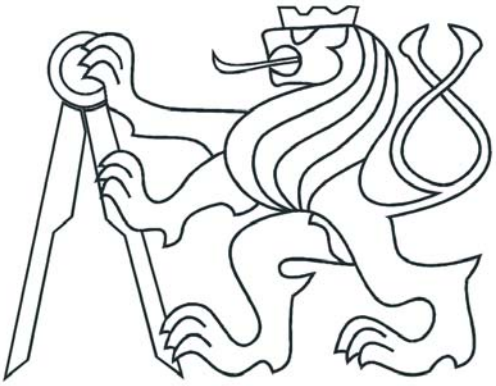
České vysoké učení technické v Praze

Fakulta elektrotechnická



BAKALÁŘSKÁ PRÁCE

Grafické uživatelské rozhraní optimalizačního nástroje

Vypracoval: Michal Orlík

Vedoucí práce: Ing. Libor Waszniowski

Praha, 2007

České vysoké učení technické v Praze, fakulta elektrotechnická

Katedra řídicí techniky Školní rok:2006/2007

Zadání bakalářské práce

Student: Michal Orlík

Obor: Kybernetika a měření

Název tématu: Grafické uživatelské rozhraní optimalizačního nástroje

Zásady pro vypracování:

1. Seznamte se s problémem optimálního balení (Packing problem)
2. Navrhněte a implementujte grafické uživatelské rozhraní optimalizačního nástroje řešícího tento problém. Využijte třírozměrného zobrazení.

Seznam odborné literatury: Dodá vedoucí práce

Vedoucí bakalářské práce: Ing. Libor Waszniowski

Datum zadání bakalářské práce: zimní semestr 2005/06 (změna zadání říjen 2006)

Termín odevzdání bakalářské práce: 19. 1. 2007

L.S.

Prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

Prof. Ing. Zbyněk Škvor, CSc.
děkan

V Praze, dne 2. 11. 2006

Abstrakt

Problematika optimálního balení je studována katedrou řídicí techniky ČVUT-FEL v Praze. Aby bylo možné představit výsledky optimalizačních algoritmů uživatelům, byl v rámci této práce implementován GUI (Grafické uživatelské rozhraní), které výstup takových algoritmů vizualizuje. Návrh a implementace optimalizačního algoritmu není úkolem této práce. Předpokládá se, že optimalizační algoritmus bude spuštěn na výkonném výpočetním serveru, zatímco zde popisované GUI na průmyslových konzolách. Tomu také odpovídá ovládání GUI.

Program je napsán v jazyce C# a pro zobrazování využívá technologii Direct X, konkrétně Direct3D. Transporting Assistant uživateli umožňuje mimo jiné pracovat s kontejnery, načítat zboží a po setřídění optimalizačním algoritmem ho zobrazit v interaktivním zobrazení. Je možné scénu otáčet, přibližovat, přidávat a odebírat zboží nebo pouštět automatické nakládání a to pomocí klávesnice a myši. Tím program napomáhá uživateli s nakládáním a organizací zboží v kontejneru.

Abstract

Bin packing problem is beeing studied on Department of Control Engineering at Faculty of Electrical Engineering in Prague. To give us possibility to present results of optimalization algorithms, this GUI (Graphic User Interface) was implemented. Making this algorithms in not part of this work. We suppose that the algorithms will run on powerful server while the visualization will run on local machines connected to the server.

Program si written in C# language. For the projection of the container and stock, the Direct3D technology is applied. This program allows to its users to work with containers, to load and project stock in interactive way. The scene can be turned and zoomed, the stock can be added or removed. Also the automatic loading of stock can be used.

Poděkování

Rád bych poděkoval Ing. Liborovi Waszniowskému za ochotu a pomoc při volbě a zpracování bakalářské práce.

V Praze dne 19. ledna 2007

…….

podpis

Prohlášení

Prohlašuji, že jsem svou bakalářskou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu § 60 Zákona č.121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne 19. ledna 2007

…….

podpis

Obsah

Úvod

Lidská společnost se velmi rychle rozvíjí a rozrůstá a s tím jsou spojeny neustále se zvyšující potřeby a nároky na dopravu a efektivní organizaci přepravovaného zboží. Pro optimalizaci tohoto procesu se čím dále tím více využívá výpočetní techniky. Ta buď přímo řídí automatizovaná zařízení, která balení vykonávají, jako např. obrovské automatizované jeřáby v loděnicích, které přemístí desítky kontejnerů během několika minut, nebo je výstup vizualizován pro potřeby obsluhy. Generalizovaný problém balení však v sobě skrývá uplatnění i v jiných praktických aplikacích.

Hlavním cílem mé práce je vytvořit grafické rozhraní pro optimalizační algoritmy , které umožní uživatelům efektivně pracovat s přepravovaným zbožím, a to prostřednictvím interaktivní 3D vizualizace. Problém optimálního balení ve 3D prostoru je výpočetně náročný, optimalizační algoritmus (jeho tvorba není cílem této práce) poběží na výkonném serveru, který bude poskytovat již seřazené objekty mnou navrženému programu. Jelikož tento mezičlánek zatím neexistuje, umožňuje program také zadání kontejneru a načtení zboží ze souboru. Obsahuje také algoritmus řazení, který je však pouze dočasný a slouží pouze pro účely testování.

Grafické znázornění je nutná část balicího procesu, je nezbytně nutné aby uživatel snadno a rychle pochopil, jaké řešení mu algoritmus doporučil a mohl ho efektivně zrealizovat.

Tato verze programu není určena pro aplikaci v ostrém provozu, ale jako grafické rozhraní pro testování optimalizačního algoritmu jehož vývoj je na katedře řídicí techniky zamýšlen.

Dalším cílem je teoreticky shrnout hlavní aspekty problematiky optimálního balení a popsat použité technologie.

Práce je rozčleněna do tří částí. V první části se budu zabývat rozdělením a definicí problému optimálního balení jako celku. U každého typu uvedu jeho konkrétní aplikace v praxi. Existuje řada osvědčených algoritmů a postupů specifických pro konkrétní problém, tato práce se

zabývá tímto problémem obecně, nesnaží se žádný konkrétní problém řešit, ale snaží se dát ucelený pohled.

Druhá část mé práce bude věnována popisu technologií, které jsem zvolil a dále důvodům této volby a porovnání s jinými možnostmi, které jsem vyzkoušel.

Ve třetí části této práce se budu zabývat tvorbou a návrhem programu s grafickým uživatelským rozhraním. Program bude v jazyce C# . Pro zobrazení 3D scény je bude použita technologie DirectX, konkrétně Direct3D. Zobrazení bude s možností interakce pomocí myši a klávesnice. Aplikace také umožní práci s kontejnerem i zadaným zbožím.

I. Problém optimálního balení

1.1 Úvod

Tato práce se nezabývá konkrétní aplikací tohoto problému nebo jeho řešením, ale snaží se popsat problém optimálního balení v širší souvislosti a naznačit možná řešení a úskalí která mohou nastat.

Základní myšlenkou optimálního balení je objekty různých velikostí a tvarů "naskládat" do konečného počtu úložišť tak, abychom jich potřebovali co nejméně, tedy abychom maximálně využili daný prostor.

Existuje mnoho různých obměn tohoto problému, jako třeba 2D balení, 3D balení, balení podle váhy, podle ceny apod. Řešení tohoto problému nachází uplatnění v řadě oblastí, jako např. plnění kontejnerů, rozvoz balíků kurýrní službou, optimální postupné nakládání a vykládání zboží na různých místech (setřídění zboží dle pořadí vykládky), nakládání vozidel s omezenou nosností, ale také zálohování dat a jiné.

Snažíme se dosáhnout co nejmenšího počtu nevyužitých míst, které se snažíme minimalizovat vždy, ale většinou se jim nevyhneme. Dále pak přesahů, které jsou v některých aplikacích tolerovány, ale v jiných nikoli (otevřený vs.uzavřený kontejner).

Známou aplikací je problém „naplnění batohu", kde mají jednotlivé prvky, kterými chceme batoh naplnit, určenu mimo svou hmotnost také hodnotu. Cílem je naplnit batoh tak, aby hodnota prvků v něm obsažená byla co nejvyšší.

Při návrhu algoritmu optimálního balení musíme sledovat dva hlavní cíle. Prvním z nich je nalezení algoritmu výpočtu, který nalezne výsledek použitelné kvality, ideálně nejlepší možný Druhým cílem je tento algoritmus nalézt v čase, který je pro danou úlohu přijatelný.

Obecně se může jednat o kombinatorický problém až složitosti NP (nedeterministicky polynomiální), u kterého není známo jak nalézt přesné řešení v rozumném čase, přičemž není známa ani skutečnost, zda vůbec může daný algoritmus existovat. V takovém případě je vhodné použít pro řešení aproximační nebo heuristický algoritmus, který dává nejlepší výsledky. „Since it is NP-hard, the most efficient known algorithms use heuristics to accomplish very good results in most cases. "[5].

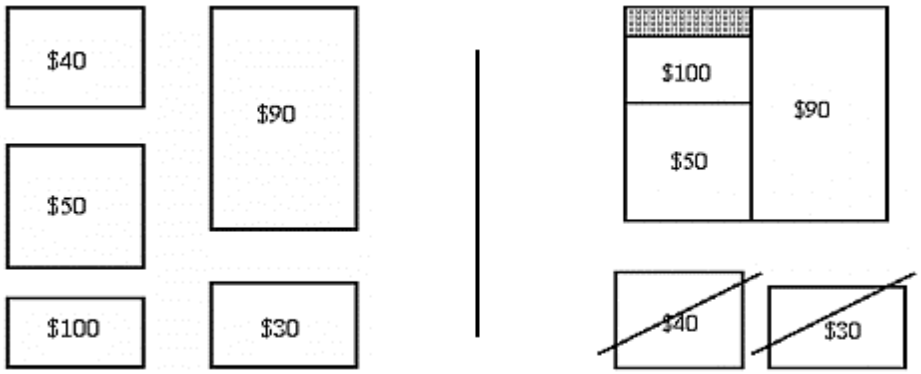
Heuristika je postup, který nestanoví přesné řešení daného problému a ani nezaručuje nalezení řešení v krátkém čase. Pro většinu heuristických algoritmů lze vymyslet takovou sadu vstupních dat, že jejich řešení zabere algoritmu více času, nebo výsledek nedokáže určit vůbec. Navzdory výše konstatovaným skutečnostem jsou takové algoritmy pro většinu praktických úloh dostatečné a výhodné je použít.

V následujícím textu se budu zabývat teoretickým vymezením nejznámějších typů problému optimálního balení.

1.2 Problém „naplnění batohu“

Jednotlivé prvky, kterými chceme batoh naplnit, mají určenu mimo svou hmotnost také hodnotu. Cílem je naplnit batoh, který má omezenou nosnost tak, aby hodnota prvků v něm byla co nejvyšší.

Obrázek č. 1: Grafické znázornění problému naplnění batohu



Zdroj: http://www.cs.sunysb.edu/~algorith/files/knapsack.shtml

1.2.1 Definice [2]

Máme n druhů prvků (položek) x_1 až x_n. každý prvek x_i má přiřazenu hodnotu p_i a váhu w_i. Maximální váha kterou můžeme do „batohu“ umístit nazveme C.

Existují tři základní modifikace tohoto problému.

1. **Do „batohu“ můžeme umístit žádný nebo maximálně jeden prvek od každého druhu.**

Popis:

Snažíme se maximalizovat cenu $\sum_{i=1}^{n} p_i x_i$

A váha nesmí překročit danou mez $\sum_{i=1}^{n} w_i x_i \leq C$

kde $x_i = 0$ nebo $x_i = 1$, $i \in \{1, n\}$

2. **Do „batohu“ můžeme přidat pouze dané množství *a* daného prvku**.

Popis:

Snažíme se maximalizovat cenu $\sum_{i=1}^{n} p_i x_i$

A váha nesmí překročit danou mez $\sum_{i=1}^{n} w_i x_i \leq C$

kde $0 \leq x_i \leq a_i$, $i \in \{1, n\}$

3. **Neklademe žádné limity nato kolik a jekého prvku do „batohu“ umístíme.**

Popis:

Snažíme se maximalizovat cenu $\sum_{i=1}^{n} p_i x_i$

A váha nesmí překročit danou mez $\sum_{i=1}^{n} w_i x_i \leq C$

kde $x_i \geq 0$, $x_i \in Z$, $i \in \{1, n\}$

Pro každou modifikaci problému existují různé algoritmy, které se liší podle konkrétní aplikace.

Více informací je možné nalést na stránce http://www.or.deis.unibo.it/knapsack.html.

1.3 Problém „bin packing“

Bin packing je souhrnný název pro problém, kdy se snažíme určitým způsobem řešit a optimalizovat vkládání malých částí, do co nejmenšího množství větších, definovaných kontejnerů.

Jak jsem již zmínil, obecně se může jednat o kombinatorický problém až složitosti NP. Pokud se budeme snažit tento problém řešit metodou, kdy projdeme všechny prvky a vyzkoušíme všechny možnosti, tak se již při poměrně nízkém počtu prvků dostaneme do problémů, neboť složitost výpočtů narůstá s počtem prvků exponenciálně. Je tedy vhodné používat heuristické či přibližné algoritmy, které v praxi v naprosté většině případů vyhoví.

1.3.1 Matematický popis

Máme kontejner (prostor) o kapacitě $C > 0$ a skupinu objektů $\{o_1, o_2, o_3, \ldots o_n\}$. Hledáme takové uspořádání objektů, při kterém bude zapotřebí co nejnižší počet kontejnerů.

Předpokládáme že každý objekt o_n má velikost s_n, pro kterou platí $0 < s_n < C$, tzn. žádný objekt není větší než samotný kontejner.

1.3.2 Balení v jedno-dimenzionálním prostoru

Tento problém nachází uplatnění v těchto praktických aplikacích:

- Televizní či rádiové reklamy určitých délek musí být přiřazeny do komerčních přestávek ve vysílání. Žádná přestávka nesmí trvat déle než určenou dobu.
- Výrobek, jako např. drát, lano či potrubí se distribuují ve standardizovaných délkách. Pro danou vyrobenou délku materiálu se snažíme navrhnout co nejmenší počet kusů standardních délek a snažíme se rovněž o co nejmenší dále nepoužitelné zbytky.

- Je třeba vyrobit do určité doby výrobek, u kterého známe výrobní časovou náročnost, snažíme se minimalizovat počet strojů (lidí) potřebných ke splnění úkolu.

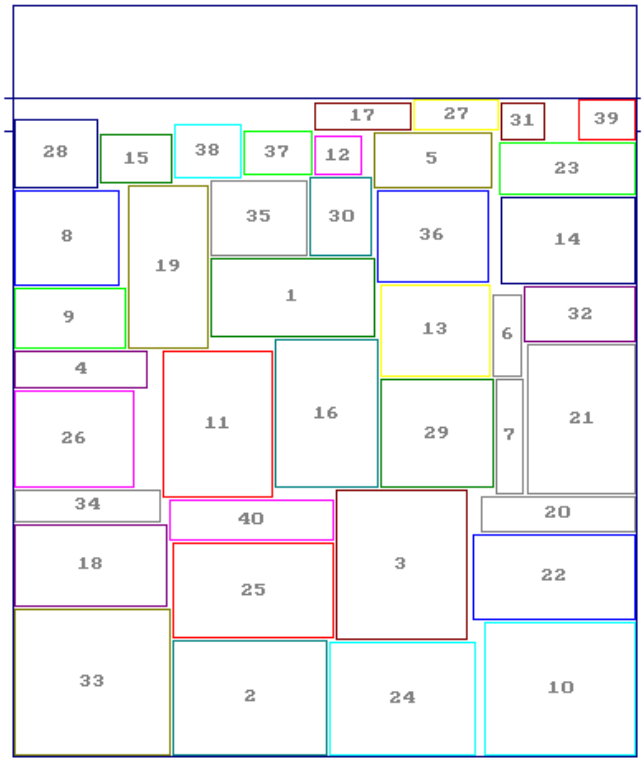
1.3.3 Balení v dvou-dimenzionálním prostoru

Snažíme se sbalit objekty různých velikostí (často stejného tvaru) do pevně daných dvou-dimenzionálních kontejnerů. Tak abychom jich potřebovali co nejméně.

Tento problém nachází uplatnění v těchto praktických úkolech

- Stříhání plechů definovaných velikostí z velkého kusu tak, abychom minimalizovali nevyužitelné zbytky.
- Optimalizace využívání paměťových zdrojů počítače.

Obrázek č. 2: Grafické znázornění problému balení ve 2D prostoru



Zdroj: http://www.astrokettle.com/

1.3.3.1 Balení pásu

Tento speciální případ 2D balení uvažuje kontejner nekonečné délky a určité šířky, do kterého skládáme *n* obdélníků tak, abychom minimalizovali celkovou délku. Obdélníky se nesmí otáčet libovolně, v některých aplikacích je možné je otáčet o devadesát stupňů.

Na tento případ je možno také nahlížet jako na problém s pamětí u multiprocesního počítače. Obdélníky představují jednotlivé úlohy, jejich šířky odpovídají jejich paměťové náročnosti a jejich délka představuje potřebný výpočetní čas. Šířka kontejneru představuje množství paměti, které máme k dispozici. Délka představuje celkový čas potřebný k provedení všech úkolů. Znamená to tedy, že pokud snížíme délku na minimum, snížíme také výpočetní čas. Toto rozhodování je důležité z důvodu optimalizace práce s počítačovou pamětí a jejího efektivního využití. V tomto případe obdélníky otáčet nelze.

Další praktický případ můžeme najít v průmyslu zpracovávajícím plechy, papír, sklo apod. Dlouhé pásy je potřeba efektivně rozdělit na obdélníky určitých šířek a délek. Cílem je vyrobit veškeré požadované tvary, a to s použitím minimálního množství vyrobeného pásu suroviny. Zde je možné uvažovat i otočení jednotlivých obdélníků.

1.3.4 Balení v tří-dimenzionálním prostoru

Tato problematika je v porovnání s předchozími typy balení složitější a klade vyšší nároky na výpočetní a optimalizační algoritmy, neboť pracuje s třírozměrným prostorem, čímž se počet možných kombinací (řešení) mnohonásobně zvyšuje.

Tento problém nachází uplatnění např. v těchto praktických aplikacích:

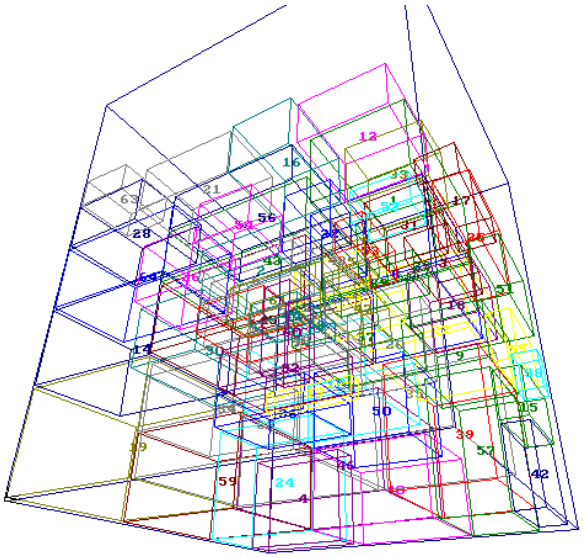
- Plnění kontejnerů zbožím.
- Optimalizace přepravy balíků poštovní službou.
- Nakládáni vozidel které mají omezenou hmotnost.
- Nakládání zboží při nutnosti zvažovat průběžnou vykládku.

Jedná se jednoznačně o nejtěžší úlohu, která nejčastěji využívá heuristické a aproximační algoritmy. Stejně jako u ostatních typů balení se i zde využívá pro výběr a porovnávání těchto algoritmů optimální hodnota. Tato hodnota je často teoretická a udává ideální stav, ke kterému se snažíme co nejvíce přiblížit.

Existuje velká řada postupů, které řeší tento problém. Ty jsou specifické pro konkrétní aplikace a jejich detailní popis není předmětem této práce.

Většina algoritmů se snaží nejprve umístit velké, špatně optimalizovatelné objekty. Postupuje tedy od větších k menším. Tato metoda se nazývá Less Flexibility First (LFF) a je používána také při 2D balení, kde dosahuje velmi dobrých výsledků, tedy největší hustoty prvků.

Obrázek č. 3: Grafické znázornění řešení problému v tří-dimenzionálním prostoru



Zdroj: http://www.astrokettle.com/

II. Technologie .NET

Platforma .NET je knihovna funkcí, stejně rozsáhlá jako vlastní Windows API. Je to vrstva mezi operačním systémem a aplikacemi, která poskytuje efektivní prostředí pro vývoj a spouštění aplikací.

2.1 Důvod volby této technologie

Jde o novou technologii, která je pro společnost Microsoft Co. klíčová. Je zde možné psát programy v různých programovacích jazycích a ty jsou poté díky společnému jazyku (viz. sekce 2.2.2), do kterého jsou při spuštění překládány, schopny navzájem spolupracovat a komunikovat. Lze programovat v jazyku C#, VB.NET, J# a v řízeném C++.

Prostředí, NET i jazyk C# jsou od počátku postaveny na důsledně objektově orientovaných základech. Základní knihovny tříd byly znovu celé navrženy a díky dřívějším zkušenostem a vyvarováním se předchozích chyb umožňují efektivní a velmi intuitivní práci.

Zjednodušená je také práce s externími daty a to díky množině komponent souhrně označovaných jako knihovna ADO.NET (Microsoft ActiveX Data Objects). Ta umožňuje efektivní práci jak s relačními databázemi jako je MySQL, MSSQL, ORACLE, POSTGRESQL a dalšími, tak s mnoha dalšími zdroji dat jako např. souborový systém, Excel soubory a mnoho jiných. Kromě toho je do všech tříd této knihovny zabudována podpora jazyka XML, díky čemuž lze přenášet data i mezi systémy.

Byl zrevidován způsob sdílení kódu mezi aplikacemi a zaveden nový pojem sestavení (assembly, viz. sekce 2.2.2) tak, aby se odstranil problém známý jako DLL Hell, čili přepsání společné knihovny jinou verzí, která nemusí být plně kompatibilní s předchozí. Systém může obsahovat více verzí jednoho sestavení současně. Sestavení může také obsahovat integrovanou bezpečnostní informaci, kterou lze přesně určit, kdo může volat metody jednotlivých tříd.

Přináší vylepšené mechanismy správy paměti - tzv. Garbage Collector - monitorující paměť a zdroje, které jsou v aplikaci použity a dle potřeb je vrací zpět systému.

Prostředí .NET nabízí také integrovanou podporu pro webové aplikace a to díky nové technologii ASP.NET.

2.2 Jazyk C# a architektura .NET Framework

2.2.1 Jazyk C#

Jde o relativně nový programovací jazyk, který byl navržen speciálně pro použití v pracovním prostředí .NET Framework. Jelikož se vývojáři společnosti Microsoft mohli opřít o více než dvacetileté zkušenosti s vývojem jazyků jako byly (C, C++ a jiné) a také díky zdravé konkurenci ze strany společnosti Sun Microsystems, jmenovitě jazyka Java, vznikl robustní jazyk založený na moderním objektově orientovaném návrhu, který je velice intuitivní a výkonný.

Vlastnosti jazyka

C# je case-sensitive, což znamená, že významově odlišuje velká a malá písmena. Je koncipován hlavně pro psaní řízeného kódu, lze jej však v případě potřeby využít i pro tvorbu kódu neřízeného (unsafe). Použití takového kódu znamená, že běhové prostředí CLR neověřuje, zda je kód bezpečný (např. se neověřuje jinak vyžadovaná typová bezpečnost aj.).

Plně podporuje také znakovou sadu unicode.

Sestavení (assemblies)

Jak jsem již zmínil, sestavení zabraňuje problému známému jako DLL Hell. Je to logická jednotka, která obsahuje zkompilovaný kód, metadata popisující metody a typy definované v odpovídajícím kódu a metadata, která popisují samotné sestavení. Ta jsou uložena v manifestu a umožňují zjišťovat verzi a mnohé jiné informace. Mohou být uloženy v jednom nebo ve více souborech či paměti. Je-li sestavení uloženo ve více souborech, existuje vždy jeden soubor, který je hlavní a obsahuje popis všech dalších souborů.

Existují dva typy sestavení. Je to sestavení soukromé a sdílené.

Soukromá sestavení

Data v něm jsou přístupná pouze aplikaci, pro kterou byla vytvořena a jsou většinou dodávána přímo s aplikací. Takové programy nevyžadují složitou instalaci a není potřeba nic zapisovat do registrů. Taková instalace se nazývá „instalace s nulovým dopadem“.

Sdílená sestavení

Taková sestavení je možné adresovat a používat v rámci celého systému (.NET), jedná se tedy o společné knihovny. Z toho plynou rizika, proti kterým musí být sestavení chráněno. Sdílená sestavení jsou proto uložena ve speciální složce s názvem ”mezipaměť sestavení (assembly cash)” a je nutné je speciálně instalovat.

Hlavní problémy, kterým je třeba se vyhnout, je konflikt názvů a možnost přepsání jinou verzí. Konflikt názvů je řešen automatickým přidělováním názvů, založeném na kryptografickém klíči. Takový název označujeme jako “silný”. Přepsání novou verzí nemůže nastat, neboť každé sestavení má v sobě (v manifestu) uloženou informaci o verzi, takže je možné instalovat paralelně více verzí jednoho sestavení.

2.2.2 Architektura .NET frameworku

.NET framework stojí jako nadstavba nad operačním systémem. Základem zajištujícím základní funkcionalitu celého .NET frameworku je CLR (Common Language Runtime) neboli operační prostředí. Kód spouštěný pod kontrolou CLR se nazývá řízený (managed code). Na tomto Common Language Runtimu jsou postaveny všechny další základní knihovny a objekty.

Většina aplikací, vytvořených například v jazyce C nebo Visual Basic je zkompilována přímo pro danou platformu. To znamená, že zdrojový kód je při kompilaci převeden do strojového kódu počítače typického pro danou platformu. To zaručuje dobrou rychlost běhu výsledného programu, ale plynou z toho i některé nevýhody, jako např. nepřenositelnost mezi jednotlivými platformami.

Princip řízených běhových prostředí, použitý právě u platformy .NET (podobně jako i u známé platformy Java), to řeší jiným způsobem a přidává k převodu ještě jednu vrstvu, takzvaný mezikód, do kterého jsou zdrojové kódy zkompilovány. Mezikód je převeden do strojového kódu až běhovým prostředím na cílové platformě (Windows, Linux) virtuálním strojem. Nevýhoda je pomalejší inicializace díky kompilaci (optimalizaci) kódu. Z tohoto důvodu se tento způsob nepoužívá pro vývoj výpočetně náročných aplikací, pro většinu je ale výpočetní výkon a rychlost běhu daných aplikací naprosto vyhovující.

Mezi hlavní výhody při využití mezikódu, MSIL(Microsoft Intermediate Language) v případě .NET, patří:

- Možnost optimalizovat kód pro konkrétní hardwarové vybavení počítače, jelikož virtuální stroj ví, zda konkrétní CPU je Athlon či Pentium, jestli podporuje např. MMX či 3DNow a podobně.

- Interoperabilita programovacích jazyků, tzn. že třídy napsané v určitém jazyce jsou schopny komunikovat přímo (nikoli přes rozhraní COM, které má svá úskalí) s třídami napsanými v jiném jazyce. Je dokonce možné, aby třída napsaná v jednom jazyce byla potomkem třídy v jiném jazyce, objekt může přímo volat metody objektu napsaného v jiném jazyce a je možné ladit (krokovat) volání metod mezi různými jazyky.

Jazyk MSIL

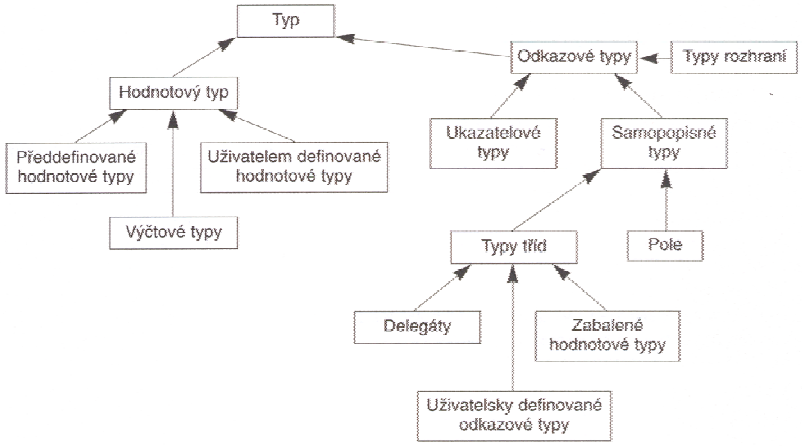
Jak jsem již zmínil, jazyk IL (jak se často označuje), hraje v postředí .NET zásadní roli, je proto dobré mu porozumět. Nejdůležitější jeho vlastnosti jsou:

- **Objektová orientace**
  Je navržen podle pravidel klasického, objektově orientovaného programování, založeného na dědění implementace tříd od jediného předka.
- **Podpora rozhraní**
  Definuje vlastnosti a metody, které musí dodržovat všechny třídy obsahující jejich implementaci.
- **Odlišuje hodnotové a odkazové typy**
  U proměnné hodnotového typu (např. integer, short, struktury atd.) obsahuje proměnná přímo data a je uložena v zásobníku, zatímco proměnné odkazového typu (např. string, objekty, pole atd.) nesou pouze adresu kde na haldě (spravované úložiště dat) jsou data uložena.

- **Přísná typizace dat**

  Všechny objekty jsou důsledně označeny jako objekty určitého datového typu. To v prostředí .NET zajišťuje Společný systém typů (CTS – Common Type System), ten definuje předdefinované datové typy jazyka MSIL.

Obrázek č. 4: Strom datových typů



Zdroj: [3]

- **Společný jazyk CLS**

  CLS čili Common Language Specification spolu s CTS zajišťuje interoperabilitu jazyků a obsahuje množinu standardů podporovaných širokou skupinou překladačů a nástrojů.

- **Automatická správa paměti**

  O správu paměti se stará program **G**arbage Collector. Ten vychází ze skutečnosti, že veškerá dynamicky vyžádaná paměť je přidělována na haldě. Pokud operační prostředí zjistí, že halda některého procesu je plná, zavolá právě Garbage Collector. Ten projde všechny platné proměnné a zkoumá odkazy na objekty uložené na haldě a následně smaže všechny objekty na haldě, na které neexistuje platná reference. To je možné právě díky jazyku MSIL.

III. XML - eXtensible Markup Language

XML, neboli rozšiřitelný značkovací jazyk, je podmnožinou historicky známého jazyku SGML. SGML (Standard Generalized Markup Language), jak byl definován v normě ISO 8879 z roku 1986, je standardní jazyk určený k formálnímu popisu struktury různých dokumentů.

Jazyk XML byl vyvinut a standardizován konsorciem W3C a to hlavně pro výměnu dat mezi aplikacemi a pro ukládání různých druhů dat. Umožňuje navrhnout strukturu dokumentu z hlediska věcného obsahu. Data jsou uchovávána v hierarchické struktuře.

Samotný jazyk XML se nezabývá tím, jak budou data následně zpracována/zobrazena. Pro zobrazování se používají styly (CSS,XSL) nebo je možné dokument převést, pomocí transformace do jiného typu souboru.

Strukturu navrhujeme pomocí tagů, které by měly věcně odpovídat jejich obsahu (viz. příklad). Tyto tagy můžeme definovat v deklaraci dokumentu - DTD (Document Type Definition) a s její pomocí později dokument validovat, zda daným definicím odpovídá. Každý tag může také obsahovat své atributy.

Povinná součást dokumentu je také hlavička (*<?xml version="1.0" encoding="utf-8" ?>*) , kterou definujeme verzi a kódování dokumentu. Doporučené kódování je UTF-8 (8-bit UCS/Unicode Transformation Format), které umožňuje zapsat a uchovat jakýkoli standardní písmenný znak. Po případě je možné použít UTF-16.

Jako příklad uvádím definici v XML, kterou používám pro reprezentaci kontejneru.

```
<?xml version="1.0" encoding="utf-8"?>
<container title="Kvádr">
  <description>Základní a nejčastěji používaný typ kontejneru...</description>
  <parameters>
    <width>13000</width>
    <hight>10000</hight>
    <depth>5000</depth>
  </parameters>
</container>
```

IV. Technologie DirectX

Microsoft DirectX je rozsáhlá a výkonná sada programátorských rozhraní (APIs), které je možno zdarma stáhnout z webu společnosti Microsoft. DirectX poskytuje vývojové prostředí, které nabízí programátorům možnost efektivně využívat možnosti hardware. Tato technologie byla poprvé představena v roce 1995, a od té doby je brána jako standard pro multimediální aplikace a hry na platformě Windows.

DirectX je také možno stáhnout pouze jako instalace pro koncové uživatele, která umožňuje spouštění takto vyvinutých produktů.

Součásti DirectX rozdělené podle svého účelu:

- **DirectX Graphics**
  Jeho části jsou DirectDraw (již není podporován) a Direct3D. Tato část zajišťuje podporu pro grafické aplikace, 3D vykreslování atd. Tuto část využívá i program pro zobrazování a plnění kontejneru.
- **DirectInput**
  Zajišťuje podporu vstupních zařízení jako např. myši, joystiku a různým typům gamepadů. Zajišťuje také funkcionalitu technologie Force Feedback.
- **DirectPlay**
  Podpora hry více hráčů po síti.
- **DirectSound**
  Podpora přehrávání a zaznamenávání zvuků.
- **DirectMusic**
  Podpora přehrávání a zpracování hudby.
  Dříve spolu s DirectMusic označováno souhrnným názvem **DirectX Audio**
- **DirectShow**
  Podpora pro multimediální aplikace, přehrávání a zpracování videa a zvuku.

- **DirectSetup**

  Nástroj umožňující instalaci a update knihovny DirectX na počítač.

- **DirectX Media Objects**

  Podpora pro tvorbu multimediálních efektů, kodeků a pod.

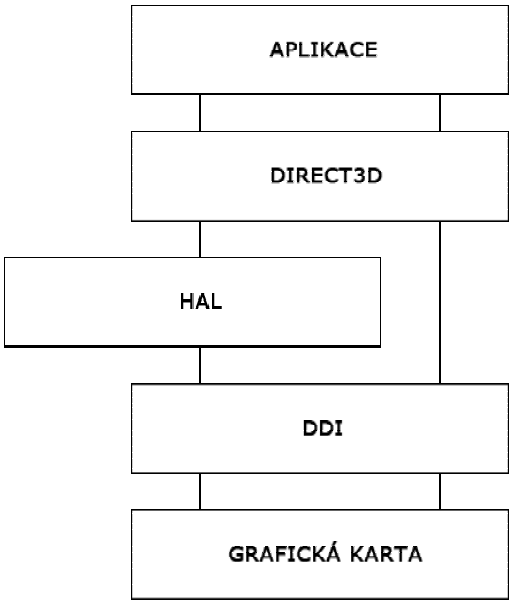
4.1 Direct3D

Nyní se budu blíže zabývat touto částí DirectX, neboť tvoří jádro vlastního zobrazování.

4.1.1 Architektura

Direct3D je programátorské rozhraní (API), které nabízí programátorům možnost efektivně využívat možnosti hardware. Tato vrstva v operačním systému pracuje přímo s HAL (Hardware Abstraction Layer). Výrobce grafické karty dodává také svou vrstvu HAL, která implementuje funkce Direct3D, pouze však ty, které daná grafická karta podporuje.

Obrázek č. 5: Hierarchie vrstev aplikace



Zdroj: [4]

Aplikace využívající Direct3D tedy komunikuje s vrstvou Direct3D, která dále komunikuje s vrstvou HAL, nebo přímo s DDI (Device Driver Interface). Device Driver Interface

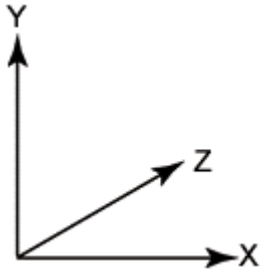
neboli ovladač zařízení je software, který umožňuje operačnímu systému komunikovat s hardwarem

4.1.2 Popis Technologie

Trojrozměrný souřadnicový systém

V prostředí Direct3D se používá levotočivý souřadný systém souřadnic, pro většinu aplikací je to logické uspořádání, jelikož zachovává stejný průběh os X a Y, jako tomu je při zobrazování 2D objektů.

Obrázek č. 6: Levotočivý souřadnicový systém



Zdroj: [5]

4.1.2.1 Lokální a obrazovkové souřadnice

Každý objekt v trojrozměrném prostoru je složen z bodů a z hran. Každý bod je reprezentován třemi souřadnicemi: souřadnicí X, Y a Z. Tyto souřadnice se nazývají lokální. Takové souřadnice však nejsou vhodné při zobrazování na monitoru, jelikož ten pracuje pouze se dvěma souřadnicemi X a Y. Tyto souřadnice se nazývají obrazovkové. Je tedy nutné zobrazit třetí rozměr také do dvou-dimenzionálního prostoru, a to pomocí transformace, neboli projekce. Existují dvě základní:

1. Paralelní projekce

   Tato transformace pouze ignoruje všechny Z-ové souřadnice. Reprezentuje nereálný pohled z "nekonečné" vzdálenosti.

2. Perspekrivní projekce

   Napodobuje přirozený pohled na scénu. Vytváří iluzi hloubky pomocí techniky známé jako perspektivní zkracování.

4.1.2.2 Transformace trojrozměrných objektů

Tyto transformace se využívají při změně velikosti objektu, při jeho posunu nebo otočení, při změně místa pohledu, směru pohledu apod. Při transformaci objektu je nutné transformovat souřadnice každého vrcholu. To provádí Direct3D a využívá k tomu matici, kterou mu předáme. Tuto matici, nejčastěji reprezentující posunutí a otočení, vytváříme pomocí násobení jednotkové matice další maticí, resp. maticemi. Dále v textu uvádím několik základních operací, které je potřeba s maticí provádět, je však zbytečné pracovat s maticemi přímo, je vhodné použít předdefinovanou třídu *Microsoft.DirectX.Matrix*. Operace vždy bod se souřadnicemi (x, y, z) transformuje do bodu nového, se souřadnicemi (x', y', z').

Jsou to tyto operace [5]:

1. **Posun**

$$\begin{bmatrix} x' & y' & z' & 1 \end{bmatrix} = \begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ T_x & T_y & T_z & 1 \end{bmatrix}$$

2. **Změna velikosti**

$$\begin{bmatrix} x' & y' & z' & 1 \end{bmatrix} = \begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Kde složka *x* se zvětší s_x krát, složka *y* s_y krát a složka *z* s_z krát.

3. **Rotace**

Kolem osy X.

$$\begin{bmatrix} x' & y' & z' & 1 \end{bmatrix} = \begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Kolem osy Y.

$$[x'y'z'1] = [x\ y\ z\ 1]\begin{bmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Kolem osy Z.

$$[x'y'z'1] = [x\ y\ z\ 1]\begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

4.1.2.3 Osvětlení

V reálném světě existuje řada typů světel, Direct3D, jejichž nejdůležitější typy také implementuje.

Jedná se o tyto čtyři základní typy:

1. Bodové

   Bodové světlo je vyzařováno na všechny strany od zdroje. Jedná se o obdobu žárovky.

2. Směrové

   Toto světlo produkuje rovnoběžné paprsky z určitého směru, nema určenu přesnou polohu.

3. Kuželové

   Vyzařuje pouze kužel světla, jehož intenzita směrem k okrajům slábne.

4. Okolní

   Světlo které nemá směr ani polohu, vyskytuje se všude ve scéně a působí shodně na všechny objekty.

4.1.3 Výhody použití Direct3D

Direct3D je velice výkonný nástroj pro tvorbu jak her tak grafických aplikací.

Umožňuje stoprocentně využít možnosti grafické karty díky přímé komunikaci s ní přes HAL. Následující tabulka porovnává vlastnosti Direct3D s ostatními grafickými prostředky.

Tabulka č. 1: Porovnávající vlastnosti Direct3D s ostatními grafickými prostředky

	GDI+	**GDI**	**DirectX**	**OpenGL**
Použití	Snadné	Složitější	Náročné	Obtížnátvorba komplikovanějších obrazů
Rychlost	Malá	O málo lepší	100%	cca 70-100%
Přenositelnost na jiné platformy	Ano	Ne	Ne	Ano
Zaměření	Snadné použití	Zastaralé	Rychlost	Rychlost a přenositelnost

Zdroj: [6]

V. Grafické rozhraní optimalizačního nástroje

Cílem tohoto programu je vizualizovat výstup optimalizačního algoritmu, jehož cílem je nalezení nejefektivnějšího způsobu uložení zboží při kontejnerové přepravě. Tak, aby obsluze usnadnil přepravu velkého množství zboží v krátkém čase při minimalizaci nákladů na přepravu.

Program je napsán v jazyce C# (viz. sekce 2.2) na rychle se rozvíjející platformě .NET (viz. sekce 2.), přičemž snaží se využít všechny její výhody. Pro zobrazování třírozměrného prostoru využívá rozhraní Direct3D a perspektivní projekci.

Při navrhování grafického rozhraní optimalizačního nástroje jsem se snažil o co největší uživatelskou přívětivost a přehlednost. Veškeré úkony lze provádět přes menu, nebo použít tlačítka v dané sekci. K dispozici je také nápověda s přehledným návodem, která je kdykoli přístupná jak přes menu, tak přes klávesu F1.

Program se skládá z několika částí a formulářů, jejichž detailní popis a náhled je uveden níže v sekci 5.4.

5.1 Implementace programu

Program je napsán v jazyce C# na platformě .Net verze 2.0. Je složen ze tříd, které uvádím níže. Každá třída obsahuje komentáře u důležitých metod i proměnných.

5.1.1 MainForm

Hlavní třída obsahuje definice konstant a proměnných používaných v celém programu. Definuje menu a jeho akce a spouští veškeré další třídy. Umožňuje pracovat s kontejnery, měnit jejich parametry, načítat je a ukládat. Dále umožňuje načítat zboží ze souboru.

5.1.2 Helpers

Pomocná třída, jenž je určena pro statické metody, které je vhodné volat z různých částí programu. Obsahuje např. funkce pro práci se soubory.

5.1.3 Primitives

Třída ve složce 3D definuje obecný objekt „shape“, od kterého mohou být odvozeny další tvary, konkrétně obdélník. Tvar „shape“ reprezentuje zjednodušený 3D objekt. Načtené zboží je převedeno na tento typ a na skutečné 3D objekty se následně převádí pouze ty objekty „shape“, které obsahují platná data. Tato třída tedy tvoří mezivrstvu mezi načtenými daty a vlastními 3D objekty.

5.1.4 Object3D

Třída ve složce 3D, která obsahuje definice všech využitelných 3D objektů. Všechny jsou odvozeny od společné třídy Object3D.

5.1.5 RenderForm

Tato třída kontroluje, zda jsou správně zadané vstupní informace, tedy zda je zadáno zboží a kontejner. Následně je vykresluje pomocí Direct3D. Obsahuje definice ovládacích možností, jejichž seznam je zobrazen v horní části pro usnadnění ovládání.

5.1.5 SortingAlgorithm

Tato třída je dočasná a slouží pouze k demonstraci při zobrazení, tj. pouze řadí objekty vedle sebe tím, že jim přiřazuje určité souřadnice v 3D prostoru. Pracuje s načteným polem zboží, které je typu „shape“ a v případě úspěchu vrací pole 3D objektů. Tuto funkcionalitu by měl v budoucnu převzít externí optimalizační algoritmus.

5.1.6 AboutBox

Třída, která zobrazuje informace o aplikaci, jako jsou verze, název a popis. Všechny informace načítá z assembly.

5.1.6 FormHelp

Třída, která zobrazuje nápovědu uloženou v resources. Soubor nápovědy se do nich přidává při kompilaci.

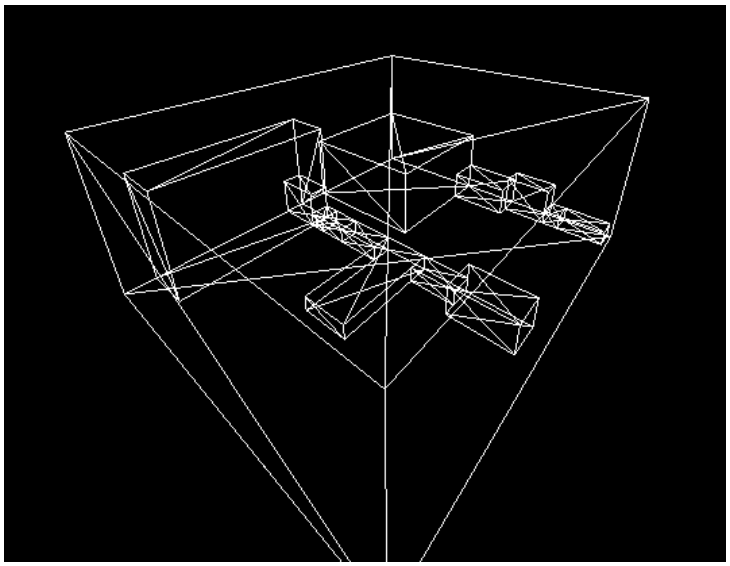
5.2 Implementace 3D zobrazení

První představa a realizace se opírala pouze o knihovnu GDI. Snažil jsem se vykreslovat pouze drátové modely a pro perspektivní zobrazeni používat násobení matic. Tato metoda se však brzy ukázala jako neefektivní a nevhodná. Realizace touto cestou by byla časově velmi náročná a výsledek by se nevyrovnal technologiím, které jsou na zobrazování ve 3D specializované. Další nevýhodou je nízká výkonnost GDI. V úvahu proto přichází technologie OpenGL a Direct3D (viz. sekce 4.1), které jsou výkonově takřka srovnatelné. Zvolil jsem Direct3D, neboť je tato technologie vyvíjena také společností Microsoft, tudíž je zaručena její dobrá spolupráce se zbytkem aplikace. Nevýhodou je nepřenositelnost na jiné platformy.

Velký problém bylo zprovoznění knihovny Direct3D. Nejnovější verze (DirectX SDK - (December 2006)) obsahuje současně dvě verze knihoven, a to verzi 1.1 a verzi 2.0. Při běžné instalaci se nainstalují obě současně a následně způsobují řadu problémů. Jelikož jsou v systému přítomny najednou a mají shodné jméno, např. Microsoft.DirectX.Direct3D, překladač při kompilaci hlásí chybu – neví, kterou má zvolit. Je nutné proto jednu verzi ze systému odstranit. Rozhodl jsem se pokračovat dále s verzí 1.1, jelikož je osvědčená a nové možnosti, které nabízí verze 2.0, bych ve vizualizaci nevyužil. Odstranil jsem proto všechny knihovny ve verzi 2.0.

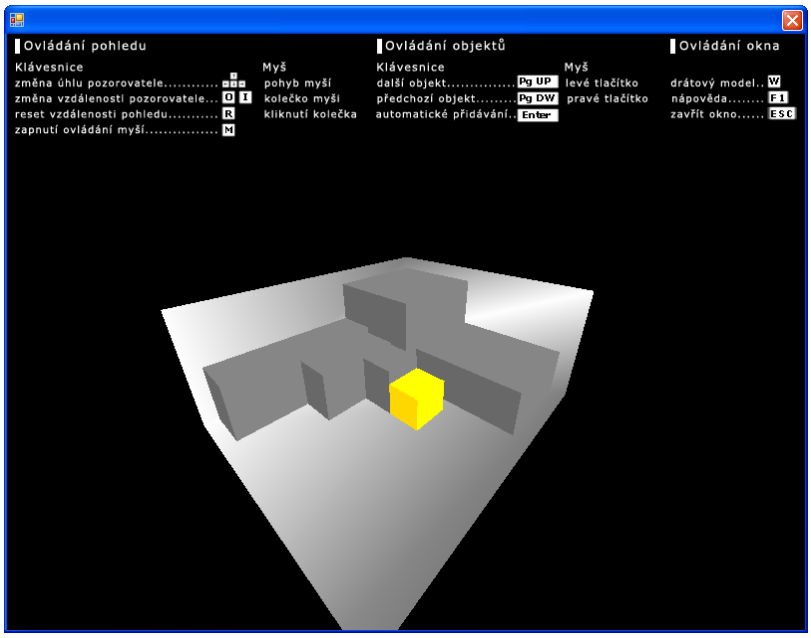
Snažil jsem se minimalizovat výpočetní a paměťové nároky, a proto jsem objekty i kontejner zprvu vykresloval pouze jako drátové modely (viz. obrázek č.7). Výhodou byla snadnější implementace a skutečnost, že byl obdélníkový kontejner průhledný a tedy umožnil nahlížet na zboží. S tímto způsobem vizualizace jsem se rovněž setkal v ukázkových náhledech z komerčních programů, které se zobrazováním také zabývají (viz obrázek č. 4). Pokud je však ve scéně více objektů, obzvláště pokud jsou malých rozměrů, začne být vizualizace značně matoucí a nepřehledná. Drátový model sám o sobě není vhodný a dosti názorný. Proto jsem se rozhodl vykreslovat kvádry plné, tedy zepředu neprůhledné, které pozorovateli umožní udělat si lepší obrázek o tom, jak jsou objekty ve scéně umístěné.

Obrázek č. 7: Zobrazení scény drátovými modely



Pro zobrazeni plných stěn původní objekt nepostačoval a musel jsem ho dále rozšířit a upravit. Plné jednobarevné objekty se však opticky slévaly dohromady a také nedávaly jasný a srozumitelný obraz scény. Bylo proto nutné do scény přidat osvětlení, které by jí dodalo na realističnosti. Díky tomu jsou objekty názorné a je na první pohled patrné, kde je který objekt umístěn a jak je velký (viz. obrázek č. 8). K vykreslování osvětlení používá Direct3D hodnoty normálových vektorů jednotlivých vrcholů. Bylo tedy nutné upravit 3D objekty tak, aby v sobě nesly i tyto informace.

Obrázek č. 8: Zobrazení zboží v režimu plných stěn

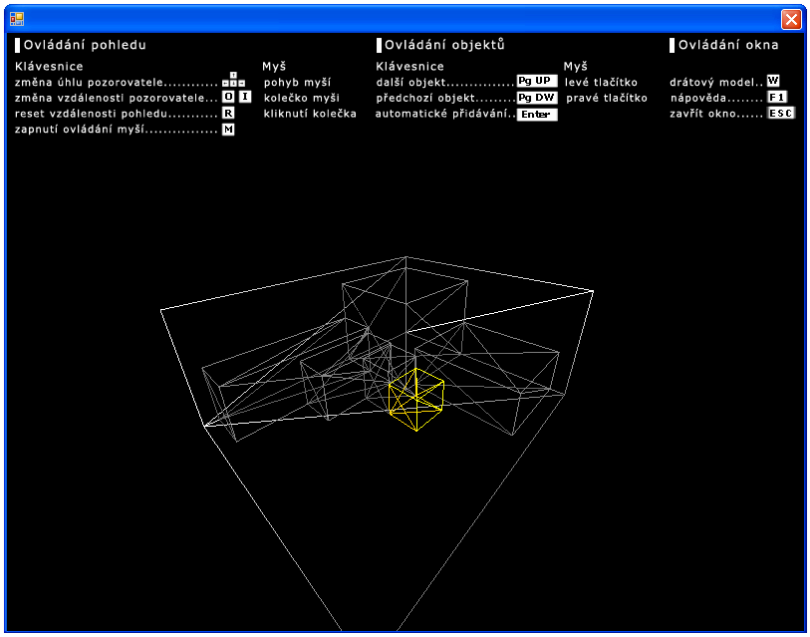


Neprůhlednost zboží nepředstavuje pro uživatele problém, neboť zboží není zobrazeno vše najednou, ale je přidáváno postupně, a to přímo uživatelem nebo automaticky po spuštění automatického přidávání. Stav na obrazovce by měl odpovídat stavu zboží ve skutečném kontejneru. Nevadí proto, že nevidí všechny objekty najednou. Pokud by přesto potřeboval další informace o objektech, je možné se scénou otáčet, nebo zapnout již zmíněné zobrazování drátových modelů (viz. obrázek č. 9).

Model kontejneru je tvořen pouze třemi stěnami, může tedy představovat jak kontejner plněný předem, tak z vrchu. Na zboží lze rovněž nahlížet i z pohledu zezadu či z boku, aby si uživatel mohl v případě nejasností prohlédnout zboží ze všech stran.

Jednotlivým fázím odpovídají i různé objekty v třídě Object3D. Zůstaly tam zachovány pro případ, že by jich bylo v budoucnu zapotřebí. Poslední a tedy doporučená verze třídy objektu zboží se nazývá oBox a kontejneru oContainer v třídě Object3D.

Obrázek č. 9: Zobrazení zboží v režimu drátových modelů



Pro zobrazení kvádrů je možné použít i předdefinovaný objekt Box typu mesh reprezentující kvádr. Ten je však výpočetně náročný a nezapadá do mnou navržené třídy Object3D, která je navržena tak, aby bylo možné jí dále rozšiřovat na další typy objektů.

5.3 Datové zdroje

Pro uchovávání a zadávání dat zboží, které má být naplněno jsem zvolil soubory programu Excel, tedy soubory s příponou xls, a to z několika důvodů. Hlavní důvod je, že tento program má sloužit primárně pro vizualizaci, nikoli pro ukládání a spravování zboží. Pro tyto účely je Excel postačující zdroj dat. Další důvod je uživatelská přívětivost a známost technologie, v neposlední řadě pak průhlednost směrem k uživateli, který má možnost vidět jaké zboží je vloženo, a které také bude zobrazeno.

Údaje o kontejnerech jsou ukládány/načítány ze souborů typu XML (viz. sekce III). Díky struktuře a povaze jazyka XML bude v budoucnu možné kontejnery snadno rozšířit o další parametry nebo vytvářet další typy. Uložení v takovéto formě také dovoluje snadnou manipulaci a efektivní práci. Kontejner ve formě XML lze mimo jiné i snadno distribuovat.

5.4 Uživatelské rozhraní

Uživatelské rozhraní jsem se snažil navrhnout tak, aby bylo maximálně intuitivní a dobře se s ním pracovalo. Aplikace má jednu hlavní obrazovku, ze které lze provádět veškeré úkony.

Uživatelské rozhraní se tedy skládá z několika částí, přičemž zde uvádím jejich výčet a popis.

Jsou to tyto části:

1. **Hlavní menu**
2. **Pracovní plocha aplikace**
3. **Okno zobrazující 3D scénu**
4. **Dialog O aplikaci**
5. **Dialog Nápověda**

5.4.1 Hlavní menu

Z hlavního menu lze obsluhovat všechny důležité akce. Je možné pracovat s kontejnerem, načítat zboží, zobrazovat 3D návrh a další.

5.4.2 Pracovní plocha aplikace

5.3.2.1 Záložka Kontejner

Tato část umožňuje práci s kontejnerem. Formulář obsahuje několik ovládacích a editačních prvků. Je zde možné kontejner vytvářet, editovat, ukládat a vybírat kontejner pro zobrazení. Veškeré funkce jsou k dispozici v hlavním menu nebo přímo jako tlačítka na záložce.

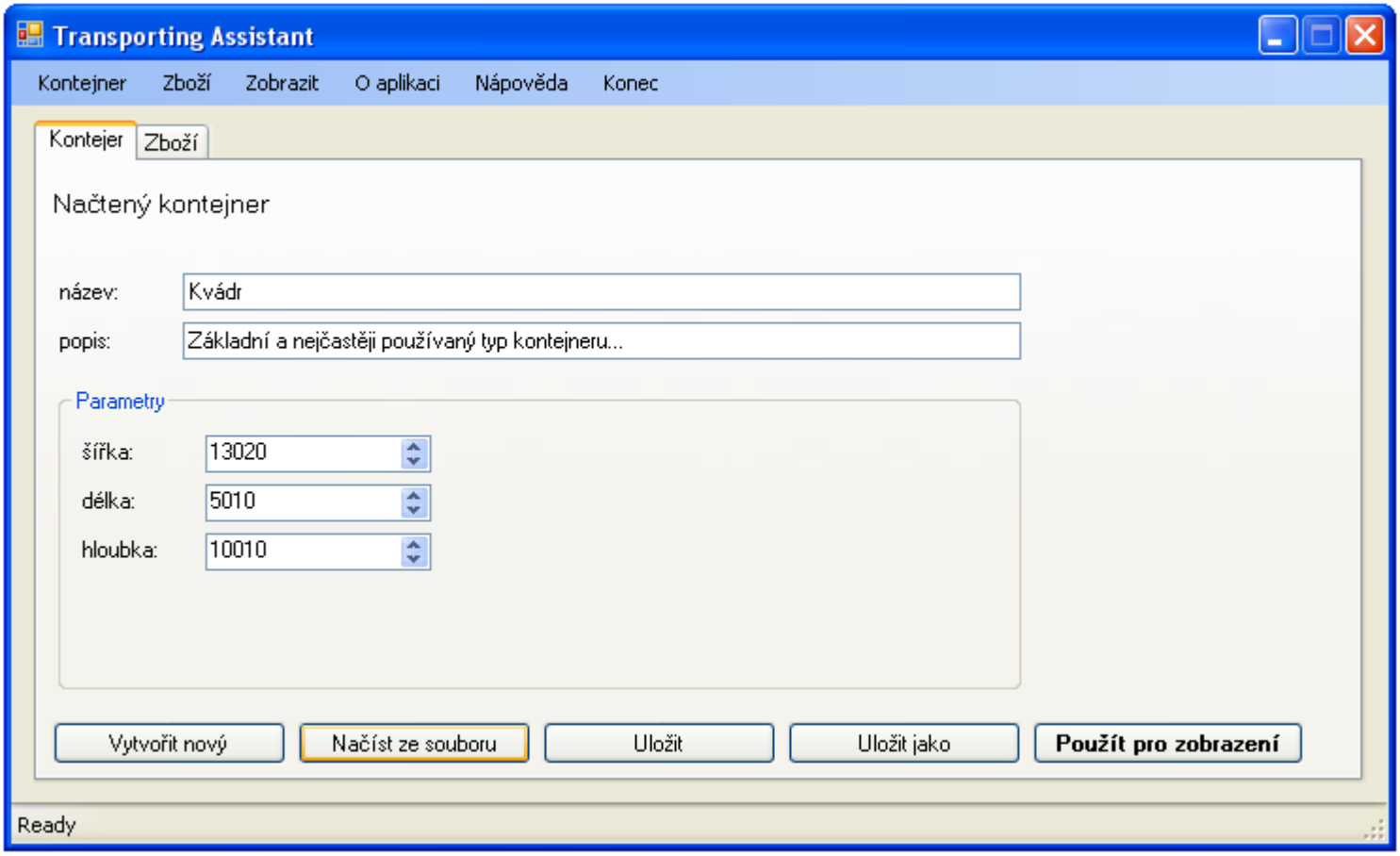
Textové pole „název“ je povinné, „popis“ kontejneru je volitelný, ale doporučuji ho používat. Oblast „Parametry“ je zatím koncipována pro zadáváni kontejnerů kvádrového typu a slouží k nastavení parametrů kontejneru.

Tlačítko „vytvořit nový“ resetuje všechny prvky a umožní vytvořit nový kontejner. Ten je možné následně uložit pomocí tlačítka „Uložit“ nebo „Uložit jako“. V dialogu pro uložení je možné uložit pouze soubor typu XML (viz. sekce III). Dříve uložený či předpřipravený kontejner je možné načíst, následně ho modifikovat, nebo přímo použít pro zobrazení.

Pokud pracujeme s existujícím souborem, změny se po uložení promítají přímo do něj, ale je možné ho také uložit jako jiný soubor.

Před zobrazováním je nutné nějaký kontejner vybrat, a to tlačítkem „Použít pro zobrazení“.

Obrázek č. 10: Záložka Kontejner

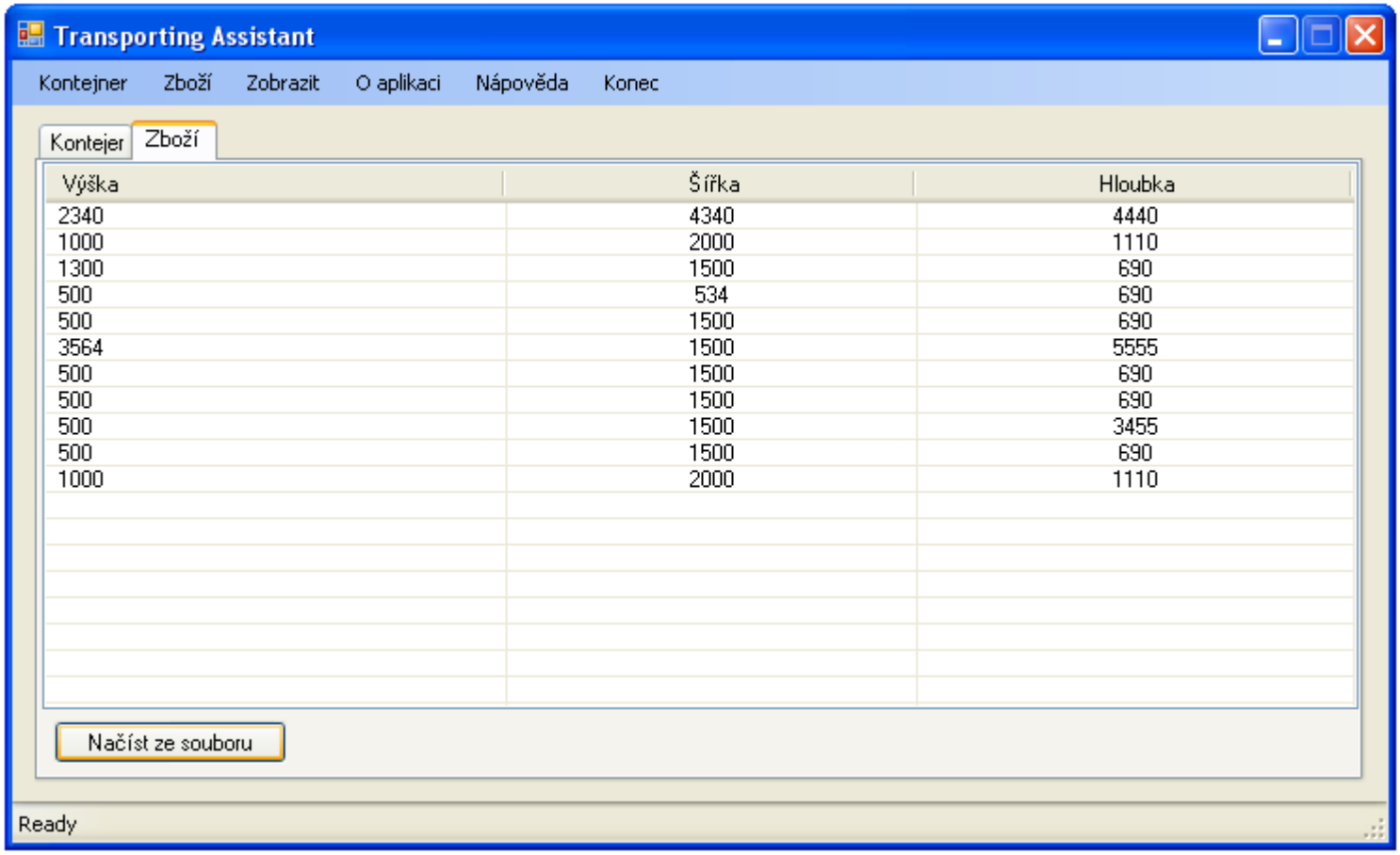


5.3.2.1 Záložka Zboží

Před zobrazením je potřeba zboží načíst. Jako úložiště dat používáme soubor Excel s předdefinovanou oblastí, ze které se zboží načítá. Je to uživatelsky nejsnadnější a nejpřehlednější. Pokud chce uživatel zboží přidat nebo odebrat, stačí změnit obsah v souboru.

Vyvolat lze z hlavního menu položkou „Zboží“ - “Načíst ze souboru” nebo přímo na záložce “Zboží” tlačítkem “Načíst ze souboru”. Následně je uživatel vyzván k výběru příslušného souboru, přičemž lze vložit pouze soubory typu Excel. Pokud vybere správný soubor, načte se zboží do tabulky a je připraveno k dalšímu zpracování.

Obrázek č. 11: Záložka Zboží



5.4.3 Okno zobrazující 3D scénu

Před zobrazením tohoto okna je třeba mít načtené alespoň nějaké zboží a musí být vybrán kontejner pro naplnění. Pokud tomu tak není, aplikace na tuto skutečnost upozorní. Po spuštění se zobrazuje jeden objekt a kontejner.

5.3.3.1 Ovládání

Zobrazení lze ovládat prostřednictvím klávesnice či myši, případně oběma současně. Je možné měnit úhel pohledu i vzdálenost od objektů, přidávat a odebírat objekty, vypnout osvětlení apod. Klávesové ovládání jsem volil neboť program v praxi pravděpodobně nepoběží na běžném počítači, ale na konzoli ve skladu apod.

Po spuštění je na obrazovce pouze jeden objekt, je tedy aktivní (aktivní objekt je zvýrazněn). Pokud chceme zobrazit další objekt, máme více možností, a to buď kliknout levým,

resp. pravým tlačítkem myši pro zobrazení dalšího, resp. předchozího objektu. Objekty lze přidávat a ubírat také klávesami *Page Up* a *Page Down* nebo jednoduchým zmáčknutím tlačítka *Space*. Automatické přidávání objektů v definovaných časových intervalech je možné spustit stiskem klávesy *Enter*.

V případe problémů s porozuměním ohledně přesného umístění objektu lze scénu otáčet. Směr pohledu je možné upravovat pomocí šipek na klávesnici. Šipka nahoru, resp. dolů, upravuje pohled ve vertikální rovině, zatímco šipka doleva, resp. doprava, upravuje pohled horizontálně. Na scénu je možno nahlížet i pomocí myši, a to po aktivování klávesou *M* (**M**ouse look).
V případě potřeby lze stiskem klávesy *W* (**W**ireframe) vypnout vykreslování stěn objektů a zobrazit takzvaný drátový model.

Pokud chceme scénu oddálit resp. přiblížit, použijeme buď kolečko myši, jehož stisknutím se vzdálenost vrátí na původní nastavení, nebo klávesami „+“ a „-“.

Pro zavření okna stačí stisknout klávesu *Esc* (Escape) nebo okno zavřít pomocí myši.

Pokud uživatel zapomene některý ze způsobů ovládání stačí zmáčknout klávesu *F1* a zobrazí se nápověda.

V následující tabulce uvádím přehled všech ovládacích možností.

Tabulka č. 2: Přehled všech ovládacích možností

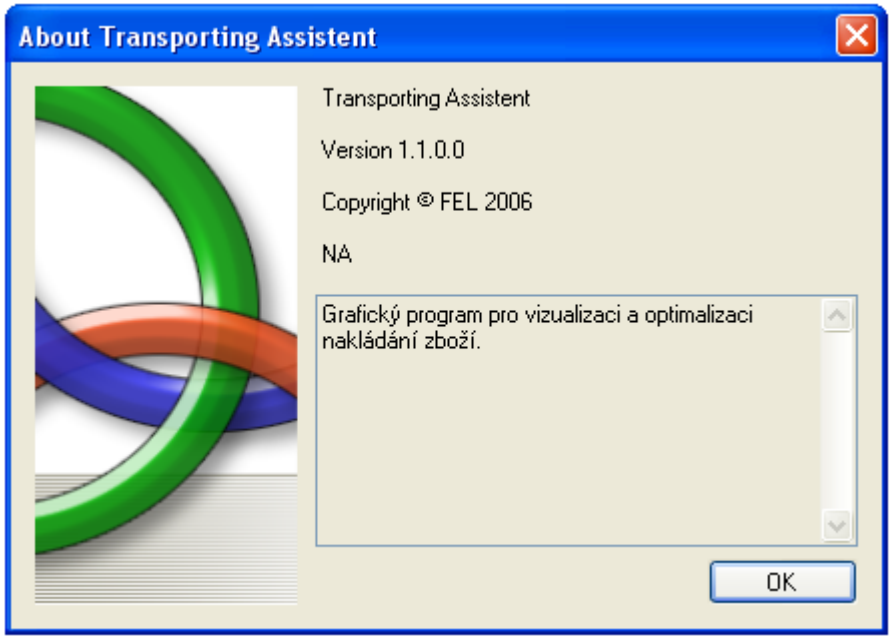
Ovládací prvek	Popis funkce
Ovládání pohledu	
Stisk kolečka myši	Resetuje vzdálenost pozorovatele od scény na původní velikost.
Otáčení kolečka myši	Mění vzdálenost pozorovatele od scény.
Klávesy „I“ (In) a „O“ (Out)	Mění vzdálenost pozorovatele od scény.
Šipky doleva a doprava	Mění úhel pozorovatele v horizontálním směru, otáčí scénu.
Šipky nahoru a dolu	Mění úhel pozorovatele ve vertikálním směru, natáčí scénu.

„M“	Zapíná a vypíná prohlížení scény pomocí myši (**M**ouselook).
Ovládání objektů ve scéně	
„Enter“	Zapíná automatické přidávání objektů v definovaném časovém intervalu. Zobrazí postupně všechny objekty.
„Page Up“	Přidá další objekt, pokud existuje.
„Page Down“	Odebírá aktuální objekt, jestliže není poslední.
Levé tlačítko myši	Přidá další objekt, pokud existuje.
Pravé tlačítko myši	Odebírá aktuální objekt, jestliže není poslední.
„Space bar“	Přidá další objekt, pokud existuje.
Ovládání zobrazování	
„W“	Zapíná a vypíná zobrazení v režimu drátového modelu, tzv. Wireframe.
Ovládání okna	
„Esc“	Zavře okno.
„F1“	Zobrazí nápovědu, mimo jiné obsahující i popis ovládání.

5.4.4 Dialog O aplikaci

Zobrazuje základní informace o aplikaci, jako je verze, název aplikace, popis atd. Tyto informace jsou načítány z assembly (viz. sekce 2.2.1). Tlačítko „OK“ tento dialog zavírá.

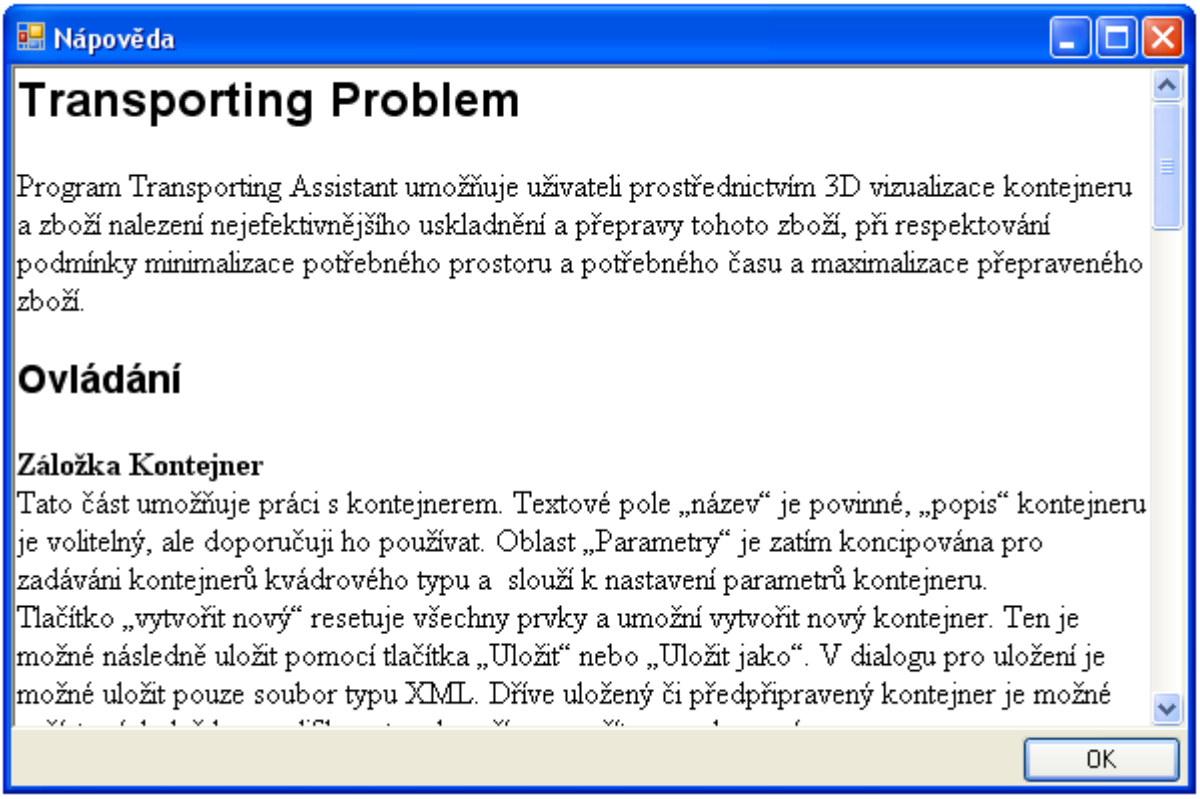
Obrázek č. 12: Dialog O aplikaci



5.4.5 Dialog Nápověda

Textové pole obsahuje základní popis aplikace a popis ovládání. Používá se soubor uložený při kompilaci do resources. Je přístupná také přes klávesu *F1*

Obrázek č. 13: Nápověda programu



Závěr

Problematika optimálního balení se zabývá „naskládáním“ objektů různých velikostí a tvarů do konečného počtu úložišť tak, aby byl maximálně využit daný prostor a aby byl minimalizován počet úložišť. Mezi nejčastější obměny tohoto problému patří 2D balení, 3D balení, balení podle váhy a podle ceny. Řešení nachází uplatnění v řadě oblastí, jako je plnění kontejnerů, rozvoz balíků kurýrní službou, optimální postupné nakládání a vykládání zboží na různých místech (setřídění zboží dle pořadí vykládky), nakládání vozidel s omezenou nosností, ale také zálohování dat a jiné. Seznámil jsem se s tímto problémem a pro jeho konkrétní případ, tedy balení v třírozměrném prostoru jsem navrhl grafické uživatelské prostředí.

Program slouží k vizualizaci výstupu algoritmu, jehož cílem je nalezení nejefektivnějšího způsobu uložení zboží při kontejnerové přepravě a k práci se zbožím a kontejnery. To usnadňuje obsluze přepravy přepravit efektivně velké množství zboží v krátkém čase a minimalizuje náklady na přepravu. K dosažení tohoto cíle je program vybaven řadou prostředků. Jedním z těchto prostředků je možnost zadat kontejner, který potřebuje naplnit a umožnit mu s ním dále pracovat. Dále pak možnost nahrát do programu zboží, které potřebuje přepravit, to předat balícímu algoritmu (Nyní mnou vytvořenému - určenému pouze pro testování. Tvorba plnohodnotného algoritmu není cílem této práce.) a výsledek vizualizovat pomocí interaktivního 3D zobrazení. Toto zobrazení lze ovládat pomocí definovaných klávesových zkratek nebo pomocí myši. Objekty lze postupně přidávat a odebírat, lze také zapnout automatické přidávání dalších objektů v definovaném časovém intervalu. To mu umožní uskladňovat zboží simultánně s programem, podívat se na zboží ze všech stran a efektivně a rychle tak kontejner naplnit. Program je koncipován za účelem dosažení co nejvyšší přehlednosti a snadnosti ovládání. Veškeré úkony lze provádět přes menu, nebo použít tlačítka v dané sekci. K dispozici je také nápověda s přehledným návodem, která je kdykoli přístupná jak přes menu, tak přes klávesu F1.

Nutnou podmínkou k dosažení stanoveného cíle byla volba vhodných technologií. Testoval jsem možnost zobrazovat scénu pomocí vestavěných metod GDI, tato volba se však ukázala jako nevhodná. Program Transporting Assistant je napsán v jazyce C# a pro zobrazení 3D scény využívá technologii společnosti Microsoft Co. Direct3D.

Určitým omezením mého programu je skutečnost, že v současné podobě není koncipován pro zadání kontejneru složitějšího tvaru a zboží jiného tvaru než je kvádr. Tento nedostatek lze v budoucnu odstranit rozšířením původního programu.

Seznam použité literatury

[1] Wikipedia [online], < http://en.wikipedia.org >

[2] Operation Research [online], <http://www.or.deis.unibo.it/knapsack.html>

[3] Simon Robinson a kol., C# Programujeme profesionálně, ISBN:80-251-0085-5

[4] Clayton Walnum, Programujeme grafiku v Microsoft Direct3D, ISBN: 80-251-0136-3

[5] Microsoft MSDN [online], http://msdn2.microsoft.com

[6] Programovací jazyky pro řízení - přednášky [online], http://dce.felk.cvut.cz/pjr/prednasky/

Příloha A. Obsah přiloženého CD-ROM

K této práci je přiložen CD-ROM, obsahuje:

- zdrojové kódy programu s komentáři
- instalaci frameworku Microsoft .NET
- instalaci Direct3D
- text bakalářské práce