**UNIVERZITA KOMENSKÉHO V BRATISLAVE**

**FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY**

# Výukový program demonštrujúci fyzikálny princíp

**2011** **Lukáš Slovák**

**UNIVERZITA KOMENSKÉHO V BRATISLAVE**

**FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY**

# Výukový program demonštrujúci fyzikálny princíp

## Bakalárska práca

Evidenčné číslo: bbf59003-a74e-4173-8377-18669603fb27

Študijný program: Aplikovaná informatika

Študijný odbor: 9.2.9. aplikovaná informatika

Školiace pracovisko: Katedra aplikovanej informatiky

Školiteľ: Mgr. Pavel Petrovič, PhD.

**Bratislava 2011** **Lukáš Slovák**

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

# ZADANIE ZÁVEREČNEJ PRÁCE

**Meno a priezvisko študenta:** Lukáš Slovák
**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 - naprogramovať 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, rezistancia, induktancia, RC, LC obvody a podobné rezonátory...), ideálne skombinované aj s jednosmernými obvodmi - čiže kirchhofove zákony, ohmov zákon... alebo modely atómu, teória elementárnych častíc, kvantová teória... alebo momenty rotačného pohybu, momenty zotrvačnosti, vzťažné sústavy a pod.

**Vedúci:** Mgr. Pavel Petrovič, PhD.

**Spôsob sprístupnenia elektronickej verzie práce:**
bez obmedzenia

**Dátum zadania:** 02.10.2010

**Dátum schválenia:** 25.10.2010

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

Slovák
.....
študent

Pavel Petrovič
.....
vedúci

Čestne prehlasujem, že som túto bakalársku prácu

vypracoval samostatne s použitím citovaných zdrojov.

.....

# Poďakovanie

Týmto by som sa chcel poďakovať vedúcemu svojej bakalárskej práce Mgr. Pavlovi Petrovičovi, PhD. za pomoc, trpezlivosť a inšpiráciu pri tvorbe mojej práce, doc. RNDr. Vladimírovi Černému, PhD. za pomoc s fyzikálnou stránkou problému a všetkým ostatným, ktorý mi akokoľvek pomohli pri vypracovaní mojej bakalárskej práce.

# Abstrakt

Cieľom mojej bakalárskej práce bolo naprogramovať výukový program, ktorý pomocou simulácie demonštruje fyzikálny princíp. Konkrétne ide o vlnovo-časticový dualizmus a objav jadra atómu. Program mal byť interaktívny, jednoducho použiteľný, určený primárne pre stredoškolákov. V práci najskôr rozoberám východiská, ktoré sa mi stali podkladmi pre ďalšiu prácu, ďalej uvádzam špecifikáciu programu, návrh programu vychádzajúci zo špecifikácie a implementáciu programu. Na záver spomeniem spôsob použitia programu a popisujem jeho testovanie žiakmi gymnázia.

Kľúčové slová: simulácia, komponent, synchronizácia, sieťová komunikácia

# Abstract

The goal of my bachelor thesis was to program education software that uses simulation to demonstrate a physical principle. Specifically, it is a wave-particle duality and discovery of atomic nucleus. A program should be interactive, easy to use and primarily dedicated to high school students. Firstly, I discuss background knowledge that was a basis for another work. Afterwards I introduce the specification of the program, design based on the specification, and finally implementation of the program. At the end, I am going to mention how to use the program and I am going to describe the testing of the program by high school students.

Key words: simulation, component, synchronization, network communication

# Obsah

# 1 Úvod

Informačné technológie sú v dnešnej dobe neoddeliteľnou súčasťou nášho života. Zasahujú takmer do všetkých oblastí nášho života.

Vzdelávanie nie je výnimkou. Pedagógovia využívajú počítače pri tvorbe prezentácií, môžu pomocou *mailing listov* posielať nové informácie študentom o kurzoch, či zasielať im hodnotenia. Dokonca celé kurzy môžu prebiehať v rámci špecializovaných informačných systémov. Mohli by sme povedať, že informačné technológie sa dajú využiť vo výučbe akéhokoľvek predmetu. Fyzika nie je výnimkou. Pri výučbe fyziky sa často používajú aplikácie, ktoré znázorňujú nejaký fyzikálny jav či experiment. Študenti môžu vidieť ako jav prebieha a nie sú odkázaní iba na textový popis v učebniciach. Tým program môže prispieť k lepšiemu pochopeniu javu. Práve tvorbe takejto aplikácie sa venuje naša bakalárska práca.

Cieľom našej bakalárskej práce bolo vytvoriť výukový program, ktorý pomocou simulácie vysvetľuje fyzikálny princíp. Zvolenou fyzikálnou oblasťou bola fyzika mikrosveta, konkrétne vlnovo-častícový dualizmus a objav jadra atómu. Táto oblasť je veľmi zaujímavá a ponúka pohľad na to, z čoho sa vlastne všetko hmotné skladá. Práve preto sme si ju vybrali.

Existuje veľmi veľa aplikácií, ktoré vysvetľujú, či zobrazujú najrôznejšie fyzikálne javy, i tie, ktoré sme si vybrali pre svoju bakalársku prácu. Niektoré z nich sme si našli na internete. Ich analýzou sme získali predstavu o tom, ako by asi mohla vyzerať naša aplikácia, ako robiť animácie konkrétnych javov či experimentov , a že vhodným doplnením animácií by mohol byť vysvetľujúci text k nim. Nakoniec sme sa rozhodli do aplikácie pridať aj komunikáciu medzi jednotlivými programami bežiacimi na dvoch rôznych počítačoch v lokálnej sieti, aby mohol program použiť aj učiteľ na vyučovanie študentov.

V nasledujúcich kapitolách sme popisovali východiská pre tvorbu našej aplikácie. Konkrétne informatické, fyzikálne a pedagogické východiská. Ďalej sme sa zaoberali podrobnou špecifikáciou programu. Potom sme rozoberali jeho návrh. Pokračovali sme opisom problémov, na ktoré narazili pri implementácii, a opisom spôsobu riešenia

komunikácie po sieti. Nakoniec sme popísali ako program používať a jeho odskúšanie žiakmi gymnázia. V závere zhrnieme výsledky práce a navrhneme, ako by sa dalo ďalej pokračovať v práci na programe.

# 2 Východiská

V tejto kapitole sme popisovali didaktické, fyzikálne a informatické poznatky, ktoré sme pri tvorbe nášho programu využívali.

## 2.1 Tvorba didaktického softvéru

Pri tvorbe didaktického softvéru je dôležitá cieľová skupina študentov, ktorým je daný program určený. Pri našom programe ide o stredoškolákov, ktorý majú záujem dozvedieť sa niečo viac o fyzike mikrosveta, kvantovaní energie a vlnových vlastnostiach elementárnych častíc. Zároveň to môže byť pre nich lákadlo k ďalšiemu štúdiu v tejto oblasti. Treba si uvedomiť, že ide o takmer dospelých ľudí, a teda nie je potrebné snažiť sa podať im látku hravou formou. Vysvetľovanie tém je realizované pomocou simulácií fyzikálnych experimentov a jednoduchej vizualizácie istého fyzikálneho poznatku.

Pri simuláciách treba klásť dôraz na to, aby sa v nich používali reálne hodnoty. Ak to experiment umožňuje, aby sa dali nastaviť vlastné experimentálne hodnoty. Takisto je dôležitá vizualizácia simulácie. Ak je to možné a nápomocné pri vysvetľovaní, je vhodné použiť krokovanie pri vizualizácii. Vizualizácia by takisto mohla ponúkať pohľad takpovediac dovnútra daného javu. Môže sa teda využívať „zoomovanie“.

## 2.2 Fyzika mikrosveta

### 2.2.1 Výber experimentov

Fyzika mikrosveta je stále živou a neustále sa meniacou oblasťou fyziky. Zákonitosti, ktoré platia v tomto svete sú podstatne odlišné od tých, s ktorými sa stretávame v každodennom živote. Správanie mikročastíc sa často nedá jednoducho opísať, nieto ešte znázorniť. Preto samotný výber poznatkov, ktoré sme chceli vizualizovať v našom programe, nebola jednoduchá vec. Bol to dlhší proces, počas ktorého sme prezerali rôzne existujúce aplikácie venujúce sa viziualizácii javov z tejto oblasti. Z tých sme čerpali inšpiráciu takisto pre vzhľad našej aplikácie. Ďalej sme sa stretávali i s odborníkmi v danej oblasti, od ktorých sme takisto získavali námety, a s ktorými sme konzultovali naše nápady. Išlo väčšinou o poznatky, ktoré sme chceli v aplikácii vizualizovať a o spôsob ich vizualizácie. V mnohých prípadoch sme prišli na to, že vizualizácia takým spôsobom, ako sme si predstavovali, vlastne vôbec nie je možná,

pretože samotná povaha javu ju nedovoľuje. Výsledkom tohto zdĺhavého procesu bol výber jedného javu, dvoch experimentov a jednej hypotézy, ktoré sme nakoniec v programe vizualizovali.

### 2.2.2 Fotoelektrický jav

Fotoelektrický jav je jav, pri ktorom sa z povrchovej vrstvy kovu ožiareného elektromagnetickým žiarením uvoľňujú elektróny. Jav po prvýkrát pozoroval H. Hertz v roku 1887. Meraniami sa zistilo, že čím je dané elektromagnetické žiarenie kratšej vlnovej dĺžky, tým majú uvoľnené elektróny väčšiu energiu. Zvyšovaním intenzity žiarenia sa nemení energia elektrónov, iba rastie ich počet.

Vedelo sa, že elektromagnetické žiarenie je vlnový proces. Avšak pomocou vlnových vlastností žiarenia nebolo možné jav vysvetliť. Jeho analýzu podal v roku 1905 Albert Einstein. Podľa neho sa žiarenie skladá z určitých energetických balíčkov ( kvánt) a pohlcuje sa po týchto kvantách. Pričom jedno kvantum energie ( nazývané aj fotón) je pohltené jedným elektrónom. Kvantum má energiu:

$$E = \frac{2\pi c\hbar}{\lambda}$$

kde $c$ = 300 000 000 m.s$^{-1}$ je rýchlosť svetla, $\hbar$ = 1,054 . 10$^{-34}$ J.s je Planckova konštanta a $\lambda$ je vlnová dĺžka žiarenia. Z tohto vzorca je zrejmé, že energia fotónu, a teda aj uvoľneného elektrónu, závisí iba od vlnovej dĺžky žiarenia.

Elektrón časť prijatej energie spotrebuje na uvoľnenie sa z kovu, zvyšok mu zostane v podobe kinetickej energie. Zapísané rovnicou:

$$\mathrm{E_e} = \varphi + \frac{1}{2} mv^2$$

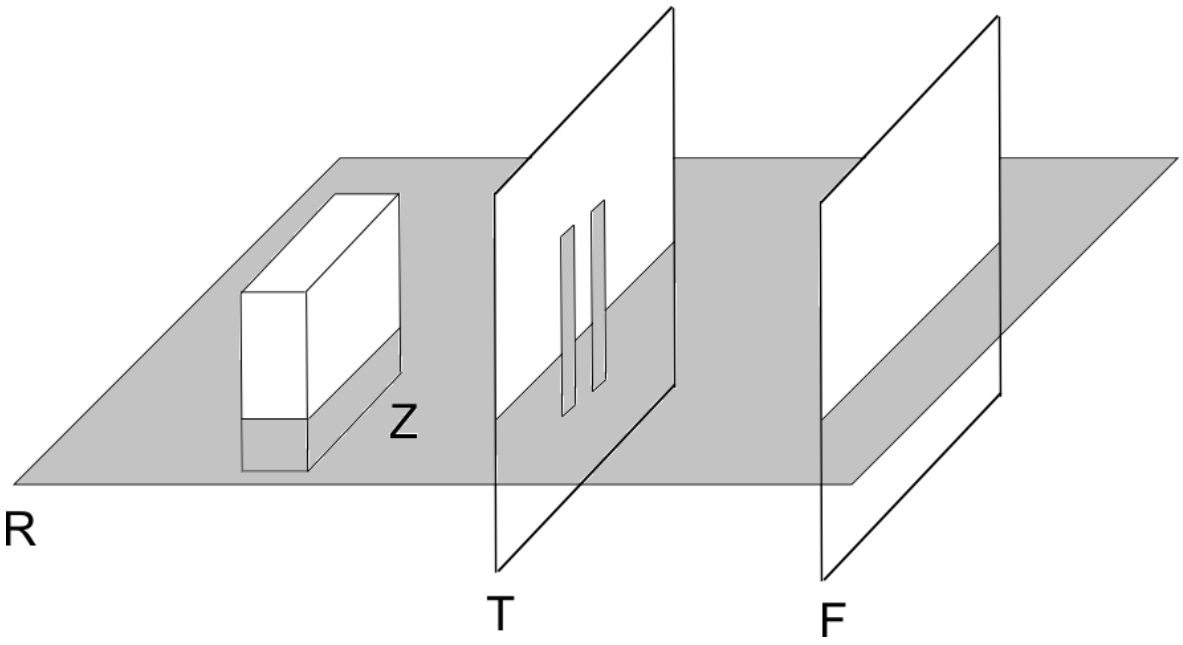
kde $\varphi$ je energia spotrebovaná na uvoľnenie elektrónu z kovu ( výstupná práca) a $\frac{1}{2} mv^2$ je kinetická energia. Zo vzťahu pre kinetickú energiu je možné určiť rýchlosť elektrónu.

### 2.2.3 Dvojštrbinový experiment

Dvojštrbinový experiment je jedným zo základných experimentov vo fyzike mikrosveta. Demonštruje, že mikročastice nie sú ani klasické častice, ktoré sa riadia zákonmi klasickej fyziky, ani vlny klasickej fyziky, ale sú kvantovými objektmi, ktoré majú niektoré vlastnosti klasických častíc a niektoré vlastnosti vĺn.

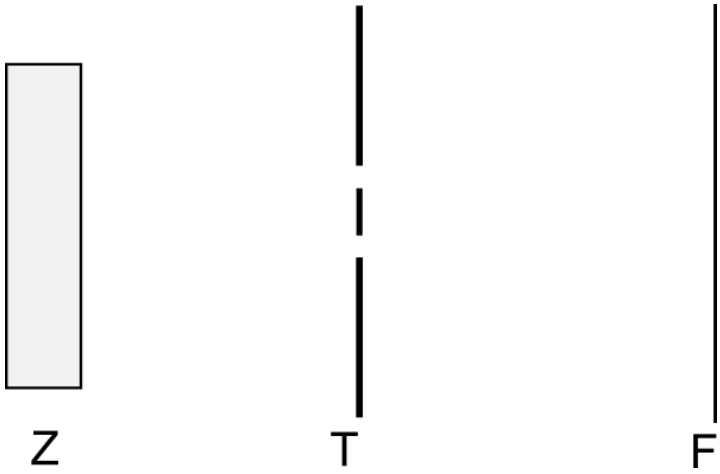
Princíp experimentu je nasledovný. Majme plošný zdroj častíc, ktorý je umiestnený za tienidlom. V tienidle sú dve pozdĺžne štrbiny. Pred tienidlom je v istej vzdialenosti fotografická platňa. Z plošného zdroja vychádzajú rovnobežné zväzky mikročastíc, v našom prípade elektrónov. Ak je vzdialenosť medzi štrbinami tienidla rádovo rovná vlnovej dĺžke vĺn, ktoré prislúchajú elektrónom vychádzajúcim zo zdroja častíc, tak sa na platni vytvorí interferenčný obrazec, čo sú vlastne striedajúce sa tmavé pásy zo svetlými. Interferencia je jav typický pre vlny v klasickej fyzike, teda vznik interferenčného obrazca na platni je dôkazom vlnových vlastností mikročastíc. Ak by sme si ale fotografickú platňu dostatočne zväčšili, zistili by sme, že tmavé pásy sú vytvorené izolovanými bodovými sčerneniami, pričom každé sčernenie je vyvolané dopadom jediného elektrónu. To zase naznačuje, že mikročastice majú aj isté vlastnosti klasických častíc. Fakt, že mikročastice majú aj vlnové aj časticové vlastnosti sa niekedy nazýva vlnovo-časticový dualizmus. Pre simuláciu experimentu je dôležité vedieť vypočítať pravdepodobnosť, s akou častica dopadne na dané miesto fotografickej platne.

Predstavme si aparatúru, pomocou ktorej sa tento experiment prevádza tak, ako sme ju popísali vyššie. Ďalej si predstavme rovinu rezu, ktorá je kolmá na rovinu tienidla, pričom priesečnica týchto dvoch rovín je kolmá na smer, v ktorom sú v tienidle urobené štrbiny ( viď obrázok).




*Obr. 1. Rovina rezu pretínajúca aparatúru. R- rovina rezu, Z- zdroj mikročastíc, T- tienidlo, F- fotografická platňa.*

V reze sa zobrazí zdroj ako obdĺžnik, tienidlo ako tri úsečky ležiace na jednej priamke a fotografická platňa ako úsečka ( viď obrázok).




*Obr. 2. Rez aparatúry. Z- zdroj mikročastíc, T- tienidlo, F- fotografická platňa.*

Ak si rez umiestnime do pravouhlej súradnicovej sústavy a R1 bude bod, kde sa nachádza prvá štrbina, R2 bude bod, kde sa nachádza druhá štrbina, tak potom pravdepodobnosť, s akou mikročastica dopadne do bodu R ležiaceho na platni, sa dá vyrátať zo vzorca

$$P_R = \left| \frac{e^{ik|R1R|}}{\sqrt{|R1R|}} + \frac{e^{ik|R2R|}}{\sqrt{|R2R|}} \right|^2 \times c$$

kde $k = \frac{2\pi}{\lambda}$ , pričom λ je vlnová dĺžka vlny prislúchajúcej mikročastici, |R1R| je veľkosť úsečky R1R ( analogicky pre |R2R|), *e* je Eulerovo číslo, *i* je imaginárna jednotka, *c* je normalizačná konštanta. Teda vieme vyrátať rozloženie pravdepodobnosti dopadu častice na úsečke, na ktorú sa zobrazila platňa v reze. Pre roztiahnutie do plochy platne môžeme vziať rovnomerné rozloženie pravdepodobnosti. Pomocou toho už vieme modelovať dopad častíc na platňu, ktorý bude síce iba približný, ale pre naše účely postačujúci.

### 2.2.4 De Broglieho hypotéza

Túto hypotézu sformuloval začiatkom dvadsiateho storočia francúzky fyzik Louis de Broglie. Hypotéza hovorí, že „každej voľnej častici, pohybujúcej sa s energiou *E* a hybnosťou *p* prislúcha rovinná monochromatická vlna s uhlovou frekvenciou ω a vlnovou dĺžkou λ.“ [Oat10] Pri uvažovaní nad touto hypotézou sa však fyzikom hneď vynorili dve otázky. Prečo neboli vlnové vlastnosti častíc už dávno pozorované? Odpoveďou bolo, že

vlnové dĺžky de Broglieho vĺn sú veľmi malé. Druhá otázka sa pýtala na súvis rýchlosti častice s rýchlosťou vlny, ktorá je tejto častici priradená. Ak by sme zobrali rovinnú monochromatickú vlnu, tak výsledok by bol, že fázová rýchlosť vlny je polovicou rýchlosti častice. Teda rýchlosti sa nerovnali. „Analógiou klasickej častice nie je monochromatická vlna, ale vlnový balík, lebo klasická častica je úplne presne lokalizovaná v určitom bode a vlnový balík je tým, čím môžeme, aspoň približne, dosiahnuť lokalizáciu vlnového procesu.“ [Oat10] Vlnový balík sa pohybuje grupovou rýchlosťou. No a grupová rýchlosť balíka sa už rovná rýchlosti častice. Teda analógiou klasickej častice je vlnový balík. Práve vizualizácii tohto poznatku sa venuje naša simulácia.

### 2.2.5 Rutherfordov experiment

Na začiatku dvadsiateho storočia bol stav poznania týkajúci sa stavby atómu nasledovný. Vedelo sa, že atóm je elektricky neutrálny a má rozmery okolo $10^{-10}$ m. „V atóme je kladný celkový náboj rovný $Ze$, kde $Z$ je poradové číslo prvku v Mendelejevovej periodickej tabuľke a záporný náboj $Z(-e)$, rozdelený medzi $Z$ elektrónov, z ktorých každý má náboj $(-e)$.“ [Oat10] Hmotnosť elektrónu je v porovnaní s hmotnosťou atómu veľmi malá, čo znamenalo, že celková hmotnosť atómu musí nejako súvisieť s kladným nábojom. Avšak ako je atóm vnútorne usporiadaný, to sa nevedelo. Všeobecne prijímaným modelom bol Thompsonov model atómu, podľa ktorého je celý objem atómu vyplnený akousi hmotou, ktorá je nositeľom kladného náboja a väčšinovej hmotnosti atómu a elektróny sa voľne vznášajú v tejto hmote.

V tomto čase Rutherford uskutočnil svoj experiment. Vzal rádioaktívny prvok, ktorý mu slúžil ako zdroj α častíc. Umiestnil ho za stenu s tenkým otvorom. α častice prenikali iba cez tento tenký otvor, čím získal lúč α častíc. Týmto lúčom bombardoval tenkú zlatú fóliu. Za fóliou bolo umiestnené tienidlo zhotovené z materiálu, ktorý po dopade α častice vydával záblesk. Tento záblesk sa už dal pozorovať. Podľa vtedy dostupných poznatkov mala väčšina α častíc zlatou fóliou preletieť bezo zmeny smeru. Niektoré z nich sa mali odchýliť o malé uhly od pôvodného smeru. Avšak keď experiment prevádzal, zistil, že častice sa odchyľujú od pôvodného smeru o oveľa väčšie uhly ako predpokladal. Niektoré z nich sa dokonca od fólie odrazili a smerovali tam, odkiaľ prišli. Podrobnou analýzou experimentálnych údajov nakoniec prišiel na to, že celý kladný náboj, a teda aj hmotnosť atómu, je sústredený v malom jadre v strede atómu, ktoré má rozmery rádovo $10^{-14}$ m.

## 2.3 Simulácie

Podľa [SMA] sa simulácie používajú na imitovanie správania sa procesov alebo zariadení reálneho sveta pomocou počítačov. Takéto zariadenie, či proces sa zvyčajne nazýva systém. Systémy bývajú často veľmi zložité a nie je jednoduché predpovedať ako sa budú správať, ak v nich urobíme nejakú zmenu. Príkladom môže byť spoločnosť, ktorá uvažuje nad rozšírením svojho závodu, avšak nevie posúdiť, či zvýšená produkcia pokryje náklady na rozšírenie závodu. Môžu však urobiť simuláciu práce rozšíreného závodu. To im poskytne odhad, aký veľký nárast v produkcii by znamenalo rozšírenie závodu. Toto je hlavný účel simulácii. Poskytnúť pohľad na systém, o ktorom nevieme ako sa bude správať v reálnom svete. Ak by sme zmenu v systéme vykonali a ona by sa neosvedčila, mohlo by nás to stáť nemalé prostriedky zase uviesť systém do pôvodného stavu. Treba však povedať, že simulácie nedávajú exaktné výsledky. Poskytujú ale odhad, ktorý môže byť relevantný a môže uľahčiť rozhodovanie.

V našej aplikácii neskúmame nejaký systém, o ktorom nevieme ako by sa správal. Fyzikálne zákonitosti sú dané a poznáme výsledky, ktoré nám experimenty priniesli. Avšak pri simulovaní experimentov používame niektoré postupy, ktoré sa pri simulácii správania sa neznámych systémov používajú.

### 2.3.1 Systém

Podľa [SMA] je systém množina entít, ktoré sa nejako správajú a navzájom sa ovplyvňujú, za účelom dosiahnutia určitého logického cieľa. Čo znamená systém v praxi je otázkou cieľov skúmania. Ak chceme skúmať iba určitú vlastnosť systému, tak množina entít môže byť iba časťou množiny, ktorá opisuje systém ako celok. „Stav systému je množina premenných potrebných na opis systému v danom čase so zreteľom na cieľ skúmania.“ (podľa [SMA]) Rozlišujeme dva druhy systémov: diskrétne a kontinuálne.

Diskrétne systémy sú také, v ktorých sa zmena stavových premenných deje v určitých časových okamihoch. Predstavme si napríklad supermarket. Počet zákazníkov v ňom je stavová premenná. Tá sa mení iba v okamihu, kedy zákazník dorazí do supermarketu, alebo v okamihu, kedy ho opúšťa.

Kontinuálne systémy sú také, v ktorých sa zmena stavových premenných deje plynule v čase. Napríklad poloha letiaceho lietadla sa v čase neustále plynule mení.

Treba však povedať, že iba málo reálnych systémov je striktne diskrétnych alebo kontinuálnych. Väčšinou však jedna vlastnosť prevláda a podľa nej sa potom daný systém kategorizuje.

### 2.3.2 Simulačný model

Ak chceme systém študovať, musíme pochopiť ako vnútorne funguje, aké sú v ňom vzťahy medzi jednotlivými entitami a pod. Množina takýchto predpokladov tvorí model systému. Ak máme vytvorený model systému, potom pomocou neho môžeme nájsť odpovede na naše otázky. Ak je model dostatočne jednoduchý, môžeme pomocou presných matematických výpočtov zistiť odpoveď na našu otázku. Skutočné systémy sú však často veľmi zložité a vystupuje v nich aj prvok náhodnosti. Nie je teda možné pomocou matematických výpočtov zistiť informácie, ktoré nás zaujímajú. V takýchto prípadoch je nutné študovať model z hľadiska simulácie. Teda pre konkrétne vstupy sledovať, aké výstupy dostaneme použitím modelu a vyvodiť z nich závery. Model, ktorý študujeme z hľadiska simulácie, nazývame simulačný model. Poznáme delenia simulačných modelov z viacerých hľadísk.

Podľa toho, či záleží na čase, sa delia na statické a dynamické. Statický simulačný model reprezentuje systém v konkrétnom časovom okamihu, alebo reprezentuje systém, v ktorom čas nezohráva žiadnu úlohu. Dynamické modely zase reprezentujú systémy ako sa vyvíjajú v čase. Podľa toho, či obsahujú náhodné prvky, sa delia na deterministické a pravdepodobnostné. Deterministické modely neobsahujú žiadne náhodné prvky, pravdepodobnostné ich obsahujú. Podľa toho ako dochádza k zmene stavu systému sa delia na diskrétne a kontinuálne. Pri diskrétnych sa zmena stavových premenných odohráva v určitých časových okamihoch, pri kontinuálnych sa stavové premenné menia spojite v čase.

Simulačné modely, ktoré nás zaujímajú, sú dynamické pravdepodobnostné diskrétne modely. Pri týchto modeloch sa stavové premenné menia v určitých konkrétnych časových okamihoch. Hovoríme, že sa menia pri určitých udalostiach. Keďže sú to dynamické modely, svoju rolu pri nich zohráva čas. Pri simulovaní systémov pomocou tohto druhu simulačného modelu musíme brať ohľad na čas, v ktorom sa práve simulácia nachádza a musíme si vybrať mechanizmus, akým budeme čas posúvať. Existujú dva základné princípy ako posúvať čas. Posúvať čas vždy na okamih nasledujúcej udalosti. Tento spôsob je asi najbežnejší pri väčšine simulácii tohto typu. Keďže vieme

predpovedať, kedy nastane ďalšia udalosť, jednoducho vždy skočíme na okamih, kedy táto udalosť nastane, zmeníme stav systému a takto pokračujeme ďalej. Druhým spôsobom je posúvanie času vždy o rovnaké časové úseky. Pri tomto spôsobe posunieme čas o zvolený časový úsek a potom sa pozrieme, či medzitým nenastala nejaká udalosť. Ak áno, tak prehlásime aktuálny čas za čas, kedy udalosť nastala. Problém pri tomto druhu posúvania času je ten, že ak nastane viacero udalostí počas časového úseku, ktorý sme preskočili, tak sa musíme rozhodnúť, ktorá zmení stav systému ako prvá, keďže čas, kedy nastali, je aktuálny čas, na ktorý sme práve skočili. Ak však vieme, že udalosti sa v systéme stanú vždy po uplynutí nejakého konštantného času, môžeme zvoliť túto metódu. Takisto sa táto metóda používa, ak chceme umelo generovať udalosti po ubehnutí určitého času.

### 2.3.3 Vizualizácia fyzikálneho javu

V našej aplikácii sú systémami konkrétne fyzikálne experimenty. Entitami sú častice, fotografická platňa, zlatá fólia, atď. Stavovými premennými sú pozície jednotlivých častíc v rámci systému, počet a pozícia čiernych bodiek na fotografickej platni a pod. Modelom systému sú fyzikálne zákonitosti, podľa ktorých sa jednotlivé entity v rámci experimentov správajú. Udalosti nastávajú v každom experimente po konkrétnych časových intervaloch. Pri týchto udalostiach sa väčšinou menia pozície jednotlivých entít v rámci systému. Entity spolu reagujú ( napr. ak α častica dosiahne zlatú fóliu, môže cez ňu prejsť, odraziť sa od nej, alebo prejsť so zmenou smeru). Takisto pri určitých udalostiach sa rozhoduje na základe istej pravdepodobnosti, čo sa stane, pri niektorých sa uskutočňuje náhodný výber. Vidno teda, že naša aplikácia sa dá charakterizovať pojmami používanými pri simuláciách, i keď jej účelom nie je skúmanie neznámych vlastností určitého systému, ale simulovanie priebehu fyzikálnych javov.

## 2.4 Multithreadové aplikácie a synchronizácia

Zjednodušene by sme mohli povedať, že proces je bežiaci program. Existujú síce aplikácie, ktoré pozostávajú z viacerých procesov, avšak naša aplikácia bude bežať ako jeden proces. Každá aplikácia obsahuje aspoň jeden *thread* ( vlákno). Ak však aplikácia obsahuje viacero *threadov,* nazývame ju *multithreadová*. To je prípad našej aplikácie.

*Thready* zdieľajú, okrem iného, virtuálnu pamäť procesu, ku ktorému patria. Môžu teda modifikovať zdieľané premenné. Počas behu programu však môže dôjsť k tomu, že dva *thready* budú chcieť naraz používať tú istú premennú. Keďže vykonávané operácie často nebývajú atomické, môže sa stať, že sa ich vykonávanie takpovediac premieša. Tu

môže nastať problém, ak aspoň jeden z *threadov* bude premennú modifikovať. Ako ilustrácia nám môže poslúžiť nasledujúci príklad. Majme jednorozmerné päťprvkové pole typu *Integer* a premennú typu *integer*, ktorá slúži ako index do tohto poľa. Majme *thread* A, ktorý k prvku na danom indexe pripočíta 1 a zvýši index o 1 a *thread* B, ktorý od prvku na danom indexe odpočíta 1 a takisto zvýši index o 1. Predstavme si, že index je nastavený na prvý prvok poľa a my spustíme oba *thready*. Očakávali by sme, že sa modifikuje hodnota prvých dvoch prvkov poľa a index bude ukazovať na tretí prvok poľa. Môže však prísť k nasledujúcej situácii. *Thread* A pripočíta k prvému prvku poľa 1, vtedy *thread* B od neho 1 odpočíta a posunie index a nakoniec *thread* A posunie index. Výsledkom bude, že obsah poľa zostane nezmenený.

Tento druh problému sa odstraňuje pomocou synchronizácie. Tá nám zabezpečí, že operácie, ktoré sa majú vykonať naraz za sebou, sa tak naozaj vykonajú a vždy dosiahneme očakávaný výsledok. Aby bolo možné používať synchronizáciu, musí byť podpora pre ňu súčasťou operačného systému. Všetky moderné operačné systémy túto podporu poskytujú.

## 2.5 Programovací jazyk Java

V tejto podkapitole sme opísali dôvody, prečo sme si zvolili na naprogramovanie aplikácie programovací jazyk Java. Ďalej sa zaoberáme riešením synchronizácie v jazyku Java a pozrieme sa na jeden zo spôsobov sieťovej komunikácie.

### 2.5.1 Dôvody výberu programovacieho jazyka Java

Hlavným dôvodom, prečo sme si vybrali programovací jazyk Java je, že programátorovi vo viacerých smeroch uľahčuje jeho prácu. V Jave sa nemusí starať o uvoľňovanie pamäte objektov, ktoré už nechce používať, robí to za neho *Garbage Collector*. Nemusí si dávať pozor, ako napr. v jazyku C++, aby pri indexovaní do poľa nebol index väčší ako je index posledného prvku v poli. V C++ by si týmto prepísal pamäť za poľom, v Jave takýto program vyhodí výnimku, ktorá ho na tento problém upozorní, a tak ho môže odstrániť. Java takisto ponúka programátorovi množstvo „packageov“ ( balíčkov), v ktorých sú naprogramované často používané triedy ( prioritný rad, množina, mapa, ...), a teda programátor si nemusí robiť starosti a strácať čas ich programovaním. Navyše Java je veľmi rozšírený jazyk a existuje preň viacero vývojových prostredí. Okrem toho je to multiplatformový jazyk, takže program, ktorý v ňom programátor napíše a skompiluje do byte kódu, môže potom bežať na rôznych operačných systémoch.

### 2.5.2 Synchronizácia v Jave

Podľa [JLS3] sa synchronizácia v Jave rieši použitím *monitorov*. S každým objektom v Jave je asociovaný *monitor*. *Thread* môže *zamknúť* alebo *odomknúť monitor*. Ak jeden *thread zamkne monitor* na danom objekte, druhý ho už *zamknúť* nemôže, ale zostane blokovaný, pokiaľ prvý *neodomkne*. Ten istý *thread* však môže na danom monitore uskutočniť *zamknutie* viackrát. Jedna operácia *odomknutia* potom *odomyká* jednu operáciu *zamknutia*.

Pri synchronizovaných metódach sa pri volaní funkcie automaticky *zamyká monitor* na objekte, ktorého metóda bola volaná. Ak nejde o statickú metódu, *zamyká sa monitor* konkrétnej inštancie triedy. Ak ide o statickú metódu, tak sa zamkne monitor na *Class* objekte danej triedy. Pokiaľ sa úspešne neukončí *zamknutie*, tak sa nezačne vykonávať telo funkcie. Po ukončení vykonávania tela metódy sa automaticky vykoná *odomknutie monitora* daného objektu.

```
synchronized void vymazVsetkyCastice(){
    alfaCastice.clear();
    detailCastice.clear();
}
```

*Obr.3. Príklad synchronizovanej metódy.*

Pri synchronizovaných sekciách sa *zamyká monitor* objektu, s ktorým je sekcia asociovaná. Telo sekcie sa nezačne vykonávať, pokiaľ sa úspešne nevykoná *zamknutie* na *monitore* daného objektu. Po ukončení vykonávania tela sekcie sa *odomkne monitor* objektu, s ktorým bola sekcia asociovaná.

```
synchronized (this){
    kontrolujZakaznika();
}
```

*Obr. 4. Príklad synchronizovanej sekcie.*

Java ponúka aj ďalšie spôsoby synchronizácie ( napr. *volatile* premené, triedy v „packagei“ *java.util.concurrent*).

### 2.5.3 Sockety

*Socket* je jeden koncový bod obojsmernej komunikačnej linky medzi dvomi programami, ktoré spolu komunikujú po sieti. (podľa [SOC]) *Sockety* sa používajú pri

aplikáciách typu *client-server*. Server s klientom spolu komunikujú prostredníctvom TCP spojenia. Na každom konci tohto komunikačného kanála je *socket*. Jeden je na strane klienta, jeden na strane servera. Serverovská i klientská aplikácia zapisujú i čítajú údaje zo svojho *socketu*, a tak spolu komunikujú.

Spôsob použitia *socketov* je nasledovný. Server má na svojom počítači vytvorený tzv. *server socket*, ktorý je spojený s konkrétnym portom a čaká na ňom na žiadosti o spojenie od klientov.

Klient pozná IP adresu počítača servera a číslo portu, na ktorom sa čaká na prichádzajúce spojenia. Klient pošle na daný počítač a port žiadosť o spojenie, v ktorej počítaču servera oznámi IP adresu svojho počítača a číslo portu, ktoré bude používať počas spojenia. Server žiadosť o spojenie akceptuje a vytvorí si nový *socket*, ktorý naviaže na IP adresu a číslo portu, ktoré dostal od klienta. Tento *socket* je takisto spojený s číslom portu, na ktorom počúva *server socket*.

Akceptáciou žiadosti o spojenie sa vytvorí komunikačná linka, a teda sa dokončí vytváranie *socketu* na strane klienta. Obe strany môžu teraz používať svoje *sockety* na vzájomnú komunikáciu. Server naďalej pomocou *server socketu* čaká na ďalšie žiadosti o spojenie.

V programovacom jazyku Java máme na prácu so *socketmi* k dispozícii dve triedy. Trieda *ServerSocket* implementuje *server socket*, pomocou ktorého server čaká na žiadosti o spojenie. Trieda *Socket* implementuje *socket*, ktorý obe strany používajú na vzájomnú komunikáciu.

### 2.5.4 Netbeans IDE

Pre tvorbu svojej aplikácie sme si vybrali vývojové prostredie Netbeans IDE. Dôvodom výberu tohto nástroja bolo, že s ním už máme praktické skúsenosti, a že poskytuje podporu pre tvorbu aplikácií s užívateľským rozhraním.

# 3 Špecifikácia cieľov

V tejto kapitole sme popísali všeobecnú špecifikáciu cieľov vychádzajúcu zo zadania práce, ktorú sme potom rozvinuli do konkrétnej špecifikácie, ktorá obsahuje podrobný popis správania sa výslednej aplikácie.

## 3.1 Všeobecná špecifikácia

Cieľom mojej bakalárskej práce bolo naprogramovať výukovú aplikáciu, ktorá demonštruje zvolené fyzikálne poznatky. Aplikácia mala byť ľahko použiteľná pre bežného stredoškoláka. Mala obsahovať simulácie napr. fyzikálnych experimentov. Simulácie by mali byť interaktívne, teda ak je to možné, aby užívateľ mohol nastavovať ich parametre.

## 3.2 Konkrétna špecifikácia

Aplikácia bude pozostávať zo štyroch simulácií, konkrétne zo simulácie fotoelektrického javu, simulácie dvojštrbinového experimentu, simulácie Rutherfordovho experimentu a vizualizácie faktu, že analógiou klasickej častice je vlnový balík. Ku každej simulácii bude pripravený test, na ktorom si bude môcť študent overiť, či správne pochopil danú tému. V spodnej časti okna aplikácie sa bude pri každej simulácii zobrazovať text, ktorý ju vysvetľuje. Vysvetľujúci text bude textové pole, ktoré sa bude dať zväčšiť, aby sa dalo ľahko študovať. Aplikácia musí umožňovať, aby sa dalo medzi jednotlivými simuláciami kedykoľvek jednoducho prepínať. Podrobná špecifikácia jednotlivých simulácií je uvedená v nasledujúcich podkapitolách.

Aplikáciu bude možné spustiť v dvoch módoch: učiteľskom a študentskom. Idea používania programu je taká, že ho bude mať počas hodiny v počítačovej učebni spustený každý študent na svojom počítači v študentskom móde a učiteľ na svojom počítači v učiteľskom móde. Počítače v učebni sú zapojené do jednej lokálnej siete. V študentskom móde bude možné študovať experimenty a testovať sa. V učiteľskom móde bude možné experimenty rovnako študovať, avšak učiteľ bude mať možnosť zvoliť výučbu študentov. Vtedy sa študentom zablokujú ovládacie prvky programu a ich program sa bude správať podľa toho, ako sa správa učiteľov program. Teda učiteľ môže sedieť za svojím počítačom, vysvetľovať a ovládať program a rovnaké akcie, ktoré on vykoná vo svojom programe, sa budú vykonávať aj v programoch študentov.

### 3.2.1 Simulácia fotoelektrického javu

Na obrazovke musí byť obrázok aparatúry na skúmanie fotoelektrického javu a ovládacie prvky na nastavovanie parametrov simulácie a spúšťanie simulácie. Konkrétne *slider* na nastavenie vlnovej dĺžky svetla, *slider* na nastavenie intenzity svetla, *combo box* pre výber kovu, z ktorého bude zhotovená katóda, tlačidlo na zasvietenie svetla, tlačidlo na zobrazenie testu a popisujúce *labely*.

Na stlačenie tlačidla na zasvietenie svetla sa podľa nastavenej hodnoty vlnovej dĺžky svetla zvolí farba svetelných lúčov. Na základe nastavenej intenzity svetla sa určí počet lúčov, ktoré sa následne vystrelia z lampy smerom ku katóde. Minimálny počet lúčov je jeden, maximálny je desať. Na katóde je desať elektrónov. Ak je lúčov menej ako desať, zasiahnu vždy náhodne vybrané elektróny. Po dopade lúčov na katódu sa podľa druhu kovu, z ktorého bola katóda zhotovená ( ten sme si vybrali pomocou *combo boxu*) a podľa vypočítanej energie fotónu určí, či sa zasiahnuté elektróny z katódy uvoľnia alebo nie. Ak áno, tak sa vypočíta ich rýchlosť a začne sa animovať ich pohyb k anóde. Po tom, ako elektróny dosiahnu anódu sa na základe ich počtu vychýli ručička na ampérmetri, čím sa signalizuje, že obvodom tečie elektrický prúd.

Po stlačení tlačidla na zasvietenie svetla sa na ňom zmení popisujúci text a pokiaľ bude simulácia prebiehať, bude slúžiť na zastavenie simulácie. Ak ho používateľ použije na zastavenie simulácie, alebo simulácia dobehne, tak sa opäť jeho funkcionalita zmení na zasvietenie svetla, a teda na spúšťanie simulácie.

Po uskutočnení výberu kovu, z ktorého je zhotovená katóda, pomocou *combo boxu*, katóda na obrázku aparatúry zmení farbu na farbu príslušného vybraného kovu.

### 3.2.2 Simulácia dvojštrbinového experimentu

Na obrazovke musí byť obrázok aparatúry na vykonanie dvojštrbinového exerimentu, tlačidlo „štart“, pomocou ktorého sa spustí simulácia, tlačidlo „stop“, pomocou ktorého sa simulácia zastaví, *slider* na nastavenie počtu častíc dopadajúcich na fotografickú platňu za sekundu, popisujúce *labely* a tlačidlo pre zobrazenie testu.

Na stlačenie tlačidla „štart“ sa zobrazí pohľad spredu na fotografickú platňu, začne sa animovať vysielanie rovnobežných zväzkov častíc zo zdroja mikročastíc ako aj vlnenie, ktorého zdrojom budú štrbiny v tienidle a na platni sa začnú zobrazovať bodové sčernenia, ktoré budú naznačovať dopad častice na dané miesto. Na stlačenie tlačidla „stop“ sa

animácia zastaví, pričom pohľad na platňu spredu spolu so sčerneniami ostane viditeľný. Počet častíc, ktoré za sekundu na platňu dopadnú, sa bude dať pomocou *slidera* kedykoľvek meniť, teda aj počas behu animácie.

### 3.2.3 Simulácia Rutherfordovho experimentu

Na obrazovke musí byť obrázok aparatúry na vykonanie Rutherfordovho experimentu, tlačidlo na spustenie simulácie, ktorá bude ukazovať, aký výsledok Rutherford očakával, tlačidlo na spustenie simulácie, ktorá bude ukazovať skutočný výsledok experimentu, tlačidlo na zobrazenie detailu zlatej fólie, tlačidlo na zastavenie simulácie a tlačidlo pre zobrazenie testu.

Na stlačenie tlačidla pre simuláciu očakávaného výsledku experimentu sa začnú zo zdroja α častíc uvoľňovať častice náhodne na všetky smery. Častice, ktoré dorazia k zlatej fólii buď cez ňu prejdú bezo zmeny smeru, alebo sa jemne odchýlia od pôvodného smeru, pričom väčšina častíc prejde bezo zmeny smeru. Po náraze častice do tienidla kruhového tvaru vyvolá častica záblesk.

Na stlačenie tlačidla pre simuláciu skutočného výsledku experimentu sa začnú častice rovnako uvoľňovať, ale pri dopade na fóliu sa s malou pravdepodobnosťou môžu odchýliť o väčšie uhly od pôvodného smeru, či dokonca odraziť sa od fólie smerom dozadu.

Na stlačenie tlačidla zobrazujúceho detail fólie sa vedľa aparatúry zobrazí zväčšený pohľad spredu na zlatú fóliu, pričom aj na tejto zväčšenine sa budú animovať prelietavajúce α častice. Animácia na detaile musí korešpondovať s animáciou na aparatúre. Na opätovné stlačenie tohto tlačidla detail fólie zmizne. Tlačidlo na zobrazenie detailu fólie musí byť viditeľné iba pri simulácii reálneho výsledku experimentu.

Tlačidlo na zastavenie simulácie nebude možné na začiatku stlačiť. Po spustení simulácie, či už očakávaného alebo skutočného výsledku, sa tlačidlo stane aktívnym ( bude sa dať stlačiť). Na jeho stlačenie sa vykonávanie simulácie zastaví.

### 3.2.4 De Broglieho hypotéza (analógiou častice je vlnový balík)

Na obrazovke musí byť obrázok dvoch prekážok, od ktorých sa bude častica ( vlnový balík) odrážať, tlačidlo „prepni“, ktorým sa bude prepínať medzi časticou a vlnovým balíkom, tlačidlo „pauza“, ktoré pozastaví a potom opätovne spustí animáciu, tlačidlo „štart“, ktoré spúšťa animáciu a tlačidlo pre zobrazenie testu.

Na stlačenie tlačidla „štart“ sa spustí animácia guľôčky. Tá sa bude pohybovať medzi dvoma prekážkami, od ktorých sa bude odrážať. Tlačidlo „štart “ sa dá stlačiť iba na začiatku animácie.

Na stlačenie tlačidla „prepni“ sa guľôčka zmení na vlnový balík, ktorý sa zobrazí na tom istom mieste, kde sa nachádzala guľôčka a bude sa pokračovať v jeho animácii. Na opätovné stlačenie tlačidla sa balík zmení znovu na guľôčku. Tlačidlo „prepni“ musí fungovať v akomkoľvek momente animácie.

### 3.2.5 Testy

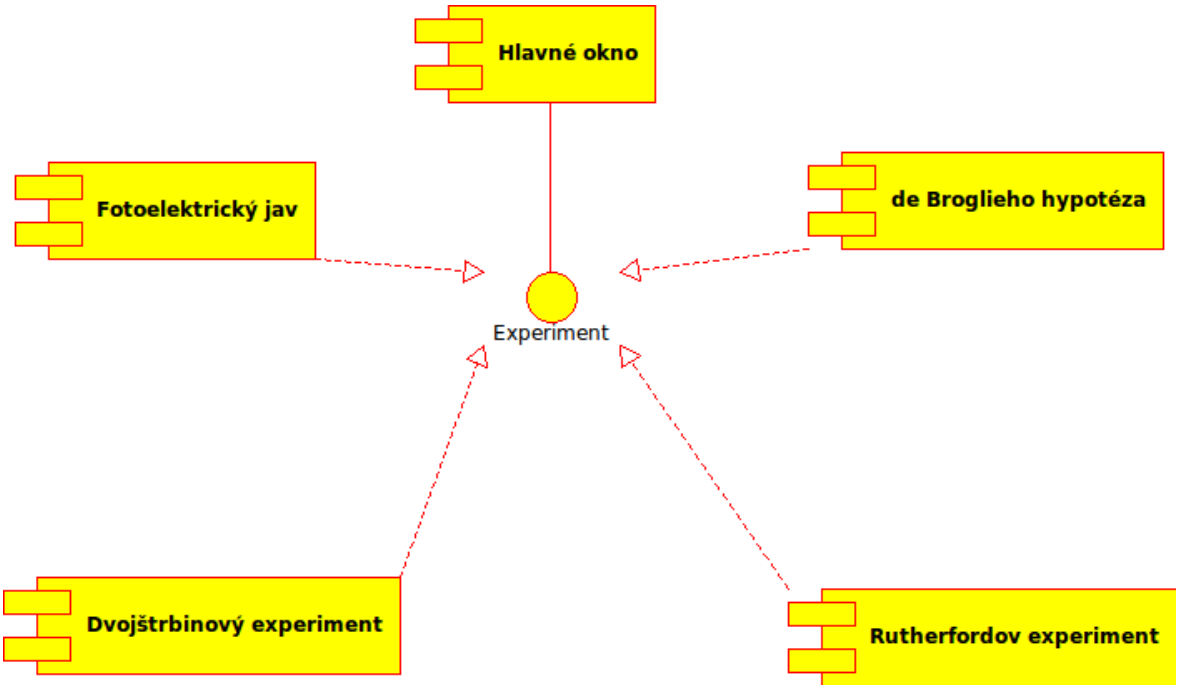
Testy sa budú zobrazovať v samostatnom okne po stlačení príslušného tlačidla na obrazovke konkrétnej simulácie. Test bude pozostávať z viacerých otázok. Na každú otázku budú štyri možné odpovede, z ktorých si bude musieť užívateľ jednu vybrať. Potom bude môcť stlačením tlačidla prejsť na ďalšiu otázku. Ak by si nevybral žiadnu odpoveď a chcel by prejsť na ďalšiu otázku, aplikácia mu to nedovolí a upozorní ho, že si nezvolil žiadnu odpoveď. Po zodpovedaní všetkých otázok sa mu zobrazí vyhodnotenie. Vo vyhodnotení bude napísané, koľko otázok zodpovedal správne, ako aj konkrétne pre každú otázku napísané, či na ňu odpovedal správne, alebo nie. Ak na ňu odpovedal nesprávne, tak sa zobrazí celé znenie otázky, užívateľova nesprávna odpoveď a správna odpoveď na otázku.

# 4 Návrh

V tejto kapitole sme opísali návrh aplikácie, ktorý sme vymysleli tak, aby spĺňal uvedenú špecifikáciu. Najprv opisujeme architektúru aplikácie, teda vzájomné logické usporiadanie komponentov aplikácie, ako aj vnútorné usporiadanie jednotlivých komponentov. Na konci návrhu opisujeme používateľské rozhranie aplikácie.

## 4.1 Architektúra aplikácie

Hlavným komponentom aplikácie je hlavné okno, ktoré spravuje komponenty jednotlivých simulácií. Komponenty simulácií sú navzájom nezávislé.



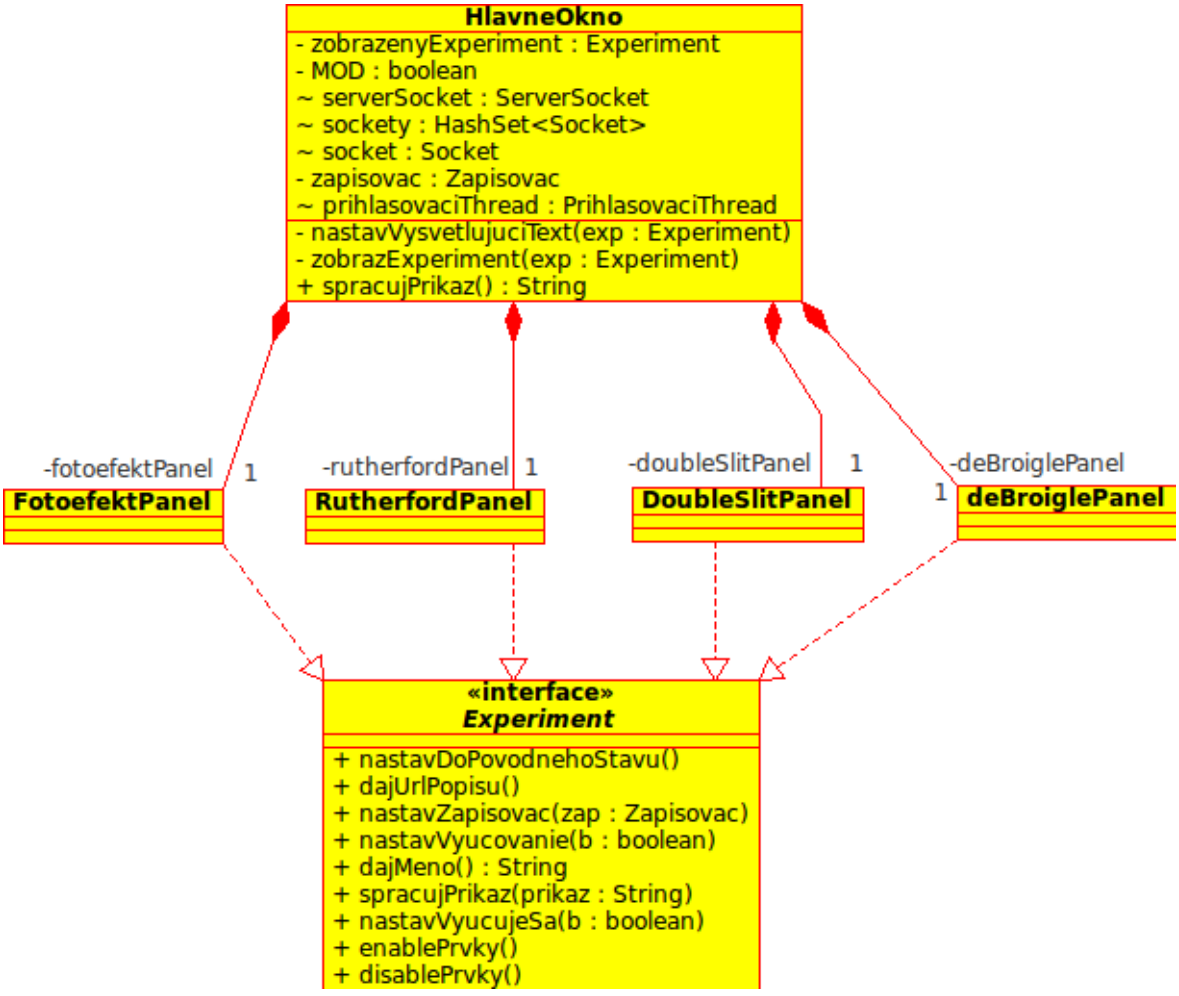

*Obr. 5. Komponenty aplikácie a ich vzájomné vzťahy.*

## 4.2 Komponent Hlavné okno

Komponent hlavné okno je samostatný *package*. Obsahuje verejnú triedu *HlavneOkno*, ktorá je potomkom triedy *JFrame*. Tá sa dá spustiť v dvoch módoch, študentskom a učiteľskom. Ďalej obsahuje interface *Experiment*, triedy *PrihlasovaciThread*, *PrijimaciThread* a *SocketThread*, ktoré sú potomkami triedy *Thread* a používajú sa na sieťovú komunikáciu medzi aplikáciou spustenou v učiteľskom móde a aplikáciami spustenými v študentskom móde, a triedu *Zapisovac*, ktorá sa takisto používa na sieťovú komunikáciu. Hlavné okno má na starosti, aby bol v danom okamihu zobrazený

práve jeden komponent simulácie. Takisto má na starosti zobrazovanie popisujúceho textu k jednotlivým simuláciám, nastavovanie simulácií, ktoré sa stávajú neviditeľnými, do pôvodného stavu a sieťovú komunikáciu.

Interface *Experiment* obsahuje dve verejné metódy *nastavDoPovodnehoStavu()* a *dajUrlPopisu()*, ktoré sa používajú pri zobrazovaní experimentov, tri verejné metódy *nastavZapisovac(Zapisovac zap)*, *nastavVyucovanie(boolean b)* a *dajMeno()*, ktoré sa používajú na sieťovú komunikáciu aplikácie spustenej v učiteľskom móde a štyri verejné metódy *spracujPrikaz(String prikaz)*, *nastavVyucujeSa(boolean b)*, *enablePrvky()* a *disablePrvky()*, ktoré sa používajú, keď aplikácia beží v študentskom móde, na spracovanie príkazov prijatých od učiteľskej aplikácie.



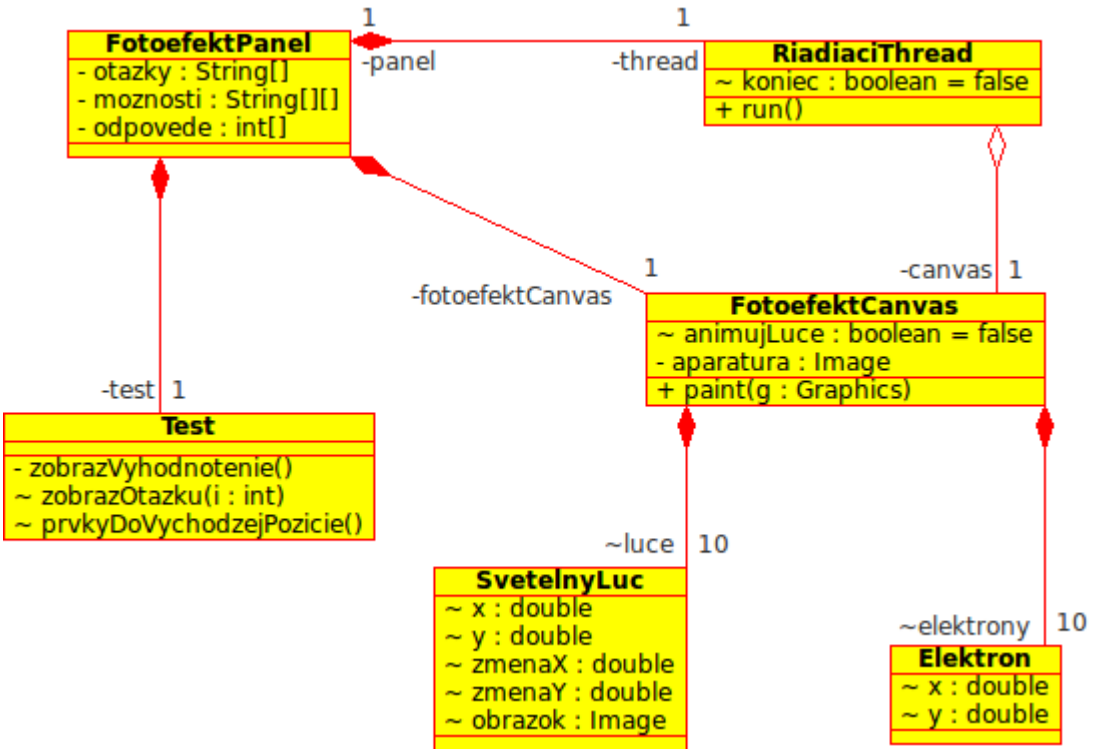

*Obr. 6. Diagram tried komponentu Hlavné okno.*

## 4.3 Komponenty simulácií

Každý komponent simulácie je samostatný *package*. Ten obsahuje verejnú triedu, ktorá je potomkom triedy *JPanel* a implementuje interface *Experiment*. Táto trieda sa zobrazuje v hlavnom okne pri štúdiu konkrétnej simulácie. *Package* ďalej obsahuje triedy, ktoré sú zodpovedné za vykresľovanie simulácie podľa zvolených parametrov, triedu *Test*, ktorá sa používa na testovanie nadobudnutých znalostí, HTML súbor s popisom simulácie a obrázky potrebné pri simuláciách.

### 4.3.1 Komponent Fotoelektrický jav

Verejná trieda *FotoefektPanel* obsahuje ovládacie prvky simulácie, *canvas*, do ktorého sa simulácia vykresľuje a referenciu na *thread*, ktorý simuláciu riadi. Ovládacími prvkami sa nastavujú parametre simulácie a spúšťa sa riadiaci *thread* simulácie. *Thread* z ovládacích prvkov zistí potrebné údaje a vykresľuje do *canvasu* simuláciu.




*Obr. 7. Diagram tried komponentu Fotoelektrický jav.*

**Spôsob animácie**

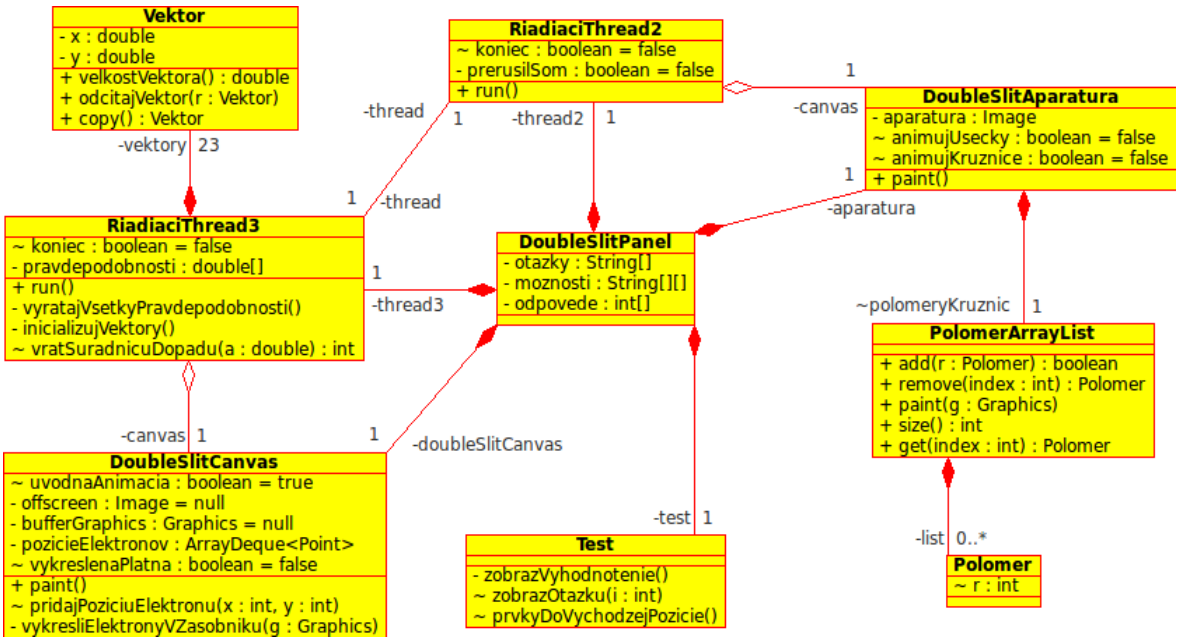
*Canvas* si pamätá pole desiatich svetelných lúčov a pole desiatich elektrónov. Riadiaci *thread* si vyberie n elektrónov, ktoré budú zasiahnuté lúčmi, na základe hodnoty

intenzity nastavenej užívateľom. Potom n lúčom zo začiatku poľa nastaví správny smer. Ďalej v cykle mení postupne súradnice lúčom a prekresľuje *canvas*, kým lúče nedorazia k elektrónom. Potom vypočíta kinetickú energiu elektrónov. Ak je väčšia ako nula, tak vypočíta z nej zmenu súradnice x pre elektróny. Elektróny v cykle postupne posúva a prekresľuje *canvas*, kým elektróny nedosiahnu anódu. Potom podľa počtu elektrónov zabezpečí animáciu ručičky ampérmetra. Nakoniec nastaví elektrónom aj lúčom pôvodné hodnoty ich súradníc.

*Canvas* na každé prekreslenie vykreslí katódu, vykreslí elektróny, vykreslí ručičku ampérmetra, a ak má nastavenú premennú *animujLuce* na *true*, tak vykreslí aj príslušné lúče. Sú to lúče zo začiatku poľa lúčov, ktorým *thread* nastavil smer na elektróny.

### 4.3.2 Komponent Dvojštrbinový experiment

Verejná trieda *DoubleSlitCanvas* obsahuje ovládacie prvky simulácie, dva *canvasy* a referencie na dva riadiace *thredy* simulácie. Do prvého *canvasu* sa animuje aparatúra experimentu, do druhého sa animuje dopad elektrónov na fotografickú platňu. Ovládacie prvky nastavujú parametre simulácie a spúšťajú prvý *thread*. Ten animuje úvodnú animáciu fotografickej platne, dopad elektrónov na ňu a spúšťa druhý *thread*. Druhý *thread* animuje aparatúru.




*Obr. 8. Diagram tried komponentu Dvojštrbinový experiment.*

**Spôsob animácie**

*DoubleSlitCanvas* si pamätá jeden *Image* a zásobník *pozicieElektronov*. *RiadiaciThread3* najprv v cykle postupne zväčšuje dĺžku strany obdĺžnika a prekresľuje *DoubleSlitCanvas*. Zobrazuje sa tým úvodná animácia. Potom spustí *RiadiaciThread2* a uspí seba samého. Keď ho *RiadiaciThread2* zobudí, pokračuje tým, že v cykle postupne generuje súradnice dopadu elektrónu na fotografickú platňu, vloží ich do zásobníka *pozicieElektronov* a prekreslí *canvas*.

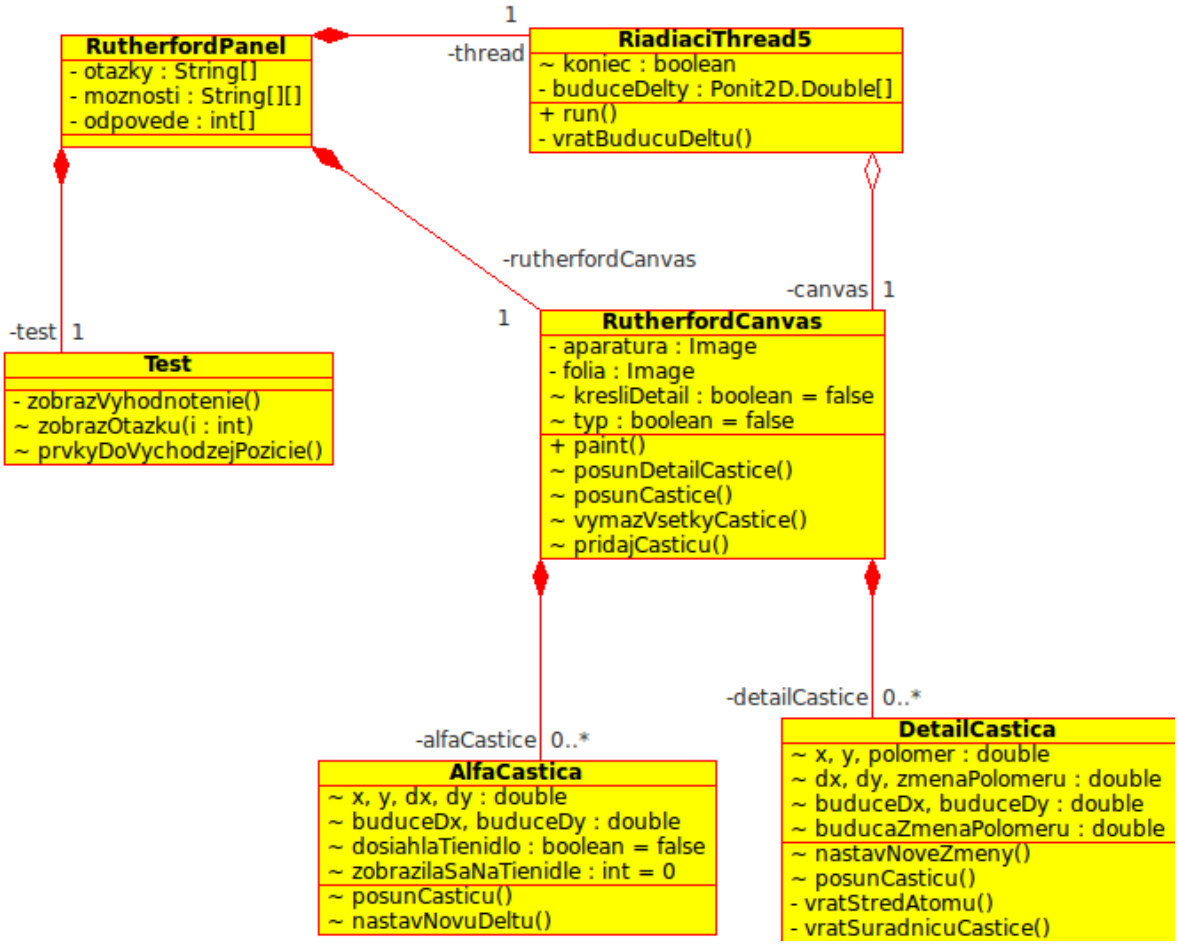
*DoubleSlitCanvas* podľa toho, či je premenná *uvodnaAnimacia* nastavená na *true* alebo *false* buď vyčistí plochu *Imageu* a nakreslí doň obdĺžnik podľa nastavenej dĺžky strany, alebo do *Imageu* vykresľuje bodky, ktorých súradnice vyberá zo zásobníka *pozicieElektronov*. Robí tak, pokiaľ nevyprázdni zásobník. Nakoniec vykreslí *Image*.

*DoubleSlitAparatura* si pamätá celočíselnú premennú *koniecUseciek* a zoznam polomerov kružníc. *RiadiaciThread2* v cykle zväčšuje premennú *koniecUseciek*, a ak je nastavená premenná *DoubleSlitAparatury animujKruznice*, tak postupne pridáva do zoznamu polomerov nové polomery, zväčšuje tie, čo tam sú a maže tie, ktoré dosiahli príliš veľkú veľkosť. Ak po prvýkrát nejaký polomer dosiahne určenú veľkosť, zobudí *RiadiaciThread3*. Na konci každej iterácie cyklu prekreslí *DoubleSlitAparaturu*.

*DoubleSlitAparatura* na každé prekreslenie vykoná nasledovné. Ak má nastavenú premennú *animujUsecky* na *true*, tak vykreslí úsečky podľa premennej *koniecUseciek*, a ak má nastavenú premennú *animujKruznice* na *true*, tak vykreslí kružnice podľa zoznamu polomerov.

### 4.3.3 Komponent Rutherfordov experiment

Verejná trieda *RutherfordPanel* obsahuje ovládacie prvky simulácie, *canvas* na vykreslenie simulácie a riadiaci *thread* simulácie. Ovládacími prvkami sa spustí *thread* v jednom alebo druhom režime. *Thread* vykresľuje simuláciu podľa zvoleného režimu ( buď očakávaný priebeh experimentu alebo skutočný). Ak vykresľuje skutočný priebeh, tak sa dá pomocou tlačidla poslať *threadu* signál, aby vykreslil aj detail zlatej fólie.




*Obr 9. Diagram tried komponentu Rutherfordov experiment.*

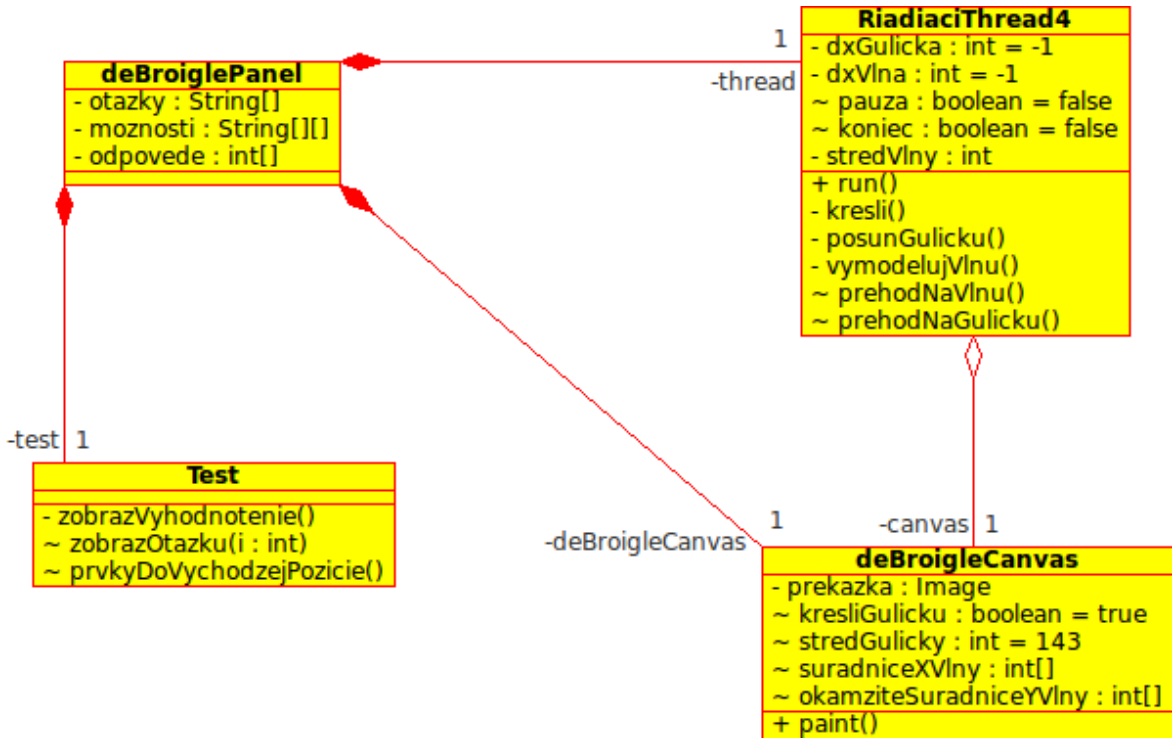
**Spôsob animácie**

*Canvas* si pamätá množinu α častíc a množinu tzv. „detail“ častíc. Riadiaci *thread* potom počas svojho behu v cykle robí nasledovné. V pravidelných intervaloch pridá novú α časticu do množiny, posunie α častice, a ak sa má kresliť detail zlatej fólie, tak posunie i „detail“ častice. „Detail“ častice sa do množiny pridávajú pri posúvaní α častíc. Ak sa má kresliť detail fólie a α častica dosiahla isté miesto v *canvase*, tak sa pridá nová „detail“ častica do množiny. Na konci iterácie cyklu sa prekreslí *canvas*.

*Canvas* pri každom prekreslení vykreslí α častice a ak sa má kresliť detail fólie, tak vykreslí obrázok fólie a všetky „detail“ častice.

### 4.3.4 Komponent de Broglieho hypotéza

Verejná trieda *deBroiglePanel* obsahuje ovládacie prvky simulácie, *canvas* na vykreslenie simulácie a riadiaci *thread* simulácie. Ovládacími prvkami sa spúšťa simulácia. *Thread* vykresľuje animáciu. Ovládacími prvkami sa môže dať *threadu* signál aby animáciu pozastavil, alebo aby zmenil mód vykresľovania. ( kreslí buď guľôčku, alebo vlnový balík)




*Obr. 10. Diagram tried komponentu de Broglieho hypotéza.*

**Spôsob animácie**

*Canvas* si pamätá logickú premennú *kresliGulicku* a súradnice potrebné na vykreslenie guľôčky a vlnového balíka. Riadiaci *thread* v cykle postupne podľa toho, či je nastavená premenná *kresliGulicku* na *true* alebo *false*, buď posúva guľôčku, alebo posúva vlnový balík. Na konci iterácie cyklu prekreslí *canvas*.

*Canvas* sa pri každom prekreslení rozhodne podľa logickej premennej *kresliGulicku*, či vykreslí guľôčku podľa daných súradníc, alebo či vykreslí lomenú čiaru pomocou daných súradníc, ktorá reprezentuje vlnový balík.

### 4.3.5 Trieda Test

Test je trieda, ktorá je potomkom triedy *JFrame*. Je navrhnutá tak, aby postupne zobrazovala otázky a štyri možné odpovede na ne a po zodpovedaní všetkých otázok ponúkla ich vyhodnotenie.

Túto triedu obsahuje každý komponent. Pri vytváraní inštancie sa testu pošlú otázky, možné odpovede na ne a zoznam správnych odpovedí. Verejná trieda každého komponentu obsahuje tlačidlo, ktorým sa okno testu zobrazuje. Po jeho zatvorení okno zmizne z plochy a uvoľnia sa zdroje, ktoré používalo, avšak inštancia stále zostane a po opätovnom stlačení tlačidla sa znovu zobrazí. Nevytvára sa teda zakaždým nová inštancia testu.

## 4.4 Používateľské rozhranie

V Hlavnom okne aplikácie v študentskom móde sú štyri tlačidlá na prepínanie sa medzi jednotlivými simuláciami, panel, v ktorom sa zobrazujú jednotlivé simulácie a *scrollovateľné* textové pole, do ktorého sa zobrazuje popis jednotlivých simulácií. Ten sa načítava z HTML súboru. Okrem toho sa pri textovom poli nachádza tlačidlo, ktorým sa dá pole zväčšiť a opätovne zasa zmenšiť. Každá simulácia poskytuje vlastné používateľské rozhranie na svoje ovládanie. Jeho opis sme načrtli vyššie, nebudeme ho tu preto opäť uvádzať. Aplikácia spustená v učiteľskom móde obsahuje navyše ešte jedno tlačidlo, ktorým sa spúšťa a ukončuje výučba.

Testovacie okno obsahuje textové pole, do ktorého sa vypisujú otázky a vyhodnotenie testu, skupinu štyroch *radiobuttonov*, ktoré predstavujú možnosti odpovede na danú otázku a tlačidlo, pomocou ktorého sa posúva na ďalšiu otázku a zobrazuje sa vyhodnotenie. Keď používateľ zodpovie všetky otázky a nechá si zobraziť vyhodnotenie, tak všetky ovládacie prvky okrem textového poľa zmiznú, samotné textové pole sa zväčší na celé okno testu a zobrazí sa v ňom vyhodnotenie.

# 5 Implementácia

V tejto kapitole sme opisovali, na aké problémy sme narazili počas programovania aplikácie a ako sme ich vyriešili a opísali sme spôsob, akým bola riešená sieťová komunikácia

## 5.1 Sieťová komunikácia

Učiteľská aplikácia ( aplikácia spustená v učiteľskom móde) komunikuje po sieti inak ako študentská aplikácia ( aplikácia spustená v študentskom móde). Učiteľská aplikácia vystupuje ako server, na ktorý sa pripájajú klienti, teda študentské aplikácie. Okrem toho učiteľská aplikácia posiela počas výučby správy pripojeným študentským aplikáciám. Podľa týchto správ vedia študentské aplikácie, aké akcie vykonáva učiteľská aplikácia.

Učiteľská aplikácia využíva triedu *SocketThread*, ktorá vytvorí inštanciu triedy *ServerSocket*. Tá potom čaká na prichádzajúce žiadosti o spojenie. Využíva aj triedu *Zapisovac*, pomocou ktorej sa pripojeným študentským aplikáciám posielajú správy.

Študentská aplikácia využíva triedu *PrihlasovaciThread*, ktorá sa snaží pripojiť na učiteľskú aplikáciu ( snaží sa vytvoriť inštanciu triedy *Socket*). Ak sa mu nepodarí prihlásiť, pretože je neznámy *host* ( nevie určiť IP adresu počítača podľa jeho názvu), tak sa už ďalej nepokúša prihlásiť. Ak sa mu iba nepodarí pripojiť na počítač, tak to po troch sekundách skúsi znovu. Po úspešnom pripojení sa vytvorí a spustí inštancia triedy *PrijimaciThread*, inštancia triedy *PrihlasovaciThread* počká na jej dobehnutie. *PrijimaciThread* prijíma správy z učiteľskej aplikácie a posúva ich hlavnému oknu na spracovanie. To spracuje správy, ktoré spracovať vie, ostatné posunie práve zobrazenému experimentu, ktorý ich spracuje. Keď sa spojenie s učiteľskou aplikáciou preruší, *PrijimaciThread* dobehne a znovu sa aktivizuje *PrihlasovaciThread*, ktorý sa opätovne snaží pripojiť k učiteľskej aplikácii.

Ak sa študent pripojí k učiteľovmu počítaču počas prebiehajúcej výučby, tak on zatiaľ nebude vyučovaný. Pripojí sa k vyučovaniu, keď učiteľ prejde na vysvetľovanie ďalšieho experimentu, alebo až počas ďalšej výučby ( teda keď sa vypne vyučovanie študentov a potom sa za nejaký čas opäť zapne). Aplikácia sa správa takto preto, lebo ak by sme chceli, aby sa študent mohol okamžite pripojiť k prebiehajúcej výučbe, tak by si to

vyžadovalo, aby sme v okamihu pripojenia študenta synchronizovali stav animácie na jeho obrazovke so stavom na obrazovke učiteľa. Potrebovali by sme teda prenášať po sieti nielen učiteľove akcie, ale aj stavy jednotlivých experimentov, čo by si vyžadovalo zložitejší komunikačný protokol, ale hlavne zásah do riešenia vizualizácie jednotlivých experimentov.

## 5.2 Problémy pri implementácii

Prvý problém sa vyskytol pri programovaní simulácie dvojštrbinového experimentu. Keďže počet dopadajúcich častíc za sekundu sa dá nastaviť až na tisíc, tak pri priamočiarom prístupe, kedy sa vykresľuje každá častica pri každom prekreslení, sa napr. po päťdesiatich sekundách muselo na jedno prekreslenie vykresliť päťdesiattisíc častíc. To spôsobilo, že animácia začala „sekať“. Riešením bolo, že sa na začiatku animovania získal *offscreen image* kresliacej plochy a do neho sa zakaždým iba vykreslili nové častice ( tie, ktoré práve dopadli) a tento *image* sa potom vykreslil na plochu. Toto podstatne zlepšilo animáciu, aj keď jemné „sekanie“ sa v závislosti od výkonu počítača stále môže objaviť.

Ďalší problém sa objavil pri programovaní animácie vlnového balíka. Vlnový balík sa dal programovať podľa istých zložitých matematických funkcií a ich vývoja v čase, konkrétne pomocou Fourierovho radu, ale to by nás stálo veľmi veľa času. Rozhodli sme sa preto naprogramovať vlnový balík iba približne, aby výsledná animácia aspoň pripomínala, ako by sa správal vlnový balík naprogramovaný striktne podľa matematických funkcií. Dlho sa nám však nedarilo vymyslieť rozumne vyzerajúci spôsob odrazu balíka od prekážky. Napokon sme zvolili jednoduchú metódu, kedy sme si balík zaznamenali ako postupnosť bodov. Každý bod mal svoju X-ovú súradnicu a Y-ovú súradnicu, ktorá sa vypočítala ako nulová súradnica y plus výchylka. Bodom sme postupne menili X-ové súradnice, teda balík sa pohyboval. Pri prekážke sa potom balík správal tak, že od prekážky sa odrazil stred balíka a výchylky bodov, ktoré zmizli v prekážke, sa opačne pripočítali k Y-ovým súradniciam bodov zrkadlovo k nim podľa prekážky. Takýto odraz potom iba sčasti pripomína odraz, aký by sme dosiahli pri matematických funkciách, ale na druhej strane je plynulý a pre naše účely postačujúci.

Ďalší problém sa vyskytol pri programovaní uzatvárania aplikácie spustenej v študentskom móde. Pri zatváraní okna je totiž nutné, aby sme ukončili bežiace *thready*, inak sa síce okno stratí z obrazovky, ale aplikácia v skutočnosti stále beží ďalej. Aplikácia v študentskom móde má bežiaci *PrihlasovaciThread*. Ak sa nám podarilo pripojiť

k učiteľovej aplikácii, tak potom *PrihlasovaciThread* čaká na dobehnutie *PrijimaciehoThreadu*. Vtedy stačí, ak zavrieme *socket* a nastavíme *PrihlasovaciemuThreadu* premennú *koniec* na *true*. To spôsobí dobehnutie *PrijimaciehoThreadu* a *PrihlasovaciThread* vie, že má skončiť, tak dobehne tiež. Ak však nie je nadviazané spojenie, tak my zavrieme *socket* a povieme *PrihlasovaciemuThreadu,* aby skončil, jemu sa vtedy podarí prihlásiť, otvorí nový *socket* a spustí nový *PrijimaciThread* a aplikácia teda neukončí svoj beh. Riešením bola synchronizácia. V synchronizovanej sekcii v *PrihlasovacomThreade* museli byť overenie, či má *thread* skončiť, a pokus o spojenie ( vytvorenie nového *socketu*). V synchronizovanej sekcii, ktorá sa vykonávala pri zatváraní okna, zase muselo byť zatvorenie *socketu* a nastavenie premennej *koniec* na *true*. Samozrejme, synchronizovať sa muselo na tom istom objekte.

# 6 Nasadenie a použitie

## 6.1 Používanie aplikácie

Adresár s aplikáciou si skopírujete do počítača. Pre spustenie aplikácie musíte mať nainštalovaný Java Runtime Environment ( JRE) minimálne verziu 6. Spustením súboru „FyzikaMikrosveta.jar“ sa spustí aplikácia v študentskom móde. Po spustení sa aplikácia v tomto móde pýta na IP adresu učiteľovho počítača. Ak ju nechcete pri každom spustení zadávať, treba si vytvoriť novú premennú prostredia „IP_CASTICE“ a nastaviť jej hodnotu na adresu, ktorú chcete používať. Ak chcete spustiť aplikáciu v učiteľskom móde, môžete ju spustiť z príkazového riadka s argumentom „ucitel“, alebo si vytvoriť premennú prostredia „CASTICE_UCITEL“, na jej hodnote nezáleží.

## 6.2 Nasadenie v praxi

Súčasťou testovania aplikácie bolo aj jej nasadenie v praxi. Konkrétne išlo o otestovanie použiteľnosti aplikácie študentmi druhého a štvrtého ročníka gymnázia na Novohradskej ulici. Odskúšanie prebiehalo v rámci jedného krúžku a aplikáciu skúšalo sedem študentov. Ich úlohou bolo samostatne si odskúšať program bez úvodnej inštruktáže k nemu. Iba ak sa nás na niečo pýtali, tak sme im pomohli. Veľkú pomoc však nepotrebovali. Na záver vyplnili dotazník.

Z vyplnených dotazníkov vyplynulo, že ovládanie programu sa im páčilo, väčšine sa páčil aj spôsob, akým boli v programe robené testy. I keď viacerí poznali niektoré simulované javy, všetci jednotne odpovedali, že sa naučili niečo nové. Niektorí sa vyjadrili aj k tomu, čo sa im v programe nepáčilo. Niektoré z týchto pripomienok boli ešte zapracované do programu.

Z tohto usudzujeme, že program sa dá používať, avšak určite by sa jeho používaním odhalili ďalšie nedostatky, ktoré by bolo dobré potom odstrániť, aby aplikácia lepšie plnila svoj účel.

# Záver

Naším cieľom bolo navrhnúť a naprogramovať výukový softvér, ktorý by simuloval niektoré fyzikálne javy, a tým by vysvetľoval určité fakty z fyziky mikrosveta. Konkrétne máme na mysli fotoelektrický jav, pri ktorého vysvetlení sa použila myšlienka, že svetlo sa skladá z jednotlivých energetických kvánt ( fotónov), Rutherfordov experiment, ktorý viedol k objaveniu atómového jadra, dvojštrbinový experiment, ktorý ukazuje časticové aj vlnové vlastnosti mikročastíc a je jedným zo základných experimentov využívaných pri štúdiu mikrosveta, a animáciu faktu, že analógiou voľnej mikročastice je vlnový balík.

Náš cieľ sa nám podarilo splniť. Vytvorili sme aplikáciu, ktorá simuluje zvolené fyzikálne javy, umožňuje študentovi nastavovať rôzne parametre, a tým zasahovať do priebehu simulácií. Podáva k simuláciám vysvetľujúci popis a ponúka možnosť preveriť si svoje znalosti v teste. Ponúka učiteľovi možnosť použiť ju ako nástroj pri výučbe. Vtedy učiteľ i žiaci v počítačovej učebni majú aplikáciu spustenú na svojich počítačoch a k študentom sa prenášajú akcie z učiteľovho počítača a vykonávajú sa. Vidia tak na monitore, čo robí učiteľ na tom svojom.

Dúfame, že aplikácia by mohla prispieť k lepšiemu pochopeniu niektorých zákonitostí záhadného sveta mikročastíc. Vyskúšali sme ju aj v praxi. Odskúšali ju študenti gymnázia na Novohradskej ulici. Tí nám vyplnili dotazník. Z neho vyplynulo, že niektoré veci sa im páčili, ale mali i svoje pripomienky. Niektoré z nich sa nám ešte podarilo zapracovať do aplikácie.

Námetom na ďalšiu prácu by bolo nasadiť program do praxe a zo skúseností zistiť, čo by bolo dobré zmeniť, alebo urobiť inak. Ak by to pomohlo pri vyučovaní, dal by sa ďalej vylepšiť komunikačný protokol a jednotlivé komponenty simulácií tak, aby umožňovali pripojenie sa študenta do už začatej výučby experimentu. Prípadne by sa dali možno doprogramovať aj ďalšie experimenty.

# Použitá literatúra


[Oat10] Pišút, J. – Zajac, R. 2010. *O atómoch a kvantovaní*. [ online]. Bratislava: Knižničné a edičné centrum FMFI UK, 2010. http://www.fmph.uniba.sk/fileadmin/user_upload/editors/sluzby/kniznica/el_materialy/svet/o_atomoch.pdf

[JLS3] Gosling J. a kol. 2005. *The Java™ Language Specification Third Edition*. [ online].

http://java.sun.com/docs/books/jls/download/langspec-3.0.pdf

[SMA] Law, A. M., *Simulation Modeling and Analysis*

[TOM] Tomcsányi, P. *Komunikácia medzi procesmi.* [online], dostupné 31. 5. 2011

http://edi.fmph.uniba.sk/~tomcsanyi/07.pdf

[SOC] *The Java Tutorials, Lesson: All about sockets*, citované 31. 5. 2011

http://download.oracle.com/javase/tutorial/networking/sockets/index.html

[API] *Java SE 6 API Documentation*, dostupné 31. 5. 2011

http://download.oracle.com/javase/6/docs/api/

[SWI] *The Java Tutorials, Lesson: Using Swing Components*, dostupné 31. 5. 2011

http://download.oracle.com/javase/tutorial/uiswing/components/index.html