

# **Tvorba interaktivní fyzikální simulace v prostředí Easy Java**

Creating of Interactive Physics Simulation in Easy Java  
Environmet

Jan Sikora

---

Bakalářská práce  
2011



Univerzita Tomáše Bati ve Zlíně  
Fakulta aplikované informatiky

---

Univerzita Tomáše Bati ve Zlíně  
Fakulta aplikované informatiky  
akademický rok: 2010/2011

## ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: Jan SIKORA  
Osobní číslo: A07092  
Studijní program: B 3902 Inženýrská informatika  
Studijní obor: Informační a řídicí technologie

Téma práce: Tvorba interaktivní fyzikální simulace v prostředí Easy Java

### Zásady pro vypracování:

1. Proveďte literární rešerši na dané téma.
2. Definujte problematiku interaktivní simulace jako součást e-Learningu.
3. Zhodnoťte jednotlivá vývojová prostředí pro tvorbu interaktivních simulací.
4. Vyberte vhodné technologie pro vytvoření interaktivní simulace na základě existujících simulací, Open Source a jeho využití.
5. Navrhněte a vytvořte simulaci pohybu tělesa po nakloněné rovině.
6. Zhodnoťte strukturu aplikace a realizaci jednotlivých částí v prostředí Easy Java.

Rozsah bakalářské práce:

Rozsah příloh:

Forma zpracování bakalářské práce: tištěná/elektronická

Seznam odborné literatury:

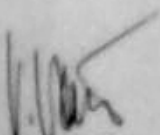
1. F. Esquembre. Easy Java simulations (Prentice Hall, Madrid, 2005). The Ejs program. Dostupný z WWW: <http://www.um.es/fem/Ejs>
2. Christian W., Belloni M. Physlet Physics: Interactive Illustrations, Explorations, and Problems for Introductory Physics. Davidson books.com. Aug 2003, 352p
3. W. Christian, F. Esquembre. Modeling Physics with Easy Java Simulations. The Physics Teacher Vol. 45. November 2007, 475
4. Paholok I. Simulácia ako vedecká metóda. E-logos Electronic Journal for Philosophy/2008. ISSN 1211-0442, 2008. Dostupný z WWW: <http://nb.vse.cz/kfil/elogos/student/paholok08.pdf>
5. Wieman C.E., Adams W.K., Perkins, K.K. Science. PhET: Simulations That Enhance Learning. 322/682-683. October 2008.
6. Wieman, C.E., Perkins, K.K., Adams, W.K. Oersted Medal Lecture 2007: Interactive simulations for teaching physics: What works, what doesn't, and why, American Journal of Physics. 76, 393, May 2008.

Vedoucí bakalářské práce: doc. RNDr. Miroslava Ožvoldová, Ph.D.  
Ústav matematiky

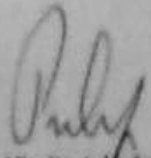
Datum zadání bakalářské práce: 25. února 2011

Termin odevzdání bakalářské práce: 7. června 2011

Ve Zlíně dne 25. února 2011

  
prof. Ing. Vladimír Vašek, CSc.  
děkan



  
prof. Ing. Vladimír Vašek, CSc.  
ředitel ústavu

## ABSTRAKT

Didaktický software je v dnešní době čím dál více nepostradatelnou součástí výuky v procese získávání informací. Bakalářská práce je zaměřena především na tu část didaktického softwaru, kam spadají interaktivní simulace. Cílem práce je naprogramovat interaktivní simulaci pro e-Learningový kurz z mechaniky, která zkoumá pohyb tělesa po nakloněné rovině spojeného lanem přes kladku s druhým tělesem. Realizovaná simulace umožňuje měnit parametry: hmotnost obou těles, smykové tření, úhel sklonu nakloněné roviny, gravitační zrychlení a sledovat důsledky jejich změny, například jednotlivých forem energie pro zkoumanou uzavřenou soustavu. Novinkou práce je, že simulace umožňuje studentovi získat experimentální data. V teoretické části práce pojednáme o simulacích a jejich zařazení do e-Learningu. Nahlédneme na jednotlivé programovací jazyky a prostředí. Odůvodníme, proč jsme pro praktickou část vybrali zrovna Easy Javu jakožto vývojové prostředí. Součástí je i náhled do Open Source zdrojového kódu. V praktické části prezentujeme návrh a konkrétní způsob vytvoření interaktivní simulace se pěti volitelnými parametry v prostředí Easy Java, s výpočty založené na „Euler-Richardsonové“ metodě.

Klíčová slova: e-Learning, Easy Java, tvorba interaktivní simulace, Open Source, programovací jazyky.

## ABSTRACT

Nowadays educational software becomes more and more essential part of education while getting knowledge. The bachelor thesis is focused mainly on the part of educational software which includes interactive simulations. The goal of the thesis is to create the interactive simulation for e-learning course in Mechanics which deals with body motion on the inclined plane tied together with another body by rope over the block. The simulation which has been realized allows to change the parameters: weight of both bodies, friction coefficient, the inclined plane angle and investigate consequences of the changes. The thesis is innovative in the way that the simulation allows to get experimental data to the student. In the theoretical part of the thesis we deals with simulations and their classification within e-Learning. We will have a look at particular programming languages

and environments. We account for the choice of Easy Java as evolutionary environment used in the empirical part of our thesis. A part of it is also the survey on source code of Open Source. In the empirical part we present the proposal and concrete procedure to create the interactive simulation with four optional parameters in Easy Java environment, followed by calculations based on Euler-Richardson's method.

Keywords: e-Learning, Easy Java, creation interactive simulation, Open Source, programming languages.

Motto:

*„Počet chyb vzrůstá kvadraticky s počtem kliknutí.“*

Beneda David

*„Proč je hudba krásnější než život?“*

Sikora Jan

**Prohlašuji, že**

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

**Prohlašuji,**

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

.....

podpis diplomanta

**OBSAH**

<b>ÚVOD</b> .....	<b>9</b>
<b>I TEORETICKÁ ČÁST</b> .....	<b>10</b>
<b>1 INTERAKTIVNÍ SIMULACE A E-LEARNING</b> .....	<b>11</b>
1.1 E-LEARNING .....	11
1.1.1 Definice e-learningu .....	11
1.1.2 E-learning a jeho přínos pro výuku .....	12
1.1.3 Výukové systémy v rámci e-learningu .....	13
1.2 POSTUP PŘI TVORBĚ DIDAKTICKÉHO SOFTWARE .....	14
1.3 ZAŘAZENÍ SIMULACÍ DO E-LEARNINGU A JEJICH VYUŽITÍ.....	16
<b>2 PROBLEMATIKA INTERAKTIVNÍCH SIMULACÍ</b> .....	<b>18</b>
2.1 DEFINOVÁNÍ PROBLEMATIKY SIMULACE - APPLET .....	18
2.1.1 Simulace jako vědecká metoda .....	19
2.1.2 Počítačová simulace .....	19
2.1.3 Zhodnocení metody počítačové simulace .....	20
<b>3 VÝVOJOVÁ PROSTŘEDÍ INTERAKTIVNÍCH SIMULACÍ A OPEN SOURCE</b> .....	<b>23</b>
3.1 PŘEHLED A ZHODNOCENÍ VÝVOJOVÝCH PROSTŘEDÍ PRO TVORBU INTERAKTIVNÍCH SIMULACÍ .....	23
3.1.1 Přehled některých programovacích jazyků.....	23
3.1.2 Simulační programovací prostředí .....	26
3.2 OPEN SOURCE A JEHO VYUŽITÍ .....	27
3.2.1 Teoretický úvod a význam Open Source.....	27
3.2.2 Úprava Open Source simulací vytvořené v prostředí Easy Java.....	29
3.3 VÝBĚR VHODNÉ TECHNOLOGIE PRO TVORBU INTERAKTIVNÍ SIMULACE.....	30
<b>II PRAKTICKÁ ČÁST</b> .....	<b>32</b>
<b>4 NÁVRH A TVORBA KONKRÉTNÍ SIMULACE</b> .....	<b>33</b>
4.1 NÁVRH A STRUKTURA APLIKACE .....	33
4.1.1 Uživateli přístupné proměnné .....	33
4.1.2 Návrh vzhledu .....	34
4.1.3 Interface a okna .....	35
4.1.4 Fyzika modelu .....	37
4.1.5 Zasazení do reálného prostředí.....	40
4.2 POPIS TVORBY .....	41
4.2.1 Seznámení se s prostředím .....	41
4.2.2 Tvorba interface a oken.....	45
4.2.3 Tvorba grafů a tabulky .....	48
4.2.4 Tvorba jádra appletu.....	51
<b>5 ZHODNOCENÍ APLIKACE A JEDNOTLIVÝCH ČÁSTÍ</b> .....	<b>60</b>
<b>ZÁVĚR</b> .....	<b>61</b>

<b>ZÁVĚR V ANGLIČTINĚ.....</b>	<b>63</b>
<b>SEZNAM POUŽITÉ LITERATURY.....</b>	<b>65</b>
<b>SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....</b>	<b>68</b>
<b>SEZNAM OBRÁZKŮ.....</b>	<b>69</b>
<b>SEZNAM PŘÍLOH.....</b>	<b>70</b>

## ÚVOD

Interaktivní fyzikální simulace. Co to je? K čemu slouží? Užitečná pomůcka použitelná pouze k výuce, nebo nenahraditelný pomocník při výzkumech a jiných experimentech? V jakém programovacím jazyku je tvořit a jak by taková interaktivní simulace měla správně vypadat? Těmto a mnoha jiným otázkám se v této práci budeme věnovat.

Interaktivní simulace je rozsáhlá problematika sama o sobě a vzhledem k tomu, že simulovat můžeme takřka jakýkoliv reálný, ale i smyšlený model skýtá toto téma řadu možností ale i problémů. Díky nim můžeme prozkoumat úskalí jak přírodních věd, ale také „nepředvídatelné“ a rozmanité chování zvířat i lidí. Z důvodu složité problematiky některých modelů jsou tyto simulace náročné na zkušenosti programátora, jenž musí mít takřka dokonalou znalost vývojového prostředí, v kterém jej tvoří, ale také by měl rozumět dané problematice, která s tvorbou simulace souvisí. Vzhledem k tomu, že budeme pojednávat o interaktivních fyzikálních simulacích jakožto o „appletu“, jenž se využívá především při výuce, tak se nevyhneme ani tématu e-learning.

E-learning je v dnešní době oblíbený a stále se rozvíjejícím odvětví, které využívá dnešních už snadno dostupných informačních technologií ke zdokonalení výuky, tak aby se student naučil z probírané látky co nejvíce a zefektivnil svůj vzdělávací proces.

Interaktivní simulace je, dá se říct „jen“ program, jenž může být napsán v různých programovacích prostředích. Pokusíme se jim porozumět a také přijít na to, které by mohlo být pro tvorbu appletu nejvhodnější. Jejich použití se však může měnit a pro ušetření času s jejich opětovným programováním někteří programátoři umožňují přístup ke své práci pod hlavičkou „open source“, takže je možné daný program upravit pro vlastní potřeby. V praktické části se pustíme do tvorby už konkrétní simulace konkrétního modelu v prostředí Easy Java za využití zmíněných informací.

Závěrem možná sumarizovat a definovat vytyčené cíle předložené práce následovně:

- definovat pojem interaktivní simulace, resp. applet a jejich zařazení do e-learningu,
- nastudovat a popsat technologie vhodné pro tvorbu interaktivních simulací,
- vybrat nejvhodnější technologii pro interaktivní simulace s fyzikálním zaměřením,
- návrh a realizace interaktivní simulace „Pohyb tělesa po nakloněné rovině spojeného lanem přes kladku s druhým tělesem“ ve vybraném prostředí.

## **I. TEORETICKÁ ČÁST**

## 1 INTERAKTIVNÍ SIMULACE A E-LEARNING

Jak již bylo naznačeno v úvodu, tak cílem této práce je vytvoření konkrétní simulace. Ty jsou tvořeny buďto za účelem experimentů, při kterých dochází k simulaci určitých dějů nebo činností. Proč daný jev simulovat, tak na to může existovat spousta možných odpovědí, jako například peníze, prevence či poznání. Tato práce se však ubírá směrem, jejíž výsledkem je simulace, která by měla být použitelná při výuce či vzdělávání. Tím se dostáváme k prvnímu bodu práce a to k e-learningu.

### 1.1 E-learning

E-learning je vzdělávací metoda nebo proces, využívající informační a komunikační technologie ke zefektivnění vyučovacího procesu. Například přispívá k lepší komunikaci mezi studenty a pedagogy, a také dokáže lépe upoutat studentovu pozornost, ale o tom až dále.

E-learning se zatím spojuje hlavně s osobními počítači, ale také se může jednat například o interaktivní tabule apod. V tomto však narážíme na různé definice e-learningu, kdy použití zmiňované interaktivní tabule nemusí spadat pod e-learning. Díky rozvoji nových technologií, jakožto výkonných komunikačních prostředků, jako jsou kapesní či osobní počítače či organizéry, ale také nová generace mobilních telefonů, které umožňují připojení k internetu, tak začínáme hovořit i o m-learningu – mobilním vzdělávání.

M-learning (mobile learning), jenž je díky dostupnějším moderním technologiím trendem spíše posledních pár let, i když je pravdou, že většina populace tuto techniku doteď používá spíše k zábavě / komunikaci než k výuce. Dá se říci, že když cestou do školy či do zaměstnání pročítáte v mobilu PDF soubor, z něhož se snažíte něco naučit, tak vlastně provozujete m-learning.

#### 1.1.1 Definice e-learningu

Vzhledem k nepřetržitému dynamickému vývoji e-learningu (od 60. let minulého století) v různých dobách se dost často definice výrazně liší. Tou nejjednodušší je asi „*E-learning je výuka s využitím výpočetní techniky a internetu.*“ (KORVINY, Petr. OPF. Moodle na OPF, 2005.). Z různých definic mě osobně na dnešní dobu přijde nejpřesnější "*E-learning je vzdělávací proces, využívající informační a komunikační technologie k tvorbě kursů, k*

*distribuci studijního obsahu, komunikaci mezi studenty a pedagogy a k řízení studia.“* (WAGNER, Jan. Nebojme se eLearningu. *Česká škola*. 2005.).

Ze všech definic ale vyplývá, že e-learning je „metoda“ výuky, jenž využívá informačních technologií a zahrnuje řadu dílčích aktivit, které mohou být propojené. To znamená, že se může jednat jak o rozsáhlé kurzy a stejně tak o doplněk výuky.

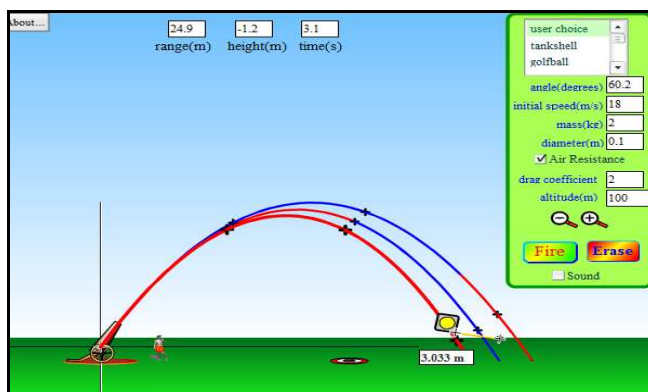
### **1.1.2 E-learning a jeho přínos pro výuku**

V rámci přírodních věd vzniklo v poslední době velké množství didaktického softwaru na podporu výuky matematiky, fyziky, biologie, chemie, dějepisu, geografie, ale i v rámci humanitních věd jako například angličtina nebo němčina.

I přes to je zapojení osobních počítačů do výuky na středních školách, v těch lepších případech na nízké až střední úrovni. Toto je dáno jednak místními podmínkami (zda je daná škola těmito prostředky na výuku vybavena) a také, zda ji učitelé dokáží naplno využít a tím přiblížit studentovi problematiku dané látky za pomoci těchto prostředků. Samozřejmě, to na jaké úrovni je využívána výpočetní technika k výuce záleží také na typu střední školy.

Vesměs všechny výukové programy jsou určeny uživateli osobního počítače. V případě jejich zapojení do vyučovací hodiny, tedy pro žáky, kteří podle pokynů objevujících se na obrazovce sledují animace, čtou texty (pokyny) a změnou vstupních proměnných mění průběh modelování dějů. Opakuje si simulace dle vlastní potřeby, odpovídá na otázky atd. Dá se říci, že tento druh výuky využívá hravosti a dedukční schopnosti studenta. Přece jen, když student na daný jev přijde sám, tak si z výuky odnese mnohem více a je také větší pravděpodobnost při kombinaci s přednášenou látkou, student danou problematiku pochopil správně.

Tohoto principu se snaží dosáhnout většina simulací. Jako příklad uvedu applet „Pohyb projektilu - Projectile Motion“, ze stránek <http://phet.colorado.edu>, jenž znázorňuje dráhu vystřeleného projektilu z děla (Obr. 1). Za pomoci proměnných jako např. počáteční rychlost, vzdálenost od terče, úhel a jiných, si může student vypočítat správné nastavení těchto proměnných a následně vyzkoušet správnost výpočtu (zda se trefí do cíle).



Obr. 1. Titulní strana simulace „Projectile Motion“ <http://phet.colorado.edu>

Možnost využití počítače při výuce je určeno vybavením učebny. V podstatě existují dvě varianty vybavení učebny:

- 1) Učebna obsahuje PC pro každého žáka, takže každý pracuje sám na svém PC. V tomto případě je lepší využití didaktického softwaru. Tento způsob výuky se hodí například pro fyziku, biologii a geografii. V dnešní době je však takovýto učeben minimálně, pokud se nejedná o učebnu informatiky. Z hlediska poměru efektivity ku ceně se dává přednost druhé variantě.
- 2) Učebna vybavena jedním PC spolu se zobrazovací technikou jako např. projektor nebo interaktivní tabule. V tomto případě studenti nemají tu možnost manipulace s aplikací, ale jsou jen pozorovatelé. Dle mého názoru se jedná o nejlepší možnost výuky předmětů, pokud vezmeme v potaz finanční náklady obou možností a přitom skýtá možnosti, jenž „klasické“ učebny nemají. Je pravdou, že využití této technologie je náročnější na čas a zkušenosti vyučujícího, ale plné využití možností takovéto učebny posouvají úroveň vzdělání o krok vpřed. Novinkou u interaktivních tabulí je použití interaktivního hlasového systému, jako je například „Senteo“, který vytváří přímé propojení mezi učitelem a studenty a dává okamžitou zpětnou vazbu. Tento intuitivní hlasový systém zahrnuje bezdrátový ovladač pro každého studenta, centrální přijímač a výkonný software, okamžitě vyhodnocuje odpovědi studentů. I toto je vize pro školy.

### 1.1.3 Výukové systémy v rámci e-learningu

Tyto systémy jsou velmi rozšířené a podle míry využití jednotlivými vzdělávacími institucemi snad už každodenní součástí života běžného studenta, aniž by si to třeba uvědomoval. To můžu potvrdit sám ze své zkušenosti.

Může se jednat například o „Learning Management System“ (LMS), jenž je řídicí výukový systém, tedy aplikace řešící administrativu a organizaci výuky v rámci e-learningu. LMS jsou aplikace, které v sobě integrují nejrůznější on-line nástroje pro komunikaci a řízení studia a zároveň zpřístupňují studentům učební materiály, či výukový obsah on-line nebo i off-line. Řada LMS je šířených jako free nebo Open Source software (například Moodle, což je balíček pro tvorbu výukových systémů a elektronických kurzů na internetu).

Běžné funkce každého takového systému by měli být:

- evidence a správa žáků,
- evidence a správa kurzů,
- katalog výukových kurzů a objektů,
- správa studijních plánů,
- evidence hodnocení žáků,
- testování a přezkušování žáků,
- správa přístupových práv,
- komunikační nástroje,
- autorské nástroje k vytváření výukových kurzů a objektů,
- úložiště výukového obsahu.

Pro všechny tyto funkce je důležitý požadavek na jejich přenositelnost a standardizaci. LMS by měl být otevřený a schopný například snadno a rychle začlenit výukový obsah, vytvořený například před zavedením LMS.

## 1.2 Postup při tvorbě didaktického softwaru

Při tvorbě didaktického softwaru je potřebné uvědomit si, pro který typ školy a vyučovací hodiny má být daný software navržen, spolu s tím, že by měl navazovat na předcházející hodiny. Hlavní problematikou je navrhnout takový program, jenž by přesně pasoval do výuky na konkrétním typu školy.

Takovýto software by měl splňovat následující podmínky:

- 1) Prezentace by měla být vyplněna grafickými modely, náčrtky, grafickými znázornění grafických veličin (animace, grafy, videozáznamy, fotografiemi) v logicky uspořádaném struktuře dané látky, vycházející ze vzájemné spojitosti faktů, což je základem trvalých vědomostí a paměťové kapacity.
- 2) Prezentace musí umožňovat žákovi osvojit si nové poznámky v maximální míře s využitím samostatného myšlení (dedukce), aktivním uplatněním u dříve osvojených poznatků. Grafické modely, videozáznamy a video-animace na zprostředkování diskuze mezi vyučujícím a žákem.
- 3) Při diskuzi očekává učitel navrhování správných řešení při pozorování daných problémů. Po dotazování ke správnému výsledku se žákovi potvrdí jeho úsudek v písemné formě, takže žák nejen že zjistí správný výsledek, ale vidí také písemnou formulaci.
- 4) Fyzikální experiment, resp. přírodovědný a technický, má velký motivační účinek a výrazně zvyšuje pozornost, aktivitu a zaměří tak pozornost žáka určitým směrem. Součástí by měli být grafické modely podporující uskutečnění fyzikálního experimentu (schéma zapojení, modelový náčrt experimentu, atd.).
- 5) Pro zopakování a zpětnou vazbu učiteli, by měla prezentace obsahovat stručný test.
- 6) Další součástí každé prezentace by měl být podrobný metodický postup vyučující hodiny, který obsahuje:
  - poznatky z vyučovací hodiny,
  - specifické cíle hodiny,
  - seznam experimentů uskutečněných na vyučovací hodině,
  - zobrazení jednotlivých snímků prezentace s návrhem a popisem a návrhem možného využití snímků v hodině. <sup>[9]</sup>

Co se prezentací týče, tak jeden z nejpoužívanějších programů pro jejich tvorbu je program „PowerPoint“ od společnosti Microsoft. Tyto prezentace se skládají ze sérií snímků a umožňuje tvorbu jednotlivých animací, ale s minimální interakcí. Tyto snímky mohou obsahovat jak klasický text, tak obrázky, animace, video-animace nebo objekty jako grafy a

jiné. Výhodou prezentací je, že je lze snadno upravit a není potřeba znalosti jakéhokoliv programovacího jazyka.

### 1.3 Zařazení simulací do e-learningu a jejich využití

Simulace je jednou z nejsilnějších součástí e-learningových nástrojů. Tyto simulace umožňují interaktivní ověřování výkladu probírané látky a to umožňuje zefektivnění učícího procesu. Takže výuka probíhá navíc zábavnější formou, jenž upoutá pozornost studenta.

Oblasti, v kterých e-learningové simulace vynikají:

- zvyšování praktických znalostí,
- umožňuje přístup řešeného pohledu z jiného úhlu pohledu,
- zvyšování motivace k učení,
- vytváření nových dovedností,
- návrh tvůrčího řešení problému a jeho nácvik,
- trénink možných situací v reálném čase,
- vlastní studijní tempo,
- rychlé vyhodnocení dosažených výsledků a ona zmiňovaná úspora nákladů a času.<sup>[21]</sup>

Simulace se v dnešní době používají v řadě odvětví a oborů, pro nejrůznější použití. Jednotlivé simulace se od sebe mohou lišit jak složitostí tak cenou, od dětských výukových simulátorů, až po vojenské simulátory.

Využití simulací v praxi:

- Školství – ke zefektivnění výuky atd. – viz. výše.
- Logistika a výrobní strategie a procesy podniku o obchodu – hlavní využití v předvýrobních etapách, pro prověření různých variant řešení s cílem eliminovat rizika a zefektivnit například výrobu => optimalizace obchodních procesů, plánování a řízení výroby, projektování nových systémů, analýza výrobních systému atd.

- Zemědělství – v tomto odvětví se například využívají k tomu, aby farmáři zasadili takové plodiny, jenž by měli co nejvyšší poměr zisku ku nákladům.
- Vojenství – zbrojení je jedno z odvětví, ve kterém se nejvíce točí peníze a hlavně světové velmoci se snaží mít ve svém arzenálu tu nejmodernější technologii. To platí i o simulacích / simulátorech, kde zkušenosti nasbírané z těchto virtuálních konfliktů či možných situací pomáhá zlepšovat kvalifikaci daného jedince před nasazením do služby a tím například zvýšit jeho šance na přežití. Zde se využívají simulátory od týmové spolupráce, přes deaktivaci bomb, až po řízení nejmodernějších strojů.
- Letectví – letecké simulátory sloužící k výcviku jak pilotů letadel, tak i vrtulníků, popř. raketoplánů. Tyto simulátory jsou provedeny buď jako počítačové hry, nebo realistické simulátory, jejich součástí je reálná kabina daného stroje.
- Lékařství – jedná se například o simulátory simulují pohyby a tahy řezu skalpelem za pomoci speciálního hardwaru.
- Meteorologie – například seizmografie. Simulace chování se vnitřních vrstev Země a předpovídání zemětřesení a cunami vln.
- Přírodní vědy - bez simulací by prakticky nebylo možné učinit jakýkoliv výzkum týkající se biologie, chemie, fyziky. Genetické inženýrství, farmaceutický, automobilový průmysl, zkoumání reakcí chemických látek apod. To vše se přímo opírá o předem nasimulované modely.
- Doprava - stejně jako je tomu u letecké dopravy, tak i u té pozemní ať osobní či nákladní existují trenažéry jak automobilové, vlakové, městské hromadné dopravy, na kterých si člověk osvojuje základní znalosti potřebné k tomu, aby dostal řidičské oprávnění nebo se zdokonalil. Krom těchto využití, je také možné za pomoci simulací navrhovat nejefektivnější trasy pro přepravu.

## 2 PROBLEMATIKA INTERAKTIVNÍCH SIMULACÍ

### 2.1 Definování problematiky simulace - applet

Co je to simulace? Asi nejvšeobecnější jsou synonyma předstírat a napodobňovat. Simulaci tedy můžeme definovat jako imitaci reálných věcí, stavů, vztahů nebo procesů, nebo také jako „*numericou metodu, která spočívá v experimentování se speciálním matematickým modelem reálných systémů na počítači*“ (What is Simulation. *SIMUL8 Corporation Products*. 2006.).

Pokud hovoříme o appletu, tak applet je software, jenž je spuštěn v rámci jiného programu, ať už se jedná o webový prohlížeč nebo o prezentaci. Bývá používán k plnění konkrétní funkce v rámci daného kontextu a nepředpokládá se u něj, že bude používán jako samostatná aplikace. Souvislost mezi simulací a appletem (pokud vezmeme v potaz zmíněné informace) je ta, že prostřednictvím appletu můžeme provést interaktivní znázornění dané simulace. Název applet vznikl na základě významu, že se jedná o „malou aplikaci“ to bylo odvozené ze zdvojnásobení – let v anglickém jazyku, app = (application) + let.

Applet se od klasického programu liší těmito vlastnostmi:

- 1) Spouští se v klientské formě.
- 2) Hostitelský program mu umožňuje přístup jen k určitým funkcím.
- 3) Může být napsán v odlišném jazyce než kontejner.

Webové stránky využívající aplikace napsané ve skriptovacím jazyce, která běží v daném prohlížeči není považována za applet.

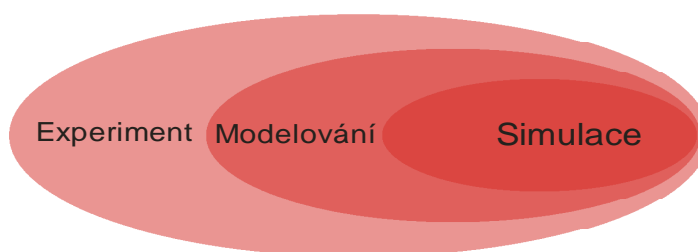
Vzhledem k tomu, že zde pojednáváme o simulacích v rámci e-learningu realizované za pomoci appletů určené hlavně pro výuku, tak musíme zmínit čím by takováto simulace měla být charakteristická:

- 1) Vyučovanou látku zasazuje do prostředí reálného života.
- 2) Při simulaci daného modelu odpadá nebezpečí možné z přímého kontaktu tomuto jevu, takže i odpadají náklady na případné pořízení experimentu.
- 3) Využívá odezvy k vysvětlení chyb, kterých se student dopustil.

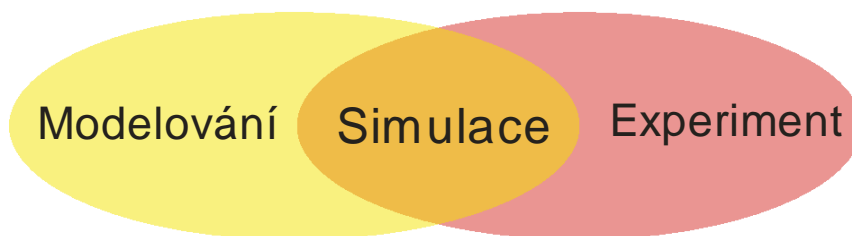
### 2.1.1 Simulace jako vědecká metoda

Simulace patří mezi vědecké metody, protože představuje určitý způsob získávání poznatků o světě. Simulace, tak jak ji definuje většina zdrojů, pracuje s určitým modelem, tedy idealizovanou podobou reálného světa. Tento fakt samozřejmě nevylučuje možnost získávání poznatků o světě reálném.

Postavení simulace v rámci všeobecných vědeckých metod lze zařadit jakožto podmnožinou „modelování“ a modelování jako podmnožinou „experimentu“. Tuto představu prezentuje následující obrázek (Obr. 2). V případě pokud by jsme modelování chápali pouze jako proces tvorby modelu, mohli by jsme simulaci chápat jako průnik „experimentu“ a „modelování“ (Obr. 3).<sup>[5]</sup>



Obr. 2. Zobrazení postavení simulace v rámci všeobecných vědeckých metod.<sup>[5]</sup>



Obr. 3. Zobrazení postavení simulace, pokud je modelování chápáno pouze jako proces tvorby modelu.<sup>[5]</sup>

### 2.1.2 Počítačová simulace

Zobrazení reality za pomoci modelu a pak následně realizovat experiment prostřednictvím počítačové simulace není snadné. Člověk realizující tento projekt musí mít znalosti nejen ze studijního oboru, ale musí ovládat i matematické, statistické a informační metody a postupy. Ani to však není zárukou, že daná počítačová simulace zachytí skutečnou realitu do poslední maličkosti. Výsledný model v počítačové podobě se nakonec stává

programem, jenž měl s největší pravděpodobností zachytit různé situace možnosti daného experimentu.

Využití počítačové simulace:

1) V kontextu matematicko-statistických postupů:

- Řešení analyticky neřešitelných diferenciálních rovnic v oborech, kde jsou předmětem výzkumu dynamické systémy (jaderná fyzika, astronomie, kinetika, chemie, klimatologie, elektrotechnika a jiné).
- Statická simulace diskrétních systémů, v kterých se události vyskytují s určitou pravděpodobností, a tyto systémy nepopisujeme diferenciálními rovnicemi (aplikace v genetice, biochemii, medicíně a pod.).

2) Konkrétní aplikace:

- Meteorologické odhady rovnovážných stavů určitých veličin a studování dynamiky jejich vývoje.
- V ekonomických odhadech znečištění životního prostředí.
- Hydrologické transportní modely využívané k předpovědi kvality vody.
- Teoretické modely zkoumající lidskou výkonnost.
- Simulace vývoje bílkovinných struktur.
- Simulace chování částic atomů ve fyzice, jaderné fyzice a chemii.
- Výpočetní technika a různé druhy softwaru. <sup>[5]</sup>

A mnoho jiných.

### 2.1.3 Zhodnocení metody počítačové simulace

Sumarizujeme v podobě výhod a nevýhod počítačovou simulaci:

Výhody:

1) Umožňuje řešit analyticky neřešitelné úlohy.

Metoda počítačové simulace v některých případech pomáhala realizovat všeobecnou vědeckou metodu jakou je analýza. Především v oblasti vztahové a strukturální

genetické analýze je možné za pomoci simulace odhalit složité závislosti, tak i dynamiku systému a reakce na různé události.

- 2) Ulehčuje řešení těžko řešitelných úloh.
- 3) Standardizuje některé kvalitativní parametry.
- 4) Proces tvorby modelu napomáhá k odhalení a specifikaci nových informací.

Detailní pozorování a popis entity nebo systému, za účelem tvorby modelu pro počítačovou simulaci, vede často ke zjištění nových informací zkoumaného systému. Tímto se zpřesní výsledky jak simulace, tak dojde k přesnější konstrukci výsledného modelu.

- 5) Rozšíření predikce i na oblasti ve kterých nejde konkrétně využít deduktivních, či deterministických metod.

Deduktivní usuzování a použití přísných deterministických modelů není využitelné všude tam, kde se v systému vyskytují náhodné proměnné. Pomocnou metodou počítačové simulace můžeme modelovat systémy s jednou, či více náhodnými proměnnými. V simulaci je možné předefinovat hodnoty závislých proměnných, ne však s absolutní přesností, jako v případě deterministických modelů, ale pouze s jistou pravděpodobností.

Nevýhody:

- 1) Všeobecné problémy spojené s využitím modelu.

Model je svým způsobem určité zjednodušení reality. Je tedy důležité, aby vytvořený model, nezjednodušoval a neskresloval vlastnosti systému, jenž jsou pro počítačovou simulaci důležité, tak aby simulace byla co nejdůvěhodnější.

- 2) Možnost selhání v procesu simulace extrémních událostí.

Tvorba modelu i celé nastavení parametrů procesu simulace bývá většinou založené na historických údajích, nebo odhadu analytika. Právě to může opomenout, nebo nedostatečně zachytit události, které mají povahu extrémních šoků. Simulace tak nemusí výskyt „extrémů“ zachytit vůbec, nebo zkreslí jejich počet a nebo simulace nebude moct na danou otázku vůbec odpovědět.

## 3) Opakování.

Charakterem vědecké metody je fakt, že při opakování konkrétní simulace je vždy stejný výsledek. Tato vlastnost však nemusí být vždy splněná s ohledem na náhodné proměnné reálném procese, kde se vyskytují různé poruchy a jiné vlivy.

### **3 VÝVOJOVÁ PROSTŘEDÍ INTERAKTIVNÍCH SIMULACÍ A OPEN SOURCE**

Název této kapitoly naznačuje, že se zde budou popisovat pouze prostředí, ve kterých lze tvořit pouze simulace. Ve skutečnosti je těchto prostředí pouze hrst oproti všem ostatním, ve kterých by uvažovaná simulace šla naprogramovat, pouze s tím rozdílem, že ono prostředí se nespécializuje pouze na programování simulací. Pokusím se vám zde nastínit alespoň zlomek toho, v čem všem by mohla jít interaktivní simulace naprogramovat a který programovací jazyk by na tom mohl být lépe oproti jiným.

#### **3.1 Přehled a zhodnocení vývojových prostředí pro tvorbu interaktivních simulací**

Prostředí pro tvorbu interaktivních simulací a appletů je z velké míry zavádějící, protože applety se dělají hlavně v Javě, nebo spíše na její platformě. I když virtuálnímu stroji spouštějící výsledný soubor s příponou „JAR“ je jedno, v kterém programovacím prostředí vznikl. To, proč se Java používá na tvorbu appletů má svůj význam, ale o tom se zmíníme až dále.

Před psaním této části práce jsem si pročítal jednotlivá fóra, debatující o tom, který programovací jazyk je nejlepší a jaké má přednosti. Co měla většina fór společného je to, že mezi jednotlivými programátory pakuje podobná rivalita, jako mezi uživateli používající operační systém Linux a Windows.

V každém případě, pokud hovoříme o interaktivních simulacích, tak prakticky nehovoříme o ničem jiném, než o programu, jakožto o jeho funkci. Takže interaktivní simulaci můžeme stvořit snad ve všech programovacích jazycích / prostředích, prakticky jde jen o to, v kterém lze simulaci vytvořit snadněji, rychleji nebo lépe.

##### **3.1.1 Přehled některých programovacích jazyků**

Následující informace pocházejí hlavně ze serverů, nabízející seriály a užitečné rady k nim. Problémem je to, že v praxi je nemožné, aby jeden člověk plně ovládal všechny programovací jazyky, takže jedinou mojí možností bylo použití informací od lidí, kteří mají v jednotlivých jazycích větší zkušenosti než já.

## Jazyk C

Jazyk, který neodmyslitelně patří k unixu. Vytvořili jej Ken Thompson a Dennis Ritchie pro potřeby vývoje prvního unixu. C je jazyk podporující strukturované programování, umožňuje přímý přístup do paměti pomocí ukazatelů a aplikace v něm jsou velmi kompaktní. Hodí se na systémové i aplikační programování a navíc v něm bývají napsány interpretery vyšších jazyků. V C je napsáno například jádro Linuxu, nebo webový server Apache. <sup>[20]</sup>

## Jazyk Java

Jazyk vytvořil James Gosling. Jeho syntaxe vychází z jazyka C++. Je silně objektově orientovaný. Java dále poskytuje virtuální stroj s automatickou správou paměti, velice silnou základní knihovnou a multiplatformost. Jde o relativně jednoduchý jazyk, mající předpoklady například pro programy pracující mezi počítači, také díky její bezpečnosti a spolehlivosti. <sup>[20]</sup>

Jak už bylo zmíněno, tak Java je silně objektově zaměřený programovací jazyk, který neumožňuje tvorbu ukazatelů, které jsou z 90% důvodem pádu programů. Proto je také stabilnější. Zdrojový kód je při vývoji přeložen do spustitelného mezikódu (bytecode), který lze pak spouštět pomocí nainstalovaného runtime prostředí (Java Virtual Machine), přímo na různých typech počítačů, či technických zařízeních bez nutnosti nového překladu.

Nevýhodou Javy je její relativní „pomalost“, která je způsobena paměťovou náročností prostředí. Dalším důvodem je Swing, který vykresluje grafické prostředí plně v režii virtuálního stroje. Naproti tomu je zpomalení „minimální“ díky optimalizacím. Dále neumožňuje vytvoření kernelu (jádra) a pokud ano, tak jsem o tom nenašel žádnou zmínku.

Naproti tomu je Java silná v programování pro web, v rozsáhlých systémech, nebo naopak v malých zařízeních (mobilní telefony) a trochu méně v psaní běžných aplikací.

Ukázka zdrojového kódu „Hello World!“ v jazyce Java:

```
1 public class helloWorld
2 {
3     public static void main(String [] args)
4     {
```

```
5         System.out.println("Hello World!");
6     }
7 }
```

### Jazyk C++

Jazyk vytvořil Bjarne Stroustrup a který je mylně chápán jako verze C. C++ je odlišný jazyk s podobnou syntaxí a překladače C++ dokáží přeložit kód napsaný v C. Má širokou škálu využití, od nízko-úrovňového programování dědičnosti až po vysokoúrovňové, což z něj dělá jeden z nejsložitějších jazyků. Podporuje jak objektově orientované, tak i procedurální programování. Byl využit třeba na projekty jako jsou: Mozilla nebo OpenOffice.org. <sup>[20]</sup>

Ukázka zdrojového kódu „Hello World!“ v jazyce C++:

```
1  #include <iostream.h>
2  int main()
3  {
4      cout << "Hello World!";
5  }
```

### Jazyk PHP

Zasloužil se o počet linuxových serverů. Jazyk, který vytvořil Rasmus Lerdorf <sup>[20]</sup>, je určený primárně na web a vyvinul se z několika cgi skriptů napsaných v jazyce C, až do objektově orientovaného jazyka, kterým se stal ve verzi 5. V PHP je napsán phpMyAdmin, IMP, MediaWiki nebo Mambo.

Ukázka zdrojového kódu „Hello World!“ v jazyce PHP:

```
1  <?php
2  echo 'Hello World!';
3  ?>
```

## Jazyk Perl

Jazyk, který vznikl jako kombinace mnoha jazyků (C, shell, awk, sed, Lisp). Jeho autorem je Larry Wall a datum jeho vzniku leží už v roce 1987.<sup>[20]</sup> Dlouho patřil mezi nejoblíbenější skriptovací jazyky vůbec. Jeho největší síla leží v oblasti zpracování textu a definoval de facto standard pro regulární výrazy. Perl stál v počátku webového programování, kde jej nahradilo PHP. Jinak se hodí od jednoduchého skriptování po aplikační programy.

Ukázka zdrojového kódu „Hello World!“ v jazyku Perl:

```
1  #!/usr/bin/perl
2  print "Hello World.\n";
```

## Jazyk Python

Jazyk, jehož autorem je Guido van Rossum a který se z podoby jednoduchého procedurálního skriptovacího jazyka vyvinul do podoby jednoho z nejpoužívanějších dynamických jazyků na světě.<sup>[20]</sup> V současné době je šířen jako Open Source. Obsahuje rozsáhlou standardní knihovnu, jejichž části vznikali nezávisle na sobě.

Charakteristika jazyka: interpretovaný, vysokoúrovňový, s objektově orientovaným programováním a dynamickými typy. Velice populární je jako jazyk pro skriptování aplikací GIMP, OpenOffice.org, Blender a jiné.

Ukázka zdrojového kódu „Hello World!“:

```
1  print "Hello world!"
```

### 3.1.2 Simulační programovací prostředí

Simulační programovací prostředí, je vlastně označení takových prostředí, které poskytující prostředky usnadňují efektivní popis modelů, struktury modelů nebo simulačních experimentů.

#### Použití simulačních programovacích prostředí

Tyto programovací prostředí jsou použity například pro:

- programování simulačních modelů,

- práci s abstraktními systémy,
- vizualizace a vyhodnocování výsledků,
- experimentování se simulačními modely.

### **Dělení simulačních programovacích prostředí**

Simulačních programovacích prostředí existuje celá řada. Můžeme je dělit dle jejich společných rysů, charakteru do několika skupin. Takto je dělíme na:

- **diskrétní** - pohlížejí na model jako na sled náhodných událostí způsobující změnu stavu. Vyznačují se použitím přesně definovaných a matematicky ohodnocených událostí. Důležitou součástí jazyků založených na diskretních událostech je schopnost generovat pseudo-náhodná čísla a proměnné z různých rozdělení pravděpodobnosti. Mezi tyto programy patří například: Arena, GASP, Simprocess, Simio software a jiné.
- **spojité** – pracuje s modelem na základě diferenciálních rovnic. Mezi tyto programy patří: DYNAMO, SimApp, VisSim a jiné.
- **kombinované** – mezi tyto programovací prostředí patří Easy Java, která je praktickou částí této práce. Easy Java je postavená na základě jazyku Java a jedná se o Open Source projekt. Samozřejmě se nejedná o jediné kombinované prostředí.

## **3.2 Open Source a jeho využití**

Nejednou zde bylo řečeno slovní spojení „Open Source“ v návaznosti na to, že jsem používal jak program, tak jednotlivé programy takto označené při realizaci praktické části této práce. Proto by bylo vhodné se vyjádřit k tomu, co to vlastně je.

### **3.2.1 Teoretický úvod a význam Open Source**

Stručně řečeno Open Source software podle definice Open Source Initiative je otevřený software jehož zdrojové kódy je možno rozšiřovat, modifikovat a používat bez omezení, pouze se zachováním této otevřenosti. Open Source programy jsou vyvíjeny dobrovolnými vývojáři, jejichž hlavní motivací je psát a sdílet kvalitní a rozšiřitelný software.

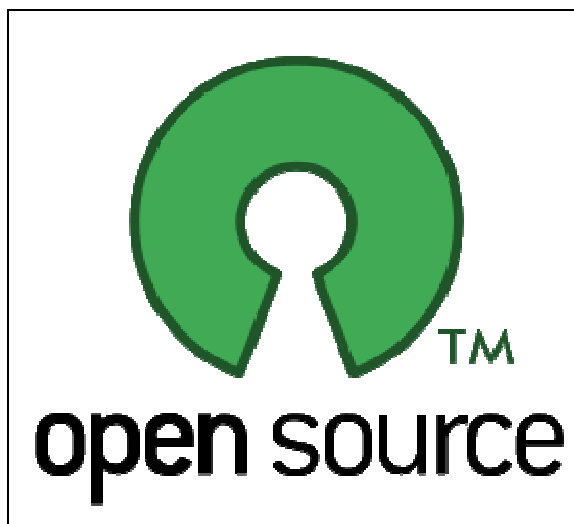
Termín Open Source software, je poměrně nový a intenzivněji začal být používán až po roce 1998. Jedním z důvodů jeho zavedení do praxe je zejména nejednoznačnost anglického slova „free“ v pojmu free software. Slovem free se totiž poměrně často označují počítačové programy, které nesplňují znaky svobodného software, avšak jsou poskytovány zdarma. Každý Open Source software lze označit za svobodný, ale naopak každý svobodný software nelze označit za Open Source. <sup>[12]</sup>

Aby mohl být určitý počítačový program označován jako Open Source, tak musí dle Open Source Initiative (<http://www.opensource.org/docs/definition.php>) splňovat tyto předpoklady:

- Volná distribuce - Licence nesmí omezovat jakoukoli stranu při prodeji nebo rozdávání softwaru jako součásti souhrnné softwarové distribuce, která obsahuje programy z řady různých zdrojů. Licence nesmí vyžadovat autorský honorář, ani žádný jiný poplatek za takovýto prodej.
- Zdrojový kód - Program musí obsahovat zdrojový kód, a musí umožňovat distribuci ve zdrojovém kódu i zkompileované formě. Zdrojový kód musí mít preferovanou formu, ve které by programátor upravil program. Záměrně zatajený zdrojový kód není povolen.
- Odvozené práce - Licence musí povolovat úpravy a odvozená díla musí umožňovat jejich distribuci pod stejnými podmínkami jako licence původního softwaru.
- Integrita autorského zdrojového kódu - Licence může omezovat zdrojový kód v distribuované, upravené formě pouze tehdy, pokud je licence umožňuje distribuci „patch souborů“ se zdrojovým kódem, za účelem úpravy programu při jeho sestavování.
- Zákaz diskriminace osob nebo skupin - Licence nesmí diskriminovat žádnou osobu nebo skupinu osob.
- Nesmí dojít k diskriminaci v oblasti užití - Licence nesmí omezovat nikoho od využití tohoto programu v konkrétní oblasti.
- Distribuce licence - Práva spojená s programem se musí vztahovat na všechny, jimž je program dále distribuován bez nutnosti provedení další licence těmito stranami.

- Licence nesmí být je pro výrobky - Práva spojená s programem nesmějí záviset na tom, že je program součástí konkrétní distribuce softwaru. Pokud je program vyňat z této distribuce a používán nebo distribuován v rámci podmínek licence daného programu, všechny strany, jimž je program dále distribuován by měly mít stejná práva jako ty, které jsou poskytovány ve spojení s původní distribucí softwaru.
- Licence musí být neutrální z pohledu použité technologie - Ustanovení licence nesmí být založena pouze na jednotlivé technologie nebo na stylu rozhraní.

Toto jsou oficiální informace ze stránek <http://www.opensource.org/docs/definition.php>., odkud je i přebraný obr. 4.



*Obr. 4. Logo Open Source Initiative.*

Z hlediska bezpečnostních děr v software je otevřenost kódu dvojsečná zbraň. Chyby v programech může hledat mnohem širší skupina lidí a je proto naděje, že se snáze opraví. Na druhou stranu zranitelnosti mohou snáze najít i útočníci a teoreticky ji mohou také vytvořit.

Některé úspěšné projekty založení jako Open Source: Apache (webový server), DNS (domain name server), Sendmail (poštovní server), PostgreSQL (databázový server) a jiné.

### **3.2.2 Úprava Open Source simulací vytvořené v prostředí Easy Java**

Možnost prohlížení si a možné následné úpravy zdrojového kódu je vždy výhodou pro každého, kdo nechce začínat od začátku a trápit se s otázkou „Jak to jenom udělali?“. Dá se říci, že na tomto základu je postavena jakákoliv výuka, protože kdyby každá generace měla

znovu objevovat něco, co je té minulé dobře známé, tak by se lidi nikdy nedostali tam, kde jsou dnes. Nakonec nikdo nemáme nekonečně času a proto, kdo chce něčeho dosáhnout s ním musí nakládat co nejefektivněji.

Open Source tuto možnost umožňuje a dá se říci, že nebýt jí, tak nikdy bych sám nezrealizoval praktickou část této práce. Samozřejmě jsem ji začal tvořit od začátku a neinovoval jsem už hotový projekt, jen jsem při tvorbě využil znalostí z jiných Open Source kódů.

Hovoříme-li o úpravě již vytvořených simulací, tak určitě musíme mít k dispozici prostředí v kterém je lze upravovat. V našem případě se jedná o Easy Javu, která je k dispozici ze oficiálních stránek <http://www.um.es/fem/EjsWiki/>. Na této stránce najdete rovněž několik již vytvořených simulací, které jsou k dispozici jak ve formě JAR souboru, tak ve formě zdrojového kódu. Samozřejmě je zde i nastíněna funkce Easy Javy a jejich jednotlivých částí. Navíc obsahuje jednoduché miniprogramy, znázorňující funkci jednotlivých objektů a jejich naprogramování ve formě stažitelného zdrojového kódu.

Za zmínku určitě stojí server <http://www.compadre.org/osp/index.cfm> (Open Source Physics), který vznikl na popud toho, že jen málo studentů je dobře připraveno pracovat s nimi a vyhodnocovat simulace. Dle jejich slov uvedených na jejich stránkách: „*Absence výpočtu a modelování je jedním z nejvýraznějších příkladů našich selhání aktualizace učebních osnov.*“. Tímto serverem chtějí tento výpadek řešit. Server obsahuje simulací označené a přístupné jako Open Source.

Co se samotné úpravy zdrojového kódu týče, tak jde jen o to, otevřít zdrojový kód v Easy Javě a upravit jej k obrazu svému. Zdrojový kód je možno upravit podle instrukcí v praktické části této práce. Jak se dále dočtete, tak úprava takového programu je relativně jednoduchá a také rychlá díky rozdělení programu do několika skupin.

### 3.3 Výběr vhodné technologie pro tvorbu interaktivní simulace

Teorie programovacích jazyků říká, že všechny jazyky jsou stejně silné. Jde jen o to, v kterém z nich je daný problém snáze řešitelný. V našem případě se jedná o interaktivní simulace. Pokud bychom však řešili otázku appletů, tak by jsme nemluvili o programovacím jazyku, ale o konkrétním prostředí.

Z mého pohledu bude tato část spíše zaujatá, vzhledem k mým zkušenostem, které v současné době nejsou na takové úrovni, abych se mohl rovnat s někým, kdo programuje ve velkých společnostech několik desítek let.

Při výběru vhodné technologie si musíme stanovit priority, co je důležité aby simulace jakožto program měla. Pokud tedy chceme vytvořit „jen“ simulaci, tak by použitý jazyk neměl být zbytečně komplikovaný a složitý. Nakonec nejde nám o vytvoření umělé inteligence. Dále by nebylo od věci, kdyby výsledná simulace byla stabilní a dala se spustit na většině strojů (multiplatformost). Dokonce stabilitu vytvořené simulace a její funkčnost na libovolném PC musíme uvažovat jako prvořadý požadavek na simulaci.

Z mého pohledu je tedy vhodnější použití jazyku Java. Pokud chceme danou simulaci naprogramovat opravdu rychle, tak proč vždy začínat od začátku? Vývojové prostředí Easy Java je postaveno na základě Javy a díky objektové povaze jazyka obsahuje předem naprogramované objekty, které stačí do nového projektu jednoduše vkládat a upravovat je dle svých potřeb. Proto, aby se člověk naučil pracovat v Easy Javě není zapotřebí dokonalé znalosti jazyka. Postačí základy programování a základní znalosti jazyka Javy v kombinaci s Open Source softwarem. Začátky jsou vždy těžké, ale věřím tomu, že kdokoliv se špetkou dedukčních schopností Easy Javu zvládne a po úspěšném dokončení prvního modelu si řekne, že je to vlastně „easy“.

Jazyk C++ je sice jedním z nejrozšířenějších a nejuniverzálnějších jazyků vůbec. Lze v něm vytvořit „vše“ od her až po operační systémy. Díky této univerzálnosti je ale jedním z nejsložitějších jazyků vůbec. Proto mě osobně přijde jednodušší a logičtější Java se svým zpracováním vyjímaj a „čistě“ objektovým programováním, bez použití jakýchkoliv ukazatelů.

## **II. PRAKTICKÁ ČÁST**

## 4 NÁVRH A TVORBA KONKRÉTNÍ SIMULACE

V procesu tvorby simulace hraje velkou roli logika, ale stejně tak i představivost, pokud chceme aby výsledný produkt nebyl pouhou napodobeninou, ale unikátní ve všech aspektech, než jiné applety zabírající se podobnou tematikou.

Dá se říci, že celá tato práce se soustředí zejména kolem tvorby konkrétního appletu „správným“ způsobem, jenž by mohl být použitý jako pomůcka pro vysvětlení daného fyzikálního jevu při výuce, ale také pro samostatnou práci s ním. Co se druhé možnosti týče, tak jde o to, aby výsledný applet byl vybaven nějakou formou výstupu dat, díky němuž budou moci jeho uživatelé výsledné data dále zpracovávat jakýmkoliv jim potřebným způsobem.

Při tvorbě si nejprve musíme stanovit cíl. V našem případě se pokusíme realizovat applet, do výuky mechaniky a to: „Pohyb tělesa po nakloněné rovině spojeného lanem přes kladku s druhým tělesem“. Samozřejmě by jsme měli postupovat podle postupů zmíněných výše. Vzhledem k tomu, že náš applet tvoříme v prostředí Easy Java, tak tomu také musíme postup přizpůsobit, i když díky flexibilitě a relativní jednoduchosti tohoto prostředí není postup tak striktní jako v jiných programovacích prostředích.

### 4.1 Návrh a struktura aplikace

Kde začít? Určitě si musíme stanovit přibližný vzhled naší aplikace a také parametry, které uživateli zpřístupníme, tak aby mohl tuto aplikaci ovládat a nastavovat pro různé situace. Tím se samozřejmě ze simulace stane interaktivní simulace.

#### 4.1.1 Uživateli přístupné proměnné

Dá se říci, že jádrem každého takového programu je vzorec, který řídí celou činnost, nebo také chování aplikace. To nás přivádí ke stanovení uživateli volitelných proměnných. Zde nám postačí čistá logika a základní znalosti z fyziky. Náš applet simuluje chování bedny na nakloněné rovině. Z toho vyplývají tři parametry: hmotnost bedny na nakloněné rovině, smykové tření – tření mezi bednou a rovinou a úhel naklonění této roviny. Zmíněná bedna je propojena lanem přes kladku se zavěšenou druhou bednou, takže získáme další parametr: hmotnost zavěšené bedny na laně přes kladku. Pro širší využití appletu jsem mezi tyto nastavitelné parametry přidal i gravitační zrychlení. Tyto parametry nám pro

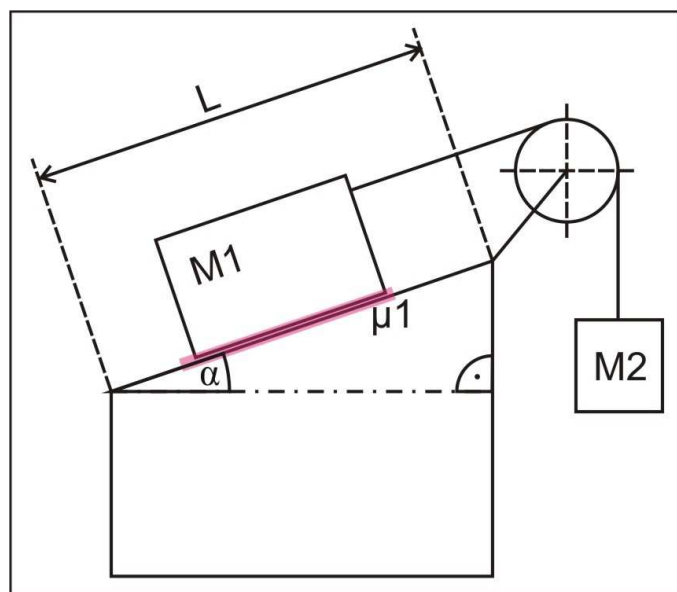
základ postačí, takže je zanedbána jakákoliv pružnost lan, tření v kladce, popřípadě její setrvačnost, vliv okolního prostředí na pohyb beden, jako například vítr apod. To však neznamená, že do budoucna by tyto parametry nešli přidat. Prakticky by to znamenalo vytvořit o jednu či více lišt navíc (pro nastavení požadované hodnoty) a úpravou vzorce řídicí chování beden. Pro samotnou ovladatelnost byl přidán ještě jeden nastavitelný parametr a to „Pozice bedny“, neboli pozice bedny vůči dráze. Je nutné dodat, že délka lana propojující bedny je konstantní.

Samozřejmě původně jsem měl v plánu více parametrů, od nichž jsem v průběhu tvorby upustil z důvodu nepřehlednosti, což mělo také za následek změnu finálního vzhledu.

#### 4.1.2 Návrh vzhledu

Je dobré před započítím tvorby popřemýšlet nad funkcí našeho programu a udělat si jistou představu o tom jak by měl fungovat v souvislosti na prostředí v kterém jej tvoříme. Proto je dobré se nejprve podívat na již vytvořené applety a zjistit co vlastně Easy Java dokáže.

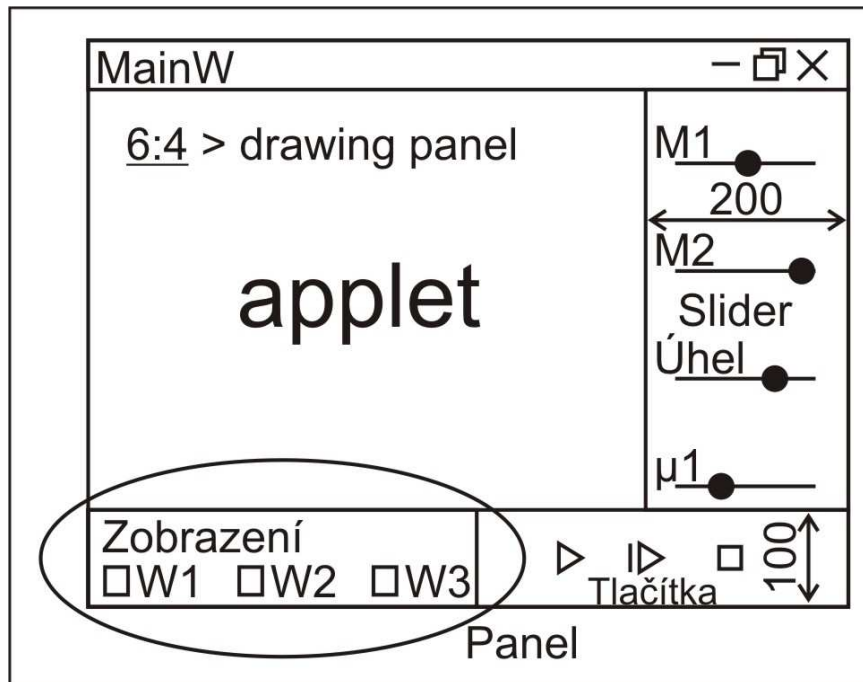
Po té, co tak učiníme už víme zhruba co vše můžeme vytvořit a kam až můžeme zajít. Naším dalším krokem je představit si přibližný vzhled naší aplikace, co se předvádění mechaniky modelu tyče. Určitě není od věci načrtnout si jej a v průběhu tvorby se k němu co nejvíce přiblížit. Popřípadě vycházet od náčrtku (Obr. 5), jako od základu a inovovat jej nápady, které Vás ještě napadnou.



Obr. 5. Náčrtek appletu.

### 4.1.3 Interface a okna

Nesmíme zapomenout ani na interface, jenž hraje také velkou roli. Nejen, že obsahuje ovládání appletu, jako třeba tlačítka řídící stav programu, ale také zprostředkovává zadávání proměnných a realizuje zviditelnění dalších oken. Stejně jako u vzhledu appletu i zde není od věci nakreslit si náčrtek (Obr. 6).



Obr. 6. Náčrtek interface.

Dá se říci, že této předlohy jsem se držel celou dobu, jen s drobnými změnami související s novými nápady a vylepšeními.

Základem ovládání celé aplikace jsou tlačítka, jejichž manipulací je applet řízen. U nich jsem se nechal inspirovat z jiných appletů a zvolil jsem „klasické“ ovládání třemi tlačítky:

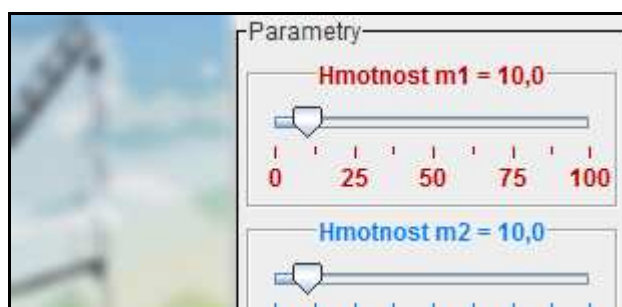
- Play(Spustit)/Pause – jedná se o dvojité tlačítko umožňující spuštění a pozastavení chodu programu.
- Krok – jde o „krokování“ programu, jenž na okamžik program spustí a hned jej opět za krátký časový interval zastaví. Jde o posun o 5 setin sekundy.
- Restart – provede reinicializaci většiny proměnných a vrátí program do původního stavu po spuštění. To se týká také vymazání grafů a tabulky.

Tyto tlačítka jsou popsány také v manuálu, který je součástí appletu. Ten se otevře vždy po jeho spuštění. Jedná se o klasické „buttons“, neboli tlačítka umístěné v panelu „Ovládací tlačítka“ (Obr. 7).



Obr. 7. Buttons / tlačítka.

Pro zprostředkovávání zadávání proměnných jsem vybral prvek nazvaný jako „slider“, neboli šoupátko (Obr. 8). Ty jsem vybral z důvodu jednoduchosti nastavení parametrů, ale také protože lze u nich jednoduše nastavit meze proměnných a v neposlední řadě kvůli vzhledu. Jedinou jejich nevýhodou je někdy nepřesné nastavení žádané hodnoty. Výjimkou je prvek pro nastavení gravitační konstanty, u kterého jsem zvolil prvek „comboBox“, umožňující zadat hodnotu jak ručně, tak vybrat již z přednastavených hodnot. Důvodem, proč jsem pro tento parametr nevybral „slider“ jako tomu je u ostatních, je již zmíněná možnost výběru již z přednastavených hodnot. Za to však neumožňuje přidat popisek k prvku, tak i k seznamu hodnot v něm obsažený. Tyto prvky jsou umístěny na pravé straně aplikace, v části označené jako „Parametry“.



Obr. 8. Slider / šoupátko – posuvná lišta.

Od začátku jsem měl v plánu do nesouvisejících oken zvlášť umístit grafy a tabulku, jenž by hráli roli výstupu z našeho programu. I pro ně jsem si vytvořil náčrtky, ale ty neobsahují natolik důležité informace, abych je zde uváděl. Zmiňované okna zobrazují závislost jedné veličiny na druhé. Tyto okna lze zobrazit v panelu „Okna“ a zobrazují:

- závislost velikosti působících sil  $F$  na čase  $t$  / „Graf [ $F = f_1(t)$ ]“,

- závislost velikosti rychlosti  $v$  na čase  $t$  / „Graf [ $v = f_2(t)$ ]“,
- závislost velikosti zrychlení  $a$  na čase  $t$  / „Graf [ $a = f_3(t)$ ]“,
- závislost okamžité polohy  $x$  na čase  $t$  / „Graf [ $x = f_1(t)$ ]“,
- tabulka vypočítaných hodnot / „Tabulka“.

Pro jejich zviditelnění jsem zvolil tzv. „checkbox“, což je buňka umožňující zatrhnutí (zarhávací rámeček). Tím se změní hodnoty typu boolean, jenž může být true nebo false. Tato proměnná následně zobrazuje, nebo zneviditelní příslušné okno. Zmíněné položky jsou umístěny v panelu „Okna“ (Obr. 9). Zde jsem umístil také možnost „Vektory“, která zviditelní zobrazení vektorů.

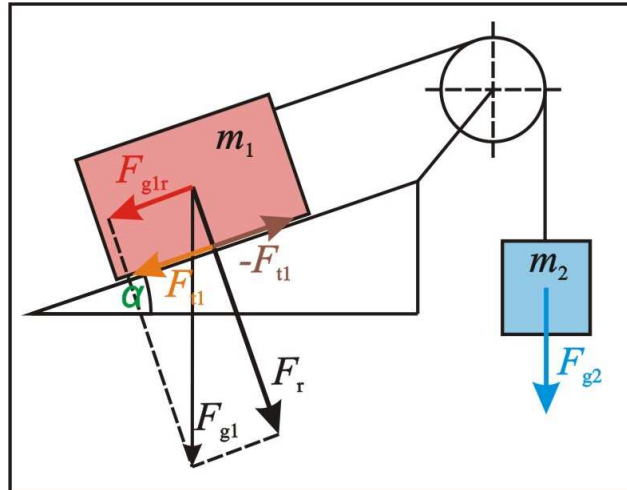


Obr. 9. Checkbox / zatrhávací rámeček.

Vzhledem k tomu, že výsledný pohyb beden v appletu může být nezřetelný, je umístění grafů nutností.

#### 4.1.4 Fyzika modelu

Tato část, je dá se říci stěžejní pro funkčnost našeho modelu. Přitom ji ale paradoxně můžeme nechat až na konec a první vytvořit to co bude onen vzorec pohánět. Vzorec vyplývá z platnosti Newtonové pohybové rovnice, která obsahuje síly působící na objekty v našem modelu. To znamená, že jejich vektorový součet udávající výslednou sílu  $F$ .



Obr. 10. Síly působící na jednotlivé objekty na nakloněné rovině.

Pokud uvažujeme, že tělesa jsou spojené nedeformovatelným lanem, velikost zrychlení, kterými se oboje tělesa pohybují bude stejné, uvažujeme jejich pohyb po přímce a zvážíme jejich směry působení, pro výslednou sílu platí:

$$F = F_{g2} - F_{glr} \pm F_{t1} , \quad (1)$$

kde  $F_{g2}$  je síla tíhy visícího tělesa na laně,  $F_{glr}$  – síla tělesa pohybující se po rovině a  $F_{t1}$  síla tření. Znaménko plus-mínus je zde z toho důvodu, protože tření vždy brzdí pohyb těles jakýmkoliv směrem, takže pokud se tělesa pohybují jedním směrem, tak do vzorce přísluší jedno znaménko, a když druhým, tak druhé znaménko.

Pokud použijeme následující označení:  $m_1$  - hmotnost bedny na nakloněné rovině,  $m_2$  - hmotnost bedny zavěšené na laně,  $\mu$  - smykové tření,  $g$  - zrychlení (v případě pohybu na Zemi – tíhové zrychlení),  $\alpha$  – úhel sklonu nakloněné roviny, jednotlivé působení sil je možné vyjádřit vztahy:

$$F_{glr} = m_1 g \sin \alpha \quad (2)$$

$$F_{t1} = m_1 g (\cos \alpha) \mu \quad (3)$$

$$F_{g2} = m_2 g \quad (4)$$

Po dosazení dostaneme:

$$F = (m_2 g) - (m_1 g \sin \alpha) \pm (m_1 g \mu \cos \alpha) \quad (5)$$

Vzhledem k tomu, že Easy Java umožňuje počítat derivace různými metodami je efektivnějším způsobem zadat tento vzorec ve formě derivací. Prakticky v našem modelu rozdíl okamžité polohy  $x$  a počáteční polohy  $x_0$  určuje dráhu bedny dráha  $s = x - x_0$ . Jestliže  $x_0 = 0$ , okamžitá poloha  $x$  určuje dráhu bedny  $s$ . Pro jednorozměrný případ možno dráhu určit ze vztahu pro okamžitou rychlost, určenou vztahem (6):

$$\frac{ds}{dt} = v \quad (6)$$

S tím, že rychlost  $v$  můžeme vypočítat na základě definici zrychlení, určené vztahem (7):

$$\frac{dv}{dt} = a \quad (7)$$

Newtonův zákon síly:

$$F = m * a \quad (8)$$

říká, že síla  $F$  působící na těleso hmotnosti  $m$  mu uděluje zrychlení  $a$ , které můžeme vyjádřit:

$$a = \frac{F}{m} \quad (9)$$

Po využití výše uvedených vztahů pro jednorozměrný pohyb je možné napsat:

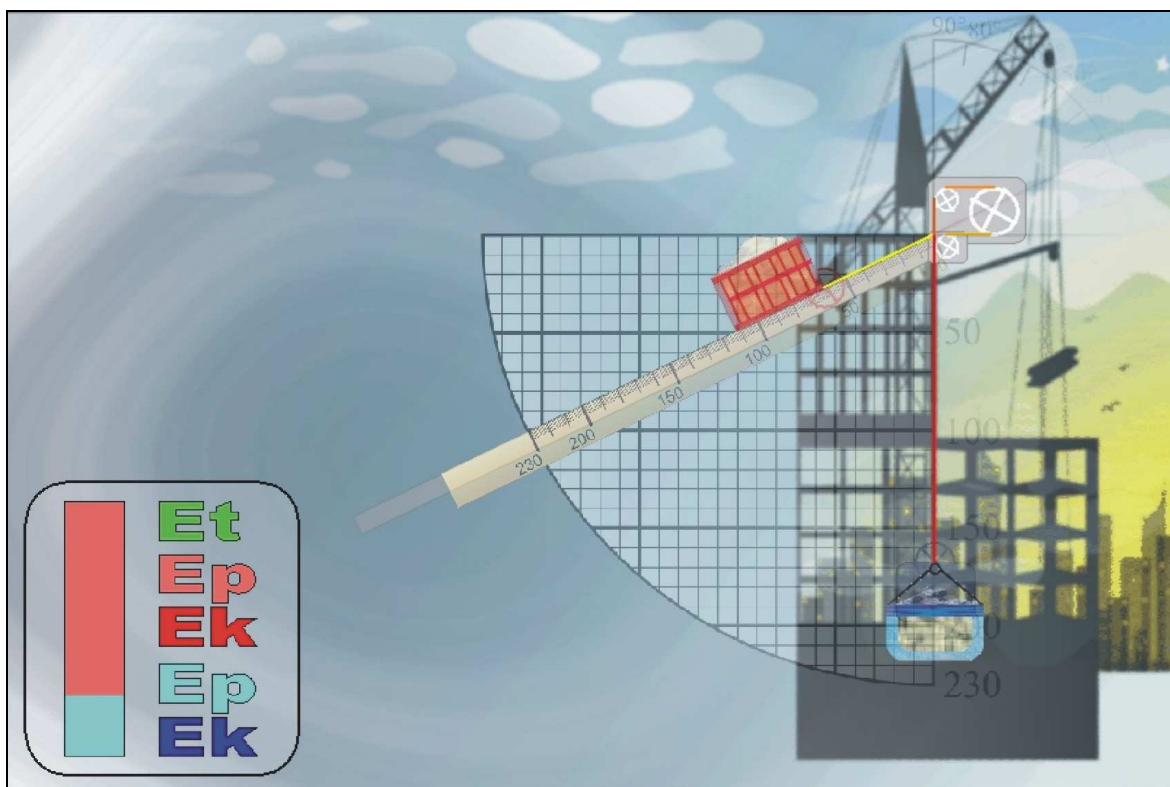
$$\frac{dv}{dt} = \frac{F}{m} = \frac{F_{g2} - F_{g1r} \pm F_{t1}}{m} = \frac{m_2 * g - m_1 * g * \sin \alpha \pm m_1 * g * \cos \alpha * \mu}{m_1 + m_2}, \quad (10)$$

kde  $g$  je tíhové zrychlení,  $m$  hmotnost beden (t.j. součet jejich hmotností),  $\alpha$  úhlem naklonění roviny (odklon od vodorovné roviny) a  $\mu$  smykové tření.

Pokud zvažujeme pohyb samotné bedny na nakloněné rovině, bez tření, tak programování není náročné. Co se fyziky týče, tak ji můžeme naprogramovat po krocích, takže od zmiňované bedny na nakloněné rovině – rovnice (2). S každým dalším krokem obtížnost exponenciálně roste, ale o tom až dále. Nyní máme potřebné vztahy realizující chování aplikace po fyzikální stránce.

#### 4.1.5 Zasazení do reálného prostředí

Nesmíme zapomenout, že tvoříme simulaci určenou hlavně do výuky a na samostudium hravou formou. Jedním z bodů charakterizující tyto simulace je zasazení appletu do reálného prostředí. Protože applet simuluje pohyb dvou beden, jenž se vzájemně ovlivňují, je těžké najít zrovna vhodné zasazení do skutečného světa. Pokud se podíváte třeba jen na náčrtek, tak je vám hned jasné, že ve skutečnosti by jsme například pro výtah použili jinou konstrukci. Proto jsem zvolil neutrální vzhled, a to staveniště. A protože reálné prostředí někdy může splynout s objekty samotnými, tak byl přidán vzhled navíc, jenž v sobě má několik prvků pro vyhodnocování výsledků. Jedná se o mřížku, pravítko a úhloměr. Změnu vzhledu lze provést v liště „Okna“ zaškrtnutím položky „Vzhled“. Následující obrázek znázorňuje přechod z jednoho vzhledu do druhého (Obr. 11). Původní obdélníky střídají kontejnery, takže obrázky s nastavenou průhledností a o zbytek atmosféry se postará pozadí staveniště.



Obr. 11. Finální vzhled appletu.

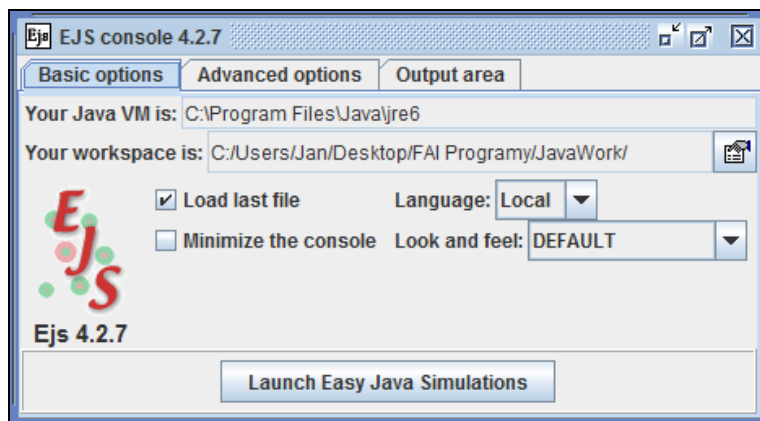
## 4.2 Popis tvorby

Pokud s Easy Javou začínáme, tak z mé osobní zkušenosti je nejlepší vycházet z jednoduchých aplikací šířených jako Open Source. Vesměs se nejedná o nic, co by se každý se špetkou dedukce a zvědavosti nemohl sám naučit, i když začátky jsou vždy těžké.

### 4.2.1 Seznámení se s prostředím

Základem je ujasnit si jak se v prostředí Easy Java pracuje. Při prvním spuštění se Vás program zeptá, kam chcete umístit tzv. „workspace“, do nějž bude Easy Java ukládat Vaši tvorbu. Z toho místa bude prostředí vycházet a je vhodné jej umístit někam, kam má uživatel rychlý přístup pro manipulaci se soubory. Do „workspace“ se musí také umíšťovat všechny projekty, jenž chcete v Easy Javě otevřít.

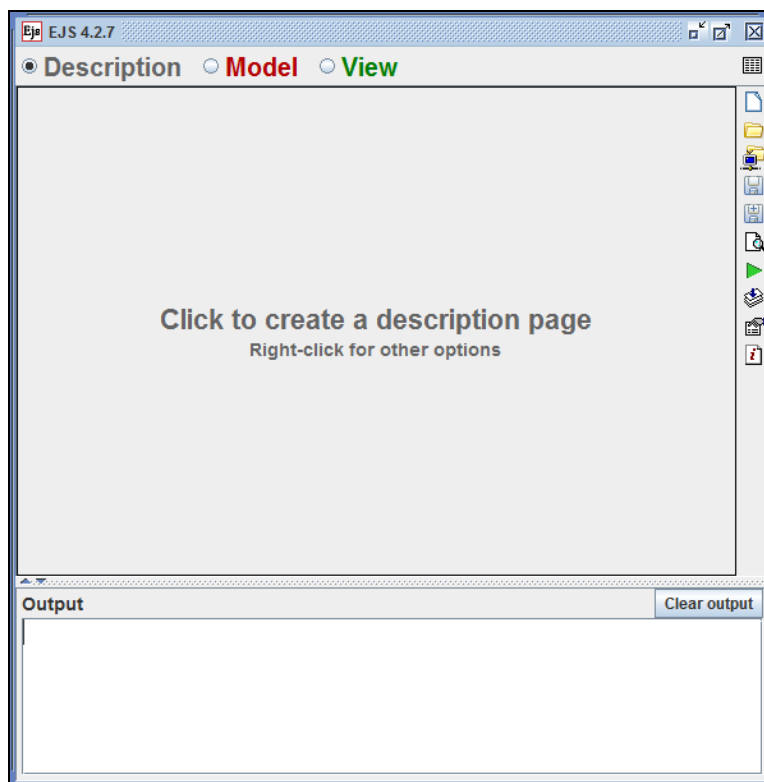
Nyní, když už máme vytvořený prostor na disku, v kterém můžeme tvořit, je třeba seznámit se s vývojovým prostředím jako takovým. Po spuštění se nám zobrazí dvě okna onačené jako konzole (Obr. 12), s kterou nebudeme vůbec pracovat a hlavní okno programu („EJS 4.2.7“). EJS je zkratka Easy Javy a 4.2.7 je verze programu (Obr. 13). Nutno podotknout, že výsledná práce byla tvořena ve verzi 4.3.2. Novější verze samozřejmě přináší několik inovací. Ty však neovlivňují cokoliv zde uvedené.



Obr. 12. Konzole Easy Javy.

Hlavní okno je rozděleno na tři základní části a to na:

- Description (popis)
- Model
- View (pohled)



Obr. 13. Hlavní okno Easy Javy.

Každé z těchto oken hraje významnou roli při tvorbě appletu s výjimkou „Description“, které se může teoreticky zanedbat.

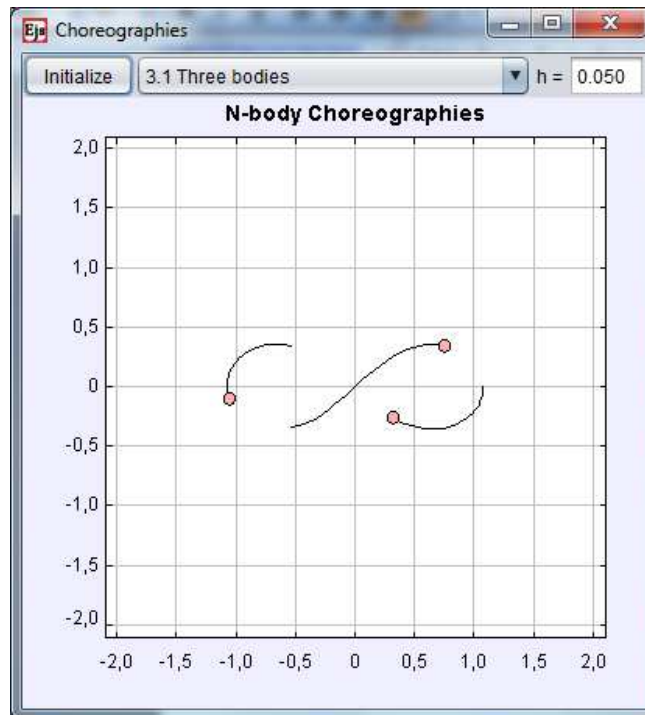
Description – panel sloužící pro popis aplikace. Hraje nejmenší roli při vytváření nové aplikace. Slouží především jako uvítací stránka, jenž se zobrazí při spuštění výsledného programu. Hlavním účelem by mělo být, informovat uživatele o funkci dané aplikace, jak ji správně ovládat a jiné informace, jenž tvůrce uzná za vhodné, aby při práci s tímto produktem uživatel věděl.

Model – jedná se o nejdůležitější část prostředí. V tomto místě se doslova programuje chování appletu a obsahuje vše co k programování patří, rozčleněné do jednotlivých skupin. Easy Java umožňuje programátorovi vyřešit problematiku dané části kódu více způsoby. Panel „Model“ obsahuje tyto pod-panely:

- *Variables – proměnné.* Slouží k nadefinování a deklaraci hlavních veličin, které bude náš program používat. Programátor si sice může uprostřed kódu nadefinovat nové proměnné („int x;“). Ty však nejdou naplno využít pro ovládání appletu a to proto, protože budou vytvořeny až v průběhu / za běhu programu, takže takováto

proměnná nelze využít třeba k ovládní objektu (program s proměnou uvedenou v jakémkoliv objektu počítá už od začátku).

- *Inicializacion – inicializace.* Zde se inicializují vytvořené proměnné na počáteční hodnoty. To je sice možné již při její tvorbě, ale v některých případech to nestačí. Nejlepším případem je asi pole „streamů“, jenž je použité v menu jako text, který vyplňuje jednotlivé možnosti. Nesmíme zapomenout na to, že při následné manipulaci s těmito proměnnými se po restartu programu provede opět re-inicializace.
- *Evolution – evoluce / rozvoj.* Pro zde umístěný kód se předpokládá, že se bude rozvíjet, nebo měnit v čase. Jde tedy o sekci stvořenou pro různé vzorce obsahující čas jako proměnou. Tento kód se bude počítat jen v případě, že bude program spuštěn ve smyslu provádění úkonů schopný. Většina appletů obsahuje ovládací prvek jako je například tlačítko označené jako „Play“, které uvede applet do provozu. Teprve po jeho stlačení se začne kód zde umístěný počítat. Je ale také možné nastavit automatické spuštění chodu po startu programu. Ve skutečnosti jde o spuštění „funkce“ `_play()`. V této části vývojového prostředí je také možné zadat vzorce ve formě derivací.
- *Fixed relations – pevné vztahy.* Veškerý zde napsaný kód se počítá neustále od spuštění programu, i když nebyla spuštěna funkce `_play()`. Je vhodné zde umístit například různé přepočty proměnných, obsluhující model po celý čas trvání. V našem případě tak počítáme úhel naklonění roviny, nebo pozici bedny. Teoreticky by v této sekci měli probíhat pouze výpočty, které jsou důležité pro interface a vzhled. Různé přepočty potřebné až při chodu by měli být umístěny v sekci „Evolution“, z toho důvodu, že v opačném případě jsou propočítávány neustále, i když jich je za potřebí jen při chodu programu. Ve skutečnosti se mohou objevit komplikace s tímto spojené jen výjimečně.
- *Custom.* Jedná se o poslední záložku v okně Model. Ta umožňuje vytvoření nové třídy, nebo funkce chcete-li. Tento panel je důležitý z hlediska kódu, který může být v jednom okamžiku spuštěn vícekrát, jen s jinými vstupními parametry. Pěkným případem toho je applet „Choreographies“ (Obr. 14), kde pro každý zobrazený pohybující se bod je napsán totožný kód.



Obr. 14. *Choreographies* – applet.

View – jde o poslední záložku v prostředí Easy Java a obsahuje jednotlivé objekty, které mohou být jak pasivní, tak i interaktivní. Tyto objekty tvoří stromovou strukturu a zprostředkovávají veškeré dění v appletu, čímž vzniká výsledný vzhled appletu. Jak už bylo zmíněno, tak celá tvorba vzhledu je založena na stromové struktuře, ve které hlavní kořen tvoří jednotlivá okna. Do těchto hlavních kořenů můžeme vkládat jednotlivé objekty ze tří základních oken umístěné v levé části okna označené jako „Elements for the view“ a to tyto:

- *Interface* – základní prvky sloužící k manipulaci s modelem, nebo k rozvržení umístění dalších objektů.
- *2D Drawables* – jedná se zejména o 2D objekty, které zprostředkovávají akci appletu, ale také umožňují vytvoření vzhledu appletu vkládáním jednotlivých obrázků na sebe apod.
- *3D Drawables* – objekty umístěné v tomto okně mají stejný význam jako u „2D Drawables“, jen se jedná o 3D objekty.

Objekty, které můžeme do našeho projektu vložit je opravdu velké množství. Jedná se o úsečky, pružiny, geometrické objekty, grafy atd. Také vzhledem k relativně velkému využití Easy Javy je to logické.

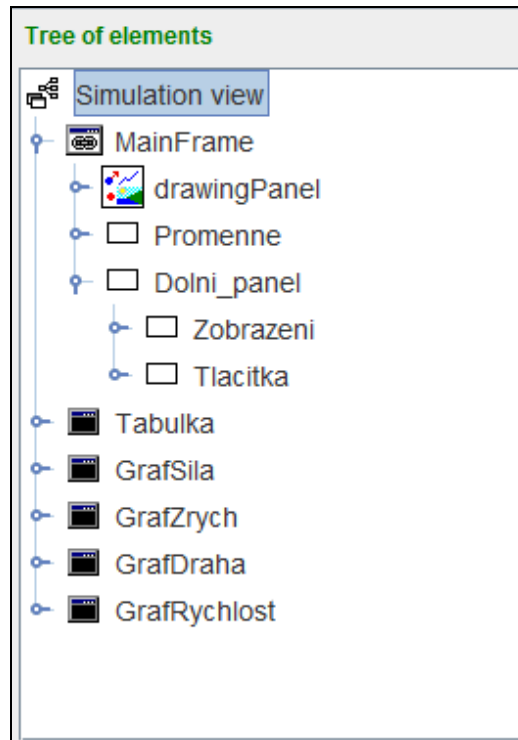
#### 4.2.2 Tvorba interface a oken

Nyní, když jsme obeznámeni s funkcemi jednotlivých částí Easy Javy, tak víme v jaké oblasti se co tvoří a můžeme tedy začít s tvorbou. Jak bylo ale již zmíněno, tak pokud jsme úplní začátečníci, není od věci vycházet z nejjednodušších aplikací stažitelné z hlavních stránek Easy Javy [www.um.es/fem/EjsWiki](http://www.um.es/fem/EjsWiki).

V tomto bodě se budeme pohybovat pouze v okně „View“. Celým základem je pouze jednoduché vkládání objektů do stromové struktury. Ještě před tím než začneme, je nutné zmínit to, že pokud kliknete pravím tlačítkem myši na jakýkoliv objekt zde, tak se objeví nabídka, která Vám nabídne online nápovědu k danému prvku.

Jako první musíme vytvořit okna, s kterými bude uživatel pracovat, takže vytvoříme nejen hlavní okno, ale další okna pro grafy a tabulku dle návrhu. Umístění jednotlivých grafů do samostatných oken je vhodné, pokud nechceme aby hlavní okno bylo zahlceno různými informacemi a tak zlepšit přehlednost tím, že uživatel si bude moc zvolit, který graf bude potřebovat podle vlastní potřeby. Abychom tak mohli udělat, tak v nabídce Interface/Windows vybereme objekt „Frame“, který vložíme 1× a objekt „Dialog“, který vložíme 5×. Vše pod sebe. Teoreticky bychom mohl pod sebe vložit 6× „Frame“, ale museli bychom jedno z oken nastavit jako hlavní a navíc pro zbylých 5 oken postačí jako objekt „Dialog“.

Do našeho hlavního okna dále umístíme „DrawingPanel“ a 2× „Panel“, s tím že při každém vložení objektu se Vás program zeptá, kam chcete daný objekt umístit. DrawingPanel zarovnáme na střed. Tím získáme prostor v kterém se bude odehrávat veškeré dění v našem appletu, co se mechaniky týče. A „Panel“ umístíme dolů a doprava. Tím získáme prostor pro ovládací prvky, jako jsou tlačítka, šoupátka a zaškrťovací pole (CheckBox). Dále do panelu „Dolní\_panel“ umístěného do spodní části okna opět 2× „Panel“. Tentokrát je však rozvrhneme doprava a doleva. Tím vytvoříme oddělený prostor pro tlačítka a zaškrťovací pole. Takto rozvržené objekty by měli vytvořit následující stromovou strukturu (Obr. 15).



Obr. 15. Okna a rozvržení interface.

Nesmíme zapomenout, že všechny objekty včetně oken mají řadu parametrů, jejichž nastavení není zanedbatelné, i když pokud žádné nezadáte bude použito základní nastavení. Pro tyto objekty jsme nastavili následující parametry:

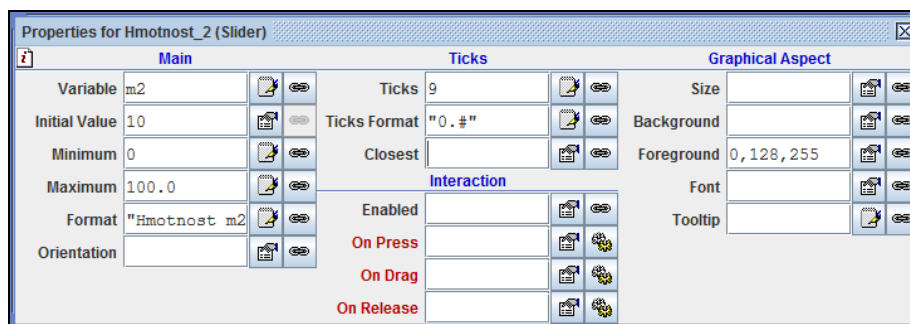
- MainFrame
  - Title: „Applet“ – název okna
  - Image – ikona okna programu
  - Visible: True – viditelnost
  - Size: 800×500 – velikost okna v pixelech
- Tabulka, Graf\* - u těchto oken stačí pouze nastavení velikosti, názvu a hlavně do kolonky „Visible“ zadat pro každé okno samostatnou proměnou typu „boolean“, podle které se bude řídit to, zda dané okno má být viditelné, nebo ne.
- Panely – u těchto objektů je zbytečné nastavovat fixní velikost z toho důvodu, že uživatel si bude sám moci okno zvětšit či jinak přizpůsobit. Hlavní je nastavit pod kolonkou „Layout“ jakým způsobem se mají objekty uvnitř řadit.

- `drawingPanel` – zde je nejdůležitější nastavit „Autoscale“ pro osu  $x$  a  $y$  na „false“ a to z toho důvodu, protože tento parametr ovlivňuje, zda se mají nastavené maximální rozměry této plochy automaticky měnit, tak aby šli vidět i objekty přesahující tuto hranici. Po té je třeba nastavit minimální a maximální hodnotu plochy, po které se budou pohybovat zde umístěné objekty. Takže při nastavení velikosti  $600 \times 400$  si můžete představit tuto plochu jako matici bodů, kde každý další objekt se musí pohybovat v rámci tohoto omezení.

Nyní máme základy, na kterých můžeme stavět. Některé následující kroky byli vytvořeny až v pozdějším stádiu vývoje appletu s ohledem na jeho funkčnost a ne najednou. To je umožněno díky tomu, že Easy Java není striktní programovací prostředí, ale je flexibilní.

K interface patří samozřejmě ovládací prvky, takže našim dalším úkolem je vložit a nastavit je. Začneme u panelu „Promenne“. Do něj vložíme tolik „slider-ů“ (šoupátek), kolik máme nastavitelných proměnných. V našem případě  $5 \times$ . Jak už bylo řečeno výše, tak jsme proměnné „Gravitační zrychlení“ přiřadili prvek „ComboBox“ u nějž nám stačilo vyplnit hlavně kolonku pro proměnnou a seznam hodnot v něm obsažené. Ve vlastnostech pro „slider“ nastavíme tyto buňky:

- Variable – název proměnné, která se bude posunutím šoupátka měnit. Nejlépe typu „double“.
- Maximum / Minimum – maximální a minimální nastavitelnou hodnotu.
- Format – text ve tvaru „Něco = 0.0“, který se bude zobrazovat nad šoupátkem, jen na místo „0.0“ bude aktuální hodnota proměnné s tím, že počet nul za desetinou čárkou bude udávat přesnost s jakou se bude proměnná zobrazovat.
- Ticks – počet čárek na šoupátku upřesňující zadání hodnoty.
- Ticks format – Zde je nejlépe zadat „0.#“. Přebírá text z buňky „Format“ a nahrazuje „0.0“ za proměnnou.



Obr. 16. Nastavení vlastností u šoupátka.

Tímto jsme vyřešili 1/3 ovládání appletu. Následují tlačítka. Ty vytvoříme vložením dvou tlačítek „Button“ a jednoho dvouhodnotového tlačítka „twoStateButton“ z okna Interface/Buttons do panelu „Tlacitka“. Nastavení těchto tlačítek není vůbec obtížné. Stačí zadat text, který mají obsahovat. Po té obrázek znázorňující jeho funkci a hlavně do kolonky „Action“ napsat akci, které mají při stisknutí provádět. Pro tlačítko „resetButton“ to je „\_reset()“ a pro „stepButton“ akci „\_step()“. Výjimkou je dvouhodnotové tlačítko, jenž obsahuje 2× ty samé parametry pro stav zapnuto a vypnuto. Do zapnuto dáme akci „\_play()“ a do vypnuto funkci „\_pause()“.

Nyní nám už jenom zbývá vložit 7× „CheckBox“ do panelu „Zobrazeni“. Tyto prvky budou sloužit jako zobrazovací prvky, takže musí obsahovat stejné proměnné v buňce „Variable“ jako jsou proměnné u jednotlivých oken grafů a tabulky. Ve výsledku to znamená to, že bude-li pole zaškrtnuto, tak se dané okno zviditelní. Samozřejmě s výjimkou dvou „CheckBoxů“, které budou zapínat viditelnost vektorů v appletu a změnu vzhledu.

Pro naše účely je také dobré vyplňovat atribut „Tooltip“ nebo-li nápověda, která se zobrazí při najetí myši na konkrétní prvek.

### 4.2.3 Tvorba grafů a tabulky

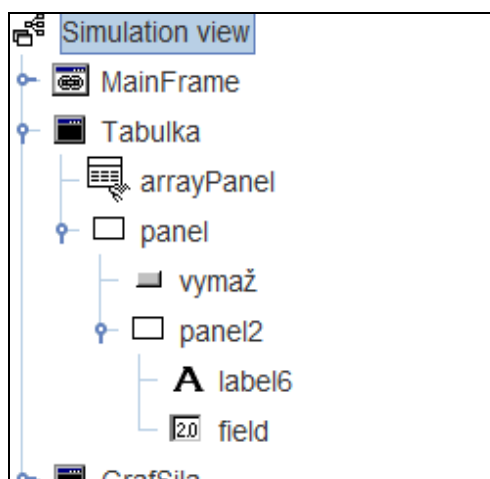
Grafy jsou sami o sobě vcelku jednoduchou záležitostí, nebo alespoň pokud se jedná o 2D grafy. Základem je vložení „PlottingPanel-u“ do oken vytvořených pro grafy a následně vložit do plottingpanelu objekt nazvaný jako „Trace“ tolikrát, kolik chceme aby graf zobrazoval trajektorii. Nastavení grafu je podobné jako u „drawingPanel-u“ výše, jen „Autoscale“ nastavíme na „true“. Tím povolíme automatické zvětšování rozmezí pro

případ, kdyby trajektorie přesáhla nastavené rozmezí. Je vhodné nastavit ještě popisky os a jiné vlastnosti.

Proto, aby graf něco zobrazoval je ještě nutné nastavit trasy. Takže musíme do buněk „Input X“ a „Input Y“ vložit ty správné proměnné, tak aby graf zobrazoval to co má. Například pro graf zobrazující rychlost na čase nastavíme do „Input X“ proměnnou  $t$  (čas) a do „Input Y“ proměnnou  $v$  (rychlost). Pro lepší přehlednost je vhodné nastavit také barvu trajektorie a tloušťku.

Vytvoření tabulky je o něco pracnější, z toho důvodu, že musíme naprogramovat zaplňování tabulky. Nejprve však ale musíme sestavit stromovou strukturu (Obr. 17). Do ní vložíme:

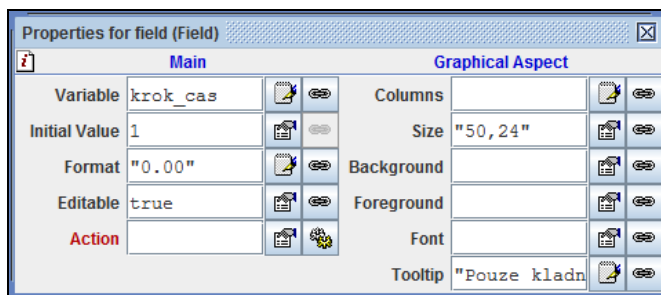
- 1× „ArrayPanel“ (tabulka) - arrayPanel
- 2× „Panel“ – panel, panel2
- 1× „Button“ - vymaž
- 1× „Field“ (pole) – field – do této buňky lze zadávat vlastní hodnoty podobně jako přes šoupátko, ale ručně.
- 1× „Label“ (štítek) – label6 – obsahuje čistý text, bez jakékoliv interakce.



Obr. 17. Stromová struktura tabulky.

Do vlastností tabulky (arrayPanel) je nutné nastavit do kolonky „Data“, ve „Variables“ nadefinované dvourozměrné pole o tolikati sloupcích a řádcích, kolik chceme aby naše tabulka zobrazovala. V našem případě by velikosti 100×4 měla stačit. Formát musíme nastavit tak, aby tabulka zobrazovala požadovaný počet desetinných míst. Zvolili jsme tedy

"0.##" (dvě desetinná místa). Dále kolonka „Column names“, nebo-li název sloupců musí obsahovat proměnou typu „string“ (pole) obsahující tolik textových řetězců, kolik máme sloupců. Nám postačí přímo zapsat: „new String[]{\"#\",\"čas t [s]\",\"rychlost v [m/s]\",\"zrychlení a [m^2/s]\",\"dráha x [m]}“. I když máme pole jen o 4. sloupcích, tak string obsahuje 5 textových řetězců, protože jeden je přidán automaticky jako počítadlo řádků / prvků. Prvek „field“ jsme nastavili podle obrázku (Obr. 18).



Obr. 18. Nastavení buňky „field“.

Uvedenou proměnnou (krok\_cas) bude potřebovat v pozdějším kroku. Do tlačítka „vymaž“ nastavíme pouze text a do kolonky „Action“ napíšeme „vymaz()“.

Zdrojový kód funkce vymaz():

```

1 public void vymaz () {
2     for(int x=0; x<100; x++)
3     {
4         for(int y=0; y<4; y++)
5         {
6             arrayTab[x][y]=0;
7         }
8     }
9     i = 0;
10 }

```

V praxi tento kód způsobí to, že pokud uživatel zmáčkne tlačítko vymazat, tak se zavolá funkce, která nastaví všechny hodnoty pole „arrayTab“ na nulu a po té co to provede nastaví proměnnou *i* rovněž na nulu. Tato proměnná slouží v jiné části programu a určuje,

na jaký řádek v tabulce se mají data zapisovat. Tento kód je umístěn v „Model/Custom/Vymaz“.

#### 4.2.4 Tvorba jádra appletu

Před tím, než začneme programovat samotnou fyziku modelu, tak je nutné do appletu umístit několik objektů představující bedny, nakloněnou rovinu a zakomponovat vztahy mezi nimi, tak aby byli reakce schopné. Proto v do hlavního okna (MainFrame) do „drawingPanel“ umístíme:

- Image (obrázek) – tyto obrázky představují pozadí a 2. bedny.
- Shape (tvar) – jedná se o nastavitelný objekt nastavitelný na jednotlivé geometrické tvary. V našem případě se jedná o obdélníky.
- Particle (částice) – jde o podobný objekt jako je „Shape“, jen s jiným přístupem k parametrům. Představuje kladku.
- Arrow (šipka) – použita jako spojnice mezi kladkou a bednami, a také jako znázornění vektorů.

Každý z těchto prvků je použit vícekrát a v kombinaci a pestrým nastavením znázorňují veškeré dění.

Jako první potřebujeme nastavit chování roviny, tak aby šla naklonit. U roviny, nebo-li „Rovina“, jako objekt ve „View“ musíme samozřejmě nastavit výšku a šířku, plus její umístění v okně. V tomto bodě nás nejvíce zajímá parametr „Transform“, který udává úhel naklonění tohoto objektu. Zde narážíme na první intuitivní problém Easy Javy.

Easy Java přistupuje k úhlu ne ve stupních, ale v obloukové míře, tj. jako k  $\pi$ . Tento přístup je nutný zohlednit u všech výpočtů s úhlem. Takže proměnou, která přímá hodnoty z šoupátka „Uhel“ ve stupních nemůžeme použít přímo do kolonky „Transform“, ale je nutné ji nejprve přepočítat následujícím kódem:

```
6 mUhel = (Math.PI / 180) * uhel;
```

Kód z „Model/Fixed relations/Nonstop Model“ přepočítává úhel ve stupních z proměnné „uhel“ na „mUhel“, kde přes Math.PI se v Javě přistupuje k  $\pi$ , takže výsledek bude přesnější. Takto převedený úhel můžeme použít jak k naklonění roviny, tak i

u geometrických funkcích. V tomto místě je umístěna většina kódu ovládající model jako takový.

Nyní, když máme rovinu, tak musíme naprogramovat chování beden, tak aby se vzájemně posouvali, pokud se bude měnit „delka“, jakožto okamžitá souřadnice. Pro druhou bednu označenou jako „bedna2“ je toto nastavení nejjednodušší. Parametr souřadnice X zůstává konstantní, pouze Y souřadnice se počítá jako rozdíl délky a nastavené souřadnice. Tento výpočet je zadaný přímo ve vlastnostech „bedna2“. Proměnná „delka“ je jednou z nejdůležitějších proměnných a celý applet stojí na výpočtu této proměnné, která udává polohu beden. Pro první bednu, jakožto bednu na naklánějící se rovině, se při pohybu po rovině mění jak X, tak i Y souřadnice. Tento problém řešící následující kód:

```
8  bednaX = delka * Math.cos(mUhel);
```

```
9  bednaY = delka * Math.sin(mUhel);
```

Proměnné „bednaX“ a „bednaY“ jsou použity ve vlastnostech objektu „bedna“.

Co se kladky týče, tak ji stačí umístit na správné místo, nastavit vzhled, velikost, barvu a hlavně do parametru „Rotate“ vložit „delka“ děleno její poloměr.

```
14 kladkaRotace = delka / 12;
```

Nyní nám už jen stačí vzájemně propojit bedny skrze kladku. Pro tento účel nám poslouží dvojice šipek, bez ukazatele směru. To provedeme nastavením zakončení ve vlastnostech. Pozice šipky se vztahuje vždy k jednomu bodu, který představuje počátek (<0,0>) pro nastavení konce. Propojení mezi druhou bednou a kladkou je jednoduché. Zadá se pozice na kladce jako počátek a pak už jen stačí pouze matematicky z délky vypočítat Y složku tak aby šipka budila dojem spojnice mezi nimi.

Problém by nastal, kdybychom chtěli propojit lanem bednu na nakloněné rovině spolu s kladkou, které by bylo výš než nejvyšší bod na rovině. Navíc by docházelo k odtržení lana od kladky, ale to by byla jen kosmetická chyba. Ve finální verzi jsem se rozhodl ono lano doslova položit na rovinu, tak jako by to byl nějaký řetěz, jímž může také být. Tak či onak to na výpočty nemá žádný zásadní vliv. Ona první varianta by akorát musela obsahovat o dva řádky kódu více. Tak tomu bylo v první verzi, ale to jsem musel změnit s ohledem na matematický vzhled, kdy je nutné aby lana „začínala“ na stejném místě. Díky tomu se můžou do vlastností šipky vepsat již dvě vypočítané hodnoty „-bednaX“ a „-

bendaY“ do atributu „Size X“ a „Size Y“ a lano je plně funkční. Samozřejmě spolu s jinými jako jsou tloušťka, počáteční poloha, orientace atd.

Vektory jsou počítány podobným způsobem, pouze mají ve vlastnostech parametr navíc pro viditelnost.

Aby však mohl náš applet ožít, je nutné zakomponovat vzorce uvedené výše do těla programu. Teoreticky pro jeho funkci by mohl stačit kód napsaný v „Model/Evolution/\*“, kam nám stačí jen správně zapsat vzorce. Prakticky tím však docílíme pouze nesprávného chování. Pokud si všimnete, tak u vzorce (10) je uvedeno plus / mínus a to proto, protože třecí síla je vždy opačného směru pohybu, než je směr pohybu a může být podle případu buď kladná nebo záporná. Pokud by tento applet neobsahoval třecí sílu, tak by jeho tvorba byla až banální záležitostí (po matematické stránce). Změnou parametrů v reálném čase může třecí síla způsobit nereálné chování appletu. K chování se vyjádřím ještě níže, u podmínek zamezující těmto situacím. Podívejme se tedy na jedny z nejdůležitějších řádků programu.

Kód jádra (Model/Evolution/Derivace):

$$1 \quad \frac{d \text{ delka}}{d t} = v$$

$$2 \quad \frac{d v}{d t} = -(F2 - F1 - (Ft - smer)) / (m1 + m2)$$

Tato stránka byla vytvořena jako „ODEs“ a přesně takto vypadá umístění kódu do „ODEs“ stránky. V praxi nám tento kód počítá derivace vedoucí k výpočtu požadované proměnné „delka“. Pro správnou funkci je nutné do kolonky zadat FPS = 20, SPD = 1, Indep.Var. = t, Increment = 0.05 a v kolonce Solver vybrat metodu „Euler-Richardson“, což metoda pro výpočet derivací, kterou jsem vybral z důvodu jejího použití v jiných appletech, které za použití této metody fungovali bezproblémově.

Kód jádra (Model/Evolution/Síly):

$$1 \quad F2=(m2*grav);$$

$$2 \quad F1=(m1*grav*Math.sin(mUhel));$$

$$3 \quad Ft=(m1*grav*Math.cos(mUhel)*mi1);$$

Tento kód mohl být zahrnut rovnou do derivací, ale pro přehlednost a možné úpravy kódu jsem je dal zvlášť.

Kód jádra (Model/Evolution/Omezení):

```
1  if (delka > 230)
2  { _pause();
3    v = 0;
4    delka = 230;
5  }
6  if (delka < 0)
7  { _pause();
8    v = 0;
9    delka = 0;
10 }
```

Prakticky jde o jedinou část programu, která program zastaví sama od sebe, bez zásahu uživatele. Nebo-li, pokud bude „delka“ větší jako 230, nebo menší jak 0, tak se program pozastaví, rychlost  $v$  nastaví na nulu a délka se nastaví na 0 nebo na 230. Přenastavení těchto hodnot je proto, protože pokud se bedny zastaví a applet bude znova spuštěn, je nutné aby se rozjeli s nulovou počáteční rychlostí. Co se délky týče, tak ta se musí přednastavit proto, protože pokud přesáhne mezi  $<0; 230>$  a applet by byl opětovně spuštěn, tak by se nerozjel i kdyby se bedny ve skutečnosti pohybovali správným směrem. Takže kdyby délka byla například 230,2 a došlo by k automatickému pozastavení. Jak bylo řečeno výše, tak doba mezi kroky činí 0.05 sekund, takže se pohyb nikdy nezastaví na přesné nule či 230.

Kód jádra (Model/Fixed relations/Chování):

```
2  if ((v > -0.2)&&(v < 0.2))
3  {
4    p_p = true;
5  }
```

```
6 else
7 {
8   p_p = false;
9 }
```

Nebo-li „p\_p“ je pravda, pouze pokud je  $v$  v rozmezí  $(-0.2; 0.2)$ . Kód níže obsahuje podmínky, které se mohou měnit pouze pokud je rychlost přibližně rovná nule. Proto je zde uvedeno toto rozmezí. Není možné napsat „ $v == 0$ “, protože při velkých zrychlení, kdy program propočítává jednotlivé kroky se může stát (a děje se tak), že program se na přesnou nulu nikdy nedostane a „přeskočí“ ji. To se dělo i při relativně malém rozmezí  $(-0.05; 0.05)$ .

```
11 pom_sila = -(F2-F1);
12 if (p_p == true)
13 {
14   if (pom_sila > 0) smer = -1.0;
15   if (pom_sila < 0) smer = 1.0;
16 }
```

Jádro problému je v tom, že třecí síla působící proti směru pohybu. A také může být při jistých nastavení větší, než síla působící na jednotlivé bedny. To by znamenalo, že by došlo ke zpomalení pohybu, až jeho zastavení. V praxi by však bedna bez jakýchkoliv podmínek například klesala, i když by měla ve skutečnosti stoupat. Proto abychom tomu mohli zabránit, tak se nejprve vypočítá skutečný směr pohybu bez tření, kde nás zajímá pouze zda výsledná hodnota je kladná nebo záporná. Další řádky nám říkají to, že pokud je rychlost přibližně rovna nule, tak „smer“ bude buď 1, nebo -1. Proměnná „smer“ nám udává směr třecí síly. Tyto řádky zamezují změně směru třecí síly uprostřed simulace (za běhu). Směr síly se tedy může měnit pouze tehdy, pokud je rychlost přibližně rovna nule.

```
18 if (p_p == true)
19 {
20   if(v > 0) {pom_p = 2;}
```

```
21 else {pom_p = -1;}
22 }
```

Těchto pět řádků řeší uplatnění dvojice nejdůležitějších podmínek v celé simulaci a říkají, kdy která podmínka má platit.

```
25 if (pom_p == -1) //!!!!
26 {
27   if (pom_sila > 0)
28   {
29     if (-(F2-F1+Ft) < 0)
30     {
31       if (v > -0.2) v = 0;
32     }
33   }
34 }
```

Tato podmínka platí pouze pokud je rychlost záporná. Bedny se pohybují dolů (zavěšená bedna), při tom skutečný směr je opačný a pokud je rychlost větší než -0.2, tak je rychlost rovna nule. Jinými slovy se rozjeté bedny při nárůstu tření začnou zpomalovat až nakonec dojde k jejich úplnému zastavení a nedovolí aby pohyb pokračoval špatným směrem. Při tom applet stále běží. Zní to jednoduše, ale uvědomění si těchto skutečností a správně je zapsat pro různé situace je velmi složité.

```
35 if (pom_p == 2)
36 {
37   if (pom_sila < 0)
38   {
39     if (-(F2-F1-Ft) > 0)
40     {
41       if (v < 0.2) v = 0;
```

```
42 }
```

```
43 }
```

```
44 }
```

Jedná se o tutéž podmínku, pouze pro kladou rychlost. I když je rychlost vždy kladná veličina, tak znaménko zde hraje roli „ukazatele“ směru pohybu (zda se zavěšená bedna pohybuje směrem vzhůru nebo dolů). Pokud se zavěšená bedna pohybuje směrem vzhůru, tak se jedná o kladnou rychlost.

Někdy se totiž stávalo, že zrychlení z ničeho nic nečekaně změnilo směr, nebo rychlost klesla na nulu a mnoho jiných problémů které jsme však vyřešili postupně.

Kód naplňující tabulku průběžnými daty (Model/Fixed realtions/Tab):

```
1  if (i == 100) i = 0;
2  if (t > pom_cas)
3  {
4    arrayTab[i][0] = pom_cas;
5    arrayTab[i][1] = v;
6    arrayTab[i][2] = v*t;
7    arrayTab[i][3] = delka;
8    i++;
9    pom_cas = pom_cas + krok_cas;
10 }
```

Pojďme od začátku. Pokud  $i$  jako počítadlo řádku nabude 100. řádku, tak se nastaví na první (na nultý). Dále k zapsání hodnot do tabulky nastane pouze tehdy, pokud čas  $t$  překročí čas nastavený pro další zápis. Dojde k zapsání jednotlivých hodnot do daných sloupců a počítadlo řádků se inkrementuje o jedna. Jako poslední se nastaví čas dalšího zápisu podle nastaveného kroku, který určuje jak často se mají hodnoty do tabulky zapisovat. Tento krok se nastavuje v buňce „field“ v okně pro tabulku vedle tlačítka „Vymazat“.

Jako poslední naprogramovanou funkcí je znázornění energií v levém dolním rohu appletu. Celé je tak počítáno na základě zákona o zachování energie.

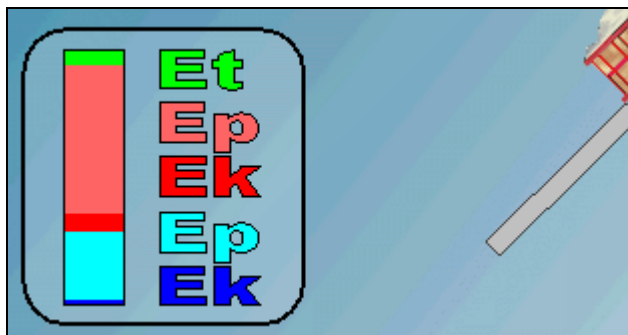
Kód zajišťující výpočet energií (Model/Fixed realltions/Energie):

```

2  double vyskap = 230 - (Math.sin(mUhel) * delka);
3  Ek = (0.5 * m1 * v * v);
4  Ep = (m1 * grav * vyskap);
5  double delka_start = Math.sqrt(((pocatek - delka) * (pocatek - delka)));
6  Et = Ft * delka_start;
8  Ek2 = (0.5 * m2 * v * v);
9  Ep2 = (m2 * grav * delka);
11 E = Ep + Ek + Ep2 + Ek2 + Et;
14 b_Ep2 = (Ep2 / (E/100)) * 1.3;

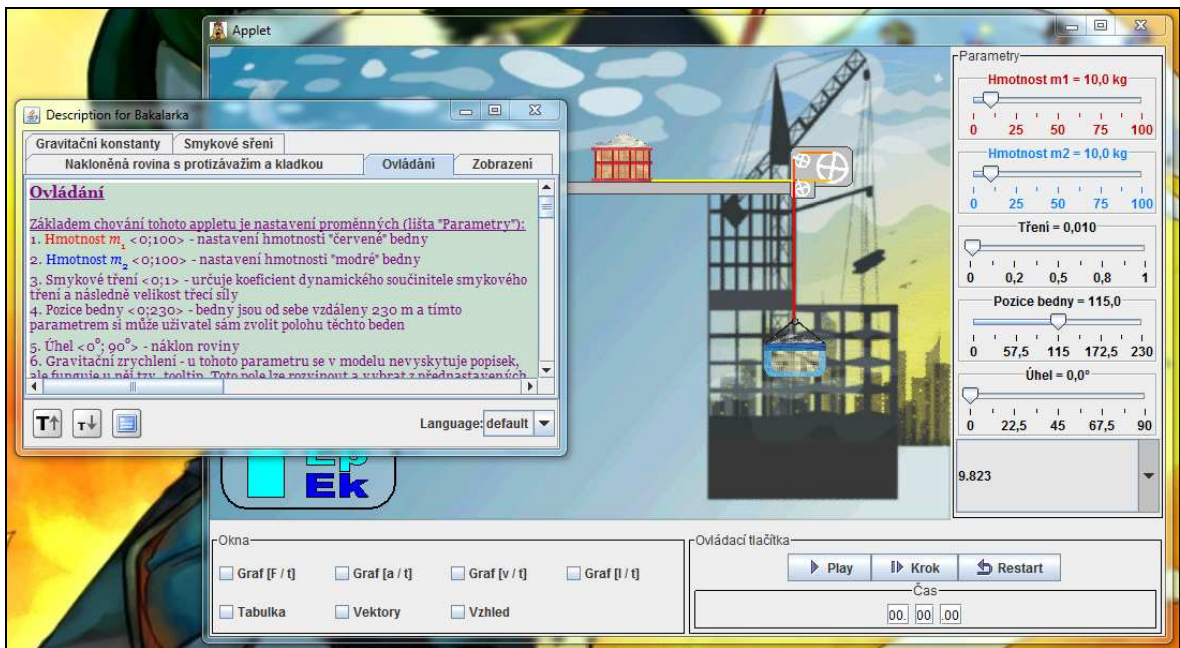
```

Tento kód počítá kinetickou, potenciální a třecí energii, kterou ve výsledku sečte a dostaneme tak celkovou energii soustavy. Nutno podotknout, že třecí energie je vlastně energie, která se třením změní na tepelnou. Abychom tak mohli udělat, tak musíme nejprve zjistit, jakou dráhu urazila bedna od spuštění appletu a tu pak vynásobit spolu s třecí silou - řádky (5) a (6). Dále vypočítáme kinetickou a potenciální energii obou beden a spolu s třecí je všechny sečteme. Samozřejmě musí platit to, že celková energie uzavřené soustavy musí být konstantní. Nyní nám už stačí znázornit jednotlivé energie do appletu. To obstarává řádek (14), který je v programu obsažen dohromady 5×, pouze s jinou proměnnou. Každý řádek obsluhuje jiný díl energie – viz. Obr. 19.

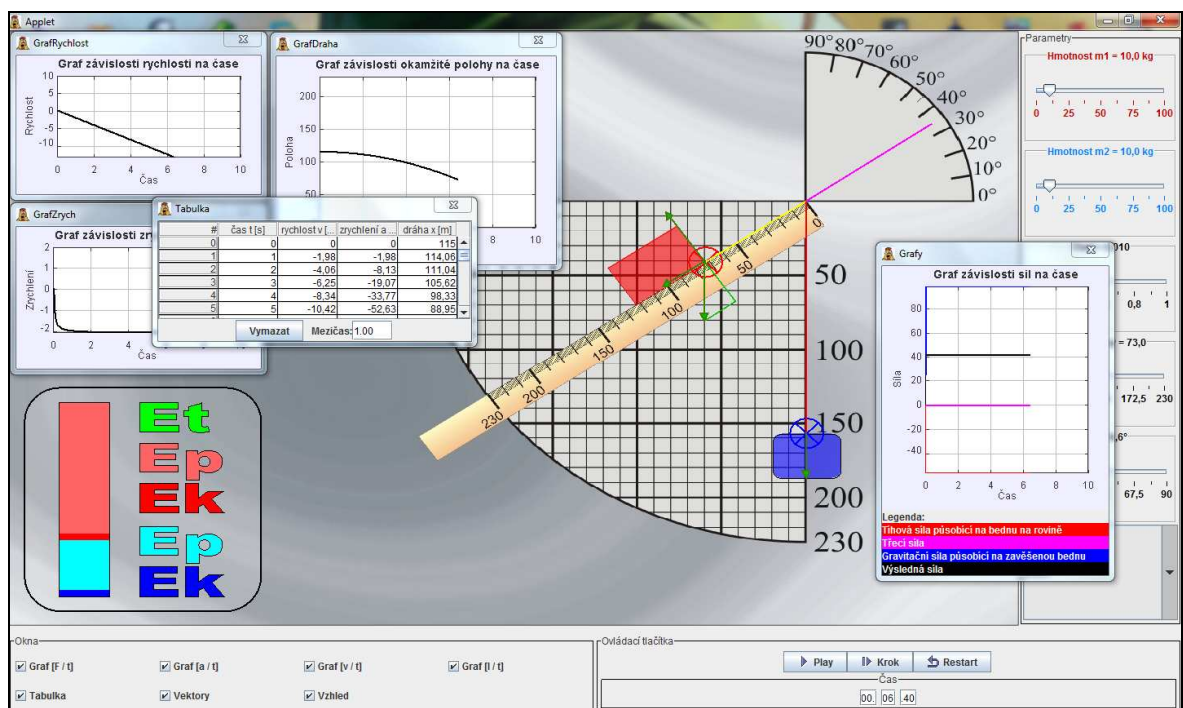


Obr. 19. Znázornění dílů energií.

Tímto jsme udělali poslední krok k realizaci našeho appletu. Nyní už stačí jej zapnout (Obr. 20 a Obr. 21) a používat jej jak každý potřebuje.



Obr. 20. Finální verze appletu – první spuštění.



Obr. 21. Finální verze appletu – se zapnutým zviditelněním oken a vzhledem.

## 5 ZHODNOCENÍ APLIKACE A JEDNOTLIVÝCH ČÁSTÍ

Pokud si podrobně všimneme Obr. 21 vidíme, že prezentuje skutečnosti, které naprogramovaný applet nabízí: zkoumání závislosti rychlosti, polohy, zrychlení jako funkci času, tak i působící síly a jednotlivé složky celkové mechanické energie při nastavených parametrech soustavy. Z detailu obrázku 21 je též vidět, že student má možnost nejen pracovat s grafy, ale také s daty nahoře uvedených veličin. Jejich přenesení do vlastního počítače mu otvírá prostor do fyzikálních aktivit s naměřenými daty podle stupně erudovanosti. Tým se liší předložená simulace od mnohých, volně přístupných na Internetu, které ve většině případech výstup dat neumožňuje.

Je těžké zhodnocovat své dílo a přitom se vyhnout tomu, aby to znělo nezaujatě. Dokázal jsem vytvořit plně funkční applet v prostředí Easy Java s použitelným výstupem a pestrými ovládacími prvky. Dá se říci, že finální verze je verzí druhou. Předložená verze appletu od započetí práce na něm prošla spoustou úprav a vylepšení.

U většiny prvků funguje nápověda ve formě tooltipu. Samotné ovládání za pomoci tlačítek umožňuje uživateli vrátit se do počátečního stavu aplikace po spuštění bez znovu spouštění. Navíc umožňuje měnění parametrů za běhu a tím pádem zjistit, jak by se taková pohybující se bedna chovala při nulové gravitaci. Dále obsahuje popis nastiňující funkci appletu, jehož součástí jsou například gravitační konstanty různých vesmírných těles a součinitel dynamického a statického tření pro různé materiály. Pro přehlednost je možno zapnout tzv. matematický vzhled, jenž zamění objekty a pozadí. Nesmíme také zapomenout na možnost zapnutí čtyř různých grafů a výstupu dat ve formě tabulky.

Závěrem je možno konstatovat, že jsem v programovacím jazyku Easy Java sestrojil applet, který napodobuje chování beden v situaci, jakou znázorňuje. Ověření fyzikální správnosti sestrojeného appletu bude ještě předmětem dalšího mého zájmu.

## ZÁVĚR

Práce se zaměřila na využívání nejnovějších interaktivních technologií ve vyučovacím procese. Představuje simulace jako vhodný didaktický prostředek pro lepší vizualizaci a pochopení různých reálných dějů.

Ze zadání práce vyplynulo, že na základě literární rešerši a nastudování základu programu Java, jsme měli jako hlavní cíl práce naprogramovat interaktivní simulaci v programu Easy Java. Jako pomůcka nám posloužili i volně přístupné simulace na Internetu spolu se zdrojovými kódy. I na jejich základě a nastudování fyzikální problematiky pohybu tělesa po nakloněné rovině spojeného s druhým tělesem lanem přes kladku, jsme navrhli a následně realizovali interaktivní simulaci s proměnnými parametry: hmotnost obou těles, smykové tření, úhel sklonu nakloněné roviny, gravitační zrychlení, která tvoří jádro praktické části bakalářské práce. Výsledkem zrealizované simulace je možnost sledovat časové závislosti fyzikálních veličin: polohy, rychlosti, zrychlení, působící síly a přeměnu jednotlivých forem celkové energie uzavřené ústavy. A také jsme odůvodnili, proč jsme pro praktickou část vybrali právě prostředí Easy Java.

Než jsme však dospěli k tomuto výsledku, v teoretické části jsme se zabývali jednak e-learningem jako formu výuky, při níž jsou využívány informační technologie, jako další most mezi učitelem a studentem. Těžiště teoretické části však tvořila problematika interaktivních simulací a to z pohledu jejich programování, tak i jak by měl vypadat správný postup při tvorbě didaktického softwaru. V neposlední řadě jsme nahlédly do početné množiny vývojových prostředí, spolu s Open Source.

Na závěr možno uvést, že programování interaktivních simulací s výstupem dat v současnosti ještě není všeobecně rozšířené. Applet byl realizovaný pro základní kurz fyziky a bude v části Mechanika ověřovaná jeho fyzikální správnost. Věříme, že výstup práce napomůže k zvládnutí objasňování fyzikálních zákonitostí zvolené problematiky pohybu. Z infromatického hlediska práce byla přínosem, protože bylo zvládnuté programování v Easy Javě, která má budoucnost pro další rozvoj e-learningu.

Na základě získaných zkušeností mohu říci, že při tvorbě simulace ve formě appletu je Java jedním z nejjednodušších jazyků jako takových. Vše mi přišlo logické a hned jasné po prostudování vhodných literárních pramenů. Navíc díky jeho výhodám dala vzniku Easy

Javě, která se nejspíše dočká nových inovací a vylepšení. Já osobně doufám, že Java s novými verzemi jazyka přinese do budoucna větší využití, než jaké má dnes.

Závěrem možná konstatovat, že všechny vytyčené cíle, uvedené v Úvodě předložené práce a v zadání byli splněny.

## ZÁVĚR V ANGLIČTINĚ

The thesis was focused on usage of the latest interactive technologies in educational process. It introduces simulations as an appropriate educational tool for better visualisation and understanding of various real processes.

It emerged from the thesis assignment, based on searching for literature and studying the foundations of Java program, that the main goal of the thesis is to create an interactive simulation in Easy Java program. As an aid we have used also free simulations on the Internet and source codes. Based on them and studying physical topic of body motion on the inclined plane tied together with another body by rope over the block we have proposed and then created the interactive simulation with variable parameters: weight of both bodies, friction coefficient, the inclined plane angle, and gravitational acceleration, which make the core of the empirical part of the bachelor thesis. The conclusion of created simulation is the opportunity to watch the time dependences of physical quantities: position, speed, acceleration, action force, and transformation of particular forms of total energy of the closed system. Also we have accounted for the choice of Easy Java in the empirical part.

Before approaching this goal in the theoretical part we have dealt with e-learning as a form of teaching which uses the information technologies as another link between teacher and student. The main point of the theoretical part is the issue of interactive simulations from the point of view of their programming as well as the point of view of correct procedure of creating educational software. At last but not least we have made survey of the large set of evolutionary environments, including Open Source.

We can conclude, that interactive simulation programming with experimental data is not commonly widespread in nowadays. The applet was made for the course of Physics Foundations and its physical accuracy will be verified in the topic called Mechanics. We believe, that the output of the thesis will contribute to explain the physical dependences of the issue of motion.

From the informatical point of view the thesis is valuable, because programming in Easy Java, which has the future in the further e-learning development, was successfully done. Based on the acquired knowledge I can say, that while creating the simulation Java is one of the clearest languages as such. Everything is logical and understandable after studying

appropriate literary sources. Furthermore, thanks to its advantages was created Easy Java which waits to see the innovations and improvements soon.

At the end we can conclude, that all set goals, mentioned in the Introduction and annotation of the bachelor thesis, was fulfilled.

**SEZNAM POUŽITÉ LITERATURY**

- [1] F. Esquembre. Easy Java simulations (Prentice Hall, Madrid, 2005). The Ejs program. 2007 [cit. 2011-03-01]. Dostupný z WWW:  
<<http://www.um.es/fem/Ejs>>.
- [2] Christian W., Belloni M. Physlet Physics: Interactive Illustrations, Explorations, and Problems for Introductory Physics. Davidson books.com. Aug 2003, 352p.
- [3] Interactive Simulations. University of Colorado at Boulder. Dostupný z WWW:  
<<http://phet.colorado.edu/research/index.php>>.
- [4] W. Christian, F. Esquembre. Modeling Physics with Easy Java Simulations. The Physics Teacher Vol. 45. November 2007, 475.
- [5] Paholok I. Simulácia ako vedecká metóda. E-logos Electronic Journal for Philosophy/2008. ISSN 1211-0442, 2008. Dostupný z WWW:  
<<http://nb.vse.cz/kfil/elogos/student/paholok08.pdf>>.
- [6] Wieman C.E., Adams W.K., Perkins, K.K. Science. PhET: Simulations That Enhance Learning. 322/682-683. October 2008.
- [7] Wieman, C.E., Perkins, K.K., Adams, W.K. Oersted Medal Lecture 2007: Interactive simulations for teaching physics: What works, what doesn't, and why, American Journal of Physics. 76, 393, May 2008.
- [8] Stoffová V., Stoffa J. Základné termíny z informačných, multimediálnych a didaktických technológií. In: Medacta, Nitra: UKF 1999. str. 64-69. ISBN-80-967746-2-X.
- [9] Beňuška J. Vyučovanie fyziky na gymnáziu s podporou grafických animovaných modelov realizovaných počítačovým prezentačným programom. Dostupný z WWW:  
<<http://www.infovek.sk/archivwebu/konferencia/2000/prispevky/fyzika.html>>.
- [10] Štědroň B. Open Source Software. Grada Publishing. Praha 2009. ISBN 978-80-247-3047-9. Dostupný z WWW:  
<[http://cs.wikipedia.org/wiki/Open\\_source\\_software](http://cs.wikipedia.org/wiki/Open_source_software)>.

- [11] Open Source Initiative. Dostupný z WWW:  
<<http://www.opensource.org/>>.
- [12] Open Source Software. Dostupný z WWW:  
<<http://www.root.cz/specially/licence/open-source-software/>>.
- [13] C programovací jazyk. Dostupný z WWW:  
<[http://wikipedia.infostar.cz/c/c\\_/c\\_programming\\_language.html#Features](http://wikipedia.infostar.cz/c/c_/c_programming_language.html#Features)>.
- [14] Java. Dostupný z WWW:  
<<http://newwiki.ceske-hry.cz/Java>>.
- [15] Python. Dostupný z WWW:  
<<http://newwiki.ceske-hry.cz/Python>>.
- [16] John M. Zelle, Ph.D. První jazyk: Python. Dostupný z WWW:  
<<http://macek.sandbox.cz/texty/prvni-jazyk-python/>>.
- [17] Simulační programovací jazyky. Dostupný z WWW:  
<[http://cs.wikipedia.org/wiki/Simula%C4%8Dn%C3%AD\\_programovac%C3%AD\\_jazyk](http://cs.wikipedia.org/wiki/Simula%C4%8Dn%C3%AD_programovac%C3%AD_jazyk)>.
- [18] Petr Hatina. Programování v jazyku Java (1) – Úvod. Dostupný z WWW:  
<[http://www.linuxsoft.cz/article.php?id\\_article=244](http://www.linuxsoft.cz/article.php?id_article=244)>.
- [19] Josef Kadlec. Obecné pojednání o programovacích jazycích. [cit. 2011-02-01].  
Dostupný z WWW:  
<[http://www.linuxsoft.cz/article.php?id\\_article=268](http://www.linuxsoft.cz/article.php?id_article=268)>.
- [20] Michal Vyskočil. Který programovací jazyk si vybrat? [cit. 2011-02-02].  
Dostupný z WWW:  
<<http://www.linuxexpres.cz/praxe/ktery-programovaci-jazyk-si-vybrat>>.
- [21] Simulace (e-learning). [cit. 2011-02-02]. Dostupný z WWW:  
<[http://cs.wikipedia.org/wiki/Simulace\\_%28e-learning%29](http://cs.wikipedia.org/wiki/Simulace_%28e-learning%29)>.

- [22] Jan Wagner. Nebojme se eLearningu. Česká škola. 2005.
- [23] Ožvoldová, M., Schauer, F., Lustig, F., Dekar, M. Real Remote Mass Spring Laboratory Experiments across Internet – Inherent Part of Integrated E- Learning of Oscillations. In: Conference ICL 2007. Villach : Kassel University Press. 2007. .Editor Michael E. Auer, International Association of Online Engineering, Wien, Austria, 10 pp. ISBN 978-3-89958-279.

**SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK**

$\pi$	Ludolfovo číslo.
$F_{g1r}$	Tíhová síla působící na těleso na nakloněné rovině.
$F_{t1}$	Třecí síla působící na proti směru pohybu tělesa na nakloněné rovině.
$F_{g2}$	Gravitační síla působící na zavěšené těleso.
$\alpha$	Úhel alfa.
t	Čas.
s	Dráha.
v	Rychlost.
a	Zrychlení.
m	Hmotnost.
g	Gravitační zrychlení (tíhové zrychlení v případě Země)
EJS	Easy Java
LMS	Learning Management Systém

**SEZNAM OBRÁZKŮ**

Obr. 1. Titulní strana simulace „Projectile Motion“ <a href="http://phet.colorado.edu">http://phet.colorado.edu</a> .....	13
Obr. 2. Zobrazení postavení simulace v rámci všeobecných vědeckých metod.....	19
Obr. 3. Zobrazení postavení simulace, pokud je modelování chápáno pouze jako proces tvorby modelu.....	19
Obr. 4. Logo Open Source Initiative.....	29
Obr. 5. Náčrtek appletu.....	34
Obr. 6. Náčrtek interface.....	35
Obr. 7. Buttons / tlačítka.....	36
Obr. 8. Slider / šoupátko – posuvná lišta.....	36
Obr. 9. Checkbox / zatrhávací rámeček.....	37
Obr. 10. Síly působící na jednotlivé objekty na nakloněné rovině.....	38
Obr. 11. Finální vzhled appletu.....	40
Obr. 12. Konzole Easy Javy.....	41
Obr. 13. Hlavní okno Easy Javy.....	42
Obr. 14. Choreographies – applet.....	44
Obr. 15. Okna a rozvržení interface.....	46
Obr. 16. Nastavení vlastností u šoupátka.....	48
Obr. 17. Stromová struktura tabulky.....	49
Obr. 18. Nastavení buňky „field“.....	50
Obr. 19. Znázornění podílů energií.....	58
Obr. 20. Finální verze appletu – první spuštění.....	59
Obr. 21. Finální verze appletu – se zapnutým zviditelněním oken a vzhledem.....	59

## SEZNAM PŘÍLOH

PI - DVD

## **PŘÍLOHA P I: DVD**

Obsahuje interaktivní simulaci, jakožto praktickou část práce, spolu se zdrojovým kódem a prezentací.