Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

# Bakalářská práce

# Vytvoření pomocného programového vybavení pro výuku předmětu KPG

Plzeň, 2007 Miloš Buršík

## Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 23. srpna 2007 Miloš Buršík


## Poděkování

Na tomto místě bych rád poděkoval vedoucí své bakalářské práce Doc. Dr. Ing. Ivaně Kolingerové za odborné rady a náměty, které zásadním způsobem přispěly k dokončení této práce.

**Abstract**

**Title:** Creation auxiliary software for the KPG course

**Abstract:** This bachelor thesis was created to prepare educational materials for computer graphic designers and help to the students at the beinning of the Beauty Computer Graphics (KPG) course. Thesis deals with basics of computer graphics programming in four programming languages, in C++, C#, Delphi, Java and informs us about benefits of these languages. The main part of the thesis is a set of programs in these programming languages which demonstrate the use of graphical commands and components. Last chapter of the bachelor thesis is devoted to creation new web pages for the course.

# Obsah

# Seznam obrázků

# 1 Úvod

Počítačová grafika je zajímavým a důležitým oborem informatiky. Využívá počítačů na syntetické vytváření umělých snímků a také na úpravu zobrazitelných informací nasnímaných z reálného světa. V dnešní době již snad není odvětví, kde by se grafika nevyužívala. Počítačová grafika se převážně dělí trojrozměrnou (3D) a dvojrozměrnou (2D) počítačovou grafiku.

3D grafika se převážně využívá v počítačových hrách, ve filmech a v televizi na tvorbu animaci a speciálních efektů, v neposlední řadě se také využívá pro inženýrské, nebo lékařské účely na tvorbu reálných modelů.

2D počítačová grafika vznikla se vznikem CRT obrazovek, což byl velmi důležitý základ grafiky. Dvojrozměrná grafika nachází použití při kreslení matematických funkcí, fraktálů, grafů, úpravě digitálních fotografií, návrhu obalů knih atd. Například i tento text, který teď čtete, je dílem počítačové grafiky.

## 1.1 Cíl práce

Cílem této práce bylo vytvořit pomocné materiály pro výuku předmětu Krásy počítačové grafiky, který je zaměřen převážně na výuku dvojrozměrné grafiky. A tím studentům, kteří přicházejí ze středních škol, nebo studentům vysokých škol, kteří nemají zkušenosti s počítačovou grafikou, dodat praktické zkušenosti a rychlé návody, jak začít.

Základem této práce je jak vytvoření pomocných textů, které demonstrují inicializaci grafiky a použití základních komponent pro práci s vektorovou a rastrovou grafikou, tak i vytvoření několika programů, které ukazují použití dvojrozměrné grafiky v praxi ve čtyrech programovacích jazycích: C++, C#, Delphi, Java.

V práci jsou objasněny základní druhy grafických primitiv a jejich statické použití nebo náhodné vykreslení. Dále jsou objasněny události klávesnice, myši a jejich užití při kreslení primitiv a druhů fontů písem. V neposlední řadě je zde vysvětleno použití bitmapové grafiky, například načítání obrázků z disku a jejich editace. V práci by také měli být vidět odlišnosti, někdy až záludnosti různých programovacích jazyků.

Souhrnně by tato práce měla sloužit jako doplněk výuky, ale především by měla sloužit studentům při jejich samostudiu a k poznání základních věcí, které se již do přednášek nepodařilo vtěsnat, stejně tak by mohla sloužit i všem ostatním zájemcům, kteří chtějí poznat, na jakých základech je stavěna počítačová grafika.

# 2 Programovací jazyky

V této kapitole se zaměříme na použité programovací jazyky.

## 2.1 C++

Jazyk C++ je objektově orientovaný programovací jazyk, který vyvinul Bjarne Soustroup v Bellových laboratořích rozšířením tehdy populárního programovacího jazyka C. Původní název jazyka byl C with Classes (C s třídami) a až později, v roce 1983 byl jazyk přejmenován na C++. Tento název má zdůrazňovat evoluční povahu změn oproti jazyku C, proto „++“, jako operátor inkrementu v C. Jazyk C je až na několik jasně definovaných výjimek podmnožinou jazyka C++, takže se téměř nemusíme obávat chyb při kompilaci zdrojových souborů jazyka C kompilátorem C++. Částečnou kompatibilitou a podporou objektového programování se tento jazyk velmi rychle dostal na výsluní a stal se jedním z nejoblíbenějších a nejvyužívanějších programovacích jazyků, alespoň do nástupu nových modernějších programovacích jazyků, jako je Java a C-Sharp.

### 2.1.1 Výhody C++

- Kompilátory jazyka C i C++ lze najít v mnoha operačních systémech. Například v Linuxu se standardně setkáte s GCC(The GNU Compiler Collection).

- Snadná přenositelnost programů C++, napsaných v normě ANSI, mezi různými operačními systémy a platformami.

- Vysoká efektivita vykonávání kódu (rychlost dobře napsaných programů v C++ se téměř vyrovná stejnému programu napsanému v assembleru).

- Možnost stažení několika volně šiřitelných překladačů a vývojových prostředí jazyka C++ pro prostředí MS Windows , například:

  - Open Watcom C/C++
    URL: http://www.openwatcom.org
  - Bloodshed Dev-C++
    URL: www.bloodshed.net/devcpp.html
  - LCC
    URL: http://www.cs.virginia.edu/~lcc-win32
  - Digital Mars C, C++ and D Compilers
    URL: http://www.digitalmars.com
  - Eclipse SDK & C/C++ Development Tools
    URL: http://www.eclipse.org/cdt

- Po naučení C++, bude pro vás mnohem jednodušší naučit se třeba C-Sharp, Delphi nebo další jazyky stavějící na podobném objektovém základu.

## 2.2 C#

C-Sharp je nový vysokoúrovňový objektově orientovaný programovací jazyk vyvinutý firmou Microsoft zároveň s platformou .NET Framework. Microsoft .NET znamená novou generaci systému vývoje aplikací pro operační systémy Windows založeném na řízeném běhovém prostředí, obohaceném o sadu základních tříd, nesoucím jméno .NET Framework. Stručně řečeno .NET Framework je prostředím, ve kterém se vyvíjejí aplikace, které pak ke svému spuštění potřebují mít nainstalované knihovny z tohoto prostředí. C# vznikl roku 2002 a je založen na programovacích jazycích C++ a Java. Z jazyka C++ čerpá především syntaxi a z Javy ostatní, především princip řízených běhových prostředí. Při převodu zdrojového kódu do kódu strojového přidává

ještě jednu vrstvu. Tuto vrstvu představuje mezikód MSIL(Microsoft Intermediate Language), do kterého jsou zdrojové kódy zkompilovány, a tento mezikód je na cílové platformě (Windows, Linux) převeden do strojového kódu pomocí CLR (Common Language Runtime), který je součástí .NET Frameworku. V prostředí CLR má k pomoci Garbage Collector, který pomocí složitých algoritmů se stará o přidělování a uvolňování operační paměti aplikaci. Tím programátorům odpadá starost o přiřazování paměti a tak usnadňuje práci.

### 2.2.1 Výhody C#

- Jednoduchý, moderní objektově orientovaný programovací jazyk.
- Odstranění nekompatibility jednotlivých programovacích jazyků a s tím související obtížná spolupráce mezi programy a knihovnami napsanými v odlišných jazycích (např. C++ a Visual Basic).
- Vysoká efektivita vykonávání kódu o proti jiným vysokoúrovňovým programovacím jazykům (Javě). Při dobře napsaném programu maximálně o 10% pomalejší než C++, což je v podstatě zanedbatelné.
- Možnost tvorby webových dynamických aplikací, označováno jako ASP .NET.
- Možnost stažení volně šiřitelného plnohodnotného integrovaného vývojového prostředí jazyka C# pro prostředí MS Windows

  - SharpDevelop 2.0.
    URL: http://www.icsharpcode.net/OpenSource/SD

- Blizkost k Javě, takže když se naučíte C-Sharp, tak by pro vás Java neměla být problémem, a naopak.

## 2.3 Delphi

První verze Delphi byla vydána firmou Borland v roce 1994, obsahuje integrované grafické vývojové prostředí, určené pro tvorbu aplikací na platformě

MS Windows v jazyce Object Pascal (objektové nástavbě již zastaralého jazyka Turbo Pascal). Delphi obsahuje systém RAD(Rapid Application Development), který umožňuje velmi snadný návrh grafického uživatelského prostředí. Samozřejmostí je, že na základě návrhu uživatelského prostředí námi vytvořeného programu je vygenerována kostra zdrojového kódu. Programováni v Delphi je založeno na použití komponent. Komponenty jsou softwarové stavební díly, pomocí kterých vykonáváme nějakou činnost. Touto činností může být například zobrazeni obrázku, textu, komunikace s databází. Komponentou je ale i například zobrazení tlačítka na formulář atd. Delphi je mezi programátory oblíbeno hlavně kvůli velkému množství knihoven komponent, některé knihovny jsou dokonce volně přístupné, za jiné se musí platit. V Dephi si můžeme vytvořit i vlastní komponenty, jak je v moderních programovacích jazycích dobrým zvykem.

### 2.3.1 Výhody Delphi

- Jednoduchý objektově orientovaný programovací jazyk.
- Podpora databází, což je v dnešní době věc téměř nezbytná.
- Velké množství integrovaných nástrojů a možnost vytvářet vlastní komponenty.
- Možnost stažení verze Delphi Personal, která je zdarma pro nekomerční využití, bohužel ostatní distribuce, jako Delphi Professional, Enterprise a Architect jsou komerční verze za nemalé peníze.
  URL: http://www.codegear.com/downloads/free/delphi
- Velk8 podobnost s Borland C++ Builderem, takže po zvládnutí jednoho z prostředí by s druhým vývojovým prostředím neměl být problém.

## 2.4 Java

Java je objektově orientovaný programovací jazyk vyvinutý firmou Sun Microsystems, která jej a představila v roce 1995. Ideovým předchůdcem Javy je jazyk C, resp. jeho objektová podoba C++. Při vývoji dokázali návrháři a

programátoři eliminovat z původního jazyka C++ spoustu chyb a vznikl tak jazyk „čistší“ se zjednodušenou syntaxí a tak i celkově snadněji zvládnutelný, což vede k menší frekvenci chyb a tak k větší produktivitě. Java je v dnešní době jedním z nejpoužívanějších programovacích jazyků na světě. Obliba Javy hlavně spočívá v její přenositelnosti. Díky přenositelnosti je používána pro programy, které mají pracovat na různých systémech počínaje čipovými kartami(platforma JavaCard), přes mobilní telefony a různá zabudovaná zařízení (platforma Java ME), aplikace pro desktopové počítače (platforma Java SE), až po rozsáhlé distribuované systémy pracující na řadě spolupracujících počítačů rozprostřené po celém světě (platforma Java EE). Přenositelnost je dána zvláštností překladu, jelikož překlad neprobíhá do strojového kódu, ale do pseudojazyka nazývaného byte-code. Tento jazyk je nezávislý na platformě (soubor .class). Tento soubor je pak z disku zaváděn do paměti, současně je ověřen a po ověření je program pomocí interpretu spuštěn.

### 2.4.1 Výhody Javy

- Jednoduchý robustní objektově orientovaný programovací jazyk, který je určen pro psaní vysoce spolehlivého softwaru.
- Java bude dále vyvíjena jako open source (zdarma).
- Nezávislost na platformě (Windows, Linux, Solaris, ...).
- Navržena pro podporu webových aplikaci(Applety, JSP) a síťových aplikací (tvorba klientských aplikací a serverů).
- Přímá podpora v databázích.
- Správa paměti pomocí automatického Garbage collectoru.
- Výkonný programovací jazyk, přestože se jedná o jazyk interpretovaný, ztráta výkonu není významná, neboť překladače překládají do strojového kódu jen kód, který je opravdu zapotřebí.
- Možnost stažení několika volně šiřitelných multiplatformních vývojových prostředí jazyka Java, například:

- Eclipse
  URL: http://www.eclipse.org
- BlueJ
  URL: http://www.bluej.org
- NetBeans
  URL: http://www.netbeans.org

- Java je svojí syntaxí a způsobem objektového návrhu velmi blízká jazyku C#.

# 3 Výukové dokumenty

První úkolem mé bakalářské práce bylo vytvoření dokumentů, které podrobně popisují inicializaci grafiky a základní grafické operace pro studenty, kteří se s počítačovou grafikou, nebo s některým ze čtyř výše uvedených programovacích jazyků ještě nesetkali.

Dokumenty obsahují popis vytvoření projektu a inicializaci grafiky. K vytvoření jsou uvedeny i screenshoty vývojových prostředí a zdrojový kód inicializace grafiky plus vytvoření plátna (ohraničená plocha, kde můžeme kreslit). Dále dokumentují popis základních grafických primitiv a jejich použití, jako nastavení péra a štětce.

Pérem je zamýšleno nastavení vykreslování čar a ohraničování objektů. Bývá reprezentováno objektem *Pen* a je u něj možno nastavit barvu, šířku a styl (tečkování, čerchování, ...).

Štětcem naopak od péra vyplňujeme plochy. Zpravidla je reprezentován objektem *Brush* a můžeme u něj nastavit barvy a styl (šrafování, přechod barev, ...).

Vytvořené dokumenty jsou svázány ve zvláštní příloze a uvedeny na přiloženém CD.

# 4 Řešené ukázkové programy

Programy jsou navrženy ve čtyřech výše jmenovaných programovacích jazycích a to v C++, C#, Delphi a Javě. Cílem těchto programů je pomoci

studentům porozumět základům dvojrozměrné grafiky, a tak objasnit kreslení bodů, čar a dalších primitiv, použití fontů, načítání a ukládání obrázků atd. V neposlední řadě vnést do statického kreslení trochu dynamiky. Tím je zamýšleno objasnit události klávesnice a myši, jejich použití při kreslení a editaci obrázků, nebo použití časování grafických operací či náhody.

Celé zdrojové kódy i spustitelné verze jsou na přiloženém CD.

## 4.1 Program Fonty

### 4.1.1 Popis a ovládaní programu Fonty

Program Fonty, jak již z názvu vyplývá, demonstruje použití fontů počítačových písem. V podstatě jde o kreslení písma na plátno (viz obrázek 1 na straně 11).

Po pravé straně okna si ze seznamu vybíráme font, u kterého comboboxem měníme jeho velikost. Do textového pole zadáváme námi vymyšlený text. Stiskem tlačítka myši na plátně se text na něj trvale vykreslí. Poslední možností, kterou nám program nabízí, je vymazat celé platno, a k tomuto účelu slouží tlačítko *Vymaz*.

### 4.1.2 Postup při programování programu Fonty

Fontů počítačových písem je nepřeberné množství. Abychom zjistili jejich názvy, necháme si celou rodinu písem (Font family) vypsat do některé z výběrových komponent, například do listu, nebo comboboxu. Z těchto komponent si pak snadno vybíráme fonty podle svého vkusu.

Výběr fontu provádíme podle jeho jména, například: Arial, SansSerif, Tahoma, .... Dalšími parametry fontu je jeho velikost a styl. Základní styly fontu jsou tři:

Bold - tučný font

Italic - kurzíva

Plain - prostý font

Při kreslení na plátno využíváme události myši. Událost *mouseMove* a *mouseDown* (v případě Javy událost *mouseMoved* a *mouseClicked*). Událostí *mouseMove* zajistíme, že při pohybu přes plátno překresluje text, zvoleným fontem a velikostí, který jsme zadali do textového pole. Při stisku


Obrázek 1: Program Fonty

tlačítka myši na plátně událost *mouseDown* s pomoci dalších metod vykreslí text trvale na plátno a uloží text a jeho nastavení (pozici, font, velikost) do dynamického seznamu(*ArrayListu, Vectoru, TListu*). Nakonec ještě musíme zajistit správné překreslování okna (plátna), aby se nám okno částečně nemazalo, či nezůstávala vidět jen poslední nakreslená věc. Problém odstraníme cyklem v metodě *paint*, ve kterém vyvoláme uložené hodnoty z dynamického seznamu a opětovně vykreslíme text na plátno.

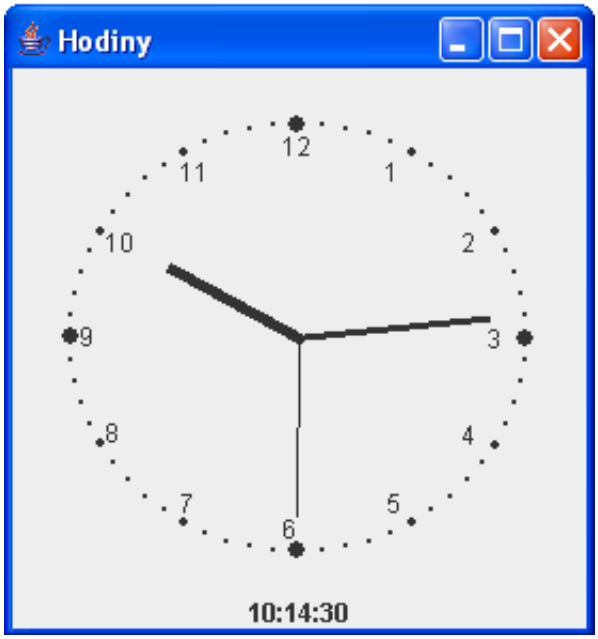
## 4.2 Program Hodiny

### 4.2.1 Popis programu Hodiny

Program Hodiny se zabývá vytvořením analogových hodin s libovolnou velikostí ciferníku (viz obrázek 2 na straně 12).

### 4.2.2 Postup při programování programu Hodiny

Abychom mohli vytvořit analogové hodiny, musíme se nejprve seznámit s časováním v aplikaci. K tomu se používá komponenta *Timer*, kterou nasta-




Obrázek 2: Program Hodiny

víme na hodnotu 1000 milisekund, tedy 1 sekundu. To znamená, že metoda *Timeru* se provede každou vteřinu. V těle této metody si zjistíme systémový čas, který převedeme a rozdělíme do číselných hodnot na hodiny, minuty a sekundy. Dále v této metodě musíme zavolat funkci, která nám zjistí pozice hodinové, minutové a sekundové ručičky. (viz následující metoda v Javě)

```
void SpoctiPolohu(float uhel, float delka){
uhel = (float)(uhel - (Math.PI/2));
x=(int)((Math.cos(uhel)* (int)(width / delka)+(width / 2)));
y=(int)((Math.sin(uhel)* (int)(height / delka)+(height / 2)));
}

//příklad zobrazeni sekund
SpoctiPolohu((float)(2*Math.PI/60*sec), 3.0f);
g2.drawLine(width/2, height/2, x, y);
```

Metoda funguje velmi jednoduše. Nejdříve je od úhlu odečteno 90 stupňů ($\pi/2$) z důvodu, že dvanáctá hodina svírá s horizontem 90 stupňů. Poté je

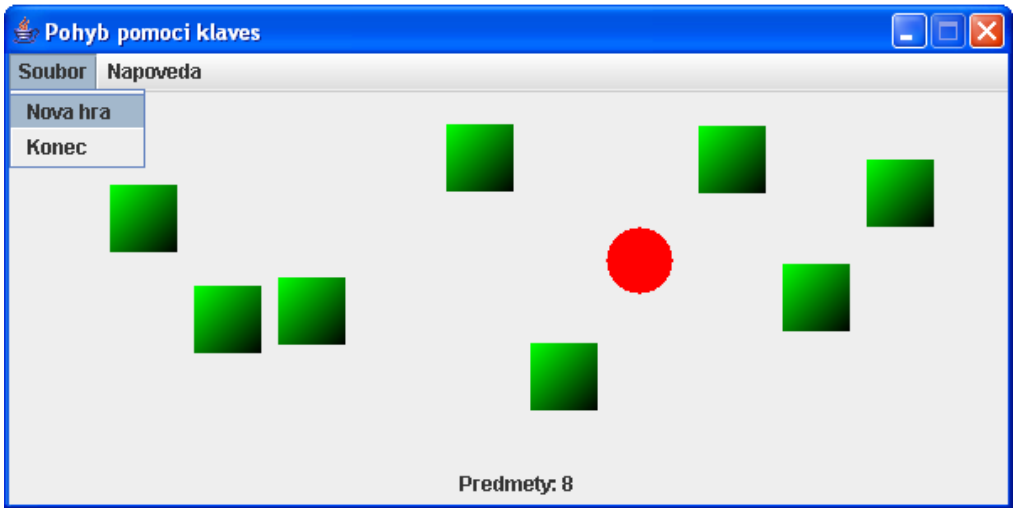
vypočtena souřadnice $x$ a $y$, a to pomocí funkce sin a cos. *Width/delka* určuje délku ručičky a *width/2* přičítá polovinu velikosti okna (střed hodin). Taktéž *height/delka* a *height/2*.

## 4.3 Program Klávesy

### 4.3.1 Popis a ovládání programu Klávesy

Program Klávesy je jakousi jednoduchou počítačovou hrou ve které máme za úkol červeným kruhem sebrat 10 zelených čtverců. (viz obrázek 3 na straně 13)

Program napsaný v programovacím jazyce Java a C# se ovládá směrovými šipkami klávesnice. Program v C++ a Delphi klávesy A D S W. Po sebrání všech 10 předmětů se objeví „vítězná“ hláška. Když si chceme hru zopakovat, v menu *Soubor* položkou *Nova hra* spustíme novou hru.




Obrázek 3: Program Klávesy

### 4.3.2 Postup při programování programu Klávesy

Při startu funkcí *Random*, která generuje náhodná čísla, náhodně rozmístíme zelené čtverce po ploše plátna a uložíme jejich pozice do pole či seznamu. Dále se budeme zabývat událostmi klávesnice. Především událostí *keyPress* (v C#

*keyDown*). Událost *keyPress* se volá po stisknutí jakékoliv klávesy. Při tom si zjistíme, o jakou klávesu se jednalo, a když je to naše směrová klávesa, tak přičteme nebo odečteme krok (zde 5 pixelů) k pozici červeného kruhu a tím zajišťujeme jeho pohyb po plátně. Abychom mohli čtverce sbírat, musíme zjistit, kdy se pozice červeného kruhu překrývá s některým ze zelených čtverců. Na to použijeme následující podmínku, kterou v cyklu kontrolujeme s každým předmětem (čtvercem). Když podmínka zjistí, že se kruh překrývá se čtvercem (předmětem), předmět odstraní ze seznamu a při překreslení okna se předmět smaže z plátna.

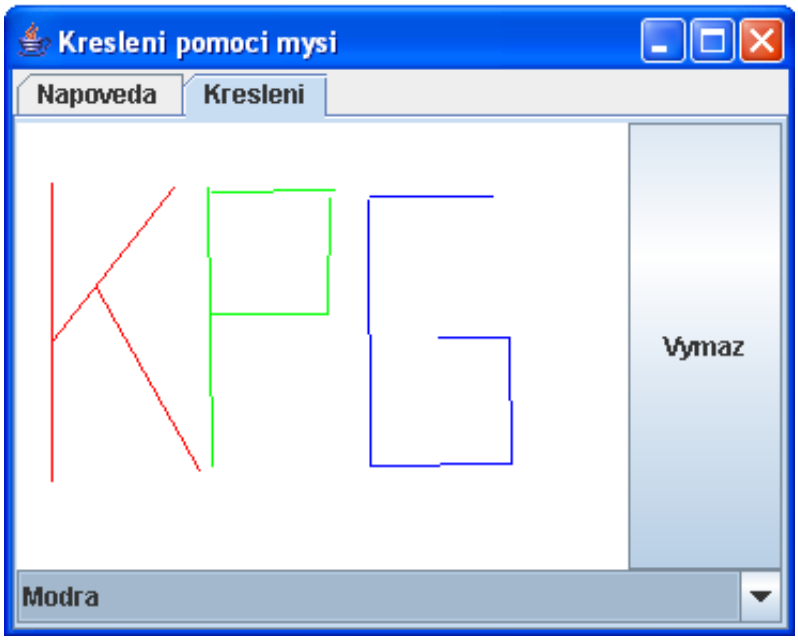
```
/* podmínka v C#, která kontroluje jestli jsme nenarazili
   do jednoho z předmětů
                      sirka - šířka čtverce a kruhu
                      vyska - výška předmětu a kruhu
                   b.X, b.Y - levý horní roh kruhu
        b.X+sirka, b.Y+vyska - pravý dolní roh kruhu
              r[i].X, r[i].Y - levý horní roh předmětu(čtverce)
  r[i].X+sirka,r[i].Y+vyska - pravý dolní roh předmětu
*/
if(((b.X>=r[i].X)      && (b.X<=r[i].X+sirka)       &&
    (b.Y>=r[i].Y)      && (b.Y<=r[i].Y+vyska))       ||
   ((b.X+sirka>=r[i].X)&& (b.X+sirka<=r[i].X+sirka) &&
    (b.Y>=r[i].Y)      && (b.Y<=r[i].Y+vyska))       ||
   ((b.X>=r[i].X)      && (b.X<=r[i].X+sirka)       &&
    (b.Y+vyska>=r[i].Y)&& (b.Y+vyska<=r[i].Y+vyska)) ||
   ((b.X+sirka>=r[i].X)&& (b.X+sirka<=r[i].X+sirka) &&
    (b.Y+vyska>=r[i].Y)&& (b.Y+vyska<=r[i].Y+vyska)))
   {
    v.RemoveAt(i);             //smazání předmětu
    pictureBox1.Invalidate();  //překreslení plátna
   }
```

## 4.4 Program Kreslení myší

### 4.4.1 Popis a ovládání programu Kreslení myší

Program Kreslení myší je primitivní obdoba programu Malování ve Windows, ve které můžeme kreslit úsečky různých barev. (viz obrázek 4 na straně 15)

Přepnutí z panelu nápovědy do panelu kreslení nám umožní kreslení. Stisknutím tlačítka myši, držením a tažením po plátně vykreslíme úsečku. Dále program nabízí vymazání plátna tlačítkem *Vymaz*.




Obrázek 4: Program Kreslení myší

### 4.4.2 Postup při programování programu Kreslení myší

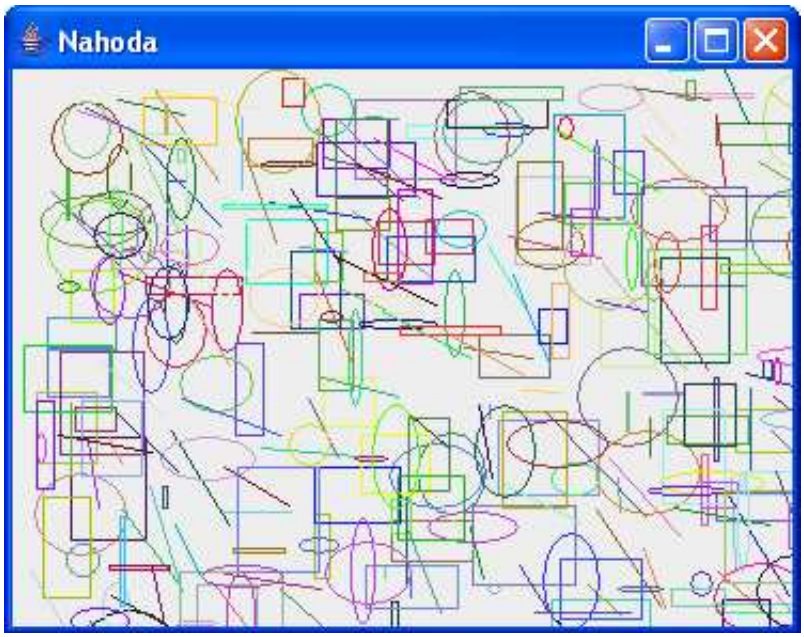
Program je podobně jako program Fonty (viz oddíl 4.1) založen na událostech myši. Při stisku tlačítka myši nebo-li při události *mouseDown* (v Javě *mousePressed*) si uložíme pozici kurzoru. Při držení tlačítka a pohybu myši(událost *mouseMove*, v Javě událost *mouseDragged*) vykresluji úsečku mezi počátečním bodem a aktuálním kurzorem myši. Mezi tím neustále překreslujeme okno a tím vznikne dojem „pružné“ čáry. Při uvolnění tlačítka

(*mouseUp*, v Javě *mouseRelased* ) uložíme počáteční a koncové body do dynamického seznamu (*ArrayListu, Vektoru, TListu*) a nezapomeňme si také uložit barvy, případně tloušťky úseček. Nakonec musíme zajistit správné překreslování okna vyvoláním všech uložených záznamů z dynamického seznamu v metodě *paint*. Následně překreslíme všechny uložené úsečky.

## 4.5 Program Náhoda

### 4.5.1 Popis programu Náhoda

Program Náhoda je jakýsi okenní spořič obrazovky. Do okna se náhodně a na náhodné umístění vykresluje několik set grafických primitiv, která se po vykreslení postupně všechny smažou. Toto probíhá do nekonečna a vzniká tím dojem spořiče obrazovky. (viz obrázek 5 na straně 16)




Obrázek 5: Program Náhoda

### 4.5.2 Postup při programování programu Náhoda

V tomto programu užijeme jako v programu Hodiny (viz oddíl 4.2) časovač (*Timer*), ale nastavíme jej na nějakou menší hodnotu než sekundu, například

na 10 milisekund. Takže každých 10 milisekund vykreslíme na plátno jedno primitivum z několika set. Abychom nevykreslovali pořád stejná primitiva, tak zavedeme funkci *Random*, která nám vygeneruje náhodné číslo. Podle čísla nastavíme primitivum, například:

0 - elipsa
1 - úsečka
2 - obdélník
... - ...

Stejným způsobem zvolíme i barvu a pozici primitiva. Opět jako v několika předchozích příkladech musíme zajistit správné překreslování okna. Takže si čísla primitiv, jejich barvy a pozice uložíme do dynamického seznamu. Následně je v cyklu v metodě *paint* překreslujeme. Toto překreslování se nám hodí i při mazání primitiv, a to tak, že vždy poslední prvek seznamu smažeme a při překreslení okna nám tak primitivum ubyde. Takhle postupujeme až do prázdného dynamického seznamu. Následně jej opět plníme a mažeme do nekonečna a tím zajišťujeme přibývání a ubývání primitiv.

## 4.6 Program Obrázek

### 4.6.1 Popis a ovládaní programu Obrázek

Program Obrázek je jednoduchý grafický editor, ve kterém si můžeme otevřít obrázkový soubor a ten následně upravit kreslením čar (obdoba tužky v programu Malování ve Windows).(viz obrázek 6 na straně 18)

Ovládání programu není složité. Chceme-li otevřít obrázek, v menu *Soubor* položkou *Nacti* zvolíme obrázkový soubor z pevného disku. Na obrázek kreslíme stisknutím a držením tlačítka myši. V dolní části okna si můžeme zvolit barvu a šířku čáry. Tlačítkem *Vymaz* smažeme vše co jsme nakreslili a na načtený obrázek můžeme začít kreslit znovu. Pokus si chceme obrázek s naší malbou uložit, tak opět v menu *Soubor* položkou *Uloz* uložíme obrázek na disk.

### 4.6.2 Postup při programování programu Obrázek

Při programování musíme v první řadě zajistit načtení obrázku na plátno. K načtení obrázku v C# použijeme grafický objekt Bitmap (v Javě *Buffered-*


Obrázek 6: Program Obrazek

*Image*) a do něj příkazy

```
Bitmap bm = (Bitmap) Bitmap.FromFile("soubor.bmp");     // C#
BufferedImage img=ImageIO.read(new File("soubor.bmp"));//Java
Image1->Picture->LoadFromFile("soubor.bmp");            //C++
Image1.Picture.LoadFromFile('soubor.bmp');              //Delphi
```

obrázek načteme a následně zobrazíme na plátno. Jak jste si jistě všimli, tak v C++ a Delphi stačí přímo načíst na komponentu *Image* a máme vše hotovo.

Pro kreslení na bitmapu musíme využít události myši, zejména událost *mouseDown* (v Javě *mousePressed*), kdy při stisku tlačítka uložíme počáteční pozici, a událost *mouseMove* a zajištěním držení tlačítka (v Javě *mouseDragged*), kdy při pohybu myši okamžitě kreslíme, ukládáme čáry do dynamického seznamu a přeměňujeme počáteční pozici čáry za za koncovou. Tím dosáhneme jakoby kreslení tužkou.

Pro uložení obrázku v C++ a Delphi použijeme grafický objekt *TBitmap*. Tento objekt obsahuje metody pro uložení svého obsahu do souboru a proto

do něho zkopírujeme obsah plátna (komponentu *Image*) a potom jej uložíme pomocí metody *SaveToFile*.

V Javě (C#) máme obrázek v instanci třídy *BufferedImage* (v C# *Bitmap*). Jelikož jsme obrázek upravovali prostřednictvím objektu *Graphics* můžeme obrázek snadno uložit metodou *ImageIO.write* (v C# *Save*).

## 4.7 Program Primitiva

### 4.7.1 Popis a ovládaní programu Primitiva

Program Primitiva je ukázkou základních grafických primitiv a jejich vlastností, kterými jsou barva a styl výplně. (viz obrázek 7 na straně 19)

Přepnutím záložky *Napoveda* do záložky *Primitiva* se dostaneme k základním primitivům. V comboboxech si vybíráme primitivum a styl jeho výplně.


Obrázek 7: Program Primitiva

### 4.7.2 Postup při programování programu Primitiva

Při programování si nejprve rozmístíme komponenty (comboboxy) po okně a naplníme je jmény primitiv. Při výběru určitého primitiva z comboboxu si zjistíme index položky primitiva. Podle indexu necháme primitivum v metodě *paint* vykreslit. Obdobně to uděláme i s barvami a styly výplně a máme hotovo.

## 4.8 Program Změna barvy

### 4.8.1 Popis a ovládání programu Změna barvy

Program Změna barvy zaměňuje zvolenou barvu obrázku za barvu bílou.

Při ovládáni programu nehledejme žádné složitosti. Kliknutím myší na libovolnou barvu v obrázku se nám tato barva v celém obrázku změní na barvu bílou (viz obrázek 8 na straně 20), kde jsme klikli do plochy šedivé a hnědé barvy.


Obrázek 8: Program Změna barvy bodu

### 4.8.2 Postup při programování programu Změna barvy

V programu musíme nejdříve načíst obrázkový soubor. K načtení používáme stejných metod, jako v programu Obrázek (viz oddíl 4.6). Následně obrázek zobrazíme na plátno. Kliknutím na obrázek zvolíme barvu, k tomu použijeme událost myši *keyDown* (v Javě *keyPressed* ) a uložíme si jí. Zvolenou barvu pak porovnáváme s ostatními barvami v následujících cyklech, které prochází bitmapou řádek po řádku a zaměňují námi zvolenou barvu za barvu bílou.

```
/* Program v C#
   bm - bitmapa
    c - námi zvolená barva
*/
for(int y=0; y<bm.Height; y++){
  for(int x=0; x<bm.Width; x++){
    if (bm.GetPixel(x,y).Equals(c)){
      bm.SetPixel(x, y, Color.White);
    }
  }
}
```

Nakonec aby se projevily změny v obrázku, překreslíme plátno metodou *repaint*.

## 4.9 Jednoduché programy

Mezi jednoduché programy patří tři prográmky:

- **Kresli úsečku**
  v největší jednoduchosti demonstruje kreslení úseček po obrazovce (obdoba programu Kreslení myší, viz oddíl 4.4)

- **Kresli pixel**
  v největší jednoduchosti demonstruje kreslení pixelu (čáry) po obrazovce (část programu Obrázek, viz oddíl 4.6)

- **Bitmapa**
  v jednoduchosti demonstruje načtení bitmapy, kreslení do ní a opětné uložení (obdoba programu Obrázek, viz oddíl 4.6)

## 4.10 Programy do přednášek

Mým posledním úkolem této sekce bylo přepsat několik zdrojových kódů z přednášek předmětu KPG z programovacího jazyka Delphi do jazyka Java. Mezi které patří algoritmy těchto objektů:

- **Pravidelný N-úhelník (Polygon)**
  - je tvořen z několika úseček a jehož vstupem je umístění, poloměr a počet vrcholů.
- **Oblouk**
  - je vykreslen podle zadaného umístění, poloměru, počátečního a koncového úhlu.
- **Rozeta**
  - je obdobou pravidelného N-úhelníku, ale spojuje každý vrchol se všemi ostatními.
- **Parametrická křivka**
  - je algoritmus pro vykreslení parametricky zadané křivky
- **Křivka Kochové a C-křivka**
  - rekurzivní algoritmus výpočtu.
- **Juliova a Mandelbrotova množina**
  - rekurzivní algoritmus výpočtu těchto fraktálů.
- **Hra Život(Life)**
  - počítačová simulace „života“.

# 5 Webová stránka grafických předmětů

Webová stránka grafických předmětů je složena ze tří rámů(viz obrázek 9 na straně 23), tzn. že plocha prohlížeče je rozdělena na tři pravoúhlé části,

z nichž se každá chová jako samostatný prohlížeč. Rámy jsou samostatnými HTML soubory:

- **index.htm**
  je hlavní „spouštěcí“ soubor webové stránky, který vytváří a spravuje všechny tři rámy.
- **head.htm**
  je horní rám, nebo-li hlavička webové stránky.
- **menu.htm**
  je levý rám, který vytváří rozbalovací menu.
- **default.htm**
  je pravý základní rám. Základní, protože se v něm zobrazují výsledky odkazů z menu.


Obrázek 9: Webová stránka

## 5.1 Menu

Soubor *menu.htm* je vertikální rozbalovací menu. Vytvořeno je pomocí JavaScriptu a kaskádových stylů CSS ze souboru *menu.css* v adresáři *gui*.

Kliknutím myši na název grafického předmětu se nám rozbalí odkazy na informace a materiály o předmětu. Kliknutím na odkazy se již dostaneme na potřebné informace, které se zobrazí v pravém základním rámu.

## 5.2 Informace o předmětech

Informace o předmětech nalezneme v rootu internetové stránky v adresáři zkratky předmětu (např. KPG, PRO, ...) v HTML souborech (*prednasky.htm, materialy.htm, ...*). Tyto soubory jsou stylované pomocí kaskádových stylů CSS, takže již máme vytvořené formátování HTML stránky.

# 6 Zpětná vazba od studentů

Jelikož je tato bakalářská práce převážně vytvořena pro studenty ZČU, mohli se na ní také podílet. Práce byla předložena studentům předmětu KPG v letním semestru 2006/2007 a studentům 1. ročníku a jejich zpětná vazba byla využita k doplnění a zdokonalení, aby vytvořené materiály co nejlépe vyhovovaly výuce základů programování grafiky. Především šlo o hledání nedostatků a chyb ve výukových dokumentech a v praktických příkladech, případně předkládání návrhů na doplnění málo jasných kapitol nebo na doplnění možných funkcí programů.

## 6.1 Kritické připomínky k výukovým dokumentům

Nejprve bych začal s kritickými připomínkami, které jsem přijal za věcné a následně je napravil:

- opravit pravopisné chyby a překlepy
- popsat ovládání metody *paint()* v Javě
- vysvětlit, co je péro a štětec

- u návodu, jak v Javě přeložit program, doplnit v jakém editoru se má program psát
- doplnit verzi MS Visual Studia
- přidat odkazy na další zdroje o grafice

Dále budu pokračovat s kritickými připomínkami, které jsem nevyužil, a zdůvodním proč:

- zbytečný návod, jak přeložit a spustit program, kdo chce začít s grafikou, měl by to znát
  - s tímto nesouhlasím, jelikož práce je navržena i pro studenty, kteří dříve programovali v jiných programovacích jazycích než Java
- příklad na použití primitiv (jak se do metod zadávají hodnoty)
  - toto je dle mého názoru v dokumentech obecně obsaženo, je možno si vše dohledat v praktických programech (především v programu Primitiva)
- vysvětlit použití listenerů v Javě (události klávesnice a myši)
  - to je věcný návrh, ale nerealizoval jsem z důvodu obsáhlosti tohoto tématu. Na druhou stranu je použití listenerů obsaženo téměř v každém praktickém programu a je i komentováno
- použití více barev textu pro lepší orientaci
  - nebývá zvykem technické dokumenty psát mnoha různými barvami. Zvýraznění důležitých věcí je použito.

## 6.2 Kritické připomínky k praktickým programům

Opět začnu s kritickými připomínkami, které jsem přijal za věcné a následně jim vyhověl:

- **Program Fonty**
  - doplnit možnost vyčistit kreslící plochu

- **Program Hodiny**
  - očíslovat ciferník
- **Program Klávesy**
  - přidat možnost nové hry
  - zabránit červenému kruhu (avatarovi) odejít mimo hranice okna
  - okomentovat podmínku sběru předmětů
- **Program Kreslení myší**
  - absence nakreslení jednoho bodu
  - doplnit možnost vyčistit kreslící plochu
- **Program Obrázek**
  - absence nakreslení jednoho bodu
  - doplnit možnost vyčistit kreslící plochu
- **Program Primitiva**
  - Grafické objekty jsou moc velké, hrany splynou s okrajem plátna

Dále pokračuji kritikou, kterou jsem nevyužil a té bylo opravdu málo. Samozřejmě opět zdůvodním proč:

- **Program Klávesy**
  - u programu v Javě v dolní části okna je vidět textové pole, který nemá žádnou logickou funkci
    - bohužel funkci má, je přes něj aktivován *KeyListener* (události klávesnice), jelikož alespoň v JDK 1.5.0 nejde *JPanelu* přidat *KeyListener*, jelikož panel je neaktivní
- **Program Kreslení myší**
  - během kreslení úsečky se neustále překreslují všechny dosud nakreslené úsečky, což je neefektivní
    - to to je pravda, ale jedná se o výukový příklad a ne o pokročilý grafický editor (obdobně je tomu tak i v programu Obrázek)

Zbylé programy, které zde nejsou popsány, byly buď bez připomínek, nebo ještě nebyly, v době podání studentům ke kritice, naprogramované, protože byly vytvořeny právě na základě připomínek studentů a vedoucí práce.

# 7 Závěr

Výsledkem této bakalářské práce je soubor výukových dokumentů a pomocných praktických příkladů pro výuku počítačové grafiky, převážně pro výuku předmětu KPG (Krásy počítačové grafiky).

Bakalářská práce by měla především sloužit studentům v začátcích s počítačovou grafikou. Proto psaní programů a dokumentů bylo ovlivněno nápady a připomínky studentů.

Dokumenty a praktické příklady jsou volně šiřitelné a v budoucnu by měly být dostupny na internetových stránkách Doc. Dr. Ing. Ivany Kolingerové.

# Literatura


[1] Herout, P. *Učebnice jazyka Java.* České Budějovice, KOPP, 2003.

[2] Herout, P. *Java - grafické uživatelské prostředí a čeština.* České Budějovice, KOPP, 2001.

[3] Pitner, T. *Java - začínáme programovat.* Praha, Grada, 2002.

[4] Písek, S. *Delphi - praktické příklady.* Praha, Grada, 2002.

[5] Hlavenka, J. *Vytváříme www stránky.* Praha, Computer Pres, 2002.

[6] Programovací jazyky. Článek na wikipedia.org [online].
URL:http://cs.wikipedia.org/wiki/Programovac%C3%AD_jazyk

[7] Kovář, D. *Programování se zaměřením na .NET a jazyk C#.* [online].
URL:http://projektysipvz.gytool.cz/ProjektySIPVZ/

[8] C++ a C#. Články na zive.cz [online].
URL:http://zive.cz/Programovani/C_CSHARP/sc-74/

[9] 2D Graphics. Články na java.sun.com [online].
URL:http://java.sun.com/docs/books/tutorial/2d/index.html

# Rejstřík