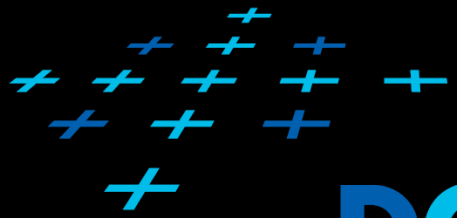


DCGI

KATEDRA POČÍTAČOVÉ GRAFIKY A INTERAKCE

Detekce kolizí

J. Bittner

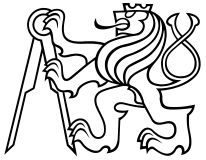
Detekce kolizí

- Nástroj pro
  - Fyzikální systém
  - Herní logiku
  - Uživatelské rozhraní

- Základní stavební kameny
  - Obalová tělesa
    - Testy průniku
  - Akcelerační struktury
    - Stavba
    - Traverzace / test průniku
    - Aktualizace

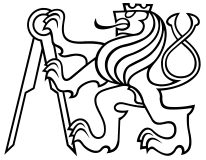
Herní “fyzika”

- Fyzikální systém
  - Dynamika tuhých těles
  - Pohyb a interakce v čase (působení sil)
  - Složitý a “chaotický” pohyb (není předpřipravený)
- Složitější simulace
  - Např. fluidní mechanika, deformovatelná tělesa
  - Může být předzpracována a využita jako animace
- Různé požadavky pro různé herní žánry
  - Fyzikální simulace pomáhá i škodí
  - Hra s příběhem – volná fyz. simulace může být komplikací
  - Fyzika – větší nároky na vývoj, přípravu/výměnu dat, HW



Middleware pro kolize a fyziku

- Kolize i fyzika
  - Havok
    - zlatý standard, stdandardní binary release zadarmo, plná verze drahá
  - PhysX
    - Původně Novodex, SDK zadarmo, zdrojový kód a podpora za poplatek
    - Integrováno do Unity
  - ODE, Bullet
    - zadarmo + zdrojový kód
- Pouze kolize
  - I-Collide, V-Collide (UNC)
- Sledování paprsku
  - OptiX, iRay, Embree

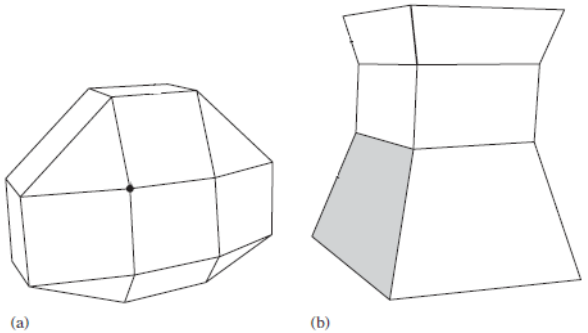


Problém detekce kolizí

- Najít geometrické průsečíky
- Fyzikální simulace typicky nejnáročnější klient detekce kolizí
- Kolize není nutné popsat přesně
  - Ano-ne
  - Hloubka průniku, normála ve styčném bodě…
- Vstup: *n* 3D objektů
  - Geometrie
  - Geometrie + Transformace
- Úkol: Chceme najít průsečíky
- Test každý s každým: složitost $O(n^2)$

Obtížnost

- Typická situace
  - Mnoho testů, ale pouze několik málo kolizí!
  - Většina objektů není v kolizi

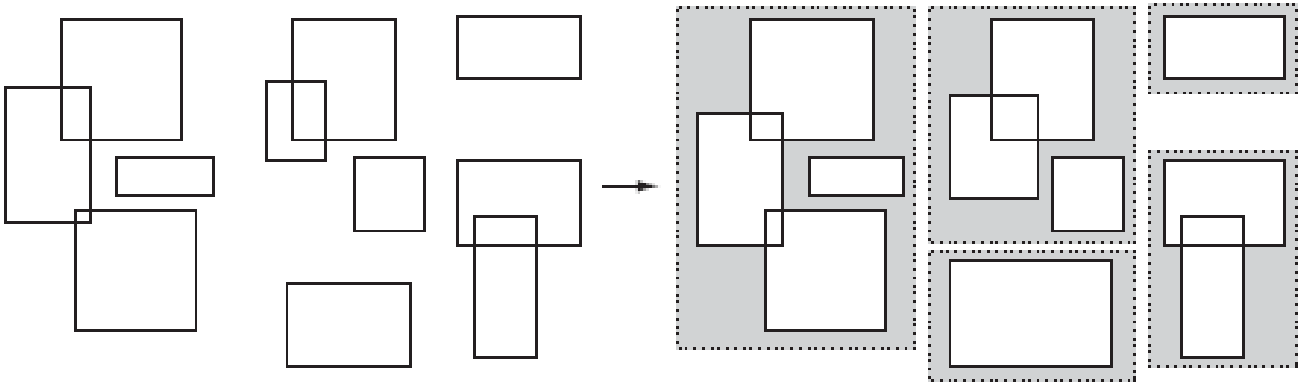


- Statická vs. dynamická detekce
  - Statická: statický “snímek” scény
  - Dynamická: uvažuje se pohyb i během výpočtu kolizí

- Zjednodušení výpočtů
  - dělení na statickou a dynamickou část scény
  - různé reprezentace objektů scény (herní, kolizní, vizuální)
  - pokud možno nepoužívat “meshe” pro kolize
  - konvexní objekty, kolekce konvexních objektů

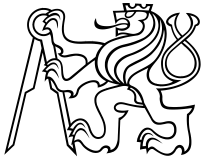
Fáze detekce kolizí

- Široká (broad phase)
  - konzervativní přístup
  - najde skupiny těles které spolu mohou kolidovat
  - rychle oddělí ty, které určitě nekolidují
- Úzká (narrow phase)
  - přesný výpočet pro dvojice objektů



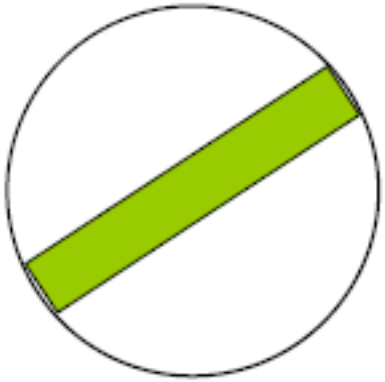
Obalová tělesa (Bounding volumes)

- Aproximace povrchu pomocí jednoduchých těles
- Požadavky
  - rychlý test průniku
  - dobrá aproximace = těsné obálky tj. minimum false positives
  - většinou kompromis
- Příklad: obalíme objekty A, B koulemi S_A a S_B
  - průsečík koulí velice rychlý
  - pokud S_A neprotíná S_B pak nekolidují ani A a B
  - v opačném případě A a B mohou (ale nemusí) kolidovat



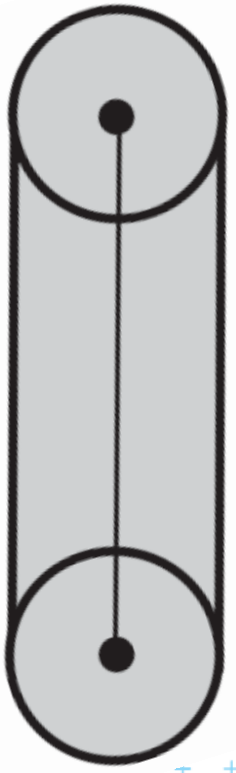
Koule

- Dána středem a poloměrem
- Velice rychlý výpočet průsečíku
- Většinou ne příliš těsná
- Snadno se aktualizuje
  - Invariantní vzhledem k rotaci



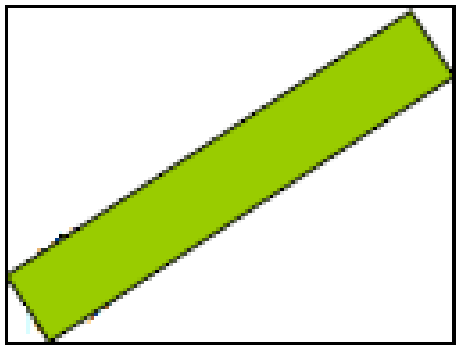
Válec nebo tažená koule

- Válec
  - aproximace postavy
  - na výpočet kolize spíše nepraktický
- Tažená koule (Capsule)
  - dobře aproximuje válec
  - rychlejší na výpočet



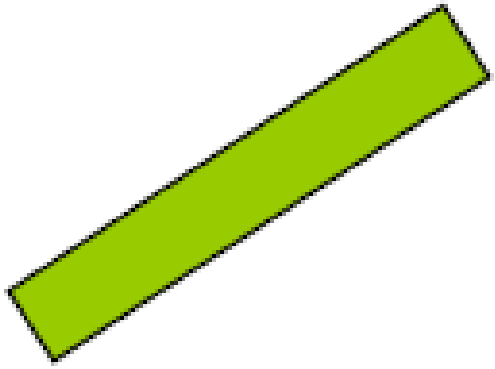
AABB

- Axis Aligned Bounding Box
  - osově orientovaný kvádr
- Obvykle o něco těsnější než koule
- Jednoduchý výpočet minimální AABB



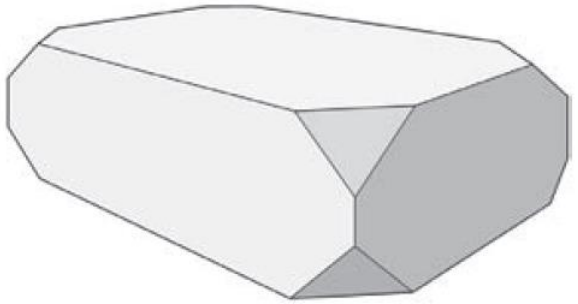
OBB

- Oriented Bounding Box
  - obecně natočený kvádr
- Dobré aproximační vlastnosti
- Netrivialní výpočet (Separation Axis Theorem – 15 os)



k-DOP

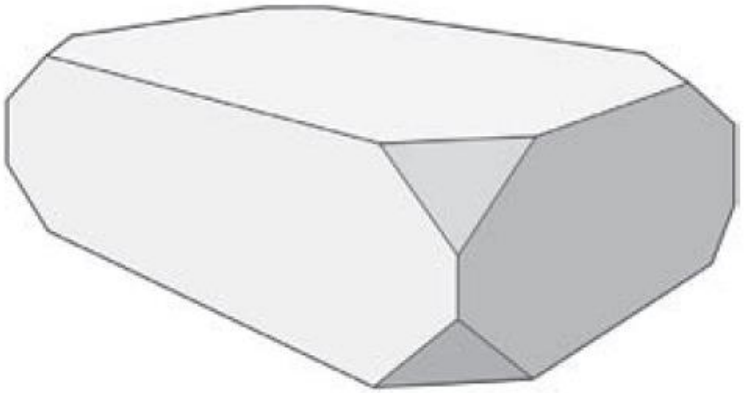
- Discreet oriented polytope
  - orientovaný mnohostěn
- Zadán množinou k/2 jednotkových vektorů $d_1,...,d_{k/2}$
- Normály stěn pochází z množiny $d_1,...,d_{k/2},-d_1,...-d_{k/2}$
- k=6: AABB
- k=14: zkosené vrcholy
- k=18: zkosené hrany
- k=26: zkosené vrcholy i hrany



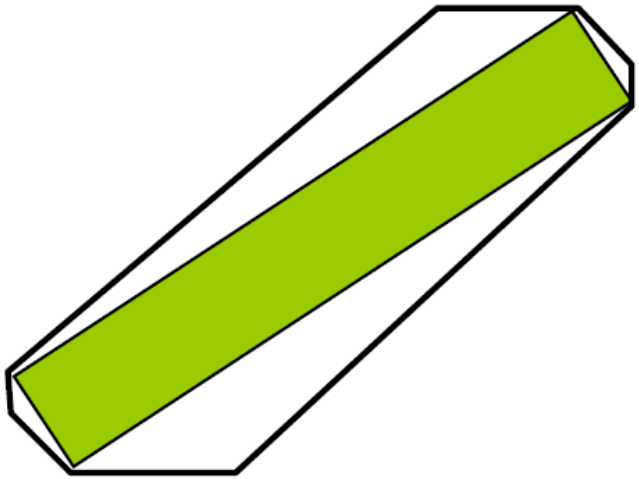
14-DOP

Příklad k-DOPu

14-DOP ve 3D

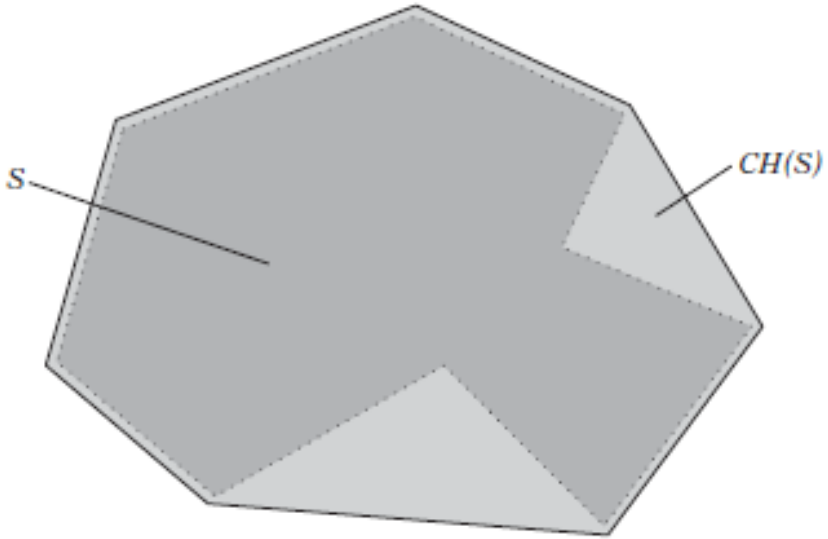


8-DOP ve 2D

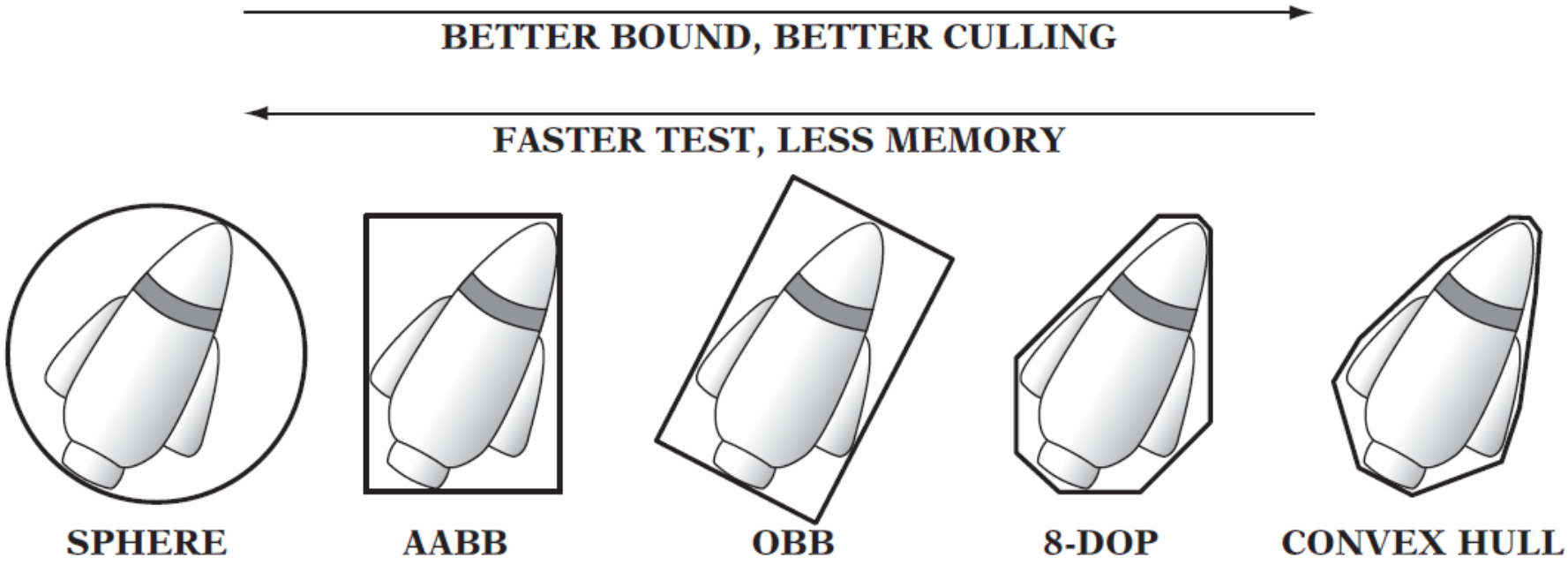


Konvexní obálka

- Convex hull
- Nejmenší možné konvexní obalové těleso
- Test průniku: pomalý
- k-DOP: aproximace konvexní obálky
  - s větším k se blíží KO



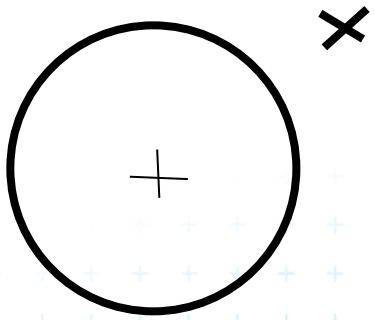
Srovnání obalových těles



Příklad 1: Koule vs Bod

- Point (Px,Py,Pz)
- Sphere center (Sx,Sy,Sz) and radius R
- Point inside a sphere:

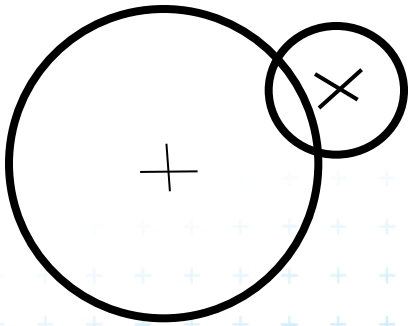
  $(Px-Sx)^2 + (Py-Sy)^2 + (Pz-Sz)^2 < R^2$



Příklad 2: Koule vs Koule

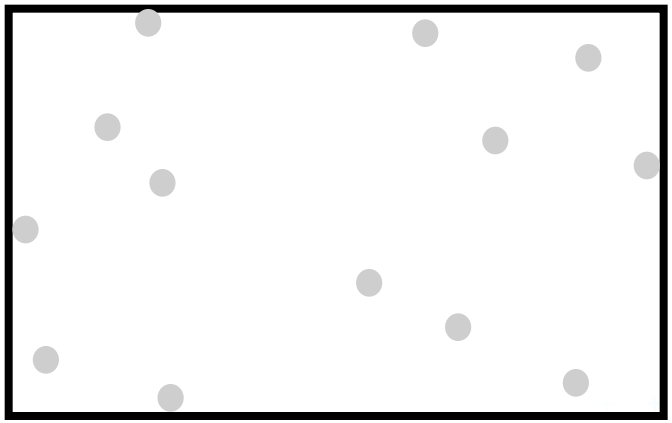
- Sphere center (Px,Py,Pz) and radius R2
- Sphere center (Sx,Sy,Sz) and radius R1
- Point inside a sphere:

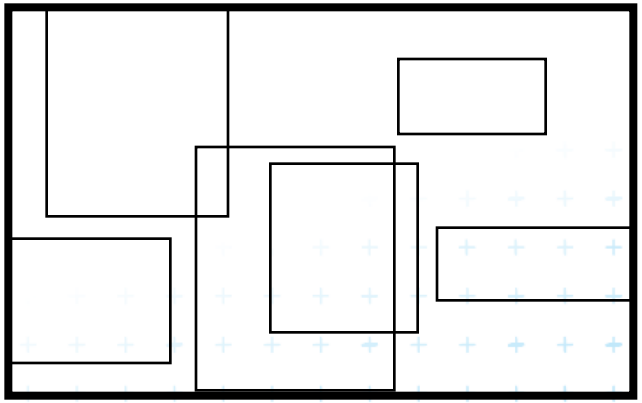
  $(Px-Sx)^2 + (Py-Sy)^2 + (Pz-Sz)^2 < (R1+R2)^2$



Příklad 3: AABB pro množinu bodů

- Pro souřadnice x, y, z
  AABB min(x) = min(x) pro všechny body
- Stejně pro AABB nad množinou AABB
- Test průniku – test průniku 1D intervalů



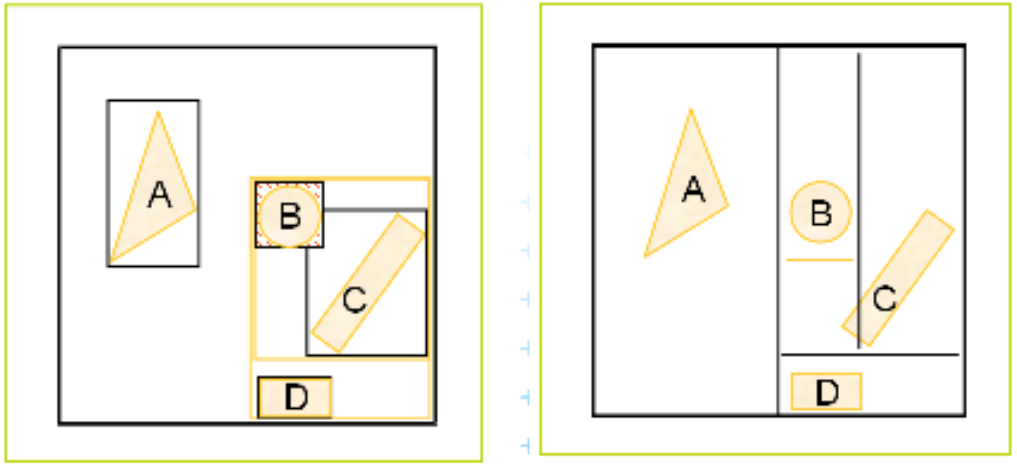


Akcelerační struktury

- Vyhledávání v logaritmickém čase
- Nutné nejdříve postavit a pak traverzovat
- Použitelné pro narrow i broad phase

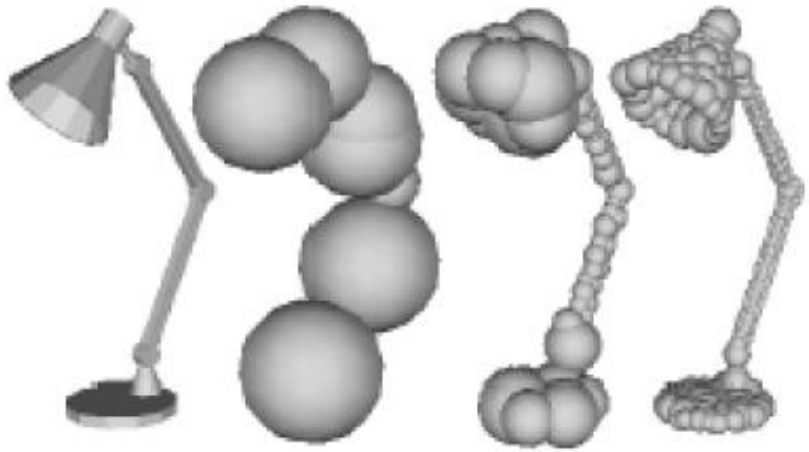
- Dělící objekty
  - Hierarchie obálek (Bounding volumes hierarchy)

- Dělící prostor
  - Mřížka
  - Octree
  - kd-tree
  - BSP



Časově kritická detekce kolizí

- Raději nepřesné než opožděné výsledky
  - výpočetní čas detekce kolizí často kolísá
  - záleží na geometrické složitosti průniku
- Časově kritické algoritmy: výpočet lze přerušit a algoritmus vydá co nejpřesnější odpověď
- Nejrychlejší: hierarchie koulí



Časově kritická detekce kolizí

- Nejdříve detekovat kolize kořenových koulí
  - dvojice uložit do seznamu
  - vrátit řízení aplikaci
  - pokud je čas, tak zpřesnit
  - využití prioritní fronty

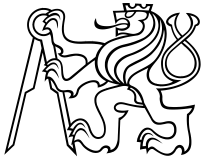
- Skončí buď prerušením, nebo dosažením poslední úrovně

Hierarchie obálek

- Stavba: zdola nahoru a shora dolů

- Zdola nahoru (slučování)
  - vytvořit obalová tělesa (a odpovídající listy stromu) pro všechny trojúhelníky
  - zvolit sousední uzel (nebo několik uzlů, podle řádu stromu) a vytvořit společného rodiče
  - opakovat

- Jaké uzly slučovat?



Hierarchie obálek

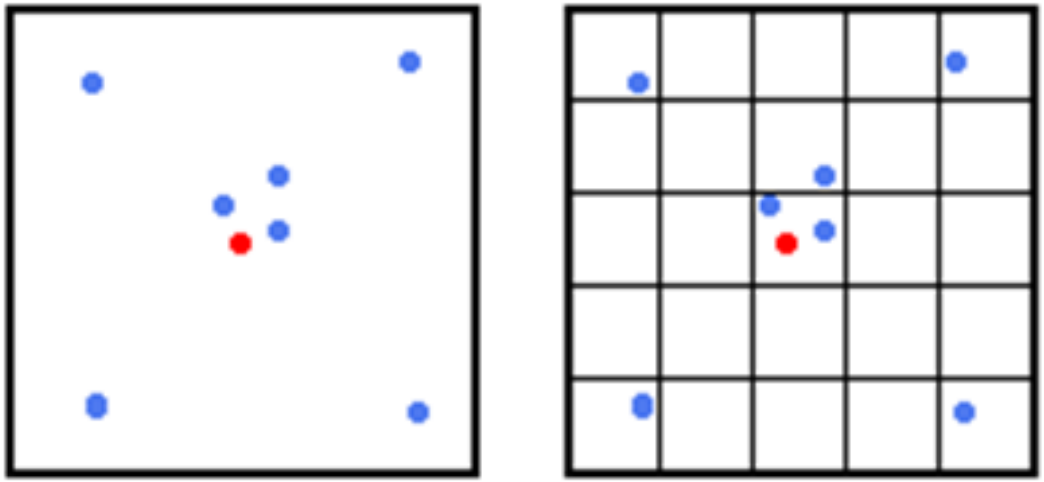
- Stavba shora dolů
  - vytvoř kořen a obalové těleso obsahující všechny trojúhelníky
  - rozděl kořen na nové uzly (děti) podle řádu stromu
  - opakuj pro každý nový uzel

- Jak dělit uzel?

- Algoritmy pro dělení nebo slučování uzlů volíme podle použití
  - v kolizních systémech se často používa i např. střílení paprsků

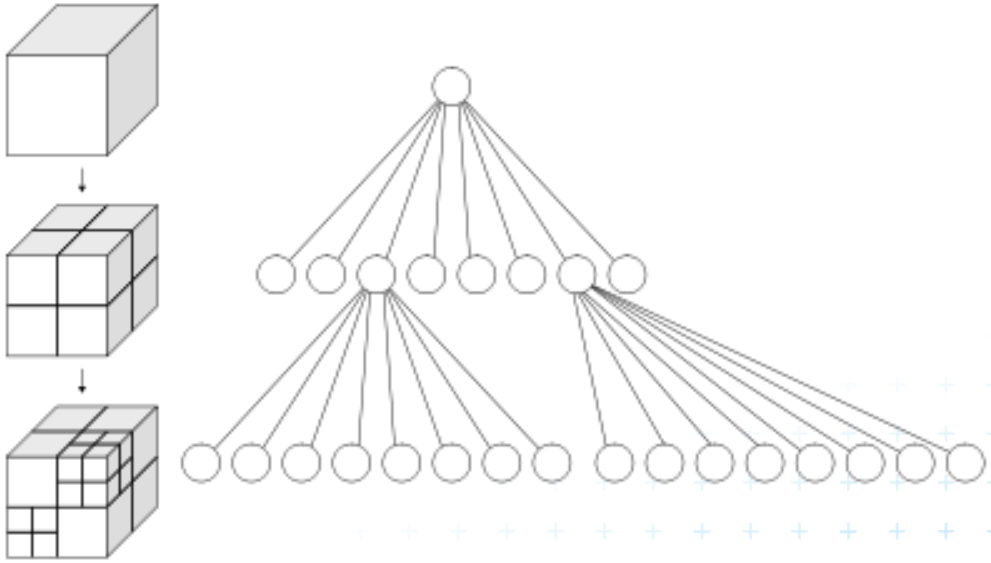
Mřížka

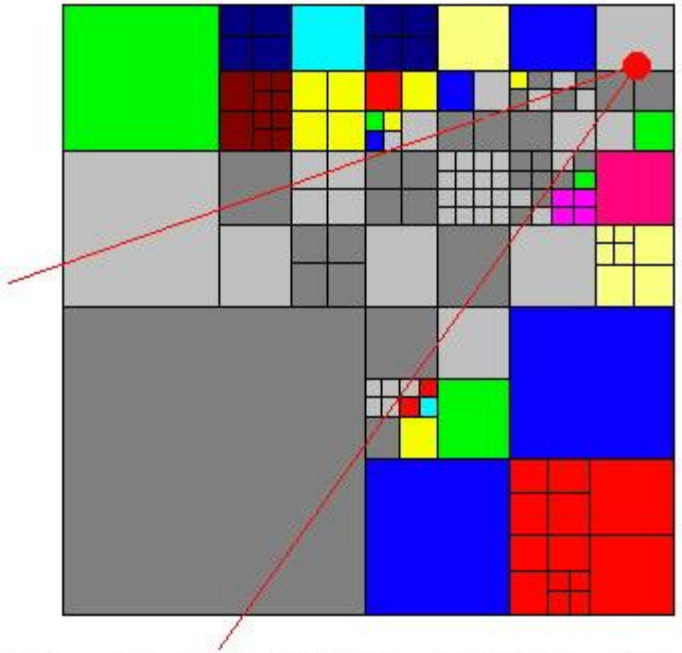
- Velice rychlá
- V konstantním čase zjistíme ve které jsme buňce



Octree

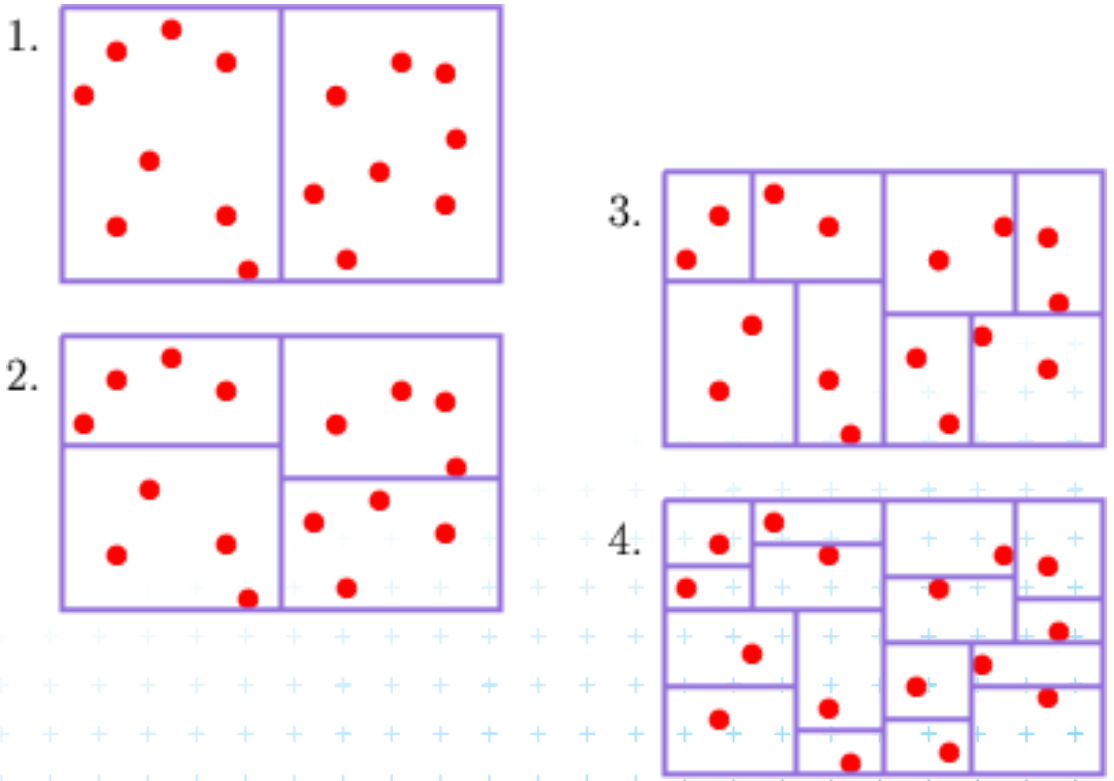
- Quadtree ve 2D
- Dělení uzlu na 2^D potomků vždy v polovině





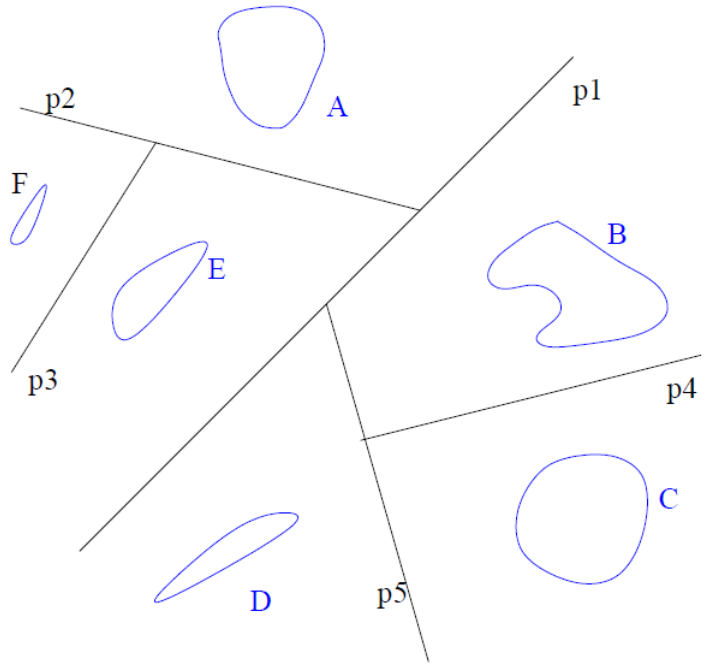
kd-tree

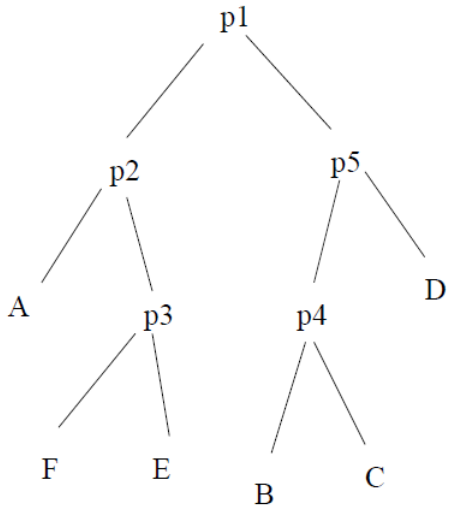
- Dělení uzlu ve vybrané ose v určeném bodě na dva potomky



BSP

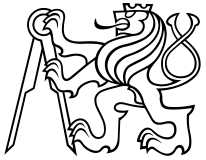
- Binary space partitioning
- Generalizace kd stromu





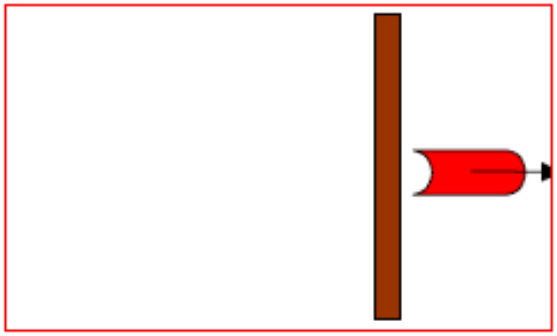
Deformace objektů

- Možnost změny objektu v čase
  - např. animované postavy
- Přepočítavat akcelerační strukturu je příliš pomalé
- Řešení např.:
  - Zachovat původní strom, ale aktualizovat obalová tělesa
    - BVH refit
  - Přepočítat jen některé části stromu
  - Pro rigidní části objektu použít předpřipravenou hierarchii



Dynamická detekce kolizí

- Statická detekce pro rychle se pohybující se objekty nestačí



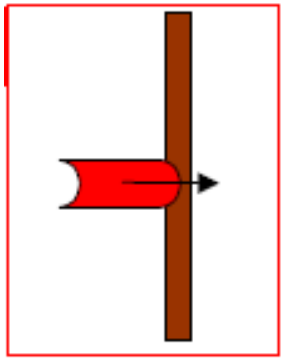
bullet through paper problem

- Problém: nalézt okamžik kontaktu
  - pro korektní reakci na kolize je nutné najít první okamžik
- Pokud uvažujeme pouze posun
  - vytvoříme konvexní obal tělesa v předchozím a aktuálním snímku
  - rozdělíme interval pokud se tělesa protínají
  - binární hledání se zvolenou přesností

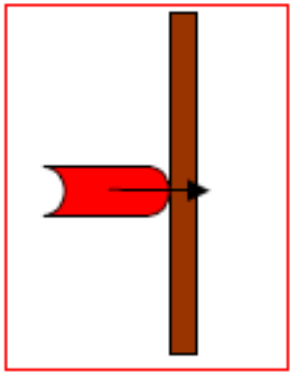
Pseudo-dynamická detekce kolizí

- Statická detekce po kratkých intervalech mezi předchozím a aktuálním snímkem

- Dvě fáze
  - vrátí libovolný čas T během první kolize
  - binární hledání v čase 0 až T pro přesný čas kontaktu



Hledání kolize



Přesný čas kontaktu

Pseudo-dynamická detekce kolizí

- Problém: určit vhodný krok Δt
  - nutné najít vhodný kompromis: malé Δt je pomalé, velké Δt je nepřesné
- Řešení: definovat maximální chybu (penetraci) ε
- Odvodit Δt pomocí horního odhadu rychlosti v libovolném bodu povrchu tělesa

Simulátor Hannover Messe 2000

(Skutečně) dynamická detekce kolizí

- Neopíra se o statickou detekci
  - nepoužíva diskretizaci, ale přímo počítá okamžik prvního doteku
- Pohyb mezi snímky neznáme, používá se model
  - nejčastěji šroubový pohyb
- Algoritmy založené na intervalové aritmetice
  - základní operace
    - [a, b] + [c, d] = [a + c, b + d]
    - [a, b] • [c, d] = [min(a•c, a•d, b•c, b•d), max(a•c, a•d, b•c, b•d)]
  - jsme si jisti, že řešení leží v intervalu
- Výzkumné téma!

Kolizní dotazy

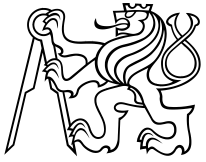
- Kolize dynamických objektů
  - Detekce všech kolizí - napojení na fyzikální engine
  - Odpověď na kolizi

- Vrhání paprsků (Ray cast)
  - Nalezení nejbližšího průsečíku s paprskem
  - Střelba, AI, detekce polohy, kola vozidla, …

- Vrhání tvarů (Shape cast)
  - Nalezení nejbližších průsečíků s koulí, kapsulí, kvádrem,…
  - Pohyb postav, plánování cest, polohování kamery, …

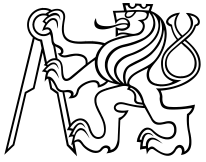
Virtuální objekty

- Fantom (Phantom)
- “Zero distance shape cast”
- Slouží k detekci kolize, ale neovlivňuje scénu
- Příklady
  - Fantom pro kameru
  - Aktivace události (trigger) při vstupu do místnosti



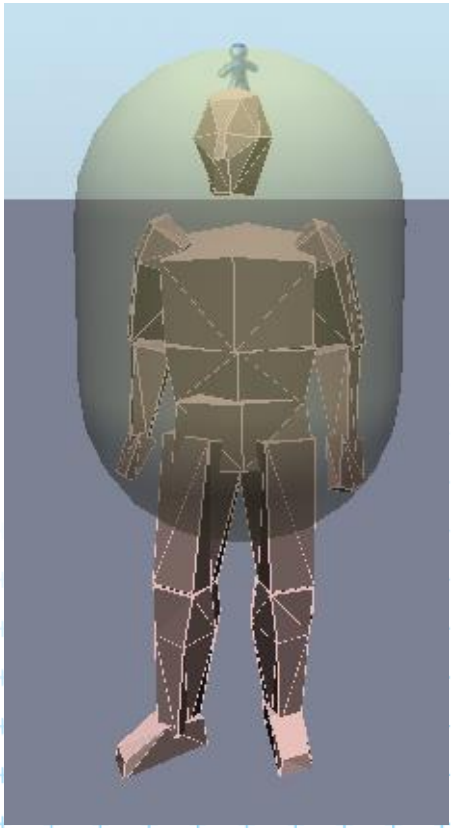
Filtrování kolizí

- Povolení / zakázání některých kolizí
- Kolizní vrstvy
  - Maska vrstev pro každý objekt (Havok)
- Callbacks
  - AddContactPoint – herní logika může kontakt zrušit
- Kolizní materiály
  - Materiál (textura) nese informaci o fyz./kolizních vlastnostech
  - Včetně např. informace o zvuku kolize



Ladění

- Cyklické pole vstupů kolizního systému
  - Při zastavení je možné prohlédnout několik vteřin zpět
- **Vizualizace kolizní geometrie**
- Referenční algoritmus
  - Ideálně co nejjednodušší
- Unit testy
  - Okrajové podmínky



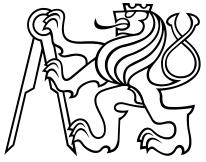
Fyzikální simulace

- Vypočítat jak síly ovlivňují tělesa
  - Reakce na kolize
- Fyzikální objekty úzce svázány s kolizními
  - např. ve PhysX totožné objekty
- Newtonovská mechanika

$$F(t) = m\,a(t) = m\,v'(t) = m\,r''(t)$$

$$r''(t) = \frac{1}{m} F(t)$$

- Řešení soustav diferenciálních rovnic
  - Separace lineárního a rotačního pohybu



Explicitní Eulerova integrace

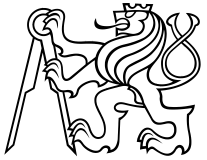
$$\mathrm{v}(t) = r'(t)$$

předpoklad … v(t) je konst. během snímku

$$r(t + \Delta t) = r(t) + v(t)\,\Delta t$$

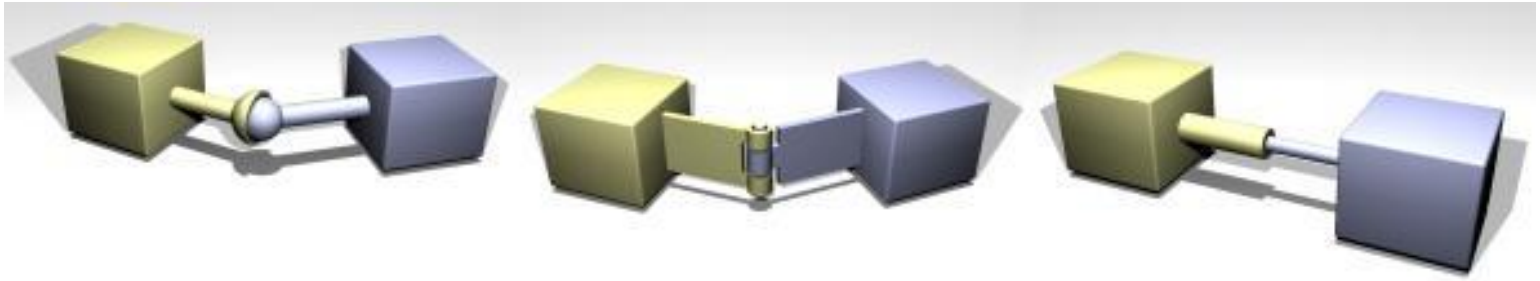
$$v(t + \Delta t) = v(t) + \frac{F(t)}{m}\Delta t$$

- Přesnější metody: Runge Kutta, Verlet, **Velocity-verlet**, …



Fyzikální simulace

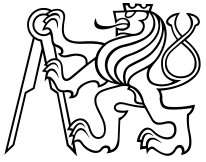
- Řešení soustav diferenciálních rovnic
- Omezující podmínky = vazby mezi objekty
  - pevné kontakty, klouby, pružiny, panty



- Minimalizace chyby soustavy
- Volit konstantní krok simulace! Δt

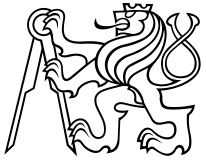
Spolupráce kolizního a fyz. systému

- Aktualizuj herní objekty (GUI inputs, herní logika)
- Aktualizuj síly, aplikuj silové impulzy, uprav vazby
- Simulační krok
  - Numerická integrace
  - Detekce kolizí
  - Vyřeš kolize
    - penalizační síly-impulzy
    - opakuj simulační krok
- Aktualizuj herní objekty
- Kolizní dotazy (shape casts, fantomy)
- Přejdi na vykreslování



Reference

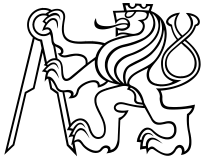
- C. Ericson, Real-Time Collision Detection
- J. Gregory, Game Engine Architecture
- D. H. Eberly, Game Physics



Numerická robustnost výpočtů

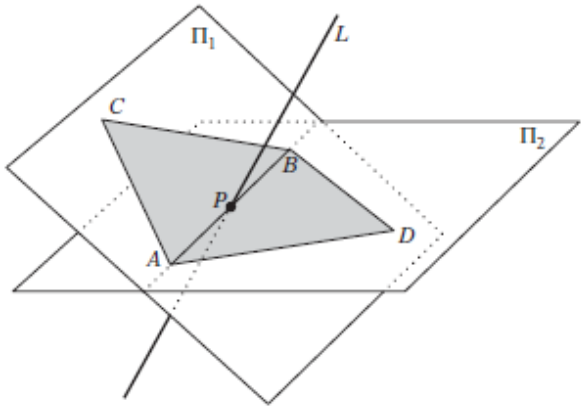
- Single-precision (float) je přesný do 6-9 desetinného místa
  - 1500 + 4.5e-9 = 1500
- Double-precision (double) je přesný do 15-17 místa
- Některá čísla nemají přesnou reprezentaci
  - 10 / 3 v desítkové soustavě
  - 0.1 ve dvojkové, po zaokrouhlení v single precision je větší než 0.1
  - nahrazení x * 0.1f za x / 10.0f není ekvivalentní !

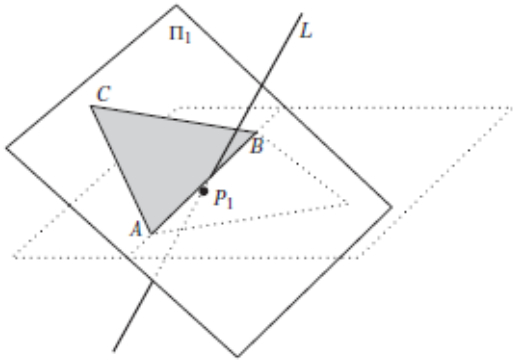
DCGI



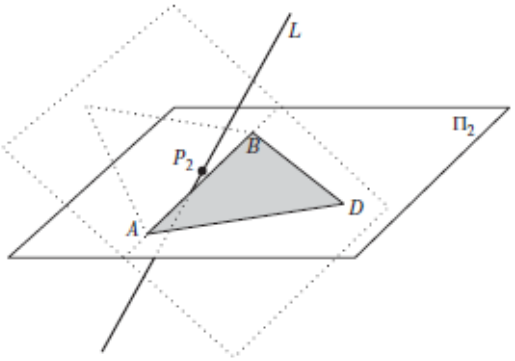
Numerická robustnost - řešení

- Epsilon tolerance
  - a == 0.f → abs(a) <= 1e-6
  - „Tlusté „ objekty
    - plochy
    - body, čáry a úsečky = koule, válce a kapsule
- Intervalová aritmetika
- Arbitrary precision arithmetic (GMP)
- Sdílení výpočtů
  - vyhnout se počítaní stejné hodnoty jinou rovnicí!





(a)

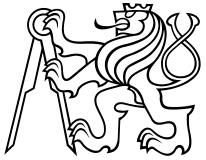


(b)

Geometrická robustnost

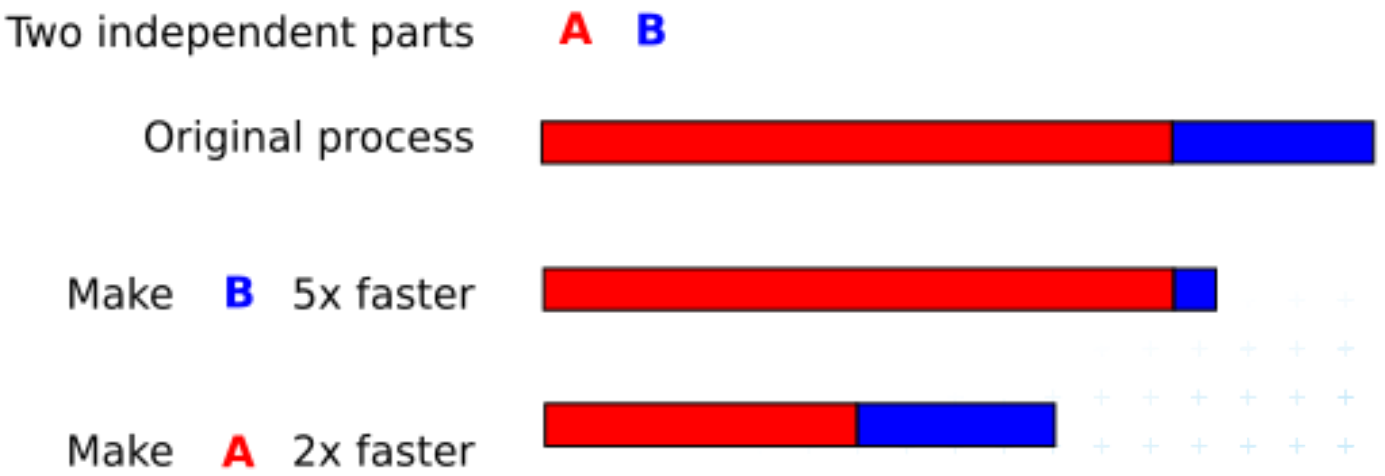
- Vstupní data se nemusí „chovat slušně“
  - Obsahují body, hrany a stěny navíc
  - Degenerované stěny do hrany nebo bodu
  - Nechtěně díry v modelu
  - Neplanární stěny
  - Nepodporovaná primitiva
- Řešení
  - Spojování bodů
  - Triangulace
  - Vyloučení degenerovaných dat
  - Randomizace
  - …

DCGI



Optimalizace

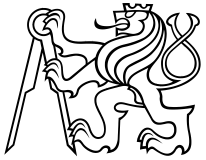
- Optimalizace úzkého hrdla



Optimalizace

- Využití cache
  - Omezení čtení/zápisů z/do paměti
  - Zarovnání na velikost cache line
  - Volné místo v ukazateli na zarovnanou adresu
  - Volné místo ve floating-point mantise
- SIMD
  - Operace nad 128B floating-point
  - Intrinsics na CPU
  - SSE, SSE2, SSE3, AVX, AVX2, AVX512, FMA, NEON, …
- …

DCGI



GPU akcelerovaná fyzika

- Pro broad phase složitější, méně nezávislých výpočtů
- Pro narrow phase ideální !

- Fragment shadery
  - data v texturách
- General purpose GPU
  - CUDA, OpenCL
- Čtení dat na CPU
  - HW podpora (occlusion queries, texture out)