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

Diplomová práce

Detekce kolizí

Plzeň, 2008 Jan Vašíček

Abstrakt

This thesis deals with collision detection among proteins of various size. The methods are based on an appropriate space division and reduction of number of atoms that might collide. There are used static, dynamic, and oscillating proteins in the paper. The methods were tested on real data and consequently mutually compared.

Poděkování

Rád bych na tomto místě poděkoval vedoucí mé diplomové práce Doc. Dr. Ing. Ivaně Kolingerové za vedení mé práce, poskytnutí celé řady materiálů a hlavně za čas, který mi věnovala.

Prohlášení

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

V Plzni dne …….. Jan Vašíček

……..……

Obsah

1. Úvod

Detekce kolizí je algoritmický problém, který se vyskytuje v celé řadě oblastí, jako je například virtuální realita, fyzikální simulace založené na vlastnostech v molekulárním modelování, CAD/CAM systémy, výrobní simulace a počítačové hry. Právě oblast fyzikálních simulací založených na vlastnostech v molekulárním modelování nás bude zajímat nejvíce. Tato oblast zahrnuje několik dalších problémů, jako jsou pohyby objektů, vykreslování a reakce na kolizi. Detekce kolizí však zůstává nejvíce výpočetně náročnou částí, u které potřebujeme, aby došlo k jejímu urychlení. Se současným rychlým vývojem počítačových systémů by sice k urychlení mělo dojít, ale s tímto tempem současně roste i množství zpracovávaných informací.

Tato diplomová práce se zabývá detekcemi kolizí mezi atomy jednotlivých proteinů. Při zjišťování kolizí v těchto scénách je kladen důraz jak na rychlost detekce, tak i na přesnost, která je nutná ke zjištění informací o kolizi (hloubka průniku jednotlivých atomů). Tato získaná informace se později může použít k vypočtení celkové reakce jednotlivých proteinů na nastalou kolizi.

Prvním cílem práce je otestování dosavadního řešení pana Ing. Martina Pokorného [9], který navazoval na práci pana Ing. Jaroslava Matouška [7], na reálných datech v podobě proteinů a následné navržení změn v jeho řešení. Druhým cílem práce je realizace navržených změn a jejich následné otestování na reálných datech. Testování v této části práce probíhá i na skutečných problémech v rámci proteinového inženýrství. Výsledkem této práce by měla být aplikace, která slouží pro porovnání upravených metod předešlého řešení na reálných datech v podobě proteinů a na reálných problémech týkajících se oblasti proteinového inženýrství.

Popsaný problém byl řešen ve spolupráci s Masarykovou univerzitou v Brně v kontextu projektu LC06008.

Samostatný text práce je rozdělen do desíti kapitol. Druhá kapitola obsahuje popis problematiky řešení detekce kolizí. Třetí kapitola se věnuje struktuře proteinu a možnostem jeho vizualizace. Čtvrtá kapitola popisuje požadavky na výslednou aplikaci. Pátá kapitola obsahuje popis dosavadního řešení. Šestá kapitola se zabývá testováním dosavadního řešení a na základě výsledků testování popisuje vhodnost dosavadního řešení. Sedmá kapitola se věnuje úpravám dosavadního řešení a v poslední části této kapitoly popisuje výslednou aplikaci. Osmá kapitola je věnována testování výsledné aplikace a jeho zhodnocení. Devátá kapitola shrnuje cíle a výsledky práce. V poslední kapitole jsou uvedeny zdroje použité k vypracování této práce.

2. Detekce kolizí

Detekce kolizí se provádí ve scéně, která se skládá z dynamických a statických objektů, aby se zjistilo, zda dynamické objekty nekolidují mezi sebou nebo s objekty statickými. Jestliže kolize nastala, vypočítá se u příslušného kolidujícího dynamického objektu reakce na kolizi. Test pro zjištění kolize by měl být co nejrychlejší, protože je na něm závislá rychlost vykreslování scény. K detekci kolize tedy dochází před každým vykreslením scény.

Dynamické objekty jsou typem objektů, které se ve scéně pohybují a musí se tedy pravidelně aktualizovat jejich umístění v datové struktuře rozdělující scénu a případně i jejich obalová tělesa. Naopak statické objekty jsou ve scéně pevně zakotveny a zařazují se tedy do struktury pouze jednou po jejím vytvoření.

Samotná detekce kolizí se často rozděluje na dvě hlavní části: širší (broad) a užší (narrow). V první části se identifikují dvojice objektů, které mohou kolidovat. V této fázi testování se používají struktury pro rozdělení scény (pravidelná mřížka, segmentový strom, intervalový strom atd.) a různé druhy obalových těles (Axis Aligned Bounding Box, Sphere Bounding atd). Struktury pro rozdělní scény jsou potřebné k systematickému rozdělení prostoru scény, protože dokáží urychlit další testy tak, že vyberou objekty se stejnou vlastností. Většinou se jedná o polohu objektu v prostoru. Smyslem těchto struktur je vyloučit co nejvíce těles, která nemůžou s testovaným objektem kolidovat. Použití různého typu struktury je závislé na typu scény, požadavcích aplikace nebo na paměťové náročnosti struktury.

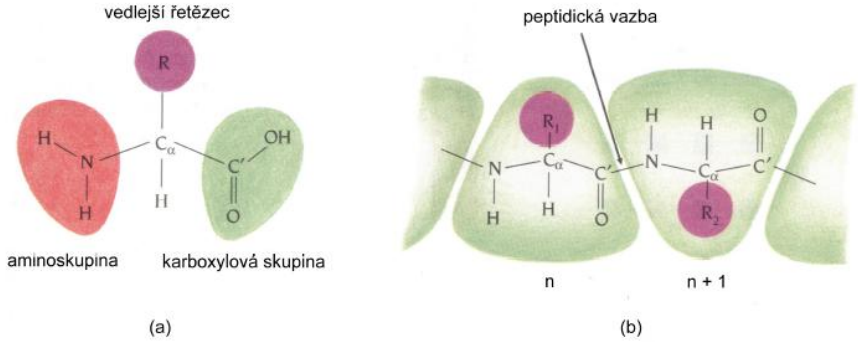
Protože mají objekty často složitý tvar, umisťují se do obálek, které určitým způsobem aproximují jejich povrch. Obalových těles je více druhů a pro použití se volí podle tvaru obalovaného tělesa. Pro těsnější obalení objektu se používají stromy složené z obalových těles. Obalová tělesa jsou opět použita pro vyloučení objektů, které nemohou kolidovat.

V druhé části detekce se zjišťuje, které objekty nalezené v první části kolidují. Testy už se týkají pouze objektů, a proto se provádí test trojúhelník vůči trojúhelníku. Jestliže je zjištěna kolize, dopočítá se na ni reakce na základě kolidujících trojúhelníků.

3. Proteiny

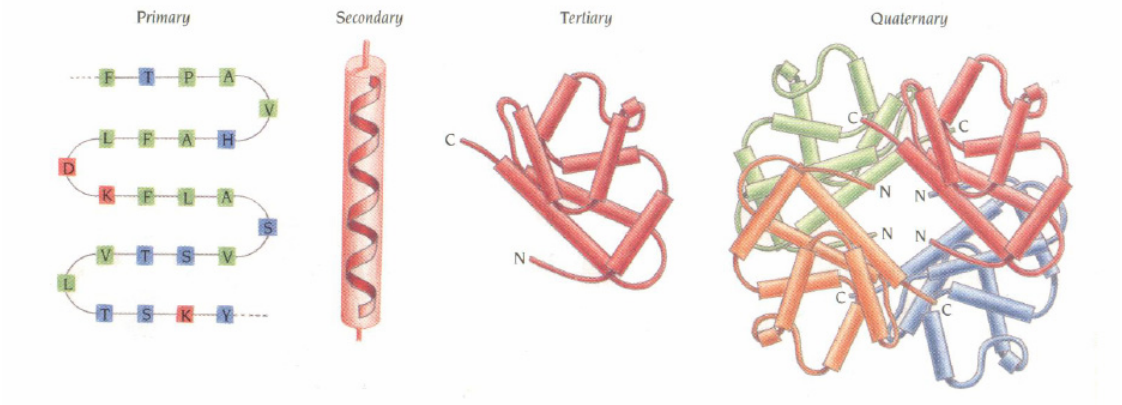
Proteiny, jejichž popis byl nalezen v [15], jsou přírodní látky tvořené spojením několika stovek až tisíců aminokyselin. Jednotlivé aminokyseliny jsou mezi sebou propojeny peptickou vazbou.

Ve všech buňkách živých organismů jsou proteiny obsaženy a plní zde různé funkce, například: stavební, transportní, regulační, řídící, obranné atd. Základní povahu proteinu rozpoznal v roce 1819 Henry Braconnot. Na obrázku 3.1a) je zobrazen strukturní vzorec aminokyseliny a obrázek 3.1b) ukazuje fragment polypeptidového řetězce (proteinu).



Obr. 3. 1: a) Aminokyselina, b) polypeptidový řetězec – převzato z [3]

Většina proteinů je složena do unikátní třídimenzionální struktury. Nejčastěji se používají čtyři odlišné druhy proteinové struktury (viz obr. 3.2.). Primární struktura je dána pořadím aminokyselinových zbytků v polypeptidovém řetězci, které je při syntéze bílkovin v organismu určeno genetickou informací. Sekundární strukturu tvoří uspořádání polypeptidového řetězce do tvaru šroubovice nebo do neuspořádaných úseků. Interakce mezi vzdálenějšími částmi polypeptidového řetězce tvoří terciární strukturu. Kvartérní struktura je vzájemná poloha hlavních řetězců proteinu, které jsou vždy totožné.



Obr. 3. 2 : Hierarchie struktur proteinů – převzato z [3]

Pro počítačové zpracování proteinu je nutné, abychom měli jeho zjednodušený model. V této práci je použit tento:

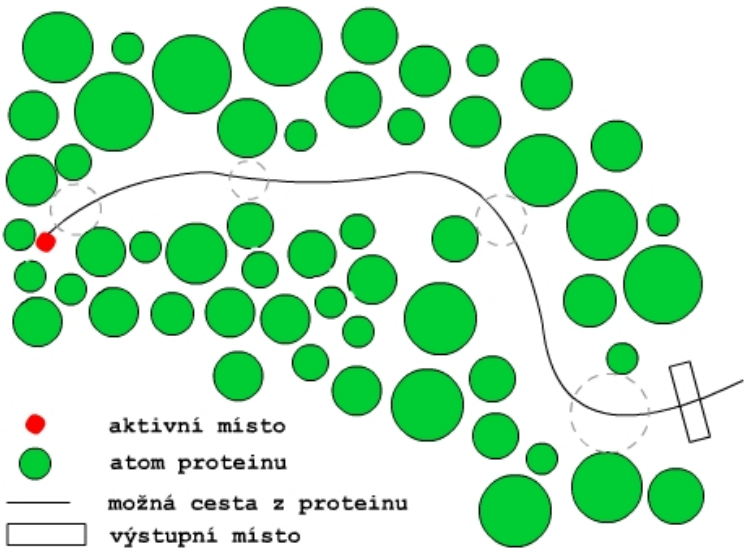
Jednotlivé atomy jsou aproximovány pomocí koulí, jež jsou definovány souřadnicí středu a velikostí. Ta je dána Van der Waalsovým poloměrem, který odpovídá typu atomu a je udán v jednotkách Ångström [A] ($1A = 10x10^{-10}m$).

Znalost chemických vazeb není v této práci použita a v modelu se projeví jako částečné překrytí koulí. Soubory s proteiny a popisy jednotlivých proteinů lze získat z [10].

3.1 Modifikace proteinů a aktivních míst

Drobné změny v primární struktuře se mohou projevit na vlastnostech dané bílkoviny. Toto se využívá v proteinovém inženýrství, jež se snaží vylepšit vlastnosti jednotlivých proteinů cílenými mutacemi.

Při úpravě proteinu je důležité k aktivnímu místu nebo od něj dopravit určitou molekulu, aniž by došlo k narušení struktury proteinu (viz obr. 3.3). Aktivní místo je bod, v němž má dojít k úpravě proteinu. Proto je důležité, zda existuje tunel s určitou velikostí vedoucí z aktivního místa proteinu na jeho povrch. Samotná existence tunelu nepostačuje k tomu, aby byla možná určitá modifikace proteinu, ale znalost geometrické aproximace tunelů v molekule může pomoci odborníkům zaměřit pozornost na její patřičné části.



Obr. 3. 3: Tunel z aktivního místa na jeho povrch

3.2 Vizualizace proteinů

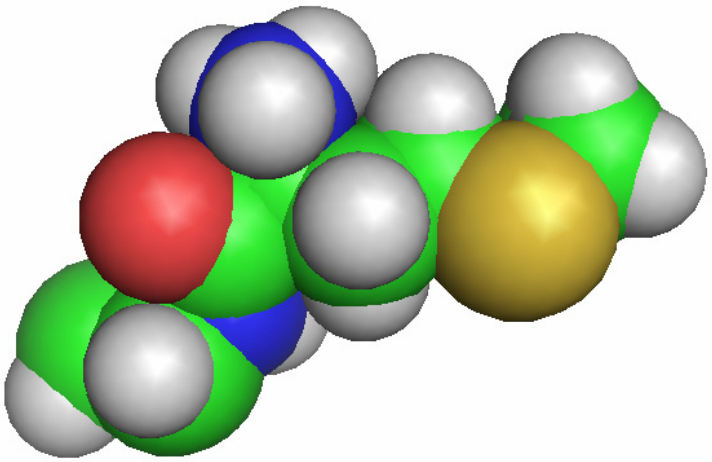
Současné metody pro zobrazení molekulárních modelů používají různá množství informací o proteinu pro jejich zobrazení. V některých modelech se zobrazují všechny

atomy a vazby uvnitř molekuly, v jiných se autoři snaží zpřehlednit zobrazované informace použitím co možná největšího zobecnění a zjednodušení. Mezi aplikace vizualizující proteiny patří například RasMol [12], Chime [5], PyMol [11], VMD [13], MOLMOL [8] a řada dalších. V dalších částí kapitoly budou popsány vizualizační metody proteinů nalezené v [6].

Van der Waalsův model – VDW

V tomto modelu je každý atom proteinů reprezentován pomocí koule s příslušným poloměrem (viz obr. 3.4). Tento poloměr je nazýván *van der Waalsův poloměr* a udává se v ångströmech. Van der Waalsovy poloměry jsou vypočteny z měření vzdáleností mezi páry atomů v krystalech a jsou určeny pro každý prvek zvlášť.

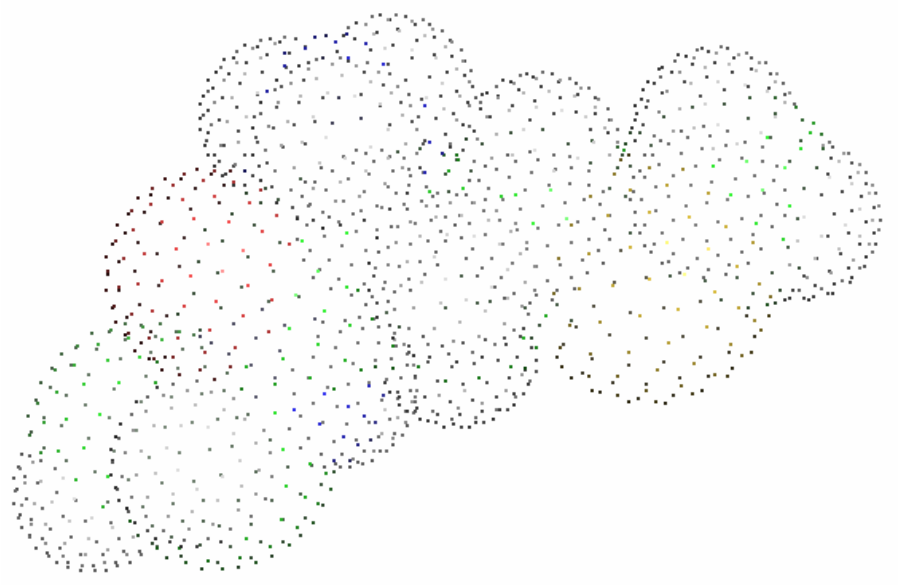
Dva atomy, které jsou spojeny vazbou, se v modelu protínají, protože vzdálenost mezi nimi je menší než součet jejich van der Walsových poloměrů. Díky této vlastnosti se nemusí v modelu vykreslovat vazba mezi jednotlivými atomy.



Obr. 3. 4: Van der Waalsův model – vygenerováno pomocí PyMolu

Tečkový model – Dot model

Jedná se o model rovněž vytvořený z van der Waalsových koulí. Rozdíl mezi tímto a předchozím modelem je ve způsobu vykreslování. V tomto modelu jsou koule zobrazovány pomocí teček (viz obr. 3.5).

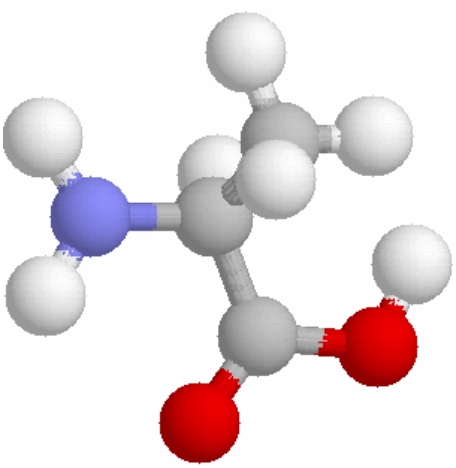


Obr. 3. 5: Tečkový model – vygenerováno pomocí PyMolu

Koule a tyčinky – Balls and Sticks model

Tento model znázorňuje všechny atomy a vazby, jež se v molekule vyskytují (viz obr. 3.6). Jednotlivé atomy jsou zobrazeny pomocí poloměru, který je poměrnou částí z Van der Waalsových poloměrů daného prvku. Koule se v tomto modelu neprotínají, a tudíž je nutná vizualizace vazeb mezi nimi. Vazby jsou tvořeny válcem rozděleným na dvě poloviny o malém poloměru. Každá část je vykreslena barvou příslušející jejímu atomu.

Tato technika vizualizuje molekuly daleko lépe než předchozí dvě metody, protože umožňuje pozorovat i vnitřní strukturu molekuly.

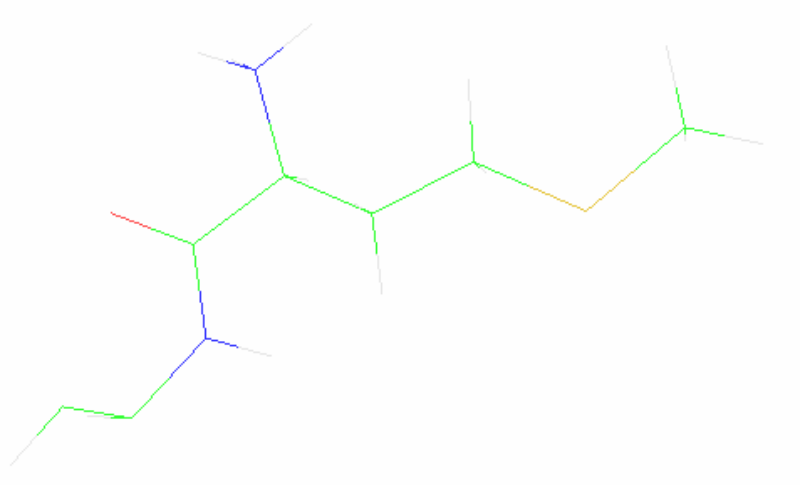


Obr. 3. 6: Koule a tyčinky

Drátový model – Wire

Molekula je u tohoto modelu vykreslována pomocí vazeb mezi atomy. Jednotlivé vazby jsou tvořeny čarami spojujícími středy příslušných atomů. Čára je obarvena podle příslušejících atomů (viz obr. 3.7).

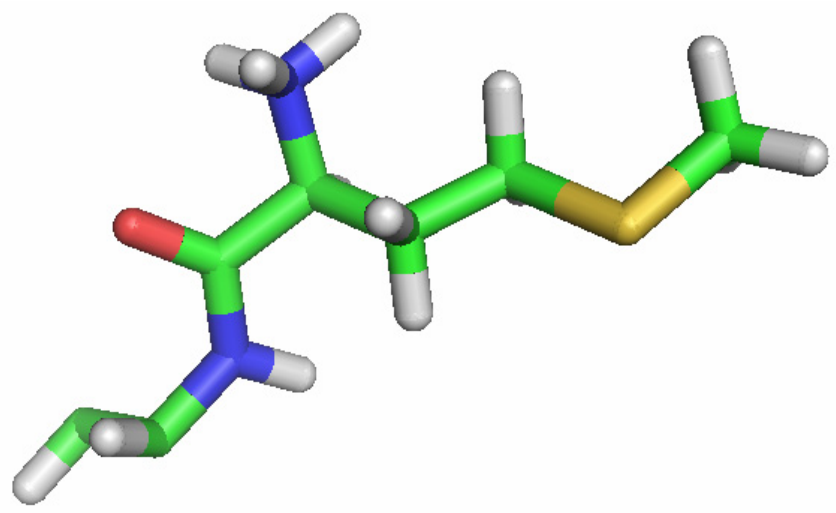
Tento model je vhodný pro případy, kdy je potřebné znát uspořádání struktury celého proteinu. Kvůli použití čar jako vazeb atomu neposkytuje metoda téměř žádný prostorový dojem.



Obr. 3. 7: Drátový model – vygenerováno pomocí PyMolu

Tyčinkový model – Sticks

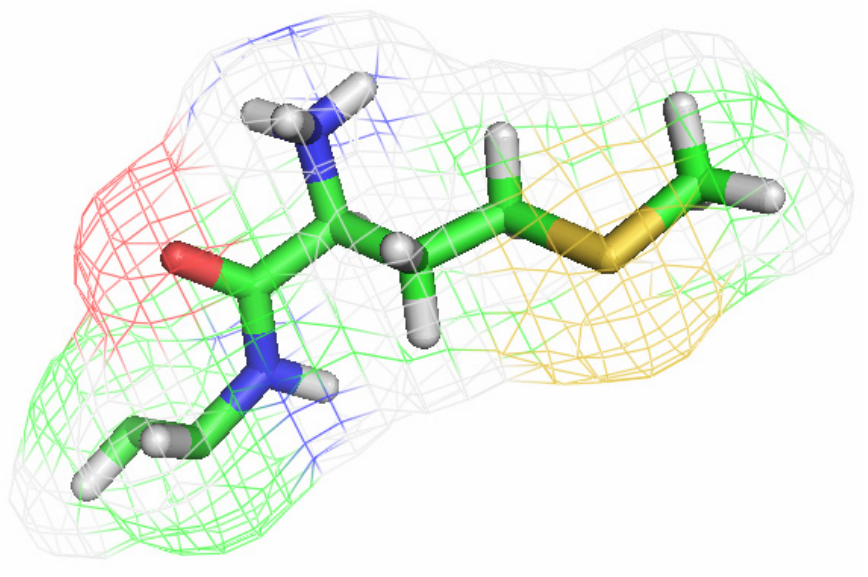
Tyčinkový model opět zobrazuje pouze vazby mezi jednotlivými atomy molekuly, ale vazba je zde tvořena válcem. Způsob obarvení válce je totožný s předcházejícím modelem. Příklad zobrazení modelu je na obrázku 3.8.



Obr. 3. 8: Tyčinkový model – vygenerováno pomocí PyMolu

Síťový model – Mesh

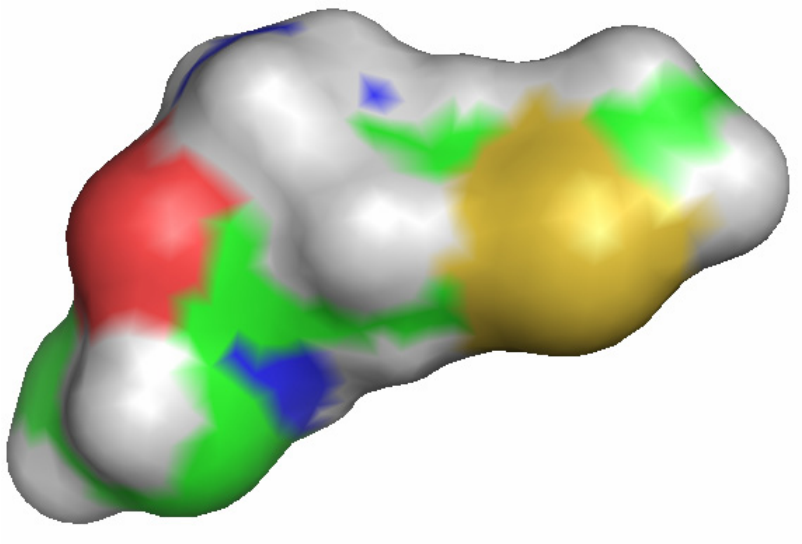
Tato metoda zobrazuje povrch molekuly pomocí sítě (mesh), tudíž je vykreslena jak vnitřní struktura, tak i celý povrch proteinu (viz obr. 3.9). Struktura je znázorněna pomocí některé z předešlých metod.



Obr. 3. 9: Mesh – vygenerováno pomocí PyMol

Povrch – Surface

Podobně jako předchozí model znázorňuje i tato metoda povrch molekuly. Tato technika zobrazuje povrch jako celistvou plochu (viz obr. 3.10). Výsledek je přehlednější, ale výpočetně náročný.



Obr. 3. 10: Surface – vygenerováno pomocí PyMol

Pro stávající a cílovou aplikaci byl vybrán Van der Waalsův model, protože poskytuje dobrou reprezentaci objemu celého proteinu, a je tudíž nejvhodnější pro zobrazování a výpočet detekce kolizí mezi jednotlivými částmi proteinu. V následující kapitole budou popsány požadavky na cílovou aplikaci.

4. Požadavky na výslednou aplikaci

V této kapitole budou popsány požadavky na cílovou aplikaci, některé z nich jsou založeny na požadavcích zadaných Masarykovou univerzitou v Brně v rámci projektu LC06008. Lze je rozdělit do dvou kategorií interaktivní a neinteraktivní. V první kategorii by měla aplikace interaktivně zobrazovat jednotlivé proteiny a zároveň počítat a upozorňovat na kolize, které mezi nimi nastanou. Současně by aplikace měla umožnit v tomto režimu výstup výsledků kolizí do souboru. Druhý režim aplikace by měl umožnit ověřování, zda mezi zadanými proteiny nedošlo k nějaké kolizi. Jako výstup by v tomto režimu měly sloužit soubory, do kterých by se ukládaly všechny možné informace o kolizích. Reakce na kolize mezi proteiny nebude vypočítávat skutečnou reakci na kolizi, protože takové výpočty jsou výpočetně i časově náročné a nejsou obsahem této práce.

4.1 Interaktivní režim

Prohlížení proteinu

Aplikace by měla umožňovat prohlížení jakéhokoli proteinu a zároveň umožnit protlačování atomu nebo atomů zobrazovaným proteinem.

Prohlížení pohybu malého proteinu po cestě velikým proteinem

V tomto režimu by aplikace měla zobrazovat pohyb menšího proteinu ve větším po předem zadané cestě. Navíc by měla umožnit ukládání informací o kolizních atomech do souboru. Program by tedy měl otestovat, zda je zadaný menší protein schopen projít větším proteinem po cestě a informovat o průběhu pohybu po cestě.

Prohlížení vytřepávání

Aplikace by měla obsahovat jednoduchý režim pro automatické hledání cesty z aktivního místa mimo protein. Proto by měl být do aplikace zařazen tento režim, který hledá cestu z proteinu náhodným zvolením směru pohybu proteinu. Aplikace by měla v aktivním místě proteinu náhodně zvolit směr, který se opět změní po kolizi. Hledání cesty by mělo končit buď po nalezení cesty mimo protein, nebo po dosažení určitého množství kolizí. Tento režim však nezaručí nalezení nejvhodnější cesty z aktivního místa mimo protein. Režim by měl umožnit uložení nalezené cesty do souboru.

Prohlížení filmu proteinů

Protože jednotlivé atomy v proteinu ve skutečnosti pořád kmitají, musí se tato vlastnost zahrnout i do této aplikace. Z tohoto důvodu by měl být do aplikace zahrnut

i tento režim, ve kterém jednotlivé snímky filmu proteinu definují různé polohy atomů při jejich kmitání.

Prohlížení pohybu malého proteinu ve filmu proteinu

Tento režim by měl být do aplikace zařazen z důvodu využití důležité vlastnosti atomů proteinu, a to již zmíněného kmitání. Pohybující se menší protein by se měl pohybovat po předem zadané cestě. Jestliže by došlo ke kolizi, měla by následovat změna snímku proteinu a pokračování pohybu po cestě. Pokud by došlo k použití všech snímků za sebou, měla by aplikace upozornit na to, že pohybující se protein nemůže projít kmitajícím proteinem. Režim by tedy měl ověřit, zda může menší protein projít kmitajícím proteinem po předem zadané cestě.

4.2 Neinteraktivní režim

Ověření polohy dvou proteinů

Účelem tohoto režimu by mělo být ověření vzájemné polohy dvou proteinů. Jestliže by některé z atomů proteinu kolidovaly, měly by se tyto informace uložit do výstupního souboru. Pro tento režim by bylo vhodnější použití dávkového zpracování, protože rychlost zpracovávání jednoho ověření by neměla být příliš veliká.

Ověření pohybu malého proteinu v cestě velikým proteinem

Tento režim by měl sloužit pro ověření cesty v proteinu. Po této cestě by se měl pohybovat menší protein. Informace o kolizích během cesty menšího proteinu by se měly opět ukládat do souboru. Protože ani toto ověření by nemělo být příliš časově náročné, bylo by vhodné zavedení dávkového zpracování.

Výpočet cesty z proteinu vytřepáváním

Jednalo by se o režim, ve kterém je vypočítávána cesta z aktivního místa proteinu mimo něj. Cesta by se opět hledala pomocí náhodně zvoleného směru, který se změní, jestliže by došlo ke kolizi mezi proteiny. Zastavení hledání cesty by opět mělo být realizováno, jestliže menší protein nalezne cestu mimo větší protein nebo po dovršení maximálního množství kolizí, které jsou při hledání cesty dovoleny. Zároveň by bylo nejvhodnější zavést počet opakování hledání cesty. Výsledná cesta, kterou tento program nalezne, nemusí být optimální, dokonce aplikace nemusí cestu nalézt.

Ověření pohybu malého proteinu v cestě filmem proteinů

Režim by měl ověřit, zda může menší protein projít kmitajícím proteinem po předem zadané cestě. Jestliže by došlo ke kolizi mezi proteiny, následovala by změna polohy atomů v kmitajícím proteinu. Po použití všech snímků za sebou bez toho, aby se

menší protein pohnul ve směru zadané cesty, by aplikace uložila do souboru informaci, že kmitajícím proteinem neprošel menší protein po zadané cestě a naopak.

Aplikace by měla sloužit pouze pro ověřování určitých možností, které mohou mezi proteiny nastat. Výsledky jednotlivých ověření by měly pomáhat v úlohách proteinového inženýrství.

V následující kapitole bude popsáno dosavadní řešení detekce kolizí, které jsem převzal od svého předchůdce.

5. Dosavadní řešení

Tato kapitola popisuje řešení detekce kolizí, které jsem převzal od svého předchůdce M. Pokorného. Toto řešení naleznete v [9]. Kapitola obsahuje tři části. První popisuje dosavadní řešení jako celek. Druhá část se zabývá použitými rozděleními scény pro zjištění nastalých kolizí a třetí se věnuje obalovým tělesům, která byla použita pro urychlení výpočtu kolizí.

5.1 Popis dosavadního řešení

Dosavadní řešení může použít pro detekci kolizí čtyři datové struktury: pravidelnou mřížku, prostorový strom, intervalový strom nebo sadu segmentových stromů. Toto řešení dále používá obalová tělesa typu: AABB box, obalová koule a OBB tree. AABB box je používán pro zařazení objektu do datových struktur. Pro zjištění kolidujících trojúhelníků jednotlivých objektů se nejprve použije obalová koule za účelem odhalení možných kolidujících objektů a poté OBB tree pro vymezení možných kolizních trojúhelníků.

Pro testování zmíněného řešení použil jeho autor dva druhy scény. První scéna byla s rovnoměrným rozdělením objektů. Druhá scéna měla pouze jeden shluk objektů s několika objekty ve zbývajícím prostoru. Scény byly uměle generovány, a neodpovídaly tedy reálným datům v podobě proteinů potřebných pro současnou aplikaci.

5.2 Použité rozdělení scény

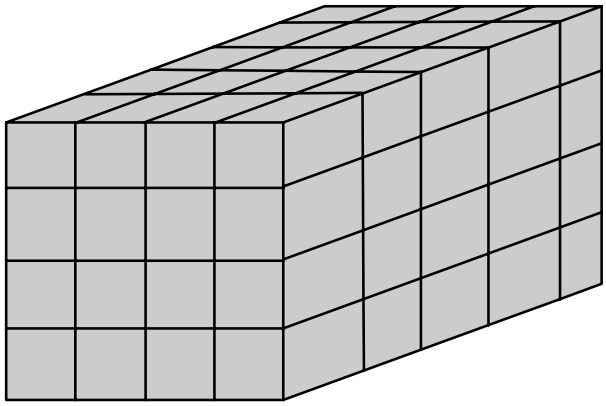
V této podkapitole jsou popsány datové struktury pro rozdělení scény, které byly implementovány v dosavadním řešení. Každý popis struktury obsahuje její charakteristiku spolu s popisem implementace její inicializace, aktualizace scény a detekce kolize.

5.2.1 Pravidelná mřížka

Toto rozdělení prostoru je jednou z nejjednodušších struktur, protože rozděluje prostor do pravidelných krychlí (viz obr. 5.1). Počet krychlí neboli jemnost dělení prostoru se udává při vytváření této struktury. Hlavní nevýhodou je paměťová náročnost, která je daná počtem buněk (krychlí). Struktura totiž obsahuje jak obsazené, tak prázdné buňky. V každé buňce jsou uloženy objekty, které do ní patří, a každý objekt obsahuje buňky, v nichž leží. Objekt se vkládá do struktury rychle pomocí jeho AABB boxu. Odstranění se provede jednoduše zrušením ukazatelů na buňky mřížky v objektu a ukazatelů na objekt v buňce.

Pravidelná mřížka je určena pro rovnoměrné rozmístění objektů ve scéně. Jestliže by byly všechny objekty v jedné části scény, budou objekty umístěny jen v několika

buňkách a struktura by pak neplnila svůj účel, protože pro další testy se vybere většina objektů.



Obr. 5. 1: Pravidelná mřížka

Inicializace struktury

Pro výpočet velikosti jedné buňky a počtu buněk se používá průměrná velikost objektu a maximální vzdálenost objektu od počátku. Velikost mřížky je nakonec zvětšena, aby kolize, které nastanou do určité vzdálenosti od načtené scény, fungovaly. Buňky se vytvářejí ve všech osách a se zvětšující vzdáleností mezi objekty se zvětšuje i jejich počet. Celkový počet buněk je omezen přibližně na 1 000, protože pro větší počet je tvorba mřížky časově a paměťově náročná.

Aktualizace scény

Nejprve se příslušný dynamický objekt vyjme z mřížky, poté se provede pohyb objektu a výpočet parametrů, které se změnily (např. AABB box objektu). Nakonec se objekt začlení do mřížky. Po posledním kroku následuje kontrola na detekci kolizí.

Detekce kolize

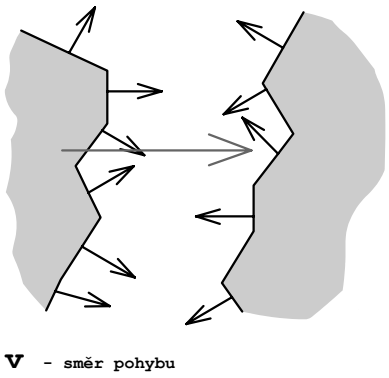
V první fázi kontroly detekce kolize se vybírají objekty z buněk mřížky, v nichž se nachází testovaný objekt. Dále se procházejí všechny nalezené objekty a zjišťuje se, zda už nebyly testovány. Pokud ne, posouvají se do následujících testů. Protože je většinou objekt přiřazen do více buněk, musí se ukládat údaj o stavu otestování.

Druhou fází je test průniku obalových koulí, který dokáže vyloučit vzdálenější objekty a objekty, jež jsou vůči mřížce malé.

Třetí test je proveden na průnik OBB boxů v OBB stromu. Tento druh obalového tělesa obepíná těleso relativně těsně, a proto výstupní objekty tohoto testu, pravděpodobně kolidují s testovaným objektem. Tento test navíc dokáže zjistit, jaký OBB box ze stromu objektu kolidoval, a tudíž do posledního testu zahrnout jen ty trojúhelníky objektu patřící do příslušného OBB boxu. Tím se velice urychlí poslední test.

Poslední fází testování je test trojúhelníků. Do tohoto testu jsou zahrnuty trojúhelníky příslušného kolidujícího OBB boxu a trojúhelníky splňující kritérium vhodné orientace normál příslušných trojúhelníků (viz obr. 5.2).

Výsledkem posledního testu je seznam kolidujících objektů, kterým se přiřadí reakce na kolizi.

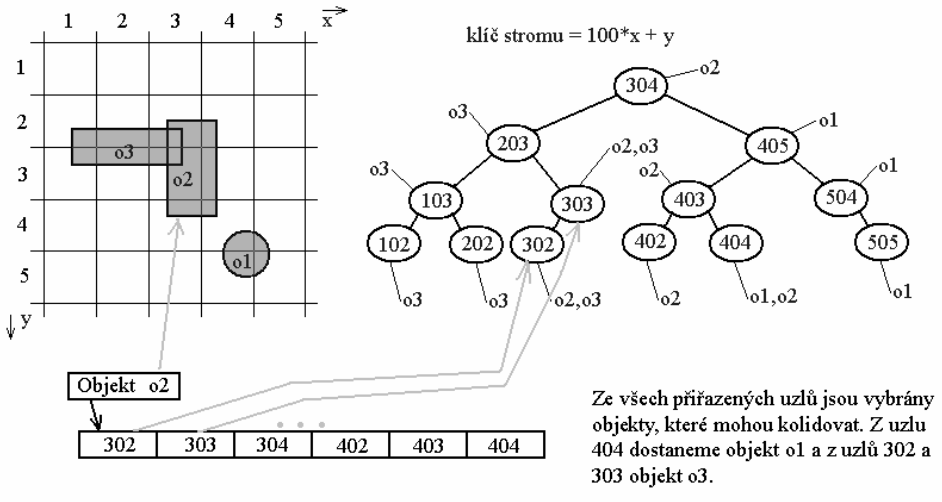


Obr. 5. 2: Výběr trojúhelníků pro test

5.2.2 Prostorový strom

Tato metoda vychází z prostorové mřížky a snaží se odstranit její paměťovou náročnost.

Hodnoty *velikost* a *počet buněk* rozdělující prostor jsou použity stejné jako u pravidelné mřížky, protože byly tyto hodnoty testovány a optimálně zvoleny už v dosavadním řešení. Maximální počet buněk v jedné ose je tedy 10, ale při korekci buněk se přidávají další 2. Maximální počet je tedy 12*12*12 = 1728. Přiřazení buněk ve stromu můžete vidět na obr. 5.3.



Obr. 5. 3: Výběr objektů z prostorového stromu – převzato z [9]

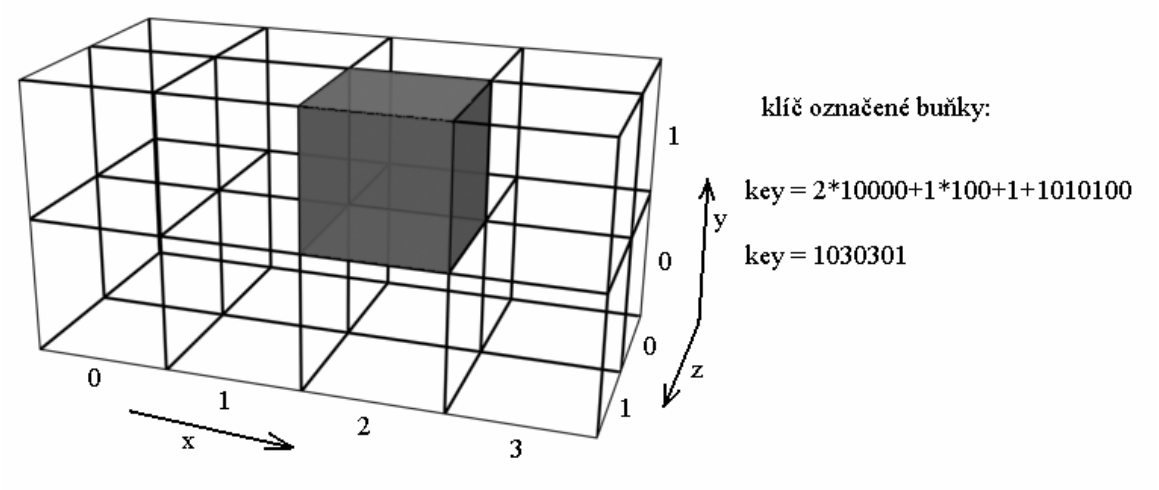
Inicializace

Jako první se musí vytvořit strom a vložit do něj všechny objekty. Pro každý vkládaný objekt se zjistí, do kterých buněk pravidelné mřížky patří, a pokud tyto buňky ve stromě zatím nejsou, vytvoří se příslušné uzly reprezentující tyto buňky. Vkládaný objekt je poté do uzlů vložen. Jestliže je objekt přidán do stromu a ten není vyvážen, následuje AVL vyvážení. Vytváření a vyvažování stromu se provádí pomocí klíče, který je složen ze souřadnic buňky v pravidelné mřížce. Příklad tvorby klíče je vidět na obr. 5.4.

Klíč je tvořen následovně:

$$key = x*100\,000 + y*100 + z + 1010100$$

kde: x – je x-ová souřadnice buňky v mřížce
y – je y-ová souřadnice buňky v mřížce
z – je z-ová souřadnice buňky v mřížce



Obr. 5. 4: Tvorba klíče pro prostorový strom – převzato z [9]

Aktualizace scény

Aktualizace scény u prostorového stromu je velmi podobná aktualizaci scény s pravidelnou mřížkou. Aktualizace se provádí postupně pro všechny dynamické objekty. Každý takovýto objekt je nejprve odebrán ze stromu, následuje změna polohy a poté je znovu zařazen do stromu. Nakonec je nutné zkontrolovat všechny uzly stromu, ze kterých byl objekt odebrán. Jestliže je některý z nich prázdný, musí být ze stromu odebrán. Po každé aktualizaci jednoho objektu je nutné znovu zkontrolovat vyváženost stromu, a v případě nutnosti ho vyvážit.

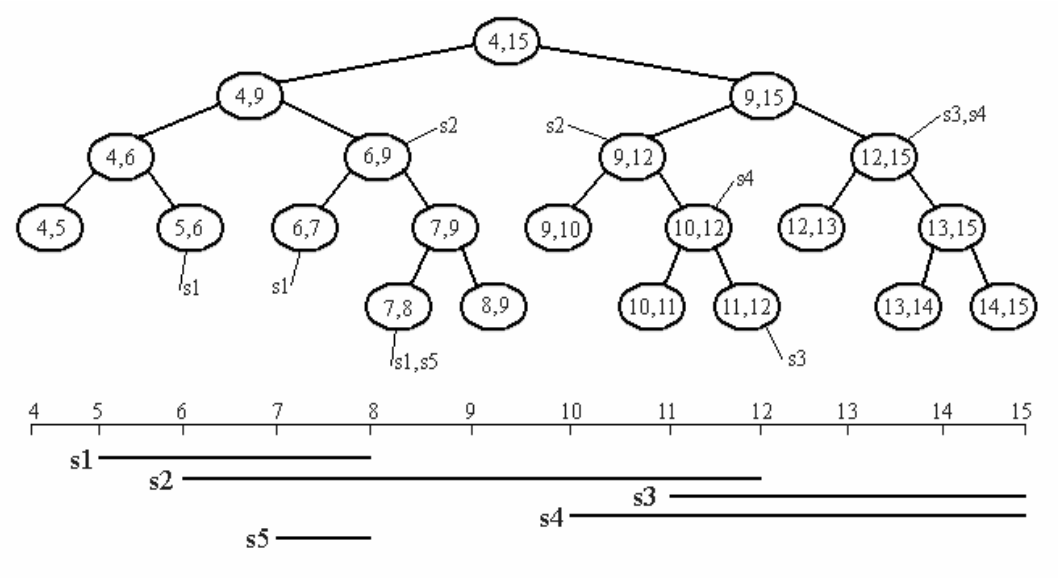
Detekce kolize

Pro testovaný objekt se prochází seznam uzlů, v nichž je objekt uložen. V každém z těchto uzlů se prohledává seznam objektů patřící právě do buňky, kterou uzel reprezentuje (viz obr. 5.3). Nalezené objekty patří do kolizní skupiny, a proto jsou

testovány dalšími metodami pro zjištění kolize. Následující metody jsou popsány u struktury pravidelná mřížka v části detekce kolizí jako fáze dvě až čtyři. Každý otestovaný objekt se označí, aby nedocházelo k opakovanému testování.

5.2.3 Sada segmentových stromů

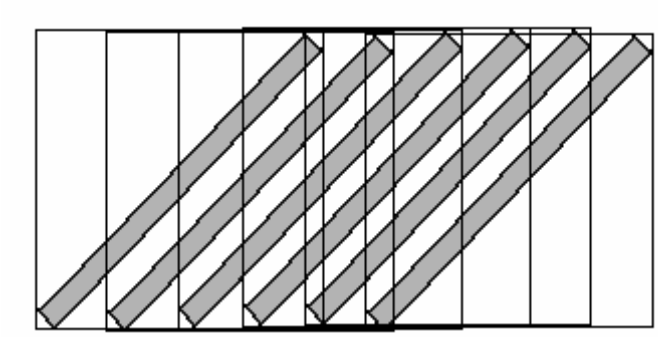
Segmentový strom je datová struktura navržená pro práci s intervaly, které jsou většinou reprezentovány úsečkami. Struktura segmentového stromu vychází z binárního stromu, v němž uzel reprezentuje interval, který je v něm zároveň uložen. Každý uzel může mít pouze dva potomky, jejichž intervaly vznikají půlením rodičovského uzlu. Nákres segmentového stromu s přiřazenými úsečkami je vidět na obrázku 5.5.



Obr. 5. 5: Přiřazení úseček v segmentovém stromě – převzato z [9]

Sada segmentových stromů si ukládá několik segmentových stromů, přičemž pro každou osu prostoru je uchován vždy jeden segmentový strom. Intervaly pro vytvoření jednotlivých segmentových stromů se získají projekcí AABB boxů příslušných objektů na jednotlivé osy. Metoda vybírá objekty, jejichž AABB boxy se protínají. Díky tomu většinou funguje velmi dobře. Jestliže je tato metoda použita na scénu, v níž AABB boxy neobepínají tělesa příliš pevně (viz obr. 5.6), nemusí metoda vhodně fungovat.

Protože pravidelná mřížka ani prostorový strom nejsou struktury vhodné pro scény se shlukem objektů, je v dosavadním řešení implementována právě tato metoda. U stromu se před vytvořením volí jeho maximální hloubka, aby scéna s velkými rozměry nezabírala hodně místa a naopak.



Obr. 5. 6: Nevhodná scéna pro množinu segmentových stromů – převzato z [9]

Inicializace

Nejprve se pro každý segmentový strom vytvoří hlavní kořen, jemuž se nastaví rozmezí celého prostoru, ve kterém se mohou objekty pohybovat a je možné zkoumat jejich kolize. Velikost zkoumaného prostoru se volí na základě maximální vzdálenosti objektu od počátku. Tato hodnota je nakonec zvětšena, aby se objekty mohly pohybovat do určité vzdálenosti i mimo prostor scény. Maximální vzdálenost objektu od počátku se počítá pomocí vzorce:

$$D = max(Li)*1,5 ,$$

kde Li je vzdálenost objektu i od počátku.

Hlavní uzel každého stromu má rozmezí od $-D$ do $+D$. Další uzly stromu jsou přidávány, jestliže je zapotřebí vložit nový objekt. V inicializaci se také zjišťuje minimální jednotka stromu, ve které strom pracuje, tato jednotka se počítá podle následujícího vzorce:

$$u = 2D/1\ 000 * 100$$

Pro každý strom je jednotka stejná, protože všechny tři mají stejný rozsah kořene stromu. Při načítání objektů se každý objekt přiřadí do příslušného uzlu a do objektu se uloží reference na počáteční a koncový uzel v každém stromě.

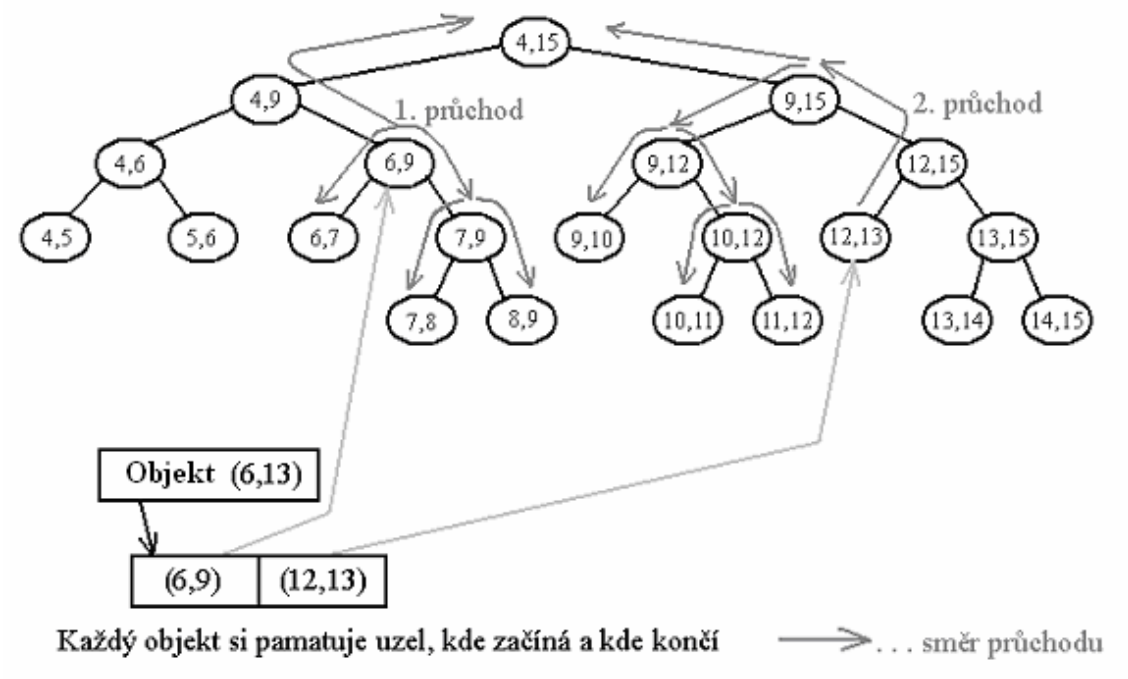
Aktualizace scény

Aktualizace scény probíhá podobně jako u předchozích metod. Objekt je tedy nejprve odebrán ze stromu, poté je změněna jeho poloha a nakonec je opět do stromu vložen. Jediný rozdíl je v tom, že se prázdné uzly stromu neruší.

Detekce kolize

V prvním kroku detekce kolizí se vytváří seznam objektů kolidujících s testovaným objektem. Zkoumaný objekt má v sobě uloženy dva uzly do každého stromu obsahující počáteční a koncový uzel v místě, kde je objekt uložen. Při procházení uzlů stromu získáme seznam objektů, s nimiž může testovaný objekt kolidovat v jedné ose. Procházení stromu při detekci je vidět na obrázku 5.7. Po

průchodu všech segmentových stromů tedy získáme seznam objektů, které mohou s objektem kolidovat, a jsou tedy zařazeny do dalšího testování kolizí. Při dalším zpracování se vynechává test na obalovou kouli, protože objekt je už testovaný na AABB box.



Obr. 5. 7: Procházení stromu při detekci – převzato z [9]

5.2.4 Intervalový strom

Tato datová struktura je modifikací segmentového stromu. U této metody se segmentový strom nachází pouze v jedné ose a zbylé osy jsou reprezentovány seznamem v každém uzlu. Objekty se do stromu a seznamů vkládají pomocí jejich AABB boxů. Přesnost výběru je totožná se segmentovým stromem. Stejně jako tomu bylo u segmentového stromu, musíme i u této metody zavést omezení hloubky stromu a vypočítat jeho jednotku.

Inicializace

Inicializace je rozdělena do dvou kroků. V prvním kroku se vytvoří segmentový strom pro osu *x,* ve druhém se v každém uzlu segmentového stromu založí seznamy pro osy *y* a *z*.

První krok inicializace zahrnuje vytvoření hlavního uzlu a příslušných parametrů segmentového stromu. Velikost scény a jednotka stromu, které se nastavují hlavnímu uzlu, se vypočítají stejně jako u segmentového stromu. Poté jsou postupně zařazovány objekty do uzlů segmentového stromu struktury. Když uzly neexistují, jsou vytvořeny.

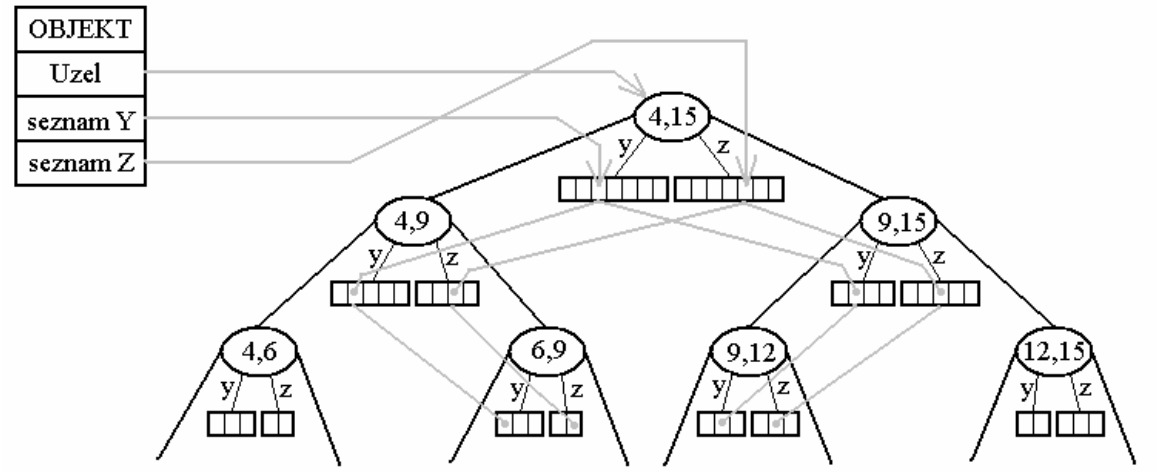
Ve druhém kroku inicializace se vytvářejí seznamy pro osy *y* a *z*. Jestliže jsou vloženy všechny objekty do segmentového stromu, musí se objekty přidat do seznamů

os *y* a *z* v uzlu, kterému objekt náleží (viz obr. 5.8). Do jednotlivých seznamů jsou objekty zařazovány pomocí AABB boxů. Pro zařazení se používá minimální hodnota v dané ose. Každý prvek seznamu obsahuje ukazatele, který představuje stejný objekt v listech aktuálního uzlu, pokud je v těchto listech objekt obsažen (viz obr. 4.8).

Aktualizace scény

Aktualizace scény v ose *x*, která je reprezentována segmentovým stromem, se provádí stejným postupem jako u segmentového stromu. Nejprve je tedy objekt vyjmut ze stromu, poté následuje změna jeho polohy a nakonec jeho zpětné zařazení.

Aktualizace pro seznamy v osách *y* a *z* se provádí změnou aktuální pozice v seznamech, protože autor současného řešení předpokládal, že se objekt nebude pohybovat příliš rychle, a mělo by tedy stačit zkontrolovat jeho pozici v seznamu vůči okolním prvkům a zařadit ho na správné místo. V případech, kdy se objekt přesune do jiných uzlů stromu, se do těchto uzlů zařadí a případně odebere z těch, ve kterých se již nenachází.



Obr. 5. 8: Struktura intervalového stromu – převzato z [9]

Detekce kolize

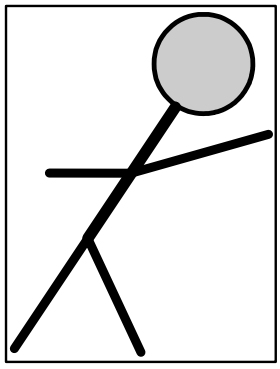
V prvním kroku detekce se prochází segmentovým stromem a hledají se prvky, které mohou s testovaným objektem kolidovat na ose *x*. Poté se prohledávají všechny uzly stromu obsahující testovaný objekt. V nich se procházejí seznamy pro osy *y* a *z*, ze kterých se vyberou objekty kolidující s testovaným objektem. Seznam těchto objektů se poté předá dalším testům.

5.3 Použitá obalová tělesa

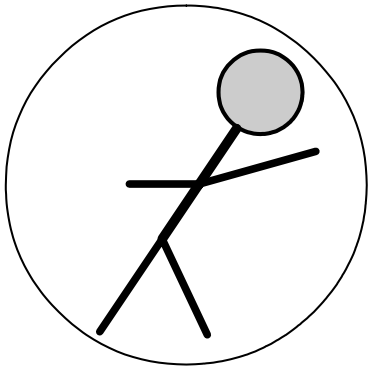
V dosavadním řešení se používají obalová tělesa typu: obalová koule, osově zarovnaný box (AABB) a orientovaný box (OBB) v OBB tree. Tato obalová tělesa budou popsána dále v textu.

5.3.1 Osově zarovnaný box (Axis-aligned bounding box – ABBB)

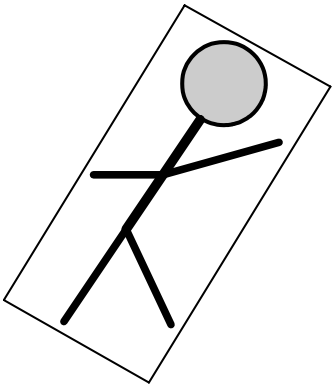
Tento druh obalového tělesa se nejvíce využívá u osově zarovnaných těles, jako je např. budova, protože takovýto druh tělesa obepíná velice těsně. Pokud o obepínaném tělese nemáme žádné informace, není vhodné ho použít, protože není jisté, že jej bude obálka těsně obepínat. Jestliže se navíc obalované těleso pohybuje a zároveň rotuje, musí se AABB pokaždé přepočítávat. Výhodou tohoto druhu obalového tělesa je rychlost zjištění kolize dvou AABB boxů, protože v nejhorším případě stačí pro zjištění kolize šest porovnání (dvě v každé ose).



Obr. 5. 9: AABB



Obr. 5. 10: Sphere bounding



Obr. 5. 11: OBB

5.3.2 Obalová koule (Bounding Sphere)

Toto obalové těleso je jedno z nejjednodušších, protože pro jeho popis je použit pouze střed obalové koule a její poloměr. Ve většině případů obepíná obálka těleso velmi volně, a tudíž se využívá v prvních fázích testování kolizí pro vyloučení objektů, které jsou od sebe velmi vzdáleny. Jeho výhodou je rychlý test na průnik, protože pro jeho stanovení stačí znát vzdálenost objektů a poloměry jednotlivých obalových koulí. Navíc je toto obalové těleso invariantní vůči rotaci obalovaného objektu.

5.3.3 Orientovaný box (Oriented bounding box – OBB)

Orientovaný box má tvar kvádru, který je v prostoru natočený podle obalovaného tělesa. Proto má lepší aproximaci tělesa než předešlá obalová tělesa a je také proto velmi často používán. Pro vytvoření OBB boxu se musí vypočítat osy objektu. Jejich výpočet je popsaný v [9].

Rychlost testu pro zjištění, zda kolidovaly dva OBB boxy, je jednou z výhod tohoto obalového tělesa, protože v nejhorším případě se provádí patnáct jednoduchých testů. Tyto testy se provedou přibližně za cenu 200 aritmetických operací.

Algoritmus testu dvou OBB boxů spočívá v promítnutí těchto boxů na přímku (např. osa *x*). Promítnutím obou boxů získáme dva intervaly, každý příslušející jednomu boxu, a pokud nedojde k jejich překrytí, obalové boxy nekolidují a test končí. Pro úplný test kolize dvou OBB boxů stačí provést test pouze s patnácti přesně určeným přímkami v prostoru.

5.3.4 Strom orientovaných boxů (OBB tree)

OBB tree je binární strom složený z OBB. Jedná se o velice efektivní metodu, která je často používaná při detekci kolize. Tvorba stromu je popsána v [9].

Při kontrole kolize pomocí OBB stromu postupujeme od kořene stromu. Nejprve otestujeme OBB boxy v kořenech stromů. Jestliže kolidují, sestoupíme o úroveň níž v prvním stromě. Následuje porovnávání OBB boxů příslušné úrovně s OBB boxy druhého objektu. Pokud některý z nich koliduje, sestupujeme ve stromě o úroveň níž, v případě, že existují synovské uzly prvního stromu. Pokud ne, prohledáme postupně uzly druhého stromu. Testování končí, když nekolidují OBB boxy nebo když prohledáme celý strom.

6. Testování dosavadního řešení

Autor původní aplikace testoval své řešení na umělých datech. Tato data byla složena z krychlí se stejnou délkou hrany. Rozložení krychlí ve scéně bylo buď rovnoměrné, nebo v podobě shluku tvaru krychle. Všechny objekty v testovaných scénách byly dynamické s klidným náhodným pohybem. Tato testovaná data se však nepodobají reálným datům v podobě proteinů, se kterými by měl náš program pracovat. Proto je nutné zjistit, zda a jak se bude dosavadní řešení chovat při použití reálných dat vhodných pro naši úlohu. Tato data mají podobu shluku koulí s různými poloměry (viz obr. 3.4). Ale před samotným testováním dosavadního řešení na reálných datech se musejí provést určité změny, jinak by při detekci kolizí docházelo k výpočtům, které nejsou zapotřebí. Tyto změny jsou popsány v následující kapitole.

6.1 Úprava v dosavadní aplikaci

Úpravy dosavadního řešení jsou žádoucí, neboť některé výpočty pro zjištění kolizí mezi proteiny nejsou nutné a jejich měření by nebylo vhodné.

V již upravené aplikaci použijeme pro testování kolizí mezi atomy proteinů pouze rozdělení scény s následným testem na průnik obalových koulí jednotlivých atomů, protože atomy proteinu mají tvar koule. Poloměry obalových koulí jsou totožné s poloměry atomů. Tím, že byly vynechány testy na průnik v OBB tree a následné testy stanovení kolidujících trojúhelníků, došlo k urychlení výpočtu detekce kolizí. K nepřesnosti ve výpočtu detekce kolizí však nedošlo, protože trojúhelníkové sítě jsou špatnou aproximací atomů. Nejlépe je atom popsán právě pomocí obalové koule, jejíž poloměr je totožný s poloměrem atomu. Reakce atomů na kolizi je vypočítávána opět pomocí obalových koulí. Jestliže dojde k průniku dvou obalových koulí atomů, je pozice kolidujícího atomu posunuta do nekolizní polohy ve směru jeho pohybu. Kolidující atom tedy obchází ostatní atomy. V již upravené aplikaci dále používáme trojúhelníkové sítě, ale pouze pro zobrazení jednotlivých atomů proteinů ve scéně.

6.2 Testování

Po upravení aplikace mého předchůdce podle výše popsaných změn byly provedeny následující testy. Aplikace je naprogramována v jazyce C++ a byla testována na osobním počítači s operačním systémem Microsoft Windows XP SP2, který obsahoval procesor Pentium 4 (Prescott) na 2,8 GHz, operační paměť o velikosti 1 536 MB a grafickou kartou ATI Radeon X1950 Pro.

Nejprve byla aplikace testována pomocí umělých dat, která se podobala datům reálným, aby se zjistilo, zda jsou datové struktury vhodné pro tento typ objektů. Pro testování byly vygenerovány čtyři druhy scény. První scéna byla složena z koulí stejného poloměru, které měly shlukovité rozdělení. Druhý typ scény měl opět

shlukovité rozdělení koulí, ale jejich poloměry byly různé. Shlukovité rozdělení se nejvíce podobá reálným datům, protože jsou koule umístěny v rovnoměrně rozdělených shlucích. Třetí typ používal rovnoměrné rozdělení koulí v prostoru se stejnými poloměry. Poslední typ scény také používá rovnoměrné rozdělení s koulemi stejného poloměru, avšak zde se pohybovala koule náhodným směrem. Během testování docházelo ke zvyšování počtu koulí ve scéně, aby se zjistilo, s jak velikým množstvím objektů dokáží jednotlivé struktury pracovat. Ve scéně se při testování vždy pohybovala jedna koule simulující pohyb menšího proteinu ve větším. Koule se buď pohybovaly po předem zadané cestě, nebo náhodným směrem, který se změnil po kolizi. Při testování byl měřen čas aktualizace scény. Tento čas byl vždy měřen 1 000x a poté z něj vypočten průměrný čas. Výsledky testování naleznete v tabulkách 6.1 až 6.4.

Průměrný čas						
Počet těles	**2 500**	**5 000**	**7 500**	**10 000**	**12 500**	**15 000**
Pravidelná mřížka	1	2	3	4	4	N/A
Prostorový strom	1	2	3	5	7	8
Sada seg. stromů	1	2	3	4	4	5
Intervalový strom	1	2	3	N/A	N/A	N/A

Tab. 6. 1: Průměrné časy v [ms] – scéna se shlukovitým rozdělením koulí se stejným poloměrem, náhodná cesta

Průměrný čas						
Počet těles	**2 500**	**5 000**	**7 500**	**10 000**	**12 500**	**15 000**
Pravidelná mřížka	1	1	2	4	4	N/A
Prostorový strom	2	2	3	3	4	4
Sada seg. stromů	2	2	2	4	4	5
Intervalový strom	1	2	3	N/A	N/A	N/A

Tab. 6. 2: Průměrné časy v [ms] – scéna se shlukovitým rozdělením koulí s různými poloměry, náhodná cesta

Průměrný čas						
Počet těles	**2 500**	**5 000**	**7 500**	**10 000**	**12 500**	**15 000**
Pravidelná mřížka	1	2	2	3	N/A	N/A
Prostorový strom	1	2	3	4	5	5
Sada seg. stromů	1	2	3	4	4	4
Intervalový strom	2	2	3	N/A	N/A	N/A

Tab. 6. 3: Průměrné časy v [ms] – scéna s rovnoměrným rozdělením koulí se stejnými poloměry, zadaná cesta

Průměrný čas						
Počet těles	**2 500**	**5 000**	**7 500**	**10 000**	**12 500**	**15 000**
Pravidelná mřížka	1	2	2	3	N/A	N/A
Prostorový strom	2	2	2	3	4	5
Sada seg. stromů	1	2	3	3	4	5
Intervalový strom	2	7	3	N/A	N/A	N/A

Tab. 6. 4: Průměrné časy v [ms] – scéna s rovnoměrným rozdělením koulí se stejnými poloměry, náhodná cesta

Z naměřených hodnot v tabulkách 6.1 až 6.4 vyplývá, že všechny struktury dokáží s koulemi představujícími proteiny dobře pracovat. Přitom rozdělení scény pomocí segmentového stromu se jeví jako nejvhodnější metoda, protože jeho časy aktualizace jsou nejvhodnější a narozdíl od pravidelné mřížky nebo intervalového stromu dokáže pracovat i s daty obsahujícími až 15 tis. objektů. Ale rozdíly mezi jednotlivými naměřenými časy jsou tak malé, že nelze přesně stanovit rychlosti řešení kolizí při použití jednotlivých struktur oproti ostatním. Hodnoty *N/A* v těchto tabulkách říkají, že struktura nebyla schopna pojmout takové množství objektů z důvodu nedostatku paměti.

Druhá část testování probíhala na reálných datech získaných z [10]. Scénou se pohybovala jedna koule po předem dané dráze. Měření času bylo totožné jako v předchozí části testování. Naměřené časy jsou uvedeny v tabulkách 6.5 až 6.7.

Průměrný čas				
Počet těles	**3185**	**6298**	**12555**	**25415**
Pravidelná mřížka	2	3	5	N/A
Prostorový strom	2	3	6	N/A
Sada seg. stromů	2	4	6	N/A
Intervalový strom	3	5	N/A	N/A

Tab. 6. 5: Průměrné naměřné časy v [ms]

Maximální čas				
Počet těles	**3185**	**6298**	**12555**	**25415**
Pravidelná mřížka	11	7	11	N/A
Prostorový strom	12	8	29	N/A
Sada seg. stromů	5	6	14	N/A
Intervalový strom	20	11	N/A	N/A

Tab. 6. 6: Maximální naměřené časy v [ms]

Minimální čas				
Počet těles	**3185**	**6298**	**12555**	**25415**
Pravidelná mřížka	1	2	4	N/A
Prostorový strom	1	2	4	N/A
Sada seg. stromů	1	3	3	N/A
Intervalový strom	1	4	N/A	N/A

Tab. 6. 7:Minimální naměřené časy v [ms]

Z tabulek 6.5 až 6.7 uvádějící naměřené časy v druhé části testování vyplývá, že datové struktury dokáží pracovat i s reálnými daty, ale na druhou stranu nejsou schopné pracovat se scénou obsahující nad 25 tis. objektů.

6.3 Vhodnost dosavadního řešení

Dosavadní řešení sice obsahuje datové struktury, které dokáží pracovat s daty popisujícím struktury proteinů, ale množství objektů je omezené z důvodu velké spotřeby paměti při použití jednotlivých struktur. Proto je nutné upravit implementaci struktur v dosavadním řešení, aby bylo možné pracovat s proteiny, které mají více než 25 tis. částí.

Při testování se ukázalo, že vizualizace proteinu v dosavadním řešení není vhodná. Vykreslování proteinu skládajícího se z více jak 5 tis. atomů zobrazovaných jako koule bylo velice pomalé, protože se pohybovalo pod 10 snímky za sekundu. Tato hodnota by měla být větší než 25, aby pohyb dynamických objektů nebyl příliš trhaný.

Popis struktury proteinu je uložen v souboru s formátem *pdb* (*protein data bank*), ale dosavadní řešení nedokáže tento formát načíst. Dosavadní řešení používá souborový formát *ase* (ASCII formát 3D Studia Max) pro definici scény, což není pro naši aplikaci vhodné, protože by se musel před každým použitím formát *pdb* překonvertovat do formátu *ase*, jako to bylo prováděno při testování.

Z předchozích odstavců vyplývá, že je nutné provést úpravy v dosavadním řešení, aby ho bylo možné použít pro práci s proteiny. Úpravy provedené na základě zjištěných nedostatků budou popsány v následující kapitole.

7. Úpravy dosavadního řešení

Z předcházející kapitoly vyplývá, že pro výslednou aplikaci pracující s molekulami je nutné udělat úpravy v dosavadním řešení. Cílem úprav je aplikace, která dokáže rychle zpracovávat úlohy popsané v kapitole 4.

Výsledky testování pozměněného dosavadního řešení ukázaly, že je nutné udělat změny v implementaci jednotlivých struktur pro rozdělení prostoru, objektů představujících jednotlivé atomy molekul, vizualizaci molekul a upravit načítání jednotlivých proteinů ze souboru. Změny, které byly provedeny na předcházejícím řešení před testováním, budou dále ponechány, protože se při testech ověřila jejich vhodnost. V následujících částí budou popsány jednotlivé úpravy.

7.1 Změny v implementaci objektů a struktur

Změny v implementaci objektů a struktur jsou nezbytné, aby se minimalizovalo využití paměti v počítači a aby bylo možné používat proteiny s více jak 25 tis. atomy.

Jednotlivé objekty ve scéně v implementaci dosavadního řešení obsahují pro každou datovou strukturu pole odkazů jejich uzlů. Obsahují tedy čtyři pole odkazů, pro každou datovou strukturu jedno (pravidelná mřížka, prostorový strom, sada segmentových stromů a intervalový strom). Každé z těchto polí má přesně danou velikost, která odpovídá maximálnímu množství uzlů, do nichž může být objekt zařazen. Právě tato pole odkazů představují jedny z největších nároků na paměť. Abychom se těchto nároků zbavili, musel by se do každého objektu scény implementovat lineární spojový seznam pro uzly jednotlivých datových struktur, tedy čtyři seznamy. Bylo tedy nutné doplnit společného předka pro uzly jednotlivých struktur a společného předka pro seznam uzlů uložených v jednotlivých objektech. Seznam uzlů každé struktury používaný v objektu scény je od společného předka zděděn, tudíž je v objektu scény vždy jeden odpovídající používané datové struktuře. Tímto krokem se podstatně snížila paměť používaná aplikací.

Aby bylo možné cílovou aplikaci rozšiřovat o další datové struktury, byl pro všechny struktury implementován společný předek. Jestliže je tedy použitá struktura děděna od společného předka, musí se pouze implementovat nutné funkce a struktura může být začleněna do aplikace. Společně s implementovanou strukturou je nutné implementovat její uzel, který musí být zděděn od společného předka všech uzlů struktur, a seznam jejich uzlů opět zděděný od společného předka. Výsledkem těchto změn bylo vypuštění objektu *Detektor*, který sloužil pro zjištění kolize mezi objekty. Jeho funkci přebraly jednotlivé datové struktury poděděné od společného předka.

Dynamické objekty scény používají v dosavadním řešení pro definici pohybu *kontrolery*, každý dynamický objekt scény obsahuje jeden kontroler. Každý kontroler má dvě části. První generuje pohyb a druhý vypočítává reakci na kolizi. Jestliže by se

scénou pohyboval protein složený z několika atomů, musel by mít každý atom vlastní kontroler, tudíž by jednotlivé atomy mezi sebou kolidovaly a následná reakce na kolizi by zničila strukturu proteinu. Proto je každému atomu z dynamického proteinu přiřazen stejný kontroler a zároveň je vypnuta kolize mezi atomy dynamického proteinu. Současně s implementacemi těchto změn byl do aplikace doplněn společný předek pro kontroler, generátor pohybu dynamického objektu a reaktoru na kolize pro zlepšení implementace kontrolerů.

7.2 Změny ve vizualizaci proteinů

V dosavadním řešení je vizualizační část aplikace velice úzce spojena s částí pro výpočet detekce kolizí, protože se vykreslované trojúhelníky používaly i pro zjištění kolizí. V cílové aplikaci již ale nejsou trojúhelníky pro výpočet kolizí potřeba, a proto má výsledná aplikace dvě části. První se stará o zobrazování jednotlivých atomů a druhá o detekci kolizí mezi nimi. První část je na rozdíl od druhé, jejíž změny byly popsány výše, nově vytvořena, protože vizualizační část dosavadního řešení není vhodná pro zobrazování proteinů. Vytvoření oddělené části pro vizualizaci také dovoluje, aby se při budoucím rozvoji aplikace vylepšovaly jen potřebné části.

Nová část pro vizualizaci nepoužívá pro načítání objektů do scény soubory *ase*, protože by musela v každém objektu ukládat i jeho trojúhelníkovou síť, což je velice paměťově náročné. Dosavadní řešení používalo tyto soubory, neboť trojúhelníkové sítě byly nutné pro zjištění kolizí. Naše výsledná aplikace již používá pro načítání struktury proteinu soubory s formátem *pdb*, které obsahují polohy středů jednotlivých atomů. Proto jsou trojúhelníkové sítě v jednotlivých objektech reprezentující atomy ukládány jako odkazy na trojúhelníkovou síť. Aby se maximalizovala rychlost vizualizace, byla do aplikace navíc implementována metoda *LOD* (*Level of Detail*), která má tři úrovně. Pro všechny atomy proteinu jsou tudíž uchovávány pouze tři trojúhelníkové sítě.

Dále bylo ve vykreslovací části aplikace upraveno načítání textur. Jednotlivé atomy jsou rozděleny do skupin podle druhu materiálu. Materiál se skládá z údaje o použitém shader programu, poloměru atomu a třech texturách. Seskupování atomů do skupin má výhodu při vykreslovacím procesu, protože při vykreslování skupiny dochází pouze jednou k nastavení použitého shader programu a textur. Nahrávání shader programu a textur do grafické karty patří k jedněm z nejpomalejších operací, a proto je nutné je provádět co nejméně krát.

Za účelem docílení maximálního využití grafických karet při vizualizaci bylo použito *API* (*Application Programming Interface*) DirectX 9, které na rozdíl od svých předchůdců poskytuje vynikající možnosti použití programovatelných *GPU* (*Graphics Processing Unit*). Zároveň s tímto rozhraním byly do aplikace naprogramovány SM1 až SM3 (Shader Model), aby byly využity jak starší grafické karty s programovatelným GPU, tak i ty nejnovější. Shader model definuje možnosti grafické karty. Například

Shader Model 3 dovoluje použít pro rychlé vykreslování atomů *geometry instancing*, který v prvním kroku pošle grafické kartě jednu trojúhelníkovou síť vykreslovaných objektů a v druhém doplňující informace. Tyto informace jsou v dané aplikaci tvořeny počtem vykreslovaných atomů, a jejich polohami ve scéně. Tato metoda tedy dokáže na jedno zavolání funkce pro vykreslování zobrazit všechny atomy příslušného LOD. Ostatní Shader modely tuto metodu nemají implementovanou a například v modelu 1 se funkce pro vykreslení musí zavolat tolikrát, kolik objektů je nutné zobrazit. Nevýhodou implementace Shader modelů je, že aplikace může být spuštěna pouze na počítači s grafickou kartou podporující jeden ze shader modelů.

V aplikaci nebylo použito nové grafické rozhraní DirectX 10, protože jeho implementace zahrnuje použití operačního systému Microsoft Windows Vista a grafické karty podporující toto rozhraní, která nebyla k dispozici. V současné době není toto rozhraní tak rozšířeno jako rozhraní DirectX 9.

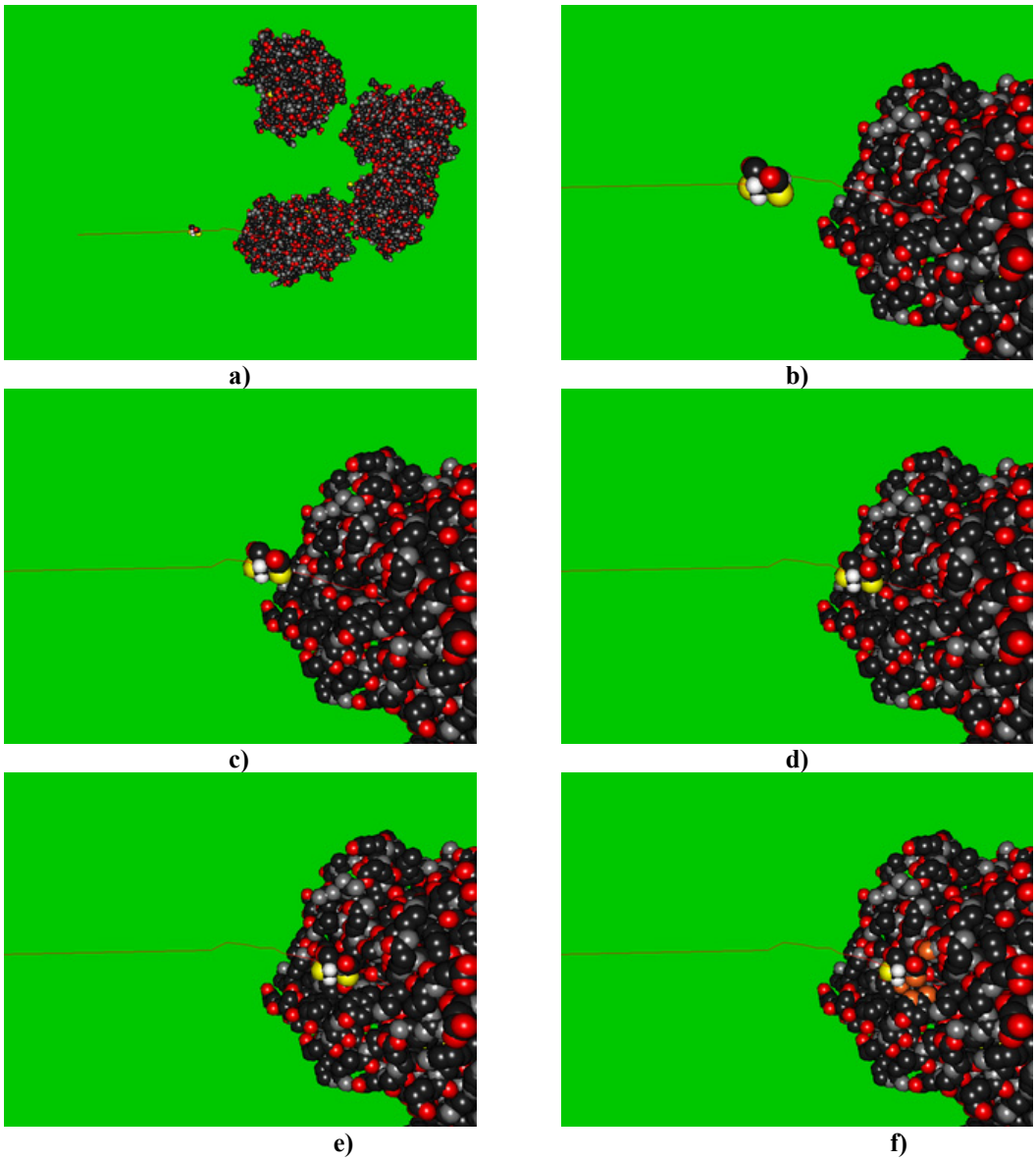
7.3 Popis výsledné aplikace

Do výsledné aplikace se podařilo implementovat jak interaktivní, tak i neinteraktivní režim, jejichž části jsou popsány v kapitole 4. Všechny možnosti interaktivního a neinteraktivního režimu byly implementovány pouze pomocí různým druhů kontrolerů.

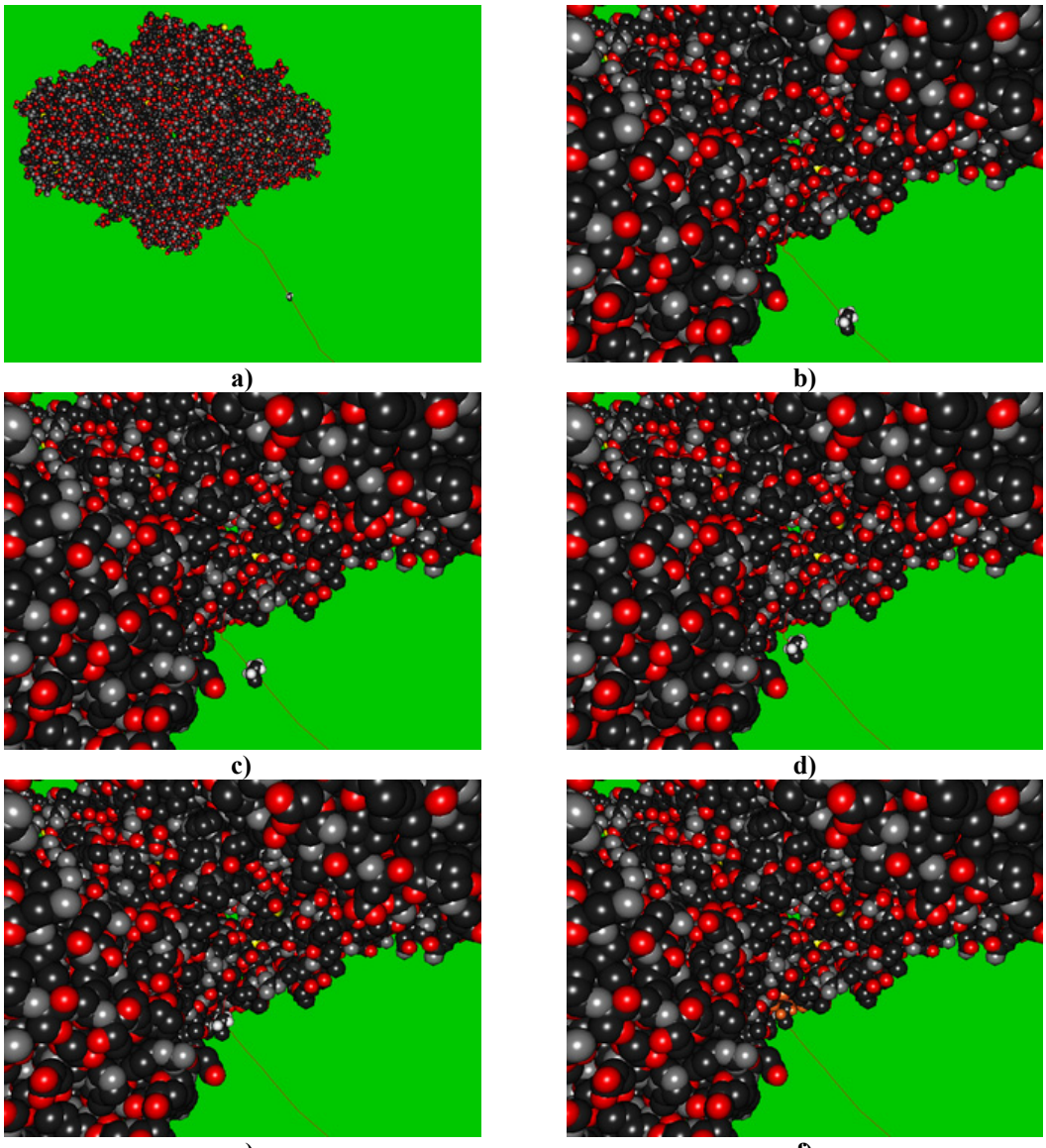
Interaktivní režim tedy obsahuje prohlížení jednoho proteinu, pozorování pohybu malého proteinu po cestě velikým proteinem, zobrazení hledání cesty z velikého proteinu vytřepáváním, prohlížení sekvence snímků proteinu a pohybu malého proteinu v cestě filmem proteinů. Neinteraktivní režim v aplikaci obsahuje ověření polohy dvou proteinů vůči sobě, ověření pohybu malého proteinu v cestě velikým proteinem, výpočet cesty z proteinu vytřepáváním a ověření pohybu malého proteinu v cestě filmem proteinů. Navíc se pro potřeby projektu LC06008 podařilo implementovat možnost nastavení počtu kolizí při průchodu menšího proteinu větším po cestě.

Nastavení počtu kolizí má tři možnosti. První možnost je povolení všech kolizí. Při tomto nastavení se reakce na kolize mezi proteiny chovají standardně. Jestliže nastane kolize a protein nemůže pokračovat po cestě ani obejít kolizní atom, zarazí se na místě kolize. Druhá možnost nastavení je, že při kolizi je dynamickému proteinu povoleno pokusit se obejít kolizní atom. Jestliže dojde k zastavení pohybu dynamického proteinu je statický kolizní atom vyřazen z testování a dynamický jim může projít. Při zapnutí poslední možnosti nastavení počtu kolizí je při každé kolizi dynamického proteinu se statickým atomem tento atom vyřazen z dalšího testování. Vyřazením statického atomu z testování je opět dynamickému proteinu umožněno jím projít. Všechny tři možnosti nastavení aplikace byly přidány, aby se mohly zjistit pouze kolidující atomy statického proteinu s dynamickým proteinem na celé cestě.

Na sekvencích obrázků 7.1 a 7.2 získaných pomocí výsledné aplikace jsou zobrazeny pohyby dynamického proteinu po cestě statickým proteinem.

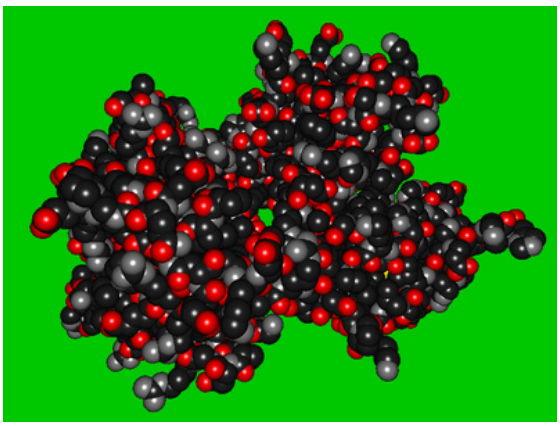


Obr. 7. 1 a) až f) – pohyb dynamického proteinu z 16 atomů po cestě ke statickému z 12 555 atomů

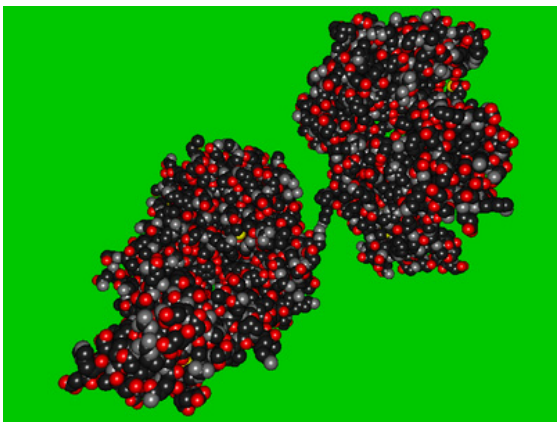


Obr. 7. 2 a) až f) – pohyb dynamického proteinu z 10 atomů po cestě ke statickému z 32 802 atomů

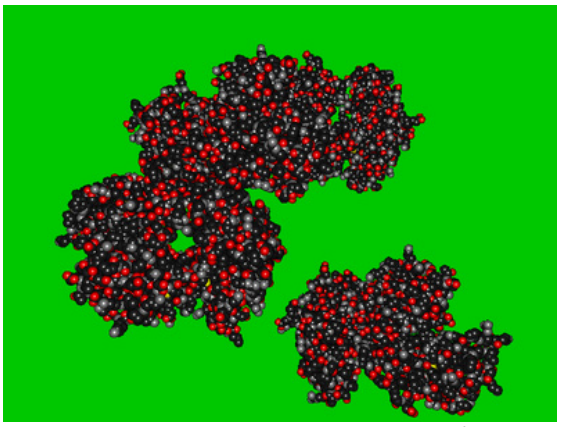
Obrázky 7.3 až 7.6 ukazují statické proteiny zobrazené výslednou aplikací, jež jsou použity při testování.



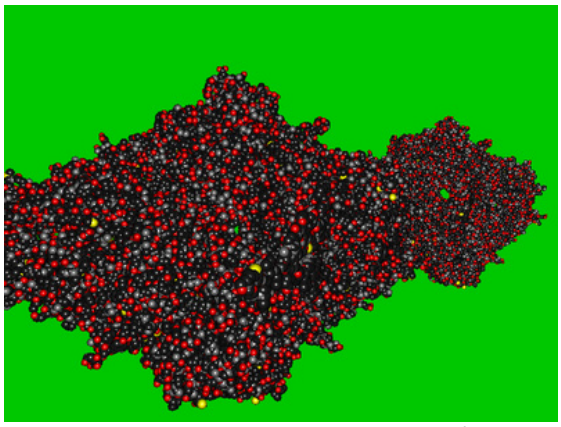
Obr. 7. 3 - protein z 3 185 atomů



Obr. 7. 4 - protein z 6 298 atomů



Obr. 7. 5 - protein z 12 555 atomů



Obr. 7. 6 - protein z 66 254 atomů

Dále aplikace obsahuje implementaci datových struktur: pravidelná mřížka, prostorový strom, sada segmentových stromů a intervalový strom. Změny v implementaci oproti předchozímu řešení jsou popsány v kapitole 7.1. Výsledná aplikace umožňuje vybírat datovou strukturu, která bude použita pro rozdělení prostoru a zjišťování kolizí.

Následující kapitola je věnována testování datových struktur implementovaných ve výsledné aplikaci na reálných datech.

8. Testování

V této kapitole budou popsány testy provedené na cílové aplikaci, jejichž cílem bylo zjistit, která z popsaných struktur by byla nejvhodnější pro detekci kolizí mezi proteiny. Aplikace byla vytvořena v jazyce C++ a testy probíhaly na stejném osobním počítači, na kterém byla testována upravená aplikace mého předchůdce (viz kapitola 6).

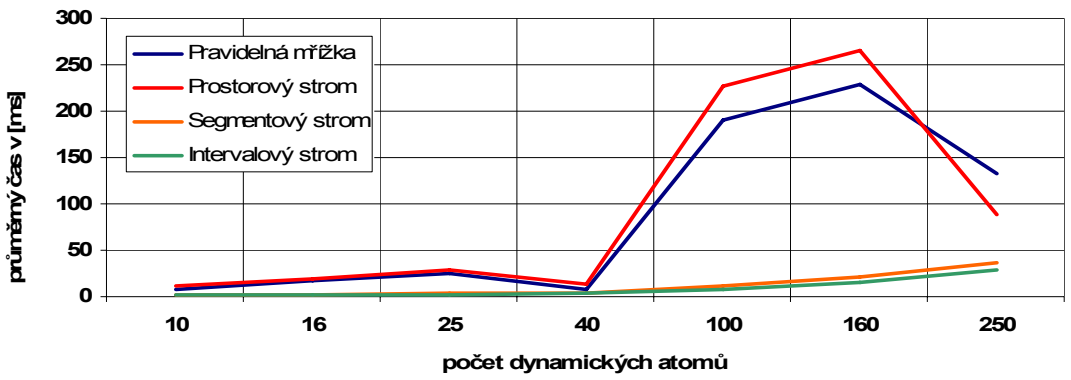
Měření probíhalo v neinteraktivním režimu aplikace a pro každou hodnou uvedenou v tabulce bylo nejprve naměřeno 1 000 hodnot, z kterých byl nakonec vypočítán aritmetický průměr. Měření probíhalo pouze na reálných datech poskytnutých Masarykovou univerzitou v Brně a nalezených v [10]. Testů bylo provedeno několik druhů a budou popsány v následujícím odstavci.

8.1 Průchod proteinem

V tomto testu prochází jedním proteinem druhý po předem stanovené cestě. Dynamický protein se vždy pohyboval směrem do statického proteinu, ale kvůli své velikosti se dovnitř nikdy nedostal. Jeho pohyb po cestě končil kolizemi s atomy na povrchu statického proteinu. Tento test by tedy měl prověřit chování jednotlivých struktur, kdy se menší protein pomalu přibližuje k většímu. Skupina atomů, která tvořila dynamický protein, byla vybrána ze souborů nalezených v [10]. Oběma proteinům byl postupně zvyšován počet atomů. Cesty používané v tomto testu byly vypočteny pomocí aplikace z [15]. Při testování každé struktury byly vždy použity stejné proteiny a cesty. Výsledky měření můžete vidět v tabulkách 8.1 až 8.5 a grafech 8.1 až 8.5.

Počet statických atomů 3185							
Typ struktury	Počet dynamických atomů						
	10	16	25	40	100	160	250
Pravidelná mřížka	8	17	25	8	191	228	133
Prostorový strom	11	20	29	14	227	266	88
Segmentový strom	1	2	3	4	12	22	36
Intervalový strom	1	1	2	3	8	15	28

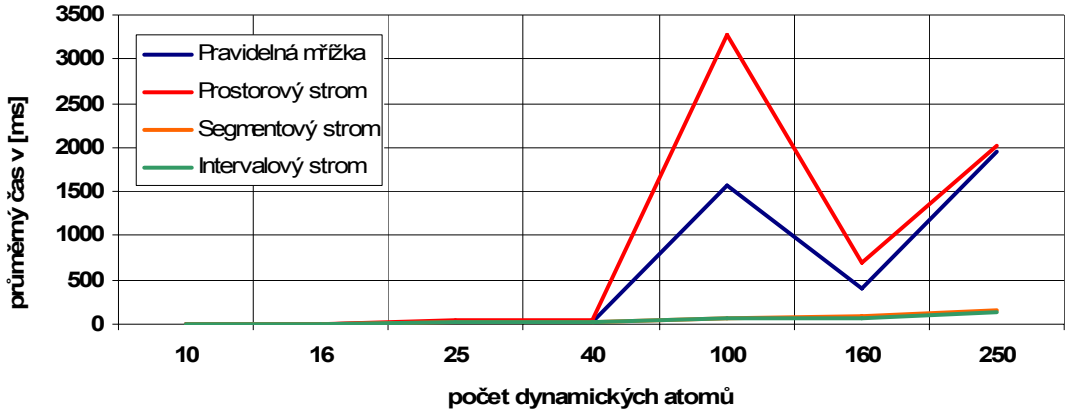
Tab. 8. 1: Naměřené průměrné časy v [ms] ve scéně obsahující 3 185 statických atomů



Graf 8. 1: Průměrné časy v [ms] ve scéně obsahující 3 185 statických atomů

Počet statických atomů 6298							
Typ struktury	Počet dynamických atomů						
	10	16	25	40	100	160	250
Pravidelná mřížka	8	8	17	20	1570	408	1954
Prostorový strom	8	11	39	49	3286	685	2015
Segmentový strom	6	9	14	23	65	85	157
Intervalový strom	6	9	14	21	60	77	145

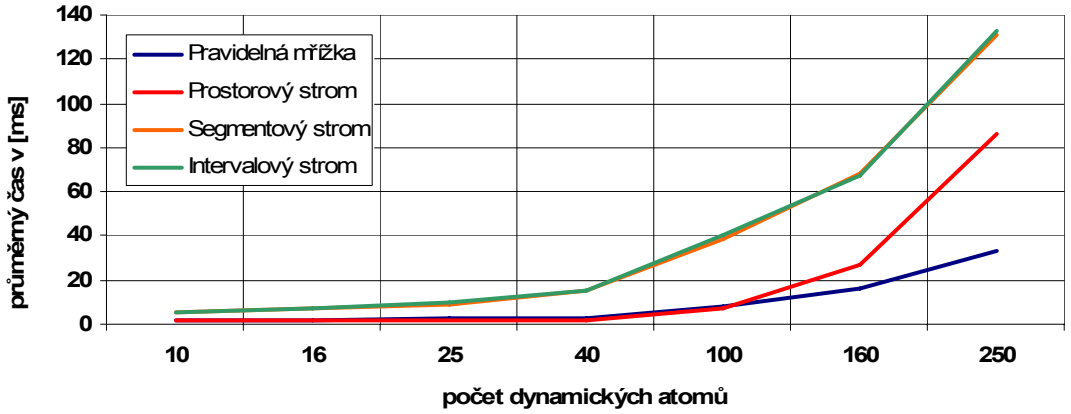
Tab. 8. 2: Naměřené průměrné časy v [ms] ve scéně obsahující 6 298 statických atomů



Graf 8. 2: Průměrné časy v [ms] ve scéně obsahující 6 298 statických atomů

Počet statických atomů 12555							
Typ struktury	Počet dynamických atomů						
	10	16	25	40	100	160	250
Pravidelná mřížka	2	2	3	3	8	16	33
Prostorový strom	2	2	2	2	7	27	86
Segmentový strom	5	7	9	15	39	68	131
Intervalový strom	5	7	10	15	40	67	133

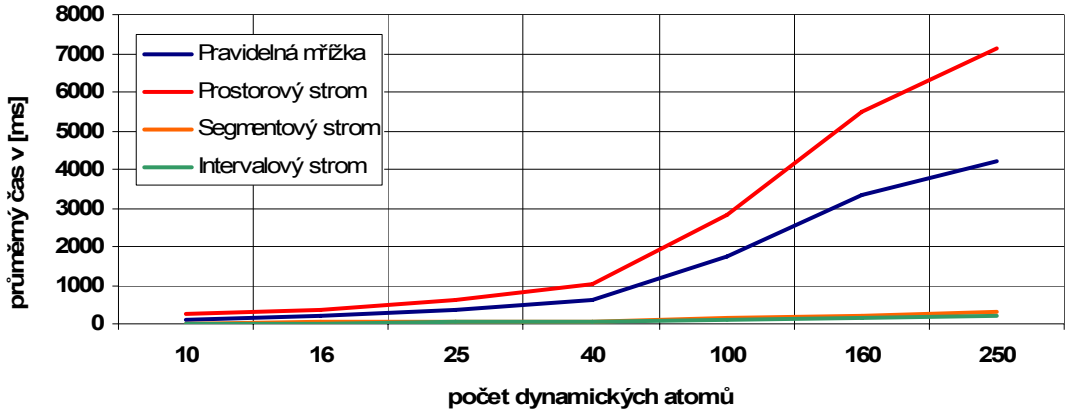
Tab. 8. 3: Naměřené průměrné časy v [ms] ve scéně obsahující 12 555 statických atomů



Graf 8. 3: Průměrné časy v [ms] ve scéně obsahující 12 555 statických atomů

Počet statických atomů 25415							
Typ struktury	Počet dynamických atomů						
	10	16	25	40	100	160	250
Pravidelná mřížka	128	204	337	625	1763	3327	4193
Prostorový strom	231	364	606	1026	2811	5473	7121
Segmentový strom	17	27	39	61	134	194	284
Intervalový strom	16	23	34	52	109	155	215

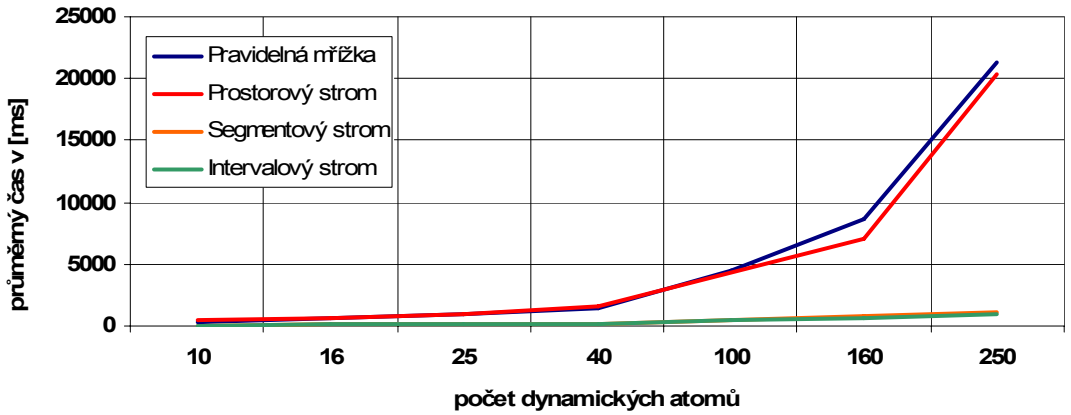
Tab. 8. 4: Naměřené průměrné časy v [ms] ve scéně obsahující 25 415 statických atomů



Graf 8. 4: Průměrné časy v [ms] ve scéně obsahující 25 415 statických atomů

Počet statických atomů 66254							
Typ struktury	Počet dynamických atomů						
	10	16	25	40	100	160	250
Pravidelná mřížka	375	595	930	1511	4544	8595	21312
Prostorový strom	411	653	1024	1662	4303	7084	20344
Segmentový strom	67	94	142	217	515	776	1106
Intervalový strom	70	97	136	198	451	673	955

Tab. 8. 5: Naměřené průměrné časy v [ms] ve scéně obsahující 66 254 statických atomů



Graf 8. 5:Průměrné časy v [ms] ve scéně obsahující 66 254 statických atomů

Z naměřených hodnot v tabulkách 8.1 až 8.5 vyplývá, že nejvhodnějšími strukturami jsou Segmentový strom a Intervalový strom, protože téměř ve všech

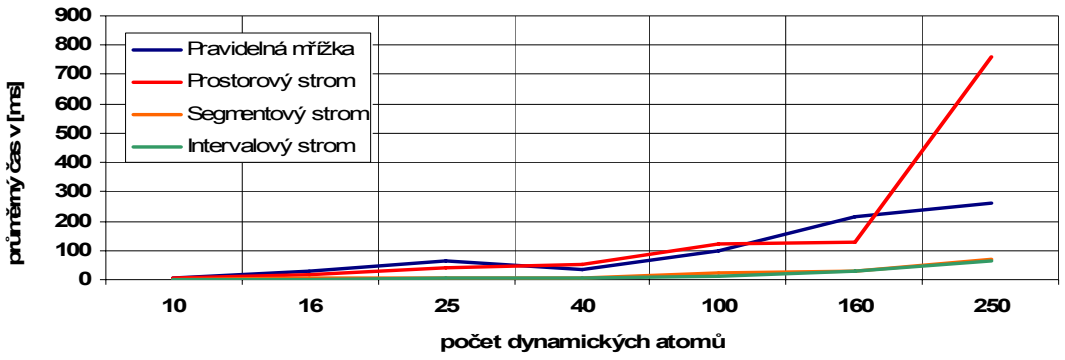
měřeních jsou rychlejší než struktury Pravidelná mřížka a Prostorový strom. Pouze ve scéně s 12 555 statickými atomy (tab. 8.3) jsou tyto struktury rychlejší oproti Segmentovému a Intervalovému stromu. Rozdělení statických atomů v této scéně tedy lépe vyhovuje strukturám Pravidelná mřížka a Prostorový strom, ale v ostatních scénách jsou tyto struktury velice pomalé. Statický protein se v této scéně skládá ze tří přibližně stejně velikých částí. Největší rozdíl v naměřených rychlostech mezi těmito dvojicemi struktur lze vidět v tabulkách 8.4 a 8.5, kde je rozdíl až 20 sekund. Z tabulek dále vyplývá, že mezi naměřenými hodnotami u Segmentového stromu a Intervalového stromu jsou jen malé rozdíly.

8.2 Náhodný pohyb

Dynamický protein se v tomto testu náhodně pohybuje uvnitř statického, kde se snaží naleznout cestu ven. Jako výchozí místo pro dynamický protein byl v každém statickém proteinu zvolen předposlední uzel cesty používané v předchozím testu. Jak se ukázalo v předchozích testech, nemůže se dynamický protein pohybovat uvnitř statického kvůli své velikosti. Jednotlivé atomy statického a dynamického proteinu mají poloměry od 1,2A do 1,85A a dynamický protein se skládá z 10 až 250 atomů. Nejvhodnější nalezené tunely pomocí aplikace z [15] pro testování měly minimální poloměr 0,9A. Aby se mohl dynamický protein pohybovat uvnitř statického a mohl naleznout cestu ven, byl vždy zmenšen tak, aby jeho obalová koule měla právě velikost 0,9A. Při testování se postupně zvětšuje počet atomů obou proteinů. Tabulky a grafy 8.6 až 8.10 ukazují naměřené hodnoty.

Počet statických atomů 3185							
Typ struktury	**Počet dynamických atomů**						
	10	**16**	**25**	**40**	**100**	**160**	**250**
Pravidelná mřížka	7	31	64	32	98	212	262
Prostorový strom	8	15	39	51	123	125	762
Segmentový strom	1	3	4	7	21	28	67
Intervalový strom	1	2	3	5	14	28	61

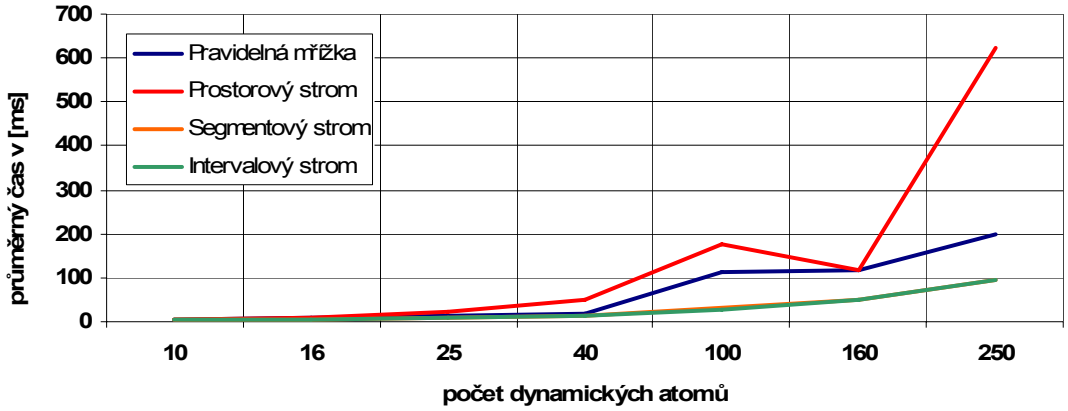
Tab. 8. 6: Naměřené průměrné časy v [ms] ve scéně obsahující 3 185 statických atomů



Graf 8. 6: Průměrné časy v [ms] ve scéně obsahující 3 185 statických atomů

Počet statických atomů 6298							
Typ struktury	Počet dynamických atomů						
	10	16	25	40	100	160	250
Pravidelná mřížka	6	9	15	18	112	119	200
Prostorový strom	6	10	23	49	177	118	623
Segmentový strom	3	6	9	12	32	50	96
Intervalový strom	4	5	9	12	27	48	94

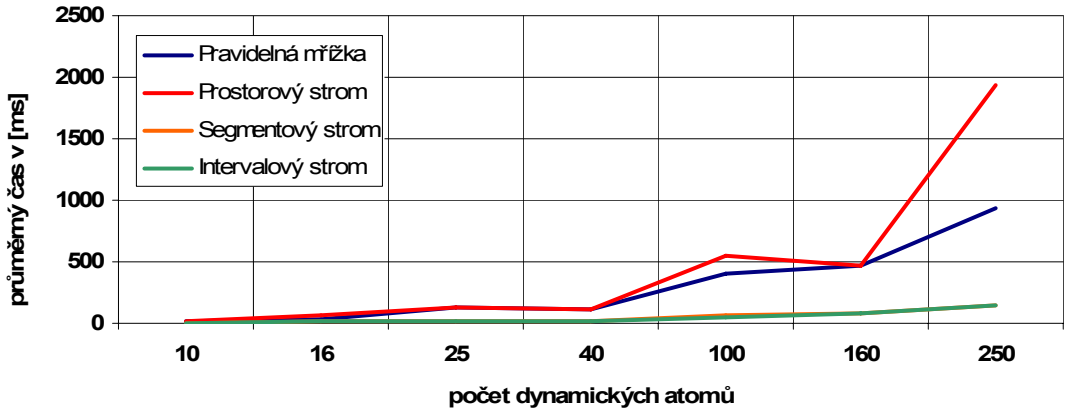
Tab. 8. 7: Naměřené průměrné časy v [ms] ve scéně obsahující 6 298 statických atomů



Graf 8. 7: Průměrné časy v [ms] ve scéně obsahující 6 298 statických atomů

Počet statických atomů 12555							
Typ struktury	Počet dynamických atomů						
	10	16	25	40	100	160	250
Pravidelná mřížka	24	33	126	108	403	461	936
Prostorový strom	20	63	129	105	544	474	1930
Segmentový strom	6	10	15	21	60	83	150
Intervalový strom	7	10	13	21	50	76	142

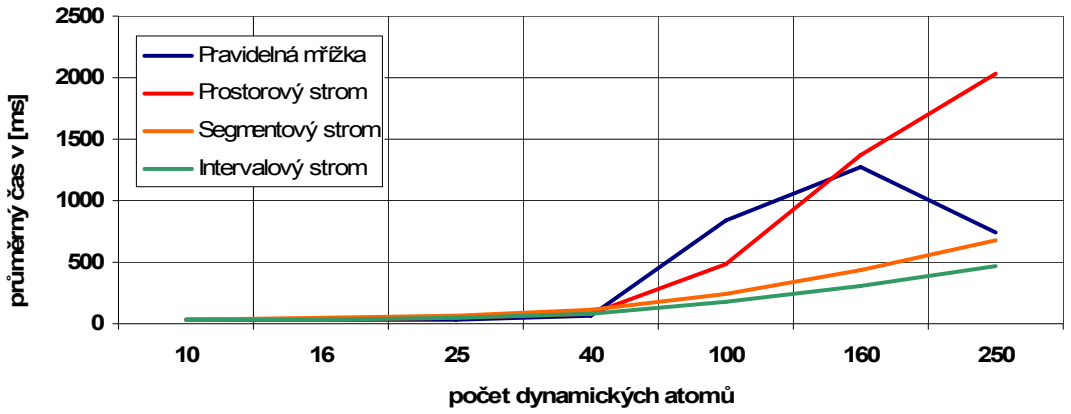
Tab. 8. 8: Naměřené průměrné časy v [ms] ve scéně obsahující 12 555 statických atomů



Graf 8. 8: Průměrné časy v [ms] ve scéně obsahující 12 555 statických atomů

Počet statických atomů 25415							
Typ struktury	Počet dynamických atomů						
	10	16	25	40	100	160	250
Pravidelná mřížka	40	30	37	64	842	1280	746
Prostorový strom	35	27	49	76	482	1371	2037
Segmentový strom	32	49	70	110	235	439	682
Intervalový strom	25	38	53	79	183	306	465

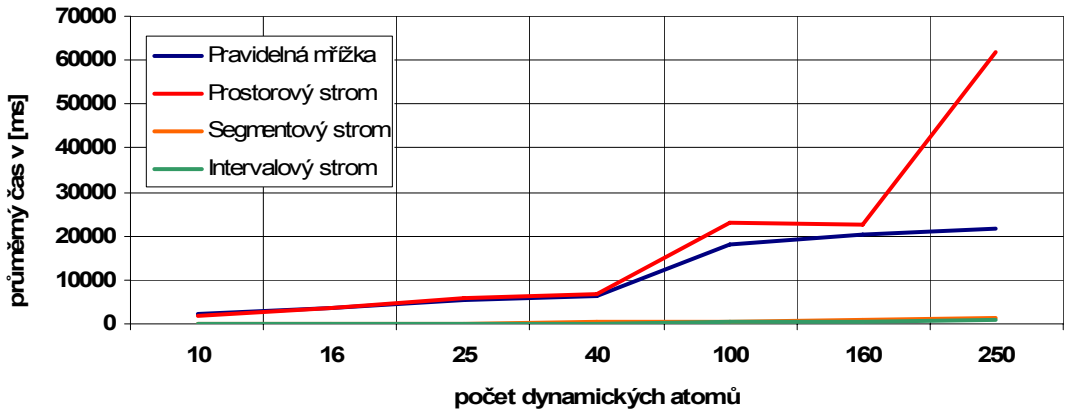
Tab. 8. 9: Naměřené průměrné časy v [ms] ve scéně obsahující 25 415 statických atomů



Graf 8. 9: Průměrné časy v [ms] ve scéně obsahující 25 415 statických atomů

Počet statických atomů 66254							
Typ struktury	Počet dynamických atomů						
	10	16	25	40	100	160	250
Pravidelná mřížka	2177	3408	5209	6377	17912	20156	21535
Prostorový strom	1688	3393	5989	6955	22933	22375	62031
Segmentový strom	68	99	142	232	517	876	1189
Intervalový strom	54	93	115	178	394	673	1078

€Tab. 8. 10: Naměřené průměrné časy v [ms] ve scéně obsahující 66 254 statických atomů



Graf 8. 10: Průměrné časy v [ms] ve scéně obsahující 66 254 statických atomů

Naměřené časy v tabulkách 8.6 až 8.10 ukazují, že struktury Pravidelná mřížka a Prostorový strom jsou téměř ve všech případech výpočetně náročnější než zbylé dvě struktury. Největší rozdíly mezi naměřenými časy jsou v případech, kdy se scénou pohybuje 250 atomů. V tabulce 8.10 se rozdíl mezi Prostorovým stromem

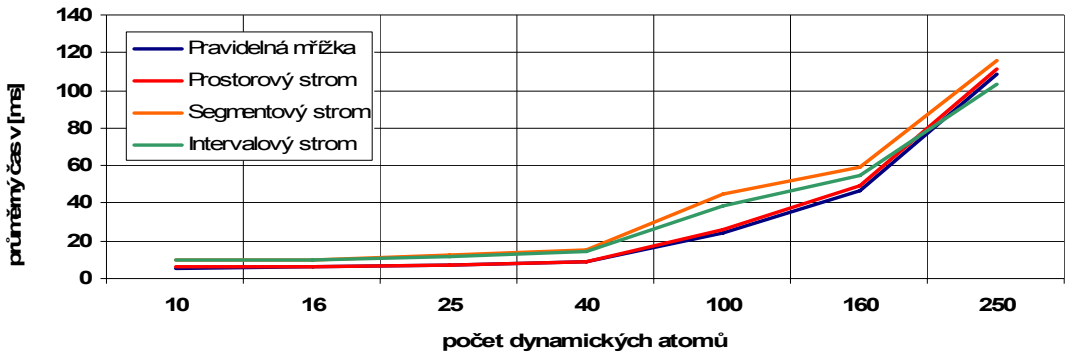
a Segmentovým stromem a mezi Prostorovým stromem a Intervalovým stromem vyšplhal až na hodnotu 60 sekund, což je veliký rozdíl ukazující, že tyto struktury jsou pro tyto velikosti proteinů a hlavně pro jejich rozložení nevhodné. Ve všech tabulkách je opět vidět, že mezi naměřenými časy u Segmentového stromu a Intervalového stromu jsou jen malé rozdíly.

8.3 Pohyb ve filmu proteinu

Při tomto testu se ověřovalo chování jednotlivých metod při průchodu menšího proteinu kmitajícím proteinem. Jestliže při tomto testu proběhla kolize mezi dynamickým a kmitajícím proteinem, došlo ke změně poloh všech atomů kmitajícího proteinu a dynamický protein se vrátil na předchozí pozici. Při měření bylo použito 15 snímků poloh kmitajícího proteinu. Opět se měřil průměrný čas za 1 000 kroků dynamického proteinu a navíc i maximální čas změny poloh všech atomů kmitajícího proteinu. Použitá cesta byla opět vytvořena pomocí aplikace z [15]. Maximální poloměr tunelu při této cestě je 0,5A, a proto byl každý dynamický protein zmenšen tak, aby jeho obalová koule měla velikost právě 0,5A. Aby bylo zaručeno, že dojde k několika kolizím při průchodu po cestě a zároveň nedojde k úplnému zastavení dynamického proteinu, byl snímek proteinu, ve kterém byla cesta vytvořena, posunut na poslední pozici v použitých snímcích. Jestliže v každém snímku dojde ke kolizi, nastane změna snímku. Poslední snímek je ten, ve kterém byla vytvořena cesta kmitajícím proteinem a tedy i snímek, kterým dynamický protein projde bez kolize. Naměřené časy jsou v tabulkách 8.11 a 8.12 a grafu 8.11.

Počet atomů 4696							
Typ struktury	Počet dynamických atomů						
	10	16	25	40	100	160	250
Pravidelná mřížka	5	6	7	9	24	47	109
Prostorový strom	6	6	7	9	26	49	111
Segmentový strom	10	10	13	15	45	59	116
Intervalový strom	10	10	12	14	39	55	103

Tab. 8. 11: Naměřené průměrné časy v [ms] ve scéně obsahující kmitající protein z 4 696 atomů



Graf 8. 11: Průměrné časy v [ms] ve scéně obsahující kmitající protein z 4 696 atomů

Počet atomů v jednom snímku	Typ struktury			
	Pravidelná mřížka	Prostorový strom	Segmentový strom	Intervalový strom
4696	0	172	328	502

Tab. 8. 12: Maximální čas aktualizace jednoho snímku proteinu v [ms]

Z hodnot v tabulkách 8.11 a 8.12 se jeví, že by nejvhodnější metodou pro ověřování cesty v kmitajícím proteinu mohla být struktura Pravidelná mřížka, protože její průměrné časy jsou nejmenší, stejně jako maximální čas aktualizace kmitajícího proteinu. Na druhou stranu se při zvětšování počtu atomů v dynamickém proteinu průměrné časy u ostatních metod, hlavně u Segmentového stromu a Intervalového stromu, začínají přibližovat k naměřeným hodnotám u Pravidelné mřížky. Toto přibližování hodnot je zapříčiněno tím, že při zvětšování počtu atomů v dynamickém proteinu se u metody Pravidelná mřížka zvětšuje čas detekce kolize.

8.4 Výsledky měření

Z výsledku měření vyplývá, že pro zjišťování kolizí při pohybu malého proteinu ve větším a pro hledání cesty z aktivního místa v proteinu mimo něj je nejlepší použít buď strukturu Segmentový strom, nebo Intervalový strom. Struktura Intervalový strom vykazuje v některých případech lepší výsledek než struktura Segmentový strom. Ale i přesto je výhodnější použít strukturu Segmentový strom, protože inicializace Intervalového stromu pro scény s větším počtem atomů je příliš dlouhá. Například ve scéně s 66 254 statickými atomy trvá inicializace až několik desítek minut.

Pro zjišťování kolizí při pohybu menšího proteinu v kmitajícím proteinu po předem zadané cestě se jako nejvhodnější struktura jeví Pravidelná mřížka, která má nejmenší čas aktualizace snímku proteinu a zároveň nejmenší průměrný čas. Při testování se ale ukázalo, že při zvyšování počtu atomů v dynamickém proteinu se průměrné časy jednotlivých struktur začínají přibližovat a při hodnotě 250 atomů u dynamického proteinu jsou téměř totožné. To je zapříčiněno tím, že u struktury Pravidelná mřížka a Prostorový strom dochází při zvyšování počtu jak dynamických, tak statických atomů k prodlužování času detekce kolize. Navíc byl k dispozici pouze kmitající protein s 4 696 atomy. Proto nelze jednoznačně doporučit Pravidelnou mřížku jako strukturu pro detekce kolize v kmitajícím proteinu, který má více jak 5 tis. atomů. Určitým kompromisem pro takovéto scény by mohla být struktura Segmentový strom, která nemá příliš vysoký čas aktualizace snímku proteinu a zároveň při zvyšování počtu statických a dynamických proteinů ve scéně nedochází k přílišnému prodlužování času detekce jako u Pravidelné mřížky nebo Prostorového stromu.

9. Závěr

Tato diplomová práce měla dva hlavní cíle. Prvním cílem bylo otestovat dosavadní řešení pana Ing. Martina Pokorného [9] na reálných datech v podobě proteinů a navrhnout změny v jeho řešení. Před otestováním dosavadního řešení na něm byly provedeny menší změny, které odstranily zbytečné výpočty při detekci kolizí mezi proteiny. Po otestování pozměněného dosavadního řešení vyplynuly z výsledků měření změny v dosavadním řešení. Mezi hlavní změny patřila změna v implementaci datových struktur, v načítání objektů a ve vytvoření nové části pro vykreslování proteinů.

Druhým cílem práce bylo implementovat navržené změny do cílové aplikace a tu následně otestovat na reálných datech. Výsledky měření na cílové aplikaci ukázaly, že nejvhodnějším řešením by mohla být struktura Segmentový strom.

V dalším možném rozvoji aplikace by bylo vhodné zaměřit se na urychlení celkového času detekce kolize nebo aktualizace snímku pomocí paralelního zpracování.

10. Zdroje

[1] Beneš, P.: *Voroného diagramy v molekulární chemii* (diplomová práce), Masarykova univerzita, Brno, 2006.

[2] Bergen, G.: *Collision Detection in Interactive 3D Computer Animation*, University Press Facilities, Eidhoven, 1999. ISBN 90-386-0671-0.

[3] Branden, C., Tooze, J.: *Introduction to Protein Structure*, 1999.

[4] DirectX Documentation in C++. *DirectX 9 Software Development Kit* (online), Microsoft, 5, 2006. < http://msdn.microsoft.com/downloads/>.

[5] *Chime*. < http:// www.mdl.com/chime/ >.

[6] Kozlíková, B.: *Interaktivní vizualizace molekulárních modelů* (diplomová práce), Masarykova univerzita, Brno, 2006.

[7] Matoušek, J.: *Detekce kolizí objektů reprezentovaných trojúhelníkovými modely* (diplomová práce), Západočeská univerzita v Plzni, 2004.

[8] Molecule analysis and Molecule display - MolMol.
< http://hugin.ethz.ch/wuthrich/software/molmol/ >.

[9] Pokorný, M.: *Detekce kolizí vhodná pro pohybující se modely v oblasti VR* (diplomová práce), Západočeská univerzita v Plzni, 2006.

[10] *Protein Data Bank*. < http://www.rcsb.org>.

[11] *PyMol*. < http://pymol.sourceforge.net/ >.

[12] *RasMol*. < http://www.umass.edu/microbio/rasmol/ >.

[13] *Visual Molecular Dynamics - VMD*. <http://www.ks.uiuc.edu/Research/vmd/ >.

[14] Watt, A., Policarpo, F.: *3D Games - Real-time Rendering and Software Technology*, Volume One, Addison Wesley. ISBN 0-201-61921-0.

[15] Zemek, M.: *Dělení prostoru pro rozsáhlá a měnící se data* (diplomová práce), Západočeská univerzita v Plzni, 2007.

[16] Žára, J., Beneš, B., Sochor, J., Felkel, P.: *Moderní počítačová grafika*, Computer press, 2004. ISBN 80-251-0454-0.

Evidenční list

Souhlasím s tím, aby moje diplomová práce byla půjčována k prezenčnímu studiu v Univerzitní knihovně ZČU v Plzni.

Datum: Podpis:

Uživatel stvrzuje svým čitelným podpisem, že tuto diplomovou práci použil ke studijním účelům a prohlašuje, že ji uvede mezi použitými prameny.

Jméno	Fakulta/Katedra	Datum	Podpis