Simulace mechaniky tuhých těles v reálném čase

L. Pohanka
Fakulta jaderná a fyzikálně inženýrská ČVUT, Břehová 7, 115 19
Praha 1
lukas.pohanka@inina.net

Abstrakt

Práce si bere za cíl předvést jednoduché možnosti konstrukce simulace tuhého tělesa v reálném čase a její 2D vizualizace v počítačovém rozhraní. Tyto pokročilé fyzikálně-vizuální modely jsou běžně používány ve filmových efektech nebo komerčních 3D aplikacích a hrách. Model se zakládá na aplikaci klasické mechaniky a matematických metodách detekce kolizí. Součástí práce je také aplikace demonstrující autorův vlastní jednoduchý 2D model klasické newtonovské mechaniky.

1 Úvod

S rostoucím výkonem dnešních počítačů roste i poptávka po reálnějších „napodobeninách“ či simulacích reality. Počítače už ve chvíli svého vzniku byly předurčeny k tomu, aby nám pomohly předpovídat chování nebo simulovat věci, které nelze řešit analyticky nebo jsou příliš obtížné pro výpočet. Fyzika nabízí spoustu problémů, které nelze řešit jinak než numericky, tj. nemají přesné či dokonce žádné analytické řešení – např.: úloha tří a více těles. Problematikou numerických řešení se zabývá mnoho oborů. Časem se zjistilo, že právě výsledky zkoumání této problematiky lze poměrně s úspěchem využít v místech, kde se snažíme nějakým způsobem přiblížit realitě (např. při použití filmových triků, ve 3D hrách).

Je jasné, že toho půjde nejlépe docílit, když se budeme snažit naši každodenní fyziku, kterou tolik známe ze života přenést i do této virtuální reality. Znamená to tedy vytvořit jakýsi malý model světa zakládající se na fyzikálních zákonech – jedná se o skloubení dvou věcí: numerické simulace fyzikálního systému (ta nám popíše jak systém vypadá číselně v kterémkoliv stavu) a následné použití těchto výsledků při konstrukci virtuální reality - tj. vizualizace.

Takový komplexní vizuální model fyziky lze pak s úspěchem využít a šetřit tak například práci mnoha animátorů. Ti by se snažili animovat předměty virtuálního světa tak, aby odpovídal jejich pojetí reality a každodenní zkušenosti s fyzikou. Je zbytečné podotýkat, že taková animace by v případě mnoha předmětů byla velmi náročná a hlavně by vůbec nemusela odpovídat tomu, jak se daná věc či předmět chová ve skutečnosti. Nakonec animátor není fyzik. S velikým úspěchem se proto používá těchto hotových fyzikálních modelů, kdy se pouze vytvoří základní grafický model předmětu a ten se poté nechá rozpohybovat. Model je tím univerzálnější, čím více vstupních parametrů obsahuje. Můžeme tak např. použít jeden grafický objekt kosmonauta a nechat animovat jeho pohyb třeba na Marsu, Měsíci, Zemi nebo v beztíži, aniž by animátor hnul prstem.

2 Rovnice klasické mechaniky

Obecně lze pohyb v takto modelovaném systému tuhých těles vyjádřit pomocí vzorců klasické mechaniky. Síla, která způsobuje pohyb, má obecně 2 faktory: posuvný účinek a rotační účinek. Posuvný účinek vyjadřuje vektor výslednice všech sil, rotační účinek sil zas vyjadřuje celkový moment síly. Newtonovy pohybové rovnice lze obecně vyjádřit i pro rotaci - to charakterizuje tabulka 1 (1).

Poloha	$\mathbf{r}$	$\boldsymbol{\varphi}$
Rychlost	$\mathbf{v} = \frac{d\mathbf{r}}{dt}$	$\boldsymbol{\omega} = \frac{d\boldsymbol{\varphi}}{dt}$
Zrychlení	$\mathbf{a} = \frac{d\mathbf{v}}{dt}$	$\boldsymbol{\varepsilon} = \frac{d\boldsymbol{\omega}}{dt}$
Hmotnost	m	$\mathbf{I}$
Síla	$\mathbf{F} = m\mathbf{a}$	$\mathbf{M} = \mathbf{I}\boldsymbol{\varepsilon}$

Tabulka 1: Pohybové rovnice posuvného a rotačního pohybu

Jak snadno vidíme z tabulky 1, všechny základní pohybové rovnice klasické mechaniky jsou diferenciální. To znamená, že numerické řešení systému postaveného na silách bude vyžadovat nějakou numerickou metodu řešení diferenciálních rovnic pro výpočet příslušných veličin.

3 Metody numerického řešení obyčejných diferenciálních rovnic

Metody se liší především tím, jak rychle jejich řešení konverguje ke správnému analytickému řešení problému. Těchto metod existuje velké množství, avšak obecně se pro typy fyzikálních simulací, se kterými se zde zabývám, používají následující tři metody (2).

Eulerova metoda

Nejjednodušší metoda, jejíž použití je však vykoupeno velkou chybou, neboť k analytickému řešení konverguje nejpomaleji.

$$x(t + \Delta t) = x(t) + \mathbf{v}\Delta t$$
$$v(t + \Delta t) = v(t) + \mathbf{a}\Delta t$$

Verletova metoda

Pravděpodobně nejčastěji používaná metoda v real-time simulacích (2). Kompromis mezi chybou a složitostí implementace.

$$x(t + \Delta t) = x(t) + \mathbf{v}\Delta t + \frac{1}{2}\mathbf{a}\Delta t^2$$

$$v(t + \Delta t) = v(t) + \frac{\mathbf{a}(t) + \mathbf{a}(t + \Delta t)}{2}\Delta t$$

Metoda Runge-Kutta

Je nejspolehlivější metoda pro numerické řešení - k analytickému řešení konverguje nejrychleji. V první řadě se však hodí spíše pro jednorázové simulace. Do obecných systémů nemusí být jednoduché ji implementovat. Pro úplnost zde uvádím rovnice metody RK čtvrtého řádu (RK4):

$$x_{n+1} = x_n + \frac{1}{6}(k_1 + k_2 + k_3 + k_4)\Delta t$$
$$k_1 = a(t_n, y_n),\ k_2 = a\left(t_n + \tfrac{1}{2}\Delta t, y_n + \tfrac{1}{2}\Delta t k_1\right)$$
$$k_3 = a\left(t_n + \tfrac{1}{2}\Delta t, y_n + \tfrac{1}{2}\Delta t k_2\right),\ k_4 = a(t_n + \Delta t, y_n + \Delta t k_3)$$

V závislosti na tom, jakou zvolíme metodu, bude simulace vykazovat buď menší, nebo větší tzv. „energetický drift“. Vzhledem k tomu, že numerické řešení se vždy bude jen blížit k ideálnímu analytickému, nemůžeme z principu docílit dokonale všech zákonů zachování (zejména zákona zachování energie - proto energetický drift). Kyvadlo se tak v prostředí bez odporu nebude kývat pořád stejně, ale na každou periodu kyvu připadne nějaká chyba. Celková energie kyvadla se tak bude buď ztrácet, nebo bude přibývat. V konečném čase se tak kyvadlo buď zastaví, nebo přepadne a začne se otáčet. Podle toho, jakou zvolíme numerickou metodu, můžeme v závislosti na její chybě ovlivnit pouze velikost driftu, nelze jej však zcela vyloučit.

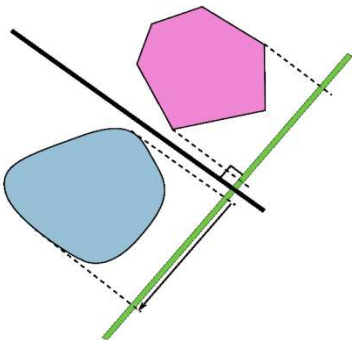
4 Detekce a reakce na kolizi

Jelikož celou simulovanou scénu nyní zobrazujeme a jedná se o soustavu těles, mohou tělesa také interagovat mezi sebou. Stojíme tak před problémem detekce srážky dvou (či více těles) a posléze pak i nutností reagovat na takovou srážku v souladu s fyzikou. Samotný problém detekce závisí na typu zvolené zobrazovací metody. Způsobů detekce je mnoho, a protože se jedná o matematický problém, uvedu jen stručně mnou zvolenou metodu ve dvojrozměrném zobrazení, poté se vrátíme k fyzikálnímu problému reakce na probíhající srážku.

Věta o oddělující přímce (Separating axis theorem)

Jeden z nejspolehlivějších způsobů detekce průniku dvou útvarů je využití tzv. věty o oddělující přímce (SAT). Tento poznatek analytické geometrie udává nutnou a postačující podmínku pro neexistenci průniku dvou geometrických útvarů v rovině (větu lze ale také upravit i pro případ v prostoru – z přímky tak bude rovina):

„Nechť jsou dány 2 konvexní útvary. Potom existuje taková přímka, na které jsou projekce těchto útvarů disjunktní právě tehdy, když neexistuje průnik těchto útvarů.“



Obrázek 1: Dělící přímka dvou konvexních útvarů

Na obrázku 1 můžeme vidět, že skutečně pro dva konvexní disjunktní útvary existuje taková společná přímka (zeleně), na které jsou i jejich projekce disjunktní. V praxi se tedy musí projít jednotlivé hrany obou kandidátů na kolizi a udělat projekci těchto hran na všechny možné přímky, které se zase vytvoří z hran. Maximální počet kroků je tedy $(k_1+k_2)^2$, kde k_1 a k_2 jsou počty hran dvou těles. Výhodou však je, že algoritmus může skončit okamžitě, jakmile se nalezne disjunktní projekce na jediné přímce. Tedy v reálném případě – kdy kolize nestávají příliš často - je tento algoritmus velmi efektivní. Nevýhodou je, že věta platí jen pro konvexní útvary, takže jakákoliv nekonvexní tělesa je nutné „zabalit“ do konvexního obalu – vznikají tak nepřesnosti v detekci kolizí.

Reakce na kolizi

Jakmile detekční algoritmus zjistí kolizi dvou těles, dává hned automaticky podnět fyzikálnímu modelu, který musí zařídit adekvátní odpověď na právě nastávající srážku. Při srážce v reálném světě působí v okamžik srážky na obě tělesa impuls síly. Tento okamžik je však nekonečně malý a tudíž nejsme schopni něco takového nasimulovat. Je tedy nutné uchýlit se k tomu, že změníme rychlosti dvou kolidujících těles přímo – rychlosti obou těles po srážce schopni přesně spočítat. V klasické mechanice existují rovnice, kterými lze popsat libovolnou srážku jen na základě hmotností a rychlostí obou těles. Lze do nich také velmi elegantně zapojit i fakt, že při srážce, která není ideálně pružná, se ztratí určitá energie v pružnosti obou těles. Tento jev vystihuje tzv. *koeficient restituce*. Udává poměr rychlosti dvou těles před srážkou a po srážce. Z definice koeficientu restituce tedy vyjdeme i při odvození (**n** je normálový vektor ke společné hraně srážky)

$$\left(\mathbf{v}_2^{\mathrm{A}} - \mathbf{v}_2^{\mathrm{B}}\right)\mathbf{n} = -\mathrm{k}\left(\mathbf{v}_1^{\mathrm{A}} - \mathbf{v}_1^{\mathrm{B}}\right)\mathbf{n} \tag{1.1}$$

Pro výstupní rychlost prvního tělesa však také platí

$$\mathbf{v}_2^{\mathrm{A}} = \mathbf{v}_1^{\mathrm{A}} + \frac{\mathrm{j}}{\mathrm{m_A}}\mathbf{n} \tag{1.2}$$

a pro výstupní rychlost druhého tělesa rovnice

$$\mathbf{v}_2^{\mathrm{B}} = \mathbf{v}_1^{\mathrm{B}} + \frac{\mathrm{j}}{\mathrm{m_B}}\mathbf{n}. \tag{1.3}$$

Rozdíl rovnic (1.2) a (1.3) dosadíme za člen na levé straně v rovnici (1.1) a vyjádříme *j* (přitom označíme $\mathbf{v_1}^{\mathrm{A}}$-$\mathbf{v_1}^{\mathrm{B}} = \mathbf{v_1}^{\mathrm{AB}}$)

$$\mathrm{j} = \frac{-(\mathrm{k}+1)\mathrm{v}_1^{\mathrm{AB}}.\mathbf{n}}{\mathbf{n}.\mathbf{n}\left(\frac{1}{\mathrm{m_A}} + \frac{1}{\mathrm{m_B}}\right)} \tag{1.4}$$

Když dosadíme *j* zpátky do rovnic (1.2) a (1.3) získáme rychlosti těles po srážce. Srážka však nemá pouze posuvný účinek, ale opět i rotační. Použitím stejného *j* z rovnice (1.4) získáme z rovnic (1.5) a (1.6) úhlové rychlosti těles po srážce ($\mathbf{r}_\perp^{\mathrm{AB}}$ je vektor kolmý k vektoru vzájemné polohy těles). Pro první těleso máme výstupní uhlovou rychlost

$$\boldsymbol{\omega}_2^{\mathrm{A}} = \boldsymbol{\omega}_1^{\mathrm{A}} + \frac{\mathbf{r}_\perp^{\mathrm{AB}}}{\mathbf{I_A}}\mathrm{j}\mathbf{n} \tag{1.5}$$

a pro druhé těleso výstupní úhlovou rychlost

$$\boldsymbol{\omega}_2^{\mathrm{B}} = \boldsymbol{\omega}_1^{\mathrm{B}} + \frac{\mathbf{r}_\perp^{\mathrm{AB}}}{\mathbf{I_B}}\mathrm{j}\mathbf{n}. \tag{1.6}$$

5 Závěr

Ve své vlastní implementaci fyzikálního modelu jsem kvůli jednoduchosti použil Eulerovu metodu. Je na ní možné demonstrovat velikost její chyby, resp. energetický drift. Model každému tělesu přiřazuje tzv. akumulátor celkové síly a momentu sil působící na těleso. Z těchto celkových veličin se pak výše zmíněnou integrační metodou získávají kinematické veličiny. Zobrazovací metodu jsem rovněž, kvůli jednoduchosti implementace, zvolil 2D. Celkově tedy tento dokument shrnuje použité metody v mojí implementaci simulace fyziky. Celá má vlastní implementace je dostupná ke stažení, a to včetně zdrojových kódů (3).

Reference

[1] I. Štoll**,** *Mechanika,* ČVUT, Praha (1995)
[2] O. Mocný**,** *Real-time fyzikální simulace pro mobilní zařízení,* MFF UK, Praha (2009)
[3] L. Pohanka, *PhysBox,* https://github.com/NumberFour8/PhysBox
[4] kol. autorů, *Numerické řešení obyčejných diferenciálních rovnic,*
http://cs.wikipedia.org/wiki/Numerické_řešení_obyčejných_diferenciálních_rovnic.
[5] Ch. Hecker, *Rigid body dynamics,* http://chrishecker.com/Rigid_Body_Dynamics