UNIVERZITA KOMENSKÉHO V BRATISLAVE

FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

Výukový program demonštrujúci fyzikálny princíp

Bakalárska práca

2013 **Jozef Belko**

UNIVERZITA KOMENSKÉHO V BRATISLAVE

FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

Výukový program demonštrujúci fyzikálny princíp

Bakalárska práca

Evidenčné číslo: 05d5d894-ceca-4bae-9976-bde41c93683d

Študijný program: Aplikovaná informatika

Študijný odbor: 9.2.9. aplikovaná informatika

Školiace pracovisko: Katedra aplikovanej informatiky

Školiteľ: Mgr. Pavel Petrovič, PhD.

Bratislava, 2013 **Jozef Belko**

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Jozef Belko
Študijný program: aplikovaná informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: 9.2.9. aplikovaná informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský

Názov: Výukový program demonštrujúci fyzikálny princíp

Cieľ: Podľa výberu - špecfikovať, navrhnúť a implementovať výukový program pre stredoškolskú fyziku, ktorý vysvetľuje nejaký fyzikálny princíp pomocou simulácie podľa výberu. Napr. problematika obvodov so striedavým prúdom vysvetlená/odvodená pomocou komplexných čísel (kapacitancia, induktancia, atď) alebo modely atómu, teória relativity, alebo elementárnych častíc, alebo momenty rotačného pohybu, momenty zotrvačnosti, vzťažné sústavy a pod.

Literatúra: L.Slovák: Výukový program demonštrujúci fyzikálny princíp, bakalárska práca, FMFI UK, Bratislava, 2011.
Oracle: Java FX API Documentation, online: http://docs.oracle.com/javafx/2/api/index.html

Vedúci: Mgr. Pavel Petrovič, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: doc. PhDr. Ján Rybár, PhD.

Dátum zadania: 15.10.2012

Dátum schválenia: 24.10.2012

doc. RNDr. Mária Markošová, PhD.
garant študijného programu

Belko
študent

Pavel Petrovič
vedúci práce

Čestné prehlásenie

Čestne prehlasujem, že bakalársku prácu som vypracoval samostatne

s použitím literatúry a zdrojov uvedených v závere práce.

V Bratislave, ...

Jozef Belko

Poďakovanie

Chcel by som poďakovať hlavne
vedúcemu mojej práce Mgr. Pavlovi Petrovičovi, PhD. za cenné rady, odbornú pomoc, inšpiráciu, trpezlivosť a optimistický prístup. Chcel by som poďakovať tiež Mgr. Jozefovi Genzorovi za pomoc s fyzikálnou časťou práce. Ďalej by som chcel poďakovať svojej rodine za podporu a pochopenie obzvlášť mojej mame, ktorá sa starala o moju gramatiku, svojej priateľke za motiváciu, bez ktorej by som to určite nezvládol a všetkým, ktorí mi akokoľvek pomohli a podporovali ma pri vypracúvaní mojej bakalárskej práce.

Abstrakt

Cieľom našej bakalárskej práce bolo vytvoriť webovú aplikáciu určenú na výučbu fyziky pre stredné školy. Úlohou aplikácie bolo simulovať viaceré fyzikálne princípy a testovať nadobudnuté vedomosti študenta. Aspekty fyzikálnych princípov, ktoré sme simulovali, boli vybrané podľa východísk a následnej špecifikácie. Špecifikácia cieľov zahrnula tiež to, že celý program mal byť modifikovateľný a rozširovateľný, podľa čoho sme následne navrhli architektúru programu, a na základe nej celý program implementovali. Celý program je rozdelený na dve časti a to applet a webové rozhranie pre prezeranie úloh. Tieto časti sú vzájomne nezávislé. Výsledkom našej bakalárskej práce je webová aplikácia pripravená na používanie.

Kľúčové slová: webová aplikácia, simulácia, komponent

Abstract

The purpose of our bachelor thesis was to create a web application designated for teaching physics in secondary schools. Task of this application was to simulate several physical principles and to test the acquired knowledge of students. The aspects of simulated physical principles were chosen according to background knowledge and consequent specification. The specification of the application included the fact, that all parts of the program should be modifiable and expandable. After that we designed the architecture of the program and implemented it. The program is devided in two parts, applet and web interface for viewing tasks. These two parts are independent of each other. The result of our bachelor thesis is a web application ready to be used.

Key words: web application, simulation, component

Obsah

Úvod

Informačné technológie sa dnes využívajú takmer všade. Život bez nich si už ani nevieme predstaviť. Listy sme vymenili za emaily, obchody za e-shopy, knihy meníme za e-booky, hračky za videohry, školské učebnice a tabuľu za výukový software.

Fyzika je odbor, ktorý je veľmi zaujímavý a pre každého je prínosom. Je to však aj odbor, o ktorý v dnešnej dobe upadá záujem. Okrem toho, že je fyzika zaujímavá, je tiež pre ľudí dôležitá. Koniec koncov fyzikálne zákony a princípy môžeme nájsť všade okolo nás. Preto je dobré, keď ľudia vedia o ich existencii, vedia aj prečo existujú a na akom princípe fungujú. Navyše všeobecný vývoj ľubovoľných technológií je väčšinou opretý o fyzikálne javy. Tie sú totiž niekedy obmedzujúcim faktorom a niekedy hlavným „pohonom“ danej technológie.

V našej práci si dávame za cieľ vytvoriť program, ktorý bude rôzne fyzikálne princípy ukazovať primárne študentom stredných škôl. Program má fyzikálne princípy simulovať tak, aby mal študent možnosť vizuálne sledovať čo sa deje v situácií, v ktorej prebieha daný fyzikálny jav. Pre ich úplné pochopenie je potrebná aj teória ich funkcionality, príkladu či využitia v praxi.

Keďže tento program má slúžiť na výuku študentov, bude obsahovať možnosť otestovať, čo sa študenti naučili o princípoch a to prostredníctvom riešenia úloh. Aby malo riešenie úloh nejaký význam, bude k nemu vytvorená aj možnosť, aby mohol učiteľ študentom riešenia hodnotiť a dávať im k nim spätnú väzbu. Spätnou väzbou si študenti ešte viac rozšíria svoje vedomosti. Doplnia si tak práve vedomosti, ktoré im pri riešení danej úlohy chýbali. Aby bolo možné riešenia hodnotiť a prezerať hodnotenia týchto riešení, či už kvôli spätnej väzbe alebo hodnoteniu fyziky ako predmetu v škole, bude potrebná možnosť prezerať tieto riešenia. V prípade študentov vlastné riešenia a v prípade učiteľov všetky riešenia. Vytvoriť rozhranie pre tieto účely si naša práca tiež dáva ako cieľ.

V našej práci budeme vytvárať simulácie 4 fyzikálnych princípov a to princíp zotrvačníka, gyroskopu, kladky a Coriolisovej sily. Keďže už len mechanika, do ktorej patria všetky naše princípy, obsahuje nespočetne veľa princípov, fyziku kompletne pokryť nemôžeme ani náhodou. Avšak ako základ alebo príklad pre náš program budeme používať tieto princípy. Samotné princípy budú opísané vo východiskovej kapitole tak ako programovacie jazyky a prostriedky, ktoré budeme počas implementácie používať.

Keďže program, ktorý by obsahoval iba 4 princípy, by nebol prakticky využiteľný, našou snahou bude vytvoriť program, pri ktorom sa očakáva, že budú pridávané ďalšie princípy. A taktiež sa bude očakávať, že bude prenositeľný, aby mohol byť využívaný

v ľubovoľnej organizácií (napríklad v škole). V tretej kapitole budeme hovoriť podrobne o cieľoch našej práce, ktorými bude aj rozšíriteľnosť a prenositeľnosť. Návrh programu, ktorý bude rozoberaný vo štvrtej kapitole, sa bude riadiť tým, aby program mohol splniť naše ciele alebo aby sa im aspoň čo najviac priblížil.

Z implementácie vyberieme časti, ktoré sme považovali za zaujímavé, problematické alebo sme uvážili že sú natoľko dôležité, aby sme im venovali pozornosť. Toto všetko bude v piatej kapitole. Poslednou kapitolou je záver, v ktorom sme zhodnotili svoju prácu, jej výsledky, dosiahnuté ale aj nedosiahnuté ciele.

2 Východiská

V tejto kapitole sme sa venovali didaktickému software-u, potom fyzike, konkrétne jej časti - mechanike a jednotlivým princípom, ktoré bude náš program simulovať. Na záver ešte spomenieme jazyky Java a PHP, ktoré sme si vybrali na tvorbu nášho programu.

2.1 Didaktický software

Didaktický software slúži žiakom a študentom na učenie prostredníctvom skúmania. Takýto software musí mať vhodné rozhranie navrhnuté pre poznávanie a učenie sa, musí byť tiež interaktívny, aby mal študent pocit, že s ním program komunikuje. Software musí byť tiež schopný prispôsobovať sa potrebám študenta, musí byť nastavovateľný, obsahovať širokú škálu možností, ktoré sa týkajú problému, ktorý sa má študent naučiť. Naopak nesmie obsahovať priveľa textov, ktoré by študenta odrádzali od toho, aby sa štúdiu venoval. Úlohou každého výukového software-u je zaujať študenta. Študent zaujatý témou, vzhľadom, či funkciami softwaru, sa naučí preberané učivo ľahšie. Preto program využíva vizualizáciu, simuláciu či multimédiá, vďaka čomu sa tiež zvyšuje pravdepodobnosť, že študent preberané učivo hneď pochopí. Didaktický software má svoj zámer, svoj edukačný cieľ. Náš software bude určený pre stredné školy, bude vyučovať fyziku, konkrétne niektoré princípy mechaniky. Preto každá časť software-u bude zameraná na daný princíp a oddelená od ostatných, ktoré s nim priamo nesúvisia.

Výukový software okrem vyučovania konkrétneho predmetu či preberaného učiva vždy vyučuje čiastočne aj informatiku. Informatická gramotnosť je v dnešnej dobe dôležitá, lebo sa využíva vo všetkých pracovných odvetviach. Preto je tento „bočný efekt“ výukových softwarov veľkým plus. Ďalším takzvaným bočným efektom je individuálny prístup. Učiteľ nie je schopný venovať sa všetkým študentom súčasne, naopak náš program áno, lebo každý študent má vlastnú kópiu programu.

Tak, ako sa menia učebné osnovy, musí byť aj software na ich výuku pripravený na zmeny. Takto sa môže software vyvíjať, rozširovať a upresňovať rozsah toho, čo bude vyučovať. Tiež je potom možné meniť jeho zámer, nakoľko je program riadený študentom alebo obsahuje možnosti, ako sa môže študent otestovať, alebo či môže učiteľ hodnotiť žiakov aj za prácu, ktorú vykonajú v rozhraní softwaru.

2.2 Fyzika - Mechanika

Mechanika je odbor fyziky, ktorý sa zaoberá pohybom a silami. Zaoberá sa javmi, ktoré prebiehajú úplne všade a ktoré každý deň môžeme kdekoľvek vidieť. Každý predmet na zemi sa nejako pohybuje, na každý predmet pôsobia rôzne sily a majú naň nejaký vplyv. Práve preto, že sú to javy, s ktorými sa denno-denne stretáva každý z nás, je to oblasť fyziky, ktorú by mal naozaj každý ovládať. Mechaniku sme si vybrali práve preto, že je prvým odborom fyziky, ktorý sa majú študenti stredných škôl naučiť. Je akýmsi odrazovým mostíkom k ďalším odborom, preto je dôležité, aby mechaniku dobre ovládali. Náš software sa bude snažiť vysvetliť podstatu niektorých princípov mechaniky.

2.2.1 Zotrvačník

Téma zotrvačníka sa zaoberá hlavne momentom zotrvačnosti, hybnosti a sily. Základný princíp zotrvačníka spočíva v pohybe po kružnici. Avšak zotrvačník sa nachádza v trojrozmernom priestore. Preto je to akási nadstavba na pohyb po kružnici.

Moment zotrvačnosti tuhého telesa je možné vypočítať súčtom momentov zotrvačností jeho častíc. Keďže hmota telesa je rozložená spojite, súčet môžeme nahradiť integrálom, a tak ho vypočítame ako:

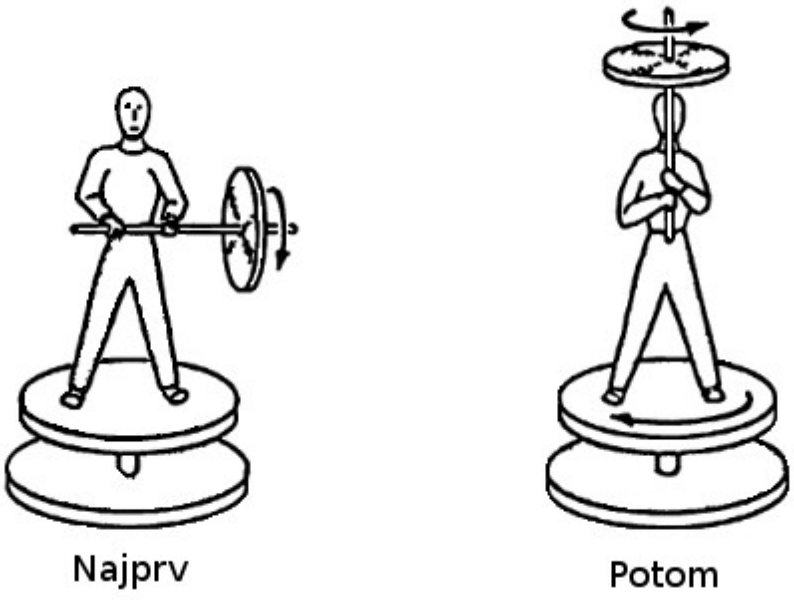
$$I = \int r^2 \, \mathrm{d}m$$

Pri zotrvačníku zohráva dôležitú úlohu aj rýchlosť. Rýchlosť rotácie je súčtom vektorov rotácií. Všetky rovinné rotácie sa dajú zovšeobecniť pre trojrozmerný prípad, rýchlosť častice je potom

$$\boldsymbol{v} = \boldsymbol{\omega} \times \boldsymbol{r}$$

kde *v* je rýchlosť rotácie, ω je uhlová rýchlosť a *r* je polomer. Tak ako sme vyjadrili rýchlosť rotácie, uhlovú rýchlosť a polomer pomocou vektorov, tak budeme pokračovať aj s ďalšími veličinami. Vektor momentu sily sa totiž rovná časovej zmene vektoru momentu hybnosti. Teda

$$\boldsymbol{N} = \frac{\mathrm{d}\boldsymbol{L}}{\mathrm{d}t}$$



Obrázok 1

Teraz keď sme ukázali fakty, ktorých sa princíp týka si predstavme zotrvačník ako je na obrázku 1, človek stojí na platni, ktorá môže rotovať a drží teleso, ktoré rotuje. Os telesa je kolmá na os platne, na ktorej stojí. Kým je os telesa horizontálne, jej hybnosť vzhľadom na vertikálnu je nulová. Keď sa potom presunie do vertikálnej polohy, hybnosť stále ostane nulová, čo však spôsobí, že človek sa na platni točí tiež avšak v opačnom smere. Vyplýva to zo zákona zachovania hybnosti spomínaného vyššie. Keďže dL vieme napísať ako L_0dv, vieme tiež odvodiť

$$\boldsymbol{N = L_0 \times \omega}$$

a keďže vektory L_0 a ω majú horizontálny smer, potom N má smer vertikálny. Na vytvorenie takéhoto momentu sily sú potrebné sily F a -F na koncoch osi telesa. Podľa tretieho Newtonovho zákona vieme, že tieto sily sú opačné a rovnako veľké. Po otočení do vertikálnej polohy teda na človeka pôsobí táto sila, na opačnej strane musí pôsobiť rovnaká sila, preto sa platňa, na ktorej stojí človek, točí do opačnej strany. [1]

2.2.2 Gyroskop

Gyroskop je zariadenie na meranie, alebo udržiavanie rovnakej orientácie, teda rovnakého smeru. Funguje na princípe zákona zachovania momentu hybnosti. Gyroskop ma vo svojom jadre zotrvačník. Tento zotrvačník sa točí okolo osi, ktorá je pripevnená na pohyblivom ráme (rám sa môže otáčať) a tento rám je pripevnený vlastnou osou na ďalšom

ráme. Zotrvačník sa teda otáča okolo zvislej osi a rámy sa môžu otáčať okolo zvyšných dvoch kolmých osí. Gyroskop je na obrázku 2. Zotrvačníkom je nejaký predmet, ideálne kruhového tvaru, ktorý ma hmotnosť sústredenú čo najďalej od osi okolo ktorej rotuje. Tiež je podmienkou funkčnosti gyroskopu vysoká rýchlosť rotácie zotrvačníka.

Obrázok 2

Gyroskop má niekoľko zaujímavých vlastnosti. Ak sa zotrvačník točí s osou v zvislej polohe môžeme jeho rámy ľubovoľne otáčať, no smer rotácie zotrvačníku sa nezmení. Je to tak preto, že hybnosť zotrvačníka vzhľadom na iné osi je nulová. Ak sa zotrvačník točí v horizontálnej polohe, časom sa bude vychyľovať. Gyroskop sa vychyľuje podľa natočenia zeme, teda na severnom póle by sa otočil o 360° za 24 hodín, a naopak, na rovníku by sa nevychýlil vôbec. Gyroskop obsahuje zotrvačník a tak funguje na podobnom princípe. Gyroskop s jedným voľným rámom (s možnosťou otáčania) sa chová úplne rovnako ako zotrvačník, pohyblivý rám sa točí v opačnom smere ako stredný kotúč.

Jednou z vlastností gyroskopu je tiež to, že jeho vyosenie spôsobí, že jeho os sa pohybuje po cykloide. Znamená to, že ak máme zotrvačník, ktorého os je v horizontálnej polohe a podopretá je len na jeho koncoch a ak podperu jedného konca odstránime, zotrvačník začne padať, ale vďaka svojmu momentu zotrvačnosti nespadne, ale začne sa kývať hore a dole až sa ustáli. Keby boli jeho ložiská dokonalé, po odstránení jednej podpery by bol ustálený a ostal by v rovnakej výške. Avšak takýto zotrvačník, ktorý ma podopretú iba jednu stranu osi okolo ktorej rotuje, rotuje aj okolo vertikálnej osi, ktorá je jeho jediná podpera. Pri vyosení gyroskopu teda pôsobí jedna zo síl F či -F, ktoré boli spomínané vyššie. Keďže tieto sily záležia od uhlovej rýchlosti, teda rýchlosti rotácie a hybnosti, ktorá záleží od hmotnosti

tak tieto sily sú značne veľké. Veľká sila, ktorá vzniká náhle je pre človeka rôznorodo využiteľná. [2]

2.2.5 Coriolisova sila

Coriolisova sila je ďalší princíp, ktorý súvisí s pohybom po kružnici. Táto sila totiž vzniká všade tam, kde niečo rotuje. Je to sila, ktorá sa prejavuje pri pohybe v rotujúcom systéme. A keďže zem je rotujúci systém, táto sila pôsobí na každého z nás, hoci ju v tomto prípade nepociťujeme, ani si jej existenciu neuvedomujeme.

Coriolisova sila je nepravá sila a je to sila, ktorá pôsobí „do strany“. Ak sa teleso pohybuje v rotujúcom systéme, moment hybnosti tohto telesa je

$$L = m\, v_t\, r = m\, \omega\, r^2$$

Ak sa toto teleso pohybuje po polomeri, uhlová rýchlosť ω sa nemení a tak možno vyrátať Coriolisovu silu takto

$$F_c = 2\, m\, \omega\, v_r$$

Z pohľadu zvonka je vidno, že teleso sa pohybuje po krivke, čo je spôsobené rotáciou. Na to aby sa toto teleso pohybovalo po krivke je potrebná sila, ktorá mu dodáva zrýchlenie a to je práve Coriolisova sila.

Coriolisova sila pôsobí na teleso aj keď sa pohybuje po obvode. Ak sa teleso pohybuje rýchlosťou v' v rotujúcom systéme z pohľadu z vonka sa teleso pohybuje rýchlosťou $v=v'+\omega r$. Z pohľadu zvonka vidíme teda pôsobiť silu $F_r = - m\, v^2 / r$. Naopak z pohľadu z rotujúceho systému je sila pôsobiaca na teleso

$$F_r = - mv'^2/r - m\omega^2 r - 2mv'\omega$$

Táto sila sa teda skladá z troch častí prvá a druhá časť tvoria odstredivé sily a posledná časť je Coriolisova sila. V tomto prípade je ale záporná. Coriolisova sila pôsobí na teleso nech sa pohybuje v rotujúcom systéme v akomkoľvek smere. Vzhľadom na smer rýchlosti pôsobí Coriolisova sila vždy rovnakým smerom, pôsobí vždy kolmo na smer rýchlosti. A veľkosť Coriolisovej sily je vždy

$$\mathbf{F_c = 2\, m\, \omega\, v}$$

Coriolisovu silu ale môžeme vidieť vďaka satelitným snímkom zeme. Sú to vzdušné víry, ktoré sa točia kvôli Coriolisovej sile. Keďže zem je guľatá, každá jej polovica je naklonená do opačnej strany. Znamená to, že Coriolisova sila pôsobí na opačnej strane zemegule do opačnej strany. Preto sa aj tornáda a iné vzdušne víry točia na južnej pologuli do

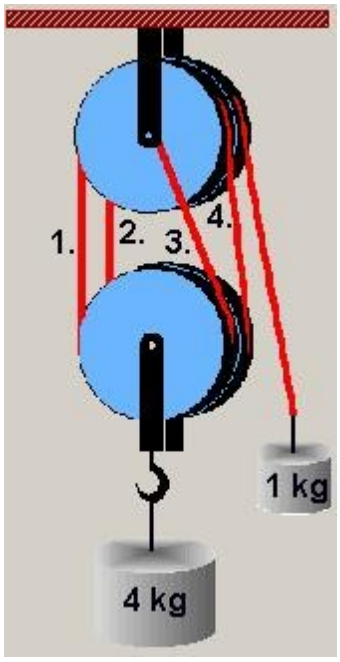
opačnej strany ako na severnej pologuli. Názorne je to ukázané na obrázku 3. Tiež voda sa točí do opačných strán na opačných pologuliach zeme. Avšak ak vypustíme umývadlo plné vody vír, ktorý sa vytvorí, kým voda z umývadla vytečie, nie je spôsobený Coriolisovou silou ale drobnými prúdmi vo vode. Je to častý omyl. [3]

Obrázok 3

2.2.3 Kladky

Kladky poznáme pevné a voľné. Používajú sa na uľahčenie práce. Pevná kladka je upevnená v osi otáčania, lano prechádza cez kladku a zatiaľ čo na jednej strane lana sa nachádza nejaký predmet, poprípade závažie, na druhej strane pôsobí sila, aby bol tento systém vyvážený. Sila na druhej strane lana musí byť rovnako veľká ako sila, ktorou je priťahované závažie k zemi. Rozdiel je však v smere sily. Vo väčšine prípadov je opačný.

Voľná kladka visí na lane, pričom jeden koniec lana je upevnený a na druhý pôsobí sila. Na osi voľnej kladky je upevnené nejaké teleso alebo závažie. Sila pôsobiaca na jeden koniec lana túto sústavu vyvažuje, avšak táto sila je rovná polovici sily, ktorou sú priťahované závažie s kladkou k zemi. Keď chceme závažie vytiahnuť hore, musíme ťahať dvakrát dlhšie lano.



Obrázok 4

Pevnú a voľnú kladku môžeme kombinovať. Kombinovanie pevnej a voľnej kladky sa nazýva kladkostroj. Kladkostroj vidíme na obrázku 4. Pomocou viacerých kladiek je možno dosiahnuť, že na vytiahnutie závažia bude treba omnoho menšiu silu akou je teleso priťahované k zemi, vždy je to však za cenu väčšej dĺžky lana, ktorú musíme vytiahnuť. Sila, ktorú treba na vyváženie závažia zaveseného na kladkostroji je

$$\mathbf{F = m\,g\,/\,n}$$

kde *m* je hmotnosť závažia a voľných kladiek, *g* je gravitačné zrýchlenie teda $g=9{,}80665 ms^{-2}$ a *n* je počet kladiek, v prípade že najbližšia kladka k pôsobiacej sile je pevná, teda sila pôsobí smerom nadol. Ak je posledná kladka pevná, nehrá rolu. Je iba obyčajnou pevnou kladkou, ktorá iba mení smer pôsobiacej sily. A dĺžka lana, ktorú treba potiahnuť, aby sme závažie zdvihli o výšku *h* je *nh*. [4]

2.3 Simulácia

Simulácia je napodobňovanie či imitovanie reálnej situácie pomocou informačných technológií. Úlohou simulačného procesu je zistiť, ako sa bude správať pre zadané vstupné dáta či parametre.

My budeme simulovať fyzikálne princípy. Našou úlohou bude teda čo najväčšmi priblížiť vizuálnu stránku software-u k prirodzeným reálnym podmienkam. Niektoré princípy budú simulovať trojrozmerný priestor a pohyb v ňom.

2.4 Programovací jazyk Java

Jazyk Java začala vyvíjať v roku 1991 firma Sun Microsystems. Jazyk mal byť určený pre vstavané systémy (bežné elektrické zariadenia) a mal zdieľať princípy jazykov C a C++. Pôvodne sa volal Oak, avšak zistilo sa, že programovací jazyk s takým menom už existuje a tak v bufete padol nápad Java čo voľne preložené znamená „kafé“. Pôvodný zámer jazyka sa nedaril, avšak vedomie, že sa dá využiť na internete viedlo k ďalšiemu vývoju. V roku 1995 bola Java oficiálne predstavená. Java a stala úspešnou vďaka rozvoju internetu, ale svoje využitie našla aj ako obyčajný programovací jazyk. [6]

Podľa Tiobe, ktorá meria popularitu programovacích jazykov, bola Java v roku 2012 najpopulárnejším programovacím jazykom, v januári 2013 bola na druhom mieste. [5]

Spracovanie programu v Jave ma 5 fáz, editovanie kódu, preklad alebo kompilácia, zavedenie, overenie a vykonávanie. Overovanie je odlišnosťou oproti ostatným jazykom, ale je krokom k bezpečnosti software-u pre používateľov. Zdrojové kódy Javy sa oproti iným jazykom prekladajú do bajtkodu. Bajtkod znamená nezávislosť na platforme, ale aj potrebu interpretera. Java je teda interpretovaný jazyk. [6]

Pri tvorbe software-u budeme používať Javu 1.6. Jazyk Java sme si vybrali preto, že je multiplatformový, kvôli jednoduchosti tvorby grafického rozhrania pre ľubovoľnú platformu. Grafické prostredie funguje na princípe kontajnerov. Vybrali sme si ju tiež kvôli bezpečnosti, ktorú Java zaručuje aj vyhadzovaním výnimiek. To sa deje v neočakávaných situáciách ako keď chce program otvoriť neexistujúci súbor alebo zapisovať mimo rozsahu pola.

2.4.1 IDE Eclipse

Eclipse je vývojové prostredie pre Javu. Vzniklo v roku 2001, na jeho vývoji sa podieľali firmy ako IBM, Borland, Red Hat, SuSE a iné. Neskôr v roku 2004 Eclipse zmenil licenciu na Open source. Eclipse funguje na princípe pluginov, takzvaných zásuvných modulov. Jeho vývojári sa pomocou nich snažia aby bol univerzálnym vývojovým prostredím. Vďaka pluginom podporuje viaceré programovacie jazyky a mnohé iné vymoženosti. [7]

Vývojové prostredie Eclipse sme si vybrali pre skúsenosti, ktoré s nim máme. Tiež pre minimum problémov, na ktoré sme pri práci s ním narazili. Samozrejme aj pre jeho flexibilitu a prispôsobiteľnosť.

2.5 Jazyk PHP

PHP je skriptovací jazyk, ktorý vznikol v 90-tých rokoch dvadsiateho storočia. Tento jazyk sa používa na tvorbu dynamických webových stránok. Funguje na princípe vkladania skriptov do html kódu Tieto skripty vlastne generujú html kód, ktorý server s podporou PHP odošle užívateľovi. [8] Tento jazyk v práci využijeme preto lebo prístupnosť cez internet znamená, že bude program bližšie pri užívateľovi. Užívateľ tak nemusí nič inštalovať, program mu nezaberá miesto na disku. Jedine čo užívateľ potrebuje, je pripojenie k internetu a webový prehliadač. Tieto podmienky dnes spĺňa absolútna väčšina počítačov na školách, čo je faktor, ktorý nás zaujíma kvôli zameraniu nášho programu, ale aj osobných počítačov, notebookov, tabletov a podobných zariadení schopných používať náš program.

2.6 Architektúra MVC (model-view-controller)

MVC alebo model-view-controller je softwarová architektúra, ktorá rozdeľuje dátový model, užívateľské prostredie a riadiacu jednotku. Robí jednotlivé komponenty nezávislými, čo znamená, že zmena niektorého z komponentov len málo ovplyvní ostatné. Všetky komponenty sú vzájomne previazané, komunikujú spolu. Najčastejšia implementácia vyzerá tak, že užívateľ robí s vizuálnym rozhraním (view), kde niečo odošle riadiacej jednotke (controller) a tá vykoná zmeny v dátovom modeli (model). Potom komponent pre vzhľad aktualizuje vizuálnu stránku aplikácie podľa nového stavu dátového modelu. Potom sa celý cyklus opakuje znova. [9]

Túto architektúru využíva viacero aplikácií a frameworkov. Programátorom či vývojárom tak uľahčuje robotu. Problém delí do troch častí, ktoré sú oddeliteľné práve preto, že naozaj spolu súvisia iba málo. Stačí im jednoduchá komunikácia. Z rovnakých dôvodov budeme túto architektúru na riešenie niektorých problémov využívať aj my.

2.7 Existujúce riešenia

Podobný program bol vytvorený v roku 2011 ako bakalárska práca. Avšak autor tejto práce mal zhodné iba všeobecné ciele. Jeho práca obsahuje simulácie princípov z fyziky mikrosveta a samotný program je desktopovou aplikáciou. Jeho práca tiež neobsahuje možnosti testovania znalostí, ktoré užívateľ pri práci s programom získal na takej úrovni, ako bude obsahovať náš program. Avšak rozhranie tohto študenta bolo jednoduché a intuitívne, a

preto sme sa rozhodli toto rozhranie napodobniť. Podobnosť rozhraní bude znamenať pre používateľov, že sa nebudú musieť zoznamovať s novým prostredím, ak prechádzajú od jedného programu k druhému. [10]

3 Špecifikácia cieľov

Cieľom práce bolo vytvoriť výukový program pre stredné školy, ktorý vyučuje niektoré fyzikálne princípy, poprípade dáva do pozornosti zaujímavosti, ktoré sú nad rámec stredoškolského učiva. V prípade všetkých princípov objasňuje ich funkčnosť, od čoho závisia, ako je možné, že tieto javy vôbec existujú.

3.1 Všeobecná špecifikácia

3.1.1 Program

Hlavnou časťou programu bude java applet, ktorý bude prístupný online a teda bude k nemu jednoduchý prístup, lebo študenti a učitelia nebudú musieť program nijak spúšťať. Spustí sa sám pri načítaní stránky, prostredníctvom ktorej bude používaný.

Tento applet bude obsahovať ako hlavné časti fyzikálne princípy, medzi ktorými sa bude môcť študent či učiteľ pohybovať. Rozhranie bude jednoduché, aby bolo ovládanie appletu intuitívne. Každý princíp bude tiež obsahovať možnosť prečítať si teóriu k danému princípu. Teória bude stručná, opisujúca či už samotný princíp ako taký, tak aj jeho funkčnosť a dôvody jeho existencie. Teória bude tiež obsahovať nejaké fyzikálne vzorce, ktoré budú dopĺňať samotnú teóriu. Teória princípu bude tiež vysvetlená na praktickom príklade.

Súčasťou princípov bude možnosť riešenia úloh. Riešenie úloh bude mať mnoho možnosti. A bude siahať až mimo appletu. Avšak všetko bude realizované v rámci jednej webovej stránky. Program bude teda zahŕňať výučbu pomocou simulácie a teórie, ale tiež aj overenie, či študent preberanú látku pochopil pomocou riešenia úloh a tiež vďaka možnosti ich hodnotenia učiteľmi. Možnosťou učiteľov bude tiež dávať žiakom spätnú väzbu k úlohám.

3.1.2 Úlohy

Študenti budú môcť riešiť úlohy, ktoré po vyplnení formulára budú môcť odovzdať odoslaním. Formulár bude obsahovať meno, priezvisko, triedu a priestor na riešenie úlohy. Priestor na riešenie úlohy bude obsahovať pole, do ktorého bude môcť študent napísať výsledok, ale okrem toho bude obsahovať aj pole na komentáre, slovné odpovede či pomocné výpočty v prípade, že by učiteľ chcel od žiaka aj postup riešenia.

Učitelia budú môcť prezerať úlohy a riešenia žiakov či tried. Učiteľ si bude môcť dať vypísať riešenia jednotlivo, riešenia jednotlivých žiakov, všetky riešenia, všetky riešenia danej úlohy alebo riešenia zadanej triedy alebo podľa iných spoločných parametrov či kombinácií spomínaných parametrov. Rozhranie pre učiteľa bude mať teda možnosti prezerať riešenia, pričom bude viacero nastavení pre prezeranie. Okrem toho bude môcť učiteľ riešenia hodnotiť. Pri hodnotení riešení bude možné zadať známku, počet bodov a komentár, ktorý bude slúžiť napríklad na slovné hodnotenie alebo podrobnejší opis počtu bodov, ktoré žiak za úlohu dostal. Za čo konkrétne body dostal alebo naopak, za čo ich stratil. Tiež môže učiteľ využiť toto pole na ďalšie aktuálne potrebné účely. Hodnotiť bude môcť naraz iba jedno riešenie.

Študent bude mať k tomuto rozhraniu prístup tiež, avšak bude si môcť prezerať len vlastné riešenia a hodnotenia vlastných riešení. Pre študenta bude mať toto riešenie iba taký význam, aby sa mohol k svojmu riešeniu vrátiť a aby mohol po ohodnotení vidieť nakoľko bolo jeho riešenie správne.

Úlohy bude do systému možné aj pridávať, aktívna bude môcť byť však len jedna úloha. Každá úloha bude mať svoje zadanie a čas začiatku riešenia a čas dokedy bude možné úlohu riešiť. V prípade, že sa študent pokúsi odovzdať riešenie úlohy mimo tohto vymedzeného času, nepodarí sa mu to. Zadávanie úloh bude samozrejme vymoženosťou učiteľov.

Úlohami by mali byť teda dostatočné pokryté požiadavky učiteľa na žiaka. Program bude teda slúžiť aj ako sprostredkovateľ zadávania a riešenia úloh. Značne tak zjednoduší učiteľom prácu na hodnotení študentov.

3.1.3 Teória princípov

Každý princíp bude obsahovať teóriu. Táto teória bude opretá o fyzikálne zákony. Tieto zákony budú v tejto časti tiež spomínané. Teória bude obsahovať vždy nejaké vzorce, ale bude obsahovať aj ich opisy, aby bolo jednoduchšie pochopiť preberaný princíp.

Preberaný princíp bude tiež vysvetlený na príklade, ktorý by mal dostatočne vysvetliť, prečo daný úkaz, jav, či princíp funguje tak, ako ukazuje simulácia. Príklad bude reálny, no zároveň čo najjednoduchší alebo najľahšie predstaviteľný v praxi. V prípade potreby bude teória obsahovať obrázky pre lepšiu predstavu opisovanej situácie.

V teórií k danému princípu bude aj využitie tohto fyzikálneho princípu v praxi. Aby bolo študentom jasné, že sa učia niečo, čo sa v reálnom svete používa. Každý princíp človek nejako využíva a tak budú spomenuté aspoň tie najhlavnejšie možnosti využitia.

3.2 Špecifikácia fyzikálnych princípov

Fyzikálne princípy budú v programe prezentované pomocou simulácii. Tieto simulácie budú používateľom priblížené tým, že ich parametre budú môcť sami zadávať. Všetky princípy budú mať možnosti nastavenia na pravom paneli. Na pravom paneli bude tiež možné zobraziť teóriu k princípom.

3.2.1 Špecifikácia zotrvačníka

Princíp zotrvačníka bude simulovaný pomocou dvoch toční, ktoré zvierajú nejaký uhol. Presnejšie ich osi zvierajú uhol medzi 90 a 180 stupňov. Tieto osi sa dotýkajú v strede systému. Obe točne sú rovnako veľké a predpokladáme ich rovnaké zloženie a hmotnosť. Dĺžky ôs sú tiež rovnaké. Jedna z týchto točni sa točí, a podľa uhla medzi osami oboch toční sa otáča celý rotujúci systém. Používateľ teda vidí, že so zmenou uhlu, ktorý zvierajú osi točni, sa prenesie krútivý moment z jednej na druhú, a teda aj celý systém. Každá točňa však inou rýchlostnou. Rovnakú rýchlosť nadobudnú iba pri 180 stupňovom uhle. Tu tiež používateľ vidí, že točne sa točia opačným smerom. Pri 180 stupňoch má celý systém tiež maximálnu rýchlosť, zatiaľ čo pri 90-stupňovom uhle má rýchlosť nulovú. Rýchlosť celého systému okrem uhla medzi osami toční závisí tiež od rýchlosti rotácie pohyblivej točne. Preto táto rýchlosť bude, tak ako aj uhol medzi osami, nastaviteľná. Tento model bude zobrazený v troj-dimenzionálnom priestore. Osnovy stredných škôl nezahŕňajú zotrvačnosť do takej miery, preto je tento princíp nadsadenou témou či zaujímavosťou. [11]

3.2.2 Špecifikácia gyroskopu

Gyroskop je v našej práci reprezentant najzložitejšieho z rotujúcich systémov a preto sme si vybrali jeho najjednoduchšiu funkciu. Bude to teda točňa v strede, ktorá bude mať okolo seba ďalšie 2 kružnice, ktoré budú mať každá inú os a budú otáčateľné okolo svojej osi, aby bolo možné, aby nastala ľubovoľná poloha v trojdimenzionálnom priestore. Náš gyroskop bude upevnený na mieste, pričom stredná točňa sa bude točiť. Ak sa bude točiť, potom pri ťahaní myšou po ploche bude možné gyroskopom hýbať, a síce bude to znamenať zmeny uhlov kružníc okolo strednej točne. Je to preto, že funkciou gyroskopu bude ukázať, že ak sa stredná točňa točí, môžeme gyroskopom ľubovoľne otáčať, ale jej os sa nevychýli.

Na ovládacom paneli bude možné nastavovať rýchlosť otáčania strednej točne a

sprístupnenie pohybu gyroskopu. Ak bude jeho pohyb prístupný, bude ním možno hýbať a kružnice sa budú vykresľovať tak, ako by sa reálne choval gyroskop.

3.2.3 Špecifikácia Coriolisovej sily

Coriolisova sila patrí tiež medzi fyzikálne princípy, zaoberajúce sa rotujúcimi systémami. Coriolisova sila je sila, ktorá existuje v rotujúcich systémoch. Bude preto simulovaná v rotujúcom systéme. Bude to koleso predstavujúce rotujúcu vodnú hladinu. Toto koleso sa bude točiť a jeho rýchlosť bude môcť byť menená pomocou nastavení. Na tomto kolese budú umiestnené dve guličky. Jedna v strede systému a jedna na jeho obvode. Pomocou tlačidla v nastaveniach s názvom „Pusť“ sa gulička v strede uvoľní a začne padať smerom nadol. Počas pádu bude zanechávať svoju trajektóriu na točni. V prípade, že rýchlosť kolesa nebude nulová, jej trajektóriou bude krivka. Táto krivka je dôkazom pôsobenia Coriolisovej sily, lebo hoci sme guličku vypustili iba smerom nadol, okrem gravitačnej sily na ňu pôsobila ešte takzvaná „sila z boku“, ktorá spôsobila zakrivenie trajektórie guličky a táto sila je práve Coriolisova sila. Keďže táto téma je nad rámec osnov stredoškolskej fyziky [11], bude pre študentov hlavne zaujímavosťou. Keďže zohráva úlohu zaujímavosti, spestrením pre študentov bude hravosť tohto princípu a síce taká, že budú môcť guličkou zo stredu kolesa triafať guličku pohybujúcu sa po obvode kolesa. Ak ju trafia, gulička bude obiehať po obvode, inak ostane ležať pod kolesom.

3.2.4 Špecifikácia Kladky

Kladka je časťou kladkostroja. Kladkostroj je systém kladiek. Kladka slúži na zmenu smeru sily alebo jej zníženie v prípade, keď chceme touto silou vyvážiť inú. Kladkostroj je systém kladiek a bude našim modelom princípu kladky. Tento model bude obsahovať teleso, ktoré bude na pravej strane a bude mať nastavovateľnú hmotnosť. Budeme predpokladať, že toto teleso je vždy z rovnakého materiálu, a tak pri zmene jeho hmotnosti sa zmení aj jeho veľkosť. Teleso bude položené na podložke. Na ľavej strane bude motor, ktorý bude teleso ťahať zadanou silou. Motor bude pripevnený k podložke. Ak sila bude dostatočná, teleso sa bude posúvať k motoru a ak sila dostatočná nebude, teleso sa nepohne. Motor a teleso budú spojené lanom, ktoré bude natiahnuté cez 0-10 kladiek. Kladkostroj bude teda nastavovateľný a meniť sa bude dať počet jeho kladiek. Ak bude motor pôsobiť dostatočnou silou a teleso sa bude pohybovať, lano sa bude navíjať do motora a kladky sa budú točiť podľa toho, ako

rýchlo sa bude teleso posúvať k motoru. Táto rýchlosť bude záležať od sily, akou bude motor ťahať a dĺžky lana. Dĺžka lana bude záležať na vzdialenosti telesa od motora a počtu kladiek v kladkostroji.

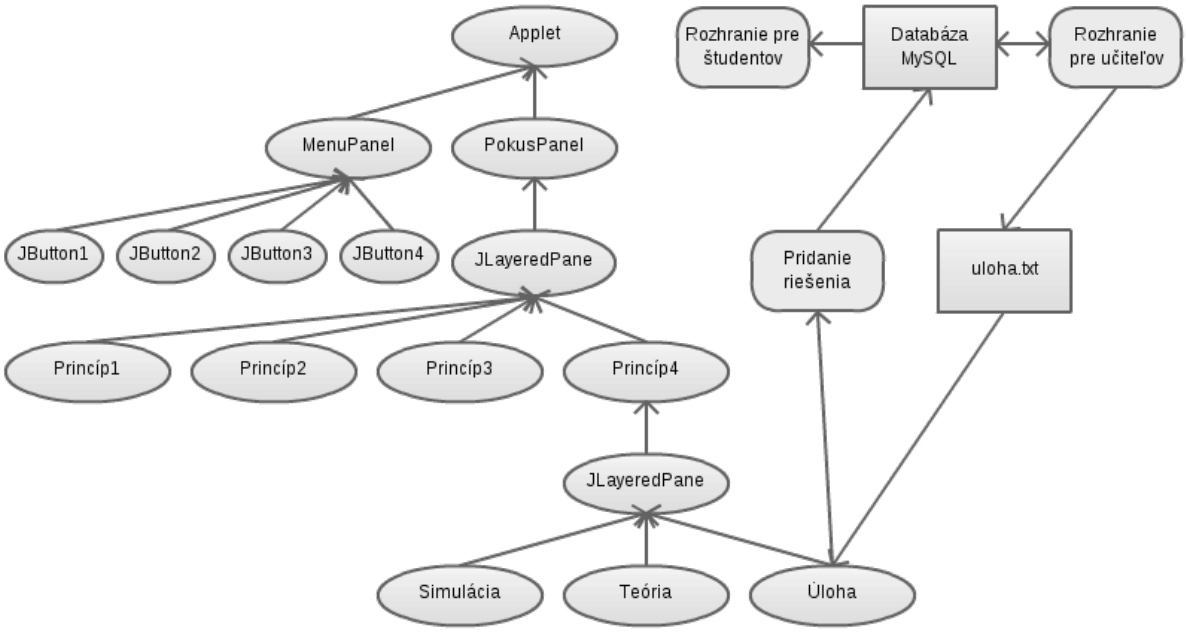
4 Návrh riešenia

Počítačové programy je dobre deliť na viaceré časti, ktoré sa delia na ďalšie časti. Delenie aplikácie na časti robí celé riešenie prehľadnejším a následne aj ľahšie rozšíriteľným. Tiež to urýchľuje vývoj, keďže jednotlivé časti je možné testovať samostatne. Riešenie nášho programu by sa dalo rozdeliť na viac častá podľa rôznych kritérií. Riešenie by sa dalo rozdeliť podľa použitých technológií či programovacích jazykov na časť vytvorenú v jazyku Java časť v jazyku PHP a časť tvorenú databázou MySQL. Rozhodli sme sa však pre rozdelenie programu na viac častí, ktoré budú do seba vnárané. Jedna časť teda bude môcť obsahovať ďalšie časti. Hlavnou časťou bude applet.

4.1 Applet

Applet bude realizovaný ako potomok triedy Japplet z Java API. [12] Applet bude rozdelený na dva panely, ktoré budú potomkami triedy Jpanel z Java API. [12] Prvý panel bude slúžiť ako menu, pomocou ktorého sa bude môcť užívateľ pohybovať medzi jednotlivými fyzikálnymi princípmi. Druhý panel bude obsahovať samotné fyzikálne princípy. Vždy bude možno vidieť iba jeden princíp, a preto bude tento panel obsahovať iba ďalší panel, ktorý bude potomkom triedy LayeredPane z balíka java.swing z Java API [12]. Pomocou tohto komponentu by malo byť splnené to, že sa bude zobrazovať vždy len jeden princíp. Každý z týchto princípov bude mať ako hlavnú triedu vlastný panel. Každý princíp bude teda panel. Tento panel bude rozdelený na dve časti. Jedna časť bude určená pre samotnú simuláciu a druhá časť bude slúžiť pre umiestnenie nastavení a možnosti simulácie. Každý princíp bude mať tiež komponent, na ktorý sa bude simulácia vykresľovať, táto bude odvodená od triedy Canvas z balíku java.awt z Java API. [12] Princípy budú mať tiež vlastné vlákno, ktoré bude potomkom triedy Thread z Java API [12] a bude slúžiť ako jednotka kontrolujúca čas a prevádzajúca výpočty týkajúce sa fyzikálnych princípov. Toto vlákno bude teda riadiacou jednotkou princípu. Poslednou častou princípov bude ich model, ktorý bude uchovávať informácie o systéme, ktorý bude objektom simulácie. Model si bude pamätať rozmiestnenie častí systému, ich rozmery a ďalšie parametre závislé od konkrétneho princípu. Každý princíp bude ako jednu z možností obsahovať možnosť zobraziť teóriu princípu.

Princípy budú obsahovať tiež zadanie úlohy a možnosti jej riešenia a odoslania. Tento komponent bude takto obsluhovať jednu funkciu z časti úloh. Touto funkciou bude pridávanie riešení do databázy. Na obrázku 5 je diagram zobrazujúci prepojenie týchto častí appletu.



Obrázok 5

4.2 Fyzikálne princípy

Fyzikálne princípy budú ako celok reprezentované ako panel appletu. Tento panel bude používať MVC architektúru, čo znamená oddelenie dátového modelu, zobrazenia a riadiacej logiky. Všetky princípy sú odvodené od triedy PokusPanel, ktorá je abstraktná. Táto trieda obsahuje metódy, ktoré budú obsahovať všetky princípy, nastavenie pôvodného stavu, sprístupnenie či zneprístupnenie nastavení.

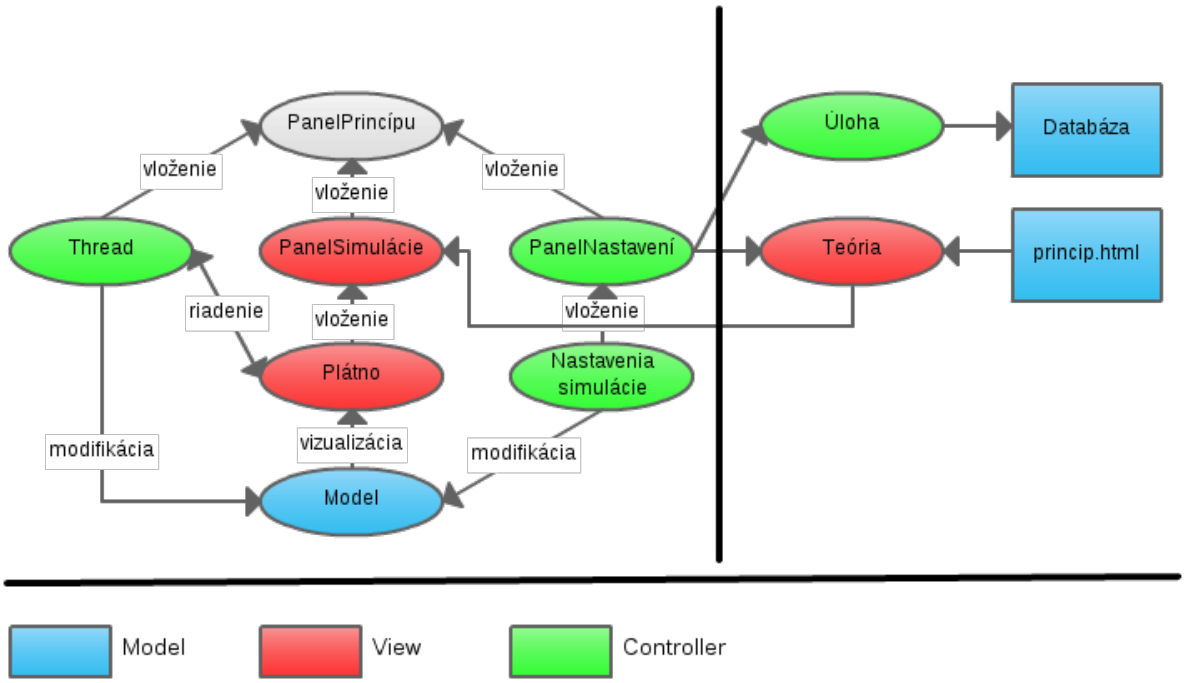
4.2.1 MVC architektúra

Model-View-Controler architektúra je architektúra, vďaka ktorej je aplikácia rozdelená na tri časti. Jedna časť zodpovedá za vzhľad, druhá za dátový model a posledná je riadiacou jednotkou. Táto architektúra sa používa preto, že takto zmena jednej časti nemá takmer žiadny vplyv na ostatné časti. [9]

V našich princípoch bude dátovým modelom model princípu. Model bude obsahovať informácie o umiestnení systému, ktorý je reprezentantom fyzikálneho princípu. Bude obsahovať tiež rozmery tohto systému a jeho časti, ďalej bude obsahovať ďalšie informácie, ktoré budú potrebné pre konkrétny princíp. View bude v princípoch implementovaný ako plátno, na ktoré sa bude vykresľovať systém, ktorý bude reprezentantom konkrétneho

fyzikálneho princípu. Toto plátno bude potomkom triedy Canvas z balíku java.awt z [12]. Na toto plátno bude systém nakreslený podľa informácií z modelu. Controler bude vo fyzikálnych princípoch reprezentovaný vláknom, ktoré bude triedou odvodenou od triedy Thread z Java API. [12] Táto trieda bude riadiacou jednotkou a tak bude mať na starosti výpočty, ktoré sú spojené s konkrétnym fyzikálnym javom. Bude tak upravovať informácie modelu. Toto vlákno bude tiež oznamovať plátnu, že sa má prekresliť.

Model bude možné meniť aj z nastavení či možností princípu, a teda časť riadiacej jednotky bude v hlavnom komponente princípov. Pomocou nastavení sa budú informácie meniť prostredníctvom udalostí. Celý model nášho MVC je na obrázku 6.



Obrázok 6

4.2.2 Model zotrvačníka

Model zotrvačníka si uchováva informácie o systéme dvoch toční, ktoré sa môžu točiť, ich osi sú spojené v strede systému a uhol medzi nimi sa môže meniť. V tomto modeli sú teda zapamätané informácie o priestore, v ktorom sa systém nachádza. O priestore potrebujeme vedieť súradnice stredu a veľkosť plátna. Systém má svoje rozmery, ktoré sú tiež v modeli uložené. Sú tam polomery platní a dĺžky osí. Platne sú rovnako veľké a ich osi sú rovnako dlhé. O platniach sa v modeli nachádzajú tiež informácie o tom, ako je práve ktorá platňa natočená a tiež aký uhol zvierajú ich osi.

Model obsahuje iba konštruktor, ktorý inicializuje premenné, ktoré obsahuje. V týchto premenných sú uložené všetky vyššie spomínané informácie. Model neobsahuje žiadne metódy. Keďže k všetkým premenným bude pristupovať plátno, ktoré bude podľa nich systém vykresľovať na obrazovku. Všetky premenné bude môcť čítať. Meniť tieto premenné bude môcť vlákno a užívateľ prostredníctvom nastavení, preto budú tieto premenné prístupné aj na čítanie. Budú teda nastavené ako verejné (public) premenné.

4.2.3 Model gyroskopu

Model gyroskopu si bude pamätať ako aj pri ostatných princípoch stred priestoru, ktorý bude aj stredom samotného gyroskopu a veľkosť tohto priestoru, od ktorého bude závislá aj veľkosť tohto gyroskopu. Gyroskop bude mať tri točne, a nás bude zaujímať veľkosť každej z nich a tiež natočenie každej z nich, čo bude uložené ako uhol v radiánoch. Ďalej si budeme v modeli uchovávať informácie o otočení gyroskopu. Otočenie gyroskopu bude v troch dimenziách, a tak to budú tri hodnoty, každá pre jednu dimenziu v priestore. Uchovávať si budeme tiež rýchlosť otáčania stredovej točne. A okrem uhlov otočení obručí okolo stredovej točne budeme potrebovať aj uchovávanie uhlov ich osí, aby bolo ľahšie tieto časti gyroskopu vykresliť správne.

Model gyroskopu, tak ako aj ostatné modely, bude uchovávať iba premenné a konštruktor, pomocou ktorého budú inicializované do pôvodného stavu. Nebude obsahovať žiadne iné metódy, pretože objekt tohto modelu bude plne prístupný pre modifikáciu jeho riadiacou jednotkou a plne prístupný pre čítanie jeho zobrazením.

4.2.4 Model princípu Coriolisovej sily

Model princípu Coliolisovej sily reprezentuje koleso, ktoré sa otáča. Na tomto kolese sa nachádzajú dve guličky. Jedna gulička na obvode kolesa a druhá v jeho strede. Gulička na obvode obieha okolo stredu po obvode. Gulička v strede sa nepohybuje do doby, kedy je užívateľom uvoľnená, aby padala smerom nadol. Model obsahuje teda informácie o prostredí, kde sa systém nachádza a to stred a veľkosť plochy, na ktorú sa kreslí. Okrem toho sú v modeli informácie o polohe kolesa, jeho veľkosti a jeho rýchlosti, o polohe a veľkosti guličky, ktorá sa nachádza v strede, o polohe guličky na obvode a o trajektórií padania guličky zo stredu. Táto trajektória bude reprezentovaná dvomi poľami, v jednom budú uchované X-ové súradnice bodov krivky, v druhom Y-ové. Okrem toho je potrebná informácia o počte bodov.

Model, rovnako ako ostatné, neobsahuje žiadne metódy s výnimkou konštruktoru, ktorý inicializuje všetky premenné podľa parametrov, ktorými sú výška a šírka plátna. Takto budú princípy fungovať, aj keď bude pri ďalšom vývoji zmenený vzhľad aplikácie v zmysle zmeny veľkosti plátna.

4.2.5 Model kladky

Model kladky obsahuje závažie, motor, podložku a kladkostroj. Tak ako pri každom modeli budeme aj v tomto modeli ukladať informácie o strede a veľkosti vykresľovacieho priestoru. Podložku budeme uchovávať ako jej súradnice. O motore budú informácie o jeho polohe a veľkosti. Závažie bude definované ako jeho rozmery, hmotnosť a X-ová súradnica. Kladkostroj bude mať model definovaný ako umiestnenie kladiek na pravej a ľavej strane, priemer kladiek, počet kladiek, ich aktuálne otočenie a určenie, kde sú upevnené. Kladky budú rozdelené na kladky pri závaží a kladky pri motore. Všetky kladky budú mať rovnaký polomer.

Trieda bude obsahovať iba konštruktor a premenné, ktoré budú verejne (public) prístupné ostatným triedam. Metódy, ktoré by sprístupňovali premenné na čítanie a zápis, sme považovali za zbytočné, pretože by nekontrolovali ich hodnoty ani nerobili žiadne výpočty.

4.2.6 View – Zobrazenie princípov

Všetky princípy budú vyžívať ako kresliacu plochu triedu odvodenú od triedy Canvas. Kresliť sa bude na objekt triedy Graphics z balíku java.awt z Java API. [12] Zobrazenie bude riešiť vykresľovanie obrázkov, ktoré budú reprezentovať časti modelov princípov. Tieto obrázky bude treba zväčšovať a zmenšovať podľa toho, aká veľká má byť časť modelu na plátne. Bude ich tiež treba otáčať či premiestňovať. Budeme tiež využívať metódy objektu Graphics. Pre transformácie budeme používať objekty typu AffineTransform z balíčku java.awt z Java API. [12] Kvôli transformáciám budeme využívať tiež triedu Graphics2D, ktorá je potomkom triedy Graphics. Na objekty typu Graphics2D sa dá kresliť aj s využitím spomínaných transformácií. Výsledný obraz budeme skladať tak, že budeme postupne nalepovať obrázky na ďalšie obrázky, medzi tým transformovať a výsledné obrázky následne nalepovať na obrázok, ktorý bude slúžiť ako buffer. Tento buffer sa na konci metódy paint(), ktorá slúži na vykresľovanie, vykreslí na naše plátno. Takýmto spôsobom budeme využívať doublebuffering, ktorý slúži na to, aby sa plátno prekresľovalo celé naraz a tak urýchlilo vykreslenie, a teda aby obrazovka neblikala.

4.2.7 Controller – ovládanie princípov

Riadiacou či logickou jednotkou princípov bude vlákno princípu odvodené od triedy Thread. Okrem toho bude časť riadenia v paneli nastavení. Riadenie bude teda rozdelené na dve časti. Vlákno bude zodpovedať za časť, ktorá sa týka fyziky, teda bude používať fyzikálne vzorce a podľa nich meniť časti modelu. Panel nastavení bude mať na starosti zmeny, ktoré bude robiť užívateľ. Teda bude môcť nastavovať model podľa možností, aké bude daný model obsahovať. Vo vlákne bude teda uchované správanie, ktoré ani v reálnom svete nemôžeme ovplyvniť a v nastaveniach budú možnosti, ktoré naopak človek má, ak by sa s princípom stretol v reálnom svete. Obe časti controller-u menia model, ale každý mení iné časti. Vlákno navyše obsahuje prepojenie s kresliacou plochou a tiež dohliada na čas. Keďže celý pokus prebieha v nejakom čase, ide skôr o čas, ktorý je medzi jednotlivými výpočtami. Prepojenie s kresliacou plochou je jednoduché. Vlákno určuje, kedy sa má plátno prekresliť.

Vlákno sa vytvorí vždy pri príchode na panel s princípom, bude však počítať podmienene. Podmienkou fyzikálnych výpočtov bude ich spustenie. Odštartovanie behu výpočtov a teda samotného princípu, bude úlohou užívateľa, v nastaveniach bude mať tlačidlo „Štart“. Po kliknutí na toto tlačidlo sa začnú vykonávať výpočty princípu a tiež sa sprístupnia možnosti ovládania princípu. Tlačidlo „Štart“ sa zmení na tlačidlo „Stop“, ktorým bude možné princíp zastaviť vrátane jeho výpočtov. Kliknutie na „Stop“ tiež zabráni užívateľovi robiť zmeny v nastaveniach. Po zastavení priebehu princípu sa tlačidlo „Stop“ zmení späť na „Štart“ a bude možné princíp znova spustiť. Spustenie princípu znova bude ale znamenať jeho pokračovanie. Tak dáme užívateľovi plnú moc nad dianím vo vykresľovacej ploche.

4.2.8 Teória princípov

Každý princíp bude mať svoju teóriu. Teória princípov je pre lepšie pochopenie samotného princípu. Aby bolo vypísanie teórie čo najjednoduchšie, bude teória uložená v .html súbore. Takto bude postarané o dôležitosť časti teórie ako vyznačenie hrubým písmom alebo kurzívou, farba písma či veľkosť písma. Tento .html súbor bude mať jednoduchú štruktúru. Na vrchu tejto stránky bude hlavný nadpis. Týmto nadpisom bude názov fyzikálneho princípu. Nasledovať bude teória prebraná zo zdrojov. Podnadpis „Príklad:“ nasledovaný samotným príkladom pre čo najľahšie pochopenie princípu. Podnadpis „Využitie“ nasledovaný príkladmi využívania princípu v praxi. Vybraté budú hlavné využitia. Celý text teórie bude stručný, aby nenudil užívateľa, no zároveň aby mu pomohol pochopiť princíp.

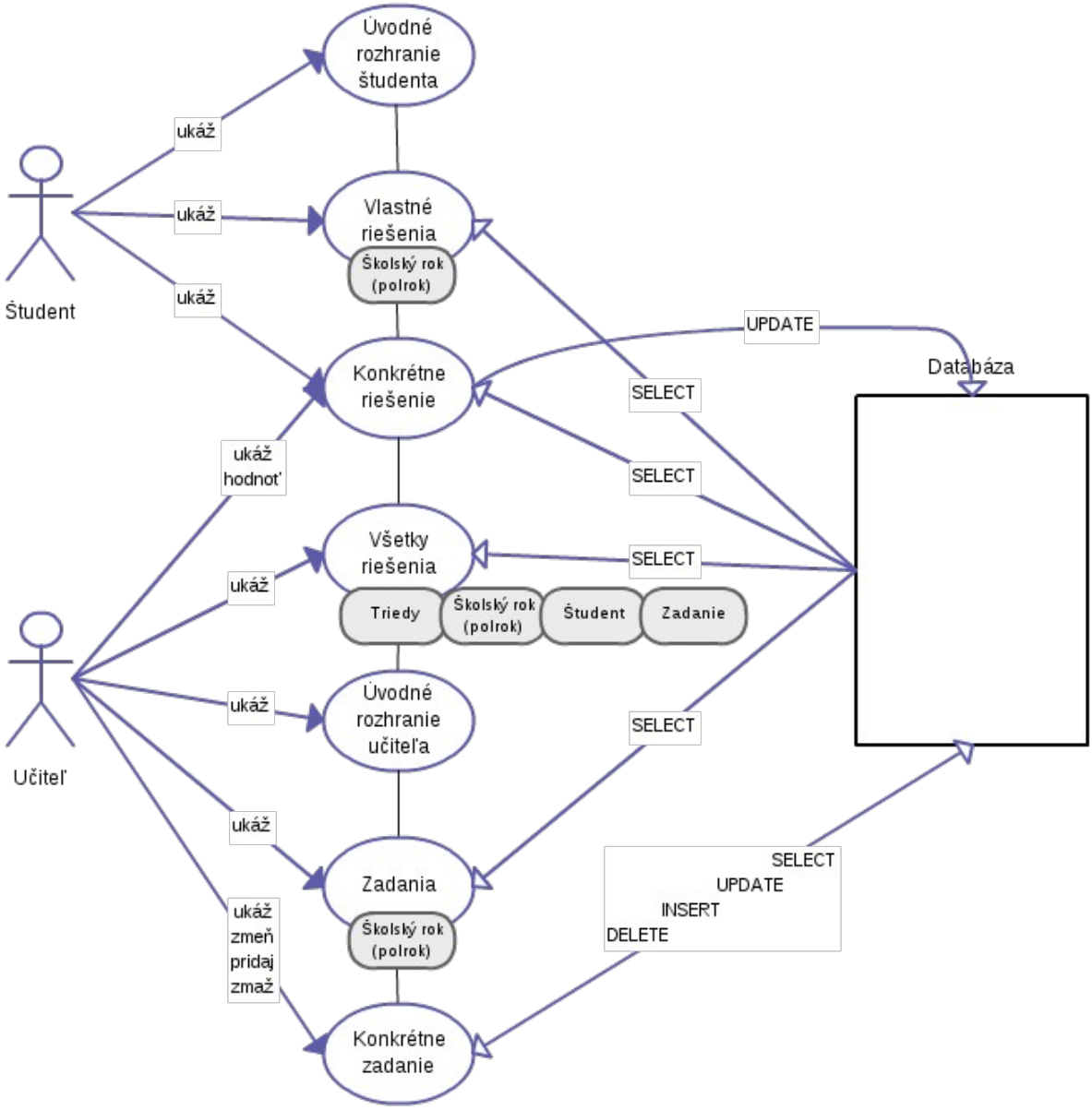
Teóriu bude možné vypísať po kliknutí na ovládací prvok s popisom „Teória“. Po kliknutí na tento prvok bude súbor zobrazený v textovom poli, ktoré nebude pre užívateľa editovateľné. Toto textové pole bude uložené v paneli pre teóriu. Panel pre teóriu sa bude zobrazovať na rovnakom mieste ako samotný princíp. Počas prezerania teórie bude stále prístupné tlačidlo pre návrat k simulácií princípu. Takto bude mať užívateľ poruke zároveň aj teóriu aj simuláciu. Výmena simulácie a teórie v paneli princípu bude riešená pomocou LayeredPane z balíčka java.swing z Java API. [12] Celý text teórie bude zobrazený skoro ako bežná webová stránka.

4.3 Úlohy

Každý princíp bude môcť obsahovať úlohu. Zadanie bude v textovom súbore, ktorý bude verejne prístupný, aby mohol byť čítaný kýmkoľvek, teda aj našim appletom. Zadania budú uložené tiež v MySQL databáze, aby sa dalo vracať k starým riešeniam. Riešenia a ich hodnotenia budú uložené v MySQL databáze.

Databáza bude mat 2 tabuľky. V prvej tabuľke budú uložené zadania pre princípy a v druhej budú uložené samotné riešenia úloh aj s hodnotením. Tabuľka so zadaniami bude mať stĺpce pre identifikačné číslo (id), názov princípu, pre ktorý bude táto úloha, čas začiatku a konca možného odovzdávania, názov úlohy, samotné zadanie, maximálny počet bodov a príznak, či je úloha aktuálne nastavená. Príznak nastavenia bude pre náš program potrebný z dôvodu, že naraz bude prístupná iba jedna úloha k jednému princípu. Druhá tabuľka bude mať stĺpce identifikačné číslo (id), názov princípu, identifikačné číslo zadania, meno, priezvisko a triedu riešiteľa, dátum a čas riešenia, riešenie, poznámky k riešeniu od študenta, hodnotenie, komentár učiteľa, ziskane body, dátum a čas hodnotenia. Tabuľky budú prepojené vďaka tomu, že tabuľka s riešeniami bude ukladať identifikačné číslo k zadaniu.

Úlohy budú teda editovateľné odosielateľné a hodnotiteľné prostredníctvom protokolu http. Takto budú úlohy priblížené užívateľovi. Bude to však vyžadovať používanie PHP skriptov. Aplikácia aj rozhranie budú iba odosielať údaje z formulárov metódou POST. Tieto údaje bude spracovávať určitý PHP skript. Pridávanie úloh či zmenu aktuálnej úlohy bude tento skript zapisovať do databázy a do súboru pre aktuálnu úlohu. V prípade ostatných požiadaviek bude skript komunikovať iba s databázou. Či už vyberať z nej údaje, mazať ich, pridávať alebo upravovať. Každý skript bude samostatný a bude spracovávať iba to, čo bude mať na starosti. Skripty na serveri budú teda komponenty nezávislé jeden na druhom a poprepájané odkazmi. Prepojenie týchto modulov ukazuje obrázok 7.



Obrazok 7

V našej aplikácií budú úlohy fungovať iba pre jeden princíp. Pre ostatné bude však funkčnosť rovnaká. Keďže bude funkčnosť úloh nezávislá na funkčnosti princípov, bude program ľahšie rozšíriteľný a modifikovateľný, nakoľko modifikácia úloh či ich princípu neovplyvní samotné princípy.

4.3.1 Odosielanie riešení

Úlohy budú v applete zobrazené ako zadanie s formulárom pre vyplnenie údajov o riešiteľovi a na vyplnenie samotného riešenia. Údaje o riešiteľovi budú meno, priezvisko a trieda, do ktorej študent patrí. Riešenie bude obsahovať samotný výsledok a poznámky k

riešeniu. Na konci formulára bude odosielacie tlačidlo. Samotné odosielanie riešenia bude rozdelené na dve časti, jedna bude odosielať riešenie z aplikácie na server a druhá bude na serveri, kde bude spracovávať prijaté dáta a obsluhovať samotné pridanie do databázy.

Odosielací mechanizmus bude fungovať tak, že po stlačení tlačidla odoslať sa najprv skontrolujú vyplnené údaje vo formulári. V prípade, že údaje nebudú vyplnené správne riešenie sa neodošle. V prípade správneho vyplnenia údajov sa vytvorí požiadavka na server, ktorá tam odošle metódou POST informácie z formulára. Táto požiadavka bude doplnená a niektoré údaje, ktoré študent nemôže vyplniť vo formulári ako napríklad identifikačné číslo úlohy. Tieto informácie sa budú odosielať na štandardný port 80 pre protokol http.

Serverová stránka odosielacieho mechanizmu bude skript, ktorý bude slúžiť na spracovávanie údajov, ktoré dostane z appletovej stránky odosielania. Skontroluje, či je možné v danom čase úlohu odovzdať a v prípade, že nie, skončí svoju činnosť. V prípade, že je možné úlohu v konkrétnom čase odovzdať, doplní prijaté údaje o ďalšie, a síce dátum a čas odoslania riešenia. Následne sa vytvorí dopyt na databázu a prostredníctvom neho sa údaje zapíšu do databázy.

4.3.2 Prehliadanie riešení učiteľom a študentom

Prehliadanie riešení bude rôzne pri prehliadaní študentom a učiteľom. Študent bude mať obmedzené možnosti prehliadania. Študent bude môcť prezerať iba svoje riešenia. Bude si však môcť vybrať medzi zobrazením všetkých riešení alebo riešení len v konkrétnom polroku ako zoznam riešení. Zoznam riešení bude v tabuľke a názvy zadaní budú fungovať ako odkazy na detaily konkrétneho riešenia úlohy. Po kliknutí na tento odkaz sa zobrazí vybrané riešenie úlohy, ktoré bude zobrazené aj s hodnotením. Študent si toto riešenie bude môcť iba prehliadať, nebude ho môcť meniť ani zmazať. Celé rozhranie bude teda fungovať na princípe webovej stránky, pričom zobrazované dáta bude vyberať z databázy na to určený PHP skript. Ten dostane parametre výpisu metódou GET alebo POST.

Možnosti učiteľov budú omnoho bohatšie. Učiteľ si bude môcť zobraziť zoznamy riešení ako všetky riešenia konkrétneho žiaka alebo jeho riešenia v danom polroku. Tiež budú môcť zobraziť riešenia všetkých žiakov z danej triedy a daného polroka alebo všetky riešenia konkrétnej úlohy. Bude možné zobraziť aj zoznam všetkých riešení. Zoznam riešení bude aj v tomto prípade zoznamom odkazov na detaily riešenia. Učiteľ však riešenia nebude môcť iba prehliadať, bude ich tiež môcť hodnotiť, čo sa na serverovej stránke teda v PHP skripte bude javiť ako úprava či aktualizácia dát v databáze. Hodnotenie však bude znamenať iba pridanie

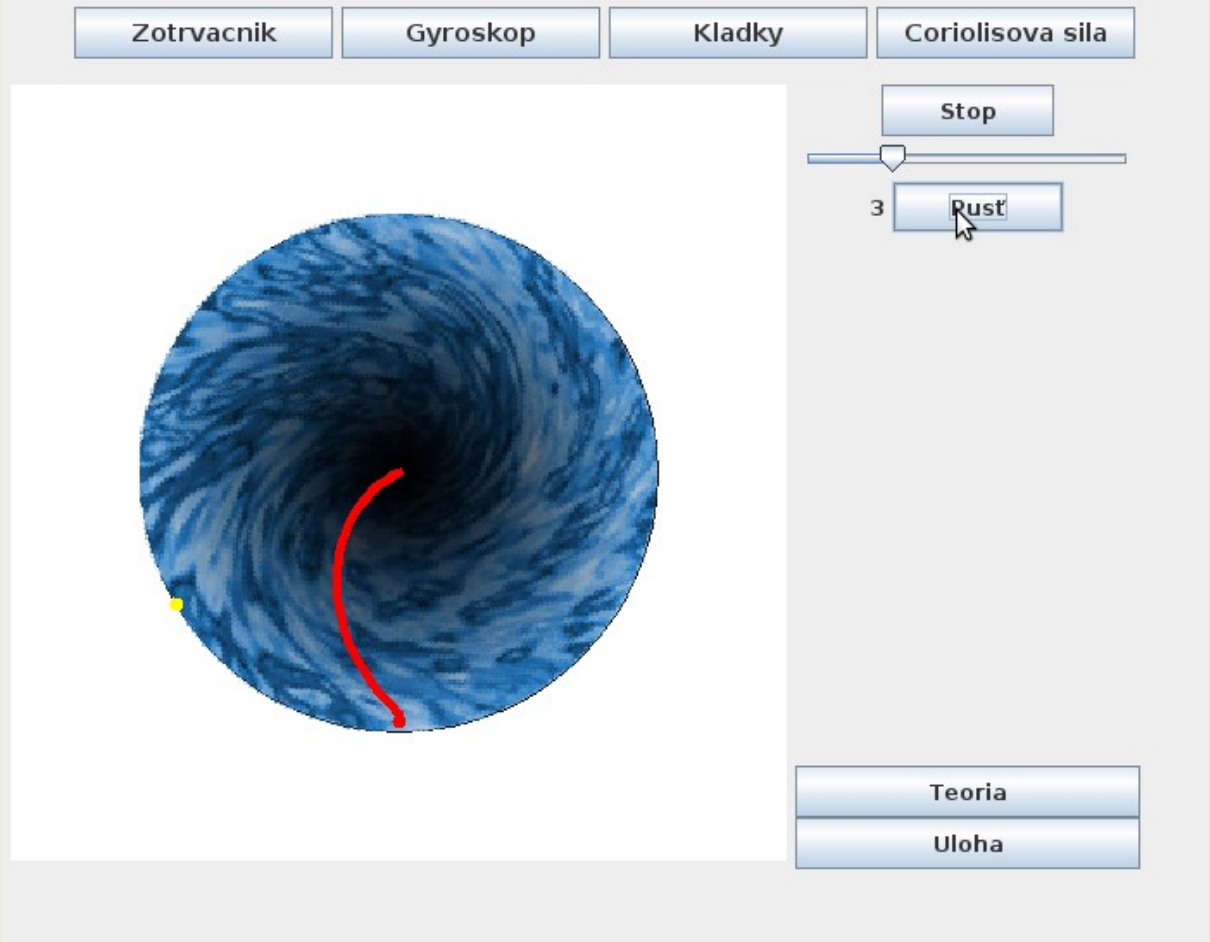
hodnotenia, učiteľ nebude môcť meniť riešenia študentov. Učiteľ nebude môcť riešenia mazať.

Učitelia budú mať tiež prístup k samotným zadaniam. Tieto budú môcť pridávať, upravovať či mazať. Samozrejmosťou bude vypísanie zoznamu všetkých úloh či vypísanie všetkých úloh prislúchajúcich k danému princípu. Tento zoznam bude, tak ako aj zoznam riešení, zoznamom odkazov na detaily prvku, v tomto prípade zadania. Úprava či pridanie zadania bude zložitejšie ako vypísanie detailov. V prípade úpravy či pridania zadania bude skript pred samotným dopytom na databázu kontrolovať správnosť vyplnených údajov a údaje zapisovať do databázy iba v prípade, že údaje boli vyplnené správne. Učitelia budú môcť zadania tiež mazať. Zmazanie zadania však bude znamenať, že riešenia k tomuto zadaniu už nebudú patriť žiadnemu zadaniu. Tieto riešenia budú teda bezpredmetné a následne zmazané. Tento spôsob mazania ma svoje výhody aj nevýhody. Nevýhodou bude, že študenti po ich zmazaní nebudú viac môcť prehliadať tieto riešenia a tak ich študenti nebudú môcť prezerať, ak by sa s ich pomocou chceli neskôr niečo naučiť alebo si niečo zopakovať. Nepredpokladáme však, že učitelia budú mazať zadania počas aktuálneho roka a tak by študentom riešenia prakticky chýbať nemali. Tento spôsob bude šetrením úložného miesta, databáza nebude musieť byť príliž rozsiahla. Čo sa týka učiteľov, pred novým rokom môžu takto zmazať zadanie, čím zmažú automaticky aj všetky riešenia tohto zadania. Následné pridanie tohto zadania nanovo sprehľadní neskoršie riešenia v prípade, že by učitelia chceli vidieť všetky riešenia.

5 Implementácia

Implementácia zahŕňa všetky komponenty nášho programu. Implementáciu sme začali samotným appletom, nasledovala tvorba hlavného rozhrania appletu. Potom sme implementovali základnú triedu fyzikálnych princípov a následne jednotlivé princípy a ich časti. Fyzikálne princípy sme implementovali ako celky, ktoré boli zložené zo samotného princípu ako simulácie. Simulácia mala tri časti a to vlákno, model a plátno, na ktoré sa model vykresľuje. Druhou častou princípu je teória a poslednou úloha. Keďže vždy môže byť zobrazený iba jeden princíp, a v prípade konkrétneho princípu vždy iba jedna časť – simulácia, teória alebo úloha, boli implementované nezávisle. Keďže úlohy sú komponentom čiastočne oddelené od celého appletu, riešené boli ako druhá časť programu. Z úloh bolo pridané najprv pridávanie zadaní, potom prehliadanie riešení a nakoniec hodnotenie riešení.

Rozhranie pre prehliadanie a hodnotenie riešení bolo implementované ako webová stránka, takže jej vzhľad bol riešený prostredníctvom html a css. Grafická stránka celého programu však nebola našou prioritou a tak sme vybrali jednoduchý vzhľad či už appletu, fyzikálnych princípov či rozhrania pre prehliadanie riešení. Vzhľad appletu vidno na obrázku 8.



Obrázok 8

Program sme implmentovali tak, aby mohol byť ľahko rozšíriteľný. Aby jeho vývoj mohol pokračovať pridávaním ďalších princípov. Nové princípy navyše nemusia byť implementované rovnako ako tie, ktoré sme vytvorili my. Ich návrh však môže slúžiť ako šablóna. Systém úloh je oddelený od appletu a tak sú jeho možnosti tiež ľahko modifikovateľné či ďalej vyvíjané bez toho, aby bolo treba robiť zmeny v applete, poprípade aby to boli len jednoduché a malé zmeny. Systém úloh sme pridali len k jednému princípu, no tým istým spôsobom je možné pridať ho k ostatným princípom či k ďalším princípom pridaným v budúcnosti. Negatívnou stránkou však je, že sa princípy nedajú do appletu pridávať dynamicky. Ba dokonca, pridanie princípov si vyžaduje pridanie viacerých komponentov do appletu. Avšak v prípade budúceho vývoja bude krok k jednoduchému, či dokonca dynamickému pridávaniu hlavnou prioritou.

5.1 Problémy pri implmentácií

Počas implementácie sme sa stretli s viacerými problémami. Jednotlivé problémy budú opísané nižšie. Vybrané budú len komponenty, ktoré nám spôsobili najviac problémov alebo nás najviac zdržali pri vývoji programu. Tieto problémy ale aj celé komponenty, v ktorých daný problém nastal, budeme opisovať podrobne. Považovali sme ich totiž za zaujímavé a dostatočne dôležité, aby sme im venovali viac pozornosti ako ostatným.

5.1.1 Vymieňanie princípov

Vymieňanie princípov v paneli pre aktuálny princíp sme pôvodne navrhli ako pridávanie a odoberanie panelov do panelu pre princípy. Tento spôsob však nefungoval. Štandardné grafické užívateľské prostredie jazyka Java je postavené tak, aby sa dali do kontajnerov pridávať a z nich odoberať ďalšie komponenty. Avšak pridávanie, či odoberanie je možné len počas inicializácie pre vyskladanie celého rozhrania. Dynamicky sa toto rozhranie meniť nedá. Našťastie tieto nedostatky štandardná knižnica java.swing rieši tak, že má komponent LayeredPane z Java API [12], ktorý môže obsahovať viac komponentov, no iba jeden je zobrazený na obrazovke. Tento komponent robí dojem, že sú komponenty v ňom nastavané jeden pred druhý a inými komponentmi určujeme, ktorý je práve navrchu a „zatieňuje“ ostatné.

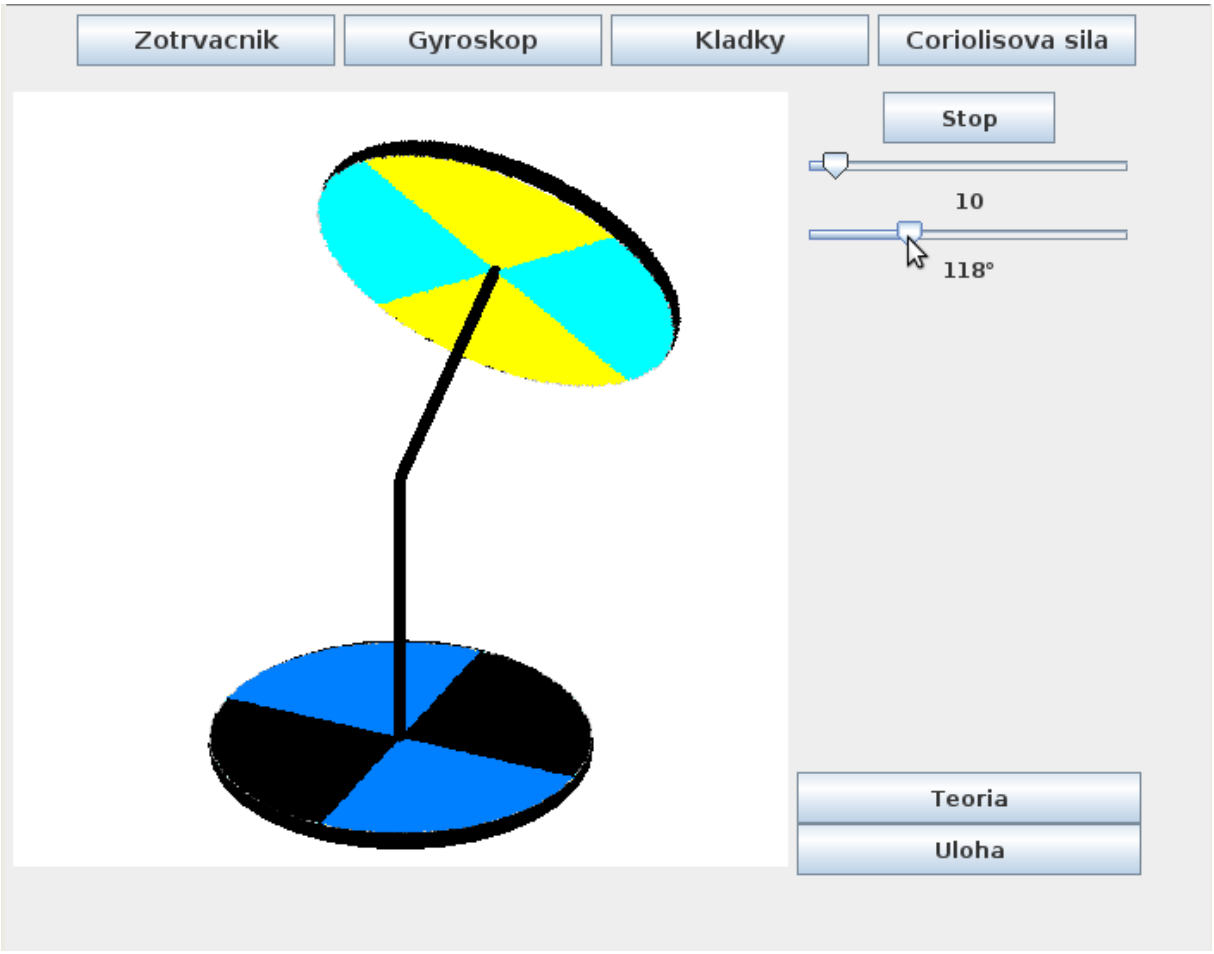
5.1.2 Vykreslenie zotrvačníka

Zotrvačník sme kreslili ako dve časti, dynamickú a statickú točňu. Statická točňa nemení polohu, iba sa otáča okolo svojej osi, takže implementácia bola jednoduchá, vykreslili sme náš obrázok na premennú typu BufferedImage. Tejto premennej sme pridelili objekt AffineTransform, pomocou ktorého sme otočili obrázok o potrebný uhol. Tak isto sme pripravili aj dynamickú točňu.

Dynamická točňa sa však môže v priestore dostať do mnohých polôh, ktoré vytvárajú pologuľu. Na celý systém sa užívateľ pozerá pod 30 stupňovým uhlom. Je to preto, aby mohol lepšie pozorovať vlastnosti tohto princípu, poprípade jeho priebeh. Rozsah pohybu dynamickej točne sme rozdelili najprv na dve časti. Prednú a zadnú polovicu pologule. Točňa sa kreslila tak, že sa najprv otočila o uhol aký bolo treba a potom sa vykreslila na pomocnú premennú typu BufferedImage, kde bola k nej prikreslená aj jej os. Následne sme obe časti otáčali o uhol, aký vidí užívateľ medzi oboma osami. Tiež bolo treba vypočítať dĺžku osi. V skutočnosti sa dĺžka osi nemení, ale pri pohľade na systém sa os javí v niektorých uhloch kratšia alebo ju dokonca vôbec nie je vidno. Tak isto sa ale javí aj celá točňa. V jednom bode sa javí ako kruh a v ostatných sa javí sploštená. V niektorých miestach je vidno dokonca len jej hranu. Táto časť bola rozdelená podľa pozície sprava doľava a z vrchného bodu po spodný. Viditeľné hrúbky v rôznych horizontálnych i vertikálnych uhloch sú závislé na funkcií sínus, presnejšie jej častiach. Tak sme vypočítali, ako sa má pred vykreslením na premennú image_buffer slúžiacu ako buffer transformovať (škálovať).

Druhú polovicu pologule sme však museli rozdeliť znova na dve časti, pretože na tejto polovici sa vyskytuje na mnohých bodoch jav, kde je vidieť iba hranu točne. Táto poloha je však pri rôznych uhloch medzi osami toční rôzna. Je to krivka, ktorá je znova časťou funkcie sínus. Bod, v ktorom je vidieť z točne minimum je teda pre rôzny uhol medzi osami toční rôzny a tak sme rozdelili tento problém na dve časti a to na oblasť pod touto „nulovou“ krivkou a nad ňou. Navyše, ak je točňa pod touto krivkou, oko vidí točňu z vnútornej strany, zatiaľ čo ak je nad touto krivkou, vidno vonkajšiu stranu točne. Keďže ohraničenie týchto častí je krivka, pribudlo viac výpočtov. Nakoľko väčšina týchto výpočtov bola závislá na funkcii sínus alebo kosínus, bolo nutné používať premenné typu double, ktorých presnosť je však obmedzená. Počas výpočtov sa muselo viackrát zaokrúhľovať a tak vo výsledku nastávajú miesta, kde sa na krátky časový úsek zobrazuje pohybujúca točňa zle. Keďže toto zobrazovanie je spôsobené numerickými výpočtami a zaokrúhľovaním, je naozaj problém opraviť to tak, aby fungovala simulácia stopercentne správne. Vzhľadom k tomu, že tieto nedostatky sa prejavujú len chvíľkovo, nevenovali sme im viac času v záujme funkčnosti celej

práce. Výsledný zotrvačník je na obrázku 9.



Obrázok 9

5.2 Rozhranie pre úlohy

Rozhranie pre úlohy sme rozdelili na dve časti a síce učiteľskú a študentskú časť, pričom každý užívateľ by mal mať prístup len do tej časti, do ktorej skupiny užívateľov patrí. V našej práci sa však dá medzi týmito prostrediami pohybovať ľubovoľne. Nie je žiadne prihlasovanie ani nič podobné, čo by odlišovalo študenta od učiteľa. Je to tak preto, že to nebolo zadaním našej práce. Program je však napísaný tak, že jeho php moduly pre študenta a pre učiteľa v reálnej prevádzke stačí vložiť na stránku, kde je rozlíšený prístup učiteľov od prístupu študentov. Naša práca teda obsahuje aj jeden súbor, ktorý je len zástupcom skutočnej stránky, na ktorej by sa mal program používať.

Toto rozhranie má jednoduchý dizajn. Ako bolo spomínané vyššie, malo by byť vložené do webovej stránky a tak aj dizajn rozhrania prenechávame na budúcu domovskú stránku. Teda vzhľad tohto rozhrania je plne modifikovateľný webovou stránkou, ktorá

program a teda aj toto rozhranie používa.

Toto rozhranie sme implementovali ako súbor modulov. O tom, ktorý modul sa vyberie pre zobrazenie, rozhoduje hlavný modul, ktorý zobrazuje aj samotný applet. Hlavný modul vyberá ostatné moduly podla informácií, ktoré obdrží http metódou POST alebo GET. Pri načítaní domácej stránky, teda bez parametrov poslaných metódami POST alebo GET sa zobrazí modul, ktorý dáva možnosti vybrať čo chce užívateľ zobraziť. V prípade študenta je to iný modul ako v prípade učiteľa. Preto sú aj hlavné moduly dva, líšia sa však iba v pôvodnom module. Každý modul totiž má možnosti prechodu iba do časti aplikácie, na ktoré má právo aj jeho odosielateľ. Teda zo študentského módu nemôže prejsť do učiteľskej časti bez zmeny hlavného modulu, čo v ostrej prevádzke zariadi stránka obsahujúca program.

5.3 Použitie programu

Celý program sme implementovali tak, aby bol použiteľný nie len na našej stránke, ale na rôznych stránkach. Znamená to, že program môže byť vložený na ľubovolnú stránku, pričom samotné vloženie je veľmi jednoduché. Naša stránka, na ktorej je program uložený, je len príklad použitia. Nepoužíva žiaden zvláštny dizajn a nerieši ani žiadne prihlasovanie, ktoré sa od stránok využívajúcich tento program očakáva. Prihlasovaním by sa dal oddeliť učiteľský mód prehliadania úloh od študentského. Na našej stránke je možné prepínať sa medzi týmito dvoma možnosťami v menu, ktoré je na vrchu stránky vytvorené len pre tento účel. Keďže sa program delí na dva hlavné moduly, ktoré oba obsahujú apllet, ale každý k nemu ukazuje iné rozhranie pre prezeranie úloh, vkladanie programu na stránku je triviálne. Samotné pridanie programu na stránku teda znamená pridanie dvoch riadkov PHP kódu, a síce jeden pre vloženie učiteľského rozhrania a jeden pre vloženie študentského rozhrania. Kódy, ktoré na stránku treba vložiť vyzerajú takto:

```
include studentske_rozhranie_fyz.php
include ucitelske_rozhranie_fyz.php
```

Toto robí náš program prenositeľným. Jediné čo stránka, na ktorú program pridávame, potrebuje, je podpora pre PHP.

Záver

Zadaním práce bolo vytvoriť výukový program pre vyučovanie stredoškolskej fyziky. Tento program mal obsahovať 4 fyzikálne princípy. Program mal byť rozširovatelný a nezávislý. Rozširovatelný v zmysle, že sa očakáva, že nebude v budúcnosti obsahovať iba 4 princípy, ale ich počet bude ľubovolný a tiež, že bude implementovaný tak, aby mali tvorcovia ďalších častí prístupné akési šablóny, podľa ktorých môžu ľahko a rýchlo vyvíjať nové časti programu. Program mal obsahovať časť, ktorá mala dať užívateľom možnosť riešiť úlohy pre otestovanie získaných vedomostí pomocou programu. Program mal obsahovať princípy tiež dostatočne názorne na to, aby užívateľ princíp, ktorý program simuluje pochopil.

Všetky ciele bohužiaľ neboli úplne splnené. Náš program obsahuje iba 3 princípy. Každý z nich je však implementovaný tak, aby užívateľ princíp pochopil. Docielené to bolo tým, že simuláciu dopĺňa priložená teória, a zároveň túto teóriu dopĺňa samotná simulácia. Všetky princípy používajú rovnaký návrhový vzor. Používajú architektúru MVC. Architektúra MVC sa používa v mnohých systémoch a frameworkoch, pretože veľmi zjednodušuje tvorbu rôznych systémov a zároveň ich rozdeľuje na časti, ktoré síce spolupracujú, ale každá časť rieši iný aspekt systému. Takto sme vytvorili dobrý základ pre vývoj ďalších princípov pre náš program. Ďalšou nevýhodou nášho programu je, že nové princípy nie je možné pridávať dynamicky. Pridanie nového princípu by si vyžadovalo úpravy v zdrojovom kóde appletu. Nebolo by to však veľa zmien.

Program je naozaj nezávislý. Je určený na používanie cez webový prehliadač, a tak sa spusti pri načítaní stránky. Že je nezávislý znamená aj to, že sa neviaže na našu stránku, na ktorej je uložený. Celý program je možno vložiť na ľubovolnú stránku. Vloženie programu na webovú stránku je maximálne jednoduché. Znamená totiž iba skopírovanie programu a pridanie dvoch konkrétnych riadkov kódu v jazyku PHP. Okrem toho je prenositeľný na ľubovolný server s podporou jazyka PHP. Náš program je pre účel tejto práce a pre možnosť jeho možnosti vyskúšať prípadnými záujemcami jeho používanie, zverejnený na webovej stránke tejto bakalárskej prace. [13]

Rozhranie pre riešenie úloh bolo implementované v celom rozsahu cieľa. Znamená to, že boli implementované všetky úkony, ktoré študenti a učitelia budú potrebovať pre výučbu. Teda, že učitelia môžu pridávať úlohy, študenti tieto úlohy riešiť a učitelia potom riešenia hodnotiť. Úlohy, ich zadávanie a riešenie bolo ale dopracované do takého štádia, kedy je možné úlohy pridávať pre ľubovoľný princíp, ktorý applet obsahuje a ktorý bude obsahovať v budúcnosti. Úlohy sú totiž implementované tak, aby fungovali všeobecne pre každý princíp rovnako. V tomto smere sme teda ciele naplnili úplne.

Celý program teda neobsahuje všetko, čo sme plánovali v začiatkoch práce. Avšak chýbajú iba časti, ktoré nie sú potrebné pre samotný beh programu a ktoré boli plánované kvôli väčšiemu rozsahu práce. Boli to práve časové dôvody, ktoré nám nedovolili tieto časti implementovať. Napriek tomu sme však vytvorili program, ktorý je funkčný a pripravený na ďalší vývoj v zmysle rozširovania záberu princípov, ktoré môže študentom predviesť, oboznámiť ich s nimi, motivovať ich a v neposlednom rade niečo naučiť.

Použitá literatúra

[1] Feynman R.,Leighton R.,Sands M.: Feynmanovy přednášky z fyziky s řešenými příklady 1/3, Praha, 2000, ISBN: 8072004050

[2] Gyroscopes - Everything you needed to know (23.5.2013) http://www.gyroscopes.org/behaviour.asp

[3] The Coriolis Effect: A (Fairly) Simple Explanation (23.5.2013) http://stratus.ssec.wisc.edu/courses/gg101/coriolis/coriolis.html

[4] Halliday D., Resnick R., Walker J.: Fyzika, Vysoké učení technické v Brně - Nakladatelství VUTIUM a PROMETHEUS Praha, 2007, ISBN: 8021418680

[5] TIOBE Software: Tiobe Index (15.1.2013) http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

[6] Pavel Herout.: Učebnice jazyka Java, České Budějovice, 2011, ISBN: 9788072323982

[7] About the Eclipse Foundation (23.5.2013) http://www.eclipse.org/org/

[8] PHP: Hypertext Preprocessor (23.5.2013) http://www.php.net/

[9] (ootips) Model-View-Controller (23.5.2013) http://ootips.org/mvc-pattern.html

[10] Lukáš Slovák ; školiteľ Pavel Petrovič.: Výukový program demonštrujúci fyzikálny princíp (Bakalárska práca), Bratislava, 2011, (online 23.5.2013) http://dai.fmph.uniba.sk/~petrovic/th11/Slovak.pdf

[11] Učebnice fyziky pre 1., 2., 3. a 4. ročník gymnázia

[12] Java™ Platform, Standard Edition 6, API Specification (23.5.2013)
http://docs.oracle.com/javase/6/docs/api/

[13] Jozef Belko ; školiteľ Pavel Petrovič.: Výukový program demonštrujúci fyzikálny princíp (Bakalárska práca) – Aplikácia (23.5.2013)
http://virtuallab.kar.elf.stuba.sk/~belko/program/

Prílohy

1. Zdrojový kód aplikácie

Kompletný zdrojový kód programu sa nachádza na priloženom CD.