsob premietania sa uplatňuje v našom vnímaní priestoru. Mechanizmus stredového premietania ilustruje obr. 9.4b). Vidíme, že na rozdiel od kolmej projekcie, stredová projekcia závisí aj na polohe pozorovateľa O.

Odvodíme vzťahy pre výpočet súradníc projekcie. Na obr. 9.4b) zavedieme súradnicový systém tak, že bod R' má súradnice $(0,0,0)$, pozorovateľ O má súradnice $(0,0,-d)$, projekčná rovina je $z=0$, bod P_1 má súradnice (x,y,z) a bod R má súradnice $(0,0,z)$. Z podobnosti trojuholníkov OP_1R a $OP_1'R'$ dostávame vzťahy

$$\frac{x'}{d} = \frac{x}{d+z}\,, \quad \frac{y'}{d} = \frac{y}{d+z}\,,$$

tj.

$$x' = \frac{d}{d+z}x, \quad y' = \frac{d}{d+z}y, \quad z' = 0. \tag{9.9}$$

9.4.3.1 Stredová projekcia rovnobežných priamok

Projekcia priamky (9.8) v tomto prípade vedie ku vzťahom

$$x'(t) = \frac{dx_A + du_x t}{d + z_A + u_z t}\,, \quad y'(t) = \frac{dy_A + du_y t}{d + z_A + u_z t}\,. \tag{9.10}$$

Uvažujme prípad $u_z = 0$, tj. priamku rovnobežnú s projekčnou rovinou. Vzťah (9.10) sa zjednoduší na

$$x'(t) = \frac{d}{d+z_A}(x_A + u_x t), \quad y'(t) = \frac{d}{d+z_A}(y_A + u_y t).$$

Vidíme, že projekciou smerového vektoru dostaneme vektor $\vec{u}' = \frac{d}{d+z_A}(u_x, u_y, 0)^T$. To znamená, že projekciou takej priamky sa jej smer nemení. Z toho vyplýva, že *dvojica priamok rovnobežných s priemetňou zostane rovnobežná i po projekcii.*

Uvažujme teraz prípad $u_z \neq 0$, tj. priamku, ktorá priemetňu pretína. Pre vzťahy (9.10) existuje limita pri $t \to \infty$

$$x'(\infty) = d\frac{u_x}{u_z}, \quad y'(\infty) = d\frac{u_y}{u_z}. \tag{9.11}$$

Všimnite si, že tieto limity závisia len od smerového vektoru. Ináč povedané, *všetky rovnobežné priamky pretínajúce priemetňu sa sprojektujú tak, že v limite dosiahnu bod* (9.11), *zvaný úbežník*[5].

9.4.3.2 Maticová reprezentácia stredovej projekcie

Chceme vyjadriť, ako závisia súradnice priemetu bodu od súradníc samotného bodu. Vzhľadom na vyššie uvedený formálny aparát založený na maticovom násobení, toto chceme využiť i pre stredovú projekciu.

Z (9.9), tj. z reprezentácie bodu v kartézskych súradniciach

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{d}{d+z}x \\ \frac{d}{d+z}y \\ 0 \end{pmatrix}$$

môžeme prejsť k vyjadreniu v homogénnych súradniciach (9.4)

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} \longleftrightarrow w \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = w \begin{pmatrix} \frac{d}{d+z}x \\ \frac{d}{d+z}y \\ 0 \\ 1 \end{pmatrix}.$$

Pre $w = \frac{d+z}{d}$ dostávame

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ 0 \end{pmatrix} \longleftrightarrow \frac{d+z}{d} \begin{pmatrix} \frac{d}{d+z}x \\ \frac{d}{d+z}y \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ 0 \\ 1+\frac{z}{d} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{d} & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}.$$

[5] Toto je práve ten bod v ktorom sa v nekonečne zbiehajú koľajnice.

Dostali sme tak maticu stredovej projekcie na rovinu xy

$$\mathcal{C}_{xy} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{d} & 1 \end{pmatrix}. \tag{9.12}$$

9.5 Zobrazovací kanál

Pri interaktívnej manipulácii s telesom sme nútení používať niekoľko súradnicových systémov, minimálne dva: súradnicový systém zobrazovanej scény a súradnicový systém výstupného zariadenia. Keďže výstupných zariadení môže byť niekoľko (obrazovka, tlačiareň – pritom takýchto zariadení môže byť niekoľko), vyššie zmienené transformácie je zmysluplné rozdeliť do troch fáz:

1. prechod od reálnych súradníc k normalizovaným,
2. realizácia transformácií v normalizovaných súradniciach,
3. prechod od normalizovaných súradníc k súradniciam výstupného zariadenia.

Prvý a tretí krok pozostávajú len z translácií a škálovania. Pritom závislosť na reálnej rozmerovej škále sa prejaví len v prvom kroku a závislosť na výstupnom zariadení sa prejaví len v poslednom - treťom kroku. Samotné transformácie sa tak stávajú nezávislé jednak na vstupných datach jednak na výstupných zariadeniach, čo sprehľadňuje celý proces zobrazovacieho kanálu. Naviac, vhodnou normalizáciou riešime výber stacionárneho bodu – viď kap. 9.2, obr. 9.2, 9.3.

9.5.1 Prechod k normalizovaným súradniciam

Keď vrcholy telesa, ktoré transformujeme, sú z oblasti

$$x \in \langle x_{min}, x_{max} \rangle, \quad y \in \langle y_{min}, y_{max} \rangle, \quad z \in \langle z_{min}, z_{max} \rangle,$$

tak prechod k normalizovaným súradniciam sa realizuje

1. posunutím o vektor

$$\vec{t} = \left(-\frac{x_{min} + x_{max}}{2}, -\frac{y_{min} + y_{max}}{2}, -\frac{z_{min} + z_{max}}{2} \right),$$

2. škálovaním hodnotou

$$q = \frac{2}{\max\left\{\left(x_{max} - x_{min}\right), \left(y_{max} - y_{min}\right), \left(z_{max} - z_{min}\right)\right\}}.$$

$$\mathcal{Q} = \begin{pmatrix} q & 0 & 0 & 0 \\ 0 & q & 0 & 0 \\ 0 & 0 & q & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} q & 0 & 0 & qt_x \\ 0 & q & 0 & qt_y \\ 0 & 0 & q & qt_z \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{9.13}$$

Týmto stred oblasti presunieme do začiatku súradnícového systému (posunutie) a oblasť vložíme do kvádra $\langle -1, 1\rangle \times \langle -1, 1\rangle \times \langle -1, 1\rangle$ (škálovanie). Takto realizovanou normalizáciou sa stacionárnym bodom pre prípadné rotácie či škálovania oblasti stáva jej stred (ťažisko). Naviac, model sa stáva nezávislým na pôvodnej veľkosti.

Všimnime si, že poradie operácií je dôležité. Poradie matíc musí byť v súlade s (9.5).

9.5.2 Prechod k súradniciam výstupného zariadenia

Keď okno výstupného zariadenia má rozsah

$$\langle wx_{min}, wx_{max}\rangle \times \langle wy_{min}, wy_{max}\rangle,$$

tak prechod od normalizovaných súradníc k súradniciam zobrazovacieho zariadenia dostaneme

1. projekciou oblasti na rovinu xy,
2. preškálovaním normalizovaných hodnôt na veľkosť výstupného okna hodnotou

$$w = \frac{\min\left\{\left(wx_{max} - wx_{min}\right), \left(wy_{max} - wy_{min}\right)\right\}}{2},$$

3. posunutím začiatku súradníc do stredu okna, tj. o vektor

$$\vec{u} \equiv (u_x,\ u_y) = \left(\frac{wx_{min} + wx_{max}}{2}, \frac{wy_{min} + wy_{max}}{2}\right).$$

Matica výstupného zariadenia $\mathcal{K}$ má preto tvar (pri použití kolmej projekcie)

$$\mathcal{K} = \begin{pmatrix} 1 & 0 & 0 & u_x \\ 0 & 1 & 0 & u_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} w & 0 & 0 & 0 \\ 0 & w & 0 & 0 \\ 0 & 0 & w & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} w & 0 & 0 & u_x \\ 0 & w & 0 & u_y \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{9.14}$$

Vďaka tomu, že v transformáciách zobrazovacieho kanálu sme vyčlenili normalizáciu dát, matica (9.14) je nezávislá na konkrétnych hodnotách vstupných dat a závisí len na charakteristikách daného výstupného zariadenia. Je súčasťou ovládačov výstupných zariadení.

Opäť si je treba uvedomiť dôležitosť poradia jednotlivých operácií – všimnime si, že poradie translácie a škálovania je opačné ako v prípade (9.13).

9.5.3 Zobrazovací kanál

Celý zobrazovací kanál môžeme vyjadriť takto:

1. Inicializuj matice elementárnych transformácií:
   - rotácie v rovinách xy, yz, zx o jednotkový uhol v oboch smeroch,
   - translácie v smeroch súradnicových osí o jednotkový krok v oboch smeroch,
   - škálovanie (zväčšenie a zmenšenie o jednotkový krok).
2. Inicializuj transformačnú maticu $\mathcal{Q}$ (normovanie súradníc).
3. Inicializuj maticu výstupného zariadenia $\mathcal{K}$.
4. Vykresli bod $P_{out} = \mathcal{K}\mathcal{Q}P$.
5. V závislosti na požadovanej transformácii vyber maticu elementárnej transformácie $\mathcal{M}$.
6. Aktualizuj transformačnú maticu $\mathcal{Q} := \mathcal{M}\mathcal{Q}$.
7. Prechod na krok 4.

V krokoch 2. a 3. inicializácie prebehnú podľa (9.13) a (9.14). Vidíme, že v cykle transformačného reťazca (kroky 4.–6.) používame len násobenie matíc a operáciu priradenia. Násobenie matíc vo všeobecnom tvare len v kroku 6. Vzhľadom na tvar matice $\mathcal{K}$, v kroku 4. násobíme len 2 riadky matice $\mathcal{K}$ maticou $\mathcal{Q}$, preto je vhodné toto realizovať jednoduchšiou procedúrou.

9.6 Urýchlenie operácií

Proces transformácií je založený na násobení matíc, kde sa počíta skalárny súčin príslušného riadku a stĺpca násobených matíc

$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + a_{i3}b_{3j} + a_{i4}b_{4j}. \tag{9.15}$$

Musíme pri tom realizovať štyri násobenia a tri sčítania. Keď k tomu použijeme jednu aritmetickú jednotku (procesor) a proces prebehne sekvenčne, tj. v jednom elementárnom

kroku jedna elementárna operácia sčítania alebo násobenia, tak na výpočet potrebujeme sedem krokov. Keď si predstavíme, že máme k dispozícii jednoduchšie, ale špecializované aritmetické jednotky, ktoré môžu pracovať súčasne, napr. dva násobiče a jeden sčítač, celý výpočet sa dá spraviť v štyroch krokoch:

1. `krok:` $m_1 = a_{i1}b_{1j}$, $m_2 = a_{i2}b_{2j}$
2. `krok:` $s = m_1 + m_2$, $m_3 = a_{i3}b_{3j}$, $m_4 = a_{i4}b_{4j}$
3. `krok:` $s = s + m_3$,
4. `krok:` $s = s + m_4$.

Vzhľadom na to, že pri násobení matíc robíme operácií (9.15) viac, v krokoch 3.-4. môžeme spraviť násobenia pre ďalší skalárny súčin, čím sa celkové násobenie matíc ešte urýchli. (Samozrejme, pre ďalšie urýchlenie je možné použiť viac aritmetických jednotiek).

Tento prístup sa v súčasnosti intenzívne využíva. Vedie k tzv. maticovým procesorom a pre potreby grafiky býva implementovaný v grafických kartách.

9.7 Záverečné poznámky – tuhé transformácie rastrového obrazu

Objekty vo vektorovej grafike spravidla konštruujeme tak, aby boli invariantné voči tuhým transformáciám. Ináč povedané, geometria objektov je daná vopred zvolenými bodami. Pre tuhú transformáciu objektu stačí realizovať ju na vopred zvolených bodoch – všetko ostatné sa dopočíta tak, že výsledná geometria sa nezmení.

Pri tuhých transformáciách rastrového obrazu musíme toto realizovať pre každý pixel, pritom indexy pixelov môžeme považovať za ich kartézske súradnice. Dôležitá zmena oproti vektorovým modelom je tá, že výsledné súradnice (po transformácii) musia byť taktiež celočíselné.

V prípade translácie toto nečiní problém. V prípade škálovania pre $s > 1$ musíme hodnoty v chýbajúcich pixeloch interpolovať (kap. 4.3, kap. 11.6). Pozrieme sa detailnejšie na rotáciu rastrového obrazu.

Obr. ukazuje, že priamou aplikáciou rotácie na každý pixel pôvodného obrazu s následnou konverziou súradníc na celočíselné, dostaneme vo výslednom obraze nevyplnené pixely. Východiskom je použitie spätnej rotácie, tj. pre každý pixel výsledného obrazu sledujeme, rotáciou ktorého vstupného pixelu vznikol. Ponúka sa pritom použiť vyhladenie váženým priemerovaním štvorice najbližších pixelov.

9.8 Zhrnutie a úlohy

- Boli zavedené homogénne súradnice zobrazovaných bodov,

- sformulovali sme maticové vyjadrenie tuhých transformácií,
- zmienili sme vzťah tuhých transformácií a projekcií,
- popísali sme fázy transformačného reťazca,
- ukázali sme, vďaka čomu sa tuhé transformácie realizujú rýchlo.

Otázky

1. Prečo je nevyhnutné zaviesť homogénne súradnice? Aký je geometrický význam homogénnych súradníc?
2. Prečo pre potreby počítačovej grafiky stačí projektovať 3D teleso na jedinú rovinu $z = 0$ a nemusíme uvažovať projekciu na obecnú rovinu?
3. Prečo a na aké fázy segmentujeme zobrazovací kanál?
4. Ktoré z uvedených častí zobrazovacieho kanálu sú súčasťou ovládačov výstupných grafických zariadení?
5. V prípade, že pixel vo výstupnom zariadení nemá štvorcový tvar, kde a ako sa to prejaví v zobrazovacom kanáli?
6. Prečo je maticové násobenie vhodný aparát pre tuhé transformácie?

Kapitola 10

Krivky v počítačovej grafike

Vieme už, ako efektívne generovať úsečku. A uviedli sme taktiež, že všeobecná krivka sa v konečnom dôsledku realizuje tak, že vygenerujeme dostatočne reprezentatívnu množinu bodov krivky a tie spojíme lineárne lomenou čiarou. Teraz sa budeme venovať práve vygenerovaniu reprezentatívnej množiny bodov požadovanej krivky.

Generovanie kriviek je problematika veľmi dôležitá. Je základom CAD/CAM systémov (Computer Aided Design/Machinery). Ich rozvoj bol stimulovaný potrebami hlavne strojárenského priemyslu. Práve vývoj CAD systémov bol jedným z hlavných faktorov rýchleho vývoja a solídneho matematického základu počítačovej grafiky. Hlavné myšlienky sú z 50. - 60. rokov 20. storočia. Možno povedať, že podstatným spôsobom zmenili prácu konštruktérov a návrhárov. V dnešnej dobe CAD/CAM systémy prenikli prakticky do všetkých priemyselných odvetví.

S krivkami sa stretneme i v geografických informačných systémoch (GIS), krivky sú napr. i základom pre generovanie písmenných fontov textových editorov a DTP systémov (Desk Top Publishing).

10.1 Matematický formalizmus generovania kriviek

Generovať body krivky na základe explicitného vyjadrenia funkcií, tj. v tvare $y = f(x)$ nie je vhodné, pretože nie je dôvod uprednostňovať jednu zo súradníc (tj. x voliť ako nezávislú a y ako závislú). Naviac, tento spôsob je nepohodlný pre generovanie kriviek v 3D. Zápis $z = f(x, y)$ totiž vyjadruje nie krivku, ale plochu. Krivku by sme preto museli reprezentovať ako prienik dvoch plôch

$$\begin{aligned} z &= f(x, y) \\ z &= g(x, y) \end{aligned} .$$

Dnešné softvérové nástroje pre generovanie kriviek (a plôch) sú založené na reprezentácii, v ktorej bod krivky P chápeme ako lineárnu kombináciu vopred zadaných riadiacich bodov

$P_0, P_1, ..., P_n$. Napríklad pre úsečku AB používame parametrický tvar

$$P(t) = A + t(B - A), \quad 0 \leq t \leq 1,$$

resp.

$$P(t) = (1 - t) A + tB, \quad 0 \leq t \leq 1. \tag{10.1}$$

Hľadaný bod je vyjadrený ako lineárna kombinácia známych bodov A, B, pričom vplyv týchto bodov je daný funkciami $f_A(t) = 1 - t,\ f_B(t) = t$.

(10.1) je východiskom pri generovaní všeobecnejších kriviek:

$$P(t) = f_0(t) P_0 + f_1(t) P_1 + \cdots + f_n(t) P_n, \quad 0 \leq t \leq 1, \tag{10.2}$$

kde $P_0, P_1, \ldots, P_n$ sú *kontrolné body* - body, ktoré zadáva používateľ a $f_0(t), f_1(t), \ldots, f_n(t)$ sú *určujúce polynomické funkcie.*

Z používateľského hľadiska je dôležité, aby kontrolných bodov nebolo veľa, a aby ich vplyv na výsledný tvar krivky bol ľahko predstaviteľný.

Kým kontrolné body majú zásadný vplyv na tvar konkrétnej realizácie krivky, voľba určujúcich polynomických funkcií definuje všeobecné vlastnosti danej triedy kriviek. (Napríklad to, že krivka (10.1) je úsečka, ktorej koncové body sú práve riadiace body, je dané tým, že polynomické funkcie sú práve $f_A(t) = 1 - t,\ f_B(t) = t$.)

Vyjadrenie (10.2) dáva predpis pre výpočet každej súradnice, tj.

$$\begin{aligned} x(t) &= f_0(t) x_0 + f_1(t) x_1 + \cdots + f_n(t) x_n \\ y(t) &= f_0(t) y_0 + f_1(t) y_1 + \cdots + f_n(t) y_n \end{aligned}$$

a v prípade 3-rozmerných kriviek ešte

$$z(t) = f_0(t) z_0 + f_1(t) z_1 + \cdots + f_n(t) z_n\ .$$

Geometrický význam derivácie funkcie je smernica dotyčnice. V nami použitej formalizácii je derivácia krivky daná deriváciami určujúcich funkcií

$$P'(t) = \sum_{i=0}^{n} f_i'(t) P_i \tag{10.3}$$

a má podobný význam – definuje smerový vektor krivky v bode $P(t)$.

10.2 Bézierove krivky

V praxi sú veľmi často používané Bézierove krivky. Pre vopred zvolené n ich definujú nasledujúce funkcie (Bernsteinove polynómy):

$$f_k(t) = \binom{n}{k}(1-t)^{n-k}t^k, \qquad 0 \le t \le 1. \tag{10.4}$$

V prípade Bézierovych kriviek prvého stupňa (n=1) potrebujeme dva riadiace body P_0, P_1. Z (10.4) dostávame funkcie: $f_0(t) = (1-t)$, $f_1(t) = t$. Dosadením do (10.2) tak vidíme, že Bézierova krivka prvého stupňa je úsečka.

Vlastnosti Bézierových kriviek demonštrujeme na krivkách tretieho stupňa. Bude nás pritom zaujímať aké sú koncové body krivky a aký smer má krivka v koncových bodoch.

Pre kubické krivky potrebujeme štyri riadiace body P_0, P_1, P_2, P_3 a z (10.4) máme:

$$f_0(t) = (1-t)^3,\ f_1(t) = 3(1-t)^2 t,\ f_2(t) = 3(1-t)t^2,\ f_3(t) = t^3. \tag{10.5}$$

Pre hodnotu $t = 0$ dostaneme $f_0(0) = 1,\ f_1(0) = f_2(0) = f_3(0) = 0$, z čoho vyplýva, (dosadením do (10.2)) že $P(0) = P_0$.

Podobne pre $t = 1$ máme $f_0(1) = f_1(1) = f_2(1) = 0,\ f_3(1) = 1$, a preto $P(1) = P_3$.

Zderivovaním (10.5) dostaneme pre smery analyzovanej krivky (10.3)

$$P'(t) = -3(1-t)^2 P_0 + 3(1-t)(1-3t)P_1 + 3t(2-3t)P_2 + 3t^2 P_3.$$

V krajných bodoch tak máme smery

$$P'(0) = 3(P_1 - P_0), \quad P(1) = 3(P_3 - P_2).$$

Podobný postup aplikovaný na Bézierovu krivku n-tého by nás doviedol k výsledku

$$\begin{array}{ll} P(0) = P_0, & P(1) = P_n, \\ P'(0) = n(P_1 - P_0), & P'(1) = n(P_n - P_{n-1}). \end{array} \tag{10.6}$$

Z uvedeného vidíme, že vplyv riadiacich bodov na tvar krivky je ľahko pochopiteľný:

- *krajnými riadiacimi bodami krivka prechádza,*
- *bezprostredne susedné riadiace body určujú smer krivky v koncových bodoch.*

Ďalšou, veľmi dôležitou vlastnosťou Bézierových kriviek je, že

- *krivka nevychádza mimo konvexný obal svojich riadiacich bodov.*

Dôkaz poslednej vlastnosti vyplýva z toho, že

1. funkcie $f_i(t)$ sú na intervale $0 \leq t \leq 1$ nezáporné – toto je celkom zrejmá vlastnosť vyplývajúca z definície (10.4) Bernsteinovych polynómov,

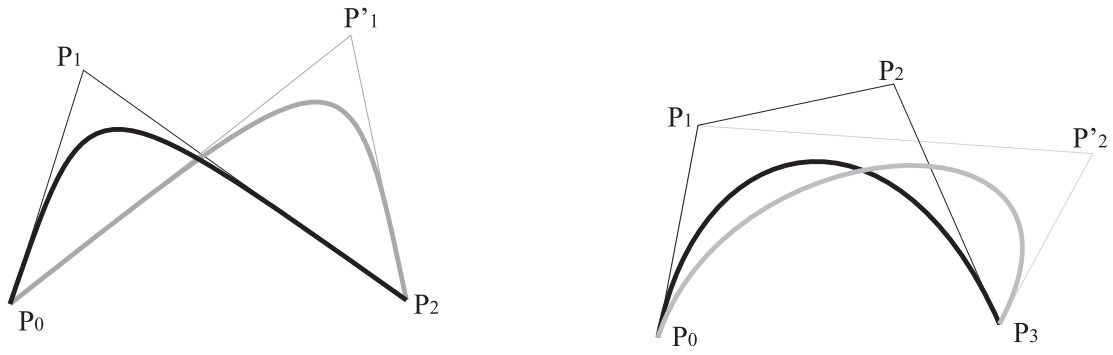
2. $\sum_{i=0}^{n} f_i(t) = 1$ – toto dostaneme dosadením $A = 1 - t$, $B = t$ do binomickej vety:

$$(A + B)^n = \sum_{i=0}^{n} \binom{n}{i} A^{n-i} B^i.$$

3. Určujúce polynómy tak majú charakter *váh* jednotlivých riadiacich bodov a výsledný bod $P(t)$ v (10.2) je vlastne *váženým priemerom* riadiacich bodov. Zostáva si už len uvedomiť, že všetky možné vážené priemery bodov pri nezáporných váhach *tvoria konvexný obal týchto bodov*.

Vidíme, (10.6), obr. 10.1, že kým v prípade Bézierovej krivky druhého stupňa zmena riadiaceho bodu P_1 zmení smer krivky v oboch krajných bodoch (P_1 je totiž druhý a zároveň predposledný riadiaci bod), tak v prípade Bézierových kriviek od tretieho stupňa vyššie, môžeme smery krivky v koncových bodoch meniť vzájomne nezávisle – smer v bode P_n závisí len na polohe bodov P_{n-1} a P_n a podobne smer v bode P_0 závisí len na bodoch P_0, P_1 .

A toto je dôvod veľmi častého používania Bézierových kriviek práve tretieho stupňa a implementácie tohto typu kriviek v grafických softvéroch.



Obr. 10.1: Editovanie Bézierovej krivky 2. a 3. stupňa.
V prípade a) (Bézierova krivka 2. stupňa) posun riadiaceho bodu P_1 zmení smer krivky v oboch koncových bodoch, v prípade b) (Bézierova krivka 3. stupňa) posun riadiaceho bodu P_2 zmení smer krivky v koncovom bode P_3, ale nezmení smer v bode P_0.

10.3 Spájanie kriviek

Keď chceme spojiť dve Bézierove kubické krivky, určené riadiacimi bodmi P_0, P_1, P_2, P_3, Q_0, Q_1, Q_2, Q_3 tak, aby výsledná krivka bola hladká, tj. aby na spojení nebol „ostrý roh“, je treba splniť nasledujúce podmienky:

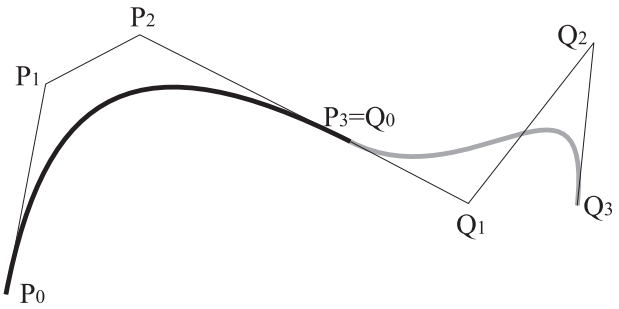
(a) $P_3 = Q_0$,

(b) *body* P_2, P_3, Q_1 *ležia na jednej priamke.*

Podmienka (a) garantuje *spojitosť*, podmienka (b) zabezpečuje, že smer na konci prvej krivky, zadanej bodami P_0, P_1, P_2, P_3, je rovnaký ako smer na začiatku druhej krivky. Hovoríme tomu *geometrická hladkosť* v bode napojenia. Namiesto podmienky (b) sa niekedy vyžaduje silnejšia podmienka –

(b') *bod* P_3 *leží uprostred úsečky* P_2Q_1.

V tomto prípade dostávame v bode napojenia *parametrická hladkosť.*



Obr. 10.2: Geometricky hladké spojenie Bézierovych kubických oblúkov.

Mnohé grafické softvéry podmienku (b') automaticky kontrolujú.

Z hľadiska matematiky geometrická hladkosť napojenia kriviek v bode $P_3 = Q_0$ znamená, že existuje prvá derivácia $y'(x) = \frac{dy}{dx}$ v tomto bode. Vychádzajúc z parametrického vyjadrenia súradníc $x(t)$, $y(t)$ dostávame

$$y'(x) = \frac{dy}{dx} = \frac{\frac{dy}{dt}}{\frac{dx}{dt}}.$$

Parametrická hladkosť predpokladá spojitosť derivácií pre obe súradnicové funkcie

$$\frac{dy_p}{dt}|_{t=1} = \frac{dy_Q}{dt}|_{t=0}, \qquad \frac{dx_p}{dt}|_{t=1} = \frac{dx_Q}{dt}|_{t=0},$$

kým pre geometrickú hladkosť stačí[1] , keď existuje také $k \neq 0$, že

$$\frac{dy_p}{dt}|_{t=1} = k\frac{dy_Q}{dt}|_{t=0}, \qquad \frac{dx_p}{dt}|_{t=1} = k\frac{dx_Q}{dt}|_{t=0}.$$

[1] Keď interpretujeme krivku ako trajektóriu pohybujúceho sa bodu, tak v prípade geometrickej hladkosti vyžadujeme len, aby rýchlosť ktorú má pohybujúci sa bod na konci prvého úseku mala rovnaký smer, ako na začiatku druhého úseku. Tj. nevylučujeme skokovú zmenu veľkosti rýchlosti. V prípade parametrickej hladkosti vyžadujeme v bode napojenia kriviek aby i veľkosti rýchlostí boli zachované.

Na prvý pohľad sa zdá, že parametrická hladkosť je silnejšia ako geometrická, no nie je tomu tak. Takto skonštruované spojenie nám zaručuje spojitosť prvých derivácií – hovoríme o *hladkosti prvého stupňa*. Spojitosť derivácií vyšších stupňov zaručenú nemáme a u Bézierovských oblúkov vyššie ukázanou konštrukciou to ani nie sme schopní dosiahnúť. Pritom vo vnútri každej krivky n-tého stupňa máme derivácie až do n-tého stupňa a každá z nich je spojitá (tj. máme parametrickú hladkosť stupňa n). Všeobecne sa také spojenie kriviek (kde stupeň hladkosti nie je vo všetkých bodoch rovnaký) nazýva *splajn*[2]. Viac o tejto problematike možno nájsť v [15].

10.4 Implementačná efektivita

Pre výpočet vnútorných bodov Bézierovej krivky 3. stupňa priamo podľa vzorcov (10.2), (10.4)

$$P(t) = (1-t)^3 P_0 + 3(1-t)^2 t P_1 + 3(1-t) t^2 P_2 + t^3 P_3 \tag{10.7}$$

cyklus výpočtu týchto bodov možno zapísať nasledovne:

```
for (t= 0; t<1; t=t+d)
{ f0=(1-t)*(1-t)*(1-t);
  f1=3*(1-t)*(1-t)*t;
  f2=3*(1-t)*t*t;
  f3=t*t*t;
  x=f0*x0 + f1*x1 + f2*x2 + f3*x3;
  y=f0*y0 + f1*y1 + f2*y2 + f3*y3;
  z=f0*z0 + f1*z1 + f2*z2 + f3*z3;
}
```

Musíme teda pre každé t nájsť štyri polynómy tretieho stupňa. Treba si uvedomiť, že súradnice riadiacich bodov $P_0, \ldots, P_3$ sa nemenia, tj. vo výpočtoch figurujú ako konštanty.

10.4.1 Maticový tvar krivky a asociativita násobenia

Roznásobením polynómov v (10.7) dostávame

$$P(t) = \left(1 - 3t + 3t^2 - t^3\right) P_0 + 3\left(t - 2t^2 + t^3\right) P_1 + 3\left(t^2 - t^3\right) P_2 + t^3 P_3 \tag{10.8}$$

a preskupením členov tak, že združíme činitele s rovnakým stupňom mocniny, máme výsledný tvar:

$$P(t) = P_0 + 3t(P_1 - P_0) + 3t^2(P_0 - 2P_1 + P_2) + t^3(-P_0 + 3P_1 - 3P_2 + P_3) \tag{10.9}$$

Uvedenému postupu odpovedá nasledujúca postupnosť úprav:

[2] z anglického *spline*, čo v prvotnom význame označovalo tenké dlhé úzke dosky, ktoré sa používali pri výrobe trupov lodí.

1. zápis (10.7) v tvare skalárneho súčinu dvoch vektorov

$$P(t) = \left((1-t)^3,\, 3(1-t)^2 t,\, 3(1-t)t^2,\, t^3\right) \begin{pmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{pmatrix}, \tag{10.10}$$

2. vyjadrenie prvého z vektorov v (10.10) s využitím násobenia matíc v takom tvare, že premenná t a konštanty sú vzájomne odseparované:

$$\left((1-t)^3,\, 3(1-t)^2 t,\, 3(1-t)t^2,\, t^3\right) = (1,\, t,\, t^2,\, t^3) \begin{pmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{pmatrix}, \tag{10.11}$$

3. výsledný maticový tvar Bézierovej krivky tretieho stupňa je

$$P(t) = \left(1,\, t,\, t^2,\, t^3\right) \begin{pmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{pmatrix} \begin{pmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{pmatrix}. \tag{10.12}$$

Keď vo výslednom tvare v (10.12) označíme vektory a maticu postupne $\mathcal{T}, \mathcal{M}, \Pi$, vidíme, že toto vyjadrenie prirodzeným spôsobom vzájomne oddeľuje vektor parametrizácie $\mathcal{T}$, mechanizmus generovania krivky $\mathcal{M}$ a systém riadiacich bodov Π:

$$P(t) = \mathcal{T}\mathcal{M}\Pi. \tag{10.13}$$

Vzťahy (10.7) a (10.9) sa líšia rôznym poradím násobenia matice a vektorov:

pre (10.7) máme $P(t) = (\mathcal{T}\mathcal{M})\,\Pi$, pre (10.9) je to $P(t) = \mathcal{T}\,(\mathcal{M}\Pi)$.

Vidíme, že v tvare (10.9) sú výpočty v tele cyklu podstatne jednoduchšie:

```
qx0=x0; qx1=3(x1-x0); qx2=3(x2-2*x1+x0); qx3=x3-3x2+3x1-x0;
qy0=y0; qy1=3(y1-y0); qy2=3(y2-2*y1+y0); qy3=y3-3y2+3y1-y0;
qz0=z0; qz1=3(z1-z0); qz2=3(z2-2*z1+z0); qz3=z3-3z2+3z1-z0;
for (t= 0; t<1; t=t+d)
{ tt=t*t;
  ttt=tt*t;
  x=qx0 + qx1*t + qx2*tt + qx3*ttt;
  y=qy0 + qy1*t + qy2*tt + qy3*ttt;
  z=qz0 + qz1*t + qz2*tt + qz3*ttt;
}
```

tj. konštanty závisiace len od riadiacich bodov vypočítame iba raz a vo vnútri cyklu sa realizuje podstatne menej operácií.

Ďalšie urýchlenie sa dá získať rekurentným výpočtom hodnôt v (10.9).

Pri označení $P(t) = Q_0 + tQ_1 + t^2Q_2 + t^3Q_3$ dostávame

$$P(t+d) = Q_0 + (t+d)\,Q_1 + (t+d)^2\,Q_2 + (t+d)^3\,Q_3,$$

a po roznásobení máme

$$P(t+d) = P(t) + dQ_1 + d^2Q_2 + d^3Q_3 + t\,(2Q_2 + 3Q_3d)\,d + t^2 3dQ_3.$$

Označiac $R_0 = dQ_1 + d^2Q_2 + d^3Q_3, \quad R_1 = (2Q_2 + 3Q_3d)\,d, \quad R_2 = 3dQ_3$, a prepočítajúc súradnice konštantných bodov R_0, R_1, R_2 jednorázovo pred cyklom, telo cyklu sa ešte zjednoduší:

```
x=x0; y=y0; z=z0;
for (t= 0; t<1; t=t+d)
{ tt=t*t;
  x+=(rx0 + rx1*t + rx2*tt);
  y+=(ry0 + ry1*t + ry2*tt);
  z+=(rz0 + rz1*t + rz2*tt);
}
```

Uvedená schéma spočívajúca v roznásobení (10.8)–(10.9) a využití rekurentného vyjadrenia hodnôt je všeobecne platná i pre iný typ kriviek (podstatný je akurát polynomický tvar určujúcich funkcií).

10.4.2 Schéma de Casteljau

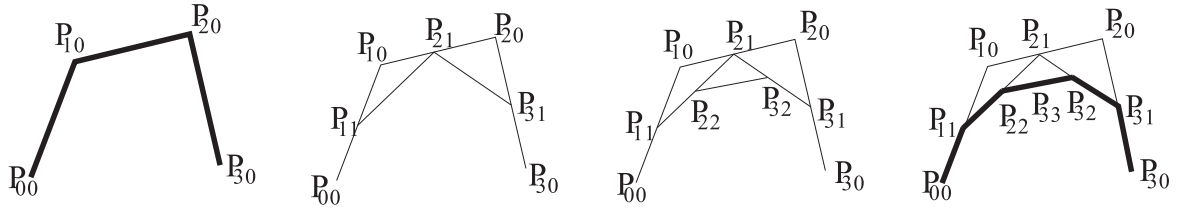
Pre Bézierove krivky existuje ešte ďalšie – oveľa silnejšie zefektívnenie celého výpočtu – *schéma de Casteljau.*

Označme tentokrát riadiace body $P_{0,0}$, $P_{1,0}$, $P_{2,0}, \ldots,$ $P_{n,0}$. Budeme generovať body:

$$P_{i,j}(t) = (1-t)\,P_{i-1,j-1} + tP_{i,j-1} \quad \text{pre } i = 1, \ldots, n, \quad j = i, \ldots, n. \tag{10.14}$$

V tejto rekurentnej schéme s narastajúcim j postupne klesá počet vygenerovaných bodov, až v kroku j=n dostávame jediný bod $P_{n,n}(t)$. Dá sa ukázať, že

1. $P_{n,n}(t)$ je bod Bézierovej krivky.



Obr. 10.3: Ilustrácia schémy de Casteljau pre kubickú Bézierovu krivku. a) Východisková aproximácia. b)-c) Postupný rekurentný výpočet bodov $P_{i,j}$. d) Aproximácia odpovedajúca jednému rekurzívnemu vnoreniu.

2. pre Bézierové krivky:
   $\mathcal{L}$ na riadiacich bodoch $P_{0,0}$, $P_{1,1}$, $P_{2,2}, \ldots, P_{n,n}$,
   $\mathcal{R}$ na riadiacich bodoch $P_{n,n}$, $P_{n,n-1}$, $P_{n,n-2}, \ldots, P_{n,0}$,
   $\mathcal{B}$ na riadiacich bodoch $P_{0,0}$, $P_{1,0}$, $P_{2,0}, \ldots, P_{n,0}$,
   platí veľmi dôležitý vzťah: *krivka $\mathcal{B}$ je zjednotením kriviek $\mathcal{L}$ a $\mathcal{R}$.*

Dôkaz vyššieuvedených tvrdení vychádza za rámec tohto texu, záujemci ho nájdu v [15].

Pre praktickú realizáciu je podstatná vlastnosť 2. Na jej základe môžeme rekurzívnou schémou adaptívneho zjemnenia skonštruovať lineárne lomenú čiaru, ktorá je priblížením Bézierovej krivky:

```
PROCEDURA Casteljau(C_0, C_1, ..., C_n, t)
1. { Vygeneruj lineárne lomenú čiaru C_0, C_1, ..., C_n ako
     priblíženie Bézierovej krivky.
2.   Ak priblíženie nie je dostatočné, potom nahraď
     lineárne lomenú čiaru C_0, C_1, ..., C_n
     nasledujúcou konštrukciou (kroky 3.-8.).
3.   { Zvoľ riadiace body P_{0,0} = C_0, P_{1,0} = C_1, P_{2,0} = C_2, ..., P_{n,0} = C_n
4.     Na riadiacich bodoch P_{0,0}, P_{1,0}, P_{2,0}, ..., P_{n,0} aplikuj
       rekurentnú schému (10.14) pre parameter t.
5.     Označ body L_i = P_{i,i}(t), R_i = P_{n,i}(t), i = 0, ..., n.
6.     Casteljau(L_0, L_1, ..., L_n, t).
7.     Casteljau(R_0, R_1, ..., R_n, t).
8.   }
9. }
```

Zvyčajne sa volí $t = \frac{1}{2}$. To dovoľuje efektívne realizovať schému de Casteljau vďaka tomu, že operáciu delenia je možné v tomto prípade nahradiť oveľa rýchlejším bitovým posunom.

Hlavný význam tejto schémy je v tom, že krivku postupne aproximujeme, pričom aproximáciu môžeme zjemňovať len lokálne v miestach, kde sa smer krivky výrazne mení.

10.5 Poznámky k všeobecnejšiemu vyjadreniu kriviek

Aj keď sa Bézierove krivky zdajú byť dostatočne všeobecným nástrojom, nie vždy vyhovujú našim požiadavkom. Najväčším problémom sú nasledujúce skutočnosti:

Neuniformita. Každý základný oblúk pri spájaní kriviek sa parametrizuje nezávisle, pričom na každom z nich je $0 \leq t \leq 1$. Pri spojení dvoch základných oblúkov veľmi rozdielnej dĺžky, sa rýchlosť pohybu bodu v bode napojenia základných oblúkov zmení skokom (smer pohybu síce zachováme, ale rýchlosť sa zmení) – obr. 10.2. Toto môže byť zdrojom problémov v praktických implementáciách CAD-technológií (napr. nespojitá zmena rýchlosti pohybu noža môže spôsobiť jeho rýchlejšie opotrebenie, ako aj defekty na reznej ploche). *Preto je rozumné voliť parametrizáciu tak, aby odpovedala skutočnej dĺžke základného oblúku.*

Racionalita: Parametrické vyjadrenie kriviek (s pomocou polynómov) bohužiaľ nedovoľuje presne popísať tak často sa vyskytujúce krivky ako napr. kužeľosečky. Najjednoduchší možný spôsob, ako parametricky vyjadriť napr. kružnicový oblúk (štvrťkružnicu), je

$$x(t) = \frac{2t}{1+t^2}, \quad y(t) = \frac{1-t^2}{1+t^2}, \quad 0 \leq t \leq 1,$$

tj. určujúce funkcie je treba uvažovať ako *racionálne polynomické funkcie.*

V praxi sa preto dnes vo veľkej miere používa všeobecnejší nástroj ako Bézierove krivky – NURBS krivky. Názov je skratkou pre *Non Uniform Rational B-Splines*. Okrem toho, že používa neuniformné delenie a racionálny tvar určujúcich funkcií, základné oblúky sú v tvare *B-splajnových kriviek,* ktoré sú určitým zovšeobecnením Bézierových kriviek. Pritom zovšeobecnenie je také, že napr. vlastnosť ohraničenia krivky konvexným obalom platí i pre B-splajn. Aj ostatné vlastnosti Bézierových kriviek sa na B-splajny v určitej miere prenášajú. Použitá formalizácia B-splajnových kriviek dovolila zjednotiť vyjadrenie rôznych používaných parametrzácií základných oblúkov (Bézierova krivka, Fergusonova krivka, Coonsova krivka). Zároveň aparát B-splajnov kriviek umožňuje používateľsky jednoducho riadiť hladkosť napojenia základných oblúkov.

Viac informácií o NURBS-krivkách poskytuje napr. [29]. Detailné odvodenie vlastností týchto kriviek, ako aj ich výsledné formálne vyjadrenie je možné nájsť v [15].

10.6 Zhrnutie a úlohy

Ukázali sme mechanizmus veľmi rozšírenej metódy generovania kriviek, na základe polynomickej parametrizácie.

- Sformulovali sme parametrický spôsob generovania kriviek.

- Zmienili sme sa o význame derivácií takýchto kriviek.
- Popísali sme základné vlastnosti Bézierovych kriviek.
- Ukázali sme niekoľko prístupov k zefektívneniu kódu.
- Sformulovali sme rýchly algoritmus realizácie Bézierovych kriviek.
- Zmienili sme sa o dôvodoch, ktoré vedú k používaniu obecnejšieho nástroja NURBS kriviek.

Kontrolné otázky:

1. Ako vyzerá Bézierova krivka 1. stupňa?
2. Prečo sú v praxi dôležité Bézierove krivky práve tretieho stupňa?
3. Aspoň akého stupňa musí byť Bézierova krivka, aby realizovala hladkú uzavretú krivku?
4. V čom je výhodný maticový zápis parametrických kriviek?

Samostatné úlohy:

1. Realizujte Bézierovu krivku vopred zvoleného stupňa.
2. Nahraďte lineárne lomenú čiaru systémom hladko napojených Bézierových kriviek druhého stupňa. Jednotlivé Bézierové kvadriky sú definované stredmi susedných úsečiek a spoločným bodom susedných úsečiek.

Kapitola 11

Modelovanie telies

V predpočítačovom období sa na popis geometrických modelov vyvinuli dva rôzne formálne prístupy. Jeden sa zameriava na *formalizáciu tvaru povrchu telies*. Využíva sa pritom aparát diferenciálnej geometrie (zjednodušene povedané je to spojenie analytickej geometrie a diferenciálneho a integrálneho počtu). Druhý prístup sa zameriava na *vyjadrenie štruktúry telesa* a používa aparát teórie grafov. Rozvoj výpočtovej techniky dovolil úzko prepojiť obidva prístupy.

Na tvorbu modelov sa môžeme pozerať z niekoľkých hľadísk.

Spôsob reprezentácie. Keďže objekty v priestore vypĺňajú určitý objem, možno ich charakterizovať práve na základe tejto znalosti. Toto vedie k *objemovému prístupu* pri modelovaniu. Najznámejšie praktické aplikácie s objemovou reprezentáciou dát sú napr. výsledky tomografického vyšetrenia, kde analyzovaný objekt je „rasterizovaný“ do 3D pravidelnej mriežky, a každej elementárnej kocke tejto mriežky je jednoznačne priradený materiál.
Iný, veľmi často používaný príklad objemového prístupu je CSG-model zmieňovaný v úvodnej kapitole.

V reálnom živote nie je veľa materiálov (napr. voda a vzduch), ktoré nám dovoľujú vidieť do vnútra objektov. Vo väčšine prípadov z telesa vidíme iba jeho povrch. Preto *povrchové reprezentácie* modelov telies majú pre vizualizáciu veľký význam.

Spôsob generovania. V úvodnej kapitole sme vysvetlili pojem vektorovej a rastrovej grafiky v 2D. Podobne možno pristupovať k modelovaniu a zobrazovaniu 3-rozmerných telies. Na príklade použitia parametrizácie v rovine a priestore si ukážeme, že tento mechanizmus sa dá rozvíjať rastrovým i vektorovým spôsobom.

11.1 CSG model

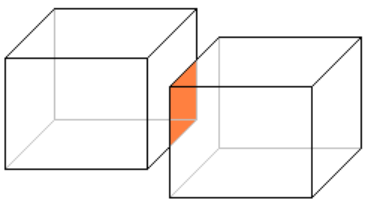
Skôr ako o „model“ ide v tomto prípade o „koncept“, ako vyjadriť zložité geometrické konštrukcie. Vychádza sa z toho, že vieme efektívne generovať jednoduché geometrické objekty, a na základe nich s pomocou množinových operácií vytvárame zložitejšie konštrukcie. Znamená to, že CSG model (Constructive Solid Geometry) pozostáva z

1. definovaných *primitívnych telies* (napr. kváder, valec, rotačný elipsoid),
2. parametrov, určujúcich *tvar primitívneho telesa* (napr. dĺžky hrán kvádra, polomer a výška valca, veľkosti hlavných poloosí elipsoidu),
3. parametrov, určujúcich *polohu a orientáciu telesa* v danom súradnicovom systéme (napr. skutočná poloha vybraných bodov telesa),
4. *atribútov* (napr. spôsob vyfarbenia hraničných plôch),
5. *množinových operácií* nad primitívnymi telesami (výsledné teleso budujeme pomocou zjednotenia, prieniku a rozdielu nad primitívnymi telesami).

Na rozdiel od štandardne používaných operácií zjednotenie, prienik, rozdiel, $\triangle \in \{\cup, \cap, \setminus\}$ používame *regularizované* množinové operácie, tj. pre množiny$A$, B, definujeme

$$A\tilde{\triangle}B = \text{uzáver}(\text{vnútrajšok}\,(A)\,\triangle\text{vnútrajšok}\,(B)). \tag{11.1}$$

Pritom *vnútrajškom* množiny rozumieme množinu bez hranice a *uzáverom* množiny rozumieme pridanie hranice k vnútrajšku. Dôvodom toho je, aby sme manipulovali len s geometrickými objektami rovnakej dimenzie a výsledkom množinových manipulácií bol opäť objekt tej istej dimenzie. Napr. pre dva kvádre, tj. 3D objekty z obr. 11.1, ktoré sa dotýkajú čiastočne len stenou, je ich štandardným prienikom vyznačený obdĺžnik, tj. 2D objekt. Regularizovaným prienikom je prázdna množina.



Obr. 11.1: Dva kvádre s čiastočným spoločným povrchým prekrytím.

Keďže množinové operácie sú binárne, táto reprezentácia vedie k použitiu formalizmu - *binárny strom*, ktorý v sebe nesie celú históriu konštrukcie modelu.

11.2 Mnohosteny

Jednoduché a často používané telesá sú mnohosteny s rovinnými povrchovými plochami. Táto trieda telies je dostatočne rôznorodá na to, aby vyjadrila presne alebo aspoň približne veľké množstvo reálnych telies. CSG model nad mnohostenami je veľmi často používaným nástrojom geometrického modelovania.

Pre definovanie mnohostenu potrebujeme informáciu dvojakého druhu:

geometria definuje primárne tvar modelovaného telesa, je daná súradnicami jeho vrcholov,

topológia určuje väzby medzi definujúcimi vrcholami telesa (hrany, steny ...).

Uvažujeme len také mnohosteny, v ktorých každá hrana je prienikom práve dvoch stien.[1]

Pre mnohosten ktorý obsahuje v vrcholov, f stien, e hrán, g dier a má c komponentov súvislosti, platí Eulerova veta (dôkaz pre $g = 0$ a $c = 1$ napr. v [21])

$$v + f - e = 2\,(c - g)\,. \tag{11.2}$$

Tento vzťah sa využíva pri editácii topológie mnohostenov, napr. pri pridávaní, resp. odoberaní vrcholov, hrán a stien. Kontrolujeme tým „syntaktickú“ konzistentnosť topologickej transformácie mnohostenu.

Dôležité je si uvedomiť, že zo vzťahu 11.2 vyplýva, že s lineárnym rastom počtu vrcholov počet hrán a stien rastie taktiež lineárne.

11.2.1 Hranová reprezentácia

Každá hrana je určená dvojicou vrcholov. Zoznam všetkých takýchto dvojíc vrcholov predstavuje hranovú reprezentáciu – *drôtový model telesa*. Túto informáciu môžeme reprezentovať rôznymi spôsobami, napr. *maticou priľahlosti* (adjacency matrix), tj. maticou susednosti vrcholov

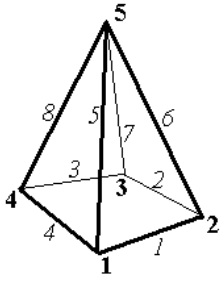
$$V = (v_{ij})\,, \quad v_{ij} = 1 \Longleftrightarrow i, j \text{ sú susedné vrcholy}\ ,$$

alebo incidenčnou maticou vrcholov a hrán (incidence matrix)

$$E = (e_{ij})\,, \quad e_{ij} = 1 \Longleftrightarrow \text{vrchol } i \text{ patrí hrane } j.$$

Pre štvorboký ihlan z obr. 11.2 sú tieto matice nasledujúce:

[1] Toto je nutná podmienka pre tzv. variety (manifolds), tj. také telesá, v ktorých okolie ľubovoľného bodu je topologicky ekvivalentné s kruhom príslušnej dimenzie. Príkladom nevarietného mnohostenu (nonmanifold) je napr. zjednotenie dvoch mnohostenov, ktoré majú spoločnú práve jednu hranu. Taká hrana je prienikom štyroch stien.



Obr. 11.2: Hranová reprezentácia (drôtový model) štvorbokého ihlanu. Aby sme odlíšili hrany od vrcholov, hrany značíme kurzívou.

$$V = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix} \qquad E = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

Keďže v každom stĺpci matice E sú práve dva nenulové členy, táto matica sa často reprezentuje ako zoznam dvojíc vrcholov[2]. V našom prípade

$$E' = \{(1,2),(2,3),(3,4),(1,4),(1,5),(2,5),(3,5),(4,5)\}. \tag{11.3}$$

Obe matice obsahujú tú istú informáciu (zo znalosti matice V vieme vyrobiť maticu E a naopak). Ktorý spôsob je použitý, závisí na konkrétnych požiadavkách manipulácie s telesami.

Vyššie zmienené štruktúry neobsahujú informáciu o stenách, ktoré sú pre popis mnohostenov a ich vizualizáciu veľmi dôležité. Preto sa používajú nasledujúce všeobecnejšie modely.

11.2.2 Plôšková reprezentácia

Stena je v našom prípade mnohouholník. Môžeme ju preto definovať ako postupnosť vrcholov, ktorá tvorí jej hranicu. V príklade z obr. 11.2 tak všetky steny môžeme vyjadriť množinou

$$\{(1,2,5),(2,3,5),(3,4,5),(1,4,5),(1,2,3,4)\}.$$

Musíme dbať na nasledujúce skutočnosti.

[2]Takto definované dvojice chápeme ako neusporiadané, tj (1,2) je to isté ako (2,1).

Orientácia povrchu

Každý rovinný n-uholník v 3D má práve dve strany. Pritom rozlíšenie týchto strán je dôležité. Prirodzenou požiadavkou býva napr. priradiť vnútorným a vonkajším stranám hraničných plôch (stien) rôznu farbu. Keďže postupnosť hraničných vrcholov môžeme zadať dvojako, buď v smere, alebo proti smeru chodu hodinových ručičiek, dá sa práve orientácia hranice steny využiť na rozlíšenie jej strán. V príklade z obr. 11.2 tak dostaneme výslednú stenovú reprezentáciu vonkajších strán (pri orientácii vrcholov proti smeru hodinových ručičiek a pohľade na teleso zvonku)

$$F' = \{(1,2,5)\,,(2,3,5)\,,(3,4,5)\,,(1,5,4)\,,(1,4,3,2)\}$$

resp. triangularizovanú reprezentáciu[3]

$$F'_{\triangle} = \{(1,2,5)\,,(2,3,5)\,,(3,4,5)\,,(1,5,4)\,,(1,3,2)\,,(1,4,3)\} \tag{11.4}$$

Rozklad plášťa na trojuholníkovú postupnosť

n-uholníková stena pre $n > 3$ nemusí byť automaticky rovinná. Preto pri vytváraní takýchto stien musíme ich rovinnosť kontrolovať. V prípade trojuholníkových stien je toto zaručené (samozrejme keď vrcholy nie sú kolineárne). Obmedzenie sa len na trojuholníkové steny zaručuje taktiež zjednodušenie dátovej štruktúry – stenová reprezentácia je tvorená *vždy usporiadanými trojicami vrcholov*. Toto sú dôvody častého používania mnohostenov s triangularizovaným plášťom. Nevýhodou je samozrejme väčšia pamäťová náročnosť: v nami popisovanom príklade štvoruholník $(1,4,3,2)$ nahradíme dvomi trojuholníkami, $(1,4,3)$ a $(1,3,2)$, alebo $(1,4,2)$ a $(2,4,3)$. Naviac, vykreslenie dvoch trojuholníkov môže byť pomalšie ako vykreslenie jedného štvoruholníka. Pre zvýšenie efektivity uchovávania i zobrazovania mnohostenov s trojuholníkovým plášťom sa preto používajú nie izolované trojuholníky, ale štruktúra pozostávajúca z

- trojuholníkových pásov (strips), kde $(P_0, P_1, P_2, P_3, \cdots, P_n)$ znamená postupnosť orientovaných trojuholníkov $(P_0, P_1, P_2)\,,\ (P_1, P_2, P_3)\,,\ (P_2, P_3, P_4)\,,\ \cdots, (P_{n-2}, P_{n-1}, P_n)$,
- trojuholníkových „vejárov“ (fans), kde $(P_0, P_1, P_2, P_3, \cdots, P_n)$ znamená postupnosť orientovaných trojuholníkov $(P_0, P_1, P_2)\,,\ (P_0, P_2, P_3)\,,\ (P_0, P_3, P_4)\,,\ \cdots,\ (P_0, P_{n-1}, P_n)$.

Okrídlená hrana

Všimnime si vzájomný vzťah hrán a stien: ako bolo zmienené v úvode, každá hrana je prienikom dvoch stien. Opačný vzťah natoľko priamočiary nie je. Napr. nie každá uzavretá postupnosť orientovaných hrán tvorí stenu (v našom príklade z obr. 11.2 hranám $(1,3)\,,(3,5)\,,(5,1)$, ktoré z reprezentácie $F'_{\triangle}$ dostaneme, nezodpovedá žiadna stena).

[3]Na rozdiel od (11.3) sú v (11.4) *usporiadané* trojice, čo znamená, že napr. trojica (1,2,5) je odlišná od (2,1,5). Ovšem v tomto kontexte (reprezentácia stien) usporiadané trojice získané cyklickou zámenou (1,2,5), (2,5,1), (5,1,2) chápeme ako totožné.

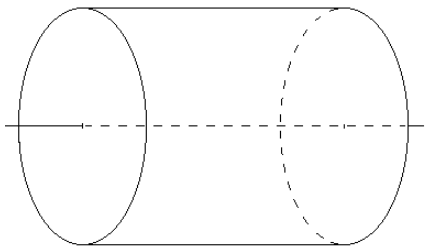
Preto je rozumné uchovávať topológiu mnohostenov primárne vo forme stien a hrany odvodiť ako ich prieniky. Kvôli rýchlosti pri manipulácii s mnohostenom je však výhodnejšie (aj za cenu redundancie dát, a tým aj väčších pamäťových nárokov) informáciu o topológii neodvodzovať, ale mať ju plne dostupnú. To vedie k štruktúre *okrídlená hrana* (winged edge) v ktorej máme odkazy jednak na steny, prienikom ktorých hrana vzniká, jednak na vrcholy, ktoré hranu definujú. Napr. hrana *5* z obr. 11.2 odkazuje na vrcholy 1 a 5 a na steny (1,5,2) a (5,1,4)[4]. Výhoda tejto reprezentácie bude ukázaná pri riešení viditeľnosti.

11.3 Parametrické modely

V kap. 10 bol ukázaný spôsob generovania kriviek na základe ich polynomického parametrického vyjadrenia. Tento prístup možno zovšeobecniť na vyjadrenie plôch a objemov.

11.3.1 Šablónovanie

Predstavme si krivku v 3D priestore. Jej pohybom po vopred zadanej trajektórii vznikne plocha. Napr. vezmime kružnicu a posúvajme jej stredom po úsečke kolmej na rovinu kružnice. Výsledkom je plášť valca, obr. 11.3.



Obr. 11.3: Šablónovanie kružnice po kolmej trajektórii.

Ponúkajú sa rôzne formy zovšeobecnenia tohto prístupu:

1. namiesto kružnice uvažujme ľubovoľnú rovinnú krivku – *profil*,
2. namiesto stredu kružnice definujme v rovine profilu ľubovoľný bod – *bod úchytu*,
3. namiesto úsečky uvažujme ľubovoľnú krivku – *os šablónovania*, po ktorej sa bod úchytu pohybuje,

[4]Tu vidíme, prečo hranová reprezentácia 3D modelov uvažuje neorientované hrany. Na priľahlých susedných stenách je hrana opačne orientovaná: na stene (1,5,2) je to (1,5), na stene (5,1,4) je to (5,1). Nie je dôvod niektorú z orientácií uprednostňovať. Je však treba rozlíšiť tento 3D prípad od 2D prípadu, kde hranicu mnohouholníka, tj. postupnosť hrán, orientujeme.

4. v každom momente pohybu definujme vzťah roviny profilu a osi šablónovania.

Prirodzeným obmedzením pre profil sú *rovinné krivky.* Podobne i os šablónovania býva často *rovinná krivka.* Je to kvôli názornosti a jednoduchosti editácie takých kriviek.

Najčastejšie sa využíva konštantný uhol (kolmý) medzi rovinou profilu a osou šablónovania.

V prípade, že osou šablónovania je úsečka, hovoríme o *posuvnom šablónovaní.* Keď je osou šablónovania kružnica, s polomerom, ktorý sa limitne blíži k nule, hovoríme o *rotačnom šablónovaní.*

Pri zovšeobecňovaní môžeme pokračovať ďalej, napr.

a) v každom momente pohybu uvažujme možnosť zmeny profilu,

b) v každom momente pohybu uvažujme rotáciu roviny profilu okolo bodu úchytu,

... atď.

Takýto spôsob modelovania povrchov sa často označuje ako 2.5D modelovanie. Využíva sa v priemyselných aplikáciách, napr. pri definovaní tvarov dosiahnuteľných obrábacími strojmi. Ďalšie dôležité aplikácie súvisia s analýzou kolízie priemyselných robotov, keď vyššieopísaným spôsobom sa definuje objem, v ktorom sa pohybuje robot, resp. jeho časti.

11.3.2 Priamkové plochy

Uvažujme dve krivky zadané parametricky $\mathcal{P}_0 = P_0(t)$, $\mathcal{P}_1 = P_1(t)$, $t \in \langle 0,1\rangle$.[5] Pre fixovanú hodnotu parametra $t = t^*$ dostaneme na krivkách body $P_0^* = P(t^*)$, $Q_0^* = Q(t^*)$. Spojme tieto body úsečkou $S(u) = (1-u)P_0^* + uP_1^*$, $u \in \langle 0,1\rangle$. Keď túto konštrukciu spravíme pre všetky možné fixované hodnoty $t \in \langle 0,1\rangle$, dostaneme plochu

$$S(t,u) = (1-u)P_0(t) + uP_1(t) \qquad t,u \in \langle 0,1\rangle,$$

ktorú nazývame *priamková plocha.*

Takýto spôsob konštrukcie je z algoritmického hľadiska jednoduchý, no geometrické vlastnosti výslednej plochy môžu byť dosť netriviálne i v prípade veľmi jednoduchých východiskových kriviek. Napr. uvažujme hraničné krivky kružnice

$$\mathcal{P}_0(\phi) = \{(x,y,z) :\ x(\phi) = r\cos(\phi),\ y(\phi) = r\sin(\phi),\ z = z_0, \quad \phi \in \langle 0,2\pi\rangle\},$$
$$P_1(\phi) = \{(x,y,z) :\ x(\phi) = r\cos(\phi),\ y(\phi) = r\sin(\phi),\ z = z_1, \quad \phi \in \langle 0,2\pi\rangle\}.$$

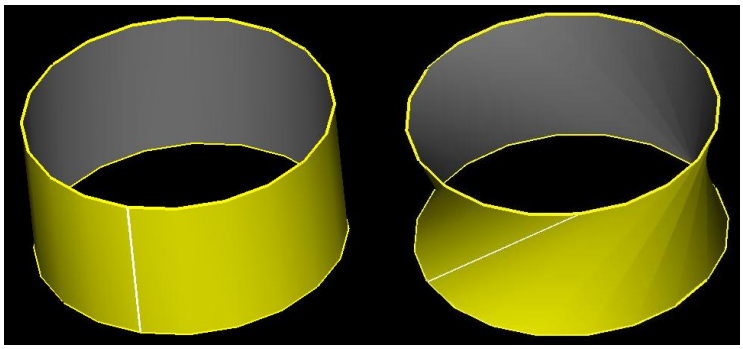
Úsečkou budeme spájať body $P_0(\phi), P_1(\phi)$. Dostaneme tak valcovú plochu, obr. 11.4a).

[5] Tj. súradnice bodu $P(t)$ sú $x = x(t), y = y(t), z = z(t)$.

V prípade, že hraničné krivky sú kružnice

$$\begin{aligned}\mathcal{P}_0(\phi) = &\quad \{(x,y,z) :\ x(\phi) = r\cos(\phi)\,,\ y(\phi) = r\sin(\phi)\,,\ z = z_0, \quad \phi \in \langle 0, 2\pi\rangle\}\\ P_1(\phi) = &\quad \{(x,y,z) :\ x(\phi) = r\cos(\phi+\alpha)\,,\ y(\phi) = r\sin(\phi+\alpha)\,,\ z = z_1,\\ &\quad \alpha = const \neq 0,\ \phi \in \langle 0, 2\pi\rangle\}\end{aligned}$$

spájaním bodov $P_0(\phi), P_1(\phi)$ dostaneme jednodielny rotačný hyperboloid¸ obr. 11.4b).



a) b)

Obr. 11.4: Priamková plocha dvoch paralelných kružníc rovnakého polomeru. a) valec – spájané body sú rovnobežné so spojnicou stredov kružníc, b) jednodielny rotačný hyperboloid – spájané body sú mimobežné so spojnicou stredov kružníc.

Príklad ukazuje rozdiel medzi parametrickou interpoláciou a šablónovaním: pri šablónovaní kružnice s bodom úchytu v jej strede a s úsečkou, ako osou šablónovania, dostaneme vždy valcovú plochu.

Ukážeme ešte jeden príklad – najjednoduchší možný prípad priamkovej interpolácie, keď $\mathcal{P}_0$, $\mathcal{P}_1$ sú úsečky definované bodami P_{00}, P_{01} a P_{10}, P_{11}. Výsledkom je bilineárna interpolačná plocha

$$S(u,v) = (1-u)\left((1-v)P_{00} + vP_{01}\right) + u\left((1-v)P_{10} + vP_{11}\right) \qquad u,v \in \langle 0,1\rangle\,. \quad (11.5)$$

Vzťah (11.5) môžeme vyjadriť v maticovom tvare

$$S(u,v) = (1-u,u)\begin{pmatrix}(1-v)P_{00} + vP_{01}\\ (1-v)P_{10} + vP_{11}\end{pmatrix} = (1-u,u)\begin{pmatrix}P_{00} & P_{01}\\ P_{10} & P_{11}\end{pmatrix}\begin{pmatrix}1-v\\ v\end{pmatrix}. \quad (11.6)$$

V prípade, že $\mathcal{P}_0$, $\mathcal{P}_1$ sú mimobežné, dostávame netriviálny výsledok – kvadratickú plochu *hyperbolický paraboloid,* čo ilustruje nasledujúci príklad.

Nech riadiace vrcholy bilineárnej plochy majú súradnice:

$$P_{00} = (0,-1,1)\,, \quad P_{01} = (1,0,-1)\,, \quad P_{10} = (-1,0,-1)\,, \quad P_{11} = (0,1,1)\,.$$

Dosadením týchto bodov do (11.5) dostávame pre súradnice nasledujúce vzťahy:

$$\begin{aligned} x(u,v) &= v-u \\ y(u,v) &= v+u-1 \\ z(u,v) &= 1-2u-2v+4uv \end{aligned}$$

Pokúsime sa vyjadriť plochu v explicitnom tvare, (v tvare $z = z(x,y)$, tj. bez použitia parametrov u, v): z prvých dvoch rovníc vyjadríme u, v na základe súradníc x, y.

$$u = \frac{y-x+1}{2}, \quad v = \frac{y+x+1}{2}.$$

Dosadením do 3. rovnice a roznásobením dostávame

$$z = y^2 - x^2.$$

Vidíme, že napr. rez tejto plochy rovinou $x = c$ je parabola $z = y^2 - a$, kde $a = c^2$. Podobne rez kolmou rovinou $y = c$ dáva parabolu $z = a - x^2$. Na druhej strane, rez rovinou $z = c$ je hyperbola $y^2 - x^2 = c$ (odtiaľ názov hyperbolický paraboloid) – obr. 11.5[6]. .

Obr. 11.5: Hyperbolický paraboloid.

11.3.3 Polynomická parametrizácia a jej maticové vyjadrenie

Vyššie opísanú konštrukciu trochu zovšeobecníme – body hraničných kriviek nebudeme spájať úsečkami, ale krivkami. Z dôvodu jednotného popisu a manipulácií, je rozumné používať jednu triedu kriviek. Ilustrujeme to na príklade bikubických Bézierových plátov.

[6]Prevzaté z http://mathcraft.wonderhowto.com/how-to/make-hyperbolic-paraboloid-using-skewers-0131751/.

Majme dve kubické Bézierové krivky. Nech $\mathcal{B}_0 = B_0(v)$ je určená riadiacimi bodami $P_{00}, P_{01}, P_{02}, P_{03}$ a $\mathcal{B}_3 = B_3(v)$ riadiacimi bodami $P_{30}, P_{31}, P_{32}, P_{33}$. V oboch prípadoch $v \in \langle 0,1 \rangle$.

Pre fixovanú hodnotu $v^\star \in \langle 0,1 \rangle$ chceme spájať body $B_0(v^\star)$, $B_3(v^\star)$ opäť Bézierovou kubickou krivkou. Potrebujeme k tomu ďalšie dva riadiace body $B_1(v^\star)$, $B_2(v^\star)$. Je zmysluplné predpokladať, že tieto body ležia na príslušných Bézierových krivkách $\mathcal{B}_1 = B_1(v)$, $\mathcal{B}_2 = B_2(v)$, ktoré sú určené bodami $P_{10}, P_{11}, P_{12}, P_{13}$ a $P_{20}, P_{21}, P_{22}, P_{23}$.

Pre fixovanú hodnotu $v^\star \in \langle 0,1 \rangle$ tak dostávame vyjadrenie Bézierovej krivky

$$S(u, v^\star) = \sum_{i=0}^{3} \left(f_i(u) B_i(v^\star) \right), \qquad u \in \langle 0,1 \rangle, \tag{11.7}$$

Zostáva už iba neuvažovať len fixovanú hodnotu $v^\star$, ale „uvoľniť" parameter $v \in \langle 0,1 \rangle$.

$$S(u, v) = \sum_{i=0}^{3} \left(f_i(u) B_i(v) \right), \qquad u, v \in \langle 0,1 \rangle, \tag{11.8}$$

Keďže $\mathcal{B}_i = B_i(v)$ sú Bézierové krivky, $\mathcal{B}_i = B_i(v) = \sum_{j=0}^{3} f_j(v) P_{ij}$, priamym dosadením do (11.8) dostávame

$$S(u, v) = \sum_{i=0}^{3} \left(f_i(u) \sum_{j=0}^{3} f_j(v) P_{ij} \right), \qquad u, v \in \langle 0,1 \rangle, \tag{11.9}$$

kde $f_j(t)$ sú Bernsteinove polynómy 3. stupňa (10.4).

Z konštrukcie vidíme, že výsledná plocha je určená šestnástimi riadiacimi bodami P_{ij}, $i, j = 0, \ldots, 3$. Podobne, ako pre bilineárnu plochu (11.5), súradnice bodov plochy sa dajú zapísať v maticovom tvare

$$\begin{aligned}
S(u,v) &= \begin{pmatrix} f_0(u) & f_1(u) & f_2(u) & f_3(u) \end{pmatrix} \begin{pmatrix} B_0(v) \\ B_1(v) \\ B_2(v) \\ B_3(v) \end{pmatrix} = \\
&= \begin{pmatrix} f_0(u) & f_1(u) & f_2(u) & f_3(u) \end{pmatrix} \begin{pmatrix} \sum_{j=0}^{3} f_j(v) P_{0j} \\ \sum_{j=0}^{3} f_j(v) P_{1j} \\ \sum_{j=0}^{3} f_j(v) P_{2j} \\ \sum_{j=0}^{3} f_j(v) P_{3j} \end{pmatrix} = \\
&= \begin{pmatrix} f_0(u) & f_1(u) & f_2(u) & f_3(u) \end{pmatrix} \begin{pmatrix} P_{00} & P_{01} & P_{02} & P_{03} \\ P_{10} & P_{11} & P_{12} & P_{13} \\ P_{20} & P_{21} & P_{22} & P_{23} \\ P_{30} & P_{31} & P_{32} & P_{33} \end{pmatrix} \begin{pmatrix} f_0(v) \\ f_1(v) \\ f_2(v) \\ f_3(v) \end{pmatrix}
\end{aligned}$$

Použijúc postup, ako v kap. 10.4.1,

$$(f_0(u) \quad f_1(u) \quad f_2(u) \quad f_3(u)) = \begin{pmatrix} 1 & u & u^2 & u^3 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{pmatrix} = \mathcal{U}\mathcal{L}$$

dostávame výsledný maticový tvar

$$S(u,v) = \mathcal{U}\mathcal{L}\Pi_2 (\mathcal{V}\mathcal{L})^T = \mathcal{U}\mathcal{L}\Pi_2\mathcal{L}^T\mathcal{V}^T, \qquad u,v \in \langle 0,1\rangle, \tag{11.10}$$

kde $\mathcal{M}^T$ značí transponovanú maticu k matici $\mathcal{M}$,

$$\mathcal{U} = \begin{pmatrix} 1 & u & u^2 & u^3 \end{pmatrix}, \quad \mathcal{V} = \begin{pmatrix} 1 & v & v^2 & v^3 \end{pmatrix},$$

$\mathcal{U}, \mathcal{V}$ sú vektory parametrov v polynomickej báze,

$$\mathcal{L} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{pmatrix}, \quad \Pi_2 = \begin{pmatrix} P_{00} & P_{01} & P_{02} & P_{03} \\ P_{10} & P_{11} & P_{12} & P_{13} \\ P_{20} & P_{21} & P_{22} & P_{23} \\ P_{30} & P_{31} & P_{32} & P_{33} \end{pmatrix}.$$

$\mathcal{L}$ je matica parametrizácie krivky a Π_2 je matica riadiacich vrcholov.

Podobne, ako sme prešli od parametrickej krivky k parametrickej ploche, môžeme prejsť od parametrickej plochy k parametrickému objemu. Tentokrát potrebujeme tri parametre t, u, v a n^3 riadiacich bodov (za predpokladu použitia polynómov n-tého stupňa). Obmedzíme sa len na formálny zápis analogický vyjadreniu (11.9):

$$V(t,u,v) = \sum_{i=0}^{n} \left(f_i(t) \sum_{j=0}^{n} f_j(u) \left(\sum_{k=0}^{n} f_k(v) P_{ijk} \right) \right), \qquad u,v \in \langle 0,1\rangle. \tag{11.11}$$

Maticový tvar zápisu bohužiaľ v tomto prípade stráca na prehľadnosti.

Mechanizmus parametrizácie nesie v sebe prvky vektorovo orientovaných modelov (krivka, plocha objem sú určené konečným počtom riadiacich bodov a nezávisia na jemnosti použitého rastru). Ukážeme si však, že parametrizácia nám dáva nástroj na určité zovšeobecnenie rastrového prístupu k popisu modelov.

11.4 Zovšeobecnená rasterizácia

V trojrozmernom priestore je ekvivalentom rastra (rozkladu obdlžníka na pixle) rozklad jednotkového kvádra na elementárne kvádre zvané *voxely* (VOLume ELement). Zariadenie ktoré po spracovaní dáva výstupnú informáciu práve takýmto trojrozmerným rastrovým spôsobom je napr. počítačový tomograf. V podobnej forme bývajú reprezentované výsledky numerických simulácií fyzikálnych modelov z oblasti mechaniky kontinua. A práve numerické modelovanie tohto typu úloh (metóda konečných prvkov) bolo silným impulzom pre metódy, ktoré môžeme vnímať ako zovšeobecnenie rastrového prístupu.[7]

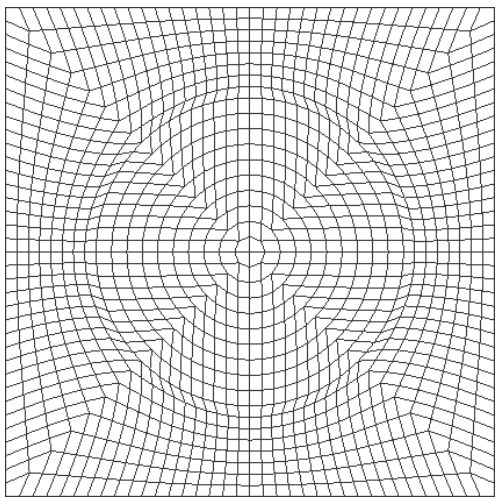
Nižšieuvedené metódy sú pre jednoduchosť formulované v 2-rozmernej verzii. Dajú sa však preniesť i do 3D priestoru.

11.4.1 Štruktirovaná sieť

Keď vo vzťahu (11.10) budeme uvažovať diskrétne hodnoty parametrov $0 = u_0 < u_1 < \cdots < u_n = 1$, $0 = v_0 < v_1 < \cdots < v_m = 1$, tak systém kriviek

$$\{S(u_i, v),\, S(u, v_j),\, 0 \leq i \leq n,\, 0 \leq j \leq m,\, u, v \in \langle 0, 1 \rangle\} \tag{11.12}$$

rozbíja plochu $S(u, v)$ na množinu *elementárnych plôšok*. Tie môžeme, podobne ako v prípade rasterizácie, vzájomne odlíšiť dvojicou indexov i, j. K takému rozkladu priestoru môžeme pristupovať podobne, ako k rastrovému obrazu: každú elementárnu plôšku môžeme považovať za „nedeliteľný pixel“. (11.12) nazývame *štrukturovaná sieť*.



Obr. 11.6: Štrukturovaná sieť.

[7] Veľmi zjednodušene povedané, skúmaná oblasť sa rozloží na materiálovo homogénne jednoduché elementy a skúma sa interakcia medzi nimi, ktorá je daná vopred určeným typom rovníc.

Okrem vyššie zmienej využiteľnosti pre numerické modelovanie, dáva uvedená konštrukcia nástroj na deformáciu obrazu a býva zahrnutá do softvérov na spracovanie obrazu práve pod názvom *image warping*.

11.4.2 Triangulácia

Princíp rozkladu oblasti na elementárne prvky môžeme ďalej zovšeobecniť.

Uvažujme napr. ľubovoľný ohraničený n-uholník $\mathcal{P}$ v 2D, určený vrcholami $P_1, P_2, \ldots, P_n$ a konečnú množinu bodov $Q_1, Q_2, \ldots, Q_m$ vo vnútri n-uholníka (pripúšťame i prázdnu množinu bodov). Pokryjme n-uholník $\mathcal{P}$ systémom trojuholníkov takých, že vrcholy trojuholníkov sú z množiny $\{P_1, P_2, \ldots, P_n, Q_1, Q_2, \ldots, Q_m\}$ a každá dvojica trojuholníkov buď nemá spoločný prienik, alebo sa dotýka práve v jednom bode, alebo má spoločnú celú hranu. Tejto konštrukcii hovoríme triangularizácia oblasti $\mathcal{P}$, resp. *obecná neštrukturovaná (trojuholníková) sieť*.

Podobným spôsobom môžeme vyrábať i neštrukturované štvoruholníkové siete.

Rozklad na trojuholníky spravidla nebýva jednoznačne určený, preto zákonite vzniká úloha nájsť najlepší možný rozklad vzhľadom k stanovenému kritériu. Väčšinou sa vyžaduje, aby trojuholníky boli podľa možností „čo najviac rovnostranné“.[8] Viac o tejto problematike nájde čitateľ napr. v [?].

11.5 Hierarchické modely

S myšlienkou hierarchie sme sa stretli pri komprimácii rastrového obrazu na základe kvadrantového stromu. Ide vlastne o viacúrovňový pravidelný rozklad oblasti na elementárne prvky, kde na „hrubšej“ úrovni hodnotu pixelu priemerujeme z hodnôt odpovedajúcich pixelov na jemnej úrovni. Pritom každá úroveň je schopna obsiahnúť väčšiu mieru detailu. V 3D rastrovom prístupe nazývame túto konštrukciu *oktálový strom* (octree).

Viacúrovňovosť je veľmi dôležitá aj vo vektorovej grafike. Príkladom viacúrovňového prístupu je schéma de Casteljau pre generovanie Bézierovej krivky (kap. 10.4.2).

Na základe viacúrovňovosti dokážeme urýchliť mnohé analýzy. Všeobecnú schému riešených problémov tohto typu ilustrujeme na úlohe nájsť prienik povrchu telesa a priamky[9].

Namiesto priameho hľadania takého prieniku zisťujeme najprv, či tento prienik vôbec existuje: uvažujeme akési zjednodušené obaľujúce teleso, napr. rotačný elipsoid alebo kváder,

[8] Často používané sú Delaunayovské triangulácie: na množine vrcholov skonštruujeme triangluláciu $\mathcal{T}$. Nájdeme minimálny uhol v rámci všetkých trojuholníkov tejto triangulácie $\alpha_{\mathcal{T}}$. Skonštruujeme inú triangluláciu $\mathcal{S}$ a opäť hľadáme najmenší uhol $\alpha_{\mathcal{S}}$. Ak $\alpha_{\mathcal{S}} > \alpha_{\mathcal{T}}$, hovoríme, že triangulácia $\mathcal{S}$ je lepšia ako $\mathcal{T}$. Delaunayovská triangulácia je zo všetkých možných najlepšia.

[9] S touto úlohou sa stretneme pri globálnych osvetľovacích modeloch – kap. 12.3.1. Úloha podobného typu – nájsť prienik dvoch telies, je dôležitá napr. pri riešení kolízie objektov.

a hľadá sa prienik priamky s takto zjednodušeným povrchom. Skutočný prienik sa hľadá len v prípade, že prienik s obaľujúcim telesom existuje. Predpokladáme pri tom, že nájsť prienik s jednoduchým obaľujúcim telesom je podstatne jednoduchšie ako nájsť prienik so samotným telesom. Preto tento prístup bude účinný, keď len malá časť priamok ktoré prichádzajú do úvahy, má prienik s obaľujúcim telesom.

Z uvedeného vidíme, že je zmysluplné vytvárať čo najmenšie obaľujúce telesá, prípadne používať viacúrovňové obaľujúce telesá. To vedie v konečnom dôsledku k viacúrovňovým rozkladom tj. k *hierarchickým modelom*.

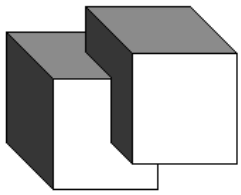
11.6 Záverečné poznámky

11.6.1 Povrchová versus objemová reprezentácia

Pre zobrazenie CSG modelov musíme vygenerovať ich výsledný povrch[10]. Tu však už i v jednoduchých prípadoch môžu nastať problémy.

Uvažujme napr. triedu jednoduchých telies – kvádrov, hrany ktorých sú rovnobežné so súradnicovými osami. Každý taký kváder Q je jednoznačne určený šiestimi číslami, tj. $Q = Q(x, y, z, c_x, c_y, c_z)$ kde x, y, z sú dĺžky hrán, c_x, c_y, c_z sú súradnice stredu kvádra.

Majme napr. regularizované zjednotenie dvoch kvádrov[11] $Q(2, 2, 2, 0, 0, 0) \tilde{\cup} Q(2, 2, 2, 1, 1, 1)$ obr. 11.7



Obr. 11.7: Povrchováreprezentácia zjednotenia dvoch pretínajúcich sa kvádrov.

Vidíme, že pre generovanie výsledného povrchu potrebujeme analyzovať vzájomný vzťah povrchov oboch telies – musíme nájsť vzájomné prieniky, niektoré zo stien pritom musíme nahradiť len ich časťami.

Ak by sme uvažovali zložitejšie objekty, napr. objemy ohraničené Bézierovskými plátmi, nájdenie priesečníkov vedie na riešenie sústavy nelineárnych rovníc, čo spravidla nie je jednoduché.

[10] Existujú i zobrazovacie metódy, kde nemusíme vopred vyrábať povrchovú reprezentáciu, napr. Ray-Tracing – viď nasledujúca kapitola.

[11] V tomto prípade je samozrejme regularizované zjednotenie totožné s prostým zjednotením.

Môže nám pomôcť zovšeobecnená rasterizácia: zvolíme taký rozklad priestoru, aby každé z modelovaných telies bolo v tomto rozklade dostatočne dobre rasterizované. A na rasterizovaných rozkladoch sa dá jednoducho aplikovať zjednotenie, prienik a rozdiel.

11.6.2 Parametrizácia ako animačný nástroj

Prechodu od dvojice parametrizovaných kriviek k ploche možno dať aj inú interpretáciu ako bolo uvedené v kap. 11.3.2. Na uvedený proces sa môžeme pozerať tak, že parameter $u \in \langle 0, 1 \rangle$ vnímame ako čas daný mechanizus môžeme interpretovať ako *postupnú zmenu pôvodnej krivky* $B_0(v)$ *na výslednú krivku* $B_3(v)$. Podobne môžneme interpretovať parametrický objem ako animáciu prechodu jednej hraničnej plochy na druhú.

Dostávame tak dôležitú animačnú techniku – *kľúčovanie*: meniaci sa objekt (krivka, resp. plocha) je zadaný v niekoľkých časových krokoch. V „medzičasoch“ je jeho tvar interpolovaný. Spravidla sa používa lineárna interpolácia (čomu zodpovedá priamková plocha, resp. priamkový objem).

11.6.3 Izolínie a izoplochy

Častou úlohou ktorá sa rieši práve na modeloch zovšeobecnenej rasterizácie, je generovanie izolínií a izoplôch. Ide o to, že každému uzlu siete je priradená skalárna veličina (spočítaná alebo nameraná) a našou úlohou je vykresliť oblasť s požadovanou hodnotou tejto veličiny. Vytvárame tak vlastne geometrický model akéhosi „abstraktného“ telesa. Jednoduchým riešením je napr. zahrnúť do výslednej oblasti len tie elementy, u ktorých všetky uzly majú požadovanú hodnotu.

Výsledok môžeme „vyhladiť“ takým spôsobom, že v elementoch, v ktorých len niektoré uzly majú požadovanú hodnotu, nachádzame hranicu oblasti interpoláciou hodnôt vo vrcholoch.

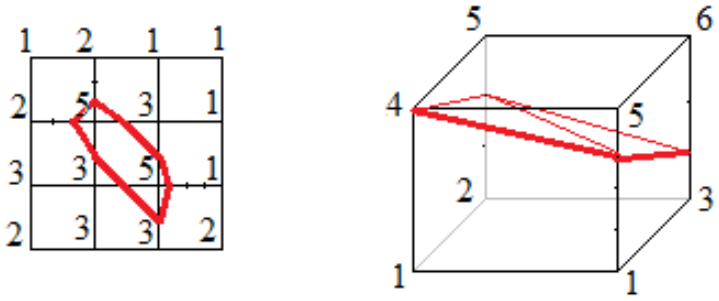
Na tomto princípe je založena veľmi populárna metoda Marching Cubes (prvýkrát použitá na vyhladenie 3D povrchov v objektoch z tomografických dát [?].

11.6.4 Duálne rozklady

Dôležitým pojmom je *duálny rozklad*. Nech je daný rozklad oblasti na elementy.

1. Každému elementu priradíme vrchol (napr. ťažisko).
2. Vrcholy spojíme práve vtedy, keď elementy pôvodného rozkladu sú susedné.

Duálne rozklady nám niekedy pomôžu jednoduchšie pochopiťvzájomné väzby medzi elementami pôvodného rozkladu. Je dôležité si uvedomiť, že napr. duálnym rozkladom štrukturovanej siete je opäť štrukturovaná sieť.



Obr. 11.8: Generovanie a) izolínie, b) izoplochy s hodnotou 4. Čísla označujú hodnoty v príslušných vrcholoch siete. Dodatočné priesečníky sú nájdené lineárnou interpoláciou hodnôt vo vrcholoch.

11.6.5 Ďalšie typy modelov

Existujú ďalšie prístupy k tvorbe modelov, napr. procedurálne modely, modely založené na implicitných plochách. Podľa mienky autora však ich popis presahuje základy počítačovej grafiky, preto nie sú zahrnuté do tohto textu . Záujemci nájdu informáciu o týchto prístupoch napr. v [29].

11.7 Zhrnutie a úlohy

- Ukázali sme si základný formalizmus používaný na vyjadrenie mnohostenov.
- Boli sformulované princípy šablónovania.
- Najväčší priestor bol venovaný parametrickým modelom.
- Ukázali sme tri pohľady na parametrické modely: a) parametrické modely ako modely vektorového typu, b) parametrizácia ako nástroj pre zovšeobecnenie rastrového prístupu k modelom, c) parametrizácia ako nástroj animácie.
- Zmienili sme základné problémy povrchovej a objemovej reprezentácie.

Otázky

1. Prečo je drôtový model mnohostenov nedostačujúci ?
2. Vytvorte štruktúru typu okrídlená hrana pre ihlan z obr. 11.2.
3. Ako by ste vyjadrili jednodielny rotačný hyperboloid na základe šablónovania?

4. Aká je interpretácia Bézierovych kriviek, určených štvoricami riadiacich bodov

$$P_{00}, P_{01}.P_{02}, P_{03}, \quad P_{30}, P_{31}.P_{32}, P_{33}, \quad P_{00}, P_{10}.P_{20}, P_{30}, \quad P_{03}, P_{13}.P_{23}, P_{33}$$

   v rámci plochy (11.9)?

5. Čo je to štrukturovaná sieť?

6. Prečo sú množinové operácie na rasterizovaných objektoch jednoduchšie ako na objektoch zadaných analyticky, resp. parametricky?

Kapitola 12

Vizualizácia 3D objektov

Predstavme si jednoduché teleso – kolmý kváder. Ako také teleso formálne popísať, to sme sa naučili v predošlej kapitole. Ako s tým telesom pohybovať, sme sa dozvedeli ešte skôr. Ako však určiť, ktoré steny sú viditeľné a ktoré nie, to zatiaľ nevieme. A nevieme ešte jednu podstatnú vec: predpokladajme, že máme kváder, ktorého všetky steny i hrany sú rovnakej farby. Predpokladajme taktiež, že vieme, ktoré steny sú viditeľné. Pri zobrazovaní kvádra (tak, ako to doteraz vieme) bude výsledným obrazom farebne homogénny buď štvoruholník alebo šesťuholník. Čiže zatiaľ nemáme zvládnuté metódy, na základe ktorých bude výsledný dvojrozmerný obraz pôsobiť dojmom skutočne trojrozmerného telesa.

Práve týmto problémom sa v tejto kapitole budeme venovať.

Nebudeme uvažovať priehľadné a polopriehľadné telesá. Preto sa môžeme obmedziť len na povrchové reprezentácie vizualizovaných objektov. Budú nás zaujímať nielen jednotlivé objekty, ale celé skupiny objektov – scény.

Predpokladáme zobrazovanie scén na štandardné výstupné zariadenie (obrazovka, tlačiareň), čo znamená, že chceme zobrazovať *rovinné projekcie* týchto scén. Preto nie je obmedzujúce zvoliť súradnicový systém tak, aby celá scéna bola v polpriestore $z > 0$ a z-tová súradnica pozorovateľa bola záporná. Za priemetňu zvolíme rovinu $z = 0$. Obmedzíme sa, keď to nebude výslovne uvedené ináč, na kolmé premietanie.

12.1 Viditeľnosť

Riešeniu viditeľnosti bola v procese vývoja počítačovej grafiky venovaná veľká pozornosť. Výsledkom je množstvo rôznych algoritmov, navrhnutých a optimalizovaných pre scény so špecifickými vlastnosťami. Našim cieľom je sformulovať podstatné vlastnosti a základné rysy týchto algoritmov. Detailnejší prehľad algoritmov možno nájsť napr. v[7],[29]. Podrobná analýza algoritmov viditeľnosti je v [3].

Riešenie viditeľnosti môžeme konštruovať v objektovom priestore (vektorový prístup) alebo v obrazovom priestore (rastrový prístup).

12.1.1 Vektorovo orientované algoritmy

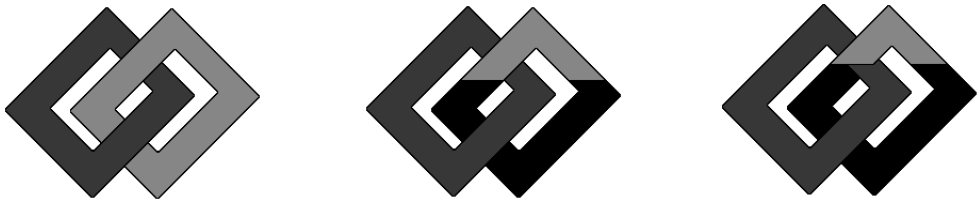
V týchto metódach je základným elementom povrchová plocha objektu scény, resp. jej časť. Keďže z celkového počtu n plôch[1] v scéne vidíme len tie (resp. ich časti), ktoré sú „najviac vpredu“, a každé teleso môže zakrývať len tie telesá, ktoré sú za ním, z formálneho hľadiska problém viditeľnosti súvisí s problémom usporiadania. Ako ukážeme, nie vždy možno usporiadať všetky povrchy v scéne. Práve tento fakt je podstatou problému, ktorý musíme riešiť.

12.1.1.1 Algoritmus hĺbkového triedenia

Postupujeme nasedujúcim spôsobom.

```
1. Povrchové plochy usporiadaj zostupne podľa súradnice z.
2. Pre každú povrchovú plochu (v poradí danom usporiadaním):
3. { Vytvor priemet.
4.   Priemet vykresli do videopamäte.
5. }
```

V prípade, že prvý krok uvedenej schémy sme schopní zrealizovať, dostávame algoritmus, v literatúre označovaný ako *maliarov algoritmus*. Priamo z formulácie vidíme, že vzhľadom na to, že kroky 3 a 4 realizujeme lineárnou zložitosťou (jedným priechodom) zložitosť celého algoritmu je $O(nlog(n))$ (je skoncentrovaná v prvom kroku algoritmu).



a) b) c)

Obr. 12.1: a) Príklad dvojice plôch, ktoré usporiadať nemožno. b) Vhodným rozdelením môžeme plochy usporiadať: čierna je najvzdialenejšia, svetlošedá je najbližšie. c) Nevhodné rozdelenie pravej plochy.

Problém je však v tom, že nie vždy je možné plochy takto usporiadať, viď napr. obr. 12.1 a). Riešením je rozdelenie plôch na menšie tak, aby usporiadanie bolo možné. Toto však nebýva bezproblémové. Napr. pre plochy z obr. 12.1 b) nemožno ľavú horizontálnu hranicu čiernej oblasti posunúť vyššie. V takom prípade (obr. 12.1 c)) aplikácia hĺbkového triedenia nefunguje správne.

K pochopeniu mechanizmu korektného delenia plôch musíme sa na problém viditeľnosti pozrieť z trochu inej strany.

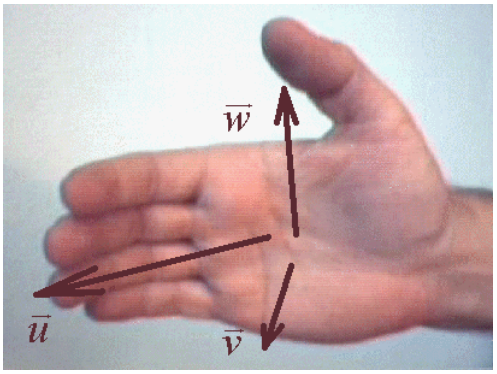
[1] V celej kapitole o viditeľnosti n značí počet stien zobrazovanej scény.

12.1.1.2 Viditeľnosť konvexného mnohostenu

S konvexným telesom sme sa už stretli v predošlých kapitolách niekoľkokrát. A predstaviť si konvexný mnohosten môžeme tak, že nech sa naň pozeráme z akéhokoľvek miesta, tak povrchové steny buď vidíme celé, alebo nevidíme vôbec. Tj. nemôže nastať situácia, že by sme videli len časť niektorej povrchovej steny. (V prípade, keď je stena kolmo k nám, tj. vidíme z nej len niektoré hraničné úsečky, považujeme stenu za neviditeľnú.) Kritérium viditeľnosti sa v tomto prípade dá sformulovať tak, že

- *vidíme práve tie steny, ktorých vonkajšie normály „smerujú proti nám“.*

Pojem *vonkajšia normála* súvisí s orientáciou plochy – (viď predošlá kapitola). Vonkajšiu normálu rovinnej plochy $(P_1P_2P_3\dots)$ orientovanej tak, ako v prípade (11.4) z obr. 11.2, dostaneme ako vektorový súčin $(P_2 - P_1) \times (P_3 - P_1)$.



Obr. 12.2: Orientácia vektorového súčinu. Keď prsty pravej ruky ukazujú v smere vektora $\vec{u}$ a vektor $\vec{v}$ vychádza z dlane, potom ich vektorový súčin $\vec{w} = \vec{u} \times \vec{v}$ je kolmý zároveň na $\vec{u}$ a $\vec{v}$ a je orientovaný tým smerom, kam ukazuje palec. Z uvedeného vidíme, že poradie vektorov je dôležité.

Formulácia *„vektor smeruje proti nám“* znamená, že uhol medzi týmto vektorom a smerom (vektorom) nášho pohľadu je tupý. To je ekvivalentné tomu, že skalárny súčin týchto vektorov je záporný.

Vychádzajúc zo súradnicového systému určeného v úvode kapitoly a pri použití kolmej projekcie, dostávame pre rovinnú stenu kritérium viditeľnosti[2]:

- *stena s vonkajšou normálou* $\vec{n} = (n_x, n_y, n_z)$ *je viditeľná práve vtedy, keď* $n_z < 0$.

Nie je ťažké na tomto základe sformulovať algoritmus pre riešenie viditeľnosti. Je dôležité si uvedomiť, že algoritmus je jednopriechodový.

[2]Algoritmus založený na tomto kritériu sa v literatúre vyskytuje pod názvom Back-Volume Culling Algorithm (BVCA).

Na základe tohto kritéria je možné klasifikovať hrany konvexného telesa do troch tried:

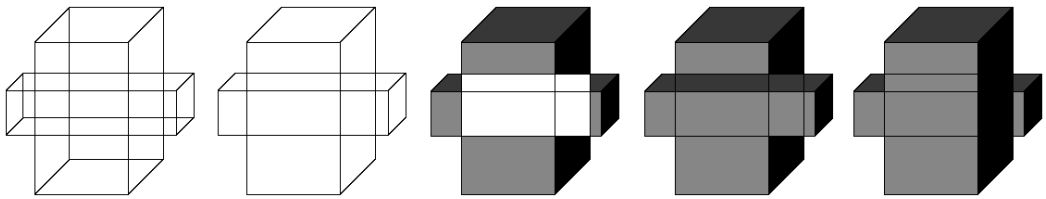
1. *predné hrany*, ktoré sú prienikom viditeľných stien, ***2.*** *obrysové hrany*, ktoré sú prienikom viditeľnej a neviditeľnej steny, ***3.*** *zadné hrany*, ktoré sú prienikom dvoch neviditeľných stien.

Z uvedenej klasifikácie očividne vyplýva užitočnosť štruktúry okrídlená hrana. Takto klasifikované hrany nám pomôžu pri riešení viditeľnosti i v prípade nekonvexných mnohostenov.

12.1.1.3 Všeobecný vektorový algoritmus viditeľnosti

Uvažujme scénu s niekoľkými konvexnými, vzájomne sa neprenikajúcimi mnohostenami[3]. Čiastočné prekrytie niektorej steny inou je spôsobené očividne *prekrytím niektorou z obrysových hrán*. Zadné hrany sú samozrejme neviditeľné, nezávisle na tom, či teleso je konvexné, alebo nie. V konečnom dôsledku tak

- *pri riešení viditeľnosti u nekonvexných mnohostenov namiesto analýzy všetkých dvojíc hrán stačí analyzovať dvojice obrysová-obrysová a obrysová-predná, predná-predná.*



Obr. 12.3: Riešenie viditeľnosti mnohostenovej scény. a) Hranová reprezentácia dvoch prekrývajúcich sa hranolov. b) Hranová reprezentácia s riešením viditeľnosti jednotlivých hranolov. c) Vyznačenie oblastí s jednoznačne určenou viditeľnosťou, nezávisle na tom, ktorý z hranolov je vpredu. d),e) Výsledná viditeľnosť v závislosti na tom, ktorý z hranolov je vpredu.

Algoritmus tak možno navrhnúť nasledujúcim spôsobom:

```
1. Spočítaj priesečníky prekrývajúcich sa priemetov dvojíc hrán:
   obrysová-obrysová, obrysová-predná a predná-predná.
2. 2D priemet zobrazovanej scény rozlož na podoblasti s použitím
   spočítaných priesečníkov.
3. Pre každú podoblasť:
4. { Nájdi všetky steny, ktoré sa do nej sprojektujú.
5.   Podoblasti priraď farbu najbližšej steny.
6. }
```

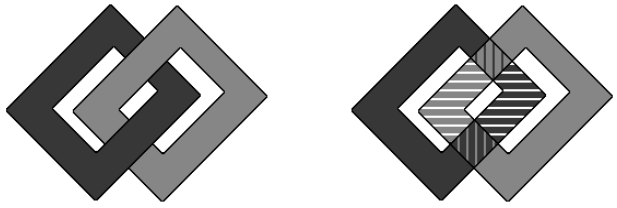
[3]Takúto scénu môžeme považovať za všeobecný (spravidla nekonvexný) mnohosten.

Analýza priesečníkov všetkých dvojíc hrán[4] – obr. 12.3a) – má zložitosť $O(n^2)$. V prvom kroku však využívame vyššie sformulované kritérium, tj. uvažujeme len dvojice hrán obrysová-obrysová a obrysová-predná, predná-predná, čo výrazne zmenší počet analyzovaných dvojíc hrán – obr. 12.3b). Tým sa evidentne zefektívni výsledný algoritmus. Asymptotická zložitosť ale zostáva nezmenená, tj. kvadratická.

Podstatou algoritmu je druhý krok. Každú z podoblastí obr. 12.3b) (podoblasti chápeme ako 2D plochy) môžeme analyzovať nezávisle na sebe (kroky 4 a 5), pretože *v rámci jednej podoblasti je vzájomný vzťah stien* (čo sa usporiadania týka) *nemenný*. Odpadá tak nutnosť úplne usporiadať všetky steny – stačí nájsť len najbližšiu.

Hľadanie efektívnejších riešení – [3], [7], [29] spočíva v analýze toho, za akých podmienok je možné prvé dva kroky uvedenej schémy zjednodušiť.

Tento postup je funkčný i pre neusporiadateľné povrchy. Napr. pre situáciu z obr. 12.1 dostaneme riešenie – obr. 12.4



a) b)

Obr. 12.4: a) Príklad dvojice plôch, ktoré usporiadať nemožno. b) Rozdelenie priemetov plôch na disjunktné súvislé oblasti .

Na rozdiel od obr. 12.1b) kde čierna plocha je vzadu a následným prekreslením tmavošedou plochou ktorá je bližšie[5], dostaneme požadované riešenie, tentokrát je použitý iný mechanizmus: v každej oblasti vykresľujeme plochu, ktorá je k pozorovateľovi najbližšie.

Vidíme, že nevyhnutné, ale vhodne realizované delenie povrchov dovoľuje nahradiť preusporiadanie celej množiny jednoduchšiou procedúrou – nájdením minimálneho prvku množiny.

12.1.2 Rastrovo orientované algoritmy

Ukazuje sa, že práve posledný analyzovaný postup je možno úspešne použiť i v rastrovom prístupe. Akurát pracne generovaný rozkald na podoblasti (krok 2. všeobecného vektorového algoritmu) nahradíme jednoduchším – pravidelným rozkladom – pixelovou sieťou.

12.1.2.1 Základný rastrový algoritmus viditeľnosti

Nasledujúci algoritmus očividne rieši problém viditeľnosti:

[4]Keď uvažujeme, že n je počet stien, tak počet hrán je evidentne $O(n)$.

[5]Práve tu je aktívne využité usporiadanie plôch.

```
1. Pre každý pixel vykonaj kroky 2.-5.
2. { Nájdi všetky plochy, rasterizácia ktorých
     obsahuje daný pixel.
3.   Z týchto plôch urči najbližšiu k pozorovateľovi.
4.   Pixelu priraď farbu najbližšej plochy.
5. }
```

Zložitosť algoritmu je $O(pn)$, kde p je počet pixlov výsledného rastrového obrazu zobrazovanej scény.

Výhody takto koncipovaného algoritmu oproti všeobecnému vektorovému sú dve.

1. V konečnom dôsledku scénu prevádzame z vektorovej do rastrovej formy – spojenie rasterizácie s riešením viditeľnosti preto neznižuje obecnosť a použiteľnosť tejto metódy.

2. Namiesto pracne generovaných rozkladov projekcií stien na podoblasti pracujeme s jednoduchou pravidelne generovanou štruktúrou – rastrovou sieťou.

12.1.2.2 Algoritmus využívajúci pamäť hĺbky (z-buffer)

Nevýhodou základného rastrového algoritmu je krok 2. Mnohonásobné hľadanie priesečníkov totiž chod algoritmu veľmi spomalí. Ponúka sa podstatne efektívnejší postup, keď vonkajší cyklus (krok 1.) nie je vedený cez pixely rastru, ale cez analyzované plochy. To je výhodné z dvoch príčin.

1. Vyhneme sa nutnosti zisťovať plochy, projekcie ktorých obsahujú daný pixel.

2. Pri rasterizácii jednej plochy, z-súradnicu každého pixelu sme spravidla schopni nájsť efektívnejšie, ako počítaním priesčníka priamky a plochy (napr. interpoláciou).

Pre efektívne využitie tohto „hromadného“ zistenia z-súradníc pixelov rasterizovanej plochy je nutné mať možnosť túto informáciu uchovať. Uvažujme preto pomocnú pamäť, v ktorej budeme priebežne uchovávať „prednú frontu“ analyzovanej scény, tj. pre každý pixel z-súradnicu, nájdenú v kroku 3. základného rastrového algoritmu. Táto pomocná pamäť sa zvykne označovať ako *z-buffer* – v nasledujúcom algoritme značíme `ZB`. Samotné vykreslenie znamená zápis do videopamäte – značíme `VRAM`. Celý proces môžeme vyjadriť takto:

```
1.  Inicializuj z-buffer, tj. ZB[x][y]=max pre všetky
    pixely (x,y).
2.  Pre každú plochu scény vykonaj kroky 3.-10.
3.  { c = farba plochy.
```

```
4.    Rasterizuj plochu a pre každý jej pixel (x,y):
5.    { Nájdi z-súradnicu z(x,y).
6.      if ( z(x,y) < ZB[x][y] )
7.      { ZB[x][y]=z(x,y);
8.        VRAM[x][y]=c;
9.      }
10.    }
11. }
```

Vidíme, že „predná fronta“ sa mení len tam, kde práve analyzovaná plocha je bližšie, ako už zanalyzované časti scény (kroky 5.–8.).

Algoritmus je veľmi jednoduchý, a hlavne, nie je závislý na zložitosti štruktúry povrchovej reprezentácie. (Presnejšie povedané, je závislý len na počte plôch, ale nie na ich vzájomnej polohe.) To dovoľuje jednoduchú HW implementáciu. Práve preto uvedený algoritmus je realizovaný v grafických kartách.

12.2 Lokálny osvetľovací model

Farba je prejavom nášho vnemu svetla (kap. 3). Predmety v danej scéne vidíme na základe toho, že svetlo buď vyžarujú, alebo sa svetlo od ich povrchu odráža. Podstatná je pritom vzájomná poloha zdroja, povrchu a pozorovateľa. Práve preto napr. biely kváder nevidíme len ako biely 4-uholník, resp. 6-uholník ale každú zo stien vidíme ako šedú – rôznej intenzity.

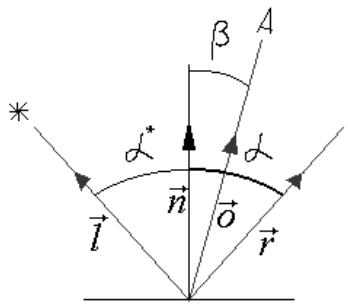
Pre jednoduchosť budeme uvažovať len zdroj „bieleho“, tj. achromatického svetla. Preto pod rôznymi farbami tu budeme rozumieť len rôzne intenzity jedného farebného odtieňa.

Podobne, ako uvažujeme mechanizmus kolmého a stredového premietania, môžeme uvažovať mechanizmus *rovnobežného* a *bodového zdroja svetla*.

Pre vizuálny vnem je ďalej veľmi dôležité, akým spôsobom sa svetlo od povrchu odráža. Práve tomu sa budeme venovať v nasledujúcej časti.

12.2.1 Phongov model osvetlenia

Z fyzikálneho hľadiska je proces šírenia a odrazu svetla vec netriviálna. Pre „bežné“ potreby počítačovej grafiky je často dostatočný veľmi zjednodušujúci model, vychádzajúci z *geometrickej optiky*, kde uvažujeme α^* – uhol dopadajúceho lúča k normále plochy, α – uhol odrazeného lúča k normále plochy, $|\alpha| = |\alpha^*|$, β – uhol pozorovateľa k normále plochy, $|\alpha|, |\alpha^*|, |\beta| \in \langle 0, \frac{\pi}{2} \rangle$, – obr. 12.5. S pojmom „pozorovateľ“ sme sa už nepriamo stretli pri zavádzaní projekcie. V nasledujúcich úvahách by mohol byť pozorovateľ definovaný nezávisle na použitej projekcii. My pre jednoduchosť stotožníme pozorovateľa s výberom



Obr. 12.5: Vzájomná poloha zdroja svetla, odrazovej plochy a pozorovateľa.

použitej projekcie. Pre jednoduchosť uvažujeme, tak ako už niekoľkokrát, spravidla kolmú projekciu do roviny $z = 0$. (Uhol β je preto uhol vonkajšej normály plochy k osi z.)

Phongov osvetľovací model predpokladá tri zložky odrazeného svetla: ***1.*** *zrkadlovú,* ***2.*** *difúznu a* ***3.*** *ambientnú.*

Zrkadlová (spekulárna) zložka odrazeného svetla vychádza z „mechanického“ zákona rovnosti uhlu dopadu a odrazu. Uplatňuje sa u zrkadlových odrazov, resp. u lesklých plôch. V ideálnom prípade intenzita tejto zložky by mala byť vyjadrená vzťahom

$$I = \begin{cases} c_s & \alpha = \beta \\ 0 & \alpha \neq \beta \end{cases} . \tag{12.1}$$

Kritérium, založené na testovaní rovnosti dvoch reálnych čísel, je vzhľadom na chybu diskretizácie nevhodné. (Je napr. číslo 0,000005 skutočne nenulové, alebo je nenulovosť dôsledkom diskretizačnej chyby?) Naviac, v reálnych situáciách zrkadlový odlesk nikdy nenastáva len v jedinom bode. A za tretie, vzhľadom na to, že na opis polohy objektov používame vektory, vieme nájsť jednoducho uhlové charakteristiky na základe skalárneho súčinu $(\vec{u}, \vec{v}) = |\vec{u}|.|\vec{v}|. \cos\omega$. Preto je pre nás výhodnejšie nepracovať priamo s odchylkou ω (uhlom) dvoch vektorov, ale s kosínusom tejto odchylky. Keďže výraz

$$I_s = c_s \left(\cos\left(\alpha - \beta\right)\right)^k$$

s rastúcou hodnotou k pre uhly $\alpha, \beta \in \left\langle 0, \frac{\pi}{2} \right\rangle$, sa limitne blíži (12.1), používa sa pre vyjadrenie zrkadlovej zložky práve táto charakteristika. Je si však treba uvedomiť, že žiaden „fyzikálny“ zmysel konštanta k nemá.

Difúzna zložka závisí len na vzájomnom vzťahu zdroja svetla a odrazivej plochy. Nezávisí na polohe pozorovateľa. Tento typ odrazu sa uplatňuje u matných povrchov a vnímame ho ako farbu povrchu. Pre intenzitu tejto zložky sa používa prirodzený vzťah projekcie vektoru osvetlenia na kolmicu k osvetľovanej ploche. Formálne vyjadrenie je

$$I_d = c_d \left(\vec{n}, \vec{l}\right) = c_d \cos\alpha^*.$$

Použitím len zrkadlovej a difúznej zložky odrazeného svetla by povrchy, ktoré nie sú priamo osvetlené, boli čierne. V reálnom svete tomu tak nie je z dôvodu rozptylu svetla v atmosfére a mnohonásobných odrazov svetla od povrchov. Modelovať toto „fyzikálne korektne“ zozložiťuje a samozrejme spomaľuje vizualizáciu (bližšie sa tomu budeme venovať v kap. 12.3). V jednoduchom priblížení túto okolnosť môžeme nahradiť ***ambientnou zložkou***, tj. akýmsi „priestorovým“ konštantným svetlom

$$I_a = c_a.$$

Koeficienty c_s, c_d, $c_a \in \langle 0, 1\rangle$ charakterizujú materiál povrchu. Spravidla sa volí $c_d = c_a$. V prípade m zdrojov svetla je výsledná intenzita v bode (elementárnej plôške) daná vzťahom

$$I = I_a c_a + \sum_{j=1}^{m} I_{L_j} \left(c_s \left(\cos\left(\alpha_j - \beta\right)\right)^k + c_d \cos\alpha_j^* \right),$$

kde I_a je intenzita ambientnej zložky, $I_{L_1}, I_{L_2}, \ldots I_{L_m}$ sú intenzity jednotlivých zdrojov. Keďže hodnotu kosínusu počítame zo skalárneho súčinu, vyššieuvedený vzťah je formulovaný spravidla v tvare

$$I = I_a c_a + \sum_{j=1}^{m} I_{L_j} \left(c_s \left(\vec{o}, \vec{r_j}\right)^k + c_d \left(\vec{n}, \vec{l_j}\right) \right), \tag{12.2}$$

kde $\vec{o}$ je vektor pozorovateľa, $\vec{l}$ je vektor dopadajúceho lúča na plochu, $\vec{r}$ je vektor odrazeného lúča od plochy, $\vec{n}$ je vektor vonkajšej normály plochy – obr. 12.5. Všetky uvedené vektory sú *normované*, tj.

$$|\vec{o}| = |\vec{l}| = |\vec{r}| = |\vec{n}| = 1.$$

Hodnota I_a síce rastie s počtom zdrojov v scéne, no nárast je menší ako lineárny – praktické experimenty ukazujú, že v opačnom prípade je ambientná zložka neúmerne veľká a výsledný obraz je silne presvetlený.

V prípade konštantných hodnôt I_{L_j} Phongov model (12.2) nezahŕňa vzájomné vzdialenosti zdrojov svetla, odrazových plôch a pozorovateľa.

Keď uvažujeme farebné plochy, potom je nutné určiť koeficienty c_s, c_d, c_a pre každú farebnú zložku zvlášť. Napr. čierny lesklý povrch má u všetkých troch zložiek c_s blízke hodnote 1, ostatné koeficienty sú nulové. Lesklý červený povrch má opäť všetky spekulárne koeficienty blízke 1, pre červenú farbu je $c_d = c_a$ blízke hodnote 1, ostatné koeficienty sú nulové.

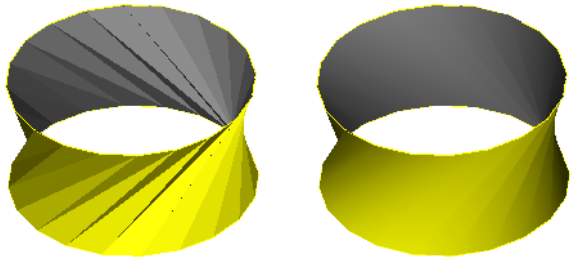
12.2.2 Lokálne osvetlenie pre mnohostenové reprezentácie

Nie je zložité obohatiť základný rastrový algoritmus viditeľnosti o Phongov model osvetlenia. Akurát namiesto kroku 4. v algoritme z kap. 12.1.2.1, resp. v kroku 8. v algoritme z kap.12.1.2.2 aplikujeme intenzitu danú vzťahom (12.2). Budeme uvažovať telesá vo forme mnohostenov, rovnobežný zdroj svetla a kolmú projekciu.

12.2.2.1 Konštantné tieňovanie

Pre všetky body jednej plochy sa uhly $\alpha_j, \alpha_j^*, \beta$ v (12.2) nemenia. Znamená to, že každá povrchová plocha má výslednú konštantnú farbu. Takému spôsobu zobrazovania hovoríme *konštantné tieňovanie*. Toto je výhodné z hľadiska rýchlosti vykresľovania: „drahý“ výpočet (12.2) nerobíme pre každý pixel, ale pre každú plochu len raz. Preto sa dá veľmi jednoducho implementovať v rámci z-bufferu. Zobrazenie mnohostenov týmto spôsobom je veľmi výstižné.

Polygonálna reprezentácia povrchu je ale spravidla dôsledok diskretizácie hladkého povrchu (napr. namiesto valcovej plochy uvažujeme n-boký hranol s dostatočne veľkým n). Tým sa dopúšťame určitej *diskretizačnej chyby*. Vizualizácia s pomocou konštantného tieňovania túto chybu môže silne zvýrazniť – obr. 12.6.



a) b)

Obr. 12.6: Konštantné a hladké tieňovanie: Povrch rotačného hyperboloidu je nahradený trojuholníkmi. a) Konštantné tieňovanie. b) Gouraudovo tieňovanie.

Nižšieuvedené dve metódy si dávajú za cieľ zmierniť prejav tejto diskretizačnej chyby.

12.2.2.2 Gouraudovo tieňovanie

Uvažujme nasledujúcu schému.

```
1. Výpočítaj výslednú farbu (intenzitu) pre každú plochu telesa.
2. Urči farbu (intenzitu) vo vrcholoch polygonálneho povrchu.
3. Pre každý vnútorný pixel rasterizovanej plochy
   interpoluj intenzitu z intenzít vrcholov polygónu.
```

V prvom kroku aplikujeme metódu konštantného tieňovania. Druhý krok sa dá realizovať niekoľkými spôsobami, napr.:

1. hodnotu výslednej farby berieme ako vážený priemer hodnôt incidentných povrchov,

2. od polygonálneho povrchu prejdeme k duálnemu povrchu (kap. 11.6.4), dostaneme tak opäť polygonálnu reprezentáciu, pričom máme definované hodnoty vo vrcholoch.

V treťom kroku môžeme použiť metódu zovšeobecneného vyplňovania (kap.8.1.4).

Súčasné grafické knižnice tento spôsob tieňovania umožňujú. Podpora pre túto metódu je dnes na HW úrovni – metóda býva implementovaná priamo v grafických kartách.

12.2.2.3 Phongovo tieňovanie

Tento spôsob tieňovania, tj. potlačenia „hranatosti" povrchu, využíva taktiež interpoláciu, ovšem nie až na úrovni výpočtu intenzity, ale na úrovni výpočtu vonkajšej normály povrchu:

```
1. Výpočítaj vonkajšiu normálu povrchových plôch.
2. Prenes normály do vrcholov siete.
3. Pre každý vnútorný pixel rasterizovanej plochy:
4  { Interpoluj normálu z normál vo vrcholoch.
5.   Normuj interpolovaný normálový vektor.
6.   Výpočítaj výsledné osvetlenie pixelov.
7. }
```

Prvý krok je bezproblémový – realizovaný tak, ako u konštantného tieňovania. V štvrtom kroku postupujeme podobne ako u Gouraudovho tieňovania s tým rozdielom, že namiesto jednej - skalárnej hodnoty (intenzita farby) pracujeme nezávisle s troma hodnotami (zložky vonkajšieho normálového vektoru). Interpolácia v tomto kroku 4 spôsobí, že smer vektoru už nebude na jednej ploche konštantný, v dôsledku čoho sa povrch vizuálne „zaoblí".

Vektor, ktorý dostamene po kroku 4 nemusí byť (a spravidla nie je) normovaný. Keďže (12.2) toto vyžaduje, krok 5 je nevyhnutný. Posledný krok dostaneme priamou aplikáciou rovníc (12.2).

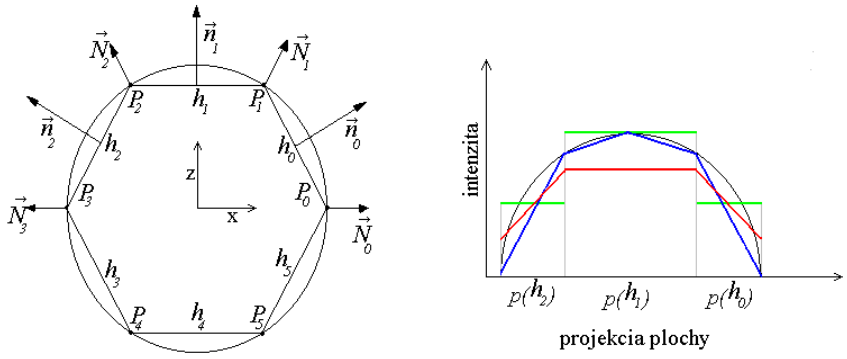
Rozdiely vyššieuvedených metód ilustruje nasledujúci príklad – obr. 12.7.

Uvažujme valcovú plochu s osou $y = 0$, $x^2 + z^2 = 1$. Nech pozorovateľ i zdroj svetla ležia na osi $x = 0$, $z > 0$. Predpokladajme kolmú projekciu a difúzny rovnobežný zdroj svetla v smere $-\vec{z}$, tj. v (12.2)

$$c_a = c_s = 0,\ c_d = 1, \quad \vec{l} = (0, 1)$$

($\vec{l}$ smeruje od plochy k zdroju svetla – obr. 12.5).

V tom prípade môžeme celú analýzu spraviť v 2D priestore – v rovine xz, tj. namiesto osvetlenia plôch uvažujeme osvetlenie príslušných hrán – obr. 12.7a).



a) b)

Obr. 12.7: Porovnanie tieňovacích metód. a) Nahradenie kružnice 6-uholníkom: uzly P_i, hrany h_i, vonkajšie normály hrán $\vec{n}_i$, dopočítané normály vo vrcholoch $\vec{N}_i$.
b) Intenzita farby pixelov na projekcii viditeľných hrán (plôch). Presné tieňovanie (čierna), konštantné tieňovanie (zelená), Gouraudovo tieňovanie (červená), Phongovo tieňovanie (modrá).

Za týchto predpokladov kruh (2D variant valcovej plochy) nahradíme pravidelným šesťuholníkom (2D variant šesťbokého hranolu) s vrcholami

$$P_0 = (1,0)\,, \quad P_1 = \left(\tfrac{1}{2}, \tfrac{\sqrt{3}}{2}\right), \quad P_2 = \left(-\tfrac{1}{2}, \tfrac{\sqrt{3}}{2}\right),$$
$$P_3 = (-1,0)\,, \quad P_4 = \left(-\tfrac{1}{2}, -\tfrac{\sqrt{3}}{2}\right), \quad P_5 = \left(\tfrac{1}{2}, -\tfrac{\sqrt{3}}{2}\right).$$

Na úsečkách $h_i = (P_i P_{i+1})\,,\ i = 0, \ldots, 4,\ h_5 = (P_5 P_0)$ máme vonkajšie normály

$$\vec{n}_0 = \left(\tfrac{\sqrt{3}}{2}, \tfrac{1}{2}\right), \quad \vec{n}_1 = (0,1)\,, \quad \vec{n}_2 = \left(-\tfrac{\sqrt{3}}{2}, \tfrac{1}{2}\right),$$
$$\vec{n}_3 = \left(-\tfrac{\sqrt{3}}{2}, -\tfrac{1}{2}\right), \quad \vec{n}_4 = (0,-1)\,, \quad \vec{n}_5 = \left(\tfrac{\sqrt{3}}{2}, -\tfrac{1}{2}\right).$$

Pozorovateľ vidí hrany h_0, h_1, h_2 – obr. 12.7a).

V prípade *reálneho tieňovania* pre danú úlohu sa dá normála k valcovej ploche vyjadriť ako $\vec{n}\,(x) = (x, z)\,, \vec{l} = (0,1)$. Za predpokladu difúzneho modelu osvetlenia $I = \left(\vec{n}, \vec{l}\right)$ dostávame (obr. 12.7 b) – čierna farba)

$$I\,(x) = \left(\vec{n}\,(x)\,, \vec{l}\right) = z\,(x) = \sqrt{1 - x^2}.$$

Pre *konštantné tieňovanie* dostávame intenzity na hranách h_0, h_1, h_2– obr. 12.7 b) – zelená farba.

$$I_0 = \left(\vec{n}_0, \vec{l}\right) = \tfrac{1}{2}, \quad I_1 = \left(\vec{n}_1, \vec{l}\right) = 1, \quad I_2 = \left(\vec{n}_2, \vec{l}\right) = \tfrac{1}{2}.$$

Pre *Gouraudovo tieňovanie* spočítame intenzity vo vrcholoch priemerovaním intenzít priľahlých hrán (hrana h_5 je neviditeľná, preto $I_5 = 0$):

$$\begin{array}{ll} I(P_0) = \frac{1}{2}(I_0 + I_5) = \frac{1}{4}, & I(P_1) = \frac{1}{2}(I_0 + I_1) = \frac{3}{4}, \\ I(P_2) = \frac{1}{2}(I_1 + I_2) = \frac{3}{4}, & I(P_3) = \frac{1}{2}(I_2 + I_3) = \frac{1}{4} \end{array}.$$

Výsledné intenzity na hranách dostaneme lineárnou interpoláciou týchto intenzít obr. 12.7b) – červená farba. Vidíme, že na hranách h_0, h_2 sa intenzita spojite mení, no na hrane h_1 je hodnota konštantná, pretože na oboch koncoch máme rovnaké intenzity $I(P_1) = I(P_2) = \frac{3}{4}$. Toto je dôvod, prečo nám Gouraudovo tieňovanie nestačí.

Ukážeme teraz priebeh interpolovaných intenzít pri použití *Phongovho tieňovania*. Najprv spočítame „normály vo vrcholoch“[6] (krok 2. algoritmu) ako aritmetický priemer normál priľahlých hrán:

$$\begin{array}{ll} \vec{N}_0 = \vec{n}(P_0) = \frac{1}{2}(\vec{n}_0 + \vec{n}_5) = \left(\frac{\sqrt{3}}{2}, 0\right), & \vec{N}_1 = \vec{n}(P_1) = \frac{1}{2}(\vec{n}_0 + \vec{n}_1) = \left(\frac{\sqrt{3}}{4}, \frac{3}{4}\right), \\ \vec{N}_2 = \vec{n}(P_2) = \frac{1}{2}(\vec{n}_1 + \vec{n}_2) = \left(-\frac{\sqrt{3}}{4}, \frac{3}{4}\right), & \vec{N}_3 = \vec{n}(P_3) = \frac{1}{2}(\vec{n}_2 + \vec{n}_3) = \left(-\frac{\sqrt{3}}{2}, 0\right). \end{array}$$

Lineárnou interpoláciou normály dostávame napr. na hrane h_1 vektor

$$\vec{N}_{h_1}(t) = (1-t)\vec{N}_1 + t\vec{N}_2 = \left(\frac{\sqrt{3}}{4} - t\frac{\sqrt{3}}{2}, \frac{3}{4}\right) \quad t \in \langle 0, 1\rangle.$$

Všimnime si, že vzhľadom na to, že y-zložka vektoru $\vec{N}_{h_1}(t)$ je konštantná, priame použitie tohto vektoru (bez normovania) pre výpočet intenzity difúzneho osvetlenia by dalo na hrane h_1 výsledok $I(t) = \left(\vec{l}, \vec{N}_{h_1}(t)\right) = \frac{3}{4}$, ktorý je indentický s Gouraudovým tieňovaním. Toto je dôvod, prečo v piatom kroku algoritmu musíme interpolovaný vektor normovať

$$I_{h_1}(t) = \frac{\left(\vec{l}, \vec{N}_{h_1}(t)\right)}{\left|\vec{N}_{h_1}(t)\right|}.$$

12.3 Globálne osvetľovacie modely

V horeuvedenom lokálnom osvetlení je zabudovaný obmedzujúci predpoklad, ktorý môže byť problematický: svetlo sa od plochy síce odrazí, a jeho množstvo priamo určuje vnímané osvetlenie tejto plochy, no *odrazené svetlo už žiadnym spôsobom neovplyvňuje osvetlenie ďalších plôch*! Preto lokálne metódy osvetlenia scény dobre funguje v prípade, keď plôch v

[6] Úvodzovky sú opodstatnené, pretože z matematického hľadiska vo vrcholoch normála neexistuje.

a) b) c) d)

Obr. 12.8: Výsledný vnem osvetlenia valcovej plochy. a) Skutočné osvetlenie, b) konštantné tieňovanie, c) Gouraudovo tieňovania, d) Phongovo tieňovanie.

scéne nie je veľa, a hlavne, keď sú dostatočne ďaleko od seba, takže sa vzájomne neovplyvňujú. Jediný príspevok Phongovho osvetľovacieho modelu, ktorý vzájomné pôsobenie plôch berie v úvahu, je ambientná zložka svetla. Vytvorením závislosti ambientnej zložky na počte plôch a počte zdrojov svetla v scéne, môžeme do modelu dostať vzájomné svetelné ovplyvnenie plôch, no od takého riešenia nemožno veľa očakávať. Lepším riešením je ***1.*** *zahrnutie niekoľkonásobného odrazu svetla od povrchov v scéne,* resp. ***2.*** *neoddeľovať striktne plochy scény, na ktorých nastáva odraz svetla, od zdrojov svetla.*

Prvý prístup používa metódy, založené na geometrickej optike, kde sa analyzuje trasa lúčov svetla od zdroja až po pozorovateľa, druhý prístup vychádza z fyzikálnej formulácie, založenej na zákone zachovania energie.

12.3.1 Ray-tracing

Budeme vychádzať z Phongovho lokálneho osvetľovacieho modelu. Zvoľme projekčnú plochu tak, aby oddeľovala pozorovateľa od scény. Uvažujeme lúč svetla ako lineárne lomenú čiaru, ktorá sa od plochy v scéne odráža pod tým istým uhlom, pod akým na ňu dopadá.

Priame sledovanie svetelného lúča od zdroja svetla až k rovine projekcie by sme mohli algoritmizovať nasledujúcim spôsobom:

```
1. Pre každý zdroj svetla generuj systém lúčov, a pre každý lúč:
2. { Kým nie je splnený príznak konca analýzy lúča:
3.   { Nájdi bod odrazu, tj. priesečník s plochou,
       ktorú lúč pretne ako prvú.
4.     Vygeneruj odrazený lúč.
5.     Aktualizuj vlastnosti lúča.
6.   }
7.   Prepočítaj výslednú hodnotu pixelu.
8. }
```

Príznakom konca je jedna z nasledujúcich možností:

```
a) cyklus 3.-6. prebehol viackrát ako vopred
   stanovený počet,
b) svetelný lúč pretne projekčnú rovinu,
c) bod odrazu neexistuje.
```

Opodstatnenie prvého prípadu možno zdôvodniť tým, že veľkým počtom odrazov sa pôvodná svetelná energia celá pohltí na odrazových plochách. Pre nás je tento prípad dôležitý, pretože zabezpečí, že algoritmus sa nezacyklí. V druhom prípade nájdeme pre daný pixel a pre daný zdroj svetla odpovedajúcu farbu. Tretí prípad znamená, že odrazený lúč opustí scénu.

Piaty krok algoritmu využíva Phongov osvetľovací model. Cyklus 2.-8. prebehne pre každý zdroj svetla, preto v kroku 7. aktualizujeme výslednú intenzitu pixelu o práve analyzovaný zdroj.

Problematické v takto navrhnutom algoritme sú dva momenty.

1. Nie sme schopní rozumne modelovať difúzny odraz svetla – to by sme museli v každom bode odrazu generovať celý zväzok lúčov difúznej zložky.

2. Diskretizácia prvotných lúčov zo zdroja svetla, aj keď je rovnomerná, vygeneruje nerovnomerne rozmiestnené prieniky s projekčnou rovinou.

Preto sa takto navrhnutý algoritmus používa len obmedzene, napr. len s jedným možným odrazom.

Namiesto hľadania toho, akým spôsobom generovať lúče zo zdroja tak, aby preťali projekčnú rovinu rovnomerne, je jednoduchšie postupovať opačne. Budeme generovať postupne lúče od pozorovateľa tak, aby rovnomerne pokryli projekčnú rovinu, a trajektóriu každého z takýchto spätne bežiacich lúčov do scény budeme analyzovať. Tento *algoritmus spätného sledovania lúča* (ray-tracing), sa často formuluje v rekurzívnej forme:

```
PROCEDURA RTr(BOD,LÚČ,PIXEL,VNORENIE)
1.  { Z pozície BOD generuj LÚČ.
2.    Nájdi BOD_ODRAZU, tj. prienik LÚČa s plochou,
      ktorú LÚČ pretne ako prvú.
3.    Nájdi všetky zdroje svetla v scéne,
      viditeľné z BODu_ODRAZU.
4.    Aktualizuj príspevok osvetlenia BODu_ODRAZU
      od viditeľných zdrojov svetla na PIXEL.
5.    Ak nie je splnený príznak konca:
6.    { VNORENIE = VNORENIE + 1.
7.      Generuj ODRAZENÝ_LÚČ.
8.      RTr(BOD_ODRAZU,ODRAZENÝ_LÚČ,PIXEL,VNORENIE)
9.    }
10. }
```

Príznaky konca sú podobné:

```
a) počet vnorení presiahol vopred stanovenú hranicu,
b) BOD_ODRAZU je na projekčnej ploche,
c) BOD_ODRAZU nenxistuje,
d) BOD_ODRAZU je v niektorom zdroji svetla.
```

Celková analýza je nasledujúca:

```
1. Pre každý PIXEL scény vykonaj:
2. { Generuj LÚČ od POZOROVATEĽa, cez PIXEL do scény.
3.   RTr(POZOROVATEĽ,LÚČ,PIXEL,0).
4. }
```

Podstatná zložitosť algoritmu je v nájdení bodu odrazu (krok 2.). Pre urýchlenie sa preto často využíva hierarchická stavba scény – kap. 11.5.

Metóda *ray tracing* prirodzeným spôsobom zovšeobecňuje metódu lokálneho osvetlenia. Prvá, nerekurzívna časť procedúry (kroky 1.–4.), realizuje lokálne osvetlenie scény. Dá sa realizovať veľmi jednoduchou úpravou základného rastrového algoritmu viditeľnosti (kap. 12.1.2.1). V literatúre je označovaná ako *ray casting.*

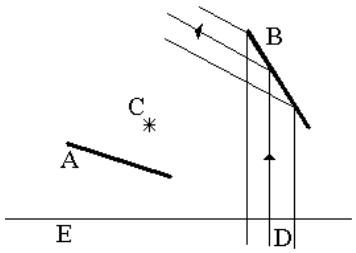
Na generovanie odrazeného lúča (krok 7.) je prirodzené použiť zákon rovnosti uhlov dopadu a odrazu z geometrickej optiky. Rekurzivne vnorenie je vlastne akési presunutie pozorovateľa do scény, alebo povedané ešte trochu ináč – je to pohľad na scénu s lokálnym modelom šírenia svetla, cez zrkadlovú plochu. Mieru „zrkadlovosti“ nastavujeme parametrom c_s (12.2) pre danú plochu[7]. Výsledná vizualizácia je samozrejme lepšia, ako v prípade čiste lokálneho osvetlenia, no nemožno ju považovať za korektnú z hľadiska fyziky. Zrkadlová zložka je korektná, ale šírenie difúznej zložky je „znásilnené“ tým, že sa uvažuje len zo smeru odrazeného lúča.

Tento nedostatok sa snaží eliminovať nasledujúca metóda.

[7]Ray-tracing dovoľuje modelovať okrem zrkadlového odrazu i lom svetla. V algoritme sa to prejaví jednoduchou modifikáciou rekurzívnej časti:.

```
5.    Ak nie je splnený príznak konca:
6.    { VNORENIE = VNORENIE + 1.
7.      Generuj ODRAZENÝ_LÚČ.
8.      CALL RTr(BOD_ODRAZU,ODRAZENÝ_LÚČ,PIXEL,VNORENIE)
9.      Generuj LOMENÝ_LÚČ.
10.     CALL RTr(BOD_ODRAZU,LOMENÝ_LÚČ,PIXEL,VNORENIE)
11.   }
```

kde pre uhol lomu svetla je rozumné brať fyzikálne opodstatnené hodnoty.



Obr. 12.9: Nekorektnosť ray-tracingu. Pri použití paralelnej projekcie na projekčnú rovinu E, generované lúče D zasiahnu plochu B a odrazia sa mimo scénu. Keďže v bodoch odrazu sa analyzuje len viditeľnosť zdroja svetla C, plocha A vôbec neovplyvní výsledok ray-tracingu. V skutočnosti sa však dá očakávať, že plocha A zosiluje účinok zdroja C pri osvetlení plochy B, tj. pôsobí ako ďalší zdroj svetla.

12.3.2 Radiačná metóda

Metóda je založená na myšlienke, že každá osvetlená plocha v scéne pôsobí zároveň ako zdroj odrazeného svetla. Aby bol prístup fyzikálne korektný, vyžaduje sa splnenie zákona zachovania svetelnej energie, ktorá na plochu dopadá, s energiou pohltenou a vyžiarenou. Ináč povedané, energia v priestore scény nemôže ani vzniknúť ani sa stratiť.

Majme v scéne n povrchov, každý z nich má konečnú plochu. Označme H_i celkovú energiu, ktorá na i-tu plochu dopadá, p_i – koeficient odrazu E_i – energiu, ktorá sa do okolia vyžiari v dôsledku toho, že daná plocha je zdrojom svetla. Celková vyžiarená energia do okolia B_i tak je

$$B_i = p_i H_i + E_i.$$

Keďže H_i je dôsledok sumárneho vyžarovania ostatných plôch v priestore,

$$H_i = \sum_{j=1}^{n} F_{ij} B_j.$$

Koeficient F_{ij} (form-faktor, resp. konfiguračný faktor[8]) vyjadruje, aká časť celkovej vyžiarenej energie z plochy j osvetlí plochu i. Dostávame tak sústavu lineárnych rovníc

$$B_i - p_i \sum_{j=1}^{n} F_{ij} B_j = E_i, \quad i = 1, 2, \ldots, n,$$

kde pre každú plochu hľadáme hodnoty vyžiarenej energie B_i. Dá sa povedať, že dnes nie je problém v samotnom nájdení riešenia sústavy lineárnych rovníc. Podstatne zložitejšie je nájsť pre danú scénu konfiguračné faktory F_{ij}. Tie závisia na geometrii analyzovanej scény

[8]Možno by sa lepšie hodil názov *koeficient vzájomného vplyvu.*

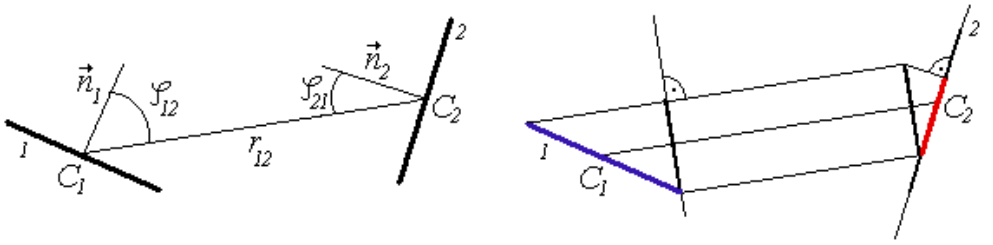
a taktiež na tom, aký model svetelného pôsobenia zvolíme. Pre jednoduchosť sa obmedzíme iba na 2D prípad. Keď predpokladáme, že

1. *úsečky sú rovnako veľké,* ***2.*** *úsečky sú elementárne, tj. ich vzájomné ovplyvnenie je konštantné),* ***3.*** *úsečka vyžaruje do všetkých smerov rovnako (difúzny model odrazu svetla),* ***4.*** *celková vyžiarená energia je priamo úmerná veľkosti (tj. dĺžke),* ***5.*** *intenzita žiarenia je nepriamo úmerná vzdialenosti,*

dostávame pre konfiguračné faktory vzťah[11][9]

$$F_{ij} = \frac{\cos\varphi_{ij}\cos\varphi_{ji}}{r_{ij}} l. \tag{12.3}$$

Vidíme, že pôsobenie elementárnych plôch je vzájomné, tj. $F_{ij} = F_{ji}$. Detailnejšie popíšeme interpretáciu tohto vzťahu.



a) b)

Obr. 12.10: Form-faktor pre dvojicu elementárnych úsečiek.

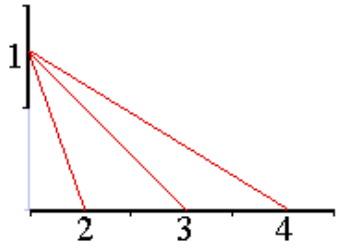
Majme úsečky 1 a 2., obr. 12.10 a). Stredy úsečiek označme C_1, C_2. Úsečku 1 dĺžky l (modrá) kolmo sprojektujeme na smer C_1C_2, túto projekciu prenesieme paralelne so smerom r_{12} až k úsečke 2 a tam túto projekciu opäť kolmo sprojektujeme na 2 (červená) obr 12.10 b). Výsledná projekcia má tak veľkosť $P_1 = l\cos\varphi_{12}\cos\varphi_{21}$. (Taký veľký je vlpyv úsečky 1 na úsečku 2.)

Pre výpočet vzájomného vplyvu väčších, prípadne prekrývajúcich sa plôch, je zmysluplné rozdeliť ich na elementárne Napr. na obr. 12.11 je jednoduchá scéna s dvomi vzájomne kolmými úsečkami nerovnakej dĺžky. Väčšia z nich (horizontálna) je rozdelená na tri elementárne (2, 3, 4) s dĺžkou $l = 1$.

Dostávame tak:

i	j	r_{ij}	$\cos\varphi_{ij}$	$\cos\varphi_{ji}$	$F_{ij} = \frac{\cos\varphi_{ij}\cos\varphi_{ji}}{r_{ij}} l$
1	2	$\sqrt{\left(\frac{3}{2}\right)^2 + \left(\frac{1}{2}\right)^2} = \frac{\sqrt{10}}{2}$	$\frac{1}{\sqrt{10}}$	$\frac{3}{\sqrt{10}}$	$\frac{3}{5\sqrt{10}} \doteq 0.19$
1	3	$\sqrt{\left(\frac{3}{2}\right)^2 + \left(\frac{3}{2}\right)^2} = \frac{\sqrt{18}}{2}$	$\frac{1}{\sqrt{2}}$	$\frac{1}{\sqrt{2}}$	$\frac{1}{\sqrt{18}} \doteq 0.24$
1	4	$\sqrt{\left(\frac{3}{2}\right)^2 + \left(\frac{5}{2}\right)^2} = \frac{\sqrt{34}}{2}$	$\frac{5}{\sqrt{34}}$	$\frac{3}{\sqrt{34}}$	$\frac{15}{17\sqrt{34}} \doteq 0.15$

[9] V 3D prípade je vzťah podobný – ovšem elementárna pôška má rozmer l^2 a so vzdialenosťou klesá vplyv nie lineárne, ale kvadraticky: $F_{ij} = \frac{\cos\varphi_i\cos\varphi_j}{r_{ij}^2} l^2$.



Obr. 12.11: Rozdelenie plôch na elementárne. Červene sú vyznačené spojnice stredov elementárnych úsečiek.

kde r_{ij} je dĺžka spojnice stredov úsečiek a φ_{ij} je uhol medzi vonkajšou normálou plochy i a spojnicou stredov ij. Hodnoty form-faktorov pre dvojice elementárnych plôch 2,3,4 sú $F_{23} = F_{24} = F_{34} = 0$.

Je dôležité si uvedomiť, že pri výpočte form-faktorov bol použitý „energetický“ prístup. Väzba medzi dvojicou plôch je tak vyjadrená skalárnou hodnotou. Možno povedať, že takto postavený model je fyzikálne korektný, no bohužiaľ nedovoľuje zahrnúť zložitejšie interakcie medzi plochami napr. zrkadlový odraz. I preto sa v súčasnosti kombinujú obe vyššieopísané metódy.

12.4 Záverečné poznámky k priestorovému vnímaniu

Vyššie opísaná viditeľnosť a osvetlenie nestačia na to, aby sa nám vygenerovaný obraz javil ako skutočne trojrozmerný. Pri vnímaní reálneho trojrozmerného sveta pôsobia rôzne faktory. Priestorovosť reálnych vnemov je veľakrát *subjektívnym* dôsledkom našej každodennej skúsenosti. Napr. väčšie a rýchlo sa pohybujúce predmety sa nám zdajú byť bližšie. Na druhej strane, predmety sfarbené do modra vnímame ako vzdialené. Mnohé optické klamy sú založné práve na subjektívnych vnemoch.

Objektívne faktory ktoré najviac ovplyvňujú priestorovosť videnia sú dva: 1. schopnosť meniť ohniskovú vzdialenosť šošovky v oku, 2. skutočnosť, že vizuálny vnem dostávame z dvoch rôznych pozícií. Oba tieto princípy sa využívajú i v súčasných systémoch, pracujúcich s 3D priestorovou informáciou.

12.4.1 Multifokálnosť

Tento princíp môžete vyskúšať jednoduchým pokusom. Pri pohľade do zrkadla jedným okom buď sledujte svoj obraz, alebo sa sústreďte na povrch samotného skla zrkadla. Rozdiel týchto dvoch stavov je daný zmenou ohniskovej vzdialenosti očnej šošovky (*akomodácia*

šošovky). V prvom prípade „zaostrujeme“ na dvojnásobnú vzdialenosť oka od zrkadla, kým v druhom prípade zaostrujeme na prostú vzdialenosť. V bežnom živote si tento mechanizmus veľmi neuvedomujeme, no na základe nadobudnutej skúsenosti sa v mozgu odvodí vzťah medzi mierou deformácie očnej šošovky a sledovanou vzdialenosťou. Toto sa podieľa na vnímaní priestorových vzťahov len na malé vzdialenosti.

V technickej praxi sa tento princíp používa napr. pri automatickom zaostrovaní v digitálnych prístrojoch, tzv. *pasívny autofokus*: analyzuje sa plocha obsahujúca dve výrazne odlišné farby, tj. ostrú hranu medzi nimi (uvažujme pre jednoduchosť čiernu a bielu).

```
1. Pri postupnej krokovej zmene ohniskovej
   vzdialenosti objektívu:
2. { Rasterizuj analyzovanú plochu.
3.   Generuj histogram výskytu prechodových
     odtieňov.
   }
4. Vzdialenosť odvoď z tej ohniskovej vzdialenosti
   objektívu, pri ktorej je v analyzovanom histograme
   najmenej prechodových odtieňov.
```

Multifokálnosť sa využíva napr. i v optických profilometroch pre detailnú analýzu nerovnosti povrchov. Základom je *optická aberácia šošovky*, tj. fakt, že ohnisková vzdialenosť šošovky závisí na vlnovej dĺžke prechádzajúceho monochromatického svetla. Prechodom bieleho svetla a spektrálnou analýzou skoncentrovaného lúča osvieteného povrchu môžeme odvodiť vzdialenosť povrchu od šošovky[10].

12.4.2 Binokulárna disparita

Keď sa na statickú konfiguráciu predmetov pozeráme postupne jedným, a potom druhým okom bez toho, aby sme sa hýbali, vidíme, že obrazy pre každé oko sú rôzne. V mozgu sa z rôznosti týchto obrazov, resp. z rôzností uhlov natočenia očí voči sledovanému predmetu, odvodí dojem priestorovosti. Modifikujme vyššie opísaný pokus tak, že svoj obraz v zrkadle a povrch zrkadla sledujeme súčasne obidvomi očami. Vidíme, že tento mechanizmus je pre vnímanie trojrozmerného priestoru pre nás dôležitejší.

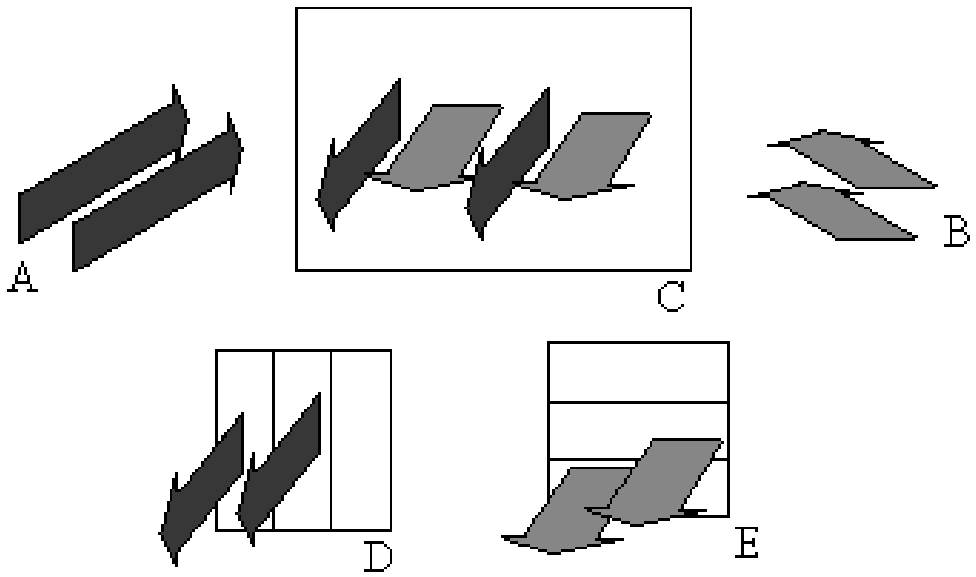
Keď chceme tento princíp využiť pre vizualizáciu virtuálnej trojtozmernej scény, musíme porobne ako v svete reálnom, každému oku dodať adekvátnu dvojrozmernú projekciu tejto scény. Riešení je niekoľko. Napr. virtuálne okuliare (*head mounted display*) obsahujú pre každé oko svoj monitor, na ktorom sa zobrazuje požadovaný priemet virtuálnej scény.

Mnoho technológií ale funguje tak, že *na jednu plochu sa premietajú obe projekcie a následne sa pre každé oko požadovaná projekcia separuje.*

[10]Napr. FRT CWL Sensor (MicroProf) je schopný merať vzdialenosti v rozmedzí 0,3–10mm. http://www.frt-gmbh.de

Technicky najjednoduchším riešením je *farebný anaglyf*, skonštruovaný v r. 1853. V čistej forme funguje nasledujúcim spôsobom. Použijeme biele pozadie scény. Pre projekciu objektov scény pre jedno oko použijeme tyrkysovú a pre druhé oko červenú farbu.[11] Obe projekcie premietame na jednu plochu. Pre prezeranie použijeme okuliare s jedným tyrkysovým a jedným červeným sklom (filtrom) zorníku. V pohľade cez tyrkysový filter biele pozadie splynie s tyrkysovou projekciou, a preto vidíme len červenú projekciu (vnímame ju ako čiernu). Podobne cez červený filter vidíme len tyrkysovú projekciu. Hlavnou nevýhodou tejto metódy je, že farebnosť výsledného vnemu je podstatne obmedzená [30].

Veľmi rozšírená je ďalšia filtračná technológia, založena na polarizácii svetla. Dá sa jednoducho implementovať pomocou dvoch dataprojektorov, osadených polarizačnými filtrami, obr. 12.12. Jeden obraz je premietaný vertikálne polarizovaným svetlom (A), druhý horizontálne polarizovaným (B). Pre separáciu obrazov z premietacieho plátna (C) používame okuliare, v ktorých je zorník pre jedno oko orientovaný vertikálne (D) a pre druhé horizontálne (E). Tento mechanizmus sa používa napr. v technológiách IMAX pre 3D kinematografickú projekciu. Táto technológia však nie je použiteľná pre výstup na monitor.

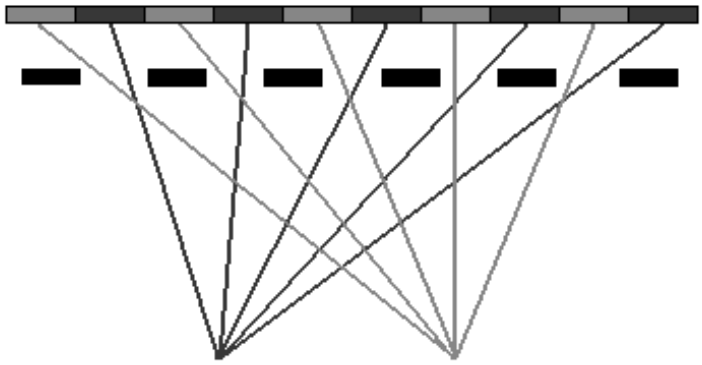


Obr. 12.12: Polarizačná metóda.

3D technológie, ktoré používajú ako výstupné zariadenie monitor, s vysokou frekvenciou striedajú zobrazenie pravej a ľavej projekcie. K sledovaniu obrazu potrebujeme okuliare, ktoré synchrónne s tou istou frekvenciou znepriehľadňujú príslušný zorník.

Existujú i technológie ktoré dovoľujú dosiahnuť 3D vnem i bez použitia špeciálnych okuliarov. Obrazy pre ľavé a pravé oko sa separujú už v rámci monitoru. Používa sa k tomu *paralaxová bariéra.* Je to mriežka, ktorá rozdeľuje obraz na systém tenkých vertikálnych

[11] Je možno použiť i iné dvojice vzájomne doplnkových farieb.



a) b)

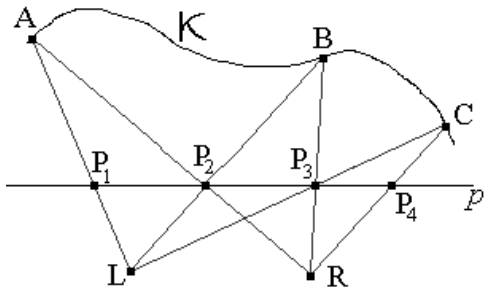
Obr. 12.13: Paralaxová bariéra. Mriežka tieni oblasti obrazu tak, že ľavé oko vidí súvislú tmavošedú plochu, kým pravé oko svetlošedú.

pruhov. Ako demonštruje obr., v určitej polohe pozorovateľa vzhľadom k takému monitoru dosiahneme stav, keď mriežka separuje pruhy tak, že z dvojice susedných pruhov jedno oko vidí jeden a druhé oko druhý pruh. Nevýhody tejto metódy sú dve. **1.** Rozsah polohy, kde dosahujeme správny vnem, je dosť malý. **2.** Rozlíšenie monitoru sa zníži na polovicu.

Prvá nevýhoda sa eliminuje použitím niekoľkonásobnej paralaxovej bariéry, čo však na druhej strane ešte viac znižuje rozlíšenie monitoru.

Na veľmi podobnom princípe funguje technológia ktorá využíva *lentikulárnu vrstvu*, nalepenú ako vrchnú fóliu na nosič obrazu (papier). Lentikulárna vrstva je tvorená systémom zvislých „polcylindrických“ šošoviek, ktoré vykonávajú funkciu paralaxovej bariéry.

Za určitých okolností sme schopni separovať obrazy z dvoch projekcií sami i bez „doplnkového hardware“ – obr. 12.14. Priestorový útvar κ projektujeme na rovinu p v dvoch stredových



Obr. 12.14: Princíp stereogramu.

projekciách, so stredami L a R. Pre bod $A \in \kappa$. nájdeme projekcie $P_1 = L_A$ a $P_2 = R_A$. Pre projekciu P_2 nájdeme bod $B \in \kappa$ tak, že $P_2 = L_B$. Pre nájdený bod B skonštruujeme projekciu $P_3 = R_B$. Podobne pre projekciu P_3 nájdeme $C \in \kappa$ tak, že $P_3 = L_C$ a nájdeme projekciu $P_4 = R_C$.

Je podivuhodné, že v zrakovom centre mozgu z projekcií P_1, P_2, P_3, P_4 dokážeme vytvoriť väzby $P_2 = R_A = L_B$, $P_3 = R_B = L_C$, v dôsledku čoho vzniká priestorový vnem.

ž š č ď ť ň ľ Í

12.5 Zhrnutie a úlohy

- x
- x

1. x
2. x

Literatúra

[1] Aukstakalnis, S., Blatner, D.: *Readž˝lnďž˝ o virtuďž˝lnďž˝ realitďž˝.* Jota, Brno 1994.

[2] Bresenham, J., E.: Algorithm for Computer Control of a Digital Plotter. *IBM Systems Journal, 4(1)*, 1965. pp 25-30.

[3] Bittner, J., Wonka, P.: Visibility in Computer Graphics. *Technical Report TR-186-2-03-03, ICGA, Vienn University of Technology.*, 2003.

[4] Bresenham, J., E.: A Linear Algorithm for Incremental Digital Display of Circular Arcs. *Communications of the ACM*, 20(2), 1977. pp 100-106.

[5] Bui,D.H., Skala,V.: Fast Algorithms for Clipping Lines and Line Segments in E2. *The Visual Computer 14*, No.1. 1998. pp 31-38.

[6] Cyrus, M., Beck, J.: Generalized two- and three dimensional clipping. *Computers and Graphics 14(1)*, 1978. pp 23-28.

[7] Ferko, A., Ruďž˝ickďž˝,E.: *Poďž˝ďž˝taďž˝ovďž˝ grafika a spracovanie obrazu.* Sapientia, Bratislava 1995. ISBN 80-967180-2-9.

[8] Feynman,R.,P., Leighton,R.,B., Sands,M.: *Feynmanove prednďž˝ďž˝ky z fyziky/2.* Alfa, Bratislava 1982.

[9] Foley, J., van Dam, A., Feiner, S., Hughes, J.: Computer Graphics – Principles and Practice. 2nd ed., Addison-Wesley, 1990.

[10] Gonzales, R., C., Woods, R., E.: *Digital Image Processing.* Addison Wesley 1992.

[11] Goral, C., M., Torrance, K., E., Greenberg, D., P., Battaile, B.: Modelling the Interaction of Light Between Diffuse Surfaces. *Computer Graphics 18*, No.3. 1984. pp 213-222. Addison Wesley 1992.

[12] Horďž˝ďž˝k,P.: *Svetelnďž˝ technika.* SNTL-Alfa, Bratislava 1989.

[13] Hudec, B.: *Zďž˝klady poďž˝ďž˝taďž˝ovďž˝ grafiky.* ďž˝VUT, Praha 1993. ISBN 80-01-00932-7.

[14] Jacobs, G. H., Nathans, J.: *Evoluce barevnďž˝ho vidďž˝nďž˝ primďž˝tďž˝.* Scientific American, ďž˝eskďž˝ vydďž˝nďž˝, duben 4/2010. (52-59). ISSN XXXXXXXXXX.

[15] Kolcun,A.: *ParametrickĂŠ modelovanie kriviek.* OU, Ostrava 2013.

[16] Kufner,J., Kadlec,A.: *Fourierovy dž˝ady.* Academia, Praha 1969.

[17] Martďž˝ďž˝ek, D.: *Matematickďž˝ principy grafickďž˝ch systďž˝mďž˝.* Littera, Brno 2002. ISBN 80-85763-19-2.

[18] Middleton, L., Sivaswamy, J.: *Hexagonal Image Processing.* Springer 2005. ISBN 1-85233-914-4.

[19] Lukasovďž˝,A.,Šarmanovďž˝,J.: *Metody shlukovďž˝ analďž˝zy.* SNTL, Praha 1985.

[20] Novďž˝k,M., Faber,J., Kufudaki,O.: *Neuronovďž˝ sďž˝tďž˝ a informaďž˝nďž˝ systďž˝my dž˝ivďž˝ch organizmďž˝.* Grada, Praha 1992.

[21] Pultr,A.: *Podprostory Euklidovskďž˝ch prostorďž˝.* SNTL, Praha 1986.

[22] Skala,V.: *Algorithms for 2D Line Clipping.* Proceedings of Eurographics'89, Elsevier Science Publishers, 1989. pp. 355-366. ISBN 0-444-8813-5

[23] Skala,V.: *Algoritmy potaov grafiky.* ZU, Plze 1992.

[24] Skala,V.: *Svtlo, barva a barevnďž˝ systďž˝my v poďž˝ďž˝taďž˝ovďž˝ grafice.* Academia, Praha 1993.

[25] Sobota,B., Miliďž˝n,J: *Grafickďž˝ gormďž˝ty.* Kopp, ďž˝eskďž˝ Budďž˝jovice 1996.

[26] Sproull, R., F., Sutherland, I., E.: A Clipping Driver. In:*AFIPS Conference Proceedings,* Vol. 33, 1968. pp 765-776.

[27] Sutherland, I., E., Hodgman, G., W.: Reentrant polygon clipping. *Communications ACM*, 17(1), 1974. pp 32-42.

[28] Watt, A.: *Fundamentals of Three-Dimensional Computer Graphics.* Addison Wesley, 1990. ISBN 0-201-15442-0.

[29] Zara, J., Benes, B., Sochor, J., Felkel, P.: *Modernďž˝ poďž˝ďž˝taďž˝ovďž˝ grafika.* Computer Press, Brno 2004. ISBN 80-251-0454-0.

[30] Krizenecky, J.: Anaglyf aneb barevnďž˝ separace obrazďž˝. http://stereofotograf.eu/navody/anaglyf/

[31] http://www.paladix.cz/serial/10047.html (20.10.2005)

[32] http://www.cgsd.com/papers/gamma.html (20.10.2005)

[33] http://www.graphics.cornell.edu/ westin/gamma/gamma.html (20.10.2005)

[34] http://www.w3.org/Conferences/WWW4/Papers/53/gq-gamma.html (20.10.2005)