Univerzita Komenského v Bratislava

Fakulta Matematiky, Fyziky a Informatiky

BAKALÁRSKA PRÁCA

BRATISLAVA 2008 RICHARD MEČÍŘ

Výučbový program na tvorbu jednoduchých elektrických obvodov

BAKALÁRSKA PRÁCA

Richard Mečíř

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY
KATEDRA INFORMATIKY

Informatika

Vedúci bakalárskej práce
RNDr. Andrej Blaho

BRATISLAVA 2008

Zadanie bakalárskej práce:

Mojím cieľom bude vypracovať záverečnú prácu s názvom: *Výučbový program na tvorbu jednoduchých elektrických obvodov* pod vedením *RNDr. Andreja Blaha* z pracoviska *KAI*.

Cieľ bakalárskej práce:

naprogramovať aplikáciu na tvorbu elektrických obvodov, ktorá bude umožňovať simulovať všetky obvody, obsiahnuté v učive stredných škôl

V Bratislave 10.2.2008

Čestne prehlasujem, že som túto bakalársku prácu vypracoval samostatne, iba s použitím uvedených informačných zdrojov.

V Bratislave 2008 .....

podpis

Chcem sa poďakovať vedúcemu svojej bakalárskej práce RNDr. Andrejovi Blahovi za jeho cenné rady a hodnotné pripomienky a všetkým tým, ktorí svojou radou prispeli k úspešnému dokončeniu tejto práce.

1) **ABSTRAKT**

V dnešnej dobe nájdeme na internete množstvo programov, ktoré sa venujú elektrickým obvodom. Napriek tomu je väčšina z nich pre bežného stredoškoláka nepoužiteľná, či už je to kvôli ich zložitosti, zlej funkcionalite alebo z dôvodu, že sa daný obvod dotyčným programom nedá navrhnúť. Preto sa snažím navrhnúť program, ktorý bude priamo vychádzať zo stredoškolského učiva a zároveň bude dostatočne jednoduchý pre každého stredoškoláka. Aplikácia je takisto využiteľná aj učiteľmi, ktorí ňou môžu prezentovať zapájanie súčiastok do jednoduchých elektrických obvodov.

Kľúčové slová: elektrický obvod, elektrická súčiastka, prúd, napätie, odpor

OBSAH

2) **ÚVOD**

Počítačové simulácie elektrických obvodov nám dávajú možnosť vytvárať a testovať použitie všetkých elektrických komponentov, a to bez rizika poškodenia súčiastok alebo ohrozenia vlastného zdravia. Elektrická energia už od svojho objavenia poskytuje jedinečné možnosti a v dnešnej dobe si už bez nej tak ľahko neporadíme. Navrhnúť elektrický obvod tak, aby fungoval, a aby robil to, čo od neho očakávame, by malo patriť k základným vedomostiam každého, kto by sa chcel vytváraniu obvodov venovať či už na laickej alebo profesionálnej úrovni. Treba mať na pamäti, že elektrická energia nie je hračka a pri neodbornej manipulácii môže spôsobiť zranenie, alebo dokonca až smrť. Úplným základom je nutnosť uvedomenia si, že každý obvod na svoju funkčnosť potrebuje byť uzavretý, a to tak, aby mohla elektrická energia prúdiť od kladného pólu zdroja k zápornému. Taktiež si treba uvedomiť, že nie každá súčiastka vydrží plné napätie zdroja a pri prekročení určitej hranice je vysoké riziko, že dotyčná súčiastka zhorí a obvod prestane fungovať. Preto boli navrhnuté jednoduché elektrické súčiastky – rezistory, ktoré dokážu napätie znížiť na prijateľnú úroveň. Po pochopení týchto jednoduchých vlastností obvodu je každý pripravený zapájať svoje vlastné elektrické obvody a taktiež je pripravený skúšať pripájať ďalšie komponenty, ktoré pridávajú obvodu nové vlastnosti. Týmto postupom sa dá prepracovať od jednoduchých obvodov, ktoré rozsvecujú žiarovky, až po profesionálne obvody, ktoré môžeme nájsť v každom elektrickom spotrebiči, ale napríklad aj v mestskom osvetlení, semaforoch a vo všetkom, čo využíva na svoj chod elektrickú energiu.

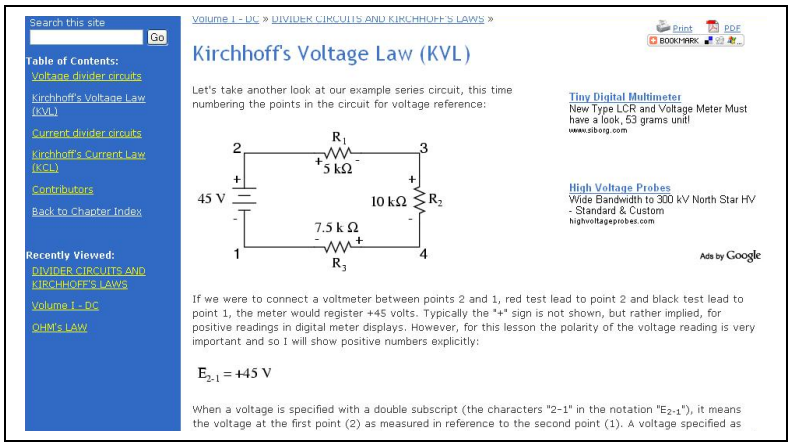
Za cieľ svojej práce som si dal vytvoriť program, pomocou ktorého si môže každý navrhnúť vlastné obvody, nastaviť si vlastnosti každej súčiastky obvodu a nakoniec simulovať, ako bude tento obvod prebiehať. Zložitosť obvodov bude odvodená od stredoškolskej fyziky, čiže bude možné zapájať komponenty, ako sú spínače, žiarovky, rezistory, ale aj diódy a tranzistory. Na začiatku bude plocha prázdna, čiže rozloženie celého obvodu bude na používateľa. Po úspešnom zapojení celého obvodu sa časť, ktorou bude prechádzať prúd zafarbí na zeleno a v tejto časti bude možné zistiť prúd, odpor a napätie v každom uzlovom bode obvodu, ktorý bude závisieť od použitých súčiastok, od zložitosti obvodu a od nastavenia veličín na jednotlivých súčiastkach. Celú túto simuláciu bude možné za behu ľubovoľne meniť

tým, že zmeníme fyzikálne veličiny niektorého komponentu, čo nám umožní pozorovať následné správanie obvodu a vplyv našej zmeny na zvyšok obvodu.

3) **VÝCHODISKÁ**

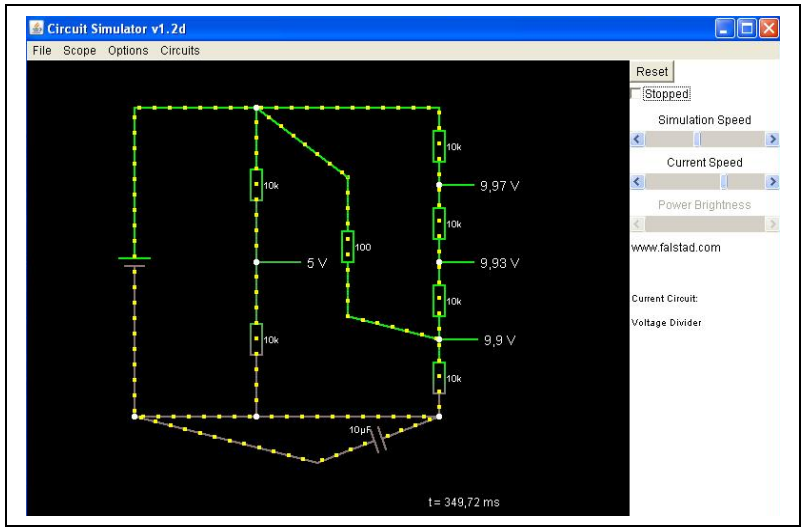
Základnú inšpiráciu pri tvorbe tejto práce poskytovali už hotové programy a stránky, ktoré sa zaoberajú danou problematikou.

Stránka *All about circuits* (Všetko o obvodoch - http://www.allaboutcircuits.com) poskytuje veľké množstvo informácii o elektrických obvodoch – sú tu spomenuté samotné základy elektriny, Ohmove a Kirchhoffove zákony, informácie o sériovom a paralelnom zapojení obvodu, výpočty prúdu a napätia pri sériovom alebo paralelnom zapojení a veľa ďalších užitočných informácií. Všetky údaje sú tu dopodrobna popísané na vysokoškolskej úrovni, čo poskytuje hlbší prehľad o probléme, ako tomu bolo v prípade stredoškolskej fyziky. Toto ale môže byť pre stredoškoláka zložité, preto by som túto stránku na výučbu pre stredoškolákov neodporúčal, ale na účely môjho projektu je vhodná, keďže metódy, akými môj program pracuje používateľ neuvidí a nemusí sa nimi zaťažovať.

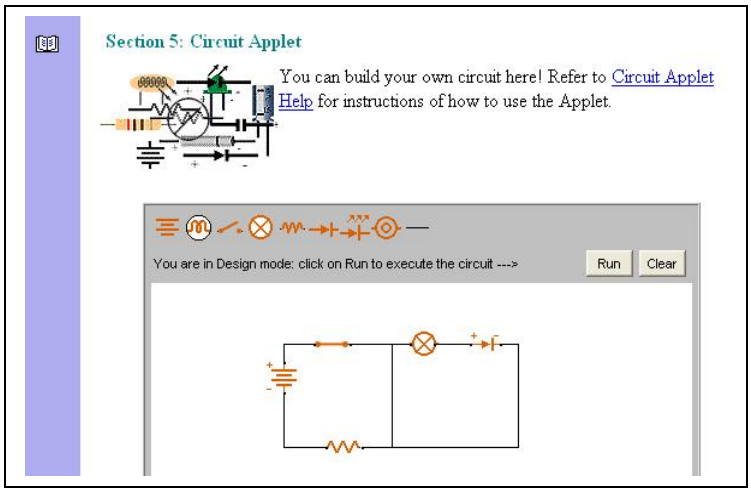


Veľkou inšpiráciou pre mňa bol java applet Circuit Simulator v1.2d, ktorý umožňuje vykresľovať asi ľubovoľne zložitý obvod a ráta veľkosti elektrických veličín. Tento applet je vypracovaný na profesionálnej úrovni a je určený pre skúsených užívateľov, keďže jeho náročnosť je pomerne vysoká. Taktiež sa skôr venuje zložitým schémam a elektrickým javom, z čoho vyplýva, že na experimentovanie so stredoškolskou fyzikou je nevhodný. Napriek tomu používa zaujímavo riešené užívateľské prostredie, konečné schémy vyzerajú pútavo a využitie rôznych farieb pre časti pod prúdom a bez prúdu je taktiež dobrá myšlienka. Spôsob ukladania komponentov, pri ktorom sa komponenty ukladajú na plochu bez obmedzení sa mi zdá veľmi vhodný, keďže sa dá zostrojiť ľubovoľne zložitý obvod, preto som sa

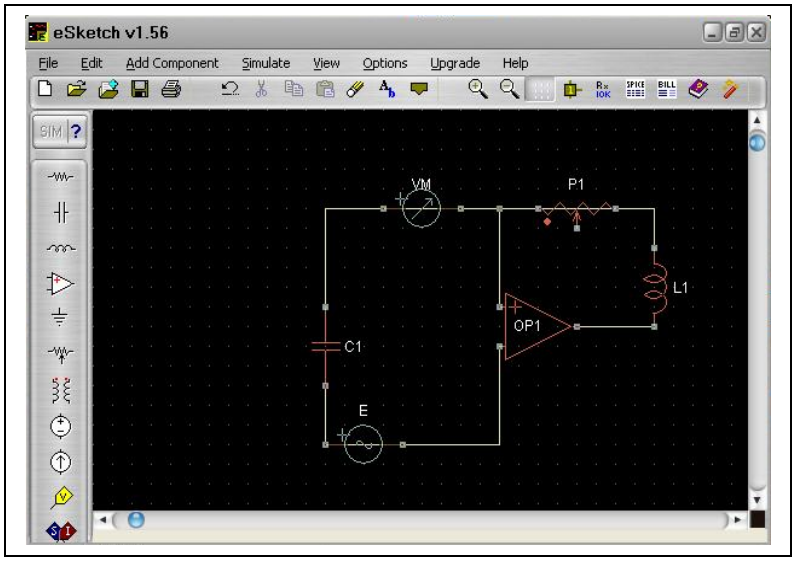
rozhodol vo svojom programe použiť rovnaký spôsob, ktorý som ešte rozšíril o podmienku, aby sa komponenty nemohli vzájomne prekrývať, keďže vzájomné prekrývanie považujem za neprehľadný, a niekedy je problém zistiť, ako sú vlastne komponenty v obvode pospájané. Ďalšia dobrá vlastnosť, ktorú som sa rozhodol prevziať, je možnosť meniť fyzikálne veličiny komponentu priamo v obvode, čiže stačí na komponent kliknúť, zmeniť hodnotu nejakej veličiny a hneď sa celá schéma prepočíta podľa nových parametrov. Je to oveľa rýchlejšie a pohodlnejšie, ako keby po pridaní komponentu už neboli možné zmeny jeho parametrov.



Ďalšou zaujímavou inšpiráciou pre mňa bola stránka: http://library.thinkquest.org/10796/ch15/ch15.htm, ktorá obsahuje popis niektorých základných elektrických súčiastok, ich význam a použitie. Všetky informácie sú podávané formou prednášky a po každej kapitole je niekoľko príkladov, kde sa dá preskúšať obsiahnuté učivo. Okrem toho táto stránka obsahuje jednoduchý java applet, ktorý umožňuje poskladať si vlastný obvod a otestovať si vyššie uvedené súčiastky v praxi. Celý applet je zameraný hlavne na prácu s diódami a žiarovkami, ale pri troche trpezlivosti sa dajú poskladať aj zložitejšie paralelné obvody. Hlavný prínos z tejto stránky bola pre mňa funkčnosť súčiastok, ako pracujú a aké fyzikálne veličiny ich ovplyvňujú a akým spôsobom.



Medzi zaujímavé programy, ktoré sa dajú voľne stiahnuť z internetu patrí eSketch. Tento program umožňuje vytvárať zložitejšie obvody a následne odskúšať ich funkčnosť, doplniť komentáre a nakoniec aj ich tlač. Ukladanie komponentov na plochu je intuitívne, celý program má vlastnú nápovedu. Nevýhodou je, že program je tvorený na tvorbu zložitejších obvodov a niektoré komponenty (napríklad spínač, žiarovku a iné) obsiahnuté v stredoškolskom učive ani neobsahuje. Hlavným prínosom z tohto programu bolo menu, z ktorého sa dajú jednoducho ukladať komponenty na plochu, pričom používateľ dopredu vidí schematické označenie a nemusí skúšať, čo presne sa pod neznámym názvom nachádza.



Ďalšou stránkou, ktorá obsahuje obrovské množstvo informácii o elektrických obvodoch, je Wikipedia. Na tejto stránke sa dajú nájsť informácie prakticky o čomkoľvek, čiže je to veľmi užitočná stránka pri akomkoľvek projekte. Vo svojom projekte som ju využil pri popise jednotlivých elektrických súčiastok, pri tvorení ich

schématických značiek, ale aj kvôli ďalším informáciám o fyzikálnych zákonoch v obvode, pri riešení matíc a podobne.

4) NÁVRH PROGRAMU

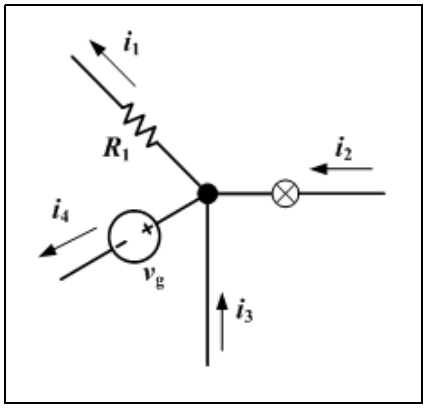
Cieľom mojej práce je implementovať program, ktorý bude mať nasledovnú funkcionalitu:

Po spustení program zobrazí prázdnu plochu, na ktorú bude možné umiestňovať rozličné elektrické komponenty. Zoznam bude zahrňovať minimálne nasledovné: zdroj, rezistor, žiarovka, spínač, dióda, tranzistor, uzol. Každý z komponentov bude mať niekoľko (zvyčajne dva) konektory, na ktoré bude možné pripájať vodiče. Týmto spôsobom bude možné zrealizovať ľubovoľný obvod. Komponenty a aj vodiče bude možné presúvať a mazať. Tiež im bude možné nastavovať fyzikálne parametre v závislosti od komponentu (v prípade rezistora je to odpor, v prípade zdroja napätie atp.)

Najdôležitejšou funkciou programu bude schopnosť simulovať obvod na základe fyzikálnych zákonov a bude umožňovať zistiť správanie obvodu. To zahŕňa schopnosť zistiť veľkosť prúdu a elektrického potenciálu v jednotlivých miestach obvodu. Taktiež bude možné do istej miery simuláciu ovplyvňovať – zapínanie a vypínanie spínačov atp.

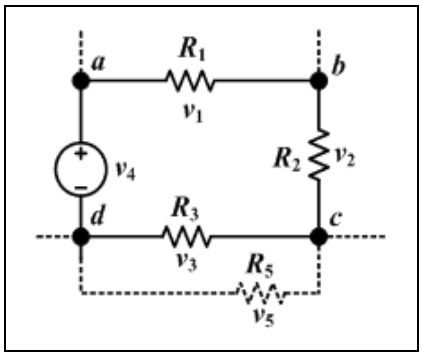
Veľmi dôležitou časťou návrhu sú fyzikálne zákony ktorými sa musí celý obvod riadiť, aby bola garantovaná správnosť všetkých vykonávaných výpočtov. Medzi najdôležitejšie patria Kirchhoffove zákony, z ktorých pomocou sa dajú realizovať všetky potrebné výpočty. Tieto zákony udávajú vzťahy pre prúdy a napätia v uzloch a slučkách elektrického obvodu. Sú nevyhnutné pri riešení elektrických obvodov.

Prvý Kirchhoffov zákon: Algebraický súčet prúdov v ktoromkoľvek uzle elektrického obvodu sa rovná nule. Súčet prúdov, ktoré do uzla vstupujú, sa rovná súčtu prúdov, ktoré z uzla vystupujú.



Pre zobrazený uzol tu platí: $i_1 + i_4 = i_2 + i_3$

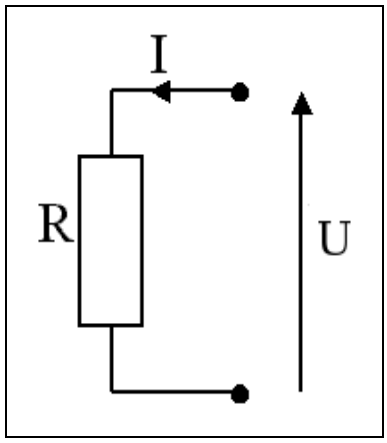
Druhý Kirchhoffov zákon: Súčet svorkových napätí prvkov elektrického obvodu v ľubovoľnej slučke sa rovná nule.



Pre zobrazenú slučku tu platí: $v_1 + v_2 + v_3 + v_4 = 0$

Ďalším dôležitým zákonom je Ohmov zákon, ktorý definuje vzájomný vzťah medzi elektrickým prúdom, elektrickým odporom a elektrickým napätím.

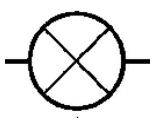
Ohmov zákon: Elektrický prúd pretekajúci vodičom je priamo úmerný rozdielu elektrických napätí na koncoch vodiča a nepriamo úmerný elektrickému odporu medzi koncami vodiča. V matematickom zápise platí U = I x R (U – napätie, R – odpor, I – prúd). Tento zákon platí pre každý obvod, ktorý je zložený z lineárnych prvkov, ktorými preteká jednosmerný prúd a za ideálnych podmienok – konštantná teplota, konštantný odpor, ideálne prívodné vodiče.



Elektrická schéma obvodu demonštrujúca vzťah medzi napätím, odporom a prúdom.

Celá aplikácia bude používať nasledovné elektrické súčiastky:

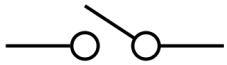
Žiarovka je druh elektrického svetelného zdroja, v ktorom sa na premenu elektrickej energie na svetelnú energiu využívajú tepelné účinky elektrického prúdu pri prechode tuhým telesom a žiarivé vlastnosti tohto telesa.



Elektrický zdroj je súhrnný názov pre elektrické zariadenia, medzi ktorých dvoma rozličnými časťami (pólmi) je aj po pripojení elektrického obvodu (tzv. záťaže) udržovaný rozdiel elektrických potenciálov alebo napätia, pričom dodávajú do obvodu elektrický prúd. Jedná sa tým vlastne o zdroj elektrickej energie.

Rezistor je lineárny elektronický prvok, ktorého prevažujúca vlastnosť je jeho elektrický odpor. Vyskytuje sa buď ako súčasť integrovaného obvodu, alebo ako samostatná elektronická súčiastka.

Spínač alebo **vypínač** je mechanická, mechanicko-elektrická, alebo elektronická súčiastka, slúžiaca na spojenie a rozpojenie okruhu. Princíp je rovnaký - úlohou spínača je zopnúť (prepnúť, prerušiť) tok média vstupnej vetvy do výstupnej vetvy. Najobvyklejšou aplikáciou je spínanie a rozpínanie elektrického okruhu, existujú aj spínače hydraulické, tlakovzdušné a pod.

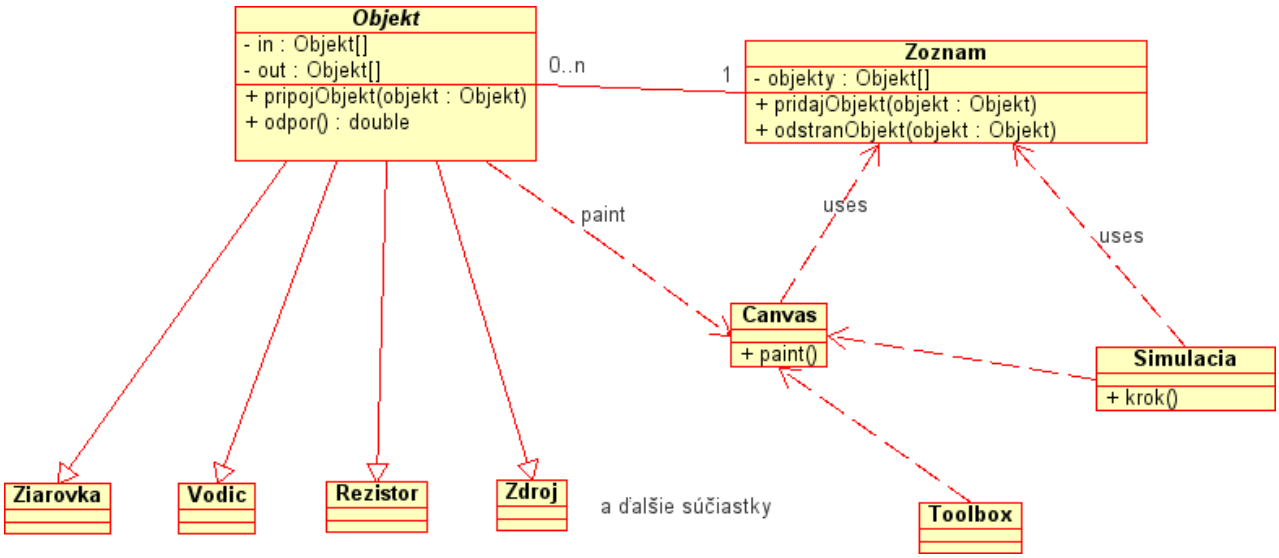


Dióda je elektronická súčiastka s dvomi elektródami. Obvykle slúži k usmerňovaniu prúdu, ale existujú i inak špecializované diódy. Okrem vákuovej diódy (elektrónky) - dnes už používanej len zriedkavo vo veľmi špecifických zariadeniach - sú diódy vyrobené z polovodičov (obvykle kremíka) a využívajú polovodičový PN priechod.

Tranzistor je polovodičová súčiastka, používaná ako zosilňovač, spínač, stabilizátor a modulátor elektrického napätia alebo prúdu.

5) NÁVRH IMPLEMENTÁCIE

V tejto časti sa budem podrobnejšie venovať prípravám na tvorbu svojho programu – návrhu tried a príprave fyzikálnych častí programu. Pri návrhu svojho programu som sa opieral o objektovo orientované programovanie s využitím jazyka Java. Najskôr som vytvoril štyri základné triedy, ktoré tvoria jadro celého programu. Sú to triedy Objekt, Zoznam, Simulácia a Canvas.



Základnou triedou, ktorá ma na starosti vlastnosti súčiastok je abstraktná trieda **Objekt**. Táto trieda vytvára základné metódy pre vykresľovanie, nastavenie súradníc, nastavenie fyzikálnych veličín jednotlivým súčiastkam. Podľa tejto triedy sú implementované triedy jednotlivých súčiastok (Zdroj, Rezistor, ...) podľa umiestnenia a nastavení od používateľa.

Keďže je veľmi dôležité, aby sme mali neustále k dispozícií informácie o každej súčiastke, ktorú sme použili, je nutné implementovať triedu **Zoznam**. Táto trieda obsahuje spájaný zoznam (LinkedList) všetkých použitých súčiastok a umožňuje k nim ľubovoľný prístup a ich následné ďalšie použitie.

Na vykresľovanie obvodov na obrazovku sa používa trieda **Canvas**, ktorá vykresľuje všetky súčiastky tam, kde boli umiestnené.

Na všetky výpočty a celú funkčnosť programu po fyzikálnej stránke slúži trieda **Simulácia**, ktorá obsahuje všetky fyzikálne výpočty a riadi funkčnosť obvodu po fyzikálnej stránke.

Na to, aby program fungoval, nestačia len samotné triedy, ale je treba rozhodnúť, ako sa budú triedy navzájom ovplyvňovať, čiže je nutné navrhnúť vzťahy medzi triedami tak, aby program fungoval tak, ako je požadované.

Základný vzťah musí byť medzi triedami Zoznam a Objekt, keďže je nutné, aby sa každá použitá súčiastka automaticky pridala do zoznamu súčiastok. Vynechanie ľubovoľnej súčiastky pokazí celú funkčnosť, preto je tento vzťah veľmi dôležitý.

Vzťah medzi triedami objekt a triedami jednotlivých súčiastok je zjavný, keďže trieda objekt je abstraktná a súčiastky ju musia nutne implementovať.

Trieda canvas využíva na svoju funkčnosť triedy zoznam a objekt tak, že pomocou triedy zoznam si zistí, aké súčiastky už sú použité a pomocou vykresľujúcej metódy, ktorú dokáže použiť každá súčiastka, dané súčiastky vykreslí na plochu. Triedu canvas takisto využíva aj trieda toolbox, ktorá má za úlohu vykresliť tlačítka na pridávanie nových súčiastok.

Trieda simulácia potrebuje na svoj beh zoznam komponentov (triedu zoznam), ako aj rozloženie komponentov na ploche (triedu canvas). Na základe informácii od týchto tried dokáže spustiť simulujúci algoritmus na výpočet všetkých potrebných fyzikálnych elektrických veličín.

Simulujúci algoritmus: Simulujeme obvod po časových úsekoch o dĺžke jednej milisekundy a v každom kroku simulácie prevedieme obvod do grafu, rozdelíme graf na jednotlivé vetvy a okruhy a za pomoci Kirchhoffových zákonov vytvoríme sústavu rovníc popisujúcich daný obvod (vzťah medzi odporom, napätím a prúdom). Z rovníc následne vytvoríme maticu, ktorú vyriešime pomocou Gaussovej eliminácie, čím získame veľkosti prúdu v jednotlivých vetvách obvodu. V každom kroku treba tieto

kroky neustále opakovať, keďže obvod môže byť zmenený – spínač môže byť zapojený alebo odpojený a tak podobne.

Na správne simulovanie zložitejších súčiastok (kondenzátor, dióda, tranzistor), potrebujeme okrem prúdu poznať aj hodnotu elektrického potenciálu v každom mieste obvodu, preto následne na základe známych veľkostí prúdov a odporov, určíme veľkosť elektrického potenciálu v každom mieste obvodu.

6) **REALIZÁCIA**

V tejto časti opíšem samotnú realizáciu svojho výučbového programu – ako jednotlivé použité triedy fungujú, ako spolupracujú s ďalšími použitými triedami a ako celý program používa stredoškolskú fyziku.

Základom programu je trieda **MainWindow**, ktorá reprezentuje hlavné okno aplikácie. Táto trieda má na starosti vytvorenie užívateľského rozhrania – toolbar, menu a interakciu s grafickou plochou.

Popis dôležitých metód triedy MainWindow:

- makeNavigationButton – táto funkcia vytvorí tlačidlá v toolbare na základe zadaných parametrov.
- createToolbar – táto funkcia vygeneruje všetky tlačidlá v toolbare, každé tlačidlo reprezentuje jednu súčiastku.
- actionPerformed – volá sa pri stlačení tlačidla toolbaru, prepne aktuálnu súčiastku.
- main – hlavná funkcia programu, vytvorí inštanciu triedy MainWindow. Statická metóda.
- obsluha tlačidiel myši

Ďalšou triedou je trieda **AComponent**, je to abstraktná trieda, ktorá reprezentuje jeden všeobecný komponent. Jej potomkovia implementujú jednotlivé konkrétne komponenty. Každý komponent má jeden alebo viac konektorov, na ktoré je možné pripájať vodiče.

```
boolean hit (int x, int y) {
     return ((x>=surX)&&(x<=x2()))&&((y>=surY)&&(y<=y2()));
 }

boolean hits (AComponent c) {
    boolean hitx = false;
    boolean hity = false;
    if((x()>=c.x()) && (x()<=c.x2())) hitx = true;
    if((x2()>=c.x()) && (x2()<=c.x2())) hitx = true;
    if((c.x()>=x()) && (c.x()<=x2())) hitx = true;
    if((c.x2()>=x()) && (c.x2()<=x2())) hitx = true;
    if((y()>=c.y()) && (y()<=c.y2())) hity = true;
    if((y2()>=c.y()) && (y2()<=c.y2())) hity = true;
    if((c.y()>=y()) && (c.y()<=y2())) hity = true;
    if((c.y2()>=y()) && (c.y2()<=y2())) hity = true;
    if (hitx && hity) return true;
    else return false;
}
```

Popis dôležitých metód triedy AComponent:

- draw – nakreslí komponent na grafickú plochu. Abstraktná metóda.
- nastav_suradnice – nastaví súradnice komponentu.
- x, y – vráti súradnice komponentu.
- sirka, vyska – vrátia rozmery daného komponentu. Abstraktné metódy.
- hit – zistí, či bod je súčasťou daného komponentu. Používa sa na zistenie, či používateľ klikol na danom komponente.
- hits – zistí, či sa tento komponent prekrýva z komponentom, ktorý dostane ako parameter. Používa sa na zabránenie prekrývania sa komponentov.
- hrany – zoznam prepojení medzi jednotlivými konektormi vo vnútri komponentu. Väčšina komponentov má dva konektory a jedno prepojenie medzi nimi. Výnimkou je tranzistor.
- napatie, odpor – vrátia generované napätie, resp. odpor komponentu medzi zadanými dvoma konektormi.

Ďalšou významnou je trieda **Wire**, ktorá reprezentuje jeden vodič. Vodič je identifikovaný počiatočným a koncovým komponentom a číslom konektora na každom z týchto komponentov.

Ďalšia je trieda **Component_List**, ktorá spravuje zoznam všetkých použitých komponentov. Keďže všetky komponenty dedia od triedy AComponent, je možné využiť polymorfizmus a všetky komponenty spravovať v jednom zozname. Na uchovávanie zoznamu komponentov a vodičov sa používa štandardná javovská trieda LinkedList.

```
public boolean hit_sth(int x, int y){
    AComponent c = null;
    for (ListIterator<AComponent> it = zoznam().listIterator(); it.hasNext();) {
        c = it.next();
        if (c != null) {
            if(c.hit (x, y)){
                return true;
            }
        }
    }
    return false;
}

public boolean hit_comp(AComponent s){
    AComponent c = null;
    for (ListIterator<AComponent> it = zoznam().listIterator(); it.hasNext();) {
        c = it.next();
        if ((c != null)&&(c != s)) {
            if(c.hits (s)){
                return true;
            }
        }
    }
    return false;
}
```

Popis dôležitých metód triedy Component_List:

- add – pridá nový komponent.
- add_vodic – pridá nový vodič.
- hit_sth – zistí, či sa na daných súradniciach nachádza nejaký komponent.
- hit_comp – zistí, či sa zadaný komponent prekrýva s nejakým iným komponentom.
- del – zmaže komponent.
- del_vodic – zmaže vodič.

Ďalšou triedou je **CompArea**, ktorá reprezentuje hlavnú grafickú plochu aplikácie. Dedí väčšinu svojej funkcionality od štandardnej javovskej triedy java.awt.Canvas. Interakciu s užívateľom realizuje trieda MainWindow, trieda CompArea má na starosti iba kreslenie.

```
public void paint (Graphics g) {
    MainWindow okno = MainWindow.okno;
    if (okno == null) return;
    Component_List zoznam = okno.zoznam();
    AComponent c = null;
    for (ListIterator<AComponent> it = zoznam.zoznam().listIterator(); it.hasNext();) {
        c = it.next();
        if (c != null) c.Draw(g);
    }
    for (ListIterator<Wire> it = zoznam.vodice().listIterator(); it.hasNext();) {
        Wire w = it.next();
        if (w != null) {
            Point p1 = w.comp1.connector(w.con1);
            Point p2 = w.comp2.connector(w.con2);
            g.drawLine(p1.x, p1.y, p2.x, p2.y);
        }
    }
    if(okno.Selected() != null){
        Point p = okno.Selected().connector(okno.Selected_Connector());
        g.setColor(Color.red);
        g.fillOval(p.x-5, p.y-5, 11, 11);
    }
}
```

Popis dôležitých metód triedy CompArea:

- paint – vykreslí všetky komponenty a vodiče daného obvodu. Komponenty sa vykresľujú použitím abstraktnej metódy AComponent.paint, preto je možné jednoducho pridávať nové komponenty bez nutnosti zmeny tejto triedy. K prekresľovaniu dochádza pri každej manipulácii s komponentmi a vodičmi, čím je možné realizovať interaktívne presúvanie komponentov a podobne.

Poslednou dôležitou triedou je trieda **Graph**, ktorá slúži na simuláciu obvodu. Použitý algoritmus je popísaný vyššie.

```
private class Vrchol{
    AComponent c;
    int i; //Ktory vrchol daneho komponentu
    LinkedList<Hrana> hrany;
    int num;
};

private class Hrana{
    Vrchol v1, v2;
    Cesta c = null;
};

private class Cyklus{
    LinkedList<Cesta> cesty = new LinkedList<Cesta>();
};

private class Cesta{
    Vrchol v1, v2;
    //zoznam hran chyba!!!
    boolean ozn = false;
    int cislo;
    LinkedList<Cyklus> cykly = new LinkedList<Cyklus>();
};
```

```
private LinkedList<Vrchol> vrcholy = new LinkedList<Vrchol>();
private LinkedList<Hrana> hrany = new LinkedList<Hrana>();
private LinkedList<Cesta> cesty = new LinkedList<Cesta>();
private LinkedList<Cyklus> cykly = new LinkedList<Cyklus>();
private int ciest = 0, vrcholov = 0, riadkovMatice = 0;

private TreeMap<Integer, TreeMap<Integer, Integer> > matica = new TreeMap<Integer, TreeMap<Integer, Integer> >();

private int matGet (int col, int row) {
    Integer x = new Integer (col);
    Integer y = new Integer (row);
    if (!matica.containsKey(x)) return 0;
    Integer res = matica.get(x).get(y);
    return (res == null) ? 0 : res.intValue();
}

private void matSet (int col, int row, int val) {
    Integer x = new Integer (col);
    Integer y = new Integer (row);
    Integer v = new Integer (val);
    if (!matica.containsKey(x))
        matica.put(x, new TreeMap<Integer, Integer>());
    matica.get(x).put (y, v);
}
```

Popis dôležitých metód triedy Graph:

- simulacia – jeden krok simulácie, využíva ostatné metódy tejto triedy.
- generate – prvý krok simulácie. Vygeneruje z obvodu graf, pričom vrcholmi grafu sú konektory komponentov a hranami grafu sú vodiče a vnútorné prepojenia medzi konektormi v rámci komponentu. Jednotlivé prvky grafu obsahujú prepojenie na pôvodné konektory a vodiče.
- prechod – táto metóda nájde všetky cesty v grafe. Cesta je postupnosť vrcholov taká, že vrcholy sú navzájom pospájané. Vnútorné vrcholy majú stupeň dva a okrajové majú stupeň buď iný ako dva, alebo sa počiatočný vrchol rovná koncovému – slučka. Pre každú cestu bude platiť, že prúd v rámci cesty je rovnaký.
- generuj_cykly – metóda nájde zoznam cyklov v obvode taký, ktorý pokrýva všetky cesty v obvode, ktoré sú súčasťou nejakého cyklu. Pri hľadaní cyklov sa minimalizuje redundantnosť tým, že každý novo pridaný cyklus obsahuje aspoň jednu cestu takú, ktorá nie je súčasťou žiadneho cyklu. Táto metóda využíva backtracking a rekurziu.
- matice – táto metóda na základe zoznamu ciest a cyklov vygeneruje maticu obvodu. Využíva pri tom prvý a druhý Kirchhoffov zákon. Matice sú reprezentované dvojrozmerným asociatívnym poľom – využíva sa štandardná javovská trieda TreeMap.
- ries_maticu – táto metóda vyrieši maticu obvodu použitím Gaussovej eliminácie.

- urci_prudy – vezme hodnoty prúdov z vyriešenej matice a priradí ich jednotlivým komponentom v obvode.
- urci_potencialy – na základe známych hodnôt elektrického prúdu a odporu vypočíta elektrický potenciál v každom mieste obvodu. Využíva rekurziu a backtracking.

7) MOŽNOSTI ROZŠÍRENIA

Možnými rozšíreniami programu sú:

- pridávanie nových elektrických súčiastok do programu. Keďže všetky komponenty dedia od spoločnej triedy AComponent, nový komponent je možné jednoducho pridať dedením od tejto triedy a pridaním nového tlačidla do toolbaru.
- výpočet ďalších fyzikálnych veličín – napríklad impedancia.
- možnosť ukladania obvodov do súboru a ich následné načítanie.
- lepšie možnosti editácie – krok späť, kopírovanie a podobne.

8) **INŠTALÁCIA**

Program vyžaduje na svoj chod behové prostredie jazyka java (Java Runtime Environment).

9) **ZÁVER**

Program na tvorbu jednoduchých elektrických obvodov pre stredné školy môže nepochybne pomôcť pri vyučovaní mnohým učiteľom tým, že budú môcť názorne ukázať rôzne aplikácie preberaného učiva v praxi. Zároveň pomôže žiakom pri riešení rôznych úloh, keďže si môžu daný obvod v počítači vytvoriť a zistiť jeho správanie pri rôznych fyzikálnych veličinách. Keďže schematické značky jednotlivých komponentov sú vo všetkých krajinách rovnaké, môžu môj program využívať žiaci a učitelia stredných škôl na celom svete.

Keďže program využíva na svoj chod jazyk java, je teoreticky možné upraviť kód do takej miery, aby sa dal program spustiť na každom zariadení, ktoré dokáže javu použiť – napríklad aj mobilné telefóny. Toto zaujímavé rozšírenie by umožňovalo žiakom pracovať s obvodmi na bežnej hodine fyziky, kedy nemajú k dispozícii počítač.

Navyše sa pokúsim zabezpečiť, aby bol tento program voľne stiahnuteľný z internetu, čím si ho v prípade záujmu bude môcť úplne každý veľmi jednoducho stiahnuť.

Vytvorením tohto programu sa mi podarilo úspešne splniť cieľ svojej práce. Bola to pre mňa cenná skúsenosť, ktorá mi určite v budúcnosti veľmi pomôže pri tvorbe ďalších javovských (ale aj iných) programov. Taktiež dúfam, že je moje dielo dostatočne zaujímavé na to, aby slúžilo ako inšpirácia a zdroj cenných informácii pre ďalších ľudí, ktorí sa rozhodnú naprogramovať podobnú aplikáciu.

10)POUŽITÁ LITERATÚRA

Všetky uvedené informačné zdroje boli v júni 2008 funkčné a obsahovali popísaný obsah.

1) < http://www.allaboutcircuits.com/ >
- stránka popisujúca elektrické obvody, zákony a vzťahy v elektrických obvodoch

2) < http://www.falstad.com/circuit/index.html>
- javovský applet simulujúci zložité elektrické obvody

3) <http://library.thinkquest.org/10796/ch15/ch15.htm>
- stránka popisujúca elektrické súčiastky, obvody a veľa ďalšieho

4) <http://www.schematica.com/esketch_files/eSketch.htm>
- program eSketch, simulátor elektrických obvodov

5) <http://www.aaroncake.net/Circuits/>
- stránka znázorňujúca rôzne zložité obvody z praxe

6) <http://www.gymspmkr.edu.sk/informatika/JavaTutorial/>
- stránka obsahujúca tutoriály k jave, množstvo informácii

7) <http://java.sun.com/>
- stránka s množstvom tutoriálov, dokumentácií, taktiež sa tu dá stiahnuť behové prostredie jazyka java (Java Runtime Environment) a veľa ďalšieho