Simulace těles s různou tuhostí

Semestrální projekt předmětu B4M39MMA

Karel Tomanec, Vojtěch Pospíšil

7. ledna 2022

Abstrakt

V této semestrální práci se zabýváme studiem a implementací simulace těles s různou tuhostí. Cílem této práce je vytvoření aproximace reálného chování měkkých těles, která umožní deformaci komplexních objektů prostřednictvím věrohodné simulace.

1 Teorie

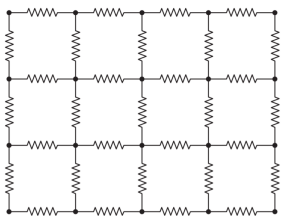
Tělesa z pevných látek v reálném světě nemění svůj tvar. Změnu tvaru způsobí až dostatečně velká síla, která na jednotlivé části tělesa působí. Tělesa z pevných látek, která mění svůj tvar při působení „běžných sil“ (gravitace, vítr, odpor vzduchu), nazýváme měkká tělesa. Mezi tyto tělesa patří například měkký míč, želé nebo molitan.

Dynamika těchto těles je v počítačové grafice simulována složitějším způsobem, než u těles tuhých. U tuhých těles se vzdálenost mezi jednotlivými dvěma body v průběhu simulace nemění a je tedy možné jej reprezentovat jedním fyzikálním stavem (rychlost, pozice, ...) pro celý objekt. Geometrii měkkého tělesa je potřeba aproximovat dostatečně velkým počtem hmotných bodů, které jsou vzájemně propojené a simulovat každý tento bod zvlášť. Vzájemné propojení hmotných bodů nám umožňuje zachování struktury tělesa do určité míry, kterou ovlivňují parametry podle zvoleného modelu simulace měkkých těles.

1.1 Fyzikální model

Jedním z nejčastěji používaných modelů pro simulaci měkkých těles je systém hmotných bodů a pružin. Tento systém si pro jednoduché těleso ve dvou dimenzích můžeme představit jako je znázorněno na obrázku 1. Tento jednoduchý model byl poprvé popsán v článku [1].

Takto vytvořená struktura je vhodná pro simulaci měkkých těles a také pro speciální případy, jako je simulace látek. U měkkých těles je ovšem potřeba vyřešit problém určitého zachování objemu tělesa. Z představy tohoto fyzikálního modelu také plyne problém, že u složitých těles je potřeba vytvoření velkého množství bodů a vzájemných propojení, což přirozeně vede k větší výpočetní náročnosti.



Obrázek 1: Vnitřní struktura měkkého tělesa.

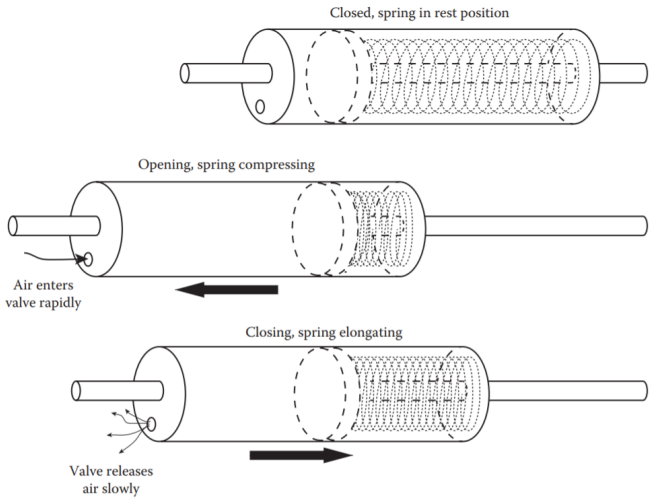
Způsob výpočtu kolizí u těchto modelů musí být také prováděn jiným způsobem, než u tuhých těles nebo u částicových systémů. Jak uvidíme později, je zde jistá podobnost z obou případů.

1.1.1 Tlumená pružina

Základním stavebním kamenem modelu naší simulace je model propojení mezi jednotlivými hmotnými body. Jak už bylo zmíněno, propojení tvoří tzv. pružiny. Úkolem těchto pružin je udržet model v klidovém stavu způsobem, který odpovídá oscilujícímu chování pružin. Dalším požadavkem je zároveň konfigurovatelnost těchto pružin, abychom mohli vytvářet měkké tělesa s různou tuhostí. Námi použitý model tohoto propojení se nazývá tlumená pružina.

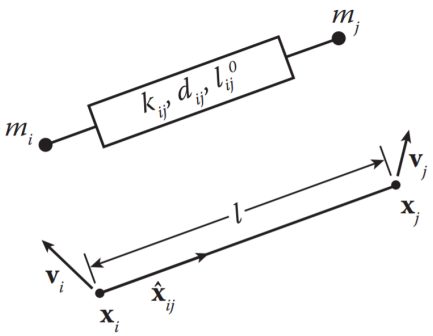
Tlumenou pružinu si můžeme představit jako dveřní zavírač, který můžeme vidět na obrázku 2, ten se skládá z pístu a pružiny. V horní části obrázku můžeme vidět pružinu v klidovém stavu. Jakmile se pružina stlačí, tak píst nasaje vzduch, který při uvolnění pružiny postupně uniká a nedovolí pružině rychlou oscilaci a díky tomu se mohou dveře pomalu zavřít. Přesně takové chování očekáváme od modelu, který představuje naši tlumenou pružinu. Je tedy potřeba tento model fyzikálně popsat.

Jelikož hrany polygonálních modelů budeme nahrazovat těmito tlumenými pružinami, tak je budeme nazývat vzpěry. Vzpěra je tedy struktura skládající se z tlumiče a pružiny. Díky tomuto rozdělení můžeme obě komponenty popsat zvlášť. Vzpěra je tedy propojení dvou bodů i a j o hmotnostech m_i a



Obrázek 2: Dveřní zavírač.

m_j jak vidíme na obrázku 3. Pružina je specifikována koeficientem pružení k_{ij}, koeficientem tlumení d_{ij} a klidovou délkou l_{ij}. Oba koeficienty ovlivňují tuhost objektu a tedy finální chování modelu měkkého tělesa. Klidová délka je stanovená geometrií modelu, který reprezentuje tvar modelu v klidovém stavu, tedy bez působení fyzikálních sil.



Obrázek 3: Struktura tlumené pružiny (vzpěra).

Nejprve začneme matematickým popisem pružiny. Mějme dva hmotné body na pozicích $\mathbf{x}_i$ a $\mathbf{x}_j$. Vektor z bodu i do bodu j označme jako $\mathbf{x}_{ij}$ a jeho délku jako $l = ||\mathbf{x}_{ij}||$. Směr z bodu i do bodu j označme jako $\hat{\mathbf{x}}_{ij} = \mathbf{x}_{ij}/l$. Pružící síla $\mathbf{f}_i^s$ působící na hmotný bod i, je určená rozdílem délek v aktuálním stavu a klidovém stavu $l_{ij} - l_{ij}^0$.

$$\mathbf{f}_i^s = k_{ij}(l_{ij} - l_{ij}^0)\hat{\mathbf{x}}_{ij} \tag{1}$$

Podle třetího Newtonova zákona pružící síla působící na hmotný bod j je:

$$\mathbf{f}_j^s = -f_i^s \tag{2}$$

Tlumení je dáno rozdílem rychlostí tělesa j a i. Tlumící síla $\mathbf{f}_i^d$ působící na hmotný bod i je tedy:

$$\mathbf{f}_i^d = d_{ij}[(\mathbf{v}_j - \mathbf{v}_i) \cdot \hat{\mathbf{x}}_{ij}]\hat{\mathbf{x}}_{ij} \tag{3}$$

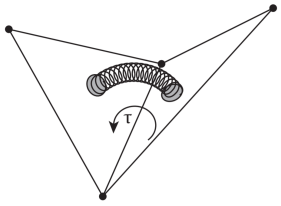
Opět podle třetího Newtonova zákona tlumící síla působící na hmotný bod i je:

$$\mathbf{f}_j^d = -f_i^d \tag{4}$$

1.1.2 Torzní pružina

Při pokusech o simulaci měkkého tělesa pouze vzpěry mezi body jsme došli k závěru, že takovýto model pro zachování objemu tělesa je pro obecný 3D model nevhodný a není škálovatelný. Rozhodli jsme se proto přidat fyzikální model torzní pružiny.

Torzní pružinu si můžeme představit tak, jako je znázorněna na obrázku 4. Torzní pružina tedy propojuje každé dva sousední trojúhelníky v 3D modelu a simuluje nám točivý moment $\boldsymbol{\tau}$.



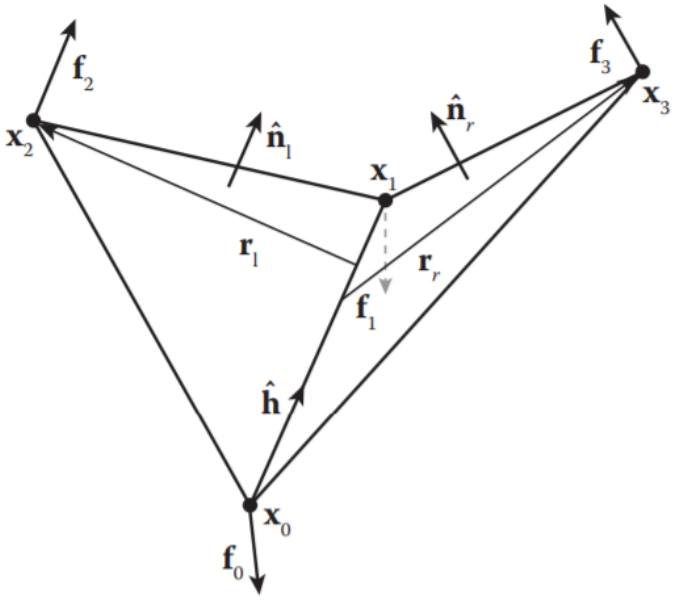
Obrázek 4: Torzní pružina.

$$\boldsymbol{\tau} = \mathbf{r} \times \mathbf{f} \tag{5}$$

Velikost tohoto točivého momentu je dána rozdílem aktuálního úhlu θ mezi normálami sousedních trojúhelníků a úhlem θ_0 v klidovém stavu. Zavádíme zde koeficient pružnosti k_θ, který reprezentuje sílu takovéto pružiny.

$$||\boldsymbol{\tau}|| = k_\theta|\theta - \theta_0| \tag{6}$$

Obrázek 5 znázorňuje matematický popis celé situace. Cílem je tedy výpočet sil $\mathbf{f}_1$ až $\mathbf{f}_4$ působící na hmotné body 1 až 4. Směrový vektor $\hat{\mathbf{h}}$ mezi body 0 a 1 nám reprezentuje tzv. pant a jedná se tedy o osu otáčení sousedních trojúhelníků.



Obrázek 5: Matematický popis torzní pružiny.

Nejprve zadefinujeme vektory $\mathbf{r}_l$ a $\mathbf{r}_r$, které vzniknout vztyčením kolmic z osy pantu k bodům 2 a 3.

$$\mathbf{r}_l = \mathbf{x}_{02} - (\mathbf{x}_{02} \cdot \hat{\mathbf{h}})\hat{\mathbf{h}} \tag{7}$$

$$\mathbf{r}_r = \mathbf{x}_{03} - (\mathbf{x}_{03} \cdot \hat{\mathbf{h}})\hat{\mathbf{h}} \tag{8}$$

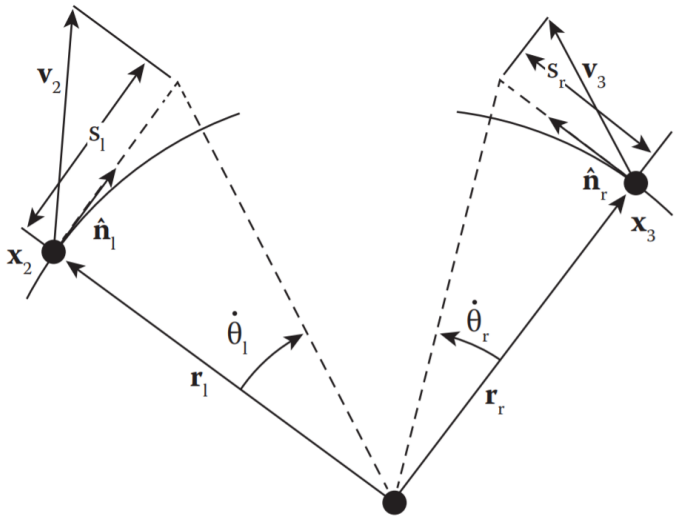
Dalším krokem je výpočet úhlu θ mezi normálami obou trojúhelníků:

$$\theta = \tan\left(\frac{(\hat{\mathbf{n}}_l \times \hat{\mathbf{n}}_r) \cdot \hat{\mathbf{h}}}{\hat{\mathbf{n}}_l \cdot \hat{\mathbf{n}}_r}\right)^{-1} \tag{9}$$

Pružící část točivého momentu vypočítáme následujícím vzorcem:

$$\boldsymbol{\tau}_k = k_\theta(\theta - \theta_0)\hat{\mathbf{h}} \tag{10}$$

Pro torzní pružinu musíme vytvořit tlumící složku. Opět musíme vycházet z rychlosti hmotných bodů a to tentokráte bodů 2 a 3. Celou situaci znázorňuje obrázek 6. Z obrázku vidíme, že je třeba zjistit velikost rychlosti (*speed*). To lze provést projekcí vektoru rychlosti na normálu daného trojúhelníku jako $s_l = \mathbf{v}_2 \cdot \hat{\mathbf{n}}_l$ a $s_r = \mathbf{v}_3 \cdot \hat{\mathbf{n}}_r$.



Obrázek 6: Matematický popis pro výpočet tlumící složky torzní pružiny.

Úhlové rychlosti v radiánech za sekundu spočítáme tedy jako:

$$\dot{\theta}_l = s_l / ||\mathbf{r}_l|| \tag{11}$$

$$\dot{\theta}_r = s_r / ||\mathbf{r}_r|| \tag{12}$$

Výslednou tlumící složku točivého momentu vypočítáme jako součet těchto úhlových rychlostí vynásobený záporným koeficientem tlumení torzní pružiny a směrovým vektorem pantu:

$$\boldsymbol{\tau}_d = -d_\theta(\dot{\theta}_l + \dot{\theta}_r)\hat{\mathbf{h}} \tag{13}$$

Celkový točivý moment je tedy součet dílčích složek:

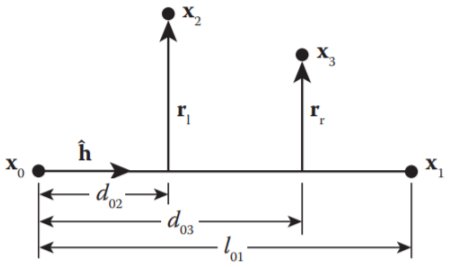
$$\boldsymbol{\tau} = \boldsymbol{\tau}_k + \boldsymbol{\tau}_d \tag{14}$$

Nyní už máme vše potřebné pro výpočet sil $\mathbf{f}_0$ až $\mathbf{f}_4$. Pokud na systém nepůsobí žádné vnější síly musí platit:

$$\mathbf{f}_0 + \mathbf{f}_1 + \mathbf{f}_2 + \mathbf{f}_3 = 0 \tag{15}$$

Kdyby předchozí rovnost neplatila, tak by těleso mohlo zrychlovat působením sil vygenerovaných uvnitř tělesa. Můžeme také využít toho, že pokud najdeme síly $\mathbf{f}_1$, $\mathbf{f}_2$ a $\mathbf{f}_3$, tak zbylou vypočítáme jako:

$$\mathbf{f}_0 = -(\mathbf{f}_1 + \mathbf{f}_2 + \mathbf{f}_3) \tag{16}$$



Obrázek 7: Popis vzdáleností v modelu torzní pružiny.

Síly působící na vnější vrcholy 2 a 3 jsou vytvořený točivým momentem a musí být tedy rovnoběžné na normály jednotlivých trojúhelníků. Platí $\boldsymbol{\tau} = \mathbf{r}_l \times \mathbf{f}_2$. Z tohoto můžeme dostat:

$$\frac{\boldsymbol{\tau} \cdot \hat{\mathbf{h}}}{||\mathbf{r}_l||} = (\hat{\mathbf{r}}_l \times \mathbf{f}_2) \cdot \hat{\mathbf{h}} \tag{17}$$

Jelikož jsou vektory na pravé straně rovnice navzájem ortogonální a vektor $\mathbf{f}_2$ je rovnoběžný s normálou trojúhelníku, můžeme psát:

$$\mathbf{f}_2 = \frac{\boldsymbol{\tau} \cdot \hat{\mathbf{h}}}{||\mathbf{r}_l||}\hat{\mathbf{n}}_l \tag{18}$$

Podobně pro bod 3:

$$\mathbf{f}_3 = \frac{\boldsymbol{\tau} \cdot \hat{\mathbf{h}}}{||\mathbf{r}_r||}\hat{\mathbf{n}}_r \tag{19}$$

Výpočet síly pro bod 1 vychází balanční rovnice pro točivé momenty a nebudeme zde uvádět jeho odvození. Detailní odvození lze nalézt v knize [2]. Vzdálenosti použité při výpočtu jsou vyobrazeny na obrázku 7.

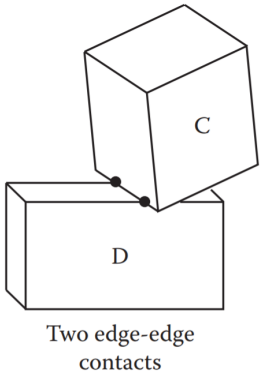
$$\mathbf{f}_1 = -\frac{d_{02}\mathbf{f}_2 + d_{03}\mathbf{f}_3}{l_{01}} \tag{20}$$

1.2 Kolize

Kolize měkkých těles se statickými tuhými tělesy se dělí na dva typy.

První typ kolize je stejný jako u částicového systému, kdy testujeme kolizi jednoho hmotného bodu měkkého tělesa a statického objektu. Druhý typ kolize můžeme vidět na obrázku 8, kdy se setká hrana měkkého tělesa C s hranou jiného tělesa D. Implementací druhého typu kolize jsme se bohužel nezabývali a plánujeme jej zahrnout v dalším rozšíření naší implementace.

Dalším problémem, kterým jsme se při implementaci nezabývali je výpočet kolizí mezi dynamickými objekty a bude taktéž zahrnut v dalším rozšíření naší implementace.



Obrázek 8: Situace při výpočtu kolizí.

2 Architektura simulace

2.1 Datové struktury

Pro reprezentaci měkkého tělesa, je potřeba převést existující mesh do vlastní reprezentace, která bude odpovídat popsanému matematickému modelu. Toto převedení se provede při spuštění aplikace a následně jsou simulace prováděny na takto upraveném modelu.

2.1.1 Starší implementace

V kódu aplikace jsou ponechány i zdrojové kódy ke starší implementaci, která nevyužívala torzních pružin a jednalo se tak u vzpěr pouze o tlumené pružiny. Všechny třídy patřící k této implementaci jsou ve složce SB a třídy uvozeny prefixem SB.

Od této implementace bylo upuštěno právě díky zavedení torzních pružin do modelu pro zachování objemu těles.

2.2 Struktura tříd

Aplikace se v současné době sestává z těchto tříd (uveden je vždy i krátký popis její existence):

- **Vertex** - hmotný bod
- **Strut** - vzpěra (= tlumená + torzní pružina)
- **Face** - jednotlivé plošky objektu
- **State** - stav hmotného bodu (rychlost, pozice, síla)
- **SpringyMeshBase** - rodičovská třídy měkkého tělesa
- **SpringyMeshPrimitive** - měkký objekt v podobě krychle nebo tetrahedronu
- **SpringyMeshIcoSphere** - měkký objekt v podobě koule
- **SpringyMeshMesh** - měkký objekt v podobně obecné meshe
- **ShapeGenerator** - statická třída umožňující procedurální generování několika primitiv
- **Plane** - rovina využívaná pro kolize s měkkými objekty
- **Triangle** - trojúhelník využívaný pro kolize
- **Cascade** - třída tvořící „kaskádu“ plošek skrze které objekt padá

Detailní popis jednotlivých funkcí a členů těchto tříd je uveden v dokumentaci v kódu a samotný proces implementace a běhu simulace bude popsán v kapitole o implementaci.

3 Implementace

V této kapitole popíšeme průběh našeho vývoje, zaměříme se na samotnou implementaci a popis toho jak je simulace strukturována, ale také popíšeme proces pomocí kterého jsme se dopracovali k finální implementaci.

3.1 Průběh vývoje

První verzi jsme se rozhodli založit na intuici a znalostech získaných z přednášek. Jednalo se tedy o pouhý převod meshe kde vrcholy byly převedeny na hmotné body a hrany na tlumené pružiny. Tento přístup má ovšem hlavní nedostatek v podobě nezachování objemu simulovaného tělesa, cože vede ke kolapsu všech plošek do jedné kolizní roviny.

Následně jsme se rozhodli dodat do objektu vzpěry mezi všechny dvojice bodů. Tento postup byl například u krychle naprosto dostatečný, ale s rostoucí složitostí geometrie a zvyšujícím se počtem vertexů nám počet vzpěr roste kvadraticky a už například u IcoSphere s detailnějším dělením docházelo ke značné výpočetní náročnosti. A bylo tedy třeba zajistit způsob pro snížení celkového počtu vzpěr.

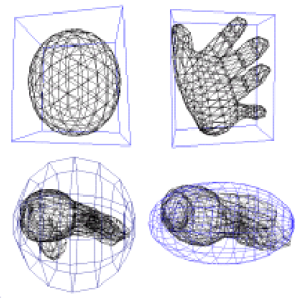
Například u zmíněné koule nás napadlo řešení v podobě aplikování síly ve směru od středu koule v každém vertexu. Tento způsob by byl použitelný dobře právě u koule, ale například u obecných meshů není jasné jakým směrem tuto sílu aplikovat a tudíž to není úplně univerzální způsob.

V tuto chvíli jsme se rozhodli provést rešerši existujících řešení, popřípadě článků pro inspiraci při řešení tohoto problému. Jedním z prvních výsledků byl článek Pressure Model of Soft Body Simulation[3], který je založen na modelu ideálního plynu, kde ze stavové rovnice

$$PV = nRT \tag{21}$$

kde P je hodnota tlaku, V je objem tělesa, n je počet částic plynu, R je konstanta ideálního plynu a T je teplota, jednoduše vyjádřit tlak, který je později aplikován ve formě síly na jednotlivé plošky ve směru jejich normálového vektoru.

Nevýhodou tohoto řešení je problém výpočtu objemu simulovaného tělesa. Autoři doporučují přístup pomocí aproximace přes různá obalová tělesa (viz 9), nicméně je to způsob vhodný spíše pro jednodušší objekty. Jelikož jsme chtěli implementovat pokud možno univerzální řešení rozhodli jsme se tuto metody nevyužít.



Obrázek 9: Ukázky aproximačních obalových těles. Zdroj [3]

Dalším směrem bylo nalezení knihy Foundations of Physically Based Modeling and Animation[2]. Tato kniha se zabývá obecně tématy fyzikálně založené simulace a animace a jedna z kapitol se zaměřuje na simulaci měkkých těles. Metodou pro zachování objemu je zde zavedení torzních pružin které jsou blíže popsány v teoretické části reportu. I přes komplikace při implementaci které budou zmíněny v příslušné kapitole, se tato metoda ukázala jako vhodná pro použití u libovolného objektu a simulace chování měkkého tělesa vypadala velice věrně. Rozhodli jsme se tedy u této metody zůstat a považujeme ji za naši finální.

3.2 Průběh simulace

V první řadě je vždy nutné zajistit tvorbu statického modelu (v podobě procedurálního vygenerování, popřípadě importu libovolné meshe pomocí Unity do scény a následně odkazování na mesh tohoto objektu). Tento statický model je ve třídě *SpringyMesh** a její metodě *InitObject* převeden do naší reprezentace v podobě hmotných bodů, vzpěr a plošek objektu. K tomuto převodu je využíváno několika pomocných tříd specifikovaných ve *SpringyMeshBase*.

Po převedení objektu do naší reprezentace je na řadě samotná simulace, která probíhá ve *FixedUpdate* metodě poskytovanou Unity enginem, která zajišťuje volání pouze jednou za snímek. V této fázi jsou k jednotlivým vertexům postupně přičteny gravitační síla, volitelně odpor vzduchu (pouze triviální zpomalení, nezávisle na tvaru objektu), volitelně vítr. Následně se aplikují síly plynoucí z pružin (aplikace na koncové body pružiny a zbylé body ve dvou přilehlých ploškách) a jako poslední část je výpočet kolizí. tento výpočet je momentálně rozdělen poněkud neoptimálně na dvě různé větve a to v případě použití kolize pouze s rovinou nebo při kolizi s obecnými trojúhelníky. O tento výpočet se tedy vždy postará samotný vertex který zkontroluje zdali kolize s příslušným objektem nastala a pokud ano, tak aplikuje síly odpovídající této kolizi.

3.3 Konfigurace simulace

Hledání vhodných parametrů je jedním z problémů při simulacích tohoto typu. Často dochází k různým jevům v podobě malých vibrací v klidovém stavu, k nelogickému „klepání“ objektu a podobně. Parametry, které lze u měkkého tělesa měnit jsou následující:

- **mass** - celková hmotnost tělesa, která bude následně rozdistribuována do hmotných bodů
- **T** - časová konstanta (čas za jaký se amplituda zmenší o faktor e)
- **P** - doba trvání netlumené periody
- $\mathbf{T}_\theta$ - stejné jako v předchozím případě, ale pro torzní pružinu
- $\mathbf{P}_\theta$ - stejné jako v předchozím případě, ale pro torzní pružinu

3.3.1 Distribuce hmotnosti

V případě jednoduchých, pravidelných těles je možno celkovou hmotnost tělesa rozdistribuovat mezi hmotné body triviálně, pouhým rovnoměrným rozdělením mezi jednotlivé body. U složitějších těles tento přístup způsobuje nerealistické chování a je zapotřebí distribuci hmotnosti přizpůsobit struktuře objektu. Způsobem doporučeným v [2] je distribuce hmotnosti do bodů na základě části plochy, kterou zabírají plošky sousedící s konkrétním vrcholem vůči celkové ploše tělesa. Aby tato distribuce zachovala správnou hmotnost, je také nutné počítat s úhlem, který konkrétní ploška svírá u vertexu. Implementace je dostupná v *SpringyMeshBase/SetupVerticesMassBasedOnAnglesAndAreas*

3.3.2 Výpočet koeficientů pružin z parametrů

Parametry nastavené uživatelem nenastavují přímo parametry pružin, ale jsou použity ve vzorcích pro výpočet koeficientu pružnosti a koeficientu tlumení. Je to z důvodu, že tyto koeficienty jsou následně upravovány podle délky pružin, tak aby bylo docíleno realistického chování po celém modelu. Vzorce ze kterých se vychází v případě tlumených pružin jsou tyto:

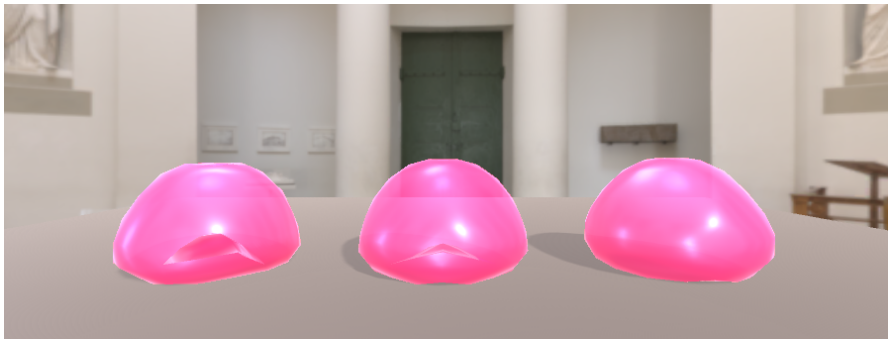
$$T = \frac{2m}{d} \tag{22}$$

$$P = 2\pi\sqrt{m/k} \tag{23}$$

Vzorce pro torzní pružiny jsou následující:

$$T_\theta = \frac{2mr^2}{d_\theta} \tag{24}$$

$$P_\theta = 2\pi r\sqrt{m/k_\theta} \tag{25}$$



Obrázek 10: Míče s různými parametry při stlačení.

Obrázek 11: Míče s různými parametry při odskoku.

V těchto vzorcích m označuje hmotnost (průměr z hmotnosti krajních bodů pružin), r je průměrná z velikosti vektorů r_l a r_r viz 4.

Více obrázků jsme do reportu nedávali, jelikož u takovéto simulace nedávají moc smysl. K reportu jsme však přiložili několik videoukázek.

3.4 Problémy

Při samotné implementaci jsme neměli zásadní problémy, pouze při tomto druhu programování se problémy velice špatně debugují a zjišťují kde se vlastně vznikají. Často jsme tedy pouze viděli že těleso se nám během tří snímků roztáhlo do obrovských rozměrů, nebo se nechovalo tak jak by mělo.

3.4.1 Detekce kolizí

Problémů u detekce kolizí bylo relativně hodně, ať už to byly problémy se samotnou implementací nebo problémy s debugem. Problémy se nakonec podařilo vyřešit kombinací implementací z několika zdrojů a zavedením správných epsilonových okolí u některých výsledků. Kolize jsou nakonec implementovány buď s rovinou, nebo s trojúhelníkem kde každá z těchto implementací je rozdělena zvlášť a nedají se použít zároveň.

3.4.2 Numerická integrace

Jako studenti oboru Počítačové hry a grafika jsme neměli žádný předmět týkající se numerických metod a tak jediné co jsme znali je Eulerova metoda využívaná ve hrách a tak jsme také iniciálně použili tu. Nevýhodou byla nutnost velmi malého timestepu simulace pro poskytnutí korektních výsledků, bohužel však krok byl tak malý že simulace nedokázala běžet v reálném čase. Museli jsme tedy použít jiné metody, konkrétně jsme zkusili Verletovu metodu, Leapfrog a nakonec jsme zůstali u sice výpočetně

náročnější Runge-Kutta 4, ale poskytovala nám dobré výsledky i s větším timestepem. Bohužel jsme integraci využívali spíše jako blackbox než abychom úplně rozuměli proč a jak se co děje.

3.4.3 Převod do naší reprezentace

Samotný převod není problematický, problémy jsou převážně v dostupných modelech. Počítáme s tzv. vodotěsným modelem (mesh je celistivá, bez děr) a zároveň s meshí složenou z jednoho jediného objektu. Objekty dostupné na internetu v této podobě často nebyly a tak bylo potřebné je patřičně upravit v modelovacích programech (např. Blender).

3.4.4 Volba parametrů

Jak již bylo zmíněno, tak volba parametrů není úplně průhlednou a intuitivní pro člověka touto tematikou nepolíbeného a tak většinou docházelo k užití metody „pokus, omyl“. Ovšem po prvotním nalezení parametrů, které byly v jistém smyslu stabilní už nebyl problém posouvat hodnoty aby se simulace chovala dle představy.

4 Potenciální rozšíření

Jelikož nás práce na tomto projektu velice bavila máme v plánu s implementací pokračovat i po semestru a dodělat ji do fáze kdy by mohla být užitečná i ostatním, jelikož jsme nikde na internetu implementaci této metody (a obecných soft-body meshů) pro Unity engine nenalezli.

4.1 Implementace mimo herní engine

V současné době herní engine používáme pouze jako renderer a veškeré výpočty ohledně simulace a fyziky provádíme sami, není tedy nutné použít moloch v podobě Unity. Pro simulace jednodušších objektů v podobě koule či jednodušších meshů jako součást například hry to smysl dává. Pro simulace složitějších popřípadě většího množství objektů už by se tento engine stával brzdou celé simulace. Pro tyto účely by tedy bylo vhodné využít implementaci na základě OpenGL popřípadě novějšího Vulkanu podloženou rychlým jazykem typu C++ či Rust.

4.2 Výpočet kolizí

Jak bylo zmíněno v současné době je naimplementována pouze kolize s rovinou nebo s trojúhelníky. Logickým krokem by bylo seskupení tohoto pod rodičovskou třídu pro možnost využití obojího.

Další možností je výpočet kolizí nejen v bodech objektů, ale řešit i případy kolize hran měkkých objektů s ostatními (statickými i dynamickými) objekty pro věrohodnější vzhled kolizí.

Poslední částí rozšíření kolizí je právě výpočet kolizí mezi měkkými tělesy zároveň. Pro tuto část by však bylo potřeba naimplementovat i akcelerační struktury a neprovádět testy kolizí v kvadratickém čase.

4.3 Paralelizace na GPU

Vzhledem k povaze úlohy a hromadnému charakteru se vybízí paralelizace úlohy pomocí výpočtu na GPU. V případě implementace v Unity by se dalo využít Compute Shaderů. Pokud by implementace byla mimo herní engine tak by se jednalo o použití CUDA popřípadě obecnějšího OpenCL. Tato paralelizace bude téměř jistě nutná pro využití u simulace složitějších či více objektů.

4.4 Iniciální odhad parametrů

Volba parametrů může být pro uživatele zdlouhavá a tak by bylo vhodné na základě vlastností objektu poskytnout iniciální odhad, který už by uživatel pouze „doladil“ dle svých představ a byl tak ušetřen zkoušení a hledání stabilního výchozího bodu.

5 Dokumentace přiložené aplikace

V této části pouze krátce popíšeme části aplikace a strukturu souborů pro snadnější orientaci.

5.1 Popis struktury repozitáře

Veškeré námi implementované a používané věci jsou v rámci Assetů enginu Unity a popíšu tak pouze strukturu složky *Assets*.

- **Materials** Složka pro použité materiály v ukázkových scénách.
- **Models** Složka s 3D modely.
- **Prefabs** Složka s před vytvořenými herními objekty pro použití.
- **Scenes** Složka s ukázkovými scénami.
- **Scripts** Složka s veškerými našimi skripty.

5.2 Popis scén

Pro snadnější kontrolu a ukázku jsme připravili několik scén s různými scénáři, které představí možnosti námi naprogramované simulace.

- **Bunny** Padající 3D model Stanford Bunny.
- **Cube** Padající kostka tvořená straší verzí implementace.
- **Fixed Single Sphere** IcoSphere zavěšená za jeden z jejích vrcholů protahující se vlastní vahou.
- **Jelly** Padající 3D model želé.
- **Ramp** Koule propadající sestavou šikmých plošin.
- **Sample** Scéna kterou jsme využívali pro obecné testy.
- **Single Sphere** Jednodušší scéna s jednou padající pružnou koulí.
- **Sphere** Náročnější scéna se třemi koulemi s různými vlastnostmi.

Reference

[1] Haumann, D. R., and Parent, R. E. The behavioral test-bed: Obtaining complex behavior from simple rules. *The Visual Computer 4* (2005), 332–347.

[2] House, D., and C., K. J. *Foundations of Physically Based Modeling and Animation.* CRC PRESS, 2020.

[3] Maciej, M., and Mark, O. Pressure model of soft body simulation. *arXiv: Computational Physics* (2004).