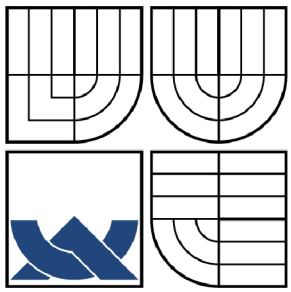


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ

ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY

FACULTY OF MECHANICAL ENGINEERING

INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

VYUŽITÍ „OPEN DYNAMICS ENGINE“ PRO MODELOVÁNÍ MOBILNÍCH ROBOTŮ

UTILISATIONS OF THE “OPEN DYNAMICS ENGINE” FOR MODELLING OF MOBILE ROBOTS

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — Bc. JIŘÍ KOVÁŘ

AUTHOR

VEDOUCÍ PRÁCE — Ing. PAVEL HOUŠKA, Ph.D.

SUPERVISOR

BRNO 2008

Abstrakt

Tato diplomová práce se zabývá problematikou virtuálního fyzikálního modelování mobilních robotů pro potřeby real-time řízení. Pro vytvoření virtuálního fyzikálního světa byl použit open-source projekt OPEN DYNAMICS ENGINE (ODE), pro zobrazovaní simulací bylo využito grafické rozhraní Microsoft DirectX. Simulované soustavy v ODE byly vytvořeny pomocí programovacího jazyka C# na platformě Microsoft.NET. Vlastnosti v ODE byly simulačně ověřovány na několika typech jednoduchých soustav a na zjednodušeném modelu robotu "Kráčmera I.". Následně byla ověřována použitelnost ODE pro řízení tohoto robotu.

Abstract

This diploma thesis deals with the problems of virtual physical modelling of mobile robots for the needs of their real-time control. To create a virtual physical world, an open-source project OPEN DYNAMICS ENGINE (ODE) was used, the results were displayed facilitating DirectX graphical interface. Simulated systems in ODE were written in C# on Microsoft.NET platform. The properites and qualities in ODE were verified by simulation in several types of simple systems and on a simplified robot model "Kracmera I.". Subsequently, the usability of ODE for its control was being verified.

Klíčová slova

OPEN DYNAMICS ENGINE, fyzikální model, simulace, dynamická knihovna

Keywords

OPEN DYNAMICS ENGINE, physical model, simulation, dynamic linked library

Bibliografická citace

KOVÁŘ, J. *Využití „Open Dynamics Engine“ pro modelování mobilních robotů*. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2008. 62 s. Vedoucí diplomové práce Ing. Pavel Houška, Ph.D.

Poděkování

Děkuji vedoucímu této diplomové práce Ing. Pavlu Houškovi, Ph.D. za odborné vedení, cenné rady a podněty při vypracování této práce. Děkuji také všem blízkým za podporu po dobu studia.

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci na téma „*Využití „Open Dynamics Engine“ pro modelování mobilních robotů*“ vypracoval samostatně pod vedením Ing. Pavla Houšky Ph.D., na základě použité literatury a dostupných informačních zdrojů, které jsem všechny odcitoval v seznamu literatury.

V Brně dne 23.5. 2008 Bc.Jiří Kovář

Obsah

1. Úvod

V průběhu doby - tak jak se člověk posouval na poli vědění - složitost soustav sestrojených člověkem vzrůstala natolik, že předpověď chování takových soustav v potřebném rozlišení už nelze pojmout jen intuitivně. Soustava je tedy nutně popisována matematickým aparátem. Využitelnost takového aparátu stoupá s výkonností výpočetní techniky.

Dostáváme tedy spojení lidské intuice a pochopení, přesnost matematického aparátu a konečně výkon výpočetní techniky. Výsledkem je tzv. model soustavy - jistá forma imaginace části reálného světa, který obsahuje zkoumanou soustavu, jejímž úkolem je usnadnit, zrychlit, zlevnit a zlepšit reálnou soustavu.

Jedním z prostředků k získání takového modelu je open-source projekt OPEN DYNAMIC ENGINE (ODE) [10], který byl původně vyvinut pro modelování fyzikálního prostředí v 3D hrách, podobně jako Newton Game Dynamics [11] nebo Vortex Physics [12]. ODE je založen na základních principech a zákonech reálného světa. Zabezpečuje tedy systém pravidel, kterému podléhá každé těleso v něm obsažené. Každý takový prvek má v tomto modelu dvě vlastnosti, a sice hmotnost a geometrii. Svět je tedy rozdělen na část, ve které jsou jen hmotná tvarově upřesněná tuhá tělesa a část, ve které je jen geometrie těles, tedy kolizní prostor.

Takový model dovoluje vytvořit a simulovat soustavu složenou z jednotlivých nebo složených těles včetně vazeb mezi nimi a poté určovat její pravděpodobné chování pomocí „časového“ krokování. Takovou soustavou může být například model robotu Kráčmera I. Použití dynamického modelu soustavy má množství použití, výhodou tohoto přístupu je rychlost určování budoucích stavů, díky čemuž může být použit i pro řízení robotu v reálném čase.

1.1. Formulace problému, účel a cíle řešení

K odhadu následného stavu robotu musíme mít takový systém pravidel, kterému bude opdovídat následný stav robotu. Stav robotu musí být pozorovatelný, řiditelný a deterministický. Máme-li řídit čtyřnohý kráčivý robot tak, aby byl schopen vykonávat určené cíle, je nutné plánovat několik stavů dopředu tak, aby nedošlo k ohrožení ani člověka, ani robotu, ale tak, aby došlo ke splnění určených cílů. Zároveň musí být zohledňována efektivnost chování robotu. Stavy robotu jsou dále označovány jako "časové" kroky.

Kráčivý podvozek je mimořádně vhodný k překonávání nerovností terénu, ve kterém se robot pohybuje, ale je adekvátně složité určovat posloupnost pohybů jednotlivých částí robotu tak, aby vyhovoval podmínkám řízení. Tento problém je dále komplikován statickou nestabilitou některých konstrukčních uspořádání kráčivých robotů; převážně se jedná o dvou a čtyřnohé roboty.

K tomu, aby se účinně dalo predikovat chování robotu, je nutné znát jeho dynamické vlastnosti. Ty jsou závislé na působících silách, momentech, rozložení hmoty. Predikovat stav robotu pomocí Langrangeových rovnic je sice přesné, ale jen ve velmi omezených případech použitelné a při každé změně hmotnosti, rozložení hmoty a okolních podmínek se musí rovnice upravovat. Navíc nejsme schopni jednoduše zakomponovat například sílu větru nebo kluzké prostředí do takové soustavy rovnic automaticky.

Řešením těchto problémů může být vytvoření modelu robotu ve virtuální realitě tak, aby se mohl automaticky model sám upravovat, přepočítávat budoucí stavy a aplikovat jistý druh učení. Jedním ze způsobů, jak vytvořit model k tomuto účelu, je použít OPEN DYNAMIC ENGINE (dále jen ODE).

Cílem této diplomové práce je tedy prověřit možnost využití ODE k vytvoření takového modelu, realizovat model čtyřnohého kráčivého robotu Kráčmera I. a navrhnout možnost použití tohoto prostředí a modelu pro potřeby řízení pohybu robotu.

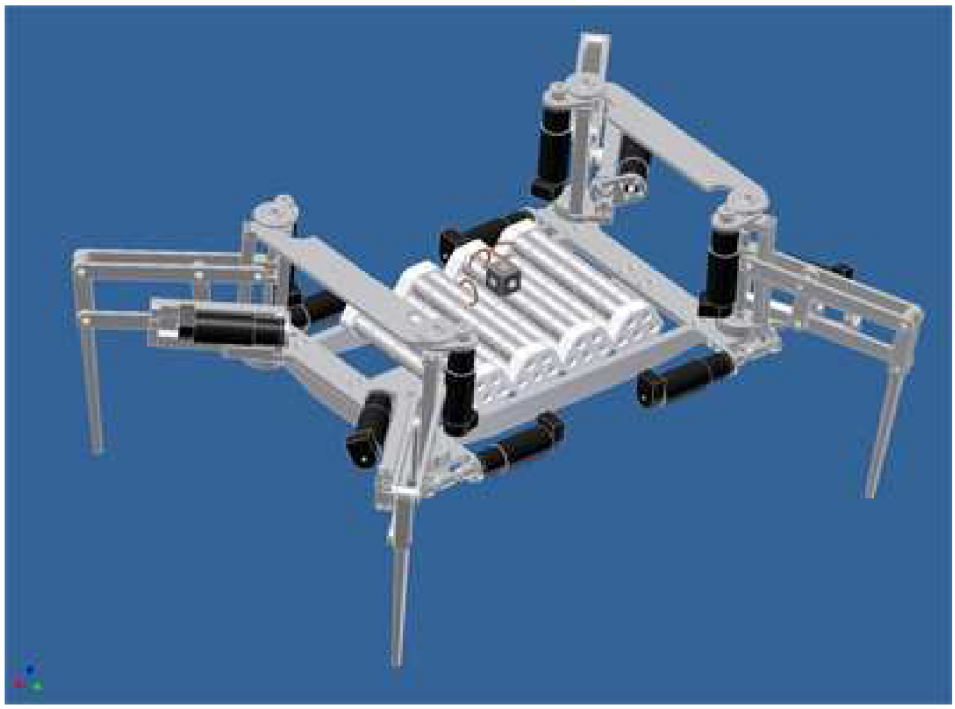
2. Mobilní kráčivé roboty

Tato kapitola čerpá ze zdrojů [16,17,1,3,14]

2.1. Dynamika mobilních kráčivých robotů

Kráčející podvozek je nelineární dynamický systém s mnoha stupni volnosti. Robot je složen z trupu a nohou. Popisem dynamiky kráčejícího podvozku rozumíme odpovídající dynamické rovnice celé soustavy. V průběhu pohybu robotu vznikají a zanikají silové a momentové vazby s terénem a se sebou samým, což má za následek změnu dynamiky soustavy a tím i jeho popisu. Dále je nutno započítat disipativní vlastnosti soustavy.

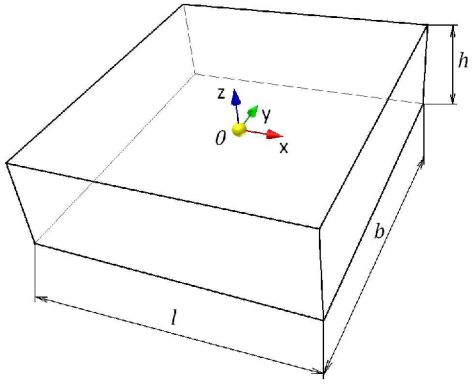
Takovým mobilním kráčivým robotem je i robot „Kráčmera I.“, jehož zjednodušený model je dále používán pro simulace v této práci.



Obr.1 Robot Kráčmera I. (Obrázek převzat z [3])

2.1.1. Dynamika trupu mobilního kráčivého robotu

Trup je hmotné těleso, jehož tvar je obecně složitý a navíc se v průběhu plnění úkolu může měnit. Pro zjednodušení aproximujeme trup jako kvádr, dále jako tuhé těleso s těžištěm v geometrickém středu.



Obr.2 Aproximace trupu

Dále předpokládáme hmotnost m a konstantní hustotu. Pak matice setrvačnosti je

$$I = \begin{bmatrix} I_X & 0 & 0 \\ 0 & I_Y & 0 \\ 0 & 0 & I_Z \end{bmatrix} \tag{1}$$

kde I_X ,I_Y ,I_Z jsou momenty setrvačnosti k jednotlivým osám a I je matice setrvačnosti.

Pro jednotlivé momenty setrvačnosti platí

$$I_X = \frac{m}{12}\left(b^2 + h^2\right) \tag{2}$$

$$I_Y = \frac{m}{12}\left(l^2 + h^2\right) \tag{3}$$

$$I_Z = \frac{m}{12}\left(b^2 + l^2\right) \tag{4}$$

kde m je hmotnost, b,l,h jsou rozměry tělesa. obr.2

Definujeme-li obecný souřadný systém, pak poloha a natočení souřadnicové soustavy svázané s trupem robotu je definována šesti zobecněnými souřadnicemi

$$q = (q_1,...,q_6)^T = (tr_X, tr_Y, tr_Z, r_X, r_Y, r_Z) \tag{5}$$

kde q je uspořádaná šestice zobecněných souřadnic, tr_X ,tr_Y ,tr_Z jsou zobecněné souřadnice pro translační pohyb a r_X ,r_Y ,r_Z jsou zobecněné souřadnice pro rotační pohyb.

Trup má tedy šest stupňů volnosti. Tři pro translaci v každé ose a tři pro rotaci v každé ose. Použitím Lagrangeovy rovnice druhého druhu získáme stavové rovnice pro řešení pohybu trupu robotu. Dynamický stav trupu je určen stavovým vektorem

$$\zeta = \left(q^T \dot{q}^T\right)^T \tag{6}$$

kde q je uspořádaná šestice zobecněných souřadnic a dq/dt je uspořádaná šestice derivací zobecněných souřadnic a ζ je stavový vektor.

Lagrangeova funkce je dána rozdílem kinetické a potenciální energie soustavy, máme tedy

$$L = W_T + W_R - W_P \tag{7}$$

kde L je Lagrangeova funkce, W_T je kinetická energie potřebná k translaci tělesa, W_R je kinetická energie potřebná k rotaci tělesa a W_p je potenciální energie soustavy.

Kinetická energie je rozložena do translační energie a rotační energie.

2.1.2. Dynamika nohy

Noha robotu Kráčmera I. Jedná se o dvousegmentovou pantografickou nohu složenou z mnoha částí. Pokud chceme dostupně vyjádřit Lagrangeovu funkci pro jednotlivé nohy tělesa, musíme přijmout zjednodušení.

Obr.3 Noha robotu Kráčmera I. (Obrázek převzat z [3])

Podobně jako u trupu předpokládejme dokonalou tuhost tělesa, hmotnost a konstantní hustotu. K určení pohybových rovnic je zde opět možné použit Lagrangeových rovnic druhého druhu. Stejným postupem jako u trupu dojdeme k podobným rovnicím určujícím pohybové rovnice nohy robotu.

2.2. Modelování soustav

Základní pojmy:

- Model je vždy zjednodušením, abstrakcí reality. Vždy se tedy od reality liší. Model je analogie mezi dvěmi soustavami [18,19].
- Modelování je náhrada zkoumané soustavy soustavou, které říkáme model, jehož cílem je získat informaci o původní soustavě, jejíž je model abstrakcí [18,19].
- Je-li soustava dynamická, pak je model simulátorem. Užívá se však obojího označení[18,19].
- Simulací rozumíme takovou techniku, která má za úkol získat informaci ze simulátoru o původní zkoumané dynamické soustavě [18,19].

Používání modelů vede k zisku (časový, ekonomický apod.) [18,19].

2.2.1. Modelování dynamických vlastností robotu

Z matematického vyjádření dynamických rovnic jednotlivých částí robotu plyne několik závěrů vzhledem k možnosti modelování.

Rovnice byly sestaveny při zjednodušení trupu a nohou robotu, z čehož můžeme usuzovat na chybovost výpočtů. Další nevýhodou je potřeba nového sestavení rovnic při každé změně hmotnosti a tvaru trupu (např.náklad). Sestrojit takové rovnice automaticky je při dnešních možnostech výpočetní techniky nerealizovatelné. Systém rovnic je nutno řešit kontinuálně v průběhu plnění úkolu robotu, je to tudíž nesmírně náročné na výpočetní část robotu.

2.3. Prostředky pro dynamické modelování

K modelování dynamických soustav je možno použít různý software, jako je Matlab/Simulink[25]., Newton Game Physics, Vertex Physics apod.

Software pro modelování dynamických soustav jako je Pro/ENGINEER Mechanica [23] nebo Autodesk Inventor Professional [1] je přesný a relativně rychlý. Nevýhodou je

nepoužitelnost pro real-time modelování z důvodů nemožnosti vstupů v průběhu simulace a další nevýhodou je relativně vysoká cena.

Matematický software typu Matlab/Simulink mají vlastní pracovní prostředí, pomocí něhož se tvoří modely soustav. Takový software má vysoké hardwarové nároky na provoz, proto jej není možné použít pro real-time řízení robotu.

Třetí způsob modelování dynamiky soustav je využití fyzikálních enginů. Jsou to knihovny tříd a funkcí navržených tak, aby simulovaly dynamiku reálných soustav. Fyzikální engine nemají pracovní prostředí a pracuje se s nimi pomocí rozhraní API[5], protože jsou určeny pro podporu dynamiky v softwarových projektech jako jsou počítačové hry, nebo letecké symulátory. Takový přístup dovoluje využít většinu hardwarových prostředků k vlastnímu simulování soustav a jednoduché zahrnutí dynamiky do projektu, což předchozí dvě skupiny software neumožňují.

2.4. Použité prostředky

Simulace byly vytvořeny v prostředí Microsoft Visual Studio (platforma .NET) [20]. Jako programovací jazyk byl zvolen C# a za vizualizační nástroj bylo zvoleno rozhraní DirectX [5, 6, 20].

2.4.1. C# 2.0

Programovací jazyk C# („síšárp“) je považován za vlajkovou loď firmy Microsoft [20]. Jedná se o objektově orientovaný jazyk, který je odvozen z jazyku C++; oproti C++ má odstraněny některé nebezpečné konstrukce a výrazně zvyšuje produktivitu tvorby kódu. Celý kód pro simulaci je napsán v režimu řízeného kódu (managed code)[5,6].

2.4.2. DirectX 9.0

Microsoft DirectX je grafické rozhraní sloužící k obsluze grafických zařízení primárně určených pro operační systém Microsoft Windows[15], je realizováno objektově. Používá se pro tvorbu počítačových her a různých multimediálních a interaktivních aplikacích. DirectX 9.0 obsahuje různé nástroje pro tvorbu 3D prostředí, animací, zvuků aj. DirectX dovoluje přímo obsluhovat grafické karty, a tak plně využít jejich hardwarové možnosti, a je oficiálně podorováno firmou Microsoft na platformě .NET.

Podobné grafické rozhraní je OpenGL [5, 6, 10], které realizováno jako balík funkcí. Pro platformu .NET, existuje několik open-source rozhraní, různé kvality.

3. OPEN DYNAMICS ENGINE

Tato kapitola čerpá z [5,6,7]

Open dynamic engine (ODE) je projekt open-source knihovny funkcí a tříd. Je poskytován zdarma pod licencí GNU Leader General Public License(upouští se se od ní) a BSD-Style License. Účelem této knihovny je simulování dynamiky tuhých těles. Zakladatem projektu ODE je Russel Smith[10,2].

ODE bylo původně vyvinuto jako prostředek k simulaci fyzikálních zákonů v počítačových hrách. V této oblasti se také hojně využívá. V posledních letech se tato knihovna začíná používat i pro řešení technických problémů. Jednou z aplikací, které byly vyvinuty s jejím využitím, je učení neuronových sítí fyzikálním zákonitostem.

Komunita lidí, kteří se věnují vývoji ODE, je rozsáhlá, a proto úpravy v ODE jsou už nepřehledné. To vede k mnohým zmatečně napsaným částem kódu, duplikaci informací v paměti, opomenutí uvolňování paměti a nedůsledně vedené dokumentaci.

Simulace v této diplomové práci byly vytvořeny s ODE ve verzi 0.9.

3.1. Základní vlastnosti ODE

Velká část knihovny je naprogramována v jazyce C, zbytek v jazyce C++. Za účelem pohodlného použití i v jiných programovacích jazycích bylo realizováno několik wrapperů (např.: OdeDotNet, Tao.ODE, PyODE…). ODE neobsahuje zobrazovací nástroj, primárně je však šířen s modulem Drawstuff, který využívá grafické rozhraní OpenGL.

Tato knihovna je programována jako ekvidistantní simulační prostředí. Zadávané i výsledné hodnoty nemají specifický rozměr (např. nikoli 10m/s, jen 10).

ODE je možné zkompilovat ve dvou módech v závislosti na požadované přesnosti. Přesnost se nastavuje volbou typu proměnných, které se budou nadále používat. V tomto případě to jsou (bylo použita kompilace s proměnnou typu System.Single) :

System.Single (float)	$1{,}5*10^{-45}$ až $3{,}4*10^{38}$
System.Double (double)	$5{,}0*10^{-324}1{,}7*10^{308}$

3.2. Wrappery ODE pro .NET

ODE lze používat jako staticky linkovanou knihovnu (lib, so) z jazyků C/C++, nebo jako dynamicky linkovanou knihovnu (dll, a). ODE jako dynamicky linkovanou knihovnu,

lze používat i z jiných programovacích jazyků buď přímo, nebo pomocí "wrapperů". Wrapperů pro ODE, které jsou použitelné pro .NET a jazyk C#, existuje několik, v této práci byly tři nejrozšířenější.

Všechny tyto wrappery jsou tvořeny různými skupinami počítačových nadšenců. Druhou skupinou lidí, kteří se zapojují do vývoje, jsou profesionální programátoři povětšinou počítačových her. Tato skupina je však v menšině. Proto lze v těchto wrapperech najít mnoho chyb nebo naopak nedodělků.

3.2.1. Ode.NET

Tento wrapper je distribuován přímo se zdojovými kódy ODE, ale obsahuje ze všech použitých wrapperů nejvíce chyb. Je realizován pomocí statické třídy, která zahrnuje většinu funkcí z ODE. Při jeho použití se nepodařilo uspokojivě naprogramovat kolizi těles. Simulace s tímto wrapperem nebyly provedeny. Přestože se momentálně chyby v něm odlaďují, daří se tvořit jeho kód přehledně a uspořádaně[7].

Byla použita verze 0.9 a 0.8.

3.2.2. Tao.ODE

Tento wrapper je součástí Tao Frameworku[8], což je multiplatformní kolekce knihoven určená k vývoji na .NET a Mono platformách. Součástí distribuce tohotu frameworku je i příklad základního využití ODE[8,9]. Tento wrapper je realizován pomocích statických tříd. Pří použití tohoto wrapperu se nepodařilo uspokojivě implementovat kolizi těles. Kolize byli vždy realizovány jako vektnutí, což je pro simulaci nepoužitelné.

Byly využity verze 2.0, 2.0 RC, 2.1.0.

3.2.3. OdeDotNet

Tento wrapper využívá wrapperu Tao.Ode a zachovává principy objektově orientovaného programování s využitím nestatických funkcí a tříd.[21]. Je ze všech tří použitých wrapperů nejmladší a existuje pouze první vydání. Přesto je tento wrapper hojně používám.

Byla použita verze OdeDotNet 0.1.0.

4. Struktura a použití ODE

Tato kapitola čeprá z [5,3,4,2,13]

ODE je rozděleno na dvě části. První část je „fyzikální“ svět, jehož smyslem je aplikovat síly a momenty na tělesa, tedy řešit dynamiku soustavy. Druhou částí je kolizní svět, jehož účelem je udržovat geometrii těles za účelem zjišťování případných kolizí a na jejich základě definovat vazby na tělesech v dynamickém světě [4,2].

4.1. Reprezentace těles v ODE

Vlastnosti tělesa jsou rozděleny na dvě části, a to hmotnou (použitá v DYNAMICS ENGINE) a geometrickou (použitá v COLLISION DETECTION ENGINE). Tělesa jsou definována jako dokonale tuhá.

Vlastnosti tělesa jako hustota, tvar, rozměry a rozložení hmoty v tělese (definováno pomocí kvadratických momentů k jednotlivým osám) jsou reprezentovány jako hmotný bod s pozicí a orientací v prostoru s přihlédnutím k prostorovému rozložení hmoty v tělese. I když se při vytváření těles v dynamickém světě nastavuje jejich tvar, je to jen kvůli dynamickým vlastnostem tělesa.

Ke zjišťování kolizí těles se používá vytváření tzv. geometrie těles a k nim se přiřazují tělesa v dynamickém světě.

4.2. Reprezentace tření a gravitačního zrychlení v ODE

Tření v dynamickém světě je aproximováno jako statické tření s konstantní velikostí podle

$$f_T = C_T * f_N \tag{8}$$

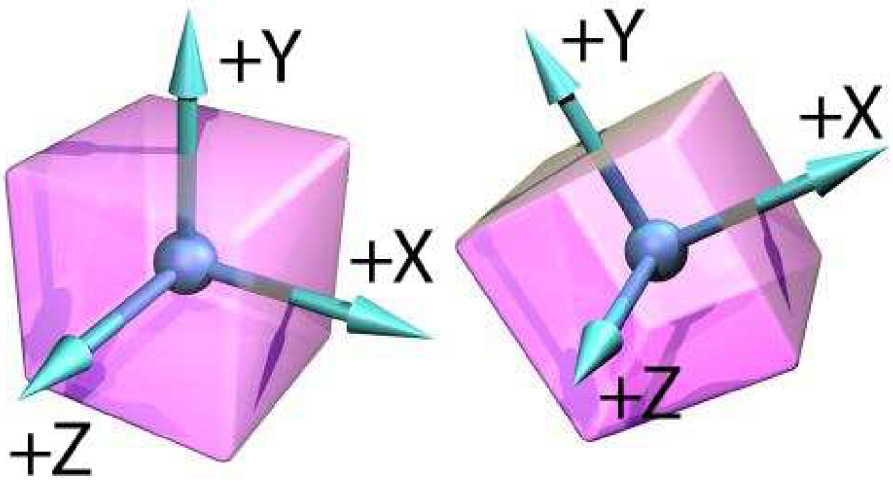
kde f_T je výsledná třecí síla, C_T je konstanta tření (nastavitelná) a f_N je normálová síla úměrná hmotnosti tělesa.

Aplikaci tření lze v ODE provést více způsoby podle různých kritérií. Hojně používanou metodou je pyramidální tření. Úměrně rychlosti je násobeno tření tak, aby se mohl aproximovat přechod mezi dynamickým a statickým třením.

Gravitační zrychlení je definováno jako statická síla působící na těleso v dynamickém světě.

4.3. Souřadný systém v ODE

V ODE může být definováno více souřadných systémů; hlavní souřadný systém je však neměnný. Nelze ho otočit ani posunout. Každý vytvořený objekt má svůj souřadný systém, který se s ním otáčí. Pozice a orientace v dynamickém světě ODE je možné zadávat jak vůči hlavnímu souřadnému systému, tak vůči lokálnímu souřadnému systému, který je svázán s tělesem.



Obr.4 Souřadný systém v ODE. Základní souřadný systém odpovídá obrázku vlevo. (Obrázek převzat a upraven z [2])

4.4. Datové typy v ODE

Datové typy v ODE slouží k vyjádření pozice a orientace tělesa v prostoru.

4.4.1. Datový typ Vector3, Vector4

Vector3,Vector4 jsou obsaženy ve jmenném prostoru OdeDotNet.

Jedná se o vyjádření pozice v třírozměrném prostoru.

```
Vector3 = [X,Y,Z];
Vector4 = [X,Y,Z,1.0];
```

Kde X, Y, Z jsou vzdálenosti od počátku hlavního souřadného systému v jednotlivých osách.

4.4.2. Datový typ Quaternion

Quaternion je součástí jmenného prostoru OdeDotNet.

Quaternion vyjadřuje orientaci tělesa v třírozměrném prostoru.

```
Queternion = [W,I,J,K];
```

Kde I, J, K jsou vzdálenosti v jednotlivých osách k bodu na hyperkouli a W je skalár navržený tak, že platí:

$$1 = \sqrt[2]{i^2 + j^2 + k^2 + w^2} \tag{9}$$

Queternion je představován jako čtyřvektor, kde jednotlivá souřadnice odpovídá vzdálenosti od počátku na povrchu čtyřrozměrné hyperkoule, což vzhledem k rovnici (9) neplatí. Čtvrtá souřadnice je uplatněna z důvodu regulérního násobení maticemi 4x4. Quaterniony jsou stejně jako matice nekomutativní.

4.4.3. Datový typ Matrix3, Matrix4

Matrix3, Matrix4 jsou součástí jmenného prostoru OdeDotNet.

Matice Matrix3 je řádu 3x3 a vyjadřuje orientaci tělesa ve třírozměrném prostoru.

Matice Matrix4 vyjadřuje pozici (translaci od hlavní souřadné osy) i orientaci.

$$M = \begin{bmatrix} R & R & R & T_X \\ R & R & R & T_Y \\ R & R & R & T_Z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{10}$$

Kde R značí rotaci a T translaci.

5. Dynamický svět

Tato kapitola čerpá z [2]

Základní částí ODE je RIGID BODY DYNAMICS ENGINE nebo také jen DYNAMICS ENGINE.

Dynamický svět je v důsledku hierarchie objektů, kde jako rodičovský objekt je objekt World. Tělesa a spoje jsou vytvořené objekty podřízené rodičovskému objektu. Dynamický svět není rodičovský objekt ve smyslu objektově orientovaného programování.

5.1. Třída World

Třída World je základní prvek ODE. Je kontejnerem v něm vytvořených těles. Simulace může mít více objektů World, ty však nemohou interagovat.

5.1.1. Základní práce s třídou World

Třída World je součástí jmenného prostoru OdeDotNet.

Každá simulace musí začínat vytvořením instance třídy World.

```
World Dynamicky_svet = new World();
```

Hlavní vlastnosti objektu World jsou (nastavení přesnosti je na float)

- Gravitační zrychlení nastavené vektorem zrychlení

```
Dynamicky_svet.SetGravity(X, Y, Z);
```

kde X,Y,Z je hodnota (typu float) zrychlení v jednotlivých osách.

- ERP

```
Dynamicky_svet.Erp;
```

hodnota v rozmezí 0.0 až 1.0 (typ float).

- CFM

```
Dynamicky_svet.Cfm;
```

hodnota v rozmezí 0.0 až 1.0 (typ float).

- Autovyřazení vzdálených a neinteragujících objektů

```
Dynamicky_svet.AutoDisableFlag;
```

nabývá hodnot true (povoleno) nebo false (zakázáno).

5.2. Třída Body

Třída Body reprezentuje tuhé těleso a ty jeho vlastnosti, které používá dynamický svět. Je objektem patřícím do třídy World, do kterého se při jeho vytvoření zařazuje. Časově stálé jsou vlastnosti jako hmotnost, pozice těžiště a kvadratické momenty. Změna těchto hodnot v průběhu simulace může vést k nestabilitě výpočtu.

Tělesa mohou mít mnoho tvarů, podporovány jsou např kvádr, koule atd. Lze ale také definovat vlastní tvar tělesa a následně ho načíst do ODE ze souboru. Tato nová funkce ale ještě není plně implementována.

Tělesa mohou být vytvořena jako prostorově přemístitelná nebo prostorově nepřemístitelná. Prostorově přemístitelné těleso může být např koule. Nepřemístitelný objekt je třeba podložka nebo stěna.

5.2.1. Základní práce s třídou Body

Třída Body je součástí jmenného prostoru OdeDotNet.

Pro vytvoření tělesa (vytvoření instance třídy)

```
Body Teleso = new Body(Dynamicky_svet);
```

- Nastavení pozice tělesa v jednotlivých osách

```
Teleso.X;
```

vzdálenost referenčního bodu tělesa v ose X (typ float) od počátku (obdobně v ostatních osách).

- Nastavení pozice tělesa pomocí vlastnosti Position

```
Teleso.Position = Pozice_telesa;
```

vektor typu Vector3 představující pozici tělesa.

- Nastavení orientace tělesa

```
Teleso.Rotation;
```

matice typu Matrix3, která reprezentuje orientaci tělesa.

Vytvoření hmoty

```
Mass Hmota_telesa = new Mass();
```

- Definování hmoty (typ kvádr)

```
Hmota_telesa.SetBox(hustota, X, Y, Z);
```

kde X,Y,Z je pozice v prostoru a hustota označuje hustotu tělesa (kladné číslo typu float).

- Přiřazení hmoty tělesa tělesu

```
Teleso.Mass = Hmota_telesa;
```

- Aplikace síly na těleso

```
Teleso.AddForce(X,Y,Z);
```

kde X,Y,Z vyjadřuje sílu působící v dané ose na těleso (typu float).

- Aplikace momentu na těleso

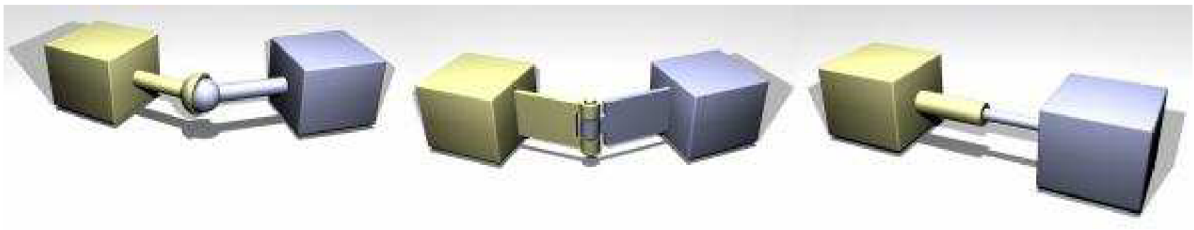
```
Teleso.AddTorqueForce(X,Y,Z);
```

kde X,Y,Z vyjadřuje moment působící v dané ose na těleso (typu float).

5.3. Třída Joint

Třída Joint představuje vazbu. Každý vytvořený spoj je třeba připojit k tělesům nebo k podkladu. Stejně jako u dynamického světa lze i pro vazby nastavit parametry ERP a CMF.

ODE podporuje mnoho typů vazeb.



Obr.5 Základní typy podporovaných vazeb. (Obrázek převzat z [4])

Možnou vazbou je také motor, který může aplikovat moment na těleso v dynamickém světě.

5.3.1. Základy práce s třídou Joint

Třída Joint je součástí jmenného prostoru OdeDotNet.Joints.

Vytvoření vazby (po vytvoření není připojena k žádnému tělesu)

```
HingeJoint Vazba = new HingeJoint(Dynamicky_svet, Grupa_vazeb);
```

- Nastavení těles, které vazba spojuje

```
Vazba.Attach(Teleso_1,Teleso_2);
```

- Nastavení pozice vazby

```
Vazba.Anchor = Vector3;
```

kde Vector3 představuje pozici vazby.

- Nastavení, ve které ose je vazba aktivní (pro rotační vazby)

```
Vazba.Axis.X;
```

hodnota 0.0 (neaktivní v této ose) nebo 1.0 (aktivní v této ose).(typ float) Tato vazba může mít nastavenu jen jednu aktivní osu, ve které umožňuje rotaci. Nastavení více os rotace vede k nestabilitě simulace.

5.4. Třída JointGroup

Třída JointGroup je kontejnerem vazeb.

5.4.1. Základy práce s třídou JointGroup

Třída JointGroup je součástí jmenného prostoru OdeDotNet.Joints.

Vytvoření grupy vazeb (vytvoření instance třídy)

```
JointGroup Grupa_vazeb = new JointGroup(hodnota);
```

kde hodnota je omezení počtu vazeb (kladné celé číslo typu float). V nynější úpravě ODE toto číslo nemá žadný smysl. Nebere se na něj zřetel.

5.5. ERP – error reduction parametr

Ke spojení dvou těles slouží vazba (objekt typu Joint), která je buď definována uživatelem nebo kolizní funkcí. V těchto případech musí být relativní poloha a orientace těles vůči sobě dána tak, aby do sebe nezasahovala, nebo aby se nevytvořil spoj tam, kde tělesa spolu nekolidují. Další chyby mohou být třeba změna osy rotace u jednoosé rotační vazby. Tyto chyby mají za následek buď nepřesné kolidování dvou těles nebo jisté povolování vzájemných vazeb, tedy jisté klouzání.

ERP parametr nabývá hodnot 0 nebo až 1. Při nastavení 0 je korekce těchto vazebních sil vynulována, při nastavení 1 je maximální. Doporučená hodnota je v rozmezí 0.6 až 1. Defaultně je nastavena na hodnotu 0.2.

Korekce vazeb probíhá tak, že se na obě spojená tělesa aplikuje síla, jejíž velikost a směr, jež jsou propočítávány pro každý simulační krok, nedovolí propojeným tělesům přerušit vazbu mezi nimi.

Nevhodně volený parametr ERP nepříznivě ovlivňuje simulaci. Parametr ERP je bezrozměrný.

5.6. CFM – constraint force mixing

Všechny vazby, definované v ODE, jsou řešeny jako absolutně tuhé. Pokud takovou vazbu definujeme, nemůže dojít k průniku tělesa do tělesa. Za účelem zmenšení tuhosti materiálu (tedy změkčení materiálu) je definován parametr CFM. Vhodně nastavený parametr CFM dovoluje průnik těles navzájem. Jelikož i dotyk těles je tvořen vazbou typu Joint, je průnik těles tvořen jako uměle vytvořená chyba ve vazbách. Doporučuje se používat CMF až po odladění simulace, protože používání tohoto parametru ovlivňuje vazby stejně jako ERP. Hodnoty se používají v rozmezí -1 až 1. Avšak záporné hodnoty jen nepříznivě ovlivňují simulaci. Doporučená hodnota je $1*10^{-7}$. I když je v manuálu ODE[2] parametr CMF definován matematicky, je možné ho považovat za bezrozměrný.

5.7. Použití ERP a CFM v simulacích

Parametry ERP a CMF můžeme nastavit pro celý dynamický svět, ale můžeme je také nastavit pro jednotlivé vazby (a to třeba pro každou jinak). Hodnoty těchto parametrů mohou být konstantní po celou simulaci nebo se mohou měnit. Toho můžeme využít pro modelování různě tuhých materiálů. Tyto proměnlivé vazby lze použít jako tlumení nebo pomocí nich simulovat elastičnost. V některých literaturách se uvádí možnost vytvořit pomocí těchto parametrů simulaci pružiny. To možné je (parametry mohou být závislé na poloze nebo čase), avšak mnohem jednodušší a spolehlivější je modelovat pružinu jako sílu, závislou na poloze tělesa.

6. Kolizní svět

Tato kapitola čerpá z [2,4]

6.1. Kolize objektů

Část ODE, která zpracovává kolize objektů, se jmenuje COLLISION DETECTION ENGINE. Kolizní část ODE není nutné zapojit do simulace. Lze jej i doprogamovat samostatně.

6.2. Třída Geom

Třída Geom reprezentuje geometrii tělesa. Tato geometrie se musí přiřadit objektu, pro který je vytvořena. Bez přiřazení je geometrie nehmotná a tudíž nemá vliv ani na těleso samotné, ani na okolí. Pro správnou funkčnost je třeba dbát na odpovídající tvar a rozměry tělesa a geometrie k němu přiřazené.

ODE podporuje jak základní tvary těles jako např. koule, kvádr, kapsle ale také obsahuje možnost načíst mesh-síť (soubor bodů reprezentující vrcholy tělesa) tělesa ze souboru.

6.2.1. Základní práce objektu třídou Geom

Třída Geom je součástí jmenného prostoru OdeDotNet.Geometry.

Pro vytvoření geometrie tělesa (vytvoření instance třídy) (typu kvádr) a jeho zařazení do kolizního světa

```
Box Geometrie_telesa = Kolizni_svet.CreateBox(X,Y,Z);
```

kde X,Y,Z reprezentují rozměry tělesa (typ float).

- Přiřazení geometrie tělesu v dynamickém světě

```
Geometrie_telesa.Body = Teleso;
```

6.3. Detekce kolizí

Ke zjišťování kolizí objektů je možno využít dvou způsobů; oba způsoby však potřebují zadanou geometrii těles a její přiřazení k tělesu v dynamickém světě.

6.3.1. Funkce Collide

Použití funkce Collide spočívá ve volání této funkce v každém kroku simulace. Po jejím zavolání je zkontrolována možnost kolize způsobem „každý s každým“. Z tohoto

přístupu plyne i její výpočtová náročnost, která při M objektů je $O(M^2)$. Je implementována jako Callback funkce.

Při použití některých wrapperů (např Tao.Ode) je nemožné tuto funkci použít bez využití kolizního prostoru Space.

6.3.2. Využití prostoru Space

Druhý způsob zjišťování kolizí těles je vytvoření kolizního prostoru Space, ve kterém jsou uchovávány geometrie těles. V každém „časovém“ kroku simulace lze zavolat funkci SpaceCollide, která hledá potenciálně kolidující tělesa. Ta tělesa, která jsou vyhodnocena jako potenciálně kolidující, jsou předána funkci Collide. Výpočetní náročnost funkce je výrazně menší než v případě zavolání Collide. Explicitně se však vyjádřit nedá; závisí totiž na rozmístění těles v kolizním prostoru.

Pokud máme v simulaci vytvořeno více kolizních prostorů typu Space, funkce prohledání kolizí těles z jiných prostorů je SpaceCollide2. Kolize mezi prostory je tedy možná. Možnost definovat více kolizních prostorů je výhodná, pokud máme více těles vzdálených od sebe a nepočítáme s jejich kolizí.

6.4. Kolizní prostor Space

Třída Space je naprogramována jako kontejner obsahující geometrii těles. Simulace mohou mít mnoho prostorů typu Space.

Existuje více druhů kolizních prostorů.

6.4.1. Druhy kolizních prostorů

V ODE lze použít tři základní typy kolizních prostorů; liší se použitím různých algoritmů k vyhledávání možných kolizí těles.

- Simple space - kontrola kolizí probíhá kontrolou všech možných párů těles užitím dat o jejich geometrii a pozici. Je preferována pro simulace s malým počtem těles.
- Multi-resolution hash table space – rozděluje kolizní prostor na třídimenzionální tabulku a zjišťuje, které těleso obsahuje tu kterou buňku tabulky. Je paměťově náročný, ale časově rychlejší než Simple space.
- Quadtree space – užívá předalokovaný hierarchický mřížkový strom. Je nejrychlejší pro simulace s více tělesy. Je paměťově nejnáročnější.

6.4.2. Základní práce s třídou Space

Simulace nemusí obsahovat kolizní prostor, naopak je možné si kolizní engine doprogramovat samostatně.

Vytvoření kolizního prostoru (vytvoření instance třídy)

```
Space Kolizni_prostor = new Space();
```

Hlavní vlastnosti objektu Space jsou

- Přidání nové geometrie (objektu) do kolizního prostoru

```
Kolizni_prostor.Add(Geometrie_telesa);
```

- Odebrání geometrie z kolizního prostoru (nebude kolidovat)

```
Kolizni_protor.Remove(Geometrie_telesa);
```

7. Typický kód simulace

Tato kapitola čerpá z [2,4,5,6,22]

Typický kód simulace může být upraven podle potřeby aplikace; je ovšem určena posloupnost kroků taková, aby simulace nevytvářela duplikované vazby a tělesa a nezatěžovala paměť a výpočetní výkon počítače víc než je nutné apod.

Posloupnost kroků tedy může být:

- Vytvořit dynamický svět
- Vytvořit tělesa, nastavit jejich pozici a rotaci v dynamickém světě
- Vytvořit vazby a pospojovat jimi tělesa v dynamickém světě
- Nastavit parametry vazeb
- Vytvořit kolizní svět a geometrii těles
- Vytvořit grupy vazeb a zařadit (není povinné) do nich vazby
- Aplikace sil a momentů na tělesa (Působí jen v prvním kroku)
- Simulační kroky ve formě smyčky
- Destrukce dynamického a kolizního světa (není třeba v platformě .NET)

7.1. Simulační krok ve formě smyčky

- Aplikace sil a momentů
- Nastavit nebo upravit vazby mezi tělesy
- Vyvolat kontrolu kolizí
- Vyvolat časový simulační krok
- Vymazat všechny spoje v grupě vazeb

V simulačním kroku můžeme zařazovat nebo vyřazovat tělesa ze simulace, lze také upravovat jejich rozměry apod.

7.2. Časový krok

Časový krok v simulaci je klíčový prvek, kterým nastavujeme a posouváme simulaci v čase vpřed.

Čas je ODE také ekvidistatntní. Zvětšíme-li těleso desetkrát a necháme-li časový krok konstantní, pak při stejných zadaných hodnotách je simulovaná doba desetkrát menší.

7.2.1. Funkce Step

Funkce Step je součástí jmenného prostoru OdeDotNet.

```
Dynamicky_svet.Step(hodnota);
```

kde hodnota (typ float) označuje „časový krok“ simulace. Čím jemnější krok, tím menší je chybovost simulace. Nevhodně velký „časový krok“ má za následek velkou chybovost až nestabilitu simulace.

Fuknce Step používá metodu „velkých matic“ s časovou náročností úměrnou m^3 a paměťovou náročností úměrnou m^2, kde m je řádek v matici a odpovídá odebranému stupni volnosti (rozumí se všech těles). Tato metoda je výpočtově náročná, ale je ze všech krokovacích funkcí v ODE nejpřesnější a nejstabilnější.

7.2.2. Fuknce QuickStep

Funkce QuickStep je součástí jmenného prostoru OdeDotNet.

```
Dynamicky_svet.QuickStep(hodnota);
```

kde hodnota (typu float) označuje časový krok simulace.

Funkce QuickStep používá iterativní metodu s časovou náročností úměrnou m*N a paměťovou náročností úměrnou m, kde m je počet odebraných stupňů volnosti (všech těles) a N je počet iterací. Tato metoda je rychlejší ale zároveň nepřesnější než funkce Step.

Pokud je tato krokovací funkce použita v modelování blízko-singulárních soustav, je pravděpodobné, že simulace „uvízne“ v lokálním minimu. Je zde také větší pravděpodobnost výskytu nestabilního chování modelů.

8. Vytvořené softwarové řešení pro realizaci simulací

Při návrhu softwaru se vycházelo z několika předpokladů a podmínek.

Pro další práci byli použity prostředky, které jsou popsány v kapitole 2 a 3.

Je nutné sledovat chování modelu v ODE. U většiny modelů je velmi obtížné a časově náročné sledovat jejich chování pouze na základě grafů průběhů jednotlivých poloh, rychlostí a orientací v čase, bylo realizováno grafické prostředí zobrazující prostor simulace a samotná simulovaná tělesa.

Pro možnost dalšího zpracování průběhů poloh jednotlivých těles během simulace, byl realizován zápis těchto poloh do textového souboru ve formátu programu MATLAB.

8.1. Struktura kódu programu

Struktura kódu se dělí na inicializaci a samotnou simulaci. Kód byl navržen tak, aby se daná simulace dala spouštět opakovaně s proměnnou rychlostí zobrazování simulace, různou dobou simulace a s různým simulačním krokem.

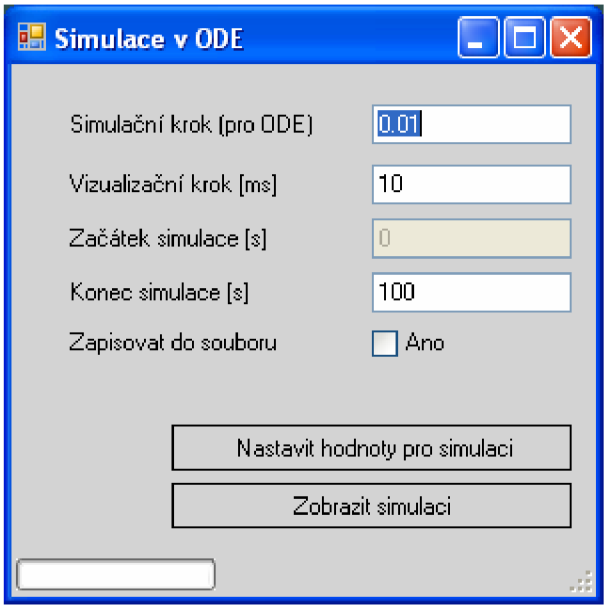
- Zobrazení komunikačního formuláře a inicializace formuláře k zobrazení simulace
- Nastavení hodnot a inicializace dynamického světa
- Spuštění a zobrazení celé simulace
- Ukončení simulace, zápis do souboru (pokud je nastaven) a zaktivování komunikačního formuláře

8.2. Komunikační formulář (úvodní formulář)

Toto okno se zobrazí vždy po startu aplikace a po skončení simulace. Obsahuje základní ovládací prvky simulace.

- Simulační krok – označuje „časový krok“ simulace. Zůstává konstantní po celý čas simulace. Je zadáván do funkce Step.
- Vizualizační krok – nastavuje rychlost invalidace okna zobrazujícího simulaci,t tedy jak rychle se bude simulace zobrazovat. Rychlost zobrazování je ovlivněna výkonností počítače.
- Začátek simulace – zůstává stejný (hodnota 0).
- Konec simulace – nastavuje čas (ekvidistantní), kdy se má simulace skončit.

- Zapisovat do souboru – pokud je povolena, zapisuje data do souboru data.mat, který může být přečten programem MATLAB. Zapisuje se pozice všech těles. Orientace zapisována není z důvodu obtížného čtení dat ze souboru (ukládají se matice).
- Nastavit hodnoty pro simulaci – nastaví hodnoty vložené uživatelem do programu. Je nutné zvolit při změně hodnot. Nově zadané hodnoty se nevkládají automaticky.
- Zobrazit simulaci – nastaví simulační okno jako aktivní a spustí simulaci.
- Běžec – Zobrazuje průběh simulace.



Obr.6 Komunikační formulář

8.3. Simulační okno – grafické zobrazení simulace

Toto okno zobrazuje stav simulace. Po spuštění simulace se toto okno nastaví jako aktivní a zůstane jím až do konce simulace (pokud nezasáhne uživatel). Velikost okna je 800x600 pixelů.

V kódu není naprogramováno ošetření ztráty zobrazovacího média apod. (nebylo to účelem diplomové práce).

8.3.1. Ovládání náhledu

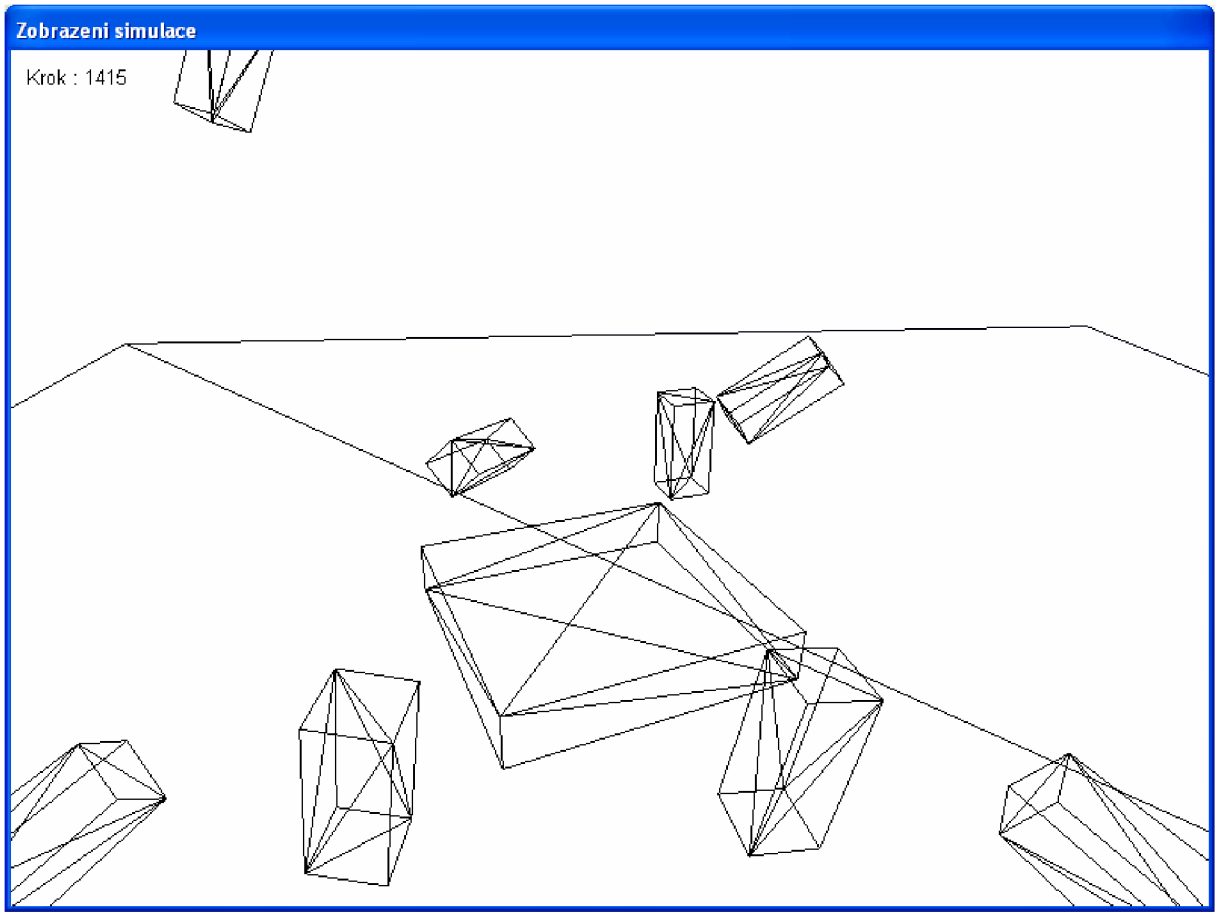
Náhled v simulačním okně je možné měnit podle potřeby. Bod, odkud je na simulaci nahlíženo, je nazván kamera.

Omezení pohybu kamery je:

- Nelze vidět podlahu ze spodní části obrazovky
- Nelze nastavit kameru do bodu kolmého k podlaze (bodem se myslí místo, okolo kterého kamera rotuje)
- Kamera rotuje kolem středu podlahy
- Nelze vidět příliš vzdálené předměty

Ovládání kamery je

- Šipka doleva, doprava – rotace kolem středu podlahy v rovině rovnoběžné s podlahou
- Šipka nahoru, dolů – rotace kolem středu podlahy v rovině kolmé na rovinu podlahy
- Num-, Num+ - vzdálení, přiblížení ke středu rotace



Obr.7 Simulační okno

8.4. Zobrazování těles pomocí DirectX

Tělesa, která jsou obsažena v ODE, mají své pozice a orientace, které se v každém simulačním kroku mění. Zobrazovat tyto průběhy pomocí programu MATLAB jako grafy je značně nepřehledné a výpočetně náročné. Bylo zvoleno vytvoření grafického okna zobrazujícího všechna definovaná tělesa. Je navrženo přímo na zobrazení robotu (devět těles typu kvádr) a podložky (plocha). Vazby nejsou zobrazeny (nemají polohy ani orientace).

DirectX je souhrn dynamicky linkovaných knihoven dovolující obsluhovat grafický hardware. Ovládání grafického hardware podléhá takovým pravidlům, aby nedošlo k nestabilitě systému. Zobrazovací aplikace dynamicky zapisuje polohy vrcholů těles do bufferů grafického hardware. Těmito buffery jsou Vertex buffer a Index buffer. Vertex buffer uchovává prostorové rozmístění, barvu atd. jednotlivých zobrazovaných bodů. Tělesa jsou zobrazena jako složené trojúhelníky, náležitost jednotlivých vrcholů do trojúhelníku uchovává Index buffer.

Pseudokód zobrazování (je volán po každém simulačním kroku):

- Načti pozici a orientaci všech referenčních bodů těles z ODE –pozice a orientace těžišť těles
- Spočítej pozici bodů (vrcholů) podle rozměrů těles
- Transformuj vrcholy těles podle orientace těžišť
- Transformuj vrcholy těles podle pozice těžišť
- Ulož souřadnice do Vertex bufferu
- Spoj jednotlivé vrcholy do trojúhelníků
- Zobraz nové pozice těles invalidací zobrazovaného okna (projektivní prostor)

Výsledná transformace je[22]:

$$A = (Z_1(Z_2)) \tag{11}$$

kde Z_1 je transformace vrcholů náležících danému těžišti o vektor z bodu (0,0,0) do bodu, kde leží těžiště tělesa a Z_2 je transformace vrcholů náležících danému těžišti v závislosti na orientaci těžiště tělesa. Teferenčním bodem tělesa je vždy těžiště tělesa (zároveň geometrický střed tělesa).

Tělesa jsou zobrazována jako drátové modely.

9. Simulace triviálních soustav

Tato kapitola čerpá ze zdrojů [2,16,17]

Simulace spočívá ve volném spuštění dvou těles různých rozměrů, přičemž v prvním simulačním kroku působí na obě tělesa stejná síla.

Těleso je kvádr. Za referenční bod tělesa je brán střed (těžiště) kvádru. Těleso se pohybuje v rovině XY. Doba simulace je 80s.

Konstantní parametry simulace:

- ERP je 1
- CMF je$3*10^{-12}$
- Gravitační zrychlení je definováno vektorem [0, -9.81, 0]
- 7850 kg/m^3 – hustota uhlíkové oceli

Rozměry těles jsou:

- 50x50x50 [mm] – Těleso 1
- 50x100x50 [mm] – Těleso 2

V prvním kroku simulace působí na obě tělesa stejná síla [17][16]:

$$F_Z = 1*10^{11} N \tag{12}$$

$$F_Y = 2*10^{13} N \tag{13}$$

Pokud budeme uvažovat impuls síly, je definován jako:

$$\vec{J} = \int \vec{F}(t)dt \tag{14}$$

kde **J** je impuls síly, **F** je síla působící na těleso.

Pak je-li t nekonečně malé a F nekonečně velké, pak mluvíme o Diracově pulsu nebo Diracově impulsu. Takový impuls síly je právě zaváděn do soustav v simulacích. Proto je síla v neúměrně velkých rozměrech vůči hmotnosti a rozměrům tělesa, na které síla působí. Ve výpočtech na počítači nelze použít hodnoty nekonečna. Lze ale zavést hodnotu „velkou“ (násobky 10^{12} apod.).

Změna hybnosti je rovna součtu všech působích sil:

$$\frac{d\vec{p}}{dt} = \sum \vec{F}_{EXT} \tag{15}$$

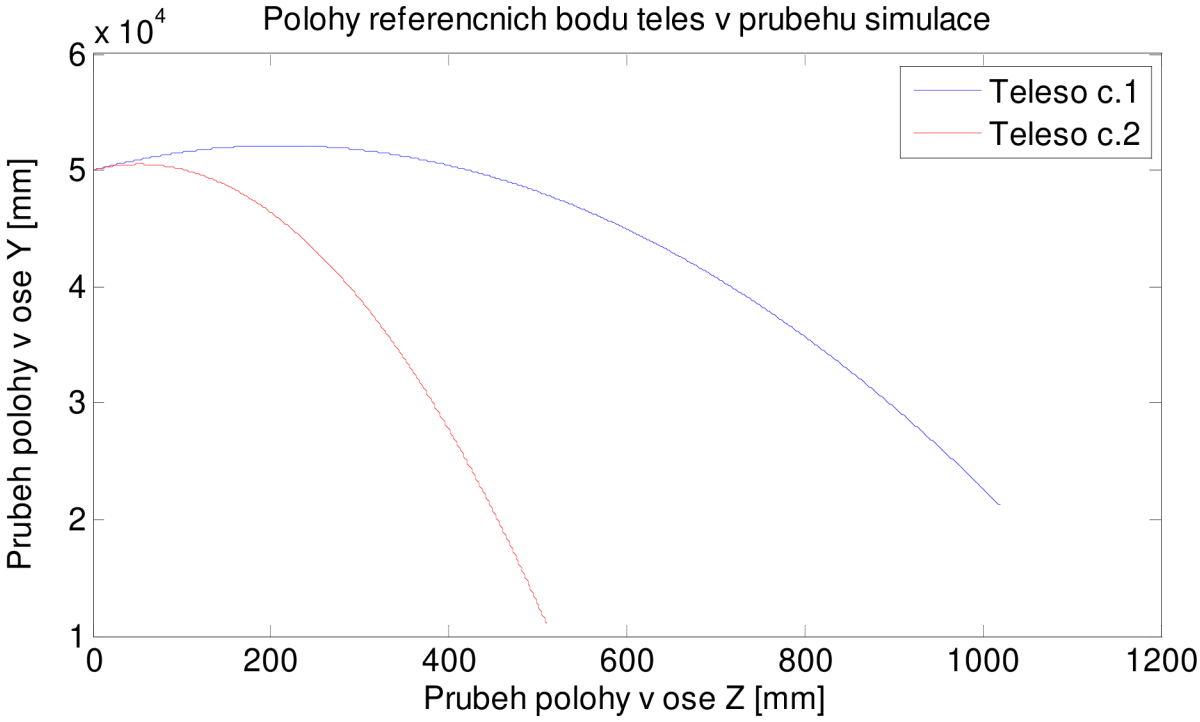
kde **p** je hybnost tělesa a **F** je působící síla.

Hybnost může být definována jako:

$$\vec{p} = m\vec{v} \tag{16}$$

kde **p** je hybnost tělesa, m je hmotnost tělesa a **v** je rychlost tělesa.

Pak, necháme-li působit na těleso impuls síly v ODE, mu udělíme okamžitou rychlost úměrnou aplikované síle a hmotnosti tělesa.



Obr.8 Graf průběhu polohy referenčních bodů těles

Poloha referenčního bodu v jednotlivých osách odpovídá:

$$x = tv_0 \cos\alpha \tag{17}$$

$$y = -\frac{1}{2}gt^2 + tv_0 \sin\alpha \tag{18}$$

kde x, y značí vzdálenost od počátku v jednotlivých osách, g je gravitační zrychelní, t je čas, v_0 je počáteční rychlost a α je elevační úhel. Neuvažujeme disipaci.

Z rovnic (15),(16) plyne, že změna rychlosti je úměrná působícím externím silám a hmotnosti tělesa. Proto lze simulovat průběh trajektorie pomocí rovnic (17),(18). Průběh trajektorie tělesa (modelován pomocí programu MATLAB) je znázorněn v příloze C.

Z obr.8 je patrné, že chování těles při simulaci je podle očekávání odpovídající reálnému chování v konzervativním systému, kde jedinou působící silou je síl gravitační.

10. Simulace netriviálních soustav

K ověření základní funkčnosti dynamického světa a kolizního prostoru ODE bylo zvoleno několik netriviálních případů. V těchto simulacích se také ověřuje vliv různých parametrů na chování modelované soustavy.

10.1. Simulace tělesa dopadajícího na podložku

Simulace ověřuje funkčnost dynamického světa a kolizního prostoru a spočívá ve volném spouštění tělesa na podložku.

Stejná simulace je opakována s různými parametry, přičemž se vždy mění jen jeden z nich a ostatní zůstávají konstantní.

Těleso má tvar koule o poloměru 8 mm. Za referenční bod tělesa je brán střed (těžiště) koule. Těleso se pohybuje po přímce, hodnoty poloh referenčního bodu tělesa v osách X a Z zůstávají v průběhu simulace nulové.

10.1.1. Vliv měnící se hustoty tělesa

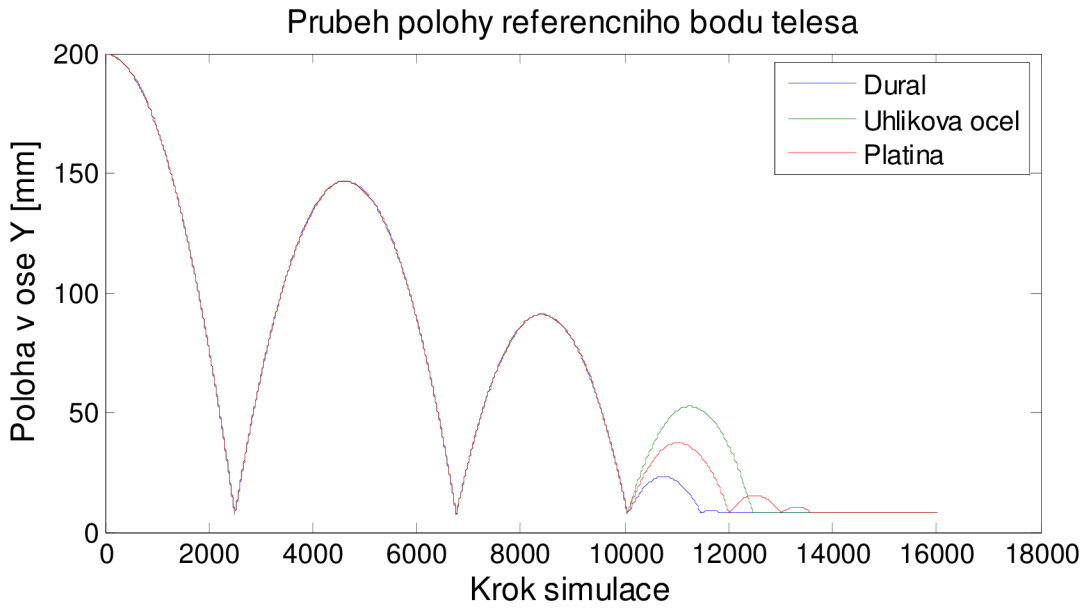
Konstantní parametry simulace:

- ERP je 1
- CMF je 0
- Gravitační zrychlení je definováno vektorem [0, -9.81, 0]

Proměnnou v této simulaci byla hustota tělesa. Ta nabývala hodnot

- 21450 kg/m^3 – hustota platiny (zvolena pro velkou hustotu)
- 7850 kg/m^3 – hustota uhlíkové oceli
- 2800 kg/m^3 – hustota duralu

Modelované těleso nemá materiálové vlastnosti reálných těles, jen jejich hustotu. Vlastnosti jako tuhost, vodivost a jiné nejsou zahrnuty.



Obr.9 Graf průběhu polohy referenčního bodu

10.1.2. Vliv měnící se ERP dynamického světa

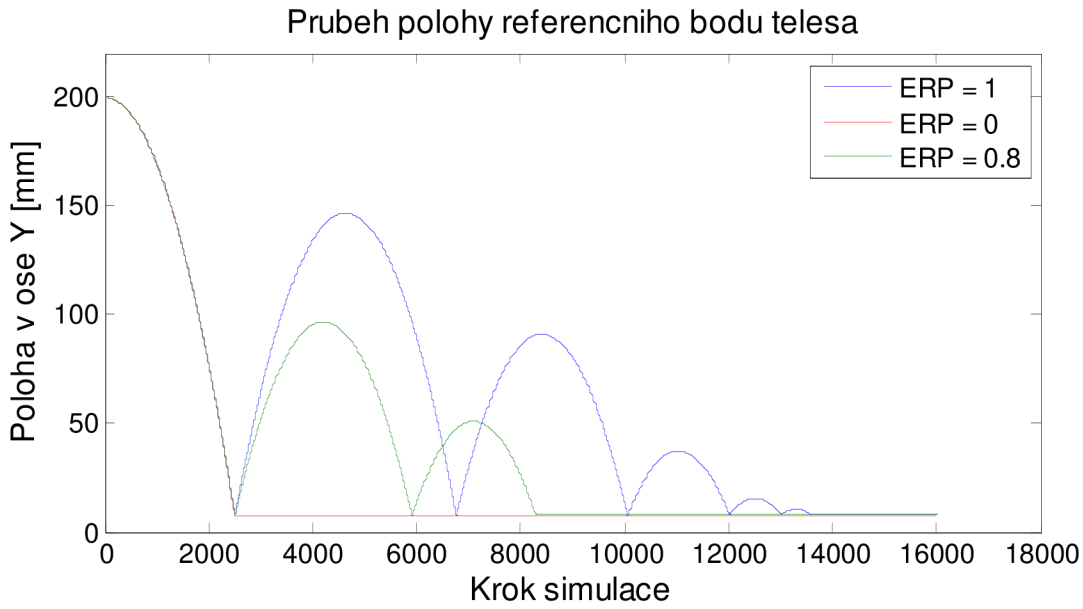
V této simulaci se mění ERP dynamického světa.

Konstantní parametry simulace:

- hustota je 21450 kg/m^3 – hustota platiny
- CFM je 0
- Gravitační zrychlení je definováno vektorem [0, -9.81, 0]

Proměnnou v této simulaci byl parametr ERP. Nabýval hodnot

- ERP = 0.8
- ERP = 1
- ERP = 0



Obr.10 Graf průběhu polohy referenčního bodu

10.1.3. Vliv měnící se CFM dynamického světa

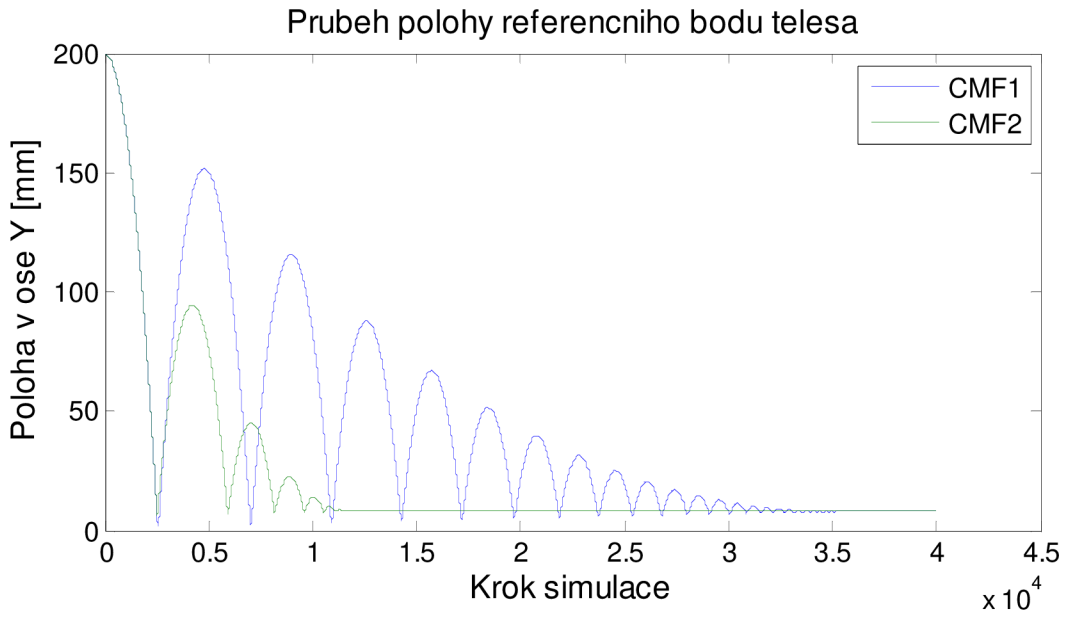
V této simulaci se mění CMF dynamického světa.

Konstantní parametry simulace:

- hustota je 21450 kg/m^3 - platina
- ERP je 1
- Gravitační zrychlení je definováno vektorem [0, -9.81, 0]

Proměnnou v této simulaci byl parametr CMF. Nabýval hodnot

- CMF1 = 0
- CMF2 = 1e^{-8}



Obr.11 Graf průběhu polohy referenčního bodu

10.1.4. Vyhodnocení

Ustálený stav u všech simulací je nepohybující se koule na podložce. V ustáleném stavu je poloha v ose y referenčního bodu 8mm, což odpovídá poloměru koule. Měření polohy je zatíženo chybou vyplývající z přesnosti použitého formátu dat.

Z grafů je patrný vliv jednotlivých parametrů na chování soustavy. Parametr ERP ovlivňuje reakční sílu v kontaktech, kde tělesa kolidují. CMF ovlňuje prostupnost těles, je tedy ekvivalentní tuhosti tělesa. Obě hodnoty je nutné pečlivě volit; doporučuje se odladit simulaci s nulovou hodnotu CMF a s hodnotu ERP mezi 0.8 až 1.

Vliv hustoty tělesa na chování simulace je patrný z obr.8. Změna hustoty se projevuje od kroku 10 000, což je fyzikálně nesprávné. Jedná se pružnou srážku, kinetická energie soustavy se zachovává. V důsledku toho by se trajektorie těles neměli lišit. Zdali je projevené chování simulací akceptovatelné vzhledem k řízení robotu, závisí na povaze simulací. Není pravděpodobné, že se vyskytne taková situace (spouštění tělesa na podložku opakovaně bez dalšího působení uživatelsky zadaných sil a momentů), ve které by uvedená chyba dovedla simulaci k nestabilnímu nebo výpočtově výrazně nesprávnému výsledku. Přesto se v důsledku těchto simulací rozšiřuje pásmo nepřesnosti ODE.

10.2. Vliv rozměrů těles na simulaci

Použitý wrapper OdeDotNet byl zkompilován pro datový typ System.Single. Používala se tedy čísla v rozmezí $1,5*10^{-45}$ až $3,4*10^{38}$. Provedené simulace se chovaly stabilně, pokud se rozměry těles pohybovaly v řádech 10^2 až 10^3. Jestliže se rozměry těles

pohybovaly v řádech jednotek, simulovaná soustava byla značně nestabilní. Zvětšení rozměrů modelované soustavy má za následek větší výpočetní náročnost na stejnou dobu simulace s menšími rozměry modelované soustavy.

10.3. Simulace chování těles při působení impulzu síly

Tato podkapitola čerpá z [8,9]

V této simulaci bylo použito wrapperu OdeDotNet verze 0.1.0.

Tato simulace spočívá ve spouštění tělesa na podložku za působení síly v prvním simulačním kroku. Těleso je koule o poloměru 8mm.

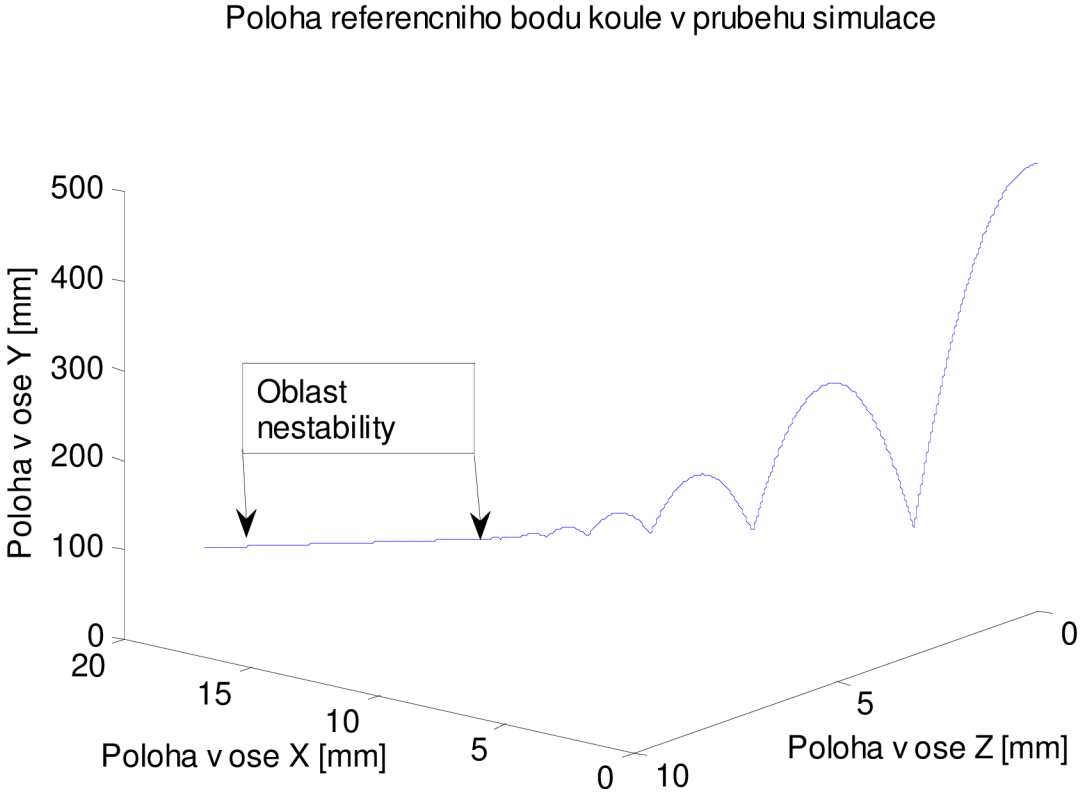
- CFM je $3*10^{-12}$
- ERP je 10.8
- Gravitační zrychlení je definováno vektorem [0, -9.81, 0]
- Těleso je spuštěno z bodu [0,500,0] [mm]
- Síla působící na těleso v prvním kroku simulace je:

$$F_X = 1*10^{12}\,N \tag{19}$$

$$F_Y = 0N \tag{20}$$

$$F_Z = 5*10^{11}\,N \tag{21}$$

- Hustota tělesa je 7850 kg/m^3



Obr.12 Graf průběhu polohy referenčního bodu tělesa

V průběhu simulace těleso mělo nulovou rotaci. V oblasti nestability se proti pohybu tělesa musí definovat třecí síla. V oblasti nestability se však rychlost tělesa neměnila. Z toho lze usuzovat na absenci třecí síly, což je neodpovídající reálnému chování.

Tření je definováno jako vazba mezi dotýkajícími se tělesy. Tuto vazbu definuje funkce Collide, kterou volá funkce SpaceCollide. Ta předává delegátu funkce NearCallBack (součást ODE) informace, které obsahují dvě geometrie, které kolidují, a data o jejich kontaktu. Ve funkci, kterou definujeme jako delegáta funkce NearCallBack, je následující (zjednodušený) pseudokód:

- Najdi tělesa, kterým patří kolidující geometrie
- Vytvoř pole kontaktů
- Nastav jejich vlastnosti
- Vytvoř vazby odpovídající jednotlivým kontaktům

Nastavení vlastností takto vytvořených vazeb se děje pomocí:

```
contact.Surface.Mode = ContactFlags.Bounce
contact.Surface.Mu = float.MaxValue;
```

```
contact.Surface.Mu2 = float.MaxValue;
```

Také se používá nastavení:

```
contact.Surface.SoftCfm = 0.000001f;
contact.Surface.SoftErp = 0.5f;
```

Surface.Mode nastavuje způsob aplikace tření v simulaci. Je definováno více způsobů, jak tření aplikovat, nejpoužívanější je pyramidální způsob ContactApprox1_1.

Surface.Mu (Mu2) nastavuje Coulombovský koeficient tření. V simulacích se však občas nastavuje na maximální hodnotu.

Surface.SoftCfm (SoftErp) nastavuje hodnoty ERP a CFM pro definované vazby. Nastavení funguje však jen příkazem Surface.Mode.ContactSoftCFM a Surface.Mode.ContactSoftErp.

Lze také nastavit, kolika vazbami se má kontakt uskutečnit. Platí úměra, že čím více vazeb je aplikováno na místě kontaktu, tím reálnější je chování soustavy.

Naznačené možnosti úpravy vazeb byly vyzkoušeny mnoha kombinacemi, avšak jakékoliv nastavení nemělo vliv na chování simulace. Jedná se o chybu wrapperu OdeDotNet ve verzi 0.1.0. Novější verze nebyla po dobu zpracování této diplomové práce vydána.

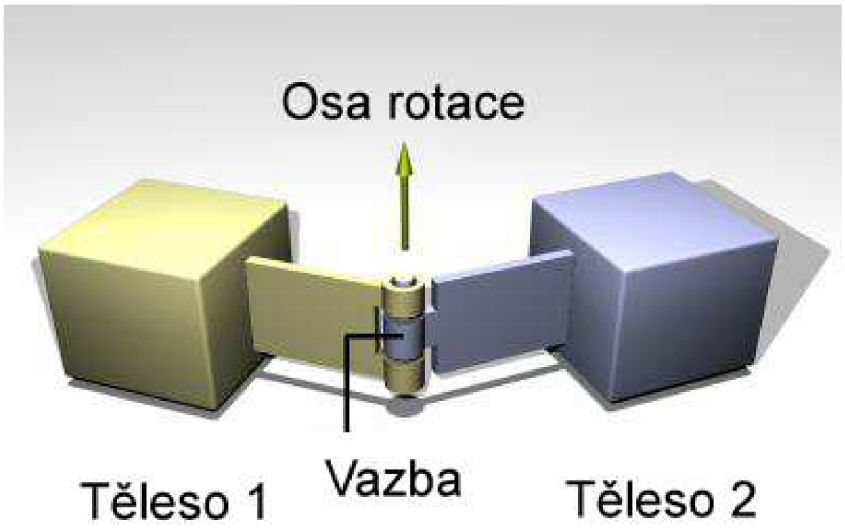
10.4. Simulace vazby Hinge

V této simulaci byla definována dvě tělesa tvaru kvádru. Protože se při použití wrapperu OdeDotNet nepodařilo použít vazbu FixedJoint (vetknutí), byla tato vazba nahrazena dvěma vazbami Hinge, každá z nich s jinou osou rotace. Tím byly odebrány podpůrnému tělesu všechny stupně volnosti. Těleso č. 1 je s podpůrným tělesem spojeno rotační vazbou typu Hinge. Na těleso č. 1 v prvnim simulačním kroku působí impuls sily.

- CFM = $3*10^{-12}$
- ERP = 1
- Gravitační zrychlení je definováno vektorem [0, -9.81, 0]
- Rozměry tělesa 1 jsou 260x60x60[mm]
- Rozměry podpůrného tělesa jsou 100x60x60[mm]

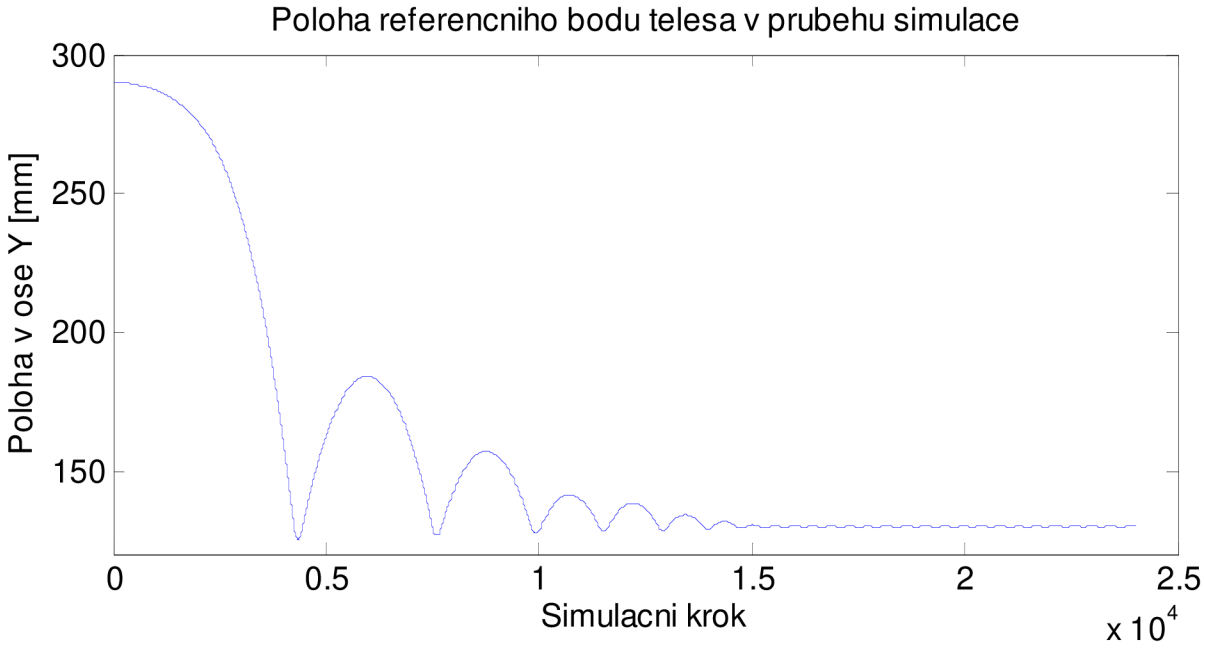
Působící síla (impulz ve smyslu ODE):

$$F_X = 1*10^8 N \tag{22}$$

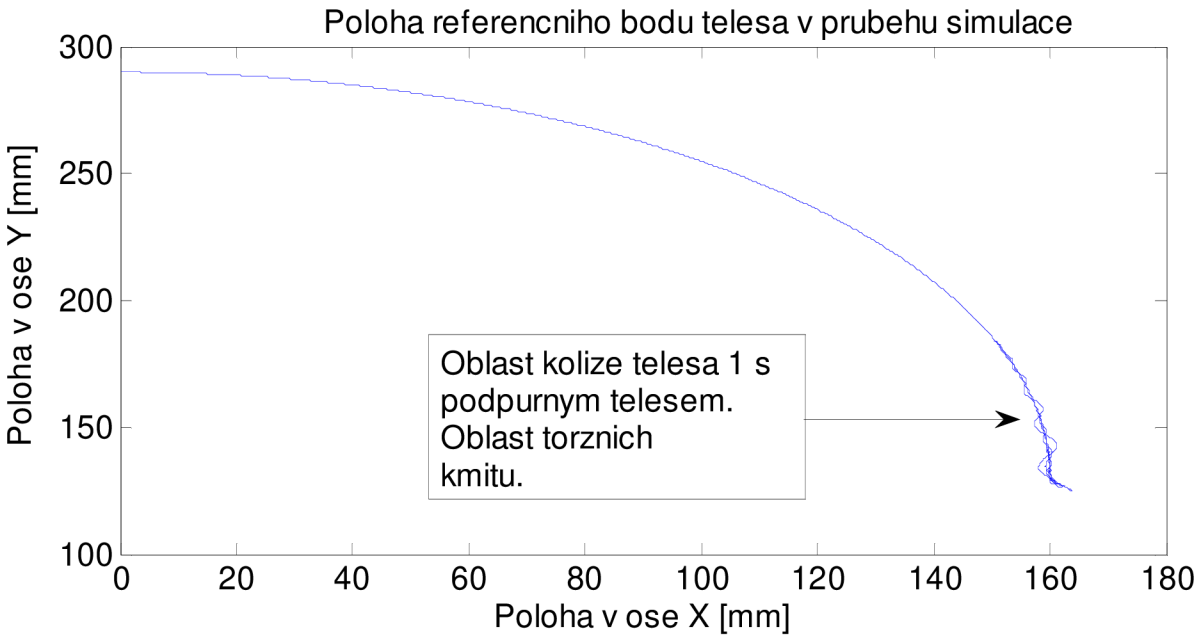


Obr.13 Vazba Hinge (Obrázek převzat a upraven z [4])

Vazba mezi tělesy je v bodě [0, 130, 0]mm. Osa rotace je rovnoběžná s osou Z.



Obr.14 Graf průběhu referenčního bodu tělesa č. 1



Obr.15 Graf průběhu referenčního bodu tělesa č. 1

Simulace ukázala, že při kolizích dochází k torzním kmitům těchto kolidujících objektů. V tomto případě došlo k torzním kmitům tělesa č. 1. Vznik torzních kmitů je možný v případě, kdy jedna nebo více součástí soustavy vykazuje nesymetrické chování nebo má nedokonalý povrch. V simulaci pomocí ODE se uvažují tvarově dokonalá tělesa (v rámci přesnosti použitého datového typu). Nesymetrické chování tedy projevuje vazba Hinge. Simulacemi bylo prokázáno, že vznik (viditelný vznik) torzních kmitů závisí na velikosti těles. Různě veliká tělesa mají za následek různě veliké síly působící ve vazbách. Pro vazby (stejně jako pro celý dynamický svět) jsou definovány parametry CFM a ERP. Kombinace těchto parametrů přímo ovlivňuje vznik a velikost torzních kmitů.

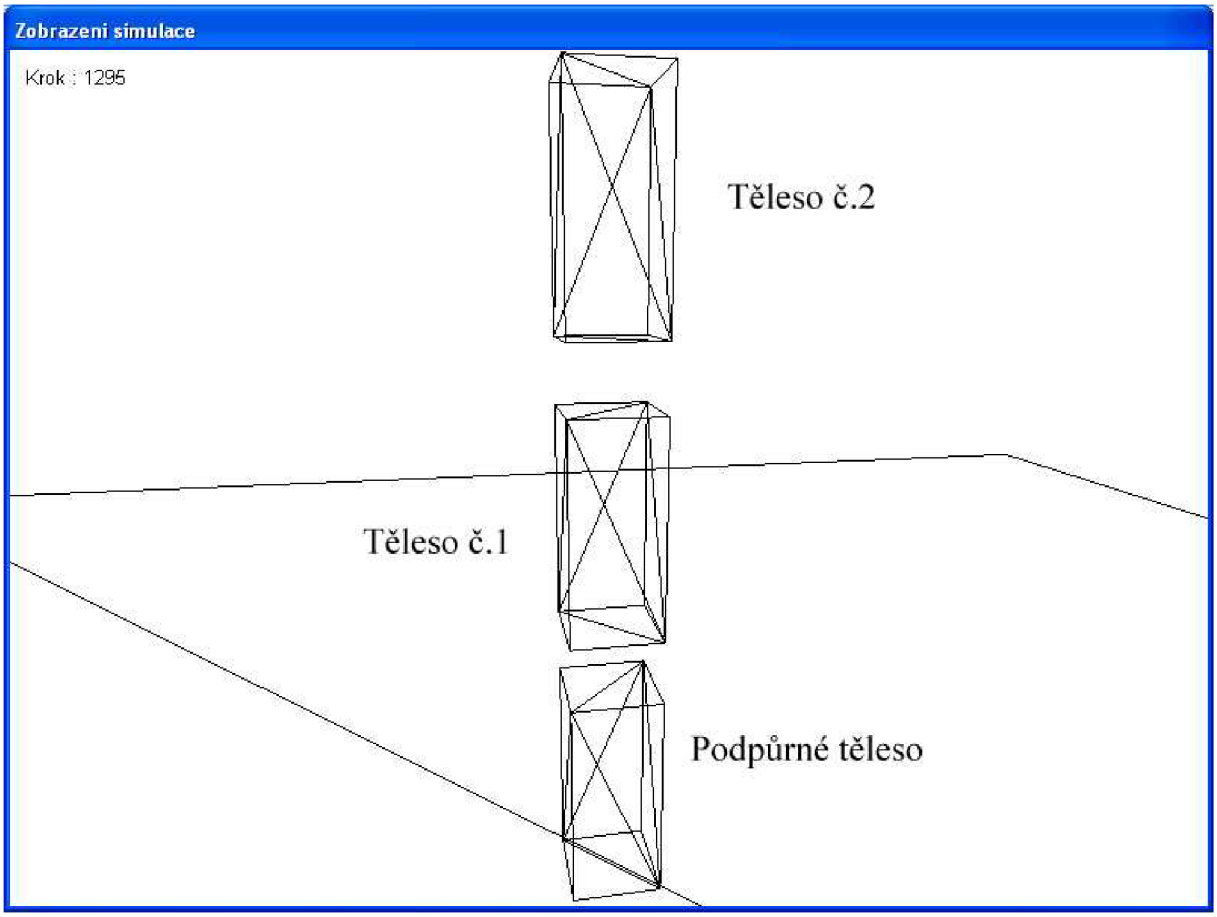
Průběh referenčního bodu v ose Y se neustálí na hodnotě 130mm, ale neustále kolem této hodnoty kmitá v malých odchylkách. Je to následkem nastavení parametrů ERP a CFM.

10.5. Simulace dvou na sobě závislých vazeb Hinge

Byly provedeny dvě simulace na sobě závislých vazeb. Obě simulace mají stejné počáteční podmínky. Podpůrný kvádr (nejníže) je vetknutý do podložky. Podpůrné těleso je vetknuto (dvěma rotačními vazbami s rozdílnými osami rotace) do podložky. Těleso č.1 je k podpůrnému tělesu připojeno pomocí vazby Hinge (osa rotace rovnoběžná s osou Z), stejně tak jako je připojeno těleso č.2 k tělesu č.1.

Nastavení simulace je:

- CFM je $3*10^{-12}$
- ERP je 1
- Gravitační zrychlení je definováno vektorem [0, -9.81, 0]
- Rozměry těles jsou 50x50x20[mm]



Obr.16 Počáteční stav obou simulací

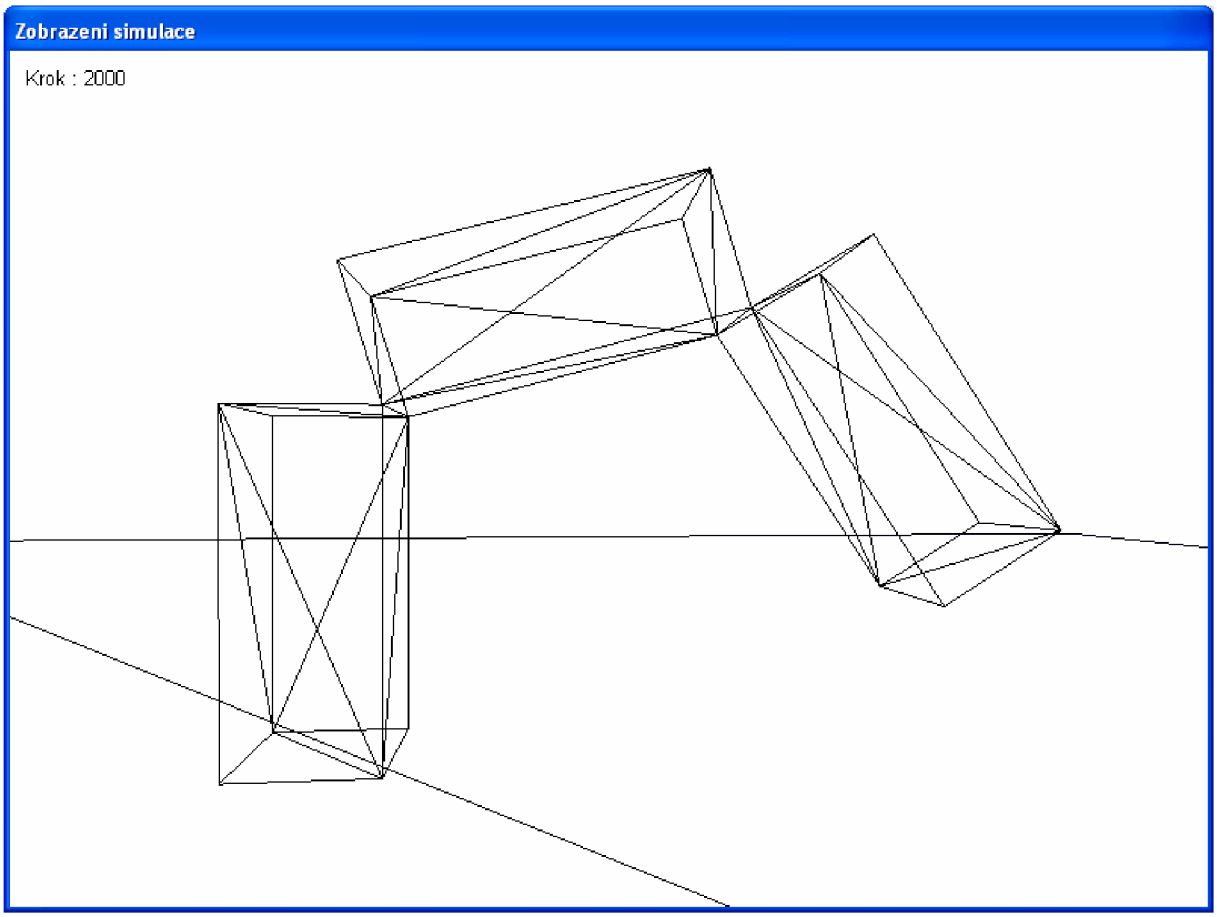
Na obr.16 jsou patrny mezery mezi tělesy. V ODE je možné zadat vazbu kdekoliv v definovaném dynamickém světě. Na obr.16 je vidět, že mezi podpůrným tělesem a podložkou není žádná mezera. Přesto jsou v tomto místě definovány dvě vazby typu Hinge.

Po celou dobu simulace se neměnily žádné parametry simulace a s pozorovanými tělesy nekolidovala žádná jiná tělesa.

V této simulaci byly definovány dvě množiny vazeb ode_contact_group a ode_contact_group_2. Skupina vazeb ode_contact_group je určena pro ty vazby, které jsou definovány kolizní funkcí. V každém simulačním kroku je tato skupina vazeb vymazána. Druhá skupina vazeb ode_contact_group_2 je skupina vazeb obsahující uživatelsky definované vazby mezi tělesy. Tato skupina těles je definována jen jednou a nikdy není

vyprázdněna. Wrapper OdeDotNet sice dovoluje vytvořit vazby a nepřiřadit je do žádné množiny vazeb, ale ze simulací plyne (prokázáno i krokováním simulaci), že tyto nepřiřazené vazby jsou vymazány jakmile dojde na vyprazdňování kterékoliv množiny vazeb.

Kód definující jednoosou rotační vazbu mezi tělesy je v příloze A.



Obr.17 Koncový stav obou simulací

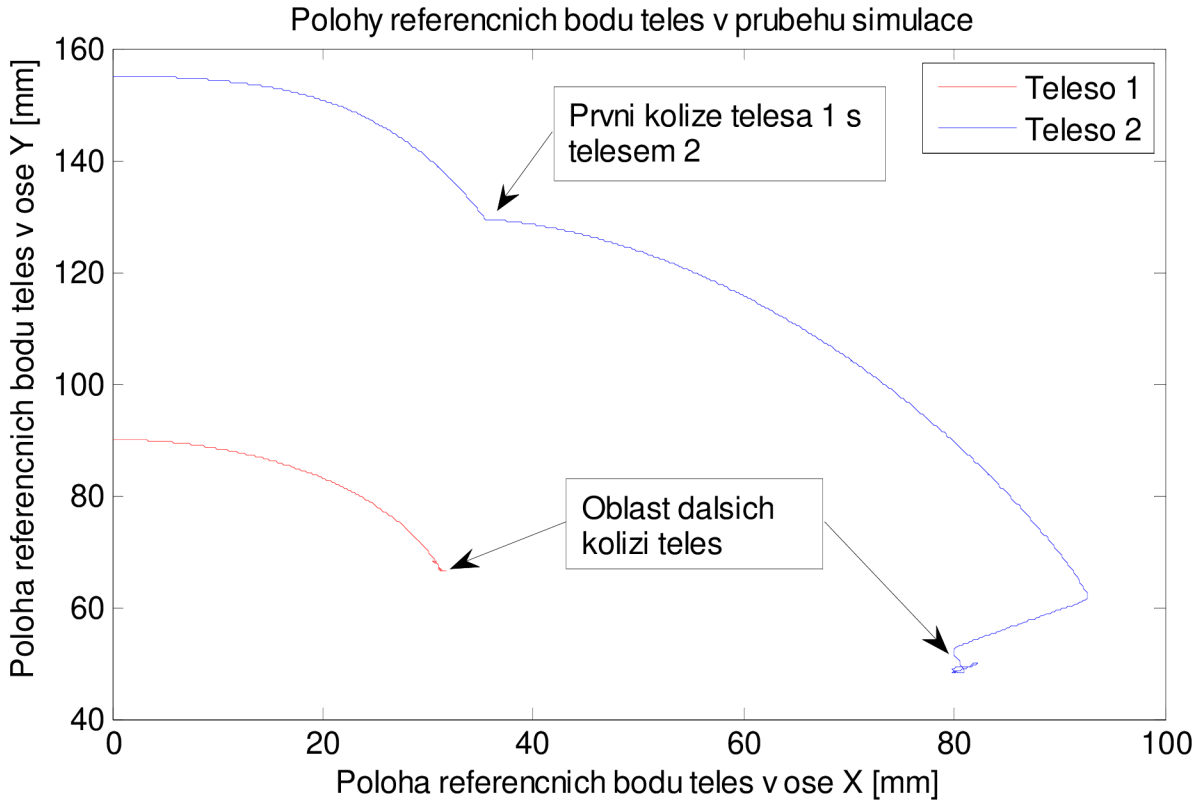
Koncový stav obou simulací je omezen vazbami a vzájemnou kolizí těles. Z obr.17 je patrné, že rozsah rotačních vazeb je omezen právě geometriemi těles. Po celou dobu simulace nedošlo ke zjistitelnému průniku těles navzájem. Pokud by se nedodržely dostatečné vzdálenosti mezi tělesy, kolize těles by znemožňovala jakýkoliv pohyb těles. Rotační vazba odebírá pět stupňů volnosti a kolize těles by odebrala zbývající šestý.

10.5.1. Simulace za působení síly

V prvním kroku simulace působí na těleso č.1 síla:

$$F_Z = 1 * 10^8 N \tag{23}$$

Tím je simulován impuls síly (ve smyslu ODE). V dalších krocích simulace nepůsobí na tělesa žádné síly ani momenty zadané uživatelem. Simulace prokázala stabilní chování soustavy po relativním ustálení referenčních bodů těles. Při nesprávném nastavení parametrů soustavy dochází k uvolňování až rozpojení vazeb u těles. To vede k nestabilnímu chování soustavy neodpovídajícímu reálnému chování.

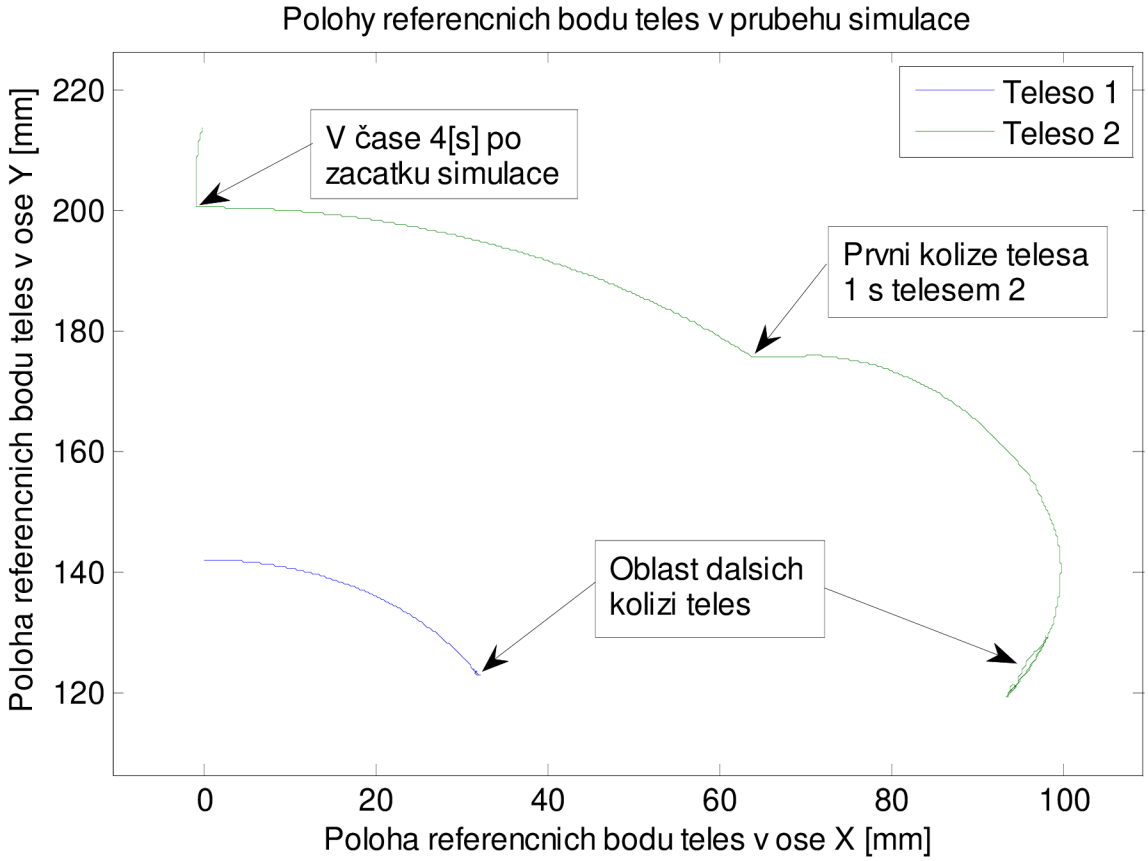


Obr.18 Graf průběhů poloh referenčních bodů těles

Soustava se ustálila přibližně 13s po začátku simulace.

10.5.2. Simulace bez působení sil a momentů

Počáteční stav simulací, tak jak je zobrazen na obr.16 , je přirozeně nestabilní soustava v důsledku působení gravitačního zrychlení. V tomto případě se nutně musí soustava dostat ze stavu počátku do stavu koncového obr.17 .Stav na obr.17 je symetrický s rovinou symetrie, která je kolmá na podložku a prochází referenčním bodem podpůrného tělesa. V tomto případě je koncový stav soustavy v oblasti kladné poloroviny XY (koncový stav tělesa je také symetrický).



Obr.19 Graf průběhů poloh referenčních bodů těles

Simulace se ustálila přibližně 21s po začátku simulace. Mezery mezi tělesy byly v této simulaci zmenšeny, což mělo za následek menší výchylku referenčních bodů těles. Spustitelný program je v příloze D.

11. Model robotu Kráčmera I.

Model mobilního robotu s kráčejícím podvozkem je nelineární dynamická soustava, pro kterou není možné použít krokovou funkci QuickStep. To určuje i výpočetní náročnost této simulace. Při navrhování takové simulace se výpočetní nestabilita, špatné nastavení nebo nevhodná posloupnost počítačového kódu může projevit jako „exploze“ simulace. Exploze simulace má různé následky na chování simulované soustavy:

- Indikace sil v nesmyslném směru a velikosti
- Samovolné tvoření a úprava stávajích vazeb
- Nevytváření vazeb při kolizích objektů a jiné

Pro simulaci chování robotu v prostředí ODE bylo přijato několik zjednodušení.

11.1. Zjednodušení nohy robotu

Noha je tvořena dvěma díly vzájemně propojenými vazbou typu Hinge. Tato vazba má jeden stupeň volnosti (jednu osu rotace).

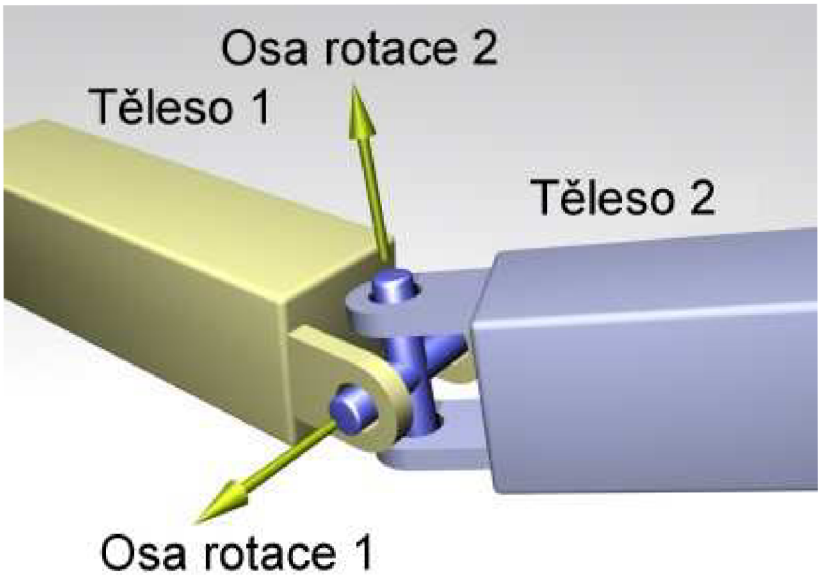
V ODE jsou jednotlivé části tvořeny dvěma částmi typu kvádr. Jejich rozměry jsou nastavitelné. Mezi tyto části je umístěna vazba. Z tohoto důvodu je třeba zachovat mezi stěnami kvádru, kde je vazba definována, mezeru. V případě nedostatečné mezery nelze zaručit plný rozsah pohybu vazby, v tomto případě by tělesa kolidovala a byly by vytvořeny další vazby funkcí Collide. Následkem toho by spojení těchto dvou těles bylo tvořeno více než jednou vazbou.

11.2. Zjednodušení trupu robotu

Podobně jako u nohou robotu je trup aproximován jedním tělesem tvaru kvádr. Rozměry a hustota trupu jsou plně nastavitelné. Trup je nejtěžší částí robotu a má tak největší vliv na dynamiku celého robotu.

11.3. Spojení nohou robotu s trupem robotu

Spojení nohou robotu s trupem robotu je provedeno vazbou typu Universal. Jedná se o rotační vazbu s dvěma stupni volnosti (dvě osy rotace). I v tomto případě je nutné zachovat mezeru mezi spojenými tělesy. Rozsah těchto vazeb je určen jen kolizí trupu a nohou.

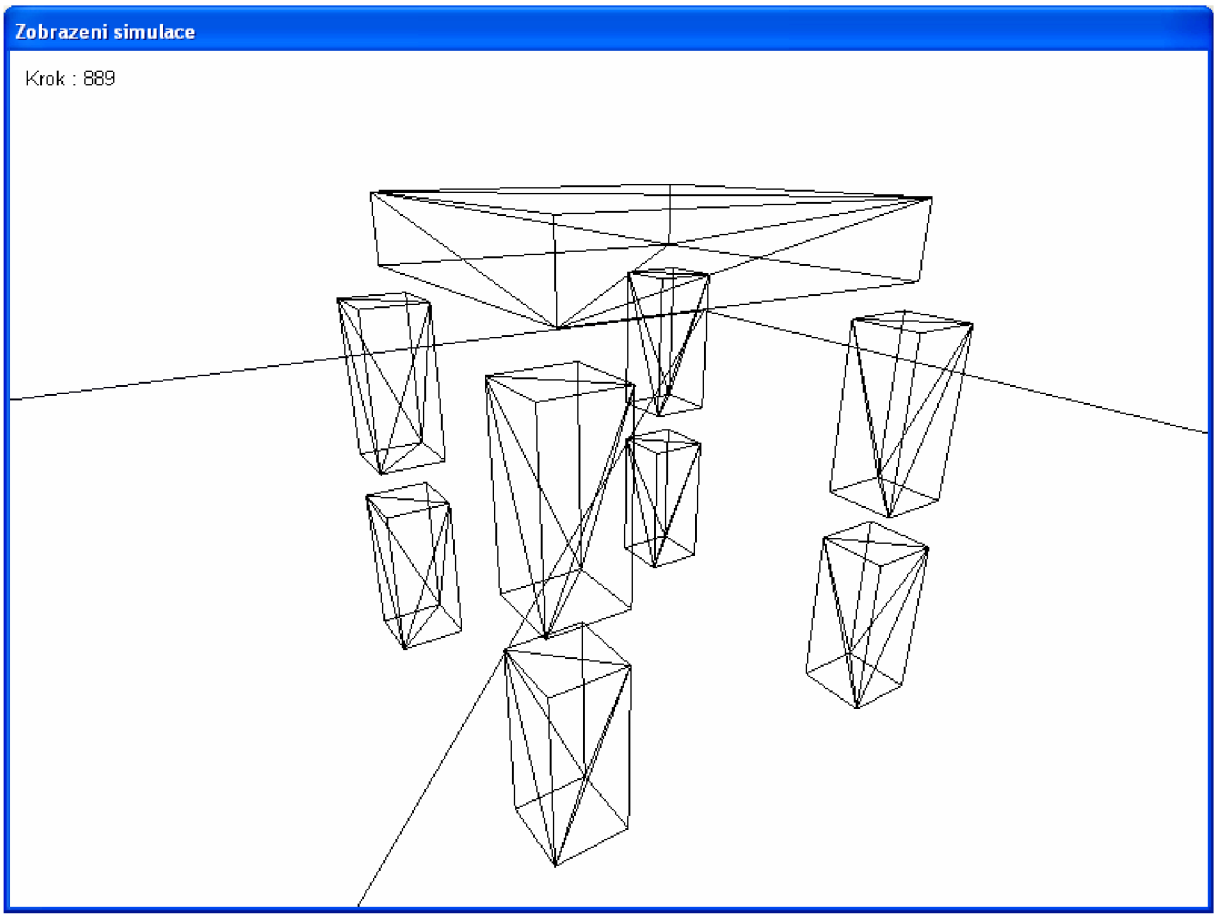


Obr.20 Vazba typu universal (Obrázek převzat a upraven z [4])

11.4. Model robotu

Počáteční stav modleu robotu „Kráčmera I.“ je zobrazen na obr.21 . V průběhu simulování došlo k několika nestabilním chováním modelu. Docházelo tedy k explozi simulace. Toto chování se vizuálně projevovalo přerušením vazeb, indikaci sil takových, že se robot vznesl aniž by byli zadávány jakékoliv síly nebo momenty.

Matematicky se toto chování dalo povětšinou charakterizovat náhlou změnou rychlosti referenčních bodů těles. Vektor rychlosti se u všech těles měnil přibližně stejně s tím, že platí čím větší je změna rychlosti (nebo polohy), tím pravděpodobnější je, že došlo k destabilizaci simulace.



Obr.21 Obrázek modelu robotu. Výchozí stav soustavy.

Simulační program robotu je součástí přílohy D. Ve zdrojovém kódu programu lze měnit zatížení, jednotlivé vazby, rozměry, hustoty těles. Je tedy možné nasimulovat různé situace.

Pro tuto simulaci byl použit wrapper OdeDotNet. U tohoto wrapperu byl prokázáno několik chyb a proto nelze brát výsledné simulování za prokazatelné.

11.5. Možnosti řízení mobilního robotu Kráčmera I. s pomocí ODE

Mějme autonomně řízenou soustavu. Pokud má být taková soustava použitelná k náhradě lidského zdroje, musí být řízena přiměřeně kontinuálně. Řízení takové soustavy se v drtivé většině děje pomocí výpočetní techniky. Pokud se taková soustava má pohybovat, musí být vydáváné příkazy k pohybu podrobeny různým omezením. Omezení mohou být různá, nejčastěji jsou to však omezení prostorová.

Ovšem nároky na řízení mobilního robotu s kráčejícím podvozkem jsou velké vzhledem k omezením vyplývajícím z jeho konstrukce a povahy podvozku. V zásadě může být několik způsobů, jak řídit pohyb robotu tak, aby se robot mohl chovat autonomně.

Požadavek autonomního chování je v důsledku série příkazů, které podléhají omezením. Zaměříme-li se na oblast stability robotu, je několik způsobů, jak klást omezení na následující pohyb robotu. Avšak všechny způsoby jsou vyjádřeny přesnými matematickými rovnicemi a ty se nemohou jednoduše automaticky upravovat v real-time řízení.

Použitelnost ODE k těmto účelům je závislé na výše uvedených faktorech. Lze očekávat, že v průběhu několika let dojde ke zlepšení ODE natolik, že bude možné tento nástroj používat i jako verifikační nástroj chování soustav.

11.5.1. Použití modelu robotu v ODE pro návrh a simulační ověřování řízení

Rychlost simulace soustav realizovaných v ODE je výrazně rychlejší než je-li realizována programech ze skupiny 1 a 2 (viz. kapitola 2.3). Další nespornou výhodou je, že v každém simulačním kroku můžeme okamžitě získat informace o poloze, rychlosti o každém tělesu modelu, z použitých vazeb lze získávatž informace o vzájemné pozici těles a velikosti působících sil (lze i vytvořit tlakový senzor). Dále lze v průběhu simulace měnit parametry modelu i vazeb. Je možné implementovat různé rozhodovacící a vyhodnocovací algorytmi, které jsou pro simulaci potřebné.

Práce s ODE je relativně jednoduchá a algorytmizovatelná, lze tedy upravovat model automaticky bez zásahu uživatele. Pomocí ODE lze rychle a dostatečně přesně odhadovat budoucí stavy simulované soustavy, jak vyplývá z výše uvedených simulací.

Tudíž lze použitím dynamického modelu soustavy realizovaném pomocí ODE výrazně urychlit simulace[13].

11.5.2. Použití modelu robotu v ODE jako prediktoru budoucích stavů pro predikční řízení

Jak již bylo uvedeno v předchozí kapitole, běh simulací v ODE je velmi rychlý. V práci uváděné simulace, které měli 10 000 simulačních kroků, byly spočítáni za dobu kratší jedné vteřiny. Tato vlastnost je pro real-time řízení rozhodující.

ODE lze využit pro řízení dvěma hlavními způsoby

Prvním způsobem je pomocí ODE „natrénovat“ neuronovou síť pro řízení nelineární soustavy jakou je kráčivý robot „KráčmeraI.“. Čtyřnohý kráčivý robot je natolik komplexní soustava, že neuronová síť, která by měla takovou soustavu řídit, by byla relativně rozsáhlá. Výpočetní náročnost takové sítě je následně značná, oproti použití ODE jako prediktoru následujících stavů v real-time řízení. Výhodou použití neuronových sítí je, že „naučení“ takové sítě probíhá jen jednou. Nezáleží nám tedy na době „učení“ sítě. Toto je obvyklý způsob použití neuronové sítě jako prediktoru, ale vzhledem k rychlosti počítání simulačního kroku v ODE, je možné že výpočet jednoho kroku neuronové sítě bude pomolaješí než výpočet simulačního kroku v ODE.

Druhým způsobem použití ODE je použití jeho použití jako prediktoru následujích stavů soustavy. Takový přístup vyžaduje uchovávání aktuálního modelu robotu pomocí ODE v paměti po celou dobu řízení. Pro robot lze tímto způsobem predikovat přibližné následky zamýšleného kroku. ODE umožňuje vytvářet více dynamických světů zároveň, aniž by se navzájem ovlivňovaly. Toho se dá využít k verifikaci údajů tak, že vytvoříme množství stejných modelů s velmi podobnými počátečními podmínkami, aplikovat potřebný počet časových kroků a dle výsledků řídit svůj následující krok. V závisloti na výkonu použitého hardwaru lze simulovat množství časových kroků dopředu. ODE je pro takový způsob řízení vhodná.

12. Závěr

Tato práce se zabývá ověřením použitelnosti fyzikálních enginů pro modelování dynamiky mobilních kráčivých robotů. V první části práce je popsána problematika modelování dynamických soustav, mezi které mobilní kráčivé roboty patří. Byly ukázány dva hlavní přístupy k modelování dynamických soustav, z nichž byl zvolen ten způsob modelování soustav pomocí fyzikálních softwarů, který se jeví výhodnější pro simulace a řízení mobilního robotu „Kráčmera I.“. V práci jsou dále popsány tři hlavní způsoby počítačového modelování dynamických soustav. Pro zbytek práce byl zvolen fyzikální engine Open Dynamics Engine (ODE), který je šířen jako open-source pod BSD licencí. Simulace s ODE jsou realizovány na platformě Microsot .NET 2.0 a kód simulací byl napsán v jazyce C# 2.0. Pro zpřístupnění ODE pod Microsot .NET 2.0 byly použity wrappery, které jsou shrnuty v kap.3.2.. Jako nejpoužitelnější wrapper se během experimentů ukázal wrapper OdeDotNet; proto byl použit pro zbytek práce. V ní dále následuje popis struktury ODE, popis hlavních objektů a způsob práce s jednotlivými částmi. Ve zbylé části jsou ukázány příklady simulací, které jsem v rámci práce realizoval. Pro realizaci simulací jsem naprogramoval jednoduché zobrazovací prostředí, které využívá grafického rozhraní Microsoft DirectX.

Během simulací a experimentování s ODE jsem narazil na množství problémů, které byly způsobeny jak chybami v použitých wrapperech, tak chybami přímo v ODE. I přes tyto problémy se ODE jeví jako použitelný nástroj pro dynamické simulace. Do budoucna je možné připojit se buď přímo k vývojovému týmu ODE nebo ODE upravit pro řešenou problematiku.

Seznam literatury

[1] *Autodesk – Autodesk Inventor.* URL: < *http://usa.autodesk.com/*> *[cit. 2008-05-21].*

[2] *Manual (Concepts) ODE – Wiki URL: < http://opende.sourceforge.net/mediawiki-1.6.10/index.php> [cit. 2008-05-08].*

[3] *HOUŠKA, Pavel, Ing,Ph.D., Model Kráčmera I.[model v programu Invertor] Brno, 2008, Model v programu Invertor, Vyžaduje počítačový program Invertor, Obrázky pořízeny z programu Invertor.*

[4] *Manual (Concepts) ODE – Wiki URL: < http://opende.sourceforge.net/mediawiki-1.6.10/index.php> [cit. 2008-05-09].*

[5] *TROELSEN, Andrew. C# a .NET profesionálně. Třetí vydání.2006.Brno.Zoner Press.ISBN : 80-86815-42-0*

[6] *MILLER, Tom. Programujeme 3D hry v jazyce C#. První vydání.Brno.Computer Press: 28.dubna 48 .ISBN : 80-251-1126-1*

[7] *PERKINS, Jason. Readme.txt. Textový soubor. Šířen jako textový průvodce k instalaci ODE.NET. Verze 0.9. [citováno 24-04-24]*

[8] *Tao Framework URL: < http://opende.sourceforge.net/mediawiki-1.6.10/index.php> [cit. 2008-05-09].*

[9] *Tao Framework URL: < http://docs.taoframework.com> [cit. 2008-05-10].*

[10] *ODE URL: < http://www.ode.org> [cit. 2008-05-09]*

[11] *Newton Game Dynamics URL: < http://www.newtondynamics.com/> [cit. 2008-05-21].*

[12] *Vertex Physics SDK CM-labs URL: < http://www.cm-labs.com/> [cit. 2008-05-21].*

[13] *An ODE related book : Robot simulation URL: < http://demura.net/simulation> [cit. 2008-04-17].*

[14] *Počítačová simulace URL: < http://www.wikipedia.cz> [cit. 2008-03-11].*

[15] *DirectX - Wikipedie URL: < http://cs.wikipedia.org/wiki/DirectX> [cit. 2008-05-21].*

[16] *ŠVANCARA, Pavel, HOUFEK, Lubomír, MALENOVSKÝ, Eduard, KREJČÍ, Petr Dynamika Studijní opory UMTMB VUTBR FSI URL: <http://www.umt.fme.vutbr.cz/~pkrejci/opory/dynamika/kapitola_6.html > [cit. 2008-05-21].*

[17] *HALLIDAY D., RESNICK R., WALKER, J. Fyzika - Mechanika. Dotisk 1.českého vydání 2003.Brno.Nakladetelství VUTIUM, Nakladatelství Prometheus:.ISBN : 80-215-1868-0(VUTIUM)*

[18] *PELÁNEK, Radek Uvod.pdf URL: <http://www.fi.muni.cz/~xpelanek/IV109/jaro07/slidy/uvod.pdf> [cit. 2008-04-03]*

[19] *KŘÍVÝ, Ivan. KINDLER, Evžen. Simulace a modelování ,SkriptaKindlerMS.pdf, 2001, Ostravská univerzita, Učební texty Ostravské univerzity, Přírodovědecká fakulta*

[20] *MSDN : Microsoft Developer Network URL : < http://msdn.microsoft.com/cs-cz/default(en-us).aspx> [citováno vícekrát v průběhu 2008-01 až 2001-05]*

[21] *SourceForge.net OdeDotNet URL : < http://sourceforge.net/projects/odedotnet> [cit. 2008-03-22]*

[22] *MARTÍŠEK, Dalibor. Matematické principy grafických systémů, 2002, Littera Brno,CENTA, spol.s.r.o.,odštěpný závod Brno, Vídeňská 113. ISBN 80-85763-19-2*

[23] *Pro/ENGINEER Mechanica* URL : < http://www.ptc.com/products/proengineer-mechanica> [cit. 2008-05-22]

[24] Přednášky doc.Březina 2008 Z předmětu UMĚLÁ INTELIGENCE
[25] Program Matlab R2007b,www.mathworks.com

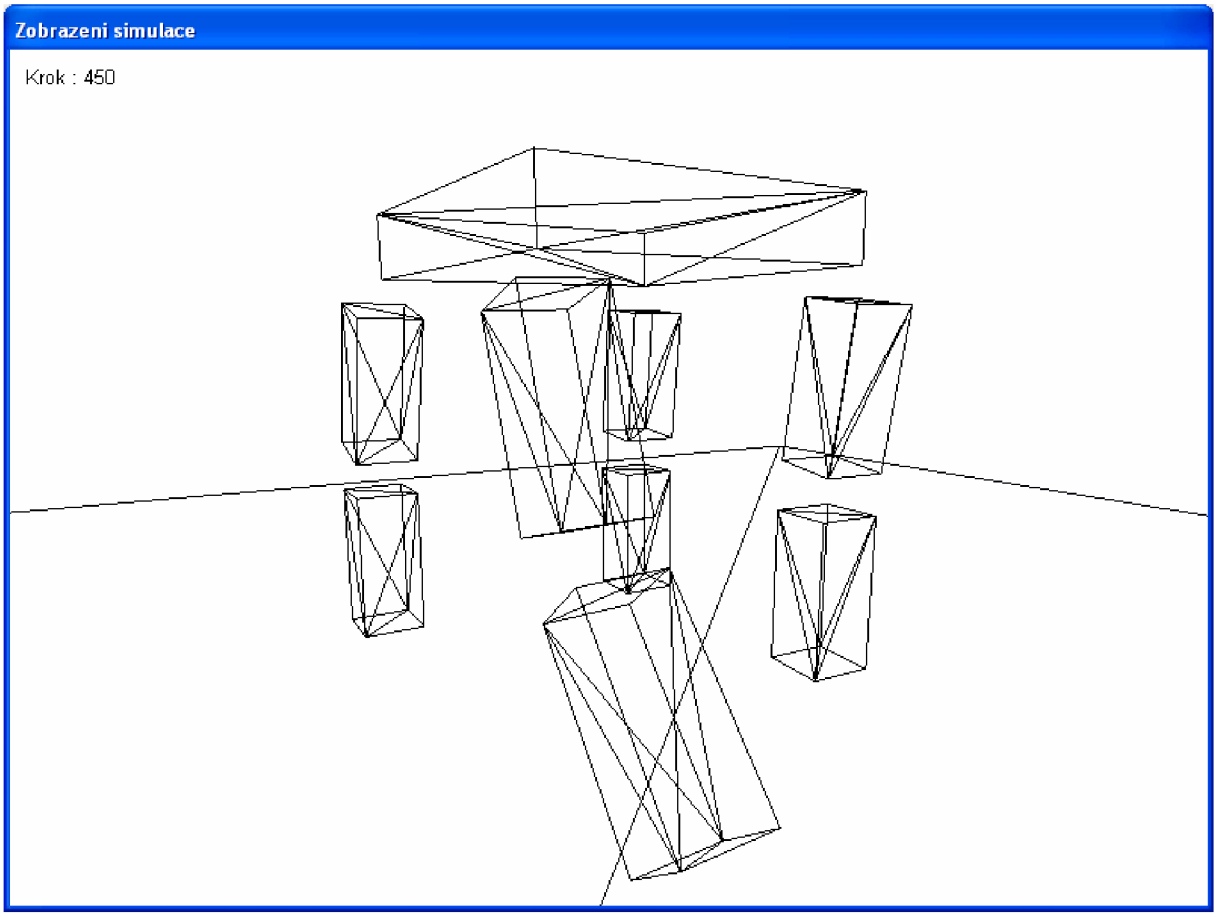
Přílohy

Příloha A

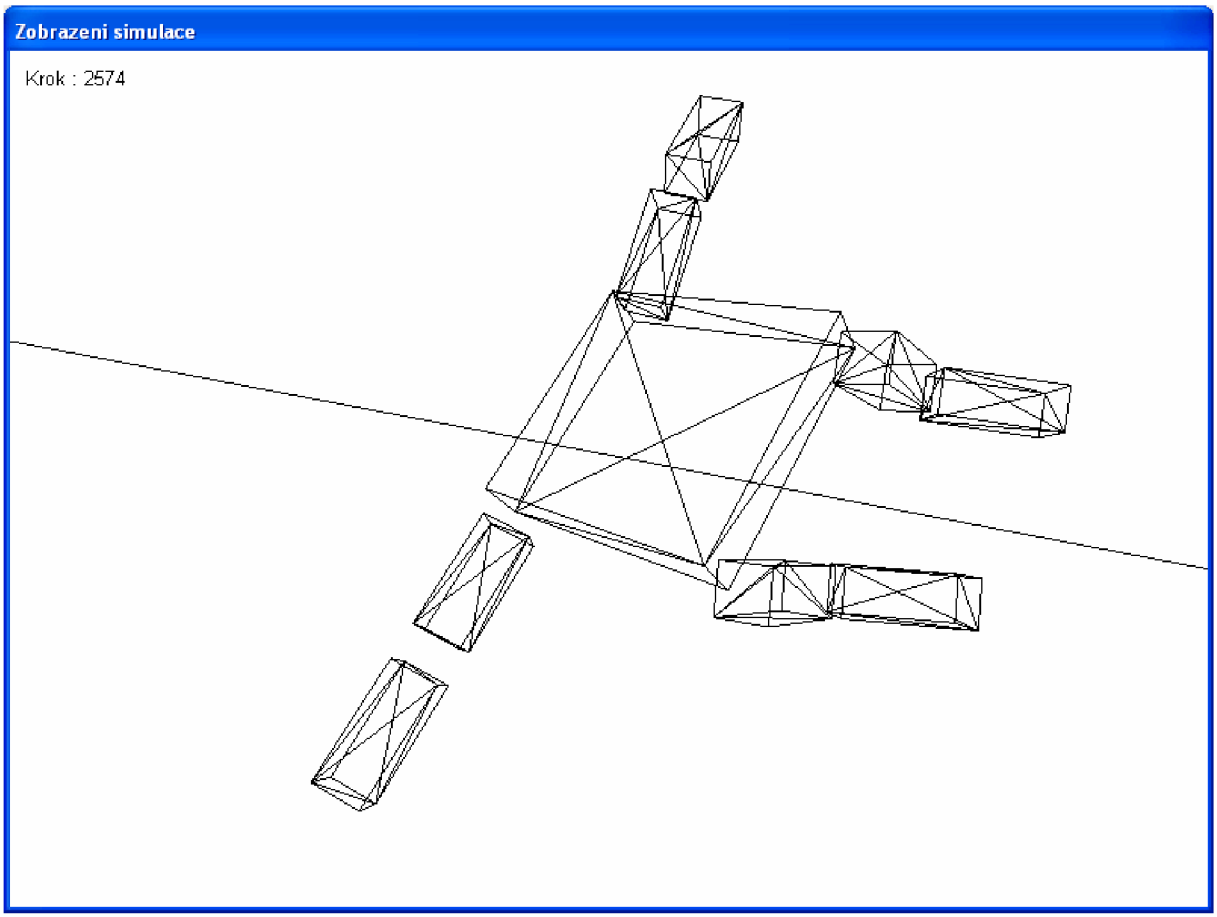
```
            OdeDotNet.Joints.HingeJoint joint = new
OdeDotNet.Joints.HingeJoint(dynamicky_svet, skupina_vazeb_ktera_se
nevyprazdnuje); // jaky typ a kam patří
            joint.Attach(součást_1, součást_2); // co spojit
            joint.Anchor = new Vector3(X,Y,Z); // pozice vazby
            joint.Axis = new Vector3(0, 0, 1); // osa rotace
```

Je nutné dbát na umístění kódu. Kód se musí nacházet v inicializační části programu. Pokud se bude kód nacházet v simulační smyčce, dojde k vytváření vazeb mezi tělesy vždy na stejném místě nezávisle na pohybu těles, což vede k nesprávnému, někdy těžko odhalitelnému chování soustavy.

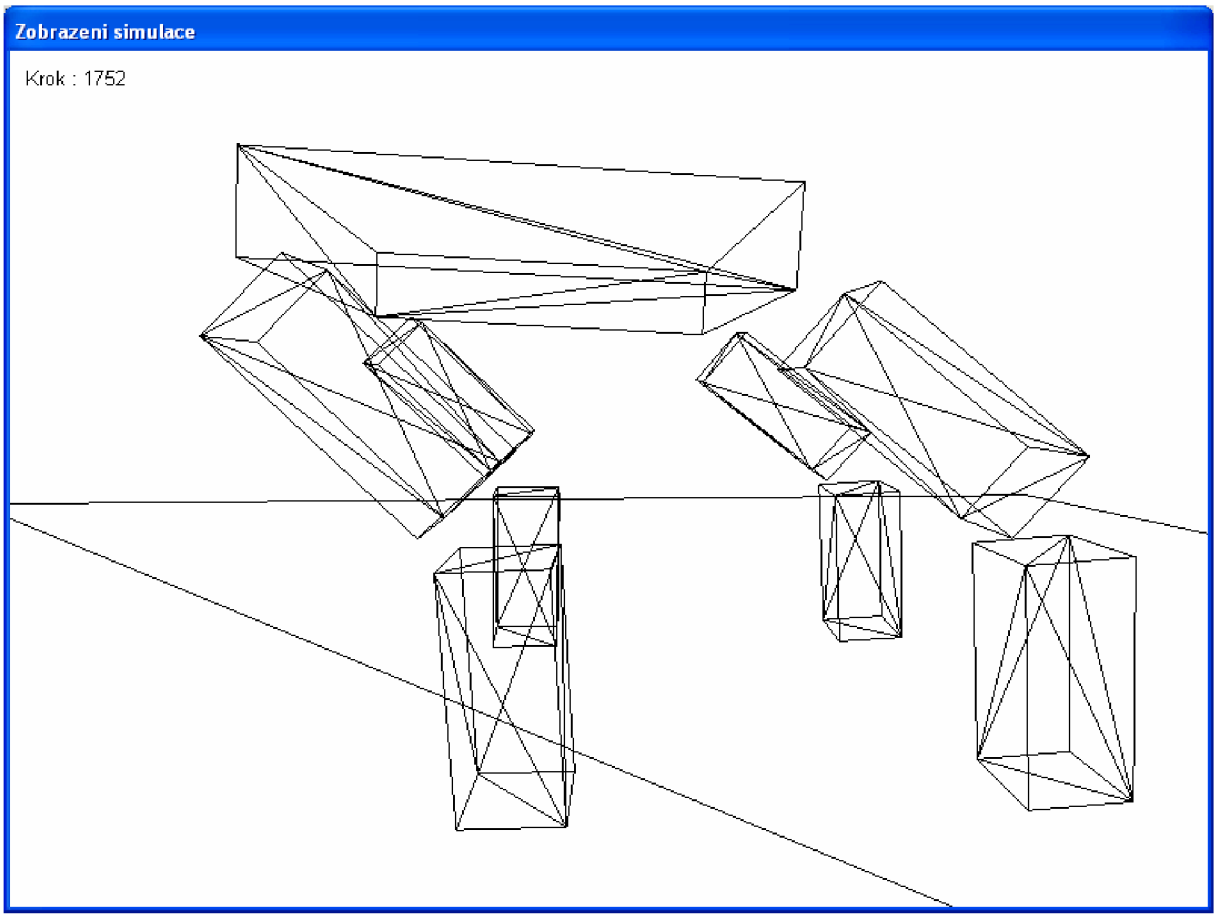
Příloha B



Obr.22 Účinkem impulsu síly (ve smyslu ODE) se modelovanému robotu přerušil kontakt jedné nohy s podkladem, následkem čehož došlo k destabilizaci celého robotu.

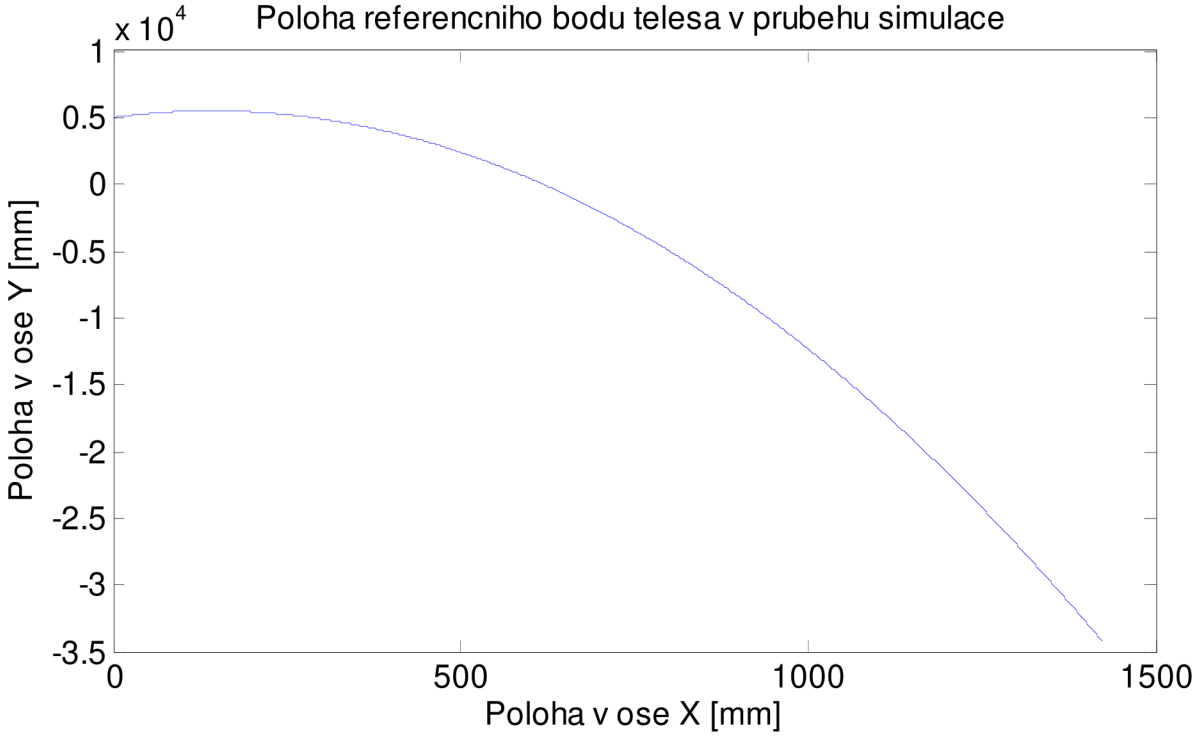


Obr.23 Jedno z možných konečných stádií simulace robotu. Vazby jsou omezeny jen kolizemi s jinými tělesy.



Obr.24 Ustálený stav robotu s vetknutými konci nohou do podkladu. Pozice je určena vazbami a kolizemi (resp rozměry jednotlivých objektů)

Příloha C



Obr.25 Zobrazení průběhu polohy tělesa v ose jednotlivých osách (spočteno a vykresleno pomocí programu MATLAB). Jde jen o znázornění korespondence průběhů pozic těles mezi výpočtem vycházejícím z rovnic a ODE.

Příloha D

CD