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

Bakalářská práce

Grafické rozhraní pro fyzikální výpočty

Plzeň, 2011 Michal Šmolík

Zadání

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: **2010/2011**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Michal ŠMOLÍK**
Osobní číslo:	**A08B0161P**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Informatika**
Název tématu:	**Grafické rozhraní pro fyzikální výpočty**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování:

1. Seznamte se s principy grafických rozhraní.
2. Analyzujte a formulujte třídy fyzikálních úloh, které je možné řešit pouze přímým dosazování, a které je možné řešit soustavou rovnic.
3. Navrhněte metodu řešení jednotlivých tříd fyzikálních úloh.
4. Navrhněte datové struktury a specifikujte programové řešení.
5. Navržený systém implementujte s vybraným grafickým rozhraním.
6. Systém ověřte na vybraných problémech.
7. Realizujte řádnou programovou a uživatelskou dokumentaci.

Prohlášení

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

V Plzni dne …… ……

Michal Šmolík

Poděkování

Děkuji vedoucímu této bakalářské práce prof. Ing Václavu Skalovi, CSc., za hodnotné rady a odborné vedení během vzniku této bakalářské práce. Dále bych chtěl poděkovat Mgr. Heleně Čížkové za odbornou konzultaci z pohledu vyučujícího fyziky. V neposlední řadě také studentům předmětu Základy počítačové grafiky, kteří mi pomohli s testováním aplikace a předali cenné rady ve vyplněném hodnotícím dotazníku.

Anotace

Předmětem bakalářské práce je vytvoření grafického rozhraní pro výpočet fyzikálních slovních úloh s úrovní složitosti středních škol. Systém využívá výpočetního softwaru Maxima.

Cílem práce je vytvoření programu, který umožní studentům výpočty slovních fyzikálních úloh a který přispěje k lepšímu pochopení fyzikální podstaty řešeného tématu.

Abstract

The subject of this bachelor thesis is to create graphical interface for calculating physical verbal tasks with the high school level of complexity. In order to perform the calculations, the Maxima software is used.

This thesis aims to create a program that allows students to solve the verbal physical tasks and which will contribute to better understand the physical nature of the subject.

Obsah

1 Úvod

Nemalá část studentů středních škol má problémy s počítáním a chápáním fyzikálních slovních úloh, ať už zadání patří mezi lehčí, nebo obtížnější. Jejich hlavní pozornost při učení se ubírá na zapamatování si všech fyzikálních vzorců z probírané fyzikální látky, ale již se nesnaží nebo jim nezbývá čas na to, aby se naučili a pochopili vyučovanou fyzikální problematiku. Tento způsob studia není ovšem vůbec správný. Student se naučí pouze seznam fyzikálních vzorců, ale o tom co znamenají, nebo jak a za jakých podmínek se mohou používat, neví zdaleka nic.

Tato bakalářská práce je zaměřena na vytvoření uživatelského systému, který umožní počítat fyzikální úlohy, přičemž se student koncentruje na fyzikální podstatu řešeného problému a formulaci řešení. Vytvořený výpočetní systém pak danou úlohu numericky vypočte. Tento přístup osvobozuje studenta od nutnosti pamatovat si všechny fyzikální vzorečky, neboť systém pro výpočet fyzikálních slovních úloh mu nabízí asistenci založenou na řešené fyzikální úloze. Tím se student může plně věnovat řešení fyzikálního problému a případně vysvětlení řešeného fyzikálního jevu.

1.1 Vize programu

Řešení, které by zajisté pomohlo studentovi s učením probírané látky, by mělo zajišťovat několik důležitých věcí. Student by měl být odstíněn od učení se fyzikálních vzorců nazpaměť. Výběr se provádí definováním daného fyzikálního jevu. Tímto by bylo docíleno, že student musí nad danou úlohou přemýšlet a pochopit její fyzikální zákonitosti. Další problém, který by měl být studentovi ulehčen, je matematické skládání rovnic a řešení soustav. Někteří studenti, kteří nejsou příliš zdatní v matematických výpočtech, mají problém

danou úlohu dopočítat a tím i většinou ztrácejí zájem a chuť řešit další fyzikální slovní úlohy.

1.2 Účel programu

Program s grafickým rozhraním pro fyzikální výpočty je vyvíjen proto, aby se student pouze neučil zpaměti veškeré fyzikální vzorce, ale zaměřil se na lepší pochopení probírané fyzikální látky.

Program má za úkol umožnit studentovi postupným vybíráním vzorců, k neznámým veličinám, vyřešit fyzikální slovní úlohu. Student si tedy nemusí pamatovat přesně vzorce, neboť mu budou nabízeny, ale musí vědět, co jaký vzorec znamená. Dále nemusí výslednou soustavu ani řešit početně, neboť to udělá program za studenta. Tímto je práce velice ulehčena a po studentovi je požadováno to nejdůležitější a to, aby se snažil pochopit fyzikální podstatu řešené úlohy. Toto by měl být i hlavní cíl při vyučování fyziky na školách. Je totiž nepotřebné, aby se student pouze učil zpaměti fyzikální vzorce, když poté nebude vědět, co znamenají, nebo kdy a v jakém případě platí a mohou se použít. Nejdůležitější je, zda pochopí probíraný fyzikální problém, díky kterému poté bude schopen vyřešit slovní úlohu.

1.3 Dosažené výsledky

Předkládaný výpočetní systém byl ověřen na netriviálních úlohách a upraven do finální podoby po konzultaci s pracovníky Gymnázia v Rokycanech a pracovníky Pedagogické fakulty na Západočeské univerzitě v Plzni.

2 Třídy fyzikálních úloh

Při výpočtu slovní úlohy je třeba zvolit, jak začít řešit danou úlohu. Jsou možné dva způsoby. Prvním z nich je začít výpočet od neznámé veličiny a druhý způsob je začít výpočet od fyzikálního vzorce.

Každý fyzikální vzorec obsahuje veličiny, které je třeba pro výpočet dosadit. Hodnota neznámé veličiny je buďto známá, nebo je potřeba neznámou veličinu vyjádřit pomocí dalšího fyzikálního vzorce. V této chvíli je důležité se zamyslet, jaký má daná veličina význam. Pokud by se jednalo například o sílu, mohla by to být síla třecí, tíhová, dostředivá, vztlaková, nebo jakákoliv jiná. Toto je velice důležité si uvědomit, neboť po upřesnění dané fyzikální veličiny se zpravidla počet možných fyzikálních vzorců zredukuje pouze na jeden, který se již pouze vloží do výpočtového stromu. Postupným přidáváním fyzikálních vzorců a zadáváním hodnot neznámých veličin se vytvoří výsledný výpočtový graf dané slovní úlohy.

Řešení slovních úloh ve fyzice není jednoduchou záležitostí. Existuje totiž více tříd těchto úloh, kde každá se řeší částečně odlišným způsobem. Jedno z možných rozdělení je na úlohy, které se řeší přímým dosazováním, a které se řeší pomocí soustavy rovnic.

2.1 Přímé dosazování

Nejjednodušší třídou fyzikálních úloh jsou takové, které se dají řešit přímým dosazováním. V takovém případě se hledaná veličina vyjádří pomocí fyzikálního vzorce, charakterizujícího danou fyzikální problematiku. Poté se postupně neznámé veličiny ve vzorci nahrazují fyzikálními vzorci, pomocí nichž se potřebná veličina počítá. Po dosazení vzorců za všechny neznámé veličiny vznikne jeden velký vzorec. Do tohoto vzorce stačí již jen dosadit známé číselné hodnoty a provést numerický výpočet.

2.1.1 Příklad

- **Zadání příkladu**

  Na těleso o hmotnosti 10kg působí stálá síla o velikosti 5N. Určete kinetickou energii tělesa na konci druhé sekundy pohybu. Těleso bylo předtím v klidu.

Neznámou veličinou ve slovní úloze je energie. Její výpočet se provede pomocí vzorce pro určení kinetické energie translačního pohybu, který má tvar

$$E_k = \frac{1}{2}mv^2$$

V tomto vzorci jsou neznámé veličiny hmotnost, která je zadána, a rychlost, která se musí vyjádřit pomocí dalšího vzorce. Protože na těleso působí konstantní síla, koná rovnoměrně zrychlený přímočarý pohyb. Pro výpočet rychlosti se tedy použije vzorec

$$v = at$$

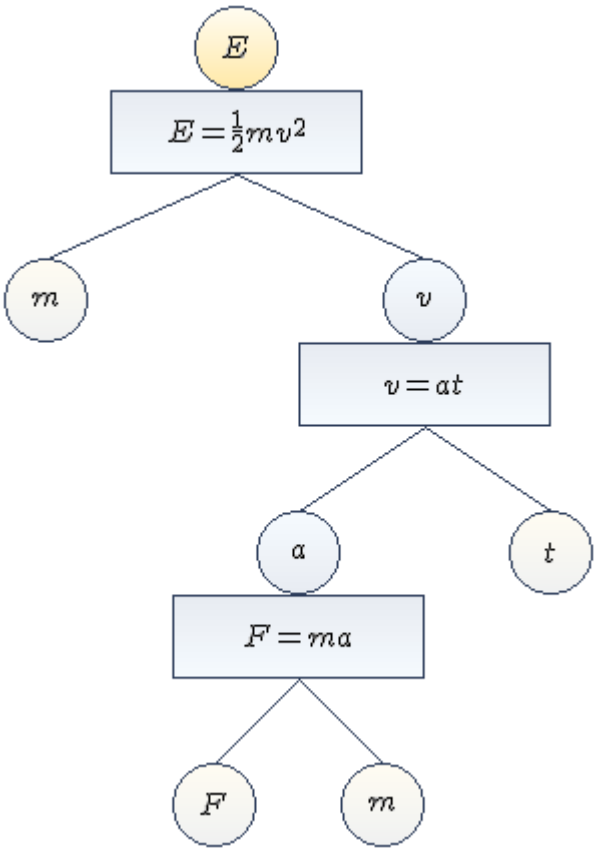
V tomto vzorci jsou neznámé veličiny čas, který je zadán, a zrychlení, které se musí vyjádřit pomocí dalšího vzorce. Jelikož na těleso působí konstantní síla, udává tělesu zrychlení podle druhého Newtonova zákona, který zní

$$F = ma \Rightarrow a = \frac{F}{m}$$

V tomto vzorci jsou neznámé veličiny síla a hmotnost. Obě tyto veličiny jsou zadány, tudíž se již může vytvořit výsledný vzorec, který bude mít tvar

$$E_k = \frac{1}{2}m\left(\frac{F}{m}t\right)^2$$

Pokud se postup zaznamená do výpočtového stromu, bude vypadat tak, jak je vyobrazen na obrázku Obrázek 2.1, který je uveden dole.



Obrázek 2.1 Grafické znázornění vzorců pro výpočet

2.2 Soustava rovnic

Další třídou fyzikálních úloh jsou takové, které již není možné řešit přímým dosazením. Tyto úlohy se tedy musí řešit pomocí soustavy rovnic nebo úpravou algebraických výrazů, což je ovšem to samé, jako částečně upravená soustava rovnic.

Typickým příkladem jsou pohybové úlohy, při kterých proti sobě jedou dvě vozidla a je úkolem spočíst dobu, za jakou se srazí, kolik mezitím ujedou a podobně. Další skupinou úloh jsou takové, u kterých se počítá s fyzikálními vzorci, v nichž se vyskytuje veličina, která výsledně nebude k výpočtu vůbec zapotřebí, neboť na ní výsledek nezáleží.

2.2.1 Příklad

- **Zadání příkladu**

  Střela o rychlosti 150 ms^{-1} vnikla do hloubky 65 cm dřevěné překážky. Určete, s jakým zpomalením se pohybovala střela v překážce.

Neznámou veličinou ve slovní úloze je zrychlení, v případě tohoto zadání zpomalení. Po nárazu do překážky se bude střela pohybovat rovnoměrně zpomaleným přímočarým pohybem, dokud se nezastaví. Výpočet zpomalení se provede pomocí vzorce

$$v = at$$

V tomto vzorci jsou neznámé veličiny rychlost, která je zadána, neboť je to rychlost před vniknutím střely do překážky, a čas, který se musí vyjádřit pomocí dalšího vzorce. Doba, během které bude zpomalování probíhat, musí být rovna času, za jakou urazí střela zadanou dráhu. Proto se neznámá veličina čas určí pomocí vzorce

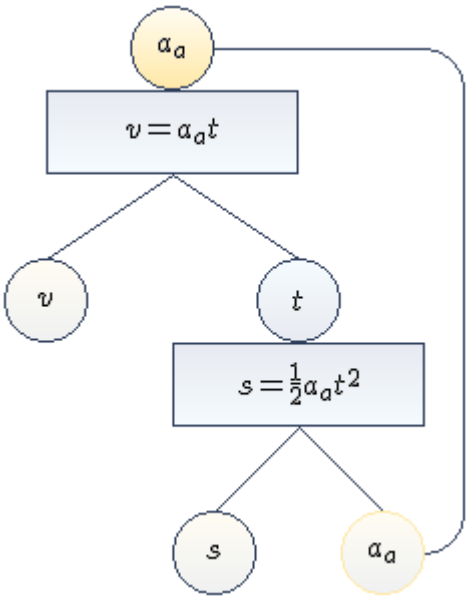
$$s = \frac{1}{2}at^2$$

V tomto vzorci jsou neznámé veličiny dráha, která je zadána, a zrychlení, neboli v této fyzikální úloze zpomalení, které je ovšem požadovanou neznámou veličinou v zadané slovní úloze. V tomto případě tedy nelze vytvořit postupným upřesňováním neznámých veličin jeden výsledný vzorec, do kterého by se dalo pouze dosadit a provést numerický výpočet. Nyní je potřeba vyřešit soustavu rovnic

$$v = at$$

$$s = \frac{1}{2}at^2$$

Po vyřešení soustavy rovnic se získá požadovaný výsledek. Pokud se postup zaznamená do výpočtového stromu, bude vypadat tak, jak je vyobrazen na obrázku Obrázek 2.2, který je uveden dole.



Obrázek 2.2 Grafické znázornění vzorců pro výpočet

3 Metody řešení fyzikálních úloh

V předchozí kapitole byly uvedeny dvě třídy fyzikálních úloh. Každá třída se dala řešit jiným matematickým postupem. Pro programové řešení je nicméně jednodušší, pokud by existoval jeden způsob, kterým by se daly řešit obě třídy úloh.

Možné řešení je počítat požadovanou výslednou hodnotu pomocí soustavy rovnic. Tímto způsobem lze totiž počítat i fyzikální úlohy, které se daly řešit přímým dosazováním.

3.1 Soustava rovnic

Pomocí soustavy rovnic lze vyřešit veškeré třídy fyzikálních úloh. Při vytváření soustavy rovnic je ovšem nutné dbát na to, aby význam veličin v soustavě byl jednoznačný. Může se totiž stát, že v soustavě existují dvě shodné značky, ale přitom každá charakterizuje jinou hodnotu nebo dokonce jinou fyzikální veličinu. Konflikt může nastat například mezi veličinou čas a teplota. Obě tyto fyzikální veličiny mají totožnou fyzikální značku a to malé *t*.

3.1.1 Rozlišení značek fyzikálních veličin

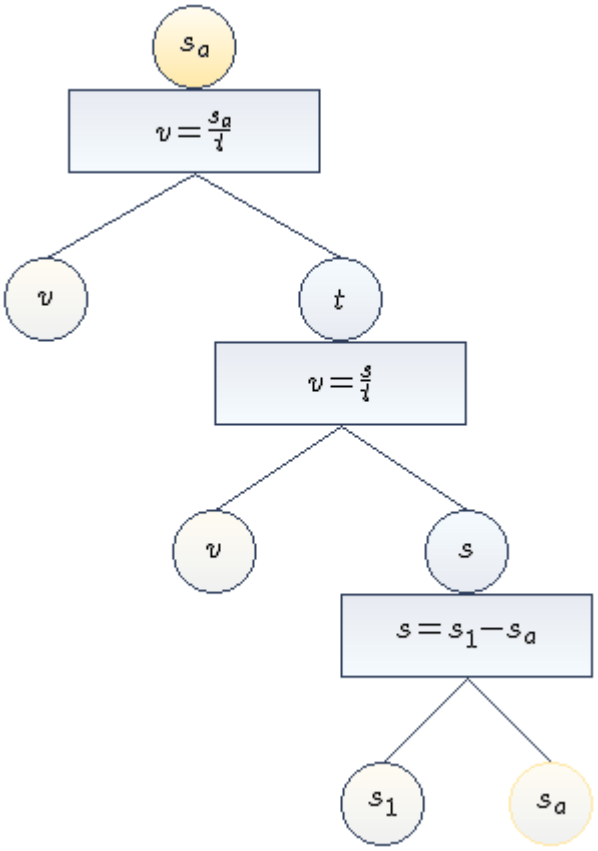
Kvůli jednoznačnosti je nutné, aby nemohl nastat případ, kdy budou dvě stejné značky zastupovat jiné číselné hodnoty nebo jiné veličiny. Z tohoto důvodu je vhodné veličiny, u kterých může nejednoznačnost nastat, indexovat. Pokud se například v rovnici vyskytne teplota a čas, bude mít teplota značku t_1 a čas t_2.

3.1.2 Příklad řešení soustavy rovnic

Na obrázku (Obrázek 3.1) jsou graficky znázorněny vzorce pro výpočet neznámé veličiny dráha.

- **Zadání příkladu**

  Dvě protijedoucí auta jsou od sebe vzdáleny 10km. První auto jede rychlostí 65km/h a druhé auto jede rychlostí 90km/h. Jakou vzdálenost ujede první auto, než se obě auta společně střetnou?



Obrázek 3.1 Grafické znázornění vzorců pro výpočet

Z obrázku (Obrázek 3.1) lze sepsat rovnice pro řešení soustavy, které budou vypadat takto:

$$v = \frac{s_a}{t} \qquad v = \frac{s}{t} \qquad s = s_1 - s_a$$

$$v = 65 \qquad v = 90 \qquad s_1 = 10$$

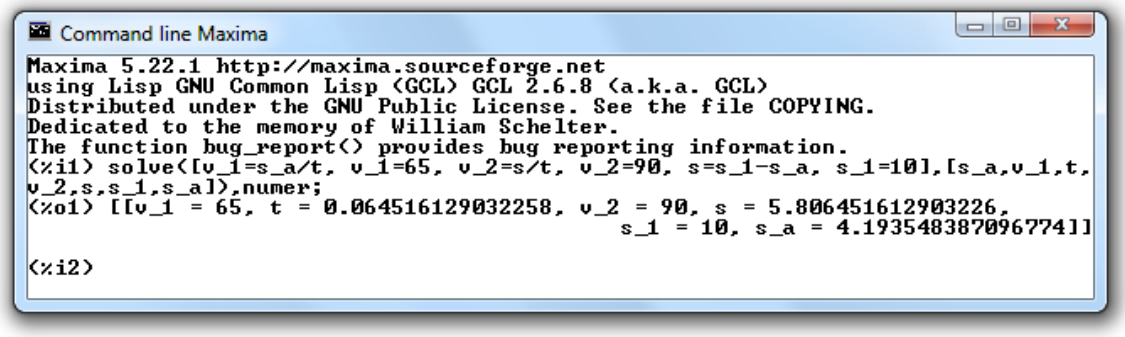
Již na první pohled je zřejmé, že soustavu rovnic nepůjde vyřešit, neboť existují dvě různé hodnoty pro značku *v*. Je tedy nutné tyto dvě značky pro

rychlost odlišit indexem. Výsledná soustava rovnic po úpravě bude vypadat následovně:

$$v_1 = \frac{s_a}{t} \qquad v_2 = \frac{s}{t} \qquad s = s_1 - s_a$$

$$v_1 = 65 \qquad v_2 = 90 \qquad s_1 = 10$$

Nyní je již soustava rovnic zadána jednoznačně a je možné ji začít řešit. K tomuto účelu je využíván program Maxima, který slouží k matematickým výpočtům. Po zadání soustavy rovnic do výše uvedeného programu Maxima bude výstup vypadat následovně (Obrázek 3.2)



Obrázek 3.2 Řešení soustavy rovnic pomocí programu Maxima

Výsledek řešení soustavy rovnic je uveden za značkou *(%o1)*, což je zkratka pro *output1*, a je:

$$v_1 = 65, \quad t = 0.0645, \quad v_2 = 90, \quad s = 5.81, \quad s_1 = 10, \quad s_a = 4.19$$

3.1.3 Pravidla pro indexování značek

Při indexování značek není nutné indexovat veškeré značky. Ale pouze ty, u kterých by mohla nastat nejednoznačnost. Je tedy důležité dodržovat určitá pravidla.

- **Číslo indexu**

  Čísla indexování musejí být unikátní. Toho lze dosáhnout tím, že index u veličiny bude odvozen z pořadového čísla rovnice.

- **Indexované veličiny**

  Indexovat se nesmí veličiny, které jsou upřesněné dalším fyzikálním vzorcem, nebo jsou označené jako shodné s jinou fyzikální veličinou.

4 Programovací jazyk

Důležitým rozhodnutím je, v jakém programovacím jazyku bude program pro výpočty fyzikálních slovních úloh naprogramován. Základním rozhodnutím je, zda musí být program multiplatformní a tudíž spustitelný na více operačních systémech. Protože je ovšem aplikace vyvíjena pro školní účely, není to nutná podmínka. Ve středních školách se totiž téměř všude využívá operačního systému Windows nebo je dostupný alespoň na některých počítačích.

Z hlediska rychlosti není nutné používat jazyk C nebo C++, protože výsledná aplikace nebude příliš náročná na rychlost. Jako programovací jazyk je tedy možné zvolit C#, ve kterém bude také program napsán.

4.1 C#

Jazyk C# vyvinula firma Microsoft, jak je uvedeno v (1). Byl představen spolu s celým vývojovým prostředím .NET. Jak název napovídá, vychází tento jazyk v mnohém z programovacího jazyka C/C++, ale v mnoha ohledech je daleko bližší programovacímu jazyku Java. Základní charakteristiky jazyka jsou:

- Jazyk C# je čistě objektově orientovaný.
- Obsahuje nativní podporu komponentového programování.
- Podobně jako Java obsahuje pouze jednoduchou dědičnost s možností násobné implementace rozhraní.
- Vedle členských dat a metod přidává vlastnosti a události.
- Správa paměti je automatická. O korektní uvolňování zdrojů aplikace se stará garbage collector.
- Podporuje zpracování chyb pomocí výjimek.
- Zajišťuje typovou bezpečnost a podporuje řízení verzí.

- Podporuje atributové programování.
- Zajišťuje zpětnou kompatibilitu se stávajícím kódem jak na binární tak na zdrojové úrovni.

Většina uvedených vlastností vychází přímo s funkcionality vývojového rámce .NET. Jazyk C# je také integrován do vývojového prostředí Visual Studio.NET, ve kterém byl vyvíjen celý program této bakalářské práce.

Překladače jazyka C# jsou case sensitive. Rozlišují tedy velká a malá písmena. Podobně jako v jiných programovacích jazycích, i v jazyce C# bylo zavedeno několik konvencí. Jména balíků, tříd, rozhraní a většiny dalších položek začínají velkým písmenem. Malým začínají privátní a chráněné (protected) atributy, lokální proměnné a parametry. Následující příklad ukazuje jednoduchou kostru programu, kterou vygeneruje Visual Studio, vytvoříte-li konzolovou aplikaci.

```
using System;

namespace ConsoleApplication
{
        /// <summary>
        /// Summary description for Program.
        /// </summary>
        class Program
        {
                /// <summary>
                /// The main entry point for the application.
                /// </summary>
                [STAThread]
                static void Main(string[] args)
                {
                        //
                        // TODO: Add code to start application here
                        //
                }
        }
}
```

Předchozí příklad také ukazuje různé typy komentářů v jazyce C#. Podobně jako v C/C++ lze používat jak víceřádkové komentáře uvozené /* */ tak jednořádkové komentáře po znacích //. Speciální význam má značka TODO. Komentář, který po ní následuje, se zobrazí v panelu aplikace Visual Studio s názvem Task List. Jednořádkové komentáře uvozené třemi lomítky budou obsaženy v dokumentaci, která je standardně generována ze zdrojového kódu (podobně jako v Javě /** */). Generovaná dokumentace využívá XML.

4.1.1 .NET Framework

Počátkem 90. let byly většinou vytvářeny samostatné aplikace s velmi malou schopností vzájemné komunikace. Tento nedostatek byl odstraněn v polovině 90. let, kdy firma Microsoft uvedla technologii COM (Component Object Model). Obrovskou výhodou komponentové technologie je její jazyková neutralita v binární formě. Pro každou komponentu bylo definováno rozhraní, které zprostředkovává komunikaci mezi klientem a příslušnou komponentou. Časem se však ukázalo, že i tato technologie má svá omezení. V dnešní době je modulární architektura čím dál používanější. Využívané komponenty jsou většinou malé a jednoduché. Hlavní nevýhodou COM komponent je, že zakrývají svou vnitřní realizaci. Jediné co komponenty popisuje je příslušné rozhraní. Toto znemožňuje dědičnost na úrovni zdrojových kódů.

.NET Framework funguje doslova jako substrát, na kterém lze pěstovat software. Jeho jádro je založené na principech objektově orientovaného programování a všechny základní služby zpřístupňuje široké škále programovacím jazykům. .NET Framework automaticky podporuje třídy, metody, vlastnosti, konstruktory, události, polymorfismus atd. Ve výsledném efektu to znamená, že není podstatné, ve kterém programovacím jazyce komponenty vytváříme případně, jaké komponenty používáme. .NET Framework také řeší některé problémy související s bezpečností. Dalším problémem, který .NET Framework řeší, je nasazování a instalace aplikací (označovaný jako DLL Hell).

5 Grafické rozhraní

Programovacím jazykem byl zvolen C#, v němž je zapotřebí zvolit vhodné grafické rozhraní pro tvorbu GUI. Grafická knihovna musí umožňovat tvorbu vlastních ovládacích prvků, pokročilé možnosti kreslení a práci s obrázky. V úvahu připadají tři možné grafické knihovny:

- Windows Forms
- Qt 4
- GTK+

Nejvhodnější bude použití grafického rozhraní Windows Forms, které je již součástí .NET Frameworku.

V následujících kapitolách jsou popsány tři grafické knihovny, mezi kterými bylo vybíráno.

5.1 Windows Forms

Windows Forms, jak je uvedeno v (2), je grafické rozhraní pro programování aplikací (API). Je zařazeno jako součást Microsoft .NET Framework. Poskytuje přístup k nativním prvkům operačního systému Microsoft Windows pomocí zabaleného existujícího Windows API. Někdy je považováno Windows Forms jako náhrada za starší a složitější C++ založené na Microsoft Foundation Class Library.

Pomocí Windows Forms je možné, jak je uvedeno v (3), vytvářet různé ovládací prvky, formuláře, kreslit a mnoho dalšího.

5.1.1 Formulář

Třída Form nabízí řadu vlastností, které ovlivňují vzhled a chování formuláře.

Aplikace typu Windows Forms je událostně řízená. Ovládací prvky vyvolávají události (events) a na tyto události může reagovat klientský kód obsloužením dané události. Aplikace typu Windows Forms začíná statickou metodou *System.Windows.Forms.Application Run()*, které je předán rámcový formulář aplikace. Uvnitř této metody se neustále čeká na události, jako například pohyb myší, kliknutí, stisk tlačítka či žádost o překreslení okna. Zaměřme se zatím na posledně zmíněnou událost, která je vyvolána, pokud je potřeba překreslit okno. Taková situace může nastat například v případě, kdy okno bylo skryto a následně zobrazeno. Je potřeba dané okno překreslit, obzvláště klientskou část okna, která je závislá na konkrétní aplikaci.

Odpovídající událostí třídy Form je událost *Paint*. Abychom tuto událost obsloužili, musíme definovat metodu, jež má stejné parametry jako delegát *PaintEventHandler*, který vypadá následovně:

public delegate void PaintEventHandler(object sender, PaintEventArgs e);

Prvním argumentem obslužné metody je objekt, který událost vysílá, druhým je objekt typu *PaintEventArgs*. Pomocí druhého argumentu získáme grafický kontext, pomocí něhož lze vykreslovat do okna prvku, který danou metodu vyslal.

Většinou se formulář odvodí ze třídy Form, v konstruktoru se nastaví jeho vlastnosti a jeho nová instance se předá metodě *Main*. V takovém případě můžeme přímo překrýt metodu *OnPaint*, která představuje obslužnou metodu pro událost *Paint*. Tato metoda má pouze jeden argument. Argument "*sender*" nemá význam, neboť je zřejmé, že událost vyslal tento formulář (*this*).

5.1.2 Kreslení křivek a ploch

V rámci .NET jsou definovány ve jmenném prostoru *System.Drawing* následující struktury, které jsou velmi potřebné pro udání polohy, rozměrů a hraničních pravoúhelníků při vykreslování v okně a na tiskárně:

- Point = pozice jednoho pixelu
- Size = velikost ovládacího prvku
- Rectangle = obdélníková plocha s daným umístěním a rozměrem

Připomeňme, že struktury nejsou (na rozdíl od tříd) odkazové (referenční) typy. Z praktického pohledu to znamená, že jsou vytvářeny na zásobníku, což má za následek zvýšení výkonu v případě, že se jedná o malé struktury, které se často vytvářejí a ruší.

Ve jmenném prostoru *System.Drawing* a *System.Drawing.Drawing2D* jsou definovány následující struktury, které se užívají při vykreslování křivek a ploch:

- Color = barva
- Brush = struktura výplně
- Pen = struktura pro kreslení křivek

Pomocí metod, které jsou instance třídy *Graphics* lze vykreslit úsečky a lomené čáry, libovolné křivky zadané parametrickými rovnicemi, pravoúhelníky a mnohoúhelníky, elipsy a kružnice a v neposlední řadě i oblouky a výseče elips, resp. kružnic.

Plochy se vyplňují štětcem. Odpovídající metody instance třídy *Graphics* se nazývají obdobně jako metody pro vykreslování křivek, pouze místo slova *Draw* se vyskytuje slovo *Fill*. Například metoda

public void DrawRectangle(Pen pen, Rectangle rect);

vykreslí obvod pravoúhelníku *rect* pomocí pera *pen*. Metoda

public void FillRectangle(Brush brush, Rectangle rect);

vyplní pravoúhelník *rect* štětcem *brush*.

5.1.3 Ovládací prvky

S nástupem grafického uživatelského rozhraní (GUI) operačního systému Windows začaly vznikat nové a nové grafické ovládací prvky - tlačítka, textová pole, zaškrtávací pole, posuvníky apod. S takovými ovládacími prvky se pracuje velmi intuitivně a každý uživatel ví, jakým způsobem je ovládat. Například tlačítko vypadá stejně jako hardwarové tlačítko a při stisku vizuálně naznačí, že skutečně bylo stisknuto. S ovládacími prvky se pracuje pomocí myši, existuje však také klávesové rozhraní, které je mnohdy časově méně náročné. Ovládací prvky se často umisťují v dialogových oknech, není problém je však umístit přímo na rámcový formulář aplikace.

Ovládací prvky můžeme rozdělit na:

- vestavěné - jedná se o třídy již definované v rámci platformy .NET
- uživatelské - jedná se o nově definované prvky s novou funkcionalitou, jejichž třídy jsou odvozené od třídy UserControl.

Základní vestavěné uživatelské prvky jsou:

- tlačítko - instance třídy *Button*
- zaškrtávací políčko - instance třídy *CheckBox*
- přepínač - instance třídy *RadioButton*
- popisek - instance třídy *Label*
- hypertextový popisek - instance třídy *LinkLabel*
- textové pole - instance třídy *TextBox*
- skupinový rámeček - instance třídy *GroupBox*

- panel - instance třídy *Panel*
- seznam - instance třídy *ListBox*
- zaškrtávací seznam - instance třídy *CheckedListBox*
- rozbalovací seznam - instance třídy *ComboBox*
- vodorovný posuvník - instance třídy *HScrollBar*
- svislý posuvník - instance třídy *VScrollBar*
- číselník - instance třídy *NumericUpDown*
- posuvný jezdec - instance třídy *TrackBar*
- ukazatel průběhu - instance třídy *ProgressBar*

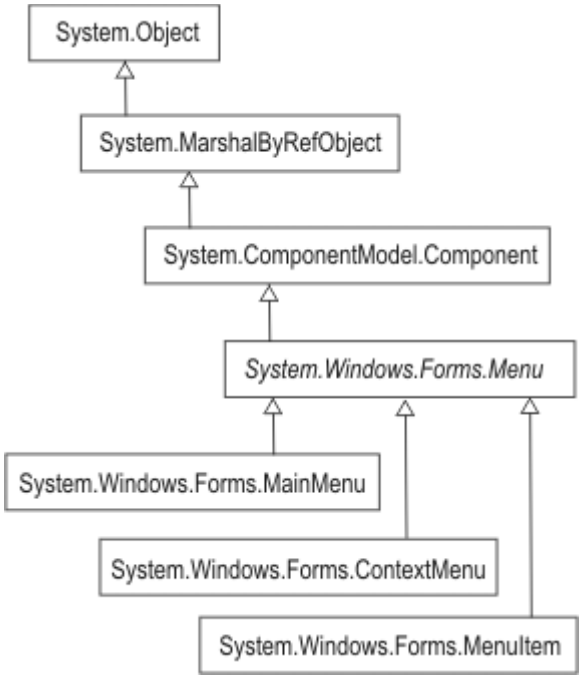
5.1.4 Panely nástrojů

Panely nástrojů sdružují ikony, jež reprezentují grafická tlačítka pro přímé provedení různých akcí, které by jinak bylo potřeba složitě vyhledávat v menu. Pro panel nástrojů můžeme vyvinout vlastní třídu, nebo použít existující třídu *System.Windows.Forms.ToolBar*. Panel nástrojů je kolekcí grafických tlačítek, přičemž bitmapy na jednotlivá tlačítka nastavíme pomocí indexu ze seznamu obrázků. Seznam obrázků je instancí třídy *ImageList*. Všechny obrázky z tohoto seznamu obrázků mají stejnou velikost i barevnou hloubku. Kolekce tlačítek panelu nástrojů je vlastně kolekcí instancí třídy *ToolBarButton*.

5.1.5 Menu

Hlavní menu je úzký pruh, který je ve většině aplikací pro Windows umístěn hned pod titulním řádkem rámcového okna aplikace. Kromě hlavního menu mají aplikace kontextové menu, které se zobrazí například při kliknutí pravým tlačítkem myši na určitý objekt v aplikaci. Kontextové menu je vždy svázáno s určitým kontextem, ke kterému se vztahuje, tzn., že se vždy naplní položkami, které jsou dostupné pro objekt, na němž se kontextové menu vyvolalo.

hlavní menu zapouzdřuje třída *MainMenu*. Tato třída je odvozena od abstraktní třídy *Menu*, a je základní třídou společně pro třídy *MainMenu*, *ContextMenu* a *MenuItem*, jak ukazuje následující diagram tříd na obrázku (Obrázek 5.1).



Obrázek 5.1 Diagram tříd převzatý z (3)

5.2 Qt 4

Qt 4 je svobodná multiplatformní knihovna sloužící primárně (ale nejenom) k vývoji grafických programů. Jejím nativním jazykem je C++, ale existuje i pro jazyky Python (PyQt), Ruby (QtRuby), C, Perl, Pascal, C#, Java (Jambi) a Haskell.

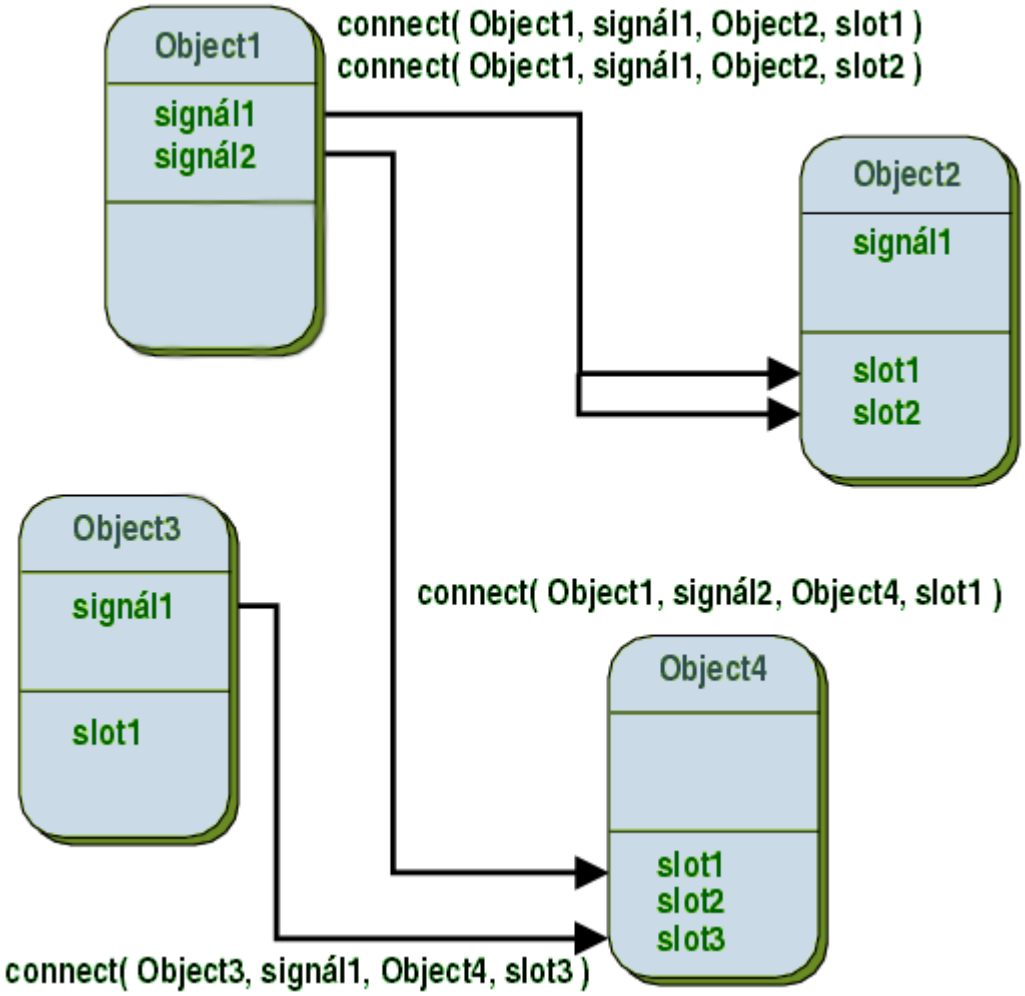
V současnosti Qt používá například populární desktopové prostředí KDE, VoIP komunikátor Skype, webový prohlížeč Opera, virtualizační software VirtualBox a spousty dalších aplikací.

5.2.1 Historie

Historie Qt, jak je uvedeno v (4), začíná v roce 1991 pod jmény Haavard Nord a Eirik Chambe-Eng (původní developer Qt a druhý je president norské společnosti Trolltech, která donedávna stála za vývojem Qt). Velká událost v používání Qt nastala v roce 1998, kdy se začalo vyvíjet grafické prostředí KDE pro GNU/Linux a bylo zvoleno právě Qt jako hlavní knihovna pro implementaci. V poslední době se stala asi nejzajímavější věc v historii tohoto frameworku, protože společnost Nokia koupila společnost Trolltech a tím získala „vládu" nad Qt, který právě Trolltech vyvíjel. Další zásadní událost se stala, když Nokia uvolnila framework pod licenci LPGL, která zaručuje, že software s ním vyvíjený, je možno prodávat bez jakékoliv licence, jež se musela dříve kupovat.

5.2.2 Signály a sloty

Mechanismus signálů a slotů, jak je uvedeno v (5), je základem pro programování v Qt. Umožňuje programátorovi v aplikaci svázat objekty dohromady, aniž by objekty o sobě věděli cokoliv navzájem.



Obrázek 5.2 Signály a sloty (staženo z (6))

Sloty jsou téměř shodné s běžnou C++ funkcí. Mohou být virtuální, mohou být přetížené, mohou být veřejné, chráněné nebo soukromé, mohou být přímo uplatňovány jako každá jiná C++ funkce a jejich parametry mohou být libovolné typy. Rozdíl je v tom, že slot může být také připojen k signálu. V tomto případě je automaticky volán pokaždé, když je vydán signál. Funkce *connect()* vypadá takto:

connect (odesílatel, SIGNÁL (signál), přijímač, SLOT (slot));

kde odesílatel a příjemce jsou ukazatelé na QObject a kde signál a slot jsou funkce s uvedeným typem parametru.

5.3 GTK+

GTK+, jak je uvedeno v (7), je knihovna pro tvorbu grafického uživatelského rozhraní. Knihovna je vytvořena v programovacím jazyce C. GTK+ je také nazýván GIMP Toolkit. Původně byla knihovna vytvořena při vývoji programu GIMP. Od té doby, se GTK+ stal jedním z nejpopulárnějších nástrojů pro Linux. Dnes je většina programů s GUI v open source světě vytvořena v Qt nebo GTK+. GTK+ je objektově orientované aplikační programovací rozhraní. Objektově orientovaný systém je vytvořen pomocí systému objektu Glib, který je základem pro GTK+ knihovny. GObject také umožňuje vytvářet jazykové vazby pro různé jiné programovací jazyky. Jazykové vazby existují pro C++, Python, Perl, Java, C# a mnoho dalších programovacích jazyků.

Podobně jako Qt (ale na rozdíl od jiných) není GTK+ založen na knihovně Xt, což umožňuje využití GTK+ na platformách, kde není X Window System dostupný. Avšak v takovém případě nemá GTK+ přístup do databáze X resources, která umožňuje uživatelské přizpůsobení aplikací v X Window System.

Programy, které využívají danou knihovnu, jsou například GIMP, Firefox nebo Inkscape.

5.3.1 Knihovny GTK+

Grafická knihovna GTK+ je vydávána spolu s ostatními knihovnami, s nimiž blízce spolupracuje.

- **Knihovna Glib**

  Glib je univerzální knihovna. Poskytuje různé datové typy, práci s řetězci, umožňuje zasílání zpráv o chybách, logování zpráv, práce s vlákny a další užitečné programovací funkce.

- **Knihovna Pango**

  Pango je knihovna, která umožňuje internacionalizaci.

- **Knihovna ATK**

  ATK je sada nástrojů pro ovládání. Poskytuje nástroje, které pomáhají tělesně postiženým lidem při práci s počítačem.

- **Knihovna GDK**

  GDK je knihovna poskytující nízko úrovňovou kresbu a funkce poskytující základní grafický systém. V poslední době se hodně funkcí přeneslo do knihovny Cairo.

- **Knihovna GdkPixbuf**

  Knihovna GdkPixbuf je sada nástrojů pro načítání obrázků a práci s pixel bufferem.

- **Knihovna Cairo**

  Cairo je knihovna pro tvorbu 2D vektorové grafiky. Byla zařazena do GTK+ od verze 2.8.

6 Výpočtový graf

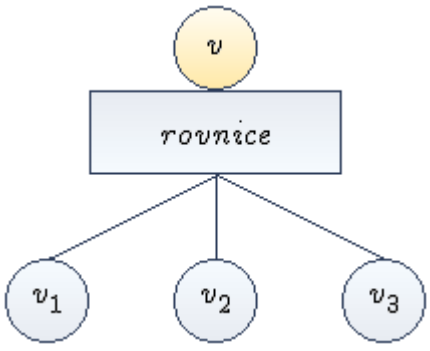
Přiřazováním rovnic k fyzikálním vzorcům vzniká grafová struktura, jež v sobě uchovává veškeré informace potřebné k výslednému výpočtu dané fyzikální slovní úlohy. Je tedy zapotřebí datovou strukturu uchovávat v paměti a pro uživatele vykreslovat.

6.1 Struktura grafu

Struktura výpočtového grafu je velmi podobná binárnímu stromu. Základní rozdíl je v tom, že každý uzel může mít libovolný počet potomků. Není tudíž možné potomky rozlišovat na pravého a levého. Z tohoto důvodu je seznam potomků uložen ve struktuře *List<TreeNode>*, do něhož je možno uložit téměř neomezené množství prvků.

6.2 Kreslení grafu

Výpočtový graf je uložen v datové struktuře, ale pro uživatele je zapotřebí data vizuálně zpodobnit. Vhodným způsobem je zobrazovat graf jako stromovou strukturu, neboli postupně rozvětvovat směrem dolů. Tento způsob zobrazení je ukázán na obrázku (Obrázek 6.1).



Obrázek 6.1 Vizuální zobrazení veličiny a její rovnice

U rozměrnějších grafů nastává problém s vykreslením dané struktury, aniž by se nic nepřekrývalo ani nekřížilo. Jednou z možností, jak postupovat při vykreslování je nejprve umístit spodní veličiny, které již nejsou upřesněny pomocí další fyzikální rovnice.

Umísťování spodních veličin se provede nejprve po x souřadnici. Projde se postupně celý graf pomocí metody procházení *Preorder*, která je zobrazena níže. Zároveň je možné dopočítat souřadnici y pomocí hloubky veličiny v grafu.

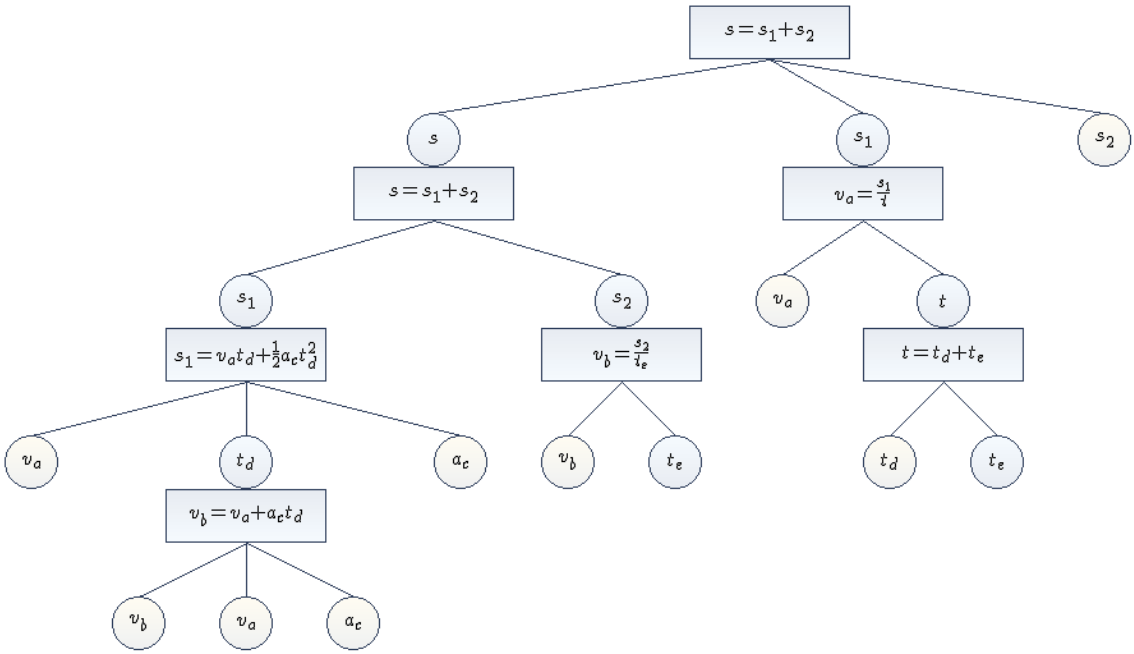
```
private void Preorder(TreeNode node)
{
    if (node.Next == null)
    {
        zleva += VELICINA_WIDTH + MEZERA_VERTICAL;
        node.Umisteni.Souradnice_X = zleva;
        node.Umisteni.Souradnice_Y = node.Umisteni.Hloubka *
                    (VELICINA_HEIGHT + VZOREC_HEIGHT +
                    MEZERA_HORIZONTAL);
    }
    else
    {
        for (int i = 0; i < node.Next.Count; i++)
        {
            Preorder(node.Next[i]);
        }
    }
}
```

Po zjištění pozice všech spodních veličin je již možné dopočítat polohu všech ostatních veličin a vzorců. Postupným procházením grafu zdola se

dopočítávají polohy veličin. Pozice na ose x se spočítá jako průměr pozic na ose x následující veličiny nejvíce vlevo a nejvíce vpravo. Zároveň je opět možné dopočítat souřadnici y pomocí hloubky veličiny v grafu. Pro procházení grafu je využívána metoda *Postorder*, která je uvedena níže.

```
private void Postorder(TreeNode node)
{
        if (node.Next != null)
        {
                for (int i = 0; i < node.Next.Count; i++)
                {
                        Postorder(node.Next[i]);
                }
                int levy = node.Next[0].Umisteni.Souradnice_X;
                int pravy = node.Next[node.Next.Count - 1].Umisteni.Souradnice_X;
                node.Umisteni.Souradnice_X = pravy + (levy - pravy) / 2;
                node.Umisteni.Souradnice_Y = node.Umisteni.Hloubka *
                                        (VELICINA_HEIGHT + VZOREC_HEIGHT +
                                        MEZERA_HORIZONTAL);
        }
}
```

V této chvíli mají již všechny veličiny určené přesně své souřadnice. Poloha obrázku rovnice se jednoduše dopočte pomocí pozice veličiny, která je tímto vzorcem upřesněna. Výsledný vykreslený graf může mít vzhled například, jak je uvedeno na obrázku (Obrázek 6.2).



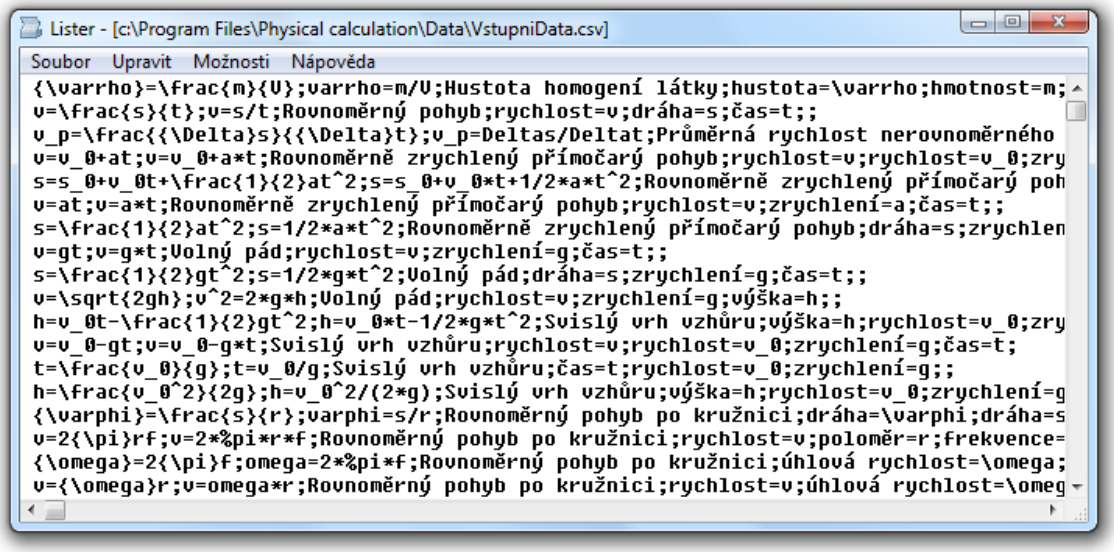
Obrázek 6.2 Vizuální zobrazení grafu (výpočet předjíždění automobilů)

7 Vstupní fyzikální data

Program potřebuje k běhu vstupní data, jež bude uživateli nabízet. Ty jsou uložena ve dvou různých souborech typu *.csv*. Z důvodu velmi malého množství vstupních dat nemá téměř význam ukládání do databáze. Obsah obou souborů se při spuštění programu načte do paměti. Přístup k datům je pak velice rychlý a aplikace není zbytečně zpomalována častým čtením dat z disku.

7.1 Fyzikální vzorce

Při vytváření výpočtového grafu jsou k tvorbě zapotřebí fyzikální vzorce. Tyto vzorce jsou uloženy v souboru *\Data\VstupniData.csv*. Z koncovky souboru je na první pohled zřetelné, že se jedná o formát, kde jsou jednotlivá data oddělena středníkem. Ukázka části souboru je na obrázku (Obrázek 7.1).



Obrázek 7.1 Vstupní data s fyzikálními vzorci

V souboru jsou středníkem odděleny různé druhy informací, které musejí dodržovat následující pořadí:

vzorec TeX; vzorec pro výpočet; popis vzorce; název veličiny=značka veličiny; …

- **vzorec TeX**

  Vzorec fyzikální rovnice ve formátu sázecího systému TeX. Z tohoto vzorce jsou generovány obrázky fyzikálních rovnic, které se zobrazují uživateli.

- **vzorec pro výpočet**

  Vzorec fyzikální rovnice ve formátu, jež je běžně používán v matematických výpočtech. S tímto vzorcem se bude počítat vytvořená soustava rovnic pro výpočet fyzikální úlohy.

- **popis vzorce**

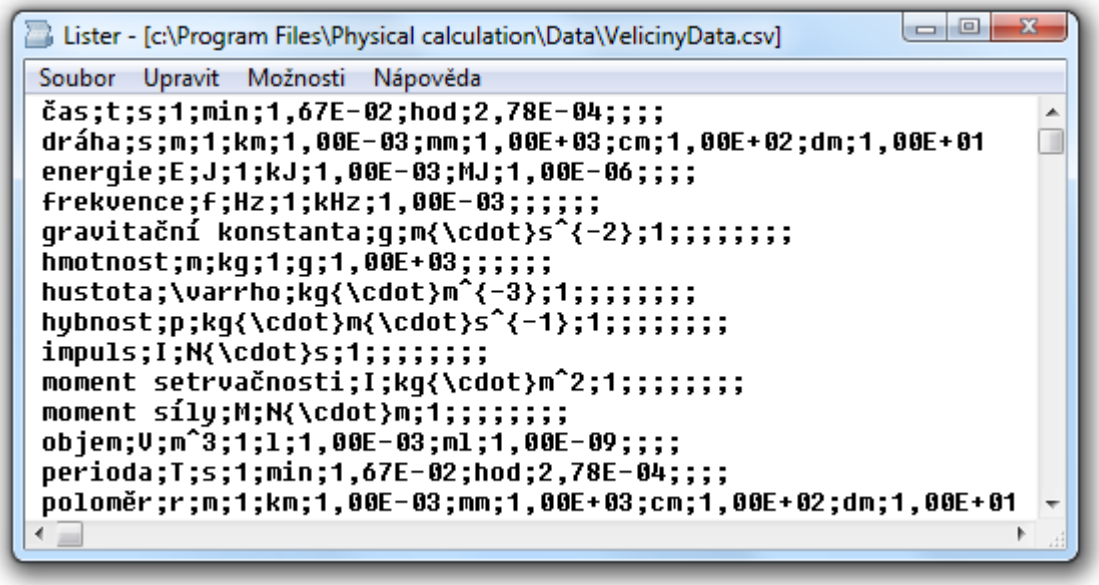
  Popis, který charakterizuje daný fyzikální vzorec nebo jeho fyzikální použití. Pomocí tohoto textu bude uživatel vybírat požadovaný fyzikální vzorec. Je tedy nutné, aby byla informace stručná a výstižná.

- **název veličiny=značka veličiny**

  Dvojce názvu a značky veličiny. Značka je taková, jež se vyskytuje ve vzorci TeX, protože z ní bude tvořen obrázek. Ten se poté bude vkládat do výpočtového grafu a bude zobrazen uživateli. Název je slovní popis fyzikální veličiny a bude se u této veličiny zobrazovat.

7.2 Fyzikální jednotky

Při zadávání hodnoty je možné vybrat ze seznamu požadovanou jednotku fyzikální veličiny. Po zadání hodnoty je poté možné provádět převody mezi jednotkami pouhým zvolením jiné jednotky. Každá fyzikální veličina má přiřazený seznam jednotek i s převodními poměry. Ukázka části souboru je na obrázku (Obrázek 7.2).



Obrázek 7.2 Vstupní data s fyzikálními jednotkami

V souboru jsou středníkem odděleny různé druhy informací, jež musejí dodržovat následující pořadí:

název veličiny; značka veličiny; značka jednotky; převodní vztah; …

- **název veličiny**

  Název je slovní popis fyzikální veličiny a bude se u této veličiny zobrazovat.

- **značka veličiny**

  Značka veličiny je zapisována ve formátu sázecího systému TeX, protože z ní bude tvořen obrázek. Ten se poté vkládá do výpočtového grafu a bude zobrazován uživateli.

- **značka jednotky**

  Značka jednotky je zapisována, stejně jako značka veličiny, ve formátu sázecího systému TeX, protože z ní bude tvořen obrázek. Pomocí něhož bude uživatel vybírat požadovanou jednotku fyzikální veličiny.

- **převodní vztah**

  Převodní vztah je číselná hodnota, jež značí převod mezi hlavní jednotkou a aktuální jednotkou dané fyzikální veličiny. Číselná hodnota se zapisuje ve formátu *1,00E±00*.

8 Používaný software třetích stran

Program Fyzikální výpočty využívá při běhu dvě již existující aplikace. Jedná se o program pro numerické výpočty, který se jmenuje Maxima a program pro tvorbu rastrových obrázků rovnic, který se jmenuje MimeTeX.

8.1 Maxima 5.22.1

Maxima je svobodný počítačový algebraický systém, napsaný v Lispu (resp. jeho dialektu Common Lisp) a distribuovaný pod GNU General Public License. Je dostupný pro všechny platformy standardu Posix, jakými jsou Unix, BSD nebo Linux. Dostupné jsou také binární soubory pro MS Windows. wxMaxima je multiplatformní verzí s grafickým uživatelským rozhraním, založenou na wxWidgets.

8.1.1 Historie

Systém MacSyma, jak je uvedeno v (8), byl vytvořen v průběhu let 1968 až 1982 jako součást projektu MAC v MIT (Massachusetts Institute of Technology). V roce 1982 byl zdrojový kód systému Maxima předán Oddělení energie (Department of Energy). Tato verze je známa jako DOE MacSyma. Program byl poté udržován profesorem Williamem F. Schelterem z univerzity v Texasu, a to až do jeho smrti v roce 2001. V roce 1998 získal Schelter souhlas ke zveřejnění zdrojového kódu programu DOE MacSyma pod veřejnou licencí GNU a v roce 2000 inicializoval projekt Maxima na SourceForge, aby se program DOE MacSyma mohl nadále udržovat a vylepšovat pod svým novým názvem Maxima. Od té doby prochází program pravidelnými aktualizacemi.

8.1.2 Řešení soustavy rovnic

- **Zadání příkladu**

Jakou rychlostí dopadne těleso při volném pádu z výšky 1 metr? Při výpočtu využijte zákona zachování energie.

- **Řešení příkladu**

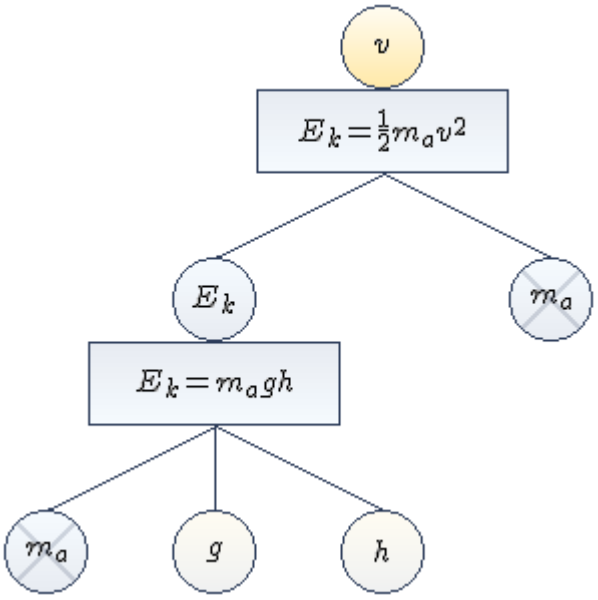
Rychlost tělesa těsně před dopadem bude udávat velikost kinetické energie tělesa. Při využití zákona zachování energie platí:

$$E_{k1} + E_{p1} = E_{k2} + E_{p2}$$

Pokud se vhodně zvolí nulová potenciální energie při dopadu, bude výsledný vzorec vypadat takto:

$$E_{p1} = E_{k2}$$

Kinetická energie při dopadu tělesa se tedy bude rovnat potenciální energii na začátku volného pádu.



Obrázek 8.1 Grafické znázornění vzorců pro výpočet

Z uvedeného obrázku (Obrázek 8.1) se získá soustava rovnic, jejímž vyřešením se vypočte požadovaná rychlost tělesa. Rovnice pro výpočet jsou následně uvedeny v (Rovnice 8.1).

$$E_k = \frac{1}{2} m_a v^2$$

$$E_k = m_a g h$$

$$g = 9.81$$

$$h = 1$$

Rovnice 8.1 Soustava rovnic

Nyní je již známa soustava rovnic pro zadanou slovní úlohu a je tedy možné vyřešit tuto soustavu pomocí programu Maxima. Jako vstup je nutné uvést řetězec v přesném tvaru

solve([rovnice1, rovnice2, …],[veličina1, veličina2, …]), numer;

- solve

  Příkaz, který určí programu Maxima, že se bude počítat soustava rovnic, které mohou být jakéhokoliv typu. Například tedy lineární, kvadratické, logaritmické, trigonometrické a tak dále.

- rovnice1, rovnice2, …

  Seznam rovnic, které se mají řešit. Jsou odděleny čárkami a jejich tvar je shodný s běžným zápisem.

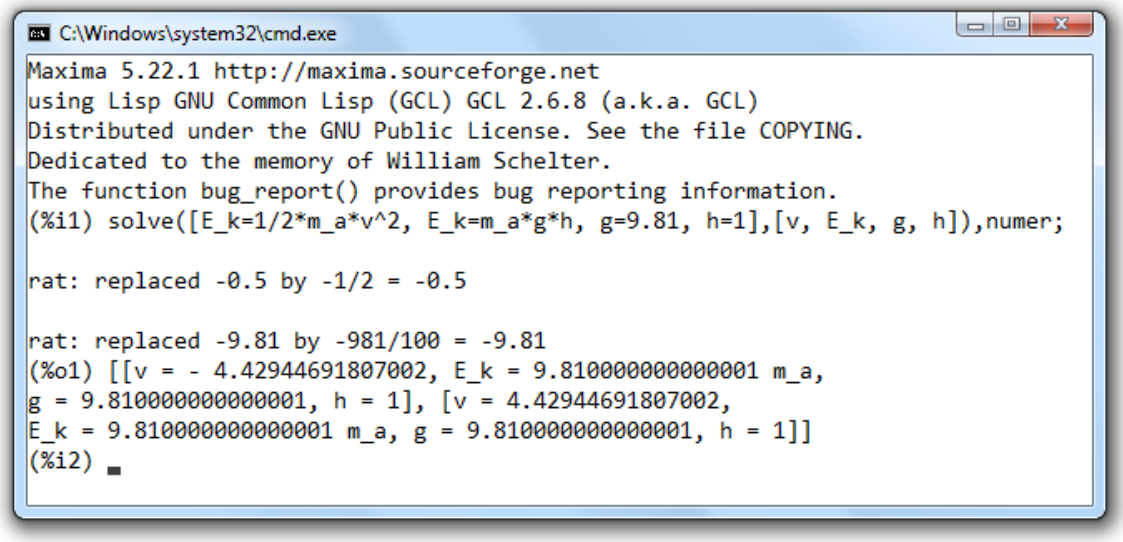
- veličina1, veličina2, …

  Seznam veličin, jejichž hodnotu má program spočíst. Jsou odděleny čárkami a ve značce veličiny se mohou objevovat téměř všechny znaky, kromě znaků užívaných pro matematické operace.

- numer

  Příkaz, jehož význam je ten, že výsledky řešení soustavy rovnic budou v numerické podobě. Další možná a základní podoba je, že výsledky jsou interpretovány pomocí textového výstupu do tvaru zlomků, což je zajisté přehledné, ale pro programové zpracování nevhodné.

Po zadání vstupu v přesně definovaném tvaru zahájí program Maxima numerický výpočet. Výsledek je vždy uvozen značkou *(%oČÍSLO)*. Pokud existuje více řešení, jsou uvedeny v oddělených hranatých závorkách. Ukázka zadaného příkladu v programu Maxima je znázorněna na obrázku (Obrázek 8.2).



Obrázek 8.2 Ukázka řešení soustavy v programu Maxima

Jak je vidět na obrázku (Obrázek 8.2), výsledkem soustavy rovnic jsou dvě možná řešení. Liší se pouze v hodnotě rychlosti *v*, která je buďto kladná, nebo záporná. V tomto příkladu je správné řešení pouze to s kladnou rychlostí a tudíž toto

$$[v = 4.429,\ E\_k = 9.81\ m\_a,\ g = 9.81,\ h = 1]$$

Hodnota energie není číselná, neboť závisí ještě na hmotnosti, která není zadána a nemůže být proto určena.

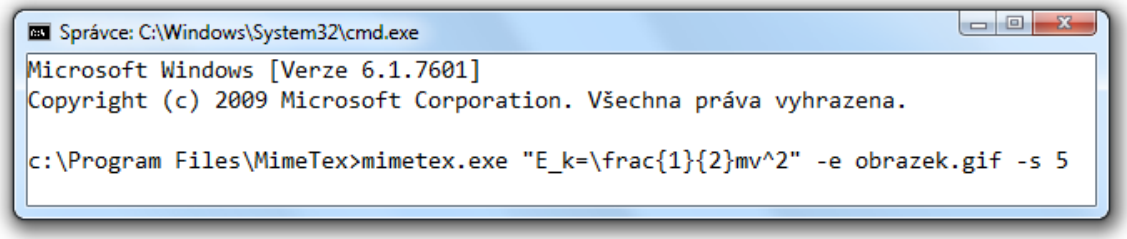
8.2 MimeTeX 1.70

MimeTeX je konzolová aplikace, jež přeloží výrazy LaTeXu a vytvoří z nich obrázky GIF bez nutnosti konverze dvi na gif. Je to samostatný program, který nepoužívá TeX.

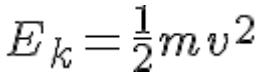
V programu Fyzikální výpočty je tento program využíván pro tvorbu obrázků fyzikálních vzorců, obrázků fyzikálních značek a obrázků fyzikálních jednotek.

8.2.1 Použití

Pro tvorbu obrázku je nutné spustit program MimeTeX z příkazové řádky s uvedením definovaných parametrů. Příklad vytvoření obrázku je uveden dole na obrázku (Obrázek 8.3) a výstup na obrázku (Obrázek 8.4).



Obrázek 8.3 Vytvoření obrázku programem MimeTeX



Obrázek 8.4 Vytvořený obrázek

Parametry předávané programu jsou:

- **výraz LaTeXu**

  Například: "E_k=\frac{1}{2}mv^2"

  Výraz musí být uveden v uvozovkách a musí splňovat veškeré náležitosti definované programem LaTeX.

- **název obrázku**

  Například: -e obrazek.gif

  Za parametrem *–e* je nutné uvést cestu pro uložení a jméno výsledného obrázku ve formátu gif.

- **velikost obrázku**

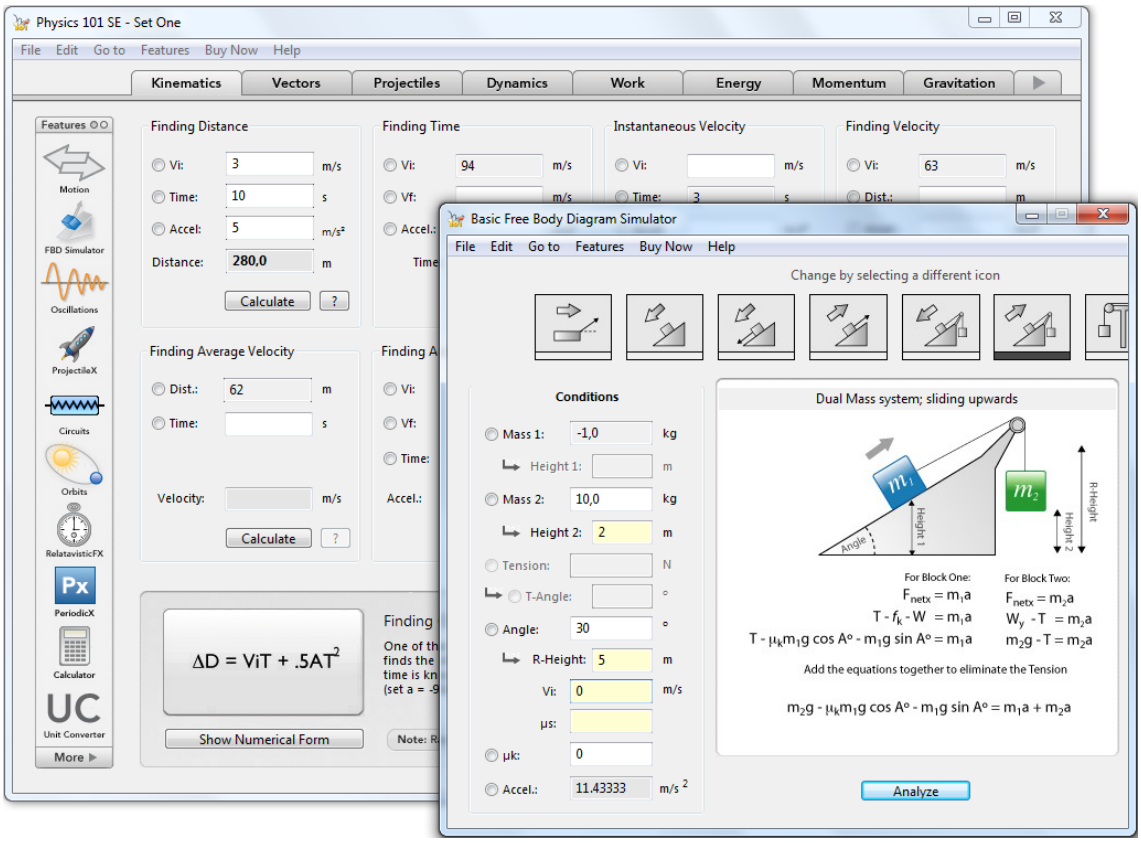
  Například: -s 5

  Za parametrem –s je nutné uvést velikost výsledného obrázku, která je v rozmezí 1 až 7.

9 Výpočetní software pro fyziku

9.1 Physics 101 SE 8.0

Program byl vydán v září 2010 firmou Praeter Software.



Obrázek 9.1 Vzhled programu Physics 101 SE 8.0

9.1.1 Popis

Jedná se o software, který umožňuje studentům řešit mnoho jednoduchých fyzikálních úloh a i některé typy složitějších, jak je uvedeno v (9). Obsahuje více než 150 předdefinovaných nejčastěji používaných fyzikálních vzorců.

Při řešení jednoduchých úloh, kdy je potřeba pouze jeden fyzikální vzorec, stačí vybrat v horních záložkách daný typ úlohy, ze seznamu najít vzorec, který potřebuji, a vyplnit neznámé hodnoty.

Pokud je potřeba vyřešit komplexnější a složitější úlohu, je zapotřebí v levé části okna vybrat daný typ fyzikální úlohy. Otevře se okno se seznamem typových úloh z dané fyzikální problematiky a je zapotřebí najít danou úlohu, kterou máme řešit. Seznam úloh ovšem není příliš rozsáhlý, a proto pokud není daný typ úlohy řešen, nemáme možnost jak danou úlohu vyřešit.

- **Analýza pohybu**

  Analýza pohybu zobrazuje graf závislosti pozice na čase, rychlosti na čase a zrychlení na čase. Přetažením kurzoru nad grafem se zobrazují aktuální hodnoty v daném čase. Takto analyzovat je možné až čtyři různé pohybující se tělesa zároveň.

- **Newtonův zákon**

  Obsahuje sedm nejčastějších problémů aplikace Newtonova zákona s podrobnými nákresy a popisy. Příklady řešených problémů jsou například pohyb po vodorovné rovině, nakloněné rovině a kladky.

- **Pohybové zákony, vrhy**

  Pomocí vyplnění neznámých veličin je možné nadefinovat libovolný vrh vzhůru, volný pád, svislý vrh, nebo jakoukoliv kombinaci těchto třech základních. Výsledkem výpočtu je konečná rychlost a doba vrhu. Výpočty jsou tedy omezeny pouze na tyto dvě fyzikální veličiny a ostatní počítat nelze.

- **Elektrické obvody**

  Do předdefinovaného elektrického obvodu lze přidávat zdroje napěti, rezistory a ampérmetry. Po zadání vlastností zdrojů

napětí a rezistorů se vypočítá napětí, proud a výkon každého rezistoru. U ampérmetru je vypočtena hodnota napětí procházející danou součástkou.

- **Speciální teorie relativity**

  Jsou k dispozici až tři vztažné soustavy, jejichž vlastnosti je možné nadefinovat zadáním fyzikálních veličin. Po provedení výpočtu je možné získat relativisticky složenou rychlost a hybnost, dilataci času a kontrakci délky.

- **Optika**

  V optice je znázorněno lámání paprsku při přechodu z jednoho prostředí do druhého pomocí Snellova zákona lomu a odrazu. Interaktivně je ukázáno, jak se paprsky světla mění při změně indexu lomu a úhlu dopadu. Není ovšem možné specifikovat více než dvě prostředí, kterými paprsek prochází.

- **Termodynamika**

  Při termodynamických výpočtech se počítá výměna tepla mezi dvěma tělesy, které se vzájemně dotýkají. Je nutné zadat počáteční teploty obou těles a konstanty, které charakterizují daný materiál. Není ovšem možné počítat výměnu tepla mezi více jak dvěma tělesy.

- **Pohyby těles ve vesmíru**

  Počítají se rychlosti a graficky znázorňují dráhy těles při pohybu ve vesmíru. Při výpočtu je možné brát v úvahu i odpor vzduchu, pokud je uživatelem zadán.

9.1.2 Výhody

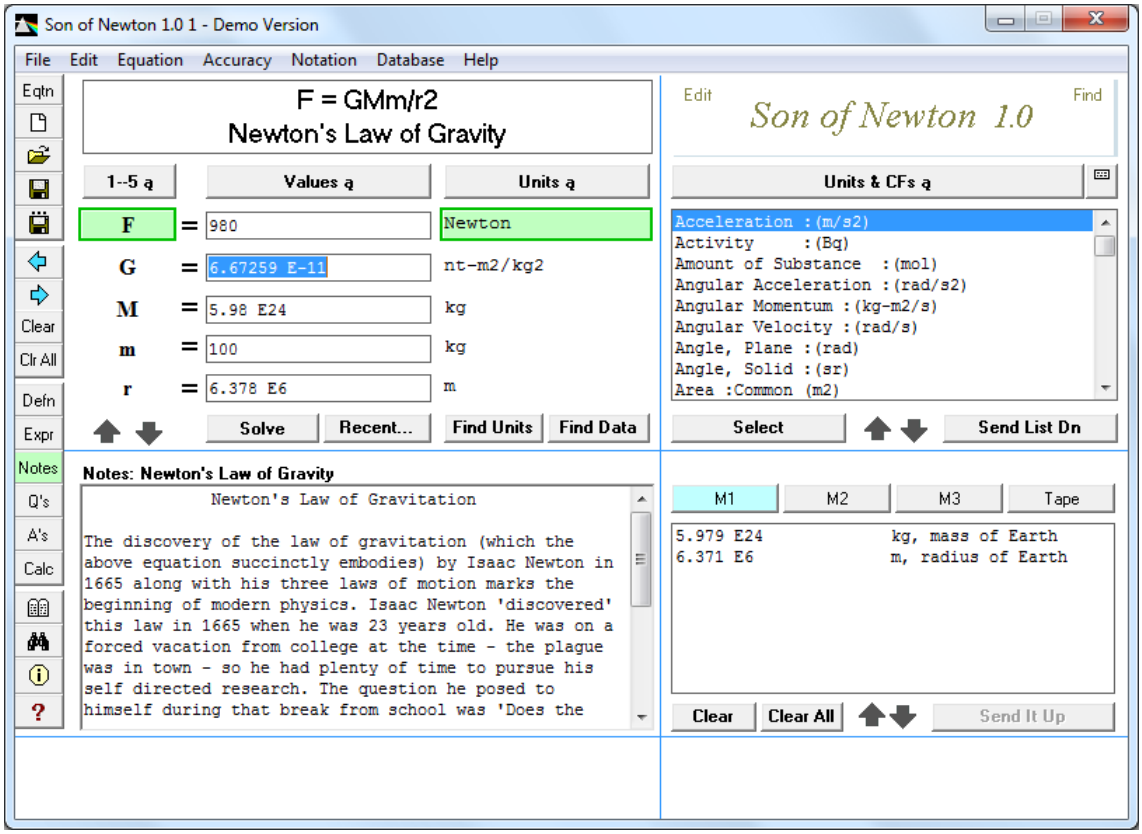
- Ovládání programu je jednoduché.
- Některé výpočty jsou doplněny nákresy a postupem výpočtu.

9.1.3 Nevýhody

- Lze řešit pouze typové úlohy, které byly implementované v programu. Nelze tedy řešit složitější a neobvyklé fyzikální úlohy.
- Při zadávání hodnot není možné vybírat jednotky fyzikální veličiny, neboť jsou již pevně nastaveny a nelze je změnit.

9.2 Son of Newton 1.01

Program byl vydán v červenci 2002 společností PhysicSoft.



Obrázek 9.2 Vzhled programu Son of Newton 1.01

9.2.1 Popis

Program umožňuje řešit fyzikální rovnice, které jsou již součástí programu nebo je možné rovnice editovat a přidávat. V plné verzi programu je přes 220 rovnic, které by měly pokrývat probíranou látku fyziky až do prvního ročníku vysoké školy.

Velký důraz je kladen na to, aby uživatel mohl editovat a přidávat fyzikální rovnice a k nim připisovat poznámky. Dále je možné přidávat a editovat fyzikální jednotky s jejich převody. Veškeré tyto úpravy se ukládají do databáze a při příštím spuštění programu jsou opět k dispozici.

Výpočet rovnice je možné provádět i bez zadání fyzikálních jednotek. V tomto případě jsou použity jednotky s převodním poměrem rovným jedné. V případě zadání jednotek jsou hodnoty převedeny do základních jednotek a poté je rovnice vypočtena.

Další funkce, kterou program obsahuje je používání kalkulačky, jejíž vzhled a funkčnost jsou podobné kalkulačce z operačního systému Microsoft Windows.

9.2.2 Výhody

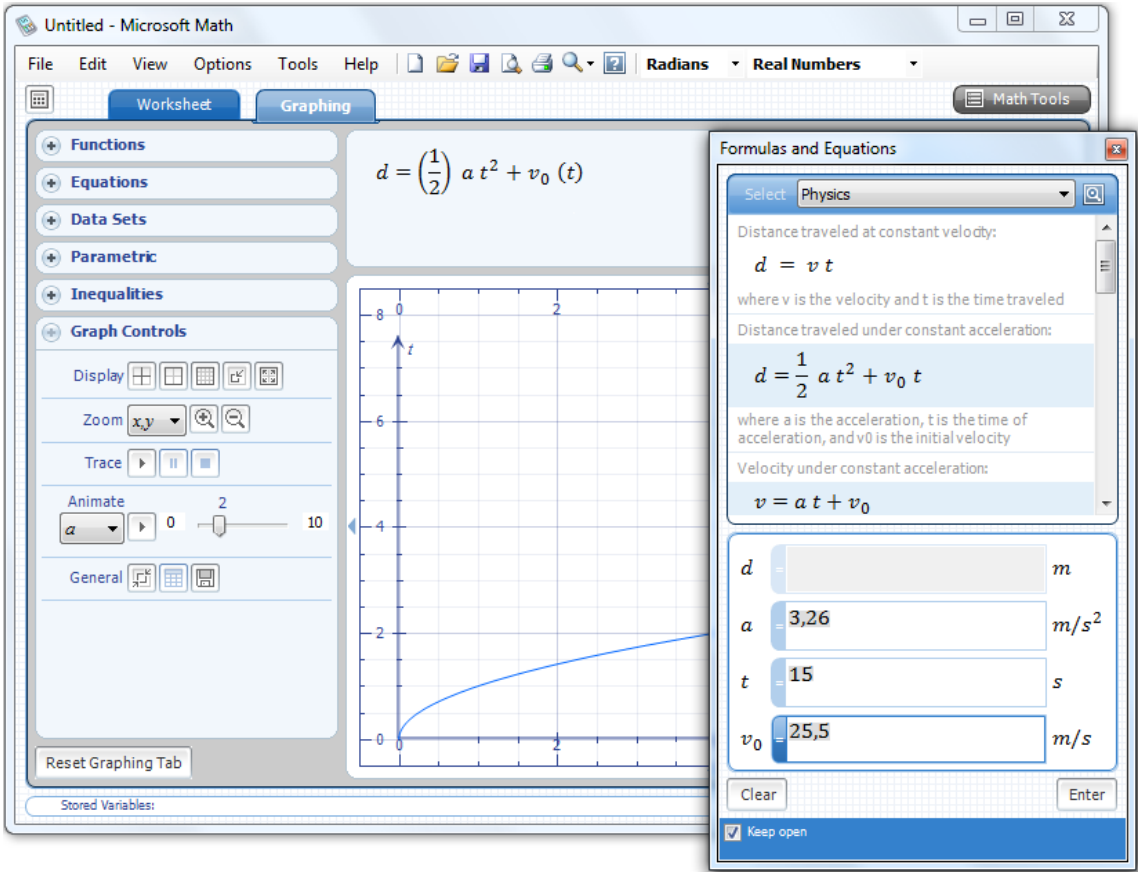
- Možnost editace databáze fyzikálních rovnic a fyzikálních jednotek.

9.2.3 Nevýhody

- Není možné řešit soustavy rovnic, ale pouze každou rovnici zvlášť.
- Ovládání je nepřehledné a příliš zdlouhavé.

9.3 Microsoft Math 3.0

Program byl vydán v roce 2006 společností Microsoft.



Obrázek 9.3 Vzhled programu Microsoft Math 3.0

9.3.1 Popis

Program umožňuje řešení soustav rovnic v početní podobě nebo i v grafické podobě. Prvotně není program určen pro fyzikální výpočty, ale pro výpočty matematické. Z tohoto důvodu je v databázi programu uloženo velmi malé množství fyzikálních vzorců.

Při řešení fyzikální úlohy je možno vzorce vkládat ze seznamu, napsat pomocí klávesnice, nebo za užití tabletu vzorce napsat ručně. Každá fyzikální veličina, která je uložena v databázi má přiřazené fyzikální jednotky.

Výpočet soustavy rovnic je možné provádět klasicky početně nebo grafickým způsobem. Při početním způsobu jsou výsledky zobrazeny jako číselné hodnoty s jednotkami. Při využití grafického způsobu výpočtu soustavy rovnic se vykreslí příslušný graf a uživatel je schopen si představit průběh fyzikálního jevu a důležitost jednotlivých veličin a jejich hodnot.

9.3.2 Výhody

- Uživatelsky jednoduché ovládání a přehledné grafické prostředí.
- Možnost zadávat fyzikální vzorce kreslením.

9.3.3 Nevýhody

- Velmi malé množství definovaných fyzikálních rovnic.
- Při řešení soustavy, je seznam rovnic nepřehledný a uživatel nemá dostatečný přehled o tom, jaký fyzikální vzorec patří k jaké fyzikální veličině.

10 Závěr

Závěrem této bakalářské práce zhodnotím výsledky, kterých se podařilo dosáhnout při vývoji grafického rozhraní pro fyzikální výpočty.

Pomocí naprogramovaného programu je možné vyřešit libovolně složité slovní úlohy z fyziky probírané na středních školách. Lze vypočítat příklady, jež vedou k řešení pomocí dosazovací metody a dokonce i ty, které vedou k řešení pomocí soustavy lineárních i nelineárních rovnic. Výsledný výpočet je velmi rychlý a uživatel není nucen zdlouhavě čekat na dopočtení úlohy.

Výhodou řešení pomocí vytvořeného programu je, že výsledné rovnice jsou přehledně znázorněny ve výpočtovém grafu. Díky tomu je program vhodný i pro názornou ukázku při výuce fyziky na středních školách. Program je intuitivní a jednoduchý na ovládání. Je zaměřen na lepší pochopení probírané fyzikální látky a ne pouze na učení se vzorců zpaměti. Mám za to, že i studenti, kteří jsou méně technicky nadaní, dokáží zajisté s pomocí mého názorného programu probíranou látku zvládnout a pochopit.

Program byl v průběhu vývoje otestován studenty předmětu Základy počítačové grafiky na FAV ZČU, kteří po otestování vyplnili hodnotící dotazník. Díky tomu bylo vylepšeno ovládání, funkčnost a drobné chyby, jež se v programu nacházely.

Do budoucna je možné program pro fyzikální výpočty zajisté dále rozšířit. Možné vylepšení je v umožnění řešit fyzikální úlohy probírané na vysokých školách, neboli začlenit i diferenciální a integrální počty. Další možností je vytvoření databáze s fyzikálními vzorci a jednotkami, kterou by si mohl uživatel sám pomocí obslužného programu editovat.

Bakalářská práce byla podporována projektem VIRTUAL 2C06002.

11 Přehled zkratek a pojmů

Framework = Softwarová struktura, která slouží jako podpora při programování a vývoji a organizaci jiných softwarových projektů. Může obsahovat podpůrné programy, knihovnu API, návrhové vzory nebo doporučené postupy při vývoji (10).

GNU = Projekt zaměřený na svobodný software, inspirovaný operačními systémy unixového typu (10).

GUI = Grafické uživatelské rozhraní (anglicky Graphical User Interface) je uživatelské rozhraní, které umožňuje ovládat počítač pomocí interaktivních grafických ovládacích prvků (10).

Pixel buffer = Umožňuje renderování obrázku na pozadí.

Posix = Zkratka z Portable Operating System Interface, je přenositelné rozhraní pro operační systémy, standardizované jako IEEE 1003 a ISO/IEC 9945 (10).

X resources = Zdroje zahrnující parametry z počítačových programů, jako je název požívaného písma, barva pozadí v menu, atd.

X Window Systém = V informatice souhrnné označení pro software, které umožňuje vytvořit grafické uživatelské prostředí (GUI). Používá se zejména v unixových systémech, kde se stalo standardem. Využívá model klient-server, skládá se z několika komponent, které jsou navzájem nezávislé (10).

12 Literatura

1. **Běhálek, Marek.** Programovací jazyk C#. *Fakulta elektrotechniky a informatiky, VŠB-TUO.* [Online] [Citace: 13. březen 2011.] http://www.cs.vsb.cz/behalek/vyuka/pcsharp/text/index.html.

2. **Sells, Chris.** *Windows forms programming in C#.* Boston : Addison-Wesley, 2004. ISBN 0-321-11620-8.

3. **RNDr. Kovář, Dušan Ph.D.** Programování Windows Forms pomocí C#. *Programování se zaměřením na .NET a jazyk C#.* [Online] [Citace: 5. duben 2011.] http://projektysipvz.gytool.cz/ProjektySIPVZ/Default.aspx?uid=4.

4. **Ludačka, Radek.** QT framework - pomocník programátora. *SWMag softwarový magazín.* [Online] [Citace: 15. březen 2011.] http://www.swmag.cz/546/qt-framework-pomocnik-programatora/.

5. **Blanchette, Jasmin a Summerfield, Mark.** *C++ GUI Programming with Qt 4, Second Edition.* místo neznámé : Prentice Hall, 2008. ISBN 978-0132354165.

6. **Trolltech.** Qt 4.7: Signals & Slots. *Qt Reference Documentation.* [Online] [Citace: 19. březen 2011.] http://doc.trolltech.com/4.7/signalsandslots.html.

7. **Bodnar, Jan.** Introduction to GTK+. *ZetCode.* [Online] [Citace: 20. březen 2011.] http://zetcode.com/tutorials/gtktutorial/introduction/.

8. **Trefilíková, Zdena.** *Systém počítačové algebry Maxima.* Masarykova univerzita Brno : Bakalářská práce, 2011.

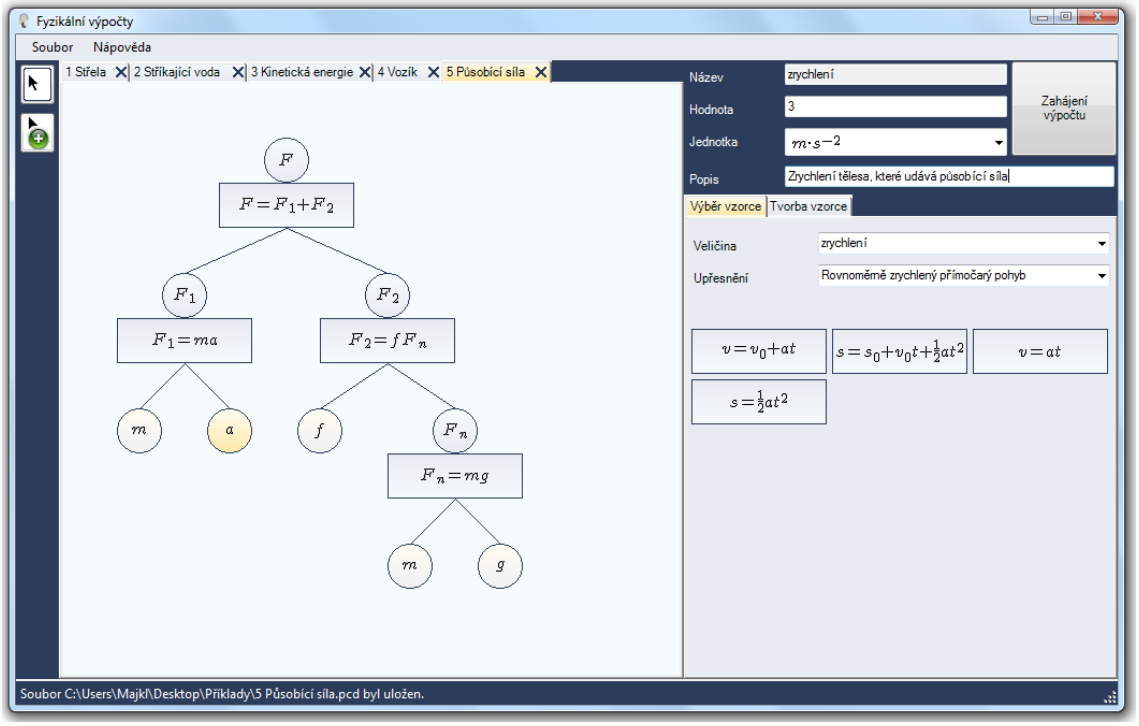
9. **Praeter Software.** Physics 101 SE. *Praeter Software.* [Online] [Citace: 15. prosinec 2010.] http://www.praetersoftware.com/physics/.

10. **Creative Commons, 3.0.** Wikipedie, otevřená encyklopedie. [Online] [Citace: 12. duben 2011.] http://cs.wikipedia.org/.

13 Seznam obrázků

14 Přílohy

A Vzhled programu

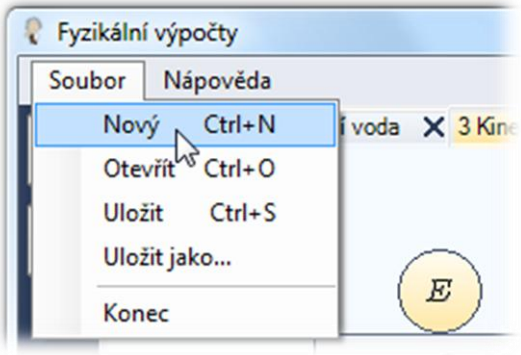


Obrázek A.1 Vzhled programu Fyzikální výpočty

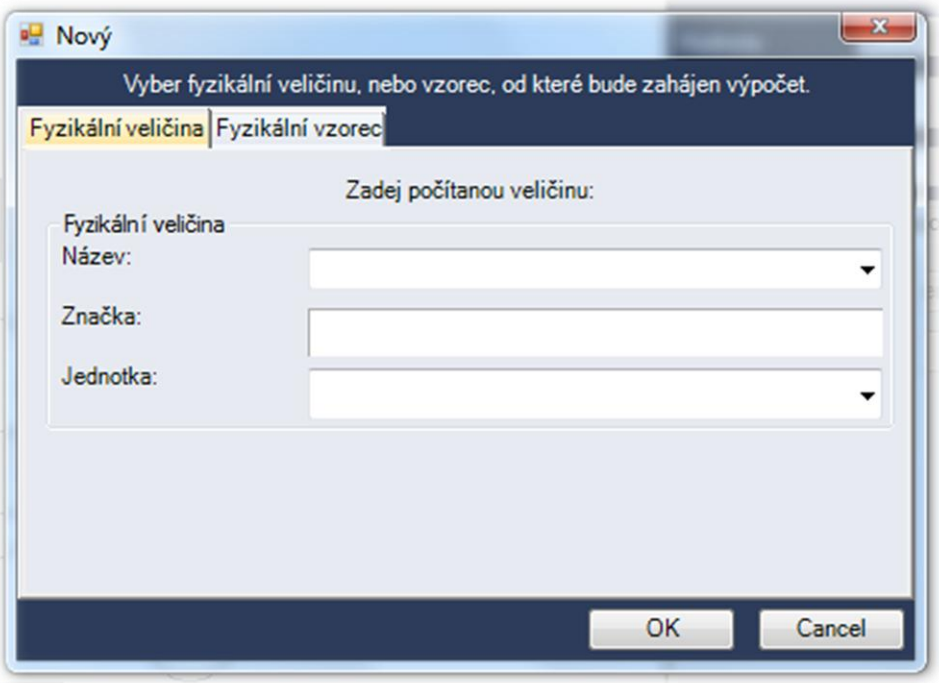
B Uživatelská dokumentace

B.1 Zahájení nového výpočtu

Pro zahájení nového výpočtu je třeba v menu aplikace zvolit *Soubor* → *Nový* nebo stisknout klávesovou zkratku *Ctrl+N*.

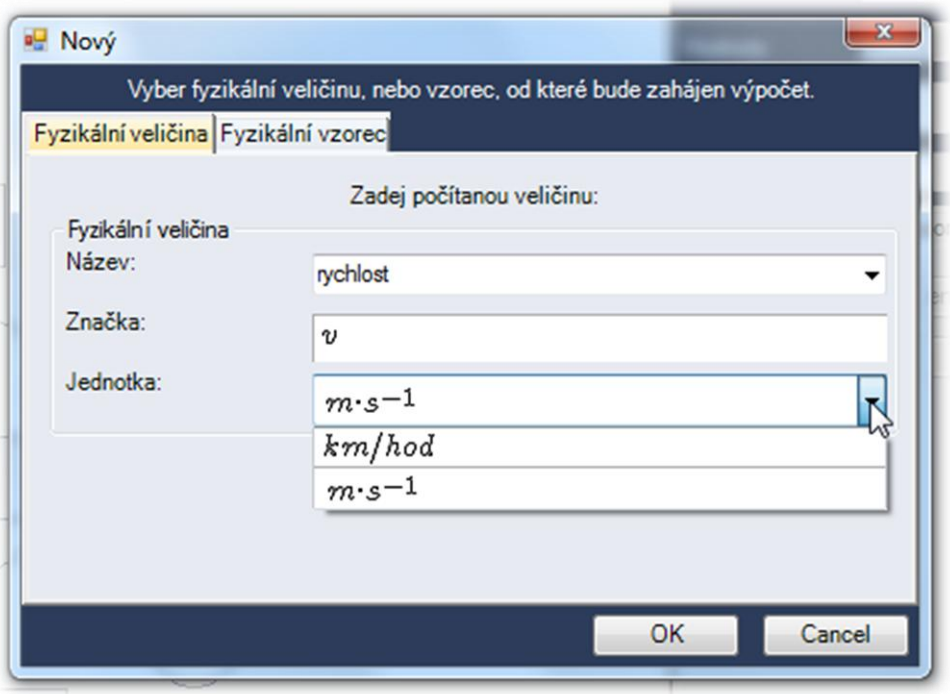


Zobrazí se okno, ve kterém je třeba zvolit, zda bude výpočet zahájen od neznámé veličiny, nebo fyzikálního vzorce.



B.1.1 Fyzikální veličina

Při zvolení zahájení výpočtu od neznámé veličiny je zapotřebí napsat, nebo zvolit název veličiny. Poté se vyplní značka a hlavní jednotka zvolené veličiny. Jednotku je možné změnit pomocí výběru ze seznamu.

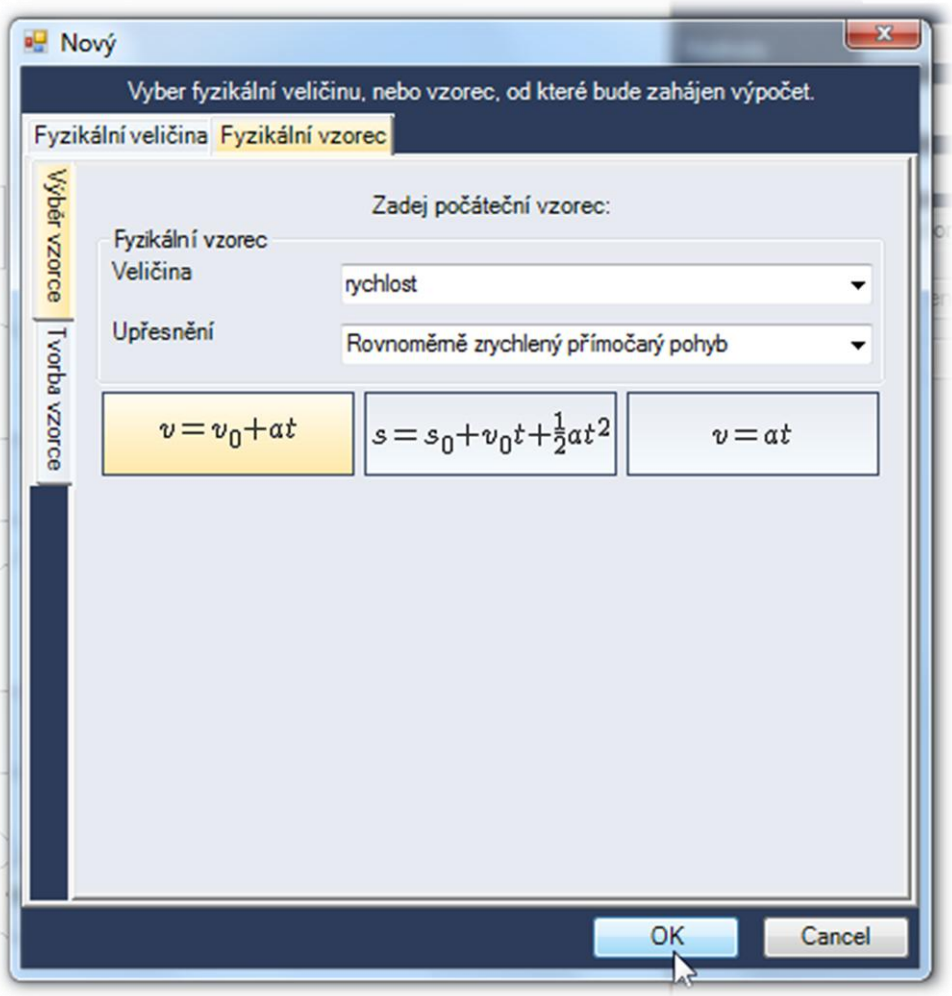


B.1.2 Fyzikální vzorec

Při zvolení zahájení výpočtu od fyzikálního vzorce je zapotřebí vybrat v levé části okna, zda se jedná o existující vzorec, nebo zda bude vzorec vytvořen.

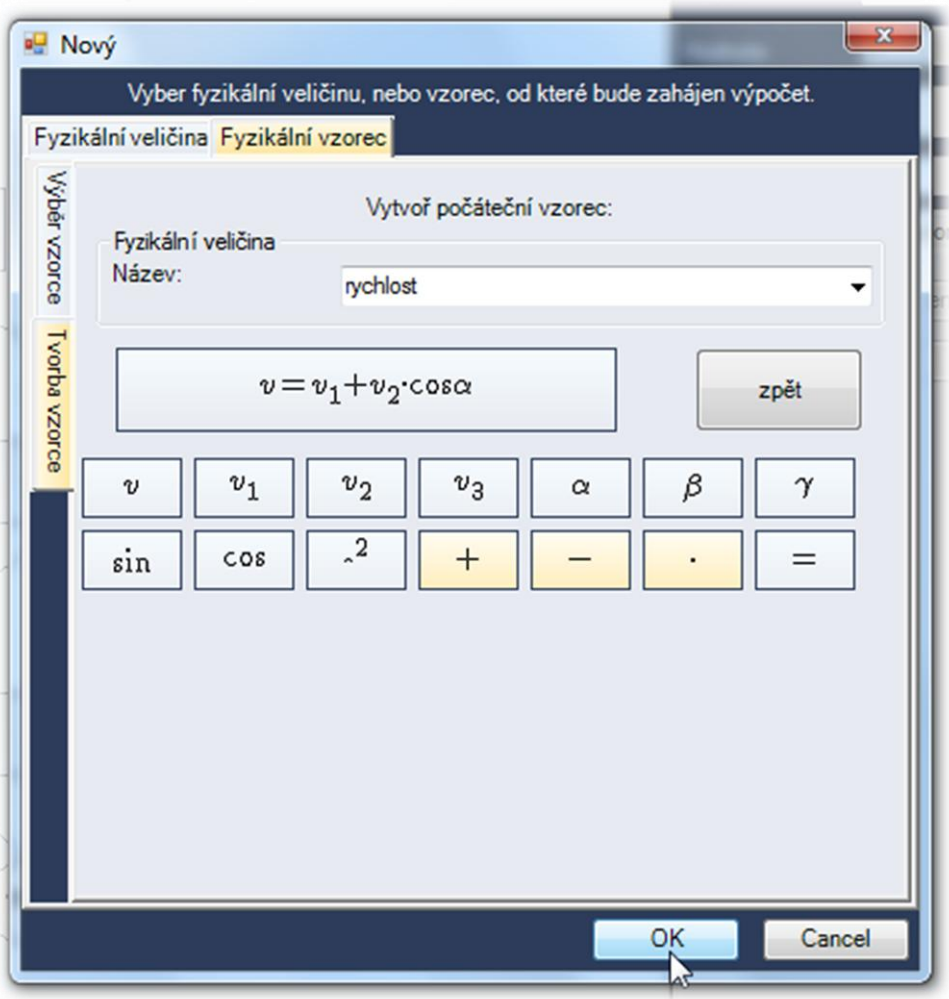
B.1.2.1 Výběr vzorce

Při zvolení výběru vzorce je zapotřebí vybrat fyzikální veličinu, která se v požadovaném vzorci vyskytuje. Poté upřesnit o jaký fyzikální jev se jedná a ze seznamu nabízených vzorců označit ten požadovaný.



B.1.2.2 Tvorba vzorce

Při zvolení tvorby vzorce je zapotřebí vybrat fyzikální veličinu, která se v požadovaném vzorci vyskytuje. Poté se pomocí tlačítek vytvoří požadovaný vzorec. Vytvořený vzorec může obsahovat všechny základní matematické operace, které jsou pro fyziku potřebné.

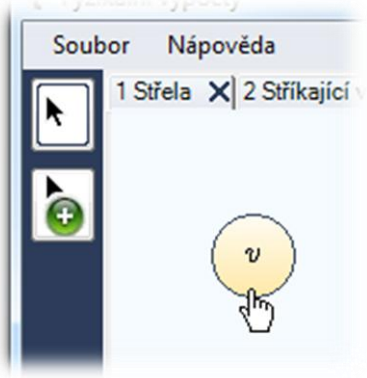


B.2 Vzorce

Ke každé veličině v grafu je možné přidělit vzorec, neboli jí upřesnit, ale naopak je také možné vzorec odebrat.

B.2.1 Upřesnění veličiny

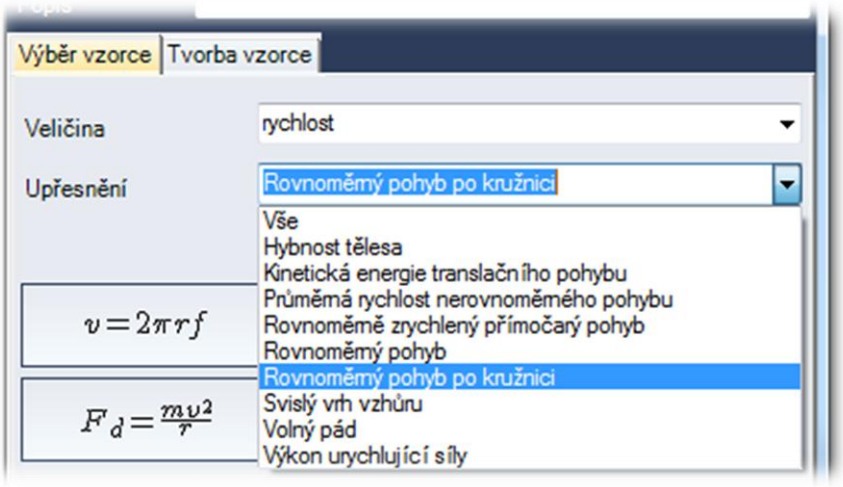
Před upřesňováním veličiny je nutné mít zvolen *základní kurzor pro ovládání*. Tento kurzor se zvolí stisknutím tlačítka s obrázkem myši. Poté je zapotřebí označit upřesňovanou veličinu kliknutím.



Po vybrání veličiny je možné provést upřesnění již existujícím fyzikálním vzorcem, nebo si požadovaný vzorec vytvořit. Volba jedné ze dvou možností se provede zvolením příslušné záložky v pravé části okna.

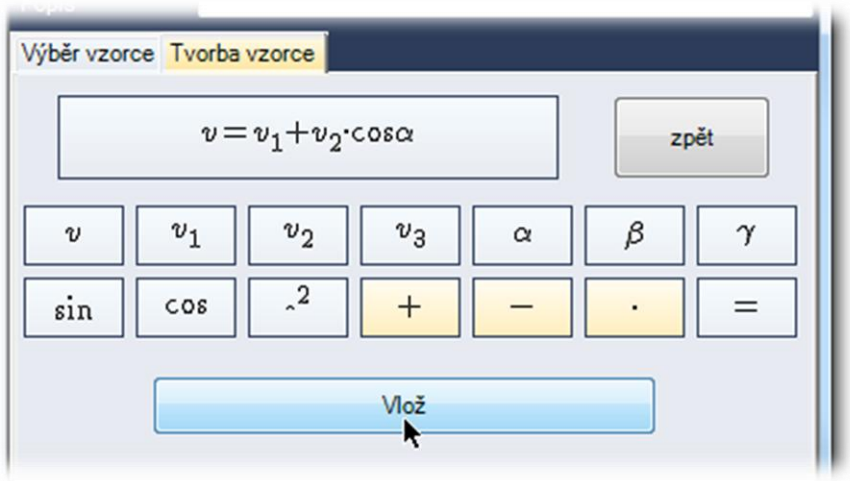
B.2.1.1 Výběr vzorce

Při zvolení výběru vzorce je zapotřebí vybrat fyzikální veličinu, která se v požadovaném vzorci vyskytuje. Ve většině případů bude volba veličiny správně provedena automaticky. Poté je zapotřebí upřesnit o jaký fyzikální jev se jedná a ze seznamu nabízených vzorců kliknout na ten požadovaný. Tím se provede vložení vybraného vzorce do výpočtového grafu.



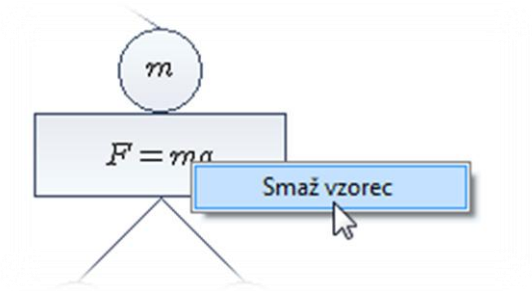
B.2.1.2 Tvorba vzorce

Při zvolení tvorby vzorce se pomocí tlačítek vytvoří požadovaný vzorec. Vytvořený vzorec může obsahovat všechny základní matematické operace, které jsou pro fyziku potřebné. Poté se klikne na tlačítko vložit. Tím se provede vložení vytvořeného vzorce do výpočtového grafu.

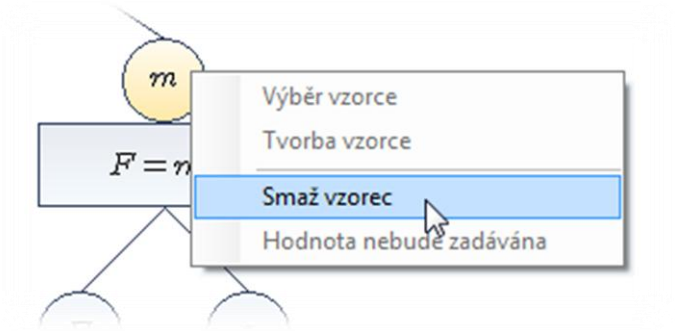


B.2.2 Smazání vzorce

Pro smazání vzorce je zapotřebí kliknout pravým tlačítkem na příslušný vzorec v grafu a vybrat možnost *Smaž vzorec*.



Další možností je kliknout pravým tlačítkem na veličinu v grafu, která je příslušným vzorcem upřesněna, a vybrat možnost *Smaž vzorec*.

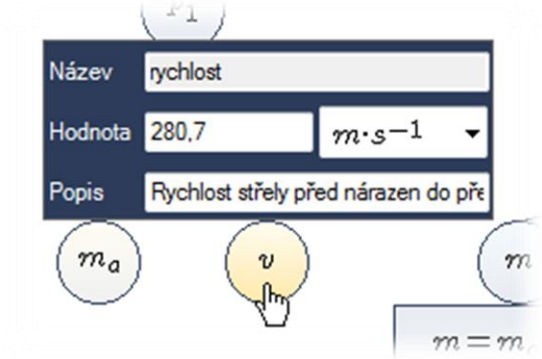


B.3 Vlastnosti veličiny

Každá veličina ve výpočtovém grafu obsahuje název, hodnotu, jednotku a popis. Název veličiny nelze měnit, neboť je definován automaticky. Zbylé tři vlastnosti je možné upravovat a měnit.

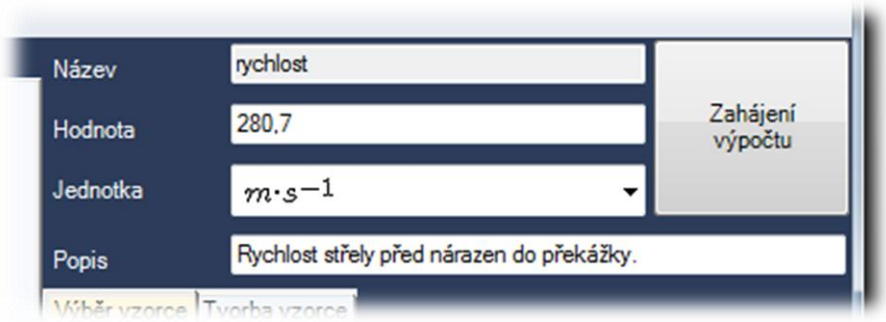
B.3.1 Upravení vlastností v balónovém okně

Po najetí myši nad upravovanou veličinu se po dvou sekundách zobrazí balónové okno s informacemi o dané veličině.



B.3.2 Upravení vlastností v hlavním okně

V pravém horním rohu je zobrazena informační část, ve které se po kliknutí na veličinu v grafu zobrazí informace o vybrané veličině.

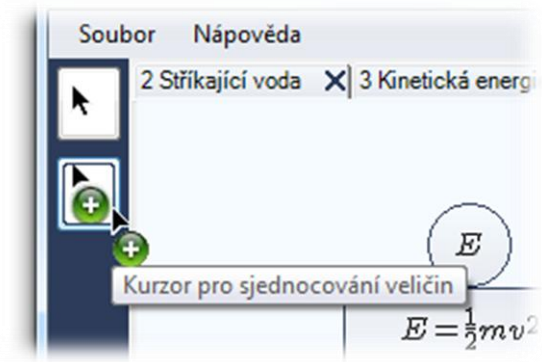


B.4 Sjednocené veličiny

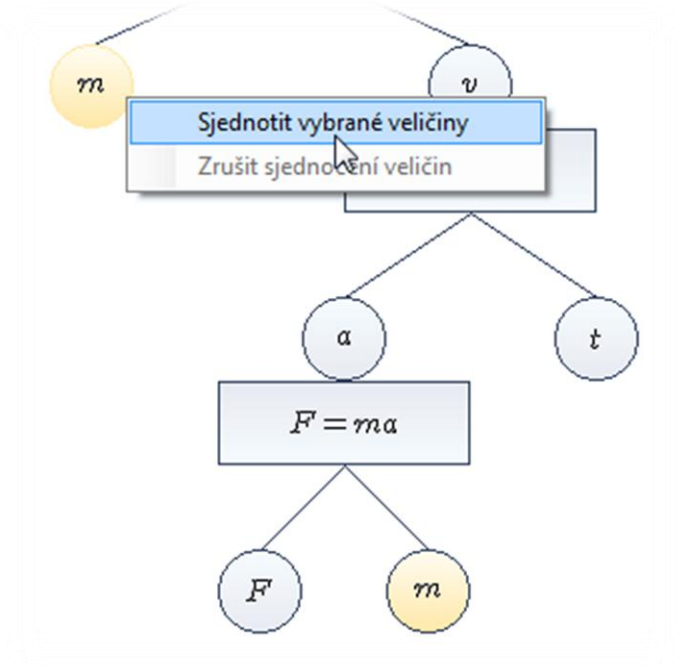
Pokud mají být dvě veličiny shodné, neboli určují stejnou fyzikální vlastnost, je vhodné tyto veličiny sjednotit. Hodnota, jednotka i popis se poté zadávají pouze u jedné a u ostatních je toto automaticky nastaveno shodně.

B.4.1 Sjednocení veličin

Před sjednocováním veličin je nutné mít zvolen *kurzor pro sjednocování veličin*. Tento kurzor se zvolí stisknutím tlačítka, které je uvedeno na obrázku.

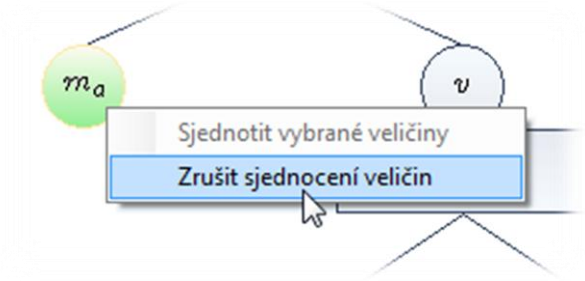


Poté se klikáním označí veličiny, které mají být shodné a pomocí pravého tlačítka myši se v kontextovém menu zvolí *Sjednotit vybrané veličiny*.



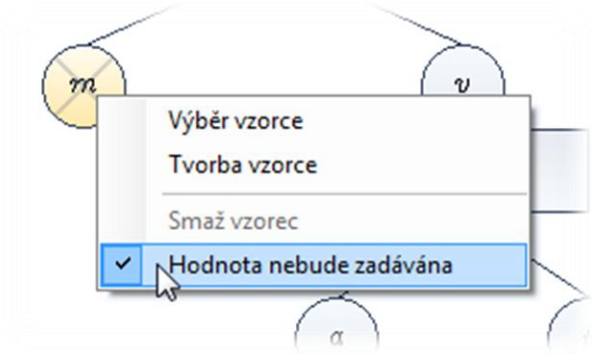
B.4.2 Zrušení sjednocení veličin

Pro zrušení sjednocení veličin je opět nutné mít zvolen *kurzor pro sjednocování veličin*. Poté stačí pouze kliknout pravým tlačítkem na sjednocenou veličinu a zvolit možnost *Zrušit sjednocení veličin*.



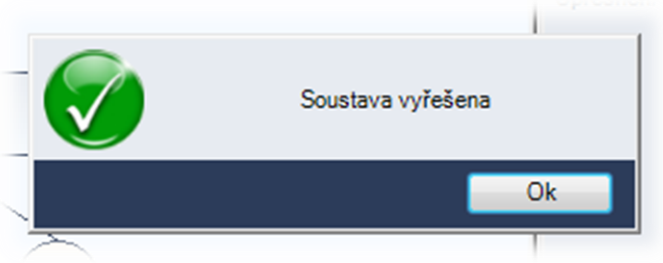
B.5 Nezadávaná hodnota

Pokud se ve výpočtovém grafu vyskytuje veličina, jejíž hodnota není k výsledku potřebná a zároveň není známá, je možné ji označit jako nezadávanou veličinu. Toto se provede pomocí kliknutí pravého tlačítka na veličinu a vybráním možnosti *Hodnota nebude zadávána*.



B.6 Výpočet úlohy

Pokud je již vytvořen kompletní výpočtový graf, je možné přejít k výpočtu úlohy. Numerický výpočet se spustí kliknutím na tlačítko Zahájení výpočtu, které je umístěno v pravém horním rohu. Po úspěšném dokončení numerického výpočtu se zobrazí informační zpráva.



Postupným klikáním na veličiny ve výpočtovém grafu je možné zobrazovat číselné hodnoty dané veličiny. Lze si tak prohlédnou i různé mezivýsledky při výpočtu.

B.7 Ukládání a otevírání souborů

Vytvořený graf pro výpočet fyzikální úlohy lze uložit do souboru a později znovu otevřít. Ukládaný formát má koncovku *.pcd*, která znamená *Physical calculation data*.

B.7.1 Uložení souboru

Ukládání funguje stejně jako ve všech jiných programech. Pro uložení je třeba zvolit *Soubor* → *Uložit jako…* a vybrat místo uložení. Pokud byl soubor již dříve uložen, stačí pouze zvolit *Soubor* → *Uložit*, nebo stisknout klávesovou zkratku *Ctrl+S*.

B.7.2 Otevření souboru

Otevírání souboru funguje stejně jako ve všech jiných programech. Pro otevření je třeba zvolit *Soubor* → *Otevřít*, nebo stisknout klávesovou zkratku Ctrl+O. Poté je nutné vybrat dříve uložený soubor pro otevření.

Pokud se asociují soubory s koncovkou *.pcd* s programem *Fyzikální výpočty*, je možné soubor v otevřít v tomto programu jeho pouhým spuštěním.

C Postup instalace

Před prvním spuštěním programu je zapotřebí nejprve provést několik kroků. Nejprve je zapotřebí zkopírovat program Fyzikální výpočty a poté nainstalovat výpočtový software Maxima.

C.1 Požadavky

Pro nainstalování a spuštění programu Fyzikální výpočty je nutné splňovat následující požadavky:

- operační systém Windows XP SP3, Windows Vista SP1, Windows 7
- .Net Framework 4.0
- minimálně 70Mb volného místa na disku

C.2 Zkopírování souborů

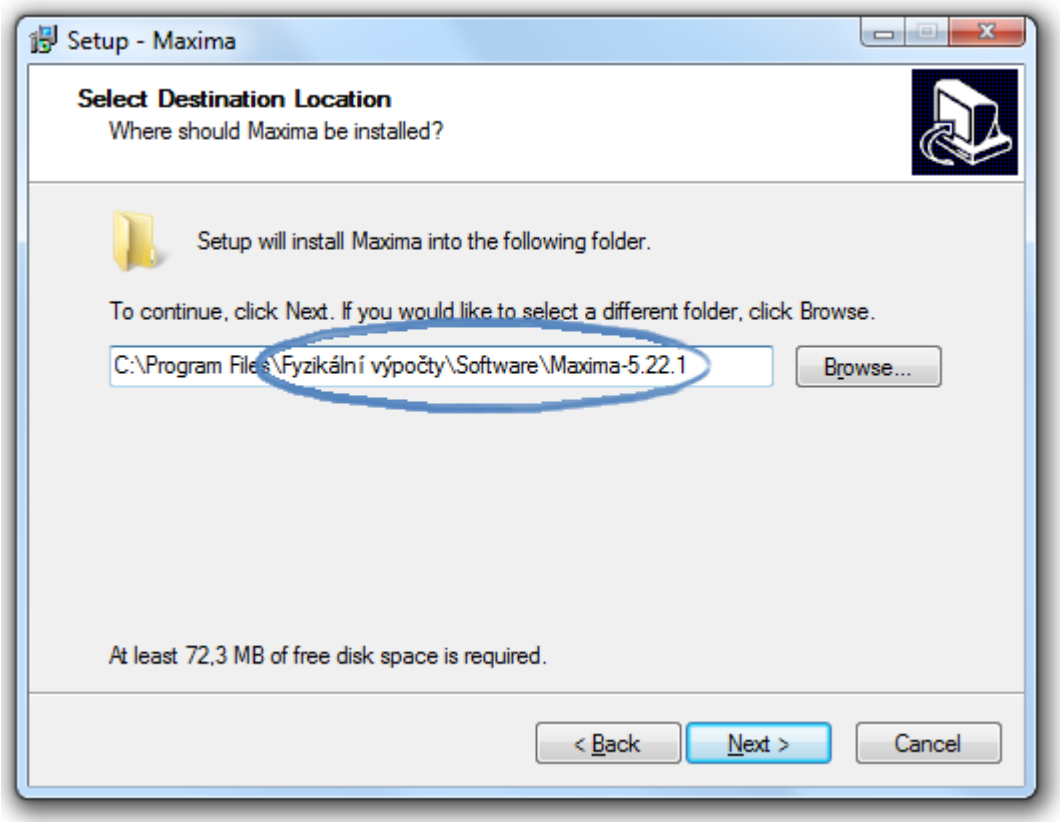
Nejprve je potřeba zkopírovat celou složku *Fyzikální výpočty* do zvoleného umístění na disku, kde budete chtít, aby byla aplikace uložena.

C.3 Instalace programu Maxima

Ke spuštění aplikace je zapotřebí, aby byl nainstalován program Maxima. Tato aplikace je využívána k numerickým výpočtům při řešení slovních úloh.

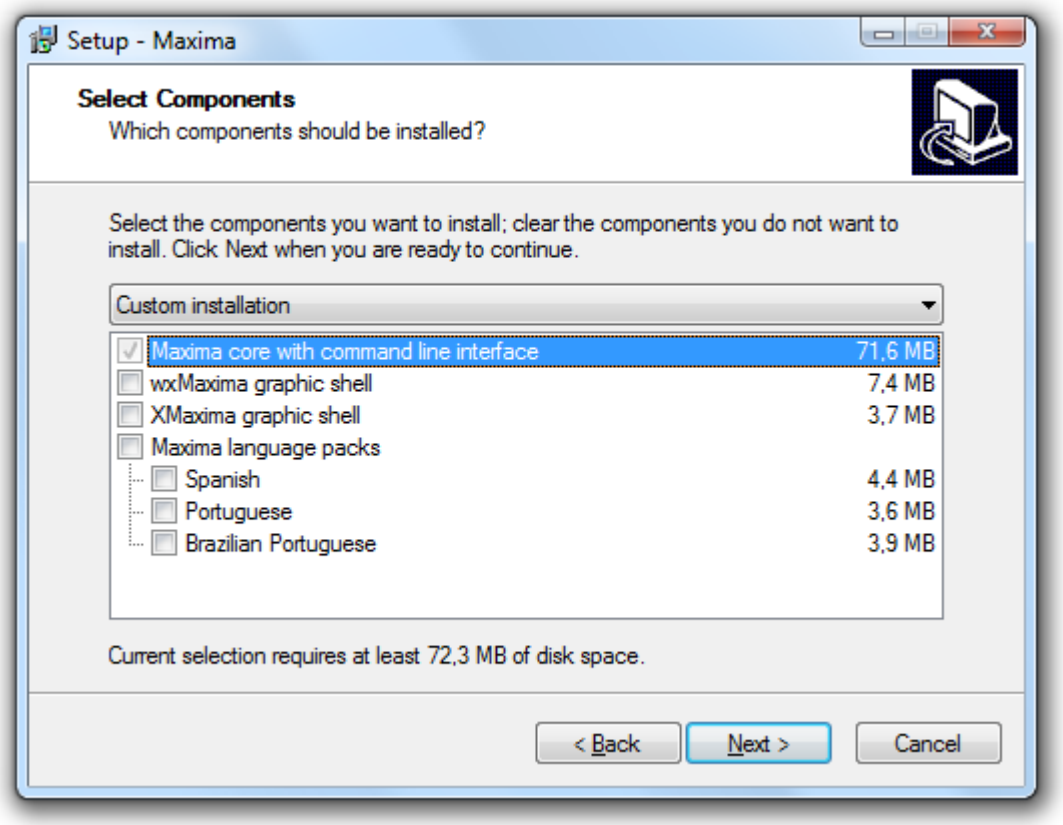
Instalace se zahájí spuštěním souboru *Maxima.exe*. Postupným potvrzováním dotazů se dostanete až do požadavku pro zadání místa instalace. V tomto kroku je důležité zvolit umístění

…\Fyzikální výpočty\Software\Maxima-5.22.1



Obrázek C.1 Volba umístění při instalaci programu Maxima

V dalším kroku se zobrazí dotaz pro volbu komponent, které mají být nainstalovány. Pro plnou funkčnost programu Fyzikální výpočty stačí nainstalovat pouze *Maxima core with command line interface*, jak je uvedeno na obrázku (Obrázek C.2), který je níže.



Obrázek C.2 Volba instalovaných komponent

Po nainstalování výpočtového programu Maxima je již aplikace Fyzikální výpočty připravená ke spuštění.

C.4 Spuštění programu Fyzikální výpočty

Pro spuštění programu Fyzikální výpočty je potřeba otevřít soubor *Fyzikální výpočty.exe*, který se nachází v

…\Fyzikální výpočty\Fyzikální výpočty.exe

D Programátorská dokumentace

Vzhledem k rozsáhlosti je programátorská dokumentace k dispozici na přiloženém CD.