Obsah

Pozn. : Jednotlivé příspěvky jsou řazeny chronologicky podle programu konference. Texty neprošly redakční ani jazykovou úpravou.

DATOVÉ STRUKTURY PRO POČÍTAČOVOU GRAFIKU

Ing. Pavel Slavík, CSc. ČVUT Praha

V tomto příspěvku se budeme zabývat datovými strukturami pro popis dvou- a třírozměrných objektů. Tvar a typ datových struktur je ovlivněn zvolenou reprezentací objektu. Máme na mysli případ, kdy například dvourozměrný objekt je reprezentován buď v rastrové nebo vektorové formě. Jak uvidíme v dalším, lze však vytvořit obecnější techniky, které jsou použitelné pro oba typy reprezentací. Další důležitou skutečností je fakt, že řada postupů použitých pro práci s datovými strukturami pro dvourozměrné objekty je použitelná (v modifikované podobě) i pro práci se strukturami pro objekty třírozměrné.

Zkoumání vlastností datových struktur pro popis objektů nabývá v poslední době na důležitosti, poněvadž řada přístupů vhodných pro počítačovou grafiku je vhodná i pro příbuzné aplikace jako je počítačové vidění nebo zpracování obrazu. To odpovídá i nejnovějším trendům, kdy vývoj pracovních stanic pro příští desetiletí předpokládá nejen výstup grafické informace, ale i vstup grafické informace v komplikovaném tvaru.

Jednou z možných technik popisu rovinných oblastí je použití čtvercové mřížky. Popis oblasti pak může být zadán ve formě seznamu čtverců nacházejících se uvnitř oblasti. Čtverec v seznamu je například určen svými souřadnicemi i, j (chápeme-li mřížku jako matici a čtverec jako prvek matice). Základním problémem této reprezentace je její přesnost. Jedná se především o to, s jakou přesností je objekt určen na svých hranicích. Jedná-li se například o objekty, jejichž hranice obsahuje kruhové oblouky, pak je přesnost reprezentace v obecném případě poměrně nízká. Zvýšit přesnost lze zmenšením rozměrů čtverců, což obvykle vede ke značnému zvýšení jejich počtu. Je patrno, že rozměr čtverců je v síti všude stejný - nejen na hranici oblasti, ale i uvnitř oblasti, kde ke zmenšení rozměru čtverců nebyl žádný důvod.

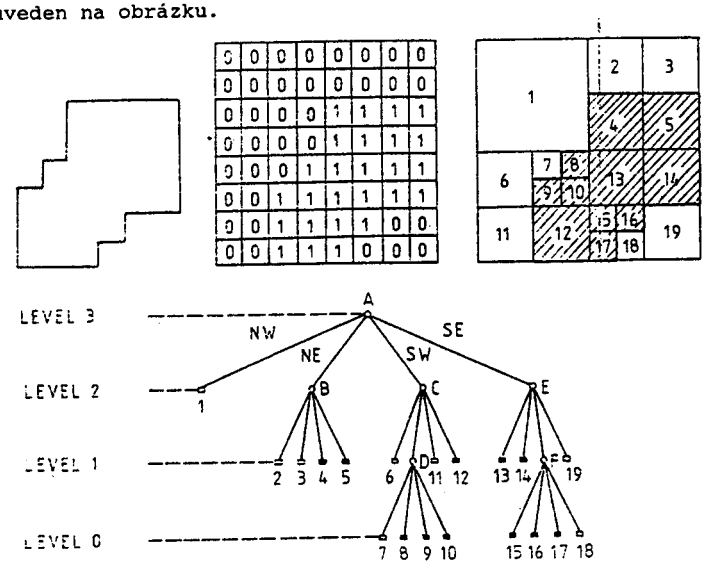
Proto by bylo žádoucí, aby ke zmenšení rozměru čtverců (a následně ke zvětšení jejich počtu) docházelo pouze tam, kde je to vhodné (tzn. například na hranici oblasti). Lze tedy říci, že zmenšení rozměru čtverce by mělo probíhat adaptivně. Proto se metodám, které vycházejí z předchozí úvahy, říká metody adaptivního dělení obrazového prostoru. Jednou z těchto metod je metoda kvadrantových stromů (quadtree) .

Pokud hovoříme o kvadrantových stromech, máme obvykle na mysli kvadrantový strom popisující nějakou rovinnou oblast. Jak uvidíme v dalším, existují i varianty kvadrantových stromů popisující i jiné objekty (např. čárové). V dalším budeme rozumět pod pojmem kvadrantový strom takový kvadrantový strom, který popisuje rovinnou oblast. Pokud se budeme zabývat jinými typy, bude to vždy zdůrazněno.

Kvadrantový strom obecně je založen na rekurzivním dělení určité oblasti na čtyři stejné kvadranty. Pokud vzniklý kvadrant leží celý buď uvnitř oblasti nebo mimo oblast, pak kvadrant není dále dělen. Nastane-li situace, kdy pouze část kvadrantu leží uvnitř oblasti, je třeba kvadrant dále rozdělit na čtyři stejné kvadranty. Proces dělení kvadrantů končí buď v situaci, kdy všechny kvadranty patří jednoznačně k oblasti nebo mimo oblast, anebo když bychom se dalším procesem dělení dostali pod hranici rozlišitelnosti (například pod rozměr pixelu).

Proces dělení kvadrantu může být formálně popsán prostřednictvím stromu. Kvadrant je reprezentován uzlem stromu. Je-li kvadrant rozdělen, vycházejí z uzlu čtyři hrany. Tento strom (kvadrantový strom) obsahuje tak pouze uzly stupně čtyři a stupně nula. Této významné vlastnosti pak využíváme pro různé reprezentace kvadrantového stromu.

Příklad rovinné oblasti a příslušného kvadrantového stromu je uveden na obrázku.



V kvadrantovém stromě tak odpovídá kořen stromu celému vyšetřovanému prostoru, který má rozměr $2^N \times 2^N$ délkových jednotek. Každý syn uzlu stupně čtyři reprezentuje kvadrant. Ty jsou označené symboly NW, NE, SW, SE v uvedeném pořadí (jedná se

o mnemotechnické zkratky identifikující polohu kvadrantu - northwest, northeast ap.)

V souvislosti s kvadrantovými stromy je nutno řešit otázku datových struktur, jejichž pomocí je kvadrantový strom uložen v paměti. Nejjednodušší je z hlediska implementace datová struktura reprezentující strom pomocí ukazatelů - z každého uzlu stupně čtyři vedou čtyři ukazatelé na podřízené uzly. Jednoduchost implementace a manipulace se stromy je však vyvážena značnými paměťovými nároky. Proto se tento způsob reprezentace příliš nepoužívá.

V praxi se používají různé linearizované zápisy stromu, které nejsou náročné na paměť. Velmi používaná je reprezentace založená na procházení stromem typu preorder (tzv. depth-first). Výsledkem je textový řetězec skládající se ze symbolů 'G', 'W' a 'B', které odpovídají "šedým", "bílým" a "černým" kvadrantům. Bílým kvadrantem nazýváme kvadrant ležící mimo dvojrozměrný útvar v oblasti. Černým kvadrantem nazýváme naopak kvadrant uvnitř útvaru a šedý (smíšený) kvadrant leží pouze částečně v oblasti.

Procházeli-li bychom uvedeným způsobem stromem, který reprezentuje oblast na obrázku, na němž jsme reprezentovali vlastnosti kvadrantového stromu, pak získáme řetězec

GWGWWBBGWGWBBBWBGBBGBBBWW

Popis generování řetězce je uveden v následující proceduře zapsané v pascalském tvaru:

```
procedure BUILDSTRING (P: node);
var I:integer;
begin
  if P=GRAY then
    begin
      write('G');
      for I in ('NW','NE','SW','SE') do
        BUILDSTRING(SON(P,I));
    end
  else
    write (NODETYPE(P))
end;
```

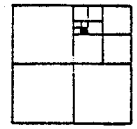
Uvedený zápis kvadrantového stromu je často nazýván DF-výrazem (DF-expression). Vzhledem k tomu, že v řetězci se vyskytují pouze tři symboly ('G','W','B'), postačí na jejich kódování pouze dva bity. Z toho vyplývá, že tento způsob reprezentace je velmi úsporný. Je však třeba mít na paměti, že tohoto způsobu lze s výhodou využít pouze v aplikacích vyžadujících procházení celého stromu. V případě, kdy je požadován náhodný přístup ke kvadrantům, není tato reprezentace vyhovující.

Další lineární reprezentací kvadrantového stromu je reprezentace pomocí tzv. umísťovacího kódu (locational code). Zde se vychází z toho, že v kvadrantovém stromu nesou vlastní informaci o vyplnění kvadrantů příslušnou barvou pouze listy. Vnitřní uzly tuto informaci nenesou. Je tedy výhodné kódovat pouze listy kvadrantového stromu. Další úspora spočívá v tom, že nejsou kódovány kvadranty vyplněné barvou pozadí. Například v případě, kdy se jedná o černobílý obrázek, kdy je kresba vytvořena černě na bílém pozadí, jsou kódovány pouze černé kvadranty.

Kódování kvadrantů spočívá v tom, že je kódována cesta od kořene stromu k listu, který reprezentuje kvadrant. Kódování směrů, které odpovídají volbě jednoho ze čtyř kvadrantů je prováděno dle následující tabulky:

kód	0	1	2	3
směr	NW	NE	SW	SE

Například cesta odpovídající volbě kvadrantu ve směru <NE,NW,SW,SE> je kódována posloupností číslic <1,0,2,3>. Poloha kvadrantu je patrna z obrázku:



Umístění kvadrantu lze však úsporně zachytit číselným kódem, který lze vypočítat ze vztahu:

$$C = C_i.4^l \qquad 0<=C_i<=3;\ 0<=C<4^{n-m}$$

kde n udává rozměr obrázku ($2^n x 2^n$) a m je úroveň listu ve stromu (m<n). Koeficienty C_i jsou v posloupnosti číslovány zprava (od nuly).

Výše uvedená posloupnost číslic (cesta) <1,0,2,3> je uvedeným vztahem transformována na dekadickou hodnotu 75. Je třeba si však uvědomit, že kódování kvadrantu uvedeným způsobem vyžaduje doplňkovou informaci o úrovni listu ve stromu. Důvodem je skutečnost, že při rozkladu čísla (kódu) na jednotlivé číslice (ve čtyřkové soustavě) nevíme, jaký počet vedoucích nul určuje cestu. Tzn., že umístění kvadrantu je dáno dvojicí čísel (kód, úroveň).

Příklad cest o stejném umísťovacím kódu:

<1,0,2,3> <0,0,0,1,0,2,3>

Tento problém je vyřešen pomocí změněného způsobu kódování. Příslušná tabulka má pak následující formu:

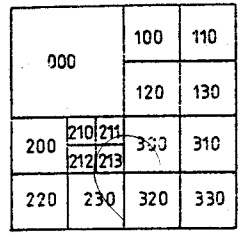
kód	1	2	3	4
směr	NW	NE	SW	SE

To pak znamená, že cesta <NE,NW,SW,SE> se dá vyjádřit jako <2,1,3,4>. Příslušný kód má pak hodnotu 294.

Výpočet hodnoty kódu se pak provádí podle vztahu

$$C = C_i.5^l; \quad 1<=C_i<=4;\ 5^{n-m-1}<=C<=5^{n-m}$$

Nevýhodou uvedeného kódu je provádění rozkladu na jednotlivé číslice pomocí celočíselného dělení pěti (a získávání příslušných zbytků po dělení). Ideálním případem kódování a dekódování z hlediska rychlosti je situace, kdy se používají operace součtu, rozdílu a aritmetických posuvů. Způsob kódování kvadrantů, který odpovídá uvedenému schématu, se nazývá FD umisťovací kód (FD locational code). Kód každého listu (kvadrantu) vybarveného barvou kresby je zapsán pomocí stejně dlouhého kódu - odpadá tak nutnost uvádět informaci o úrovni listu. Uvedený způsob kódování je použit v geografických informačních systémech. Vzhledem k tomu, že kvadrant je dělen na čtyři kvadranty (používáme tak čtyř směrů NW, NE, SW, SE), lze pro popis výběru kvadrantu další úrovně (výběr směru) použít dva bity pokrývající hodnoty reprezentující čtyři směry. Příklad kódování uvedeným způsobem je uveden na obrázku.



Speciální případy kvadrantových stromů

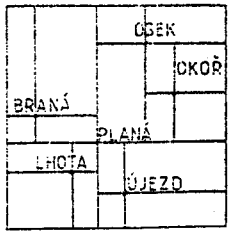
V kartografických a podobných aplikacích je často nutné mít rychlý přístup k informacím o obsahu jednotlivých částí obrazového prostoru (v případě kartografie mapy). Tradiční způsob spočíval v tom, že se obrazový prostor rozdělil mřížkou skládající se ze čtverců stejné velikosti. V každém čtverci byl nejvýše jeden významný údaj (například symbol označující město).

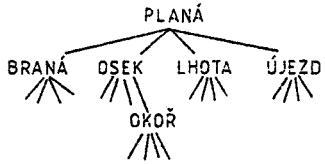
Datová struktura reprezentující takto chápaný obrázek je v podstatě čtvercová matice, kde prvky matice obsahují odkazy na informace o objektech, které se vyskytují v odpovídajícím čtverci.

Zde se vyskytuje obdobný problém, který vedl k použití adaptivních metod (např. kvadrantových stromů) při popisu spojitých oblastí . Jedná se o to, že informace není v obrazovém prostoru rozprostřena rovnoměrně, takže značná část čtverců je prázdná a kvůli lokálnímu výskytu zhuštěné informace je třeba provést poměrně jemné dělení v celém obrazovém prostoru. Proto se i v tomto případě používá adaptivního dělení prostoru. Aby v tomto speciálním případě kvadrantového stromu nedošlo k záměně s běžným kvadrantovým stromem, používá se označení "bodový kvadrantový strom" (point quadtree).

Představíme-li si například města jako body na mapě, pak čáry dělící obrazový prostor procházejí těmito body. Dělící proces lze pak formálně popsat prostřednictvím stromu, který má stejné vlastnosti jako běžný kvadrantový strom. Příklad je uveden na obrázku.

Uvedený způsob popisu 2D objektů lze rozšířit i do třírozměrného prostoru. Rovněž lze tutó metodu založenou na použití "čtvrcení" kvadrantu modifikovat na "půlení" částí obrazového prostoru, přičemž výsledkem bude popis objektu ve formě binárního stromu.

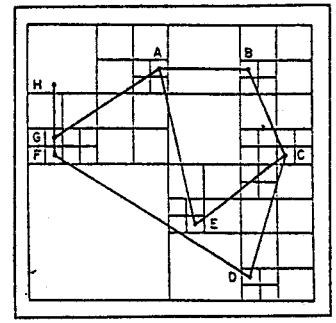




Obdobný přístup k popisu dvourozměrných objektů můžeme volit i v případě obrázku skládajícího se z čárových (vektorových) objektů, jako jsou například mnohoúhelníky. V tomto případě je nevýhodné používat klasický kvadrantový strom, který by musel obsahovat dělení na velmi malé kvadranty (případně až na úroveň pixelu) v oblastech, kde by se vyskytovaly čáry. Proto je zvolen způsob, kdy jsou kvadranty děleny na dílčí kvadranty o stejné velikosti. Dělení pokračuje tak dlouho, dokud nezískáme takové rozdělení obrazového prostoru, kde každý kvadrant neobsahuje více

než jednu čáru. V případě vrcholu, z něhož vychází více hran, může být v kvadrantu maximálně jeden takový vrchol.

Příklad dělení obrazového prostoru je uveden na obrázku. Listy v odpovídajícím kvadrantovém stromu obsahují informaci o úsečce, která prochází příslušným kvadrantem, ve formě tzv. popisovače čáry (line descriptor). Tento popisovač jednoznačně popisuje úsečku.



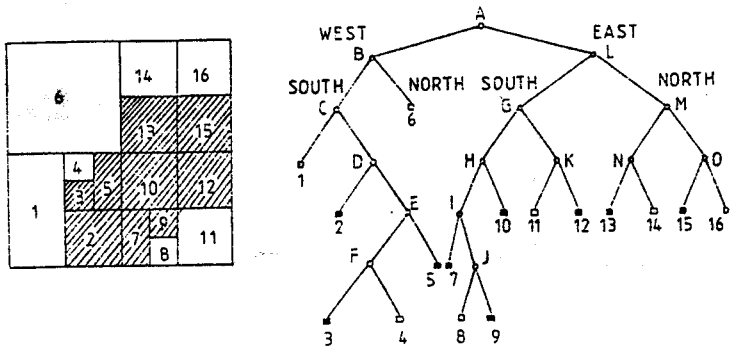
Tato metoda má určité nedostatky v případě, kdy obrázek obsahuje úsečky, které jsou velmi blízko u sebe. V takovém případě je metoda modifikována tak, že kvadrant může obsahovat určitý maximální počet úseček.

Stejně jako jiné metody založené na adaptivním dělení obrazového prostoru může být i tato metoda rozšířena do prostoru třírozměrného. Dělení prostoru pak probíhá obdobně jako ve dvourozměrném případě, to znamená, že v základní variantě metody může oktant obsahovat část maximálně jedné hrany. I v tomto třírozměrném případě platí, že požadavky na paměť jsou nižší, než v případě klasického oktantového stromu (viz příslušnou kapitolu).

Binární stromy rozkladu 2D objektu

Rozklad rovinné oblasti pomocí kvadrantového stromu má tu základní vlastnost, že v každé fázi rozkladu se kvadrant dělí na čtyři stejné části (subkvadranty). Konečným výsledkem rozkladu je pak systém obecně různě velkých čtverců, jejichž délky stran jsou mocninami dvou. Binární stromy rozkladu 2D objektu (bintree) jsou alternativním rozkladovým schématem ke kvadrantovým stromům. Rozdíl spočívá v tom, že v každé fázi rozkladu se aktuální oblast dělí na dva stejné díly. V případě aplikace metody ve dvou

rozměrech střídáme směr dělení oblasti (podle x-ové osy, podle y-ové osy, podle x-ové osy atd.). Příklad dělení oblasti je uveden na obrázku.



Existují různé varianty této základní metody. Existuje například varianta, kdy se pravidelně nestřídá dělení ve směru osy x a ve směru osy y. Dělení probíhá adaptivně v příslušném směru podle charakteru obrázku. To pak vede k výzkumu metod, jejichž pomocí lze získat strom s minimálním počtem uzlů.

Zobecněním těchto metod je metoda označovaná jako BSP (Binary Space Partition). Formálním popisem rozdělení dvourozměrného prostoru je pak BSP strom (BSP tree). Tato metoda nachází uplatnění v širokém spektru aplikací, jako například při řešení problému viditelnosti při zobrazování třírozměrných těles. V každé fázi rozkladu je oblast rozdělena na dvě části, jejichž velikost může být v podstatě libovolná. Přímky, které dělí oblast, nemusí být ani rovnoběžné se souřadnicovými osami, ani není vymezen vztah mezi dělícími přímkami ve dvou po sobě následujících rozkladech (nemusí být například na sebe kolmé). Výsledek rozkladu je pak množina různých různě velkých konvexních mnohoúhelníků.

BSP strom je binárním stromem. Dělící přímky jsou chápány jako hranice mezi dvěma polorovinami. Uvažujme rovnici přímky

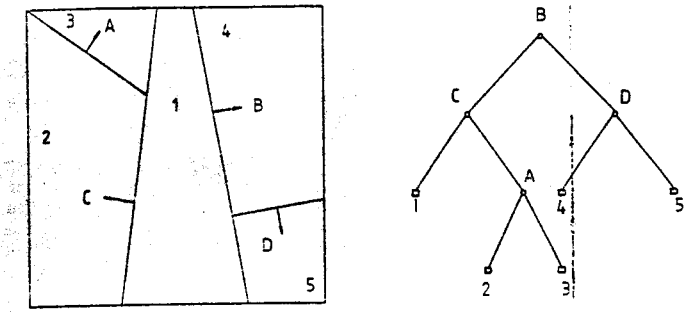
$$ax + by + c = 0$$

Můžeme pak říci, že pravý podstrom je na kladné straně přímky (v "kladné polorovině") a levý podstrom je na záporné straně (v "záporné polorovině"). V kladné straně pak budou ležet dělící úsečky dělící oblast v dalších fázích rozkladu, jejichž body budou vyhovovat podmínce

$$ax + by + c >= 0$$

Obdobně tomu bude i pro zápornou stranu dělící přímky. Představu o probíhajícím dělení poskytuje obrázek. V obrázku jsou šipkami

znázorněny kladné poloroviny (a tak i vlastní orientace přímky). Uvedená metoda je obdobně jako jiné "dvourozměrné" metody modifikovatelná i pro třírozměrný případ, kdy dělení třírozměrného poloprostoru na podprostory je realizováno rovinami.



Vytváření kvadrantových stromů a jejich transformace

Existují různé postupy pro generování kvadrantových stromů. O základních postupech se stručně zmíníme. Především je nutno si uvědomit, v jakém tvaru je zadáván vstupní popis obrázku. Budeme se zabývat obrázky jak v rastrové formě tak ve vektorovém tvaru.

V případě rastrového obrázku zadaného ve formě matice vyhledáváme čtveřice sousedních kvadrantů a v případě, že mají stejnou barvu, provádíme jejich sdružení do většího kvadrantu. Tento proces začíná na úrovni pixelu a je zřejmé, že má rekurzivní charakter. Je patrno, že hlavní složkou uvedeného postupu je stálé testování barev sousedních kvadrantů, což má bezesporu značný vliv na výpočetní náročnost algoritmu. Předmětem zkoumání jsou proto různé prediktivní metody, které předpokládají existenci homogenního uzlu (kvadrantu) o určitém rozměru, kdykoliv se při zpracování rastru narazí na levý horní roh potenciálního kvadrantu (předpokládá se procházení rastrové matice zleva doprava na řádku a průchod řádky shora dolů). Tento způsob však vyžaduje existenci pomocných datových struktur.

Vytváření kvadrantového stromu z vektorových dat je komplikovanější než v rastrovém případě. Vstup dat charakterizujících vstupní obrázek je ve tvaru seznamu úseček, z nichž se obrázek skládá. To ovšem znamená, že tak neexistuje žádné obecně vhodné prostorové uspořádání úseček, ze kterého by se dalo vycházet při tvorbě kvadrantového stromu. Vyšetřujeme-li nějakou (čtvercovou) oblast R reprezentující kořen aktuálního

podstromu, provedeme operaci ořezání seznamu úseček vzhledem k této oblasti. Pokud žádná z úseček neinciduje s danou oblastí, je vytvořen "bílý" list grafu a oblast R (kvadrant) není dále zpracovávána. Pokud oblast R má rozměry pixelu a inciduje alespoň s jednou úsečkou, je vytvořen "černý" list. V dalších případech je vytvořen tzv. "šedý" uzel, který je dále rekurzivně dělen, dokud nenastane některý ze dvou předchozích případů ("bílý", resp. "černý" list). Je zřejmé, že mírnou modifikací uvedeného postupu můžeme vytvářet kvadrantové stromy, které nejsou orientovány "pixelově", ale vektorově. To znamená, že v listech kvadrantového stromu jsou uloženy popisovače čar (line descriptor).

Pro tvorbu rastrového kvadrantového stromu z vektorových dat můžeme použít i jiné metody. Nejprve je třeba převést úsečky do tvaru tzv. řetězového kódu (chain code), přičemž vycházíme z tradičního algoritmu pro generování úsečky. V další fázi se zjednodušeně řečeno provádí syntéza, kdy se na základě informace o poloze jednotlivých pixelů syntetizuje kvadrantový strom.

Je-li obrázek reprezentován formou kvadrantového stromu, je třeba řešit otázky týkající se možností transformace obrázku. V následujícím textu si probereme některé základní principy nejčastěji používaných transformací. Jednou ze základních transformací je okénkování (windowing). Je to proces, jehož prostřednictvím získáváme část obrazu vymezenou zadaným obdélníkem. Jedná se vlastně o speciální případ ořezávání (clipping). Je-li výchozí obrázek reprezentován kvadrantovým stromem, požadujeme, aby i výsledný obrázek byl reprezentován kvadrantovým stromem.

Předpokládejme, že transformace je aplikována na obrázek v obrazovém prostoru $2^N x 2^N$ za použití okna (čtverce) $2^K x 2^K$ (K<N). Okno může být umístěno libovolně v obrázku. Základem algoritmu je postup, kdy je vstupní kvadrantový strom rozkládán a v průběhu rozkladu jsou části nalézající se v okně zpracovávány a začleňovány do výstupního kvadrantového stromu.

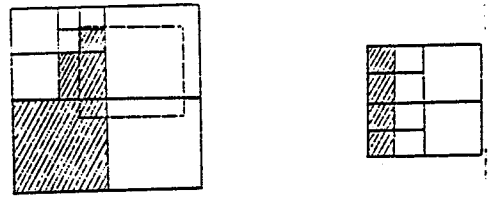
Ve stručnosti lze algoritmus popsat následovně:

1. Najděte nejmenší podstrom vstupního podstromu, který obsahuje zadané okno.

2. Představuje-li kořen podstromu okna list stromu, jsme s daným podstromem hotovi. V opačném případě se kvadrant (reprezentující kořen podstromu) rozdělí na čtyři části (kvadranty) a kroky 1 až 3 se aplikují na každý takto získaný kvadrant.

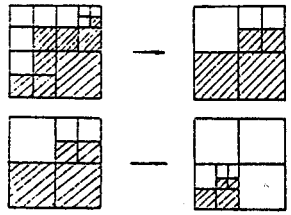
3. Získané listy je nutno začlenit do výstupního kvadrantového stromu.

Popsaný postup je patrný z obrázku. Uvažujme obrázek (označme jej I) o velikosti 8x8 pixelů, který obsahuje 10 listů. Dále uvažujme okno o velikosti 4x4 vyznačené na obrázku přerušovanou čárou. Na počátku je okno (označme jej W) reprezentováno kořenem celého kvadrantového stromu. Poněvadž tento kvadrantový strom není obsažen v některém z uzlů kvadrantového stromu obrázku, provedeme dělení okna W na kvadranty a použije se algoritmus od počátku na všechna tato okna - označme je například W_{NW}, W_{NE}, W_{SW} a W_{SE}. Poněvadž W_{NW} není obsažen v uzlu patřícím do I, musíme W_{NW} dále rozdělit. Nyní je každý z takto vzniklých kvadrantů obsažen v listu I. Dále vidíme, že ani W_{SE} není obsažen v uzlu patřícím do I. Proto W_{SE} rozdělíme na čtyři kvadranty a tyto kvadranty jsou již obsaženy v uzlech I. Poněvadž všechny čtyři kvadranty mají stejnou barvu (bílou), můžeme z nich vytvořit list jejich spojením. Výsledek je uveden na obrázku.



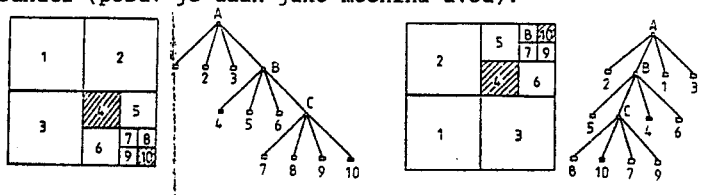
Uvedený postup může sloužit jako základ pro vytvoření algoritmů pro posuv kvadrantového stromu. Algoritmus udržuje seznam "nebílých" uzlů zdrojového stromu, které pokrývají uzly vytvářeného cílového stromu. Každý uzel v cílovém stromě je testován, zda je obsažen v černém uzlu ve zdrojovém stromě. Pokud ano, je tento uzel černým uzlem v cílovém stromě. Pokud tomu tak není, je třeba rozlišovat dva případy. Jsou-li uzly ve zdrojovém stromě bílé, je uzel v cílovém stromě bílý. V opačném případě je zkoumaný uzel (kvadrant) cílového stromu rozložen na čtyři kvadranty a proces je opakován rekurzivně. Pokud lze provádět ve výsledném stromě sloučení kvadrantů stejné barvy do většího kvadrantu, je nutno toto sloučení provést.

Kromě tohoto algoritmu existují i další přístupy, založené například na analogii provádění množinových operací nad kvadrantovými stromy. Další transformací je změna měřítka. Jednoduchý případ nastává, je-li měřítko mocninou dvou. V případě zvětšení je vybrán příslušný podstrom, který se tak stává stromem, který reprezentuje obrázek v obrazovém prostoru $2^N x 2^N$. Postup je zřejmý z obrázku.



V případě zmenšení se naopak kvadrantový strom reprezentující celý obrázek stává podstromem výsledného stromu. Postup je zřejmý z obrázku.

V případě obecnějšího měřítka je postup mnohem složitější. Zde je třeba stanovit poměr mezi velikostí kvadrantů ve zdrojovém a cílovém stromě a podle toho provádět mapování mezi kvadranty obou stromů. Z obrázku je rovněž patrno, že lze zvláštních vlastností této transformace pro zvláštní případ (měřítko je mocninou dvou) využít při zvláštních případech transformace posunutí (posuv je udán jako mocnina dvou).



Velmi jednoduchá je transformace otáčení v případě, že úhel otáčení je celistvým násobkem devadesáti stupňů. Při tomto způsobu otáčení se pouze příslušným způsobem vzájemně zamění kvadranty. Na obrázku je uveden příklad otočení kvadrantového stromu o 90° proti smyslu hodinových ručiček. Algoritmus prochází stromem (dle algoritmu "preorder") a provádí rotaci ukazatelů v každém vnitřním uzlu stromu.

Prohlédneme-li si pečlivě obrázek, zjistíme následující skutečnosti:

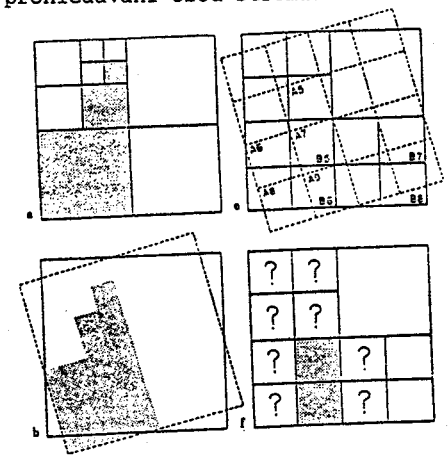
1) Všechny pixely ze vstupního kvadrantu NW jsou v cílovém kvadrantu SW.
2) Všechny pixely ze vstupního kvadrantu NE jsou v cílovém kvadrantu NW.
3) Všechny pixely ze vstupního kvadrantu SE jsou v cílovém kvadrantu NE.
4) Všechny pixely ze vstupního kvadrantu SW jsou v cílovém kvadrantu SE.

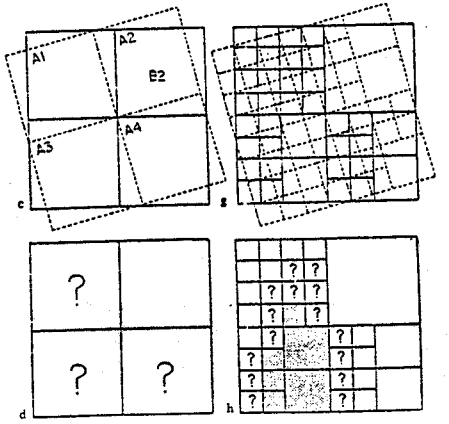
Obdobná pravidla lze vytvořit i pro jiné celistvé násobky 90°.

V dalším se budeme zabývat otočením kvadrantového stromu o obecný úhel. Vstupní kvadrantový strom bude označen jako A a cílový kvadrantový strom bude označen jako B. Čárkované čáry označují dekompozici zdrojového stromu A a nepřerušované čáry označují dekompozici cílového stromu B.

Algoritmus začíná vyšetřováním, zde je B listem. Není-li tomu tak, jsou jak A, tak i B rozděleny na čtyři kvadranty. Tento proces dělení je opakován tak dlouho, dokud všechny kvadranty v B nereprezentují listy, nebo dokud jsme při dělení nedosáhli úrovně pixelu. Symboly ? vyskytující se v obrázku znamenají, že kvadrant má být dále dělen. Kvadranty vzniklé rozdělením B jsou testovány, zda jsou pokryty kvadranty z A o stejné barvě. V pozitivním případě končíme pro daný kvadrant s dělícím procesem a v opačném případě pokračujeme dále v dělení. Na nejnižší úrovni dělení pak je třeba rozhodnout, jakou barvu přiřadíme kvadrantům stále označeným otazníkem. Je zřejmé, že se jedná o oblasti, které jsou na hranici objektu.

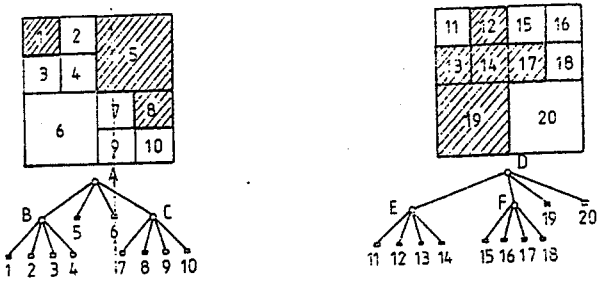
Kromě výše uvedených transformací je třeba se ještě zmínit o množinových operacích, které jsou prováděny nad kvadrantovými stromy. Základem metody je paralelní procházení oběma stromy, přičemž se testují stejnolehlé uzly v obou stromech. Jedná-li se například o operaci průniku, pak v případě, že jsou oba uzly "bílé", je výsledný uzel bílý. Je-li jeden z uzlů "černý", jsou v příslušném podstromu v druhém stromě vyhledávány černé uzly, které jsou začleněny do výsledného stromu. Jsou-li oba stejnolehlé uzly "šedé", je třeba rekurzivně pokračovat v paralelním prohledávání obou stromů.



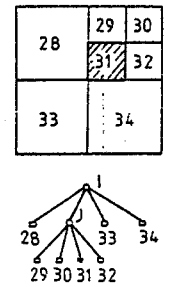


Zhruba lze říci, že v případě, kdy je jeden ze stejnolehlých uzlů "černý", je do výstupního stromu zkopírován podstrom z druhého stromu, kde kořenem podstromu je vyšetřovaný stejnolehlý uzel. Obdobný postup lze vytvořit i pro další množinové operace. Příklad provádění množinových operací nad kvadrantovými stromy je uveden na obrázku.

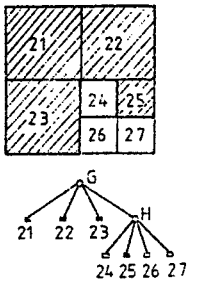
Schopnost rychle provádět množinové operace je jedním z hlavních důvodů pro velmi rozšířené používání datových struktur typu kvadrantového stromu. V případě použití alternativních metod specifikujících hranice objektů - jako je například řetězový kód (chain code) - je výpočet výsledku množinových operací mnohem náročnější. Díky hierarchické struktuře kvadrantového stromu je možno zpracovávat grafickou informaci v širším kontextu. Například v případě řetězového kódu je informace o objektu lokální, poněvadž každá úsečka tvořící řetězec představuje informaci pouze o části obrazu, s nímž sousedí - to znamená informaci o tom, že ohraničená oblast je vpravo.



průnik:



sjednocení:



Dá se určit, že množinové operace prováděné pomocí kvadrantových stromů mají nízkou výpočetní složitost. Konkrétně nejhorší případ pro algoritmy průniku a sjednocení je přímo úměrný součtu uzlů ve dvou vstupních kvadrantových stromech. Poněvadž zmíněné algoritmy jsou založeny na procházení stromem typu "preorder", algoritmy budou prováděny se stejnou efektivitou bez ohledu na konkrétní reprezentaci stromu (lineární reprezentaci, dynamické proměnné apod.).

Na závěr je ještě třeba připomenout, že všechny uvedené postupy pro provádění operací (transformace a množinové operace), kde bylo třeba současně pracovat se dvěma stromy, předpokládaly, že se pracuje s tzv. zarovnanými kvadrantovými stromy (aligned quadtrees). Těmito stromy rozumíme případ, kdy obrazové prostory $2^N x 2^N$ jsou u obou stromů stejné. Existuje další třída algoritmů, které využívají nezarovnaných kvadrantových stromů (unaligned quadtrees).

Reprezentace kvadrantových stromů na vnějších médiích

V předchozím textu jsme si uvedli charakteristické vlastnosti kvadrantových stromů. V případě, že obrázek má nepříliš velké rozměry a nepříliš komplikovanou strukturu, není příliš velkým problémem uložit obrázek do vnitřní paměti počítače. Ve speciálních aplikacích, jako je například kartografie, jsou však paměťové požadavky podstatně větší. To má pak za následek ukládání datové struktury reprezentující kvadrantový strom na vnější média. Pro aplikaci kvadrantových stromů v kartografii lze specifikovat následující požadavky na uložení informace:

1) možnost ukládání velkého objemu dat
2) možnost vynechání libovolného množství detailů
3) uložení informace musí být kompaktní

První požadavek vyplývá z charakteristiky dané aplikace. Vzhledem k časovým charakteristikám diskových operací je výhodné,

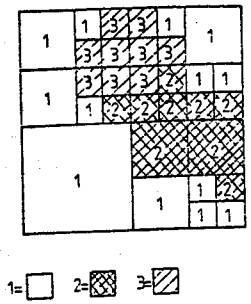
když je počet diskových operací v průběhu zpracování dat minimální - to znamená, že je třeba využít znalostí o využívání vyrovnávací paměti při diskových operacích.

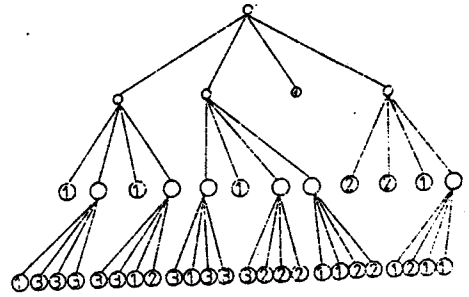
Druhý požadavek vyplývá z prvního požadavku. Je-li objem uložených dat příliš velký, nebylo by zřejmě výhodné procházet celým kvadrantovým stromem pro každou operaci. Je tedy potřebné, aby bylo možno přeskočit detaily. Jako příklad lze uvést situaci, kdy se zobrazují obrazová data v rastru 512x512 pixelů. Je zřejmé, že není třeba procházet kvadrantovým stromem do větší hloubky než 9, neboť bychom se dostali pod rozměry pixelu. Jiným případem je situace, kdy je třeba zobrazit pouze část kvadrantového stromu, kdy požadovaná část je určena zadaným oknem. Zde je třeba při zpracování pokud možno co nejrychleji vynechat část kvadrantového stromu, která leží mimo zadané okno. V případě kartografie je typickou úlohou vytváření map určité oblasti v různém měřítku. Rozdíl mezi velmi podrobnou mapou a méně podrobnou mapou spočívá v tom, že na méně podrobné mapě jistým způsobem splynou detaily z velmi podrobné mapy. Toho lze na úrovni uložení informace ve formě kvadrantového stromu docílit tak, že na různých úrovních jsou uzlům stromu, které reprezentují kvadranty, přiřazeny jisté průměrné hodnoty odvozené z hodnot v uzlech podřízených danému uzlu. To pak umožňuje zobrazovat data s různou rozlišitelností, přičemž potřebná informace je k dispozici na dané úrovni.

Poslední požadavek je rovněž vztažen k požadavku prvnímu. Je zřejmé, že čím bude uložená informace kompaktnější, tím bude soubor menší a tím bude i menší čas potřebný pro přístup k informacím. Nicméně je třeba udržovat určitou rovnováhu mezi kompaktností a jednoduchostí. Jednoduché datové struktury se rychleji zpracovávají a také se jednodušeji udržují.

Datové struktury pro reprezentaci kvadrantových stromů mohou mít nejrůznější tvar (jak bylo konečně uvedeno pro případ kvadrantových stromů uložených v paměti počítače). V případě, že používáme čtveřice odkazů v každém vnitřním uzlu stromu, stává se pak reprezentace kvadrantového stromu objemná. V případě, že kvadrantový strom uložíme jako sekvenční soubor, je objem informace menší, ale není pak možno přeskakovat různé úrovně informace. Je tedy zřejmé, že řešení leží někde mezi těmito extrémy. Tato třída reprezentací se nazývá "semi-pointered tree" a obsahuje pouze nejnutnější odkazy z jednoho uzlu do druhého.

Jedním z představitelů této třídy stromů je tzv. Goblin quadtree. Způsob kódování si ukážeme na příkladě. Na obrázku je zobrazen grafický objekt v obrazovém prostoru $2^N x 2^N$ a příslušný kvadrantový strom.





V naší reprezentaci (Goblin quadtree) je kvadrantový strom reprezentován kořenem a určitým počtem kvadrantů, které jsou realizovány jako záznamy v diskovém souboru s přímým přístupem. Záznam obsahuje pět polí: hodnoty v uzlech odpovídajících kvadrantům NW, NE, SW a SE a odkaz. Kořen celého kvadrantového stromu je reprezentován záznamem na začátku souboru. Záznam obsahuje určitou identifikační hodnotu (informaci, že se jedná o kořen) a dále již neobsahuje žádné údaje (například o větvích). Kvadrantový strom uvedený na obrázku je pak reprezentován následujícím způsobem:

Root = -1

číslo	NW	NE	SW	SE	odkaz
1	-1	-1	1	-2	11
2	1	-3	1	-3	5
3	1	3	3	3	7
4	3	3	1	2	5
5	-3	1	-2	-1	9
6	3	1	3	3	7
7	3	2	2	2	8
8	1	1	2	2	9
9	2	2	1	-1	11
10	1	2	1	1	11

Jak je patrno, hodnoty v listech jsou reprezentovány jako kladné hodnoty a průměrné hodnoty ve vnitřních uzlech jsou reprezentovány jako čísla záporná. Nulové hodnoty nejsou povoleny. Způsob procházení stromem je zřejmý z příkladu (depth-first).

Při určování průměrných hodnot ve vnitřních uzlech lze použít různých strategií. V případě, že barvy (nebo odstíny barev) tvoří určitou lineární stupnici, lze počítat skutečný průměr (např.

odstíny šedi). Jsou-li použity různé barvy, pak jejich "průměrování" nemá význam. Proto se volí určitá převládající barva, která je pak reprezentativní hodnotou pro daný uzel. Určení, která barva bude převládající, vyžaduje zkoumání poměrů na nižší úrovni. Vhodnou strategií je přiřazení vah různým typům uzlů. Je-li uzel listem (je tudíž celý vybarven určitou barvou), je mu přiřazena větší váha, než uzlu, který není listem a barva je reprezentována určitou hodnotou získanou obdobným způsobem na další úrovni.

Jako příklad lze uvést situaci, kdy listu je přiřazena váha 3 a vnitřnímu uzlu je přiřazena váha 2. Pak v případě, že kvadrant se dělí do čtyř kvadrantů:
dvou listů o hodnotách A a B
dvou vnitřních uzlů o hodnotách B a C.

Získaný poměr je pak A(3), B(5) a C(2). Hodnota reprezentující daný uzel (kvadrant) bude B.

Podle odkazu se odkazujeme na kvadrant, který má být zpracován jako další v případě, že se nespokojíme v daném kvadrantu s "hrubou" informací ("průměrnou" barvou). V případě, že přeskočení detailu pro daný uzel ukončuje procházení stromem, je pak hodnota odkazu zbytečná, a proto je obsazena nějakou nevýznamnou hodnotou (v našem případě 11, poněvadž je deset záznamů).

Další reprezentací kvadrantových stromů na vnějších médiích je tzv. Autumnal quadtree (podzimní kvadrantový strom). Jeho modifikaci si nyní ukážeme. I v tomto případě je každý kvadrant reprezentován jako záznam v souboru s přímým přístupem. Rovněž zde má záznam pět složek. První čtyři reprezentují informaci o jednotlivých dílčích kvadrantech. Záporné hodnoty nesou informaci o barvě listu a kladné hodnoty jsou odkazem na záznam, který nese informaci o daném dílčím kvadrantu. Pátá složka obsahuje informaci o reprezentativní barvě aktuálního kvadrantu.

Jak je patrno, jedná se o reprezentaci, která obsahuje v každém vnitřním uzlu stromu odkazy na podřízené uzly. Nicméně lze říci, že tato reprezentace je stejně kompaktní a má obdobné vlastnosti jako "Goblin quadtree". Nicméně má "Goblin quadtree" některé výhody. Za prvé je informace o reprezentativní hodnotě kvadrantu uložena o jednu úroveň blíže kořenu, což snižuje počet čtecích operací při zpracování informace. Dále tato reprezentace nesměšuje odkazy a hodnoty uložené v listech. To má pak výhodu v případě, kdy odkaz a hodnoty mají rozdílné rozsahy a pole hodnot může být uloženo na menším počtu bitů než v případě smíšeného uložení.

Jako příklad reprezentace ve formě "Autumnal tree" si uvedeme reprezentaci obrázku, který byl již popsán ve formě "Goblin tree". Porovnáním obou reprezentací lze zjistit rozdíly mezi oběma reprezentacemi.

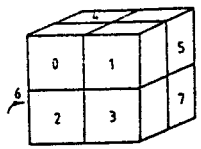
uzel	NW	NE	SW	SE	Průměr
1	2	5	-1	9	1
2	-1	3	-1	4	1
3	-1	-3	-3	-3	3
4	-3	-3	-1	-2	3
5	6	-1	7	8	1
6	-3	-1	-3	-3	3
7	-3	-2	-2	-2	2
8	-1	-1	-2	-2	1
9	-2	-2	-1	10	2
10	-1	-2	-1	-1	1

Uvedené datové struktury umožňují pohodlnou a efektivní práci s informací. Jedná se například o množinové operace nad kvadrantovými stromy reprezentovanými uvedenými způsoby. Rovněž rychlý přístup k požadovaným datům je jednou ze značných výhod uvedených metod. Porovnání obou uvedených metod vyznívají spíše ve prospěch první metody (Goblin quadtree) a to jak z hlediska paměťových nároků, tak i z hlediska náročnosti (rychlosti) výpočtu.

Reprezentace třírozměrných těles oktantovými stromy.

Obdobně jako dvourozměrné objekty mohly být popsány pomocí kvadrantových stromů, které byly založeny na adaptivním dělení obrazového prostoru, lze i třírozměrné objekty popisovat obdobnou technikou. V tomto případě hovoříme o oktantových stromech.

Definice oktantového stromu může mít následující tvar:
Nechť je dáno prostorové pole o rozměrech $2^n x 2^n x 2^n$, kde prvkem prostorového pole je jednotková krychle. Každá jednotková krychle má přiřazenu jednu ze dvou hodnot FULL, resp. VOID pro "plnou", resp. "prázdnou" jednotkovou krychli. Tak zvaný průměr prostorového pole $2^n x 2^n x 2^n$ je 2^n. Prostorové pole o průměru 2^n lze rozdělit na osm krychlových prostorových polí o průměru 2^{n-1} (tzv. oktanty). Číslování oktantů je zřejmé z obrázku.



Při použití oktantových stromů je prostorové pole děleno na oktanty; oktanty jsou pak stejným způsobem děleny rovněž na oktanty, a to tak dlouho, dokud nedosáhneme uspokojivě jemné reprezentace popisovaného objektu. Je zřejmé, že proces dělení končí na úrovni jednotkových krychlí, které již nelze dále dělit. Oktant má hodnotu FULL, resp. VOID, pokud všechny jednotkové krychle v oktantu mají hodnotu FULL, resp. VOID. Pokud oktant obsahuje krychle obou hodnot, má pak hodnotu MIXED. Prostorové pole je reprezentováno stromem, jehož kořen má stupeň 8. Uzly stromu jsou buď listy, nebo uzly stupně 8. Takový strom je nazýván oktantovým stromem. Kořen stromu odpovídá celému prostorovému poli. Uzel stupně 8 reprezentuje dělení prostorového pole o průměru 2^l na prostorová pole - oktanty - o průměru 2^{l-1}. Odpovídající uzly jsou číslovány od 0 do 7 v souladu s číslováním oktantů. Každému listu je přiřazeno ohodnocení odpovídající hodnotě příslušného oktantu (FULL nebo VOID).

Základní operací při práci s oktantovými stromy je vytvoření oktantového stromu. Uveďme si proto základní algoritmus pro tvorbu oktantového stromu zapsaný v jazyce C. Vzhledem k charakteru oktantového stromu je zřejmé, že se bude jednat o rekurzivní algoritmus (opakované dělení). Nejprve si však uveďme definici příslušné datové struktury:

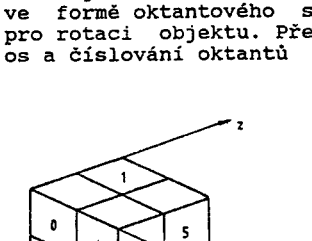
```
struct octreeglob
{
  float xmin, ymin, zmin;
  float xmax, ymax, zmax;
    /* definice vyšetřovaného prostoru */
}
struct octree
{
  char code;  /* FULL, VOID, MIXED (GREY) */
  struct octree *oct[8]; /* ukazatele na podřízené uzly - oktanty */
}
```

Důležitou roli hrají funkce classify a subdivide. Funkce classify testuje aktuální uzel (oktant) vzhledem k zadanému objektu. Funkce subdivide dělí aktuální uzel (oktant) na osm podřízených uzlů (oktantů).

```
maketree (p, t, depth)
  primitive *p;  /* p = modelovaný objekt */
  octree *t;  /*  t = uzel oktantového stromu */
  int depth;  /* maximální hloubka rekurze */
  {
    int i;
    switch (classify (p,t))
    {
      case VOID :
        t->code=VOID;
        break;
      case FULL :
        t->code=FULL;
        break;
      case MIXED :
        if (depth == 0)
        {
          t->code=FULL;
        }
        else
        {
          subdivide(t);
          for (i=0; i<8; i++)
            maketree (p, t->oct[i], depth-1);
        }
        break;
    }
  }
```

Je zřejmé, že oktantové stromy jsou rozšířením kvadrantových stromů do třírozměrného prostoru. To znamená, že i operace nad oktantovými stromy jsou rozšířením operací nad kvadrantovými stromy. V dalším si ve stručnosti uvedeme základní charakteristiky některých běžných operací nad oktantovými stromy.

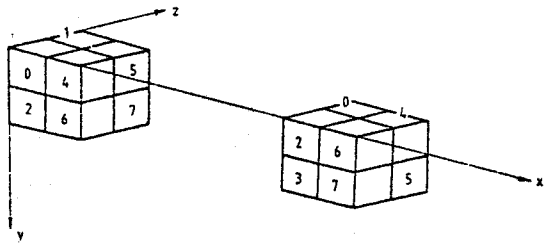
Operace sjednocení, průniku a doplňku pro objekty kódované pomocí oktantového stromu lze provádět pomocí rekurzivních algoritmů. Bylo dokázáno, že čas výpočtu roste lineárně s počtem uzlů ve stromu. Stejnou vlastnost má i operace kondenzace. Kondenzací rozumíme odstranění osmic listů (se stejným předchůdcem), majících stejné ohodnocení (FULL nebo VOID). Dále byla vyvinuta řada algoritmů umožňujících transformace reprezentace objektů ve formě oktantového stromu. Jako příklad lze uvést algoritmus pro rotaci objektu. Předpokládejme počátek souřadnic, orientaci os a číslování oktantů tak, jak je uvedeno na obrázku.



Máme-li nějaký objekt takto orientovaný, proveďme např. otočení o 90° kolem osy rotace rovnoběžné s osou x procházející středem prostorového pole. Stav prostorového pole před rotací a po rotaci je uveden na obrázku.

Neformálně lze algoritmus pro otočení popsat následovně:

```
begin
  if N is not a leaf then
    to its (numbered) offspring apply the permutation
    pi: (0,1), (1,3), (2,0), (3,2), (4,5), (5,7), (6,4),
    (7,6) and rotate each of the children recursively
end
```



Zjištění permutací pro otáčení kolem zbývajících dvou os je vhodným cvičením pro čtenáře.

Protože zmíněný algoritmus je založen na traverzování stromu, je zřejmé, že čas výpočtu je úměrný počtu uzlů ve stromu.

Změna měřítka objektu (měřítko mocniny dvou) je rovněž velmi jednoduché. Dvojnásobně zvětšení části objektu, jíž odpovídá uzel stromu první úrovně (následník kořene stromu) se provede tak, že tento uzel se prohlásí za kořen při zrušení ostatních uzlů stejné úrovně (následníků původního kořene).

Dvojnásobné zmenšení objektu se provede tak, že se vytvoří nový kořen stromu a stávající kořen se stane jeho následníkem. Je třeba ještě doplnit sedm prázdných následníků.

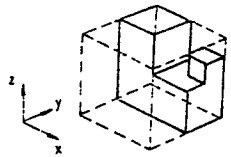
Byly vytvořeny i algoritmy pro posuv. Tyto algoritmy jsou modifikací algoritmů pro kvadrantové stromy.

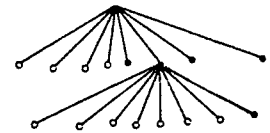
V souvislosti s použitím oktantových stromů je třeba se zabývat otázkou datových struktur, které tyto stromy budou reprezentovat. Abychom redukovali pokud možno co nejvíce paměťové

nároky, nebudeme ukládat oktantový strom podobně jako klasický strom, kde v každém uzlu je osm ukazatelů na následníky v další úrovni. Z hlediska paměťových nároků je výhodné ukládat informaci o oktantovém stromu v lineární formě. Velmi používanou formou je tzv. DF-výraz (DF-expression), založený na traverzování stromu typu preorder. Každý uzel typu MIXED je následován kódy svých osmi následníků. Poněvadž se v oktantovém stromu vyskytují tři druhy uzlů, postačí na zakódování informace o každém uzlu dva bity. Způsob kódování je patrný z obrázku.

1) MVVVVFMVVVVVVVFFF
2) M(VVVVFM(VVVVVVVF))FF

Ve druhém případu DF-výrazu byly do výrazu začleněny závorky z důvodu větší přehlednosti výrazu.



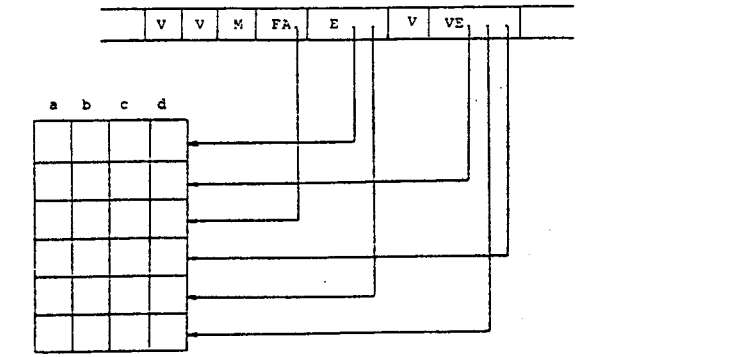


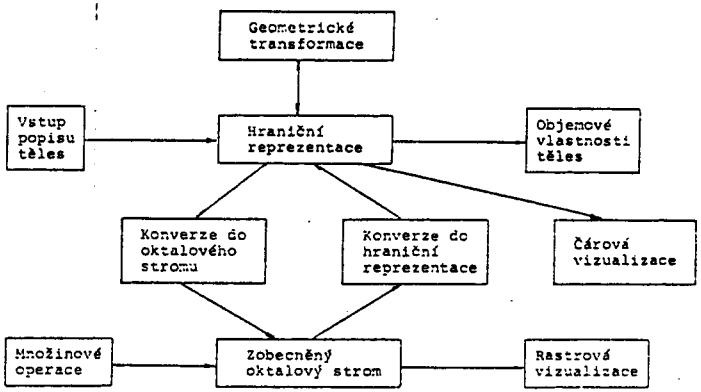
Paměť požadovaná pro reprezentaci objektu je závislá na velikosti povrchu objektu. Je zřejmé, že paměťové nároky mohou být poměrně velké v případě, kdy je požadována přesná aproximace povrchu tělesa.

Z tohoto důvodu se používají různé hybridní reprezentace ve snaze sloučit výhody agregovaných metod. Jednou z možných reprezentací je modifikovaný DF-výraz, kde kromě výše uvedených tří druhů uzlů (FULL, VOID a MIXED) jsou použity další tři typy uzlů: FACE (stěna), EDGE (hrana) a VERTEX (vrchol). Kromě DF-výrazu obsahuje datová struktura ještě informaci o rovinách, jejichž částmi jsou stěny tvořící povrch tělesa. Stěna tak může být určena jednou rovinou, hrana dvěma rovinami (průsečnice) a vrchol třemi rovinami (průsečík průsečnic). Tvar datové struktury je velmi zhruba naznačen na obrázku.

Zmíněná reprezentace je pak používána jako pomocná reprezentace spolu s hraniční reprezentací. Je zřejmé, že hlavním důvodem používání zobecněného oktantového stromu je úsporné provádění množinových operací. Hraniční reprezentace je uchovávána v paměti po celou dobu práce systému, zatímco zobecněný oktantový strom je generován pouze v případě potřeby

(provádění množinových operací). Systém obsahuje moduly pro konverzi obou reprezentací oběma směry.





Provádění množinových operací je založeno na jednoduchém simultánním traverzování obou stromů (reprezentujících tělesa vstupující do operace) typu preorder. Pro každou množinovou operaci je vytvořena tabulka rozměru 6 x 6 popisující typ výsledného uzlu. Například v případě průniku dvou těles může nastat situace, kdy aktuální uzel v oktantovém stromu A je prázdný (typ V), to znamená, že uzel ve výsledném oktantovém stromu bude rovněž prázdný. Proto příslušná řádka v tabulce bude obsahovat V. Je zřejmé, že v případě oktantového stromu B pak nepokračujeme v traverzování do hloubky stromu. V případě, že aktuální uzel ve stromu A je plný (typ F), je do výsledného

oktantového stromu kopírován podstrom příslušný aktuálnímu uzlu ve stromu B. V případě vyšetřování průniku uzlů komplexnějšího charakteru (FA, E, VE) se používají složitější algoritmy pro vyšetřování průniku dvou mnohostěnů. V případě příliš složitého výsledku je třeba tento pomocí speciálního dekompozičního algoritmu rozložit do tvaru vyjádřitelného zobecněným oktantovým stromem.

Pro ostatní množinové operace jsou rovněž k dispozici obdobné tabulky rozměru 6 x 6 spolu s příslušnými podpůrnými operacemi. Dá se dokázat, že množinové operace s pomocí uvedené reprezentace je možno provádět s lineární složitostí. Existuje možnost zobecnění uvedené reprezentace v tom smyslu, že místo rovinných stěn je možno používat kubických nebo bikvadratických plátů.

Závěr

Uvedené metody jsou hojně používány jak v oblasti počítačové grafiky, tak i v příbuzných oborech. Zvládnutí těchto metod je předpokladem pro tvorbu nejrůznějších informačních systémů pracujících s grafickou informací jak ve dvourozměrném tak i ve třírozměrném kontextu. Značnou pozornost je tak nutno věnovat i problematice ukládání grafické informace na vnějších médiích a jejím přenosu po sítích.