Reprezentace scény

KMI/3DG

Mgr. Markéta Trnečková, Ph.D.

Palacký University, Olomouc

Scéna

- **Scéna** = množina prostorových objektů doplněná o další informace potřebné pro jejich zobrazení
- Zatím víme, jak reprezentovat jednotlivé objekty
- Definujeme transformace objektů do cílové polohy, přidáme světlo a kameru
- Jak efektivně uspořádat?
- **Graf scény**

- **Scéna** obsahuje:
  - Nezobrazované objekty – kamera, osvětlení
  - Zobrazované objekty – definované geometrií, barevnými vlastnostmi, texturami
  - Prvky definující logickou strukturu scény – definice skupin a jejich instancí
  - Transformace – definované hierarchicky pro snadnější manipulaci s objekty

Scéna

- **Kamera**:
  - Představuje virtuálního pozorovatele, který si scénu prohlíží
  - Množina informací týkající se jedné pohledové transformace
  - Z umístění kamery získáme informaci o umístění pozorovatele, směru pohledu a způsobu promítání
  - Kamera se může pohybovat $\rightarrow$ trajektorie se zadává křivkou
- **Osvětlení**:
  - Údaje o světelných zdrojích
  - Buď abstraktní objekty určující pouze směr odkud přichází světlo
  - Nebo plochy, které světlo emitují – je nutno vymodelovat pro ně těleso a to umístit do scény

Scéna

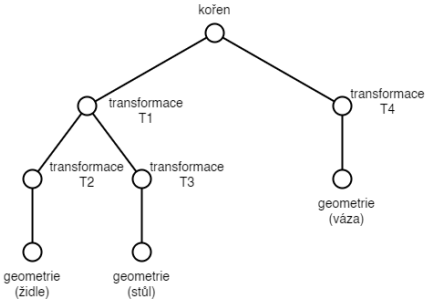
- **Uspořádání těles ve scéně** – do struktury, která umožňuje seskupovat logicky k sobě patřící části, efektivně je transformovat a jejich instance vkládat úsporným způsobem do prostoru scény
- Stůl = deska a 4 nohy
- Vhodná struktura – **graf scény**

Graf scény

- **Scene graph** – n-ární strom
- Každý uzel má právě 1 rodiče (kromě kořene)
- Graf scény může obsahovat několik stromů – les
- Definice grafu scény není jednoznačná – liší se v jednotlivých systémech
- Odlišnosti:
  - V pravidlech pro jejich stavbu
  - Typech uzlů
  - Interpretaci dat

Graf scény

- Strom vyjadřuje **dědičnost** uzlů
- V listech bývá geometrie, v nelistových uzlech např. transformace
- V jednom místě můžeme určit transformaci, která bude platná pro všechny potomky tohoto uzlu



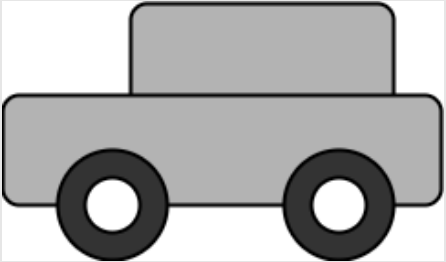
Graf scény

- **Transformace**:
  - Je vhodné definovat tělesa v lokálních souřadnicích (ne přímo v souřadnicích scény)
  - Rozmístění objektů ve scéně pomocí transformací (uložené ve stromu)
  - Díky tomu můžeme celý podstrom transformovat najednou
- **Skládání transformací**
  - Projeví se při interpretaci stromu
  - Před vykreslením scény se projde celý strom shora dolů a zleva doprava
  - Při sestupu transformace ukládáme na zásobník (při přesunu do vyšší úrovně transformace ze zásobníku odebíráme)
  - Jak dojdeme k uzlu uchovávající geometrické údaje aplikujeme na něj transformací složenou ze všech transformací mezi ním a kořenem (tedy transformace na zásobníku)
  - Na židli aplikujeme transformaci T1T2, na stůl T1T3 a na vázu T4
  - Jak bychom změnili pozici všech 3 objektů?
- Ukládání transformací do samostatných uzlů je výhodné zejména u animací – v každém kroku aktualizujeme hodnoty transformace v příslušných uzlech (např. otáčení kola)

Graf scény

Example

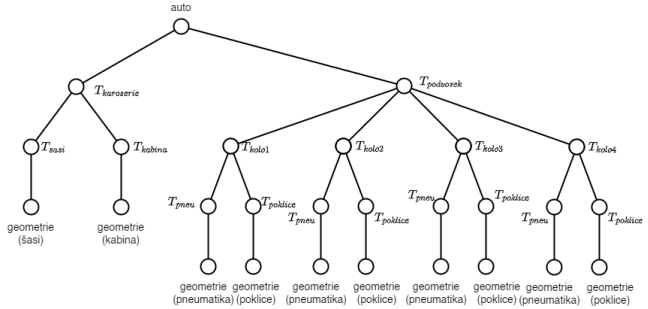
Vytvořte graf scény pro následující objekt.



Graf scény

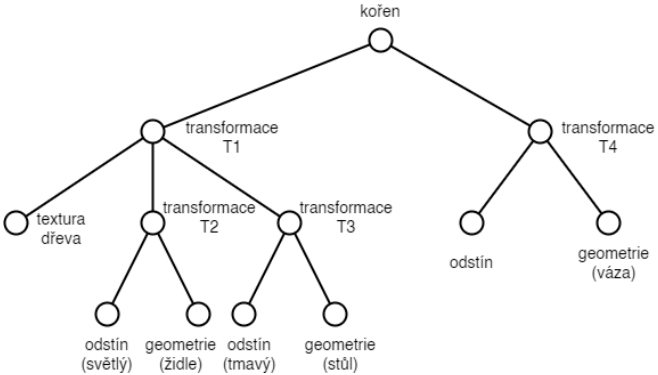
Example

Vytvořte graf scény pro následující objekt.



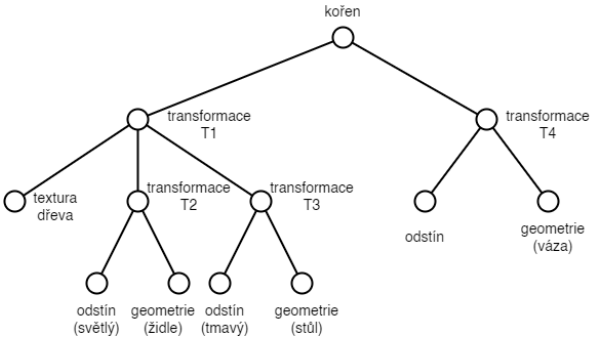
Graf scény

- V některých systémech můžeme využít i vztahu sourozenec
- Následník uzlu = pořadí uzlů zleva doprava
- Danou vlastnost mají všechny uzly vpravo od něj (na stejné úrovni, se stejným rodičem)
- Nespecifikované vlastnosti se dědí od rodiče



Graf scény

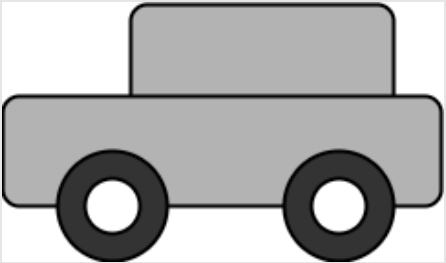
- Texturu dřeva dědí židle i stůl
- Odstín má židle i stůl různý
- Váza je v jiné části (větvi) stromu – mimo rozsah platnosti textury dřeva



Graf scény

Example

Do grafu scény pro následující objekt přidejte barvu jednotlivých geometrií.



Graf scény

- Obdobně, jako transformace, zpracováváme i vzhledové vlastnosti (textury, koeficienty pro odraz světla a jiné)
- Hodnoty se neskládají, ale je vždy aplikována poslední nalezená hodnota

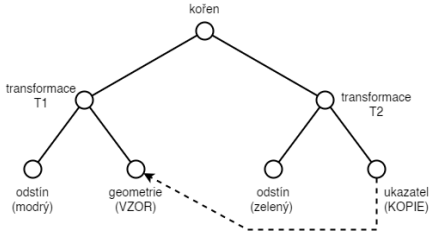
Graf scény

- **Osvětlení** – do grafu scény můžeme přidat i zdroje světla
- Tím, kam je ve stromu umístíme, můžeme ovlivnit rozsah působnosti světelných zdrojů
- Pokud víme, že nějaký zdroj ovlivní osvětlení jen části scény, přidáme ho do podstromu určující tuto část
- Např. světlo z chodby nepronikne do pokoje, umístíme zdroj do podstromu reprezentující chodbu.
- Tím urychlíme vyhodnocení osvětlovacího modelu

- **Kamera** – do grafu scény můžeme přidat i kameru
- Stromem pak můžeme definovat nejen její umístění, ale i případnou animaci (pomocí transformace)

Graf scény

- **Více stejných objektů** – pokud se ve scéně objevuje více instancí stejného objektu
- Můžeme objekt definovat jen jednou a další instance popsat úsporným způsobem
- Zavedeme uzel obsahující ukazatel na jinou část stromu
- Tím získáme strukturu – **zhroucený strom** (colapsed tree)

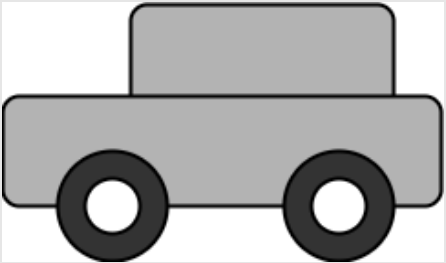


- Dva objekty mají stejný tvar (geometrii)
- První má modrou barvu a je vzorem pro druhý objekt, který má zelenou barvu
- Místo geometrie zeleného objektu je na jeho místě použit ukazatel
- Na vzor je aplikována transformace T1, na kopii pouze T2 – to plyne z postupu vyhodnocování
- Jak změnit graf scény, aby na kopii byla aplikována i T1?

Graf scény

Example

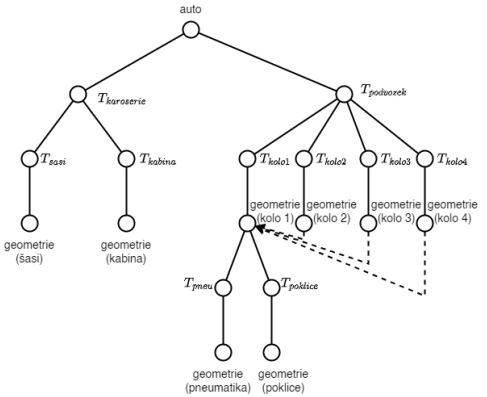
Pomocí ukazatelů zjednodušte graf scény následujícího objektu.



Graf scény

Example

Pomocí ukazatelů zjednodušte graf scény následujícího objektu.



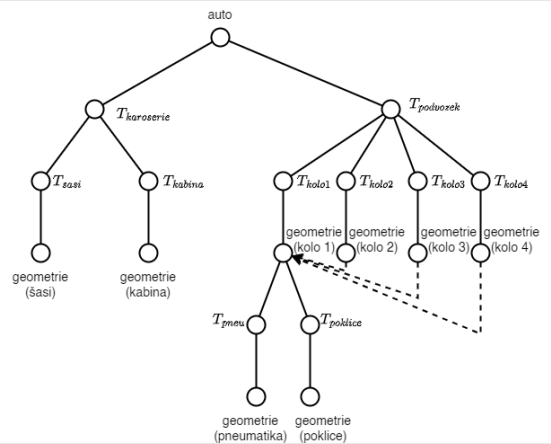
Graf scény

- Uzel, který je vzorem pro další kopie má více předchůdců
- Abychom předcházeli vzniku cyklů (ty by znemožňovaly vyhodnocení dříve popsaným způsobem) nesmí ukazatele ukazovat na předchůdce, nebo na uzel ve stromu vpravo

Graf scény

Example

Popište způsob vyhodnocení následujícího grafu scény.



Pomocné datové struktury

- Vhodné uspořádání informací ve scéně výrazně snižuje časové nároky při zpracování scény
- Zavádíme pomocné struktury, které nejsou součástí popisu scény, ale vznikají při jejím načtení
- Většinou mají hierarchický charakter – **Prostorová hierarchie**
- Po vytvoření datových struktur a jejich naplnění informacemi (odkazy) o objektech ve scéně, je používáme k vyhledávání informací o objektech ve scéně
- **Dělení**:
  - Hierarchie obálek (BVH, bounding volume hierarchies) – vznikají postupným shlukováním objektů a obálek, které je obklopují
  - Hierarchie dělení prostoru (space subdivision hierarchies) – vytvářejí se postupným rozdělováním prostoru
- U obou kategorií je výsledkem prostorová hierarchie – stromová struktura
- Kořen reprezentuje všechny objekty scény resp. prostor v němž se scéna nachází

Hierarchie obálek

- Objekt ohraničíme **obálkou** (bounding volume) – tělesem s velmi jednoduchou geometrií
- **Test polohy objektu ve scéně** (případně průsečíku s paprskem) lze realizovat rychleji s obálkou než s objektem se složitou geometrií, který obklopuje
- **Předběžný test** (hit/miss test) – pokud test selže (obálka leží mimo, paprsek jí neprotíná), pak není potřeba testovat vůči samotnému objektu
- Více typů obálek, nelze jednoznačně říci, která je nejlepší, záleží na řešené úloze

Hierarchie obálek

Koule

- Snadno se vytváří a aktualizuje
- Špatně se přizpůsobuje objektu → obklopuje velkou část prostoru kolem objektu
- Zbytečně často je předběžný test pozitivní
- Je invariantní vůči rotaci objektu

Osově zarovnaný kvádr (axis-aligned bounding box – AABB)

- Stěny kvádru jsou kolmé na souřadnicové osy
- Je to současně min-max obálka všech bodů objektu
- stejně jako koule se snadno vytváří a aktualizuje, ale nepřizpůsobuje se tvaru objektu
- Obklopuje zbytečně velkou část prostoru

Hierarchie obálek

Orientovaný kvádr (oriented bounding box – OBB)

- Obklopující kvádr je orientován tak, aby byl jeho objem co nejmenší

Orientovaný rovnoběžnostěn (k-discrete orientation polytop – k-DOP)

- Průnik několika pásů prostoru určuje tzv. polytop
- Protilehlé roviny obálky jsou rovnoběžné a jejich normály jsou definovány v diskrétních směrech
- 6-DOP – kvádr typu AABB
- 14-DOP – kvádr s odseknutými rohy
- 18-DOP – kvádr s odseknutými hranami
- 24-DOP – kvádr s odseknutými rohy a hranami

Hierarchie obálek

Konvexní obálka

- Obálka je tvořena konvexním obalem všech bodů objektu

- Složitější obálky se hůře aktualizují, ale lépe se přizpůsobují objektům

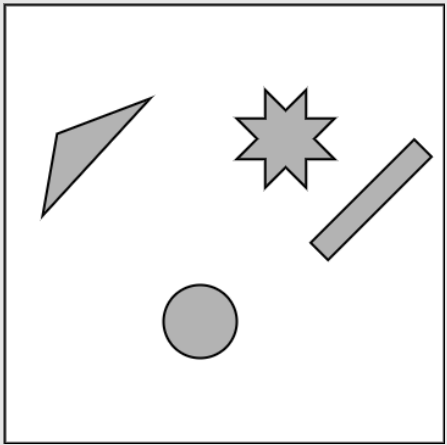
Hierarchie obálek

- Využití obálek pro předběžný test sice vede ke zrychlení výpočtů, ale ne výraznému
- Stále testujeme všechny objekty (jejich obálky) ve scéně
- Většího zrychlení dosáhneme, pokud obálky postupně sjednotíme do hierarchické struktury – **Hierarchie obálek**
- Obálky, které leží blízko sebe (nebo se překrývají), postupně shlukujeme do větších celků – objekty seskupíme a vytvoříme obálku obklopující tyto objekty
- Hierarchii sestavujeme zezdola nahoru

Hierarchie obálek

Example

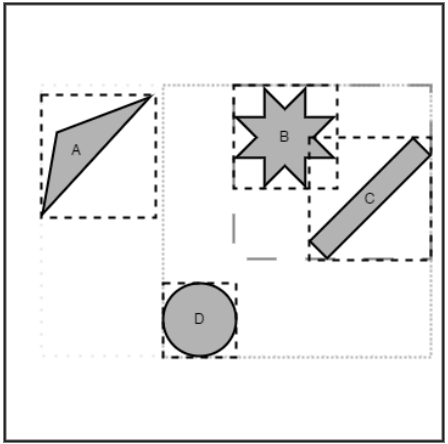
Vytvořte hierarchii obálek pro následující scénu (obálky mají tvar obdélníku).

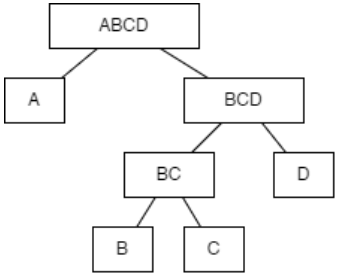


Hierarchie obálek

Example

Vytvořte hierarchii obálek pro následující scénu (obálky mají tvar obdélníku).



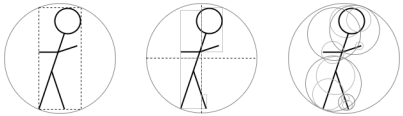


Hierarchie obálek

- Při zjišťování polohy bodu vzhledem k objektu ve scéně procházíme strom obálek od kořene dolů
- Testujeme polohu vůči obálce v uzlu
- Pokud je bod uvnitř, jsou rekurzivně dále zpracováváni potomci uzlu
- Pokud není bod uvnitř obálky, nemusíme testovat potomky
- Pokud je bod blízko A, otestujeme A a BCD, nemusíme testovat zvlášť B,C a D
- V případě vyváženého stromu – logaritmická složitost namísto lineární
- Efektivitu ovlivňuje schopnost obálky přizpůsobit se tvaru objektu

Hierarchie obálek

- Hierarchii můžeme budovat i směrem dolů
- V některých aplikacích nechceme jen obálky kolem objektů, ale kolem částí objektu – u objektů se složitou strukturou
- Příkladem je **hierarchie koulí**



- Těleso obklopíme obálkou AABB, pro kterou vytvoříme obklopující kouli
- Ta představuje nejhrubší náhradu tělesa – kořen hierarchie
- Dále rozdělíme obklopující kvádr v polovině na oktanty (3 řezy kolmé na osy)
- Pro každý neprázdný oktant zopakujeme předchozí krok
- Dělení končíme buď v prázdném oktantu nebo v okamžiku, kdy je v oktantu méně plošek než je předem stanovený počet (těleso je dostatečně jednoduché)
- Vzniká nepravidelný strom. Kolik může mít uzel potomků?

Hierarchie obálek

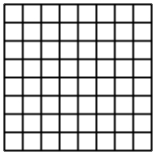
- Kolik může mít uzel potomků?
- 2-8
- Obdobně můžeme vytvořit hierarchii orientovaných kvádrů (OBB tree)

Dělení prostoru

- Opačný přístup – systematicky dělíme prostor scény (**scene partitioning**)
- Existuje celá řada způsobů, jak můžeme prostor dělit

Pravidelná mřížka (grid)

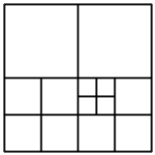
- Není hierarchií
- Pouze pravidelně dělí prostor rovinami kolmými na souřadné osy



Dělení prostoru

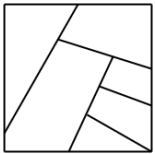
Oktalový strom (octree)

- Dělení 3 řezy kolmými na souřadné osy a umístěnými uprostřed (v polovině) prostoru
- Dělení vzniká vždy 8 podprostorů



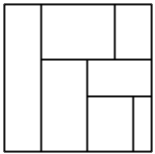
Strom BSP (Binary space partitioning)

- Binární strom s obecně umístěnými řezy



kD-strom (kD-tree)

- Speciální případ BSP stromu
- Řezné roviny jsou vždy kolmé na některou souřadnou osu a orientace se pravidelně střídají



Dělení prostoru

Algoritmus stavby stromu – H (hloubka stromu), B (právě zpracovávaná buňka)

1. Pokud je hloubka stromu H veliká, nebo B obsahuje malý počet těles, skonči
2. Rozděl B na části B_i a učiň z nich následníky uzlu B
3. Pro každou buňku B_i dělej:
   1. Do B_i zařaď odkazy na tělesa, která do ní alespoň částečně zasahují
   2. Postav strom pro B_i a $H + 1$
4. Zruš odkazy na všechna tělesa v B

Dělení prostoru

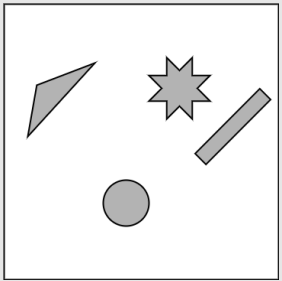
- Při dělení do uzlů zaznamenáváme informaci o dělení (poloha řezu a pod.)
- Listy obsahují odkazy na tělesa, která svým objemem zasahují do odpovídající části prostoru
- Pokud objekt zasahuje do více buněk, zvyšuje se počet odkazů na něj a tedy i paměťová náročnost pomocné struktury
- Při hierarchii obálek jsme se setkali s tím, že obálky zbytečně obklopují okolí objektů – **space redundancy**
- Při dělení prostoru se setkáváme s tím, že jsou objekty zaznamenávány opakovaně – **object redundancy**

Dělení prostoru

- Při pravidelném dělení vzniká mnoho prázdných buněk, tělesa jsou často protnuta řezy, což vede k opakovanému zaznamenávání tělesa
- Díky pravidelnosti se ale snadno sestavují a prohledávají

Example

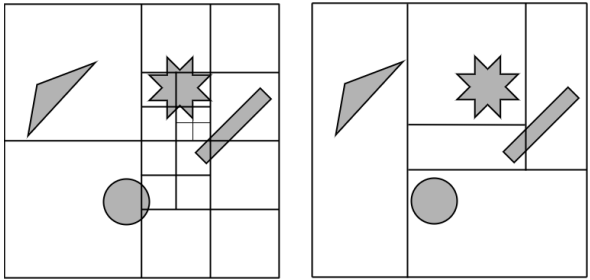
Rozdělte prostor pomocí oktalového (kvadratického) a (binárního) kD-stromu.



Dělení prostoru

Example

Rozdělte prostor pomocí oktalového (kvadratického) a (binárního) kD-stromu.



- Oktalový strom má 19 listů (6 prázdných a 13 neprázdných) – skoro třetina nenese informaci o objektech, tělesa jsou často protnuta
- kD-strom – lepší rozvržení, pouze 4 dělící roviny = 5 listů, žádný není prázdný a jen 1 těleso je protnuto

Dělení prostoru

- Úloha procházení stromu – například hledání nejbližšího tělesa, kterým prochází paprsek (ray tracing)
- Procházíme hierarchii tak, aby byly uzly zpracovávány ve správném pořadí
- Zde uvedený způsob je navržen pro binární stromy
- Začínáme v kořeni
- Pro výpočet využíváme zásobník
- Při sestupu stromem dolů, odkládáme na zásobník ty uzly, které by mohly být prozkoumány později (jsou na vzdálenější straně řezných rovin)
- Pokud paprsek mine všechna tělesa obsažená v listu, další kandidáti jsou na vrcholu zásobníku
- Ve vnitřních uzlech máme uloženy informace o řezné rovině a obálce typu AABB odpovídající buňce
- Rozhodnutí o zařazení uzlu na zásobník je určeno polohou průsečíku paprsku s řeznou rovinou

Dělení prostoru

Algoritmus procházení stromu – např. nalezení nejbližšího tělesa, protnutého paprskem

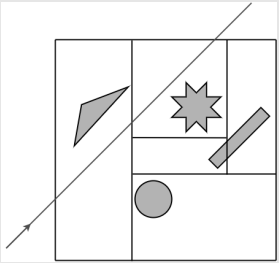
Dokud není zásobník prázdný (paprsek míří mimo scénu) nebo dokud není nalezen hledaný průsečík paprsku s tělesem dělej

1. Vezmi ze zásobníku uzel U
2. Je-li U listem dělej
   1. Pro všechna tělesa v listu zjisti, zda je paprsek protne a zapamatuj si nejbližší z protnutých těles
   2. Byl-li nalezen průsečík, skonči, jinak jdi na krok 1
3. Je-li U vnitřní, vypočítej průsečík P paprsku a řezné roviny příslušné tomuto uzlu
4. Urči průsečíky paprsku a obálky buňky U, průsečík bližší k počátku označ P_N a vzdálenější P_F
5. Potomka uzlu U reprezentující poloprostor určený řeznou rovinou a obsahující počátek paprsku označ jako bližší uzel, druhá jako vzdálenější uzel
6. Podle polohy bodu P vzhledem k P_N a P_F ulož na zásobník
   1. Bližší uzel, pokud P leží za P_F
   2. Vzdálenější uzel, pokud P leží před P_N
   3. Nejprve vzdálenější a pak bližší v ostatních případech

Dělení prostoru

Example

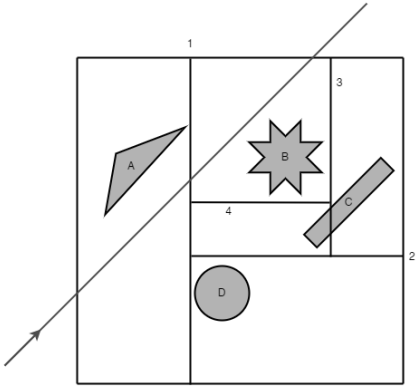
Pro předchozí příklad vytvořte kD-strom a simulujte procházení tohoto stromu pro zobrazený paprsek.

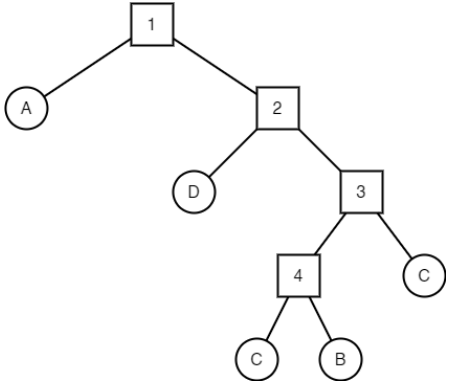


Dělení prostoru

Example

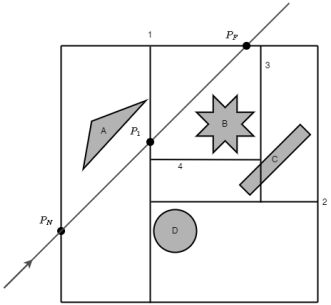
Pro předchozí příklad vytvořte kD-strom a simulujte procházení tohoto stromu pro zobrazený paprsek.

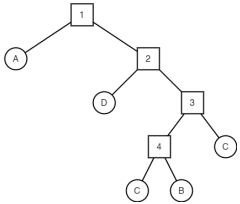




Dělení prostoru

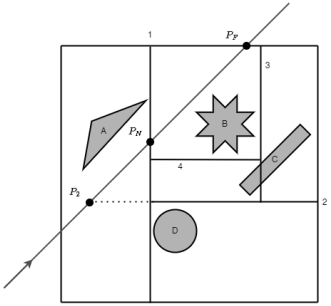
- Začneme celým prostorem
- Dle 6(3) jsou na zásobník uloženy oba potomci uzlu $\boxed{1}$
- P_1 leží mezi P_F a P_N
- V dalším kroku testujeme (A) – (A) je list, zjišťujeme, zda paprsek protne těleso

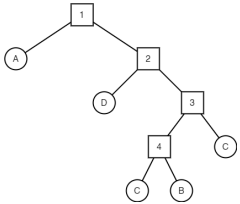




Dělení prostoru

- Zpracováváme $\boxed{2}$
- Z P_1 se stal bod P_N, P_F zůstává
- P_2 leží před P_N, dle 6(2) přidáme na zásobník vzdálenější uzel
- Prostor obsahující počátek paprsku je podprostor obsahující (D)
- Na zásobník přidáme $\boxed{3}$





Dělení prostoru

- Možná vylepšení – více uzlů obsahuje těleso C
- Jakmile ho jednou otestujeme, není potřeba ho testovat víckrát – pamatujeme si, která tělesa jsme testovali