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

Diplomová práce

Lokální navigace chodců ve virtuálních modelech měst

Plzeň, 2014 Pavel Brandejský

Zadání

Prohlášení

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

V Plzni dne ..... Pavel Brandejský

.....

Abstract

This thesis is about local navigation of pedestrians in virtual city models. The problem is very popular and there is a lot of methods how to solve it. First part of this thesis presents some of these methods, than a new approach is presented. This new approach is mainly based on the method called ClearPath: Highly Parallel Collision Avoidance for Multi-Agent Simulation.

Abstrakt

Tato práce pojednává o lokální navigaci chodců ve virtuálních modelech měst. Tento problém je v poslední době velmi aktuální, takže existuje mnoho metod jak ho řešit. V první části práce jsou představeny některé z těchto metod, a následně je představeno vlastní řešení. Toto řešení je založeno na metodě zvané ClearPath: Highly Parallel Collision Avoidance for Multi-Agent Simulation.

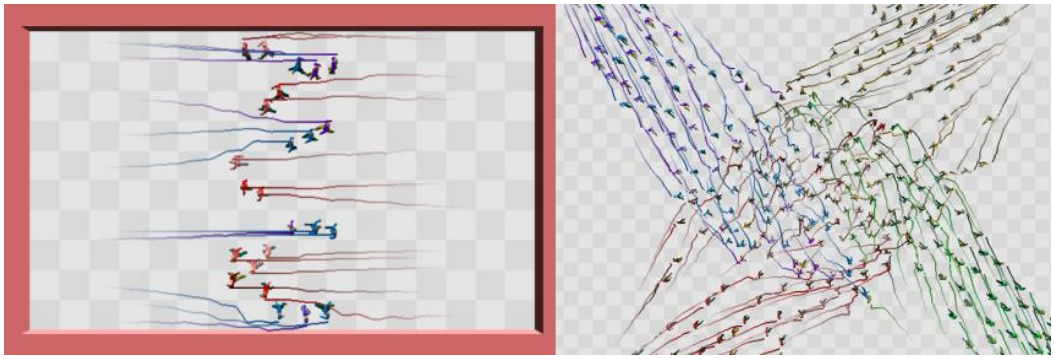
Poděkování

Rád bych poděkoval vedoucí této diplomové práce prof. Dr. Ing. Ivaně Kolingerové za užitečné rady, připomínky a čas, který mi věnovala. Poděkování patří také mým rodičům za jejich podporu po celou dobu mého studia.

Obsah

1 Úvod

Simulace pohybu chodců má několik uplatnění různících se svými požadavky na výsledek. Při použití v počítačových hrách je kladen důraz na co nejmenší výpočetní náročnost, protože simulace musí běžet v reálném čase a nesmí omezovat ostatní moduly hry. V simulaci také nesmí docházet k chybám, které případný hráč jednoduše rozezná, příkladem může být uvíznutý voják v rohu místnosti, který se stále snaží projít zdí. Další uplatnění simulací pohybu chodců je ve filmovém průmyslu, kde se simulace používají tam, kde by bylo modelování jedinců příliš složité. Zde je výhodné, že simulace nemusejí probíhat v reálném čase, proto je také možné animace přepočítat s jinými vstupními parametry pokud je zjištěna chyba. Pro filmový průmysl je také výhodné, pokud metoda umožňuje online editaci, model Group Motion Editing **[Lee08]** (úprava pohybu skupin) se například zaměřuje na udržení skupiny chodců ve formaci, čehož lze využít při simulování pohybující se armády. Modely simulace chodců lze také uplatnit při návrhu domů či měst. Lze s nimi simulovat očekávaný pohyb v těchto místech nebo pohyb při různých nebezpečích (evakuaci) a hledat slabá místa v návrzích. U takových metod je důležité, aby výsledky simulací byly co nejblíže skutečném chování chodců v tomto prostředí. Tyto metody by měly tvořit pásy a víry viděné v reálných davech, příklady těchto jevů jsou vidět na obrázku 1.1.



(a) **(b)**

Obrázek 1.1 Pásy (a) a víry (b) viděné v reálných davech [Tre06]

Tato diplomová práce se zabývá návrhem řešení pro simulaci chodců ve virtuálních městech. Původně měla být využita v projektu Ing. Tomáše Vomáčky v jeho rozsáhlejším řešení simulace chodců. K přímému propojení prací nedošlo, ale práce i nadále pokračovala tak, aby nebyl problém tyto práce v budoucnu propojit.

Prvním cílem práce bylo nastudovat existující program pro navigaci chodců (jeho spolu-autorem je Ing. Tomáš Vomáčka) používaný na katedře informatiky a výpočetní techniky (KIV) a navrhnout další možný přístup k simulaci chodců (kapitola 2.2.1). Pro složitost původního programu bylo dohodnuto, že bude napsána nová aplikace, která bude umožnovat jednodušší ladění algoritmu. Pro obecný vhled do problematiky bylo pak nastudováno několik článků zabývajících se simulací chodců (kapitoly 2.3, 2.4, 2.5, 2.6, 2.7). Poté bylo navrženo (kapitola 3) a implementováno (kapitola 4) nové řešení. Nakonec bylo nové řešení důkladně otestováno a ukázány jeho výhody a nevýhody (kapitola 5).

Práce byla také napojena na další dva projekty, týkající se stejné problematiky. Jedním byl program pro animování chodců popsaný podrobněji v kapitole 2.2.2. Druhý projekt je program pro počítání cest ve virtuálním městě, popsaný v kapitole 2.2.3.

2 Metody simulace pohybu chodců

V této kapitole jsou vysvětleny společné aspekty modelů pro simulaci pohybu chodců, představeny příklady programového vybavení na KIV a následně je stručně vysvětleno několik příkladů modelů.

2.1 Základní pojmy

Pojem simulace pohybu chodců lze vysvětlit jako řízení pohybu chodce z počátečního bodu A do cílového bodu B tak, aby nedošlo ke kolizi s jiným chodcem ani s žádnou překážkou (taková cesta se nazývá bezkolizní). Prostředí, ve kterém se chodci pohybují, se obvykle označuje jako simulační scéna.

Řízení chodce lze rozdělit na dvě části:

- **Lokální navigace** – řízení jedince tak, aby se vyhnul ostatním jedincům a statickým překážkám (předcházení kolizím – collision avoidance). Příklad lze vidět na obrázku 2.1, kde se jednotliví chodci nepřekrývají a neprocházejí statickými překážkami.
- **Globální navigace** – hledání vhodné cesty do cíle (path planning). Příklad lze opět vidět na obrázku 2.1, kde chodci opouštějí místnosti i jiným směrem, než ve kterém leží jejich skutečný cíl, a žádný neuvízne např. v rohu místnosti.

Obrázek 2.1 Ukázka simulace pohybu chodců v prostředí se statickými překážkami [Guy09]

Metody simulace pohybu chodců lze také rozdělit podle toho, zda je každý chodec řízen individuálně nebo jsou chodci řízeni hromadně:

- **Kontinuální modely** – simulují pohyb davu podobně jako pohyb tekutiny. Jedinci jsou částečně nebo úplně ovládáni pohybem (tokem) ostatních chodců. Výhodou těchto metod je, že náročné vypočítání toků lze většinou použít pro více než jednoho chodce a tím ušetřit výpočetní výkon. Nevýhodou může být nedostatečná možnost parametrizace jednotlivých chodců.
- **Agentové modely** – v těchto modelech je řízen každý chodec individuálně a je často označován jako agent. Pojem agent pochází z oblasti multiagentních systémů a označuje samostatnou entitu umístěnou do určitého prostředí, která vykonává nějakou činnost. U simulace chodců bývá touto činností např.: „jdi na místo XY a přitom se vyhni všem překážkám a ostatním chodcům“. Nevýhodou tohoto přístupu bývá větší časová náročnost výpočtu.

Protože nelze simulaci pohybu chodců počítat pro každý časový okamžik (spojitě), většina dále uvedených modelů simulace pracuje s časovými kroky simulace (také krok iterace). V každém kroku simulace v čase T získá stav simulované scény a vypočítá nový stav pro čas T+ΔT (aktualizují se pozice a rychlosti chodců).

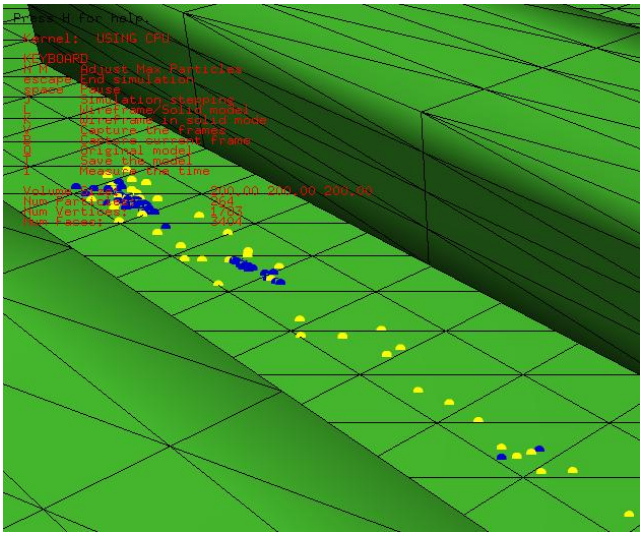
Simulační scéna se skládá ze statických a dynamických překážek. Statickými překážkami jsou označována místa ve scéně, kterými chodec nemůže projít a jejich pozice se vzhledem k scéně nikdy nezmění, na rozdíl od dynamických překážek, které mění svoji pozici. Dynamickou překážkou se rozumí i chodci, protože z pohledu chodce je jiný chodec oblast scény, na které se nemůže přemístit. Dynamické překážky lze ještě rozdělit na interaktivní a neinteraktivní. Ale toto rozdělení již může být subjektivní. Lze říci, že chodci jsou vzájemně interaktivní, všechny páry chodců se vzájemně snaží předejít kolizi, ale automobily ve scéně mohou být simulovány jako neinteraktivní vzhledem k chodci. Automobil se tak žádným způsobem nesnaží vyhnout chodci a chodec musí jejich vzájemné kolizi předejít samostatně.

2.2 Vybrané programové vybavení na KIV

2.2.1 SPH Fluid Simulator

Program pro simulaci chodců ve virtuálním městě *SPH Fluid Simulator for CPU and GPU* (simulátor tekutin pro CPU a GPU, zkratka SPH značí *Smoothed-particle hydrodynamic* a je to metoda výpočtu toku tekutin) vytvořený panem R. Hoetzleinem **[Hot09]** a upravený Ing. Tomášem Vomáčkou patří do kontinuálních simulací. Simuluje

pohyb chodců a zobrazuje ho ve 3D projekci. Program zobrazuje chodce jako barevné kuličky. Existují v něm dva typy simulace chodce, každý chodec musí být simulovaný jako tekutina nebo agent. Chodec typu agent je naimplementován tak, že jde konstantní rychlostí přímo ke svému cíli. Chodec typu tekutina řeší kolize s ostatními chodci a také se statickými překážkami silou odpuzování. Tedy chodci se mohou ocitnout uvnitř překážky (díky velké rychlosti), ale následně jsou její silou odsunuti mimo překážku. Cílem je chodec naopak přitahován. Obrázek 2.2 ukazuje uživatelské prostředí programu. Modré kuličky reprezentují chodce simulované jako tekutina a žlutí jsou implementováni agentovým algoritmem.



Obrázek 2.2: Ukázka z programu SPH Fluid Simulator [Hot09]

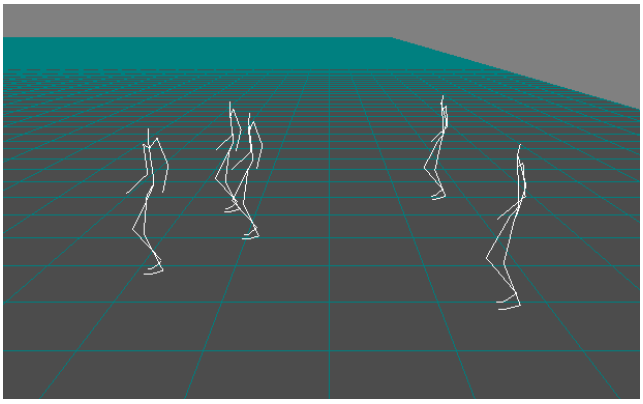
V každém kroku iterace se spustí nejprve metoda, která zkontroluje, jestli je v simulaci dostatečný počet částic (a pokud ne, tak nějaké přidá), odstraní ze simulace staré částice (jsou-li nějaké) a postupně spustí všechny kroky simulace. Celá simulace potom probíhá tak, že se vytvoří pravidelná mřížka nad celým prostorem, ve kterém simulace probíhá, do této mřížky se vloží trojúhelníky, z nichž se skládá prostředí, částice, které budou simulovány, a následně se vypočítá tlak a síla, která na částice působí, a částicemi se pohne podle těchto veličin a času, který uběhl od posledního simulačního kroku. Je dobré si uvědomit, že celá simulace využívá pro zrychlení právě tuto pravidelnou mřížku, takže pokud některá částice hledá své blízké sousedy (což se děje poměrně hodně často), hledá je jen v sousedních polích mřížky.

Ve scéně jsou takzvaná zřídla, která generují chodce. Lze u nich nastavit poměr chodců typu tekutina a agent, plošnou velikost a pozici zřídla a cíle chodců. Chodec prochází cíle od prvního do posledního a poté je smazán.

Program je napsán v programovacím jazyce C++ s využitím knihovny OpenGL pro renderování grafiky a knihovny GLUT pro vytvoření uživatelského rozhraní.

2.2.2 Animation Renderer for Crowd

Animation Renderer for Crowd **[Ben14]** (ARfC - generátor animací pro davy) je program vyvinutý na Purdue University (West Lafayette, Indiana, USA) pro zobrazení výsledků simulace vypočtených programem *SPH Fluid Simulator* popisovaným v předchozí kapitole. Program není dokončený, ale již dokáže zobrazit pohybující se kostru chodce na základě vstupních dat. Vstupní data program načítá z xml souboru a jeho podrobný formát je popsán v příloze 0. Vstupní data obsahují pozici, směr, stav a typ pohybu všech chodců ze všech simulačních kroků. Stav chodce určuje, zda byl chodec simulován jako tekutina nebo agent, pro jednoznačnou identifikaci ve výsledné animaci. Typy pohybu chodce jsou tyto tři: stojí, jde a běží. Aby byla animace plynulá, v čase mezi jednotlivými kroky jsou hodnoty pozice a směru interpolovány. Na obrázku 2.3 je vidět pět chodců animovaných pomocí ARfC.



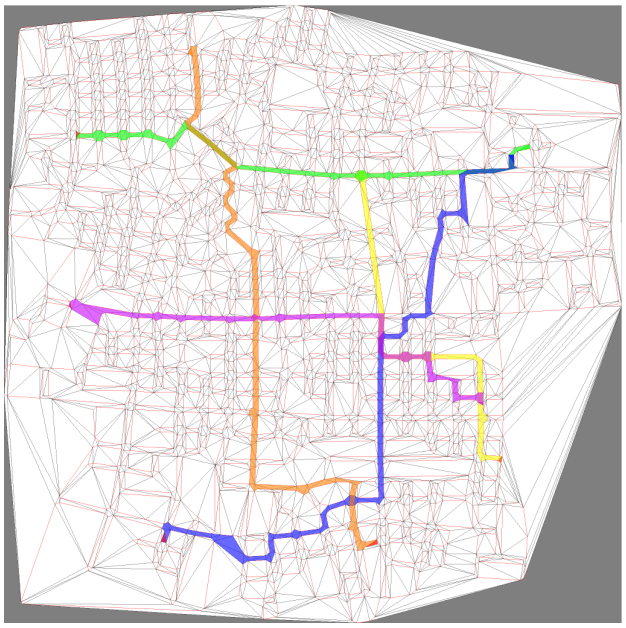
Obrázek 2.3 Ukázka z animace vytvořené programem Animation Renderer for Crowd

2.2.3 Hledání cest ve virtuálním městě

Současně s touto diplomovou prací Bc. Jakub Szkandera vyvíjí program pro hledání cest ve virtuálním městě v závislosti na definovaných preferencích skupin chodců. Ve městě jsou definované plošné oblasti a skupině chodců lze nadefinovat, jak

velký je jejich zájem projít touto oblastí při cestě z bodu A do bodu B. Lze definovat i záporné hodnoty, pokud skupina danou oblastí procházet nechce. Podobně lze nastavit i vztahy mezi skupinami chodců.

Na obrázku 2.4 jsou vidět cesty pěti chodců nalezené tímto programem. Každý chodec má svoji barvu.



Obrázek 2.4 Cesty městem nalezené pomocí preferenčního vyhledávání cest

2.3 Continuum Crowds

V modelu *Continuum Crowds* (kontinuální davy) **[Tre06]** je spojena globální navigace chodce a pohyb překážek, jako jsou ostatní chodci, bez nutnosti dalšího řešení kolizí. Patří do skupiny kontinuálních modelů. Simulace vytvořené tímto modelem jsou počítány v reálném čase.

Model je postaven na čtyřech hypotézách:

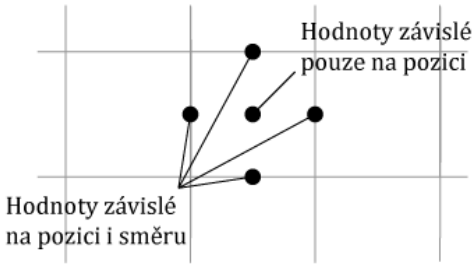
- Každý chodec se stále snaží dosáhnout nějakého místa v mapě (cíl).
- Každý chodec se pohybuje maximální možnou rychlostí.
- Existuje ohodnocení nepohodlí (*discomfort*) místa, takže chodec je raději vždy na místě s menším nepohodlím.

- Při výběru cesty se každý chodec snaží minimalizovat její délku, trvání a míru jejího nepohodlí. Soubor těchto hodnot se označuje jako cena přechodu (*unit cost*).

Pro výpočet optimální cesty chodce je předpokládáno, že je známa potenciálová funkce, která v každém místě simulované scény udává cenu přechodu až do cíle. Pak je pro chodce výhodné se pohybovat přesně opačně proti gradientu této potenciálové funkce. Tato potenciálová funkce se počítá směrem od cíle přes množinu všech optimálních cest. Tedy nejprve se ohodnotí místa nejblíže k cíli a poté okolí míst s nejnižší cenou přechodu. Výpočet této funkce je náročný, ale pokud předpokládáme, že skupina chodců sdílí stejný cíl, hodnoty nepohodlí a rychlost, může se vypočtená funkce použít pro celou skupinu. Takovýchto skupin můžeme mít v simulaci několik.

V reálném světě je rychlost chodce ve volném prostoru závislá na prostředí (chůze po svahu), ale pokud se dostává do míst s vysokou hustotou ostatních chodců, je ovlivňována pohybem ostatních chodců. Této závislosti je v modelu dosaženo zavedením hustoty davu (*crowd density*). Chodec okolo svého umístění hustotu davu zvyšuje. S hustotou davu se také stanovuje průměrná rychlost, indikující celkovou rychlost a směr proudu davu. Rychlost proudu davu je dána jako nezáporná, aby docházelo pouze ke zpomalení (ne otočení směru pohybu). Pro lepší proudění davu se také přidává otisk nepohodlí ve směru pohybu chodce, aby se ostatní chodci tomuto místu vyhnuli.

Pro diskrétní implementaci se použije rozdělení celé simulační scény do mřížky. V centru každého pole mřížky jsou uloženy všechny hodnoty závislé pouze na pozici a na jednotlivých stranách pole jsou uloženy hodnoty závisející také na směru. Obrázek 2.5 takovou mřížku zobrazuje.



Obrázek 2.5 Diskrétní mřížka simulační scény

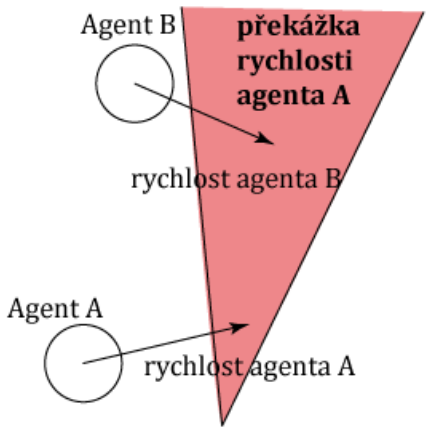
V každém kroku simulace se nejprve vypočítá hustota chodců v každém poli. Poté se pro každou skupinu chodců a pro každý směr pole spočítá cena a rychlost přechodu. Dále se pro každé pole s pomocí potenciálové funkce vypočítá hodnota ceny přechodu z daného pole až do cíle. Každý chodec tak získá požadovaný směr pohybu. Pro každého

chodce skupiny je pak rychlost získána interpolací přes hodnoty rychlosti přechodu nejbližších polí podle pozice chodce a směru pohybu a příslušný chodec je posunut na novou pozici.

Ve finální fázi kroku simulace je iterováno přes všechny páry chodců a ty se symetricky posunou na minimální vzdálenost tak, aby nedocházelo ke kolizím.

2.4 ClearPath

ClearPath (volná cesta) **[Guy09]** je agentově navržený model simulace pohybu davu, který hledá v prvé řadě bezkolizní cesty. Je založený na překážkách rychlosti (*velocity obstacles* - VO) a tento pojem dále rozšiřuje. Překážka rychlosti, jak je vidět na obrázku 2.6, je vlastně geometrické vymezení oblasti, kde nesmí ležet vektor rychlosti agenta, pokud tento agent chce jít po nekolizní cestě. Překážky rychlosti jsou vytvářeny chodcem pro statické i dynamické překážky v jeho blízkosti. Překážky rychlosti i celý simulační model jsou podrobně popsány v kapitole 3.



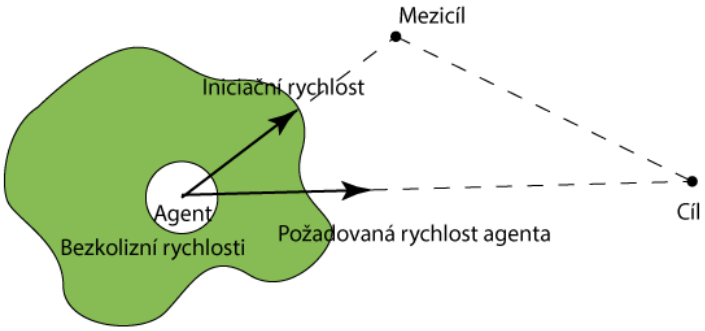
Obrázek 2.6 Překážka rychlosti

ClearPath model řeší pouze lokální navigaci agenta a je nutné vybrat globální navigační metodu, která každému chodci přiřadí preferovanou rychlost pro aktuální simulační krok. S využitím preferované rychlosti je pak vypočítána nová rychlost, která se vyhýbá blízkým překážkám a zároveň minimalizuje odklon od preferované rychlosti.

2.5 PLEdestrians

V modelu *PLEdestrians* **[Guy10]** je kladen důraz na to, aby se chodci dostali ze své stávající pozice na cílovou s využitím minimální energie a bez kolizí s jinými překážkami. Je to jev, který byl pozorován u reálných davů. Odtud je také název metody *least effort pedestrians* (chodci s nejmenším úsilím). Spadá do agentově orientovaných modelů a řídí spíše globální navigaci jednotlivých agentů.

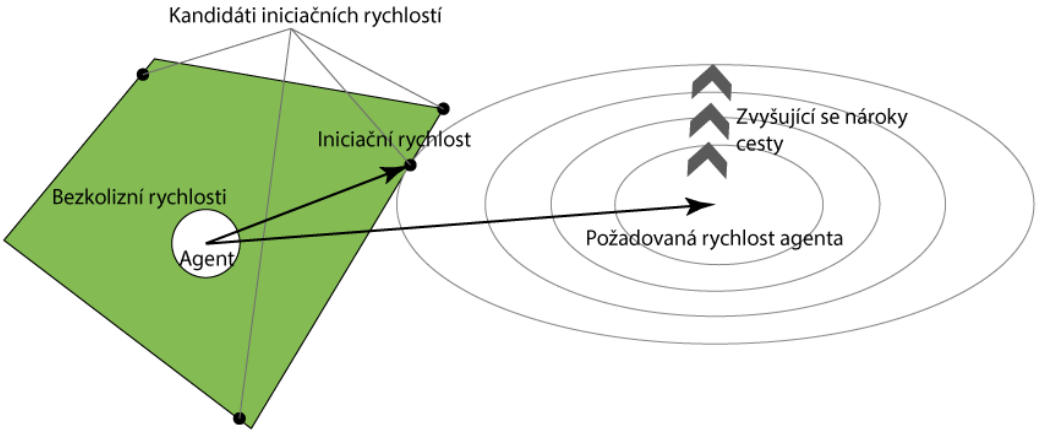
Pro využití minima energie je nutné, aby se agent pohyboval po nejkratší možné cestě a zároveň se pohyboval svou preferovanou rychlostí. Tento model neřeší lokální kolize, ale využívá modelu lokálního předcházení kolizím *Optimal Reciprocal Collision Avoidance* **[Guy11]** (ORCA - optimální vzájemné přecházení kolizím, tato metoda je také založena na rychlostních překážkách) pro zjištění množiny možných bezkolizních rychlostí, tyto rychlosti se také označují jako iniciační. Rychlost je bezkolizní, pokud u agenta pohybujícího se touto rychlostí nedojde v omezeném časovém úseku ke kolizi s překážkou nebo jiným agentem. Z množiny bezkolizních rychlostí je poté nutno vybrat tu, se kterou očekává, že bude využito minimální množství energie k dosažení cíle. Je dobré si uvědomit, že každá cesta do cíle s danou iniciační rychlostí je složená ze dvou částí, jak je zobrazeno na obrázku 2.7. První část je cesta z místa agenta do místa, kde se agent bude nacházet po konečném časovém úseku (mezicíl) s užitím vybrané iniciační rychlosti. Druhou částí je poté cesta z tohoto místa do cíle.



Obrázek 2.7 Ukázka dvoufázové cesty

Pro výpočet nové rychlosti potřebujeme minimalizovat funkci nejmenšího úsilí po celé hranici množiny bezkolizních rychlostí, dané modulem předcházení lokálních kolizí ORCA. Nejprve je nutné rozložit hraniční přímky na úsečky. Hraniční úsečky jsou získány protínáním náhodně permutovaných hraničních přímek. Na každé úsečce se pak vypočítá bod (rychlost), který vyžaduje nejmenší energii pro cestu do cílového bodu.

Z těchto kandidátů vybereme znovu ten bod, který nejvíce minimalizuje funkci nejmenšího úsilí. Výběr vhodné iniciační rychlosti je vidět na obrázku 2.8.



Obrázek 2.8 Ukázka výběru iniciační rychlosti

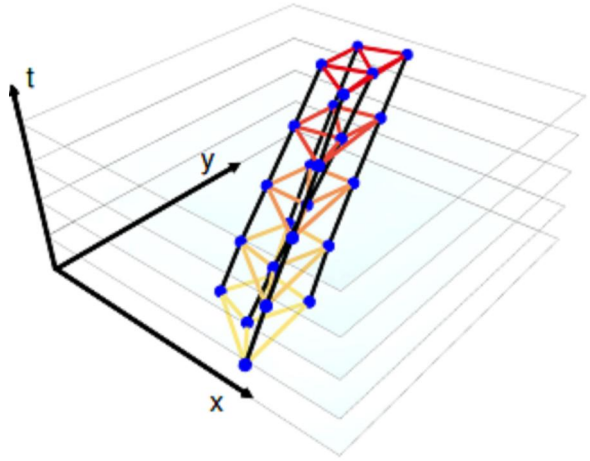
Před začátkem simulace se vypočítá statická mapa (*road map*), která zaručuje bezkolizní cesty pro statické překážky. Tato mapa je realizována jako graf. Agenti mapu využívají pro výpočet své trasy k cíli. Každá hrana tohoto grafu má ohodnocení dané velikostí spotřebované energie při jejím průchodu. Toto ohodnocení se mění každý krok simulace, protože závisí nejen na délce, ale i na množství a průměrné rychlosti agentů cestujících po této hraně.

2.6 Group Motion Editing

Simulační model *Group Motion Editing* (úprava pohybu skupiny) **[Lee08]** se zaměřuje na to, aby byla skupina chodců při pohybu jednoduše upravovatelná. Skupina musí zachovávat formaci a trajektorie jednotlivých chodců, jak jen to je možné. Uživatel může skupinový pohyb měnit přetahováním jedinců nebo naopak fixací. Také je možné skupiny spojovat nebo naopak rozdělovat, a to za úspěšného předcházení kolizím. Je také možné spojovat simulační animace skupin o stejném počtu jedinců do delších animací. Model není určen pro počítání cest v reálném čase a neobsahuje globální navigaci. Tato metoda přistupuje k jednotlivcům hromadně, ale není příkladem kontinuálního modelu. Model nepracuje se simulačními kroky, ale čas simulace rozděluje simulačními snímky.

2.6.1 Konstrukce skupinového grafu

Celá simulace je rozdělena do časových snímků a pro daný čas snímek zobrazuje aktuální pozici všech chodců skupiny. Aby bylo možné současně upravovat několik trajektorií a přitom byly zachovány prostorové relace mezi jedinci, je zde konstruován graf, ve kterém hrany spojují uzly snímané z trajektorií jedinců, jak v čase, tak v prostoru. Kompletní graf pro skupinu o pěti chodcích je vidět na obrázku 2.9. Pro každý snímek (na obrázku roviny kolmé na osu t), existuje graf formace s uzly reprezentujícími chodce (na obrázku modré body), spojený hranami formace (barevné hrany). Graf formace zaručí, že minimální pohyb chodců v rámci skupiny. Tyto snímky jsou propojené časovými hranami (na obrázku černé hrany).



Obrázek 2.9 Skupinový graf [Lee08]

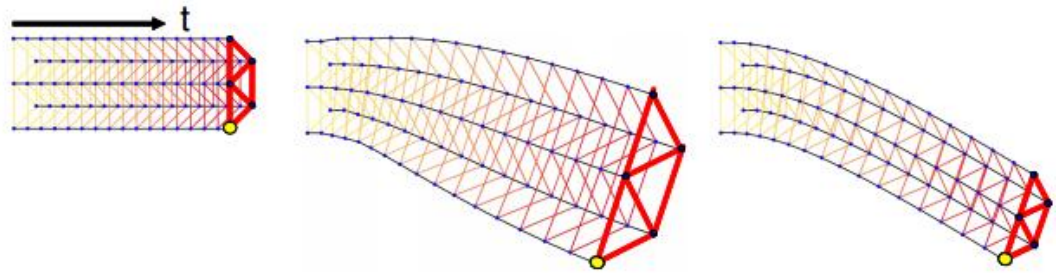
K vytváření hran formace mezi jednotlivci je použita Delaunayho triangulace, která bere ohled na prostorové vzdálenosti a vytvoří mezi jedinci trojúhelníky tak, aby byli spojeni vždy dva nejbližší chodci. I když tato triangulace má hodně předpokládaných kladných vlastností, občas se objeví i nepředvídatelné spojení mezi vzdálenými jedinci. Proto je nutné tyto grafy ručně opravit. Časové hrany jsou jednodušší, propojují se vždy stejné uzly v jiných časových snímcích.

2.6.2 Editace skupinového pohybu

Pokud byl graf zkonstruován ze skupinového pohybu, uživatel může měnit průběh tohoto pohybu přetahováním nebo fixováním jedinců v různých časových snímcích.

Pokud je jedinec posunut, tento systém přepočítává všechny uzly ve všech časových snímcích tak, aby došlo k co nejmenší změně původního tvaru skupinového grafu.

Je zavedena metrika zkreslení a ta je použita při optimalizační metodě nejmenších čtverců pro získání řešení, které najde optimální pozice pro všechny vrcholy skupinového grafu s co nejmenším zkreslením. Tato metoda ale neřeší jednotné zvětšení nebo zmenšení lokálních hran celého grafu. Proto jsou v druhé fázi porovnávány velikosti všech trojúhelníků formačních grafů a vyrovnávají se původní velikosti. Tento proces je vidět na obrázku 2.10. Na obrázku vlevo je vidět animace pěti chodců pohybujících se ve dvou řadách. Animace je upravena tažením žlutého jedince. Na obrázku uprostřed je vidět chyba jednotného zkreslení a na obrázku vpravo je vidět výsledná animace po odstranění jednotného zkreslení v druhé fázi.



Obrázek 2.10 Ukázka problému jednotného zvětšení grafu a jeho vyřešení [Lee08]

Metrika zkreslení obsahuje tři typy třívrcholových funkcí: prostorovou, časovou a prostorově-časovou. Tyto funkce popisují relativní pozici každého bodu z pohledu jeho sousedních vrcholů ve skupinovém grafu. Prostorová funkce obsahuje tři sousední vrcholy z jednoho snímku. Časová funkce obsahuje stejný vrchol ze tří snímků (předchozí, současný, příští). Prostorově-časová obsahuje stejný vrchol z předchozího snímku a sousedící ze současného snímku. Ve fázi srovnávání velikostí se řeší pouze prostorové funkce, protože u časových a prostorově-časových funkcí by to způsobovalo nepřirozené jevy.

2.6.3 Spojování skupinového pohybu a post-processing

Spojování dvou grafů vyžaduje tři kroky. Nejprve se musí porovnat, které dva vrcholy budou propojené. Poté je nutné oba pohybové grafy zarovnat přímou transformací a nakonec vyhladit. První krok je zde řešen pomocí metody párování grafu, kdy se vždy najdou kandidáti na páry a pak se vhodnou volbou párů algoritmus snaží

minimalizovat součet vzdáleností párů. Po nalezení párů se zarovnají grafy pomocí translací a rotací jednotlivých vrcholů. Pro vyhlazení se používá také třívrcholových funkcí, protože lineární vyhlazení nezachovává formaci. Nakonec se vytvoří graf postupem popsaným v minulé kapitole.

Fáze posprocesingu se používá pro předcházení kolizím a převzorkování, aby bylo použitím spline křivek dosaženo plynulých trajektorií.

Předcházení kolizím je zde řešeno pomocí aproximace trajektorií. Pokud je nějaká trajektorie ve vzdálenosti menší, než je prahová, odsuneme nejbližší dva vrcholy grafu o 10% prahové vzdálenosti a test se opakuje. Pokud leží v trajektorii překážka, posunuje se vrchol s největší penetrací k nejbližší hranici překážky. Tento proces se opakuje, dokud nejsou všechny kolize vyřešené.

2.7 Dense Crowds

Model *Dense Crowds* **[Nar09]** je zaměřený na simulaci velkých hustých davů. Je zde použita duplicitní reprezentace chodců jako agentů, a jako prvků kontinuálního systému. Také je předložen nový omezující parametr *unilateral incompressibility constraint* (UIC - jednostranné omezení nestlačitelnosti), pro modelování chování hustých davů a urychlení řešení kolizí.

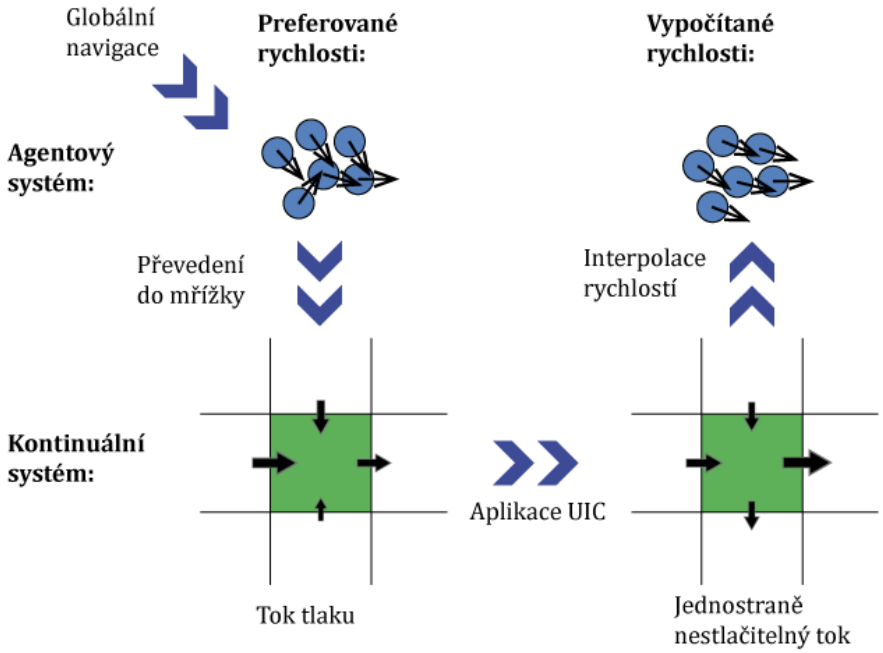
UIC specifikuje prahovou hustotu davu, která se už nemůže zvětšit. Vychází se z myšlenky, že žádný agent se nepřiblíží k jinému agentu blíže než na minimální vzdálenost. Pomocí této minimální vzdálenosti se pak jednoznačně definuje UIC. UIC je jinými slovy maximální povolená hustota chodců.

Protože hustota je do jisté míry i funkcí rychlosti (v následujícím kroku), je třeba zajistit, aby nově vypočítané rychlosti chodců neporušily podmínku UIC. Proto je zaveden pojem tlaku, který je nenulový pouze tehdy, pokud je aktuální hustota rovná UIC. Tento tlak nutí chodce, aby opustili místo s vysokou hustotou.

Tento model neobsahuje globální navigaci a je třeba, aby byl spojen s modelem pro globální navigaci, který dodá agentům jejich preferované rychlosti.

Po zjištění preferovaných rychlostí agentů se musí převést informace o stavu agentů ve scéně do kontinuální mřížky. Podle počtu a pozice agentů je zapsána hodnota hustoty chodců do centra pole mřížky i do jeho hranic. Rychlosti jsou zapsány po složkách vždy do příslušné hranice pole. Potom dochází k přepočítání hodnot rychlosti, aby byla v příštím kroku dodržena hodnota UIC.

Pro pohyb agenta je v oblastech hustého davu použita přímo interpolovaná rychlost z mřížky. Nicméně v oblastech s nižší hustotou davu by to způsobilo zamezení pohybu agenta ve směru preferované rychlosti. Proto se použije rychlost z mřížky interpolovaná preferovanou rychlostí agenta. Grafická reprezentace algoritmu je vidět na obrázku 2.11. V každém kroku simulace, každý agent získá preferovanou rychlost z globálního plánování. Pak jsou agenti převedeni do kontinuální mřížky, aby byl získán tok tlaku. Poté je tento tok upraven, aby platily podmínky UIC, a rychlost toku je interpolována s preferovanou rychlostí agentů pro získání nové rychlosti agentů.



Obrázek 2.11 Grafická reprezentace algoritmu

Protože UIC přímo nezaručuje bezkolizní pozice agentů, ve finální fázi se iteruje přes všechny páry simulace a případné kolize se řeší odsunutím agentů.

Pokud jsou v mapě přítomné překážky, počítá se s nimi v kontinuální mřížce. Globální navigační algoritmus by měl vrátit takové preferované rychlosti, které umožní obejít statické překážky, ale chodec může být silou toku davu donucen ke kolizi se statickou překážkou. Proto se zavádí hodnota volného místa v poli mřížky. Pokud je tedy v poli mřížky překážka, hodnota volného místa se sníží a s ní i poměrově UIC.

3 Detailní popis navrženého řešení

V této kapitole je nejprve detailně popsán princip překážek rychlosti a algoritmus *ClearPath*. Poté je vysvětleno navržené řešení na něm založené.

Důvodů pro vybrání algoritmu *ClearPath* jako základu řešení, bylo více. V práci Ing. Tomáše Vomáčky bylo již naimplementováno řešení navigace chodců pomocí kontinuálního systému, proto bylo žádoucí vybrat řešení založené na agentovém systému. Tím se ukazují modely *ContinuumCrowds* a *DenseCrowds* jako nevhodné. V této diplomové práci má být řešena pouze lokální navigace, a model *PLEdestrians* řeší především globální navigaci. Model *Group Motion Editing* je nevhodný, protože je zaměřený na poněkud jinou problematiku.

3.1 Předpoklady a notace

Předpokládá se, že scéna obsahuje heterogenní chodce označované jako agenty se statickými i dynamickými překážkami. Chování každého agenta je dáno vnějšími a vnitřními parametry a je vypočítáváno pro každého agenta zvlášť. Celá simulace probíhá v diskrétních časových krocích a stav každého agenta je aktualizován v každém kroku (pozice a rychlost). Pokud T je čas simulace a ΔT je časový interval, cílem je vypočítat novou rychlost pro každého agenta tak, aby nedošlo k žádné kolizi během časového intervalu $[T, T + \Delta T]$. Po vypočítání nové rychlosti je agent posunut na novou pozici danou vynásobením rychlosti a časového intervalu a původní pozicí agenta. Tato rychlost je přepočítána v každém kroku pro každého agenta.

Je předpokladem, že agenti se pohybují ve 2D prostoru, i když tento přístup lze rozšířit do 3D prostoru. V každém časovém okamžiku má agent informaci o pozici a rychlosti všech agentů ve své blízkosti. Každý agent je reprezentován kruhem nebo konvexním polygonem na ploše. Pokud by tvar agenta nebyl konvexní, je použita konvexní obálka. Přestože zde je dále popisována verze pracující pouze s kruhovými agenty, výsledný algoritmus je rozšiřitelný i pro další konvexní tvary. Pro daného agenta A je použito $\mathbf{p}_A$, r_A a $\mathbf{v}_A$ jako značení pozice, poloměru a rychlosti z minulého kroku simulace a $\mathbf{v}_A^{\text{nová}}$ značí rychlost počítanou v aktuálním kroku simulace. Je očekáváno, že preferovaná rychlost agenta $\mathbf{v}_A^{\text{pref}}$ je dodána globálním navigačním algoritmem. Jako $q^{\perp}$ je označena přímka kolmá na přímku q.

Konfigurační prostor (*configuration space*) agenta A je prostor, kde jsou všechny rychlosti a polohy zadané tak, jako by agent A ležel v počátku souřadnicové soustavy. Rychlosti agentů se nazývají absolutní, pokud se tento agent nepohybuje. Pokud se již agent pohybuje, rychlosti ostatních agentu se nazývají relativní, pokud již s tímto pohybem počítají.

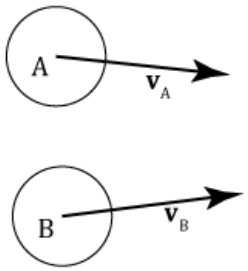
Pro použití v dalším textu definujeme Minkowského sumu **[Tom12]**. Nechť A a B jsou množiny bodů. Minkowského sumou $A \oplus B$ množin A a B s rozumí:

$$A \oplus B = \bigcup_{b \in B} A^b$$

kde množina $A^b = \{\mathbf{a} + \mathbf{b} \mid \mathbf{a} \in A\} = A + \mathbf{b}$ je množina A posunutá o vektor $\mathbf{b}$.

3.2 Překážky rychlosti [Fio98]

Nechť jsou dáni agenti A a B v čase t_0, s rychlostmi $\mathbf{v}_A$ a $\mathbf{v}_B$, jako je to na obrázku 3.1. Agent B je pro agenta A pohybující se překážkou. Stejně je i pro agenta B agent A pohybující se překážkou, ale dále se uvažuje vždy první možnost. Pro spočítání překážky rychlosti potřebujeme přenést agenta B do konfiguračního prostoru agenta A, a zároveň agenta A zredukujeme na bod $\widehat{A}$ a agenta B zvětšíme o poloměr r_A na $\widehat{B}$.



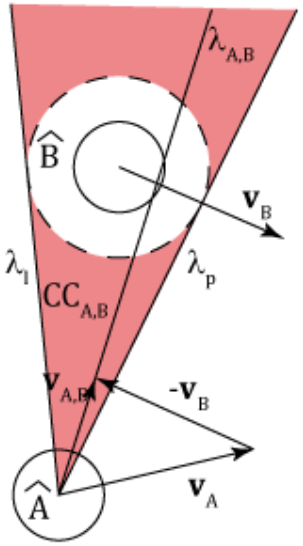
Obrázek 3.1 Dva pohybující se agenti

Kolizní kužel (*Collision Cone*) $CC_{A,B}$ je definován jako množina kolizních relativních rychlostí mezi agenty A a B:

$$CC_{A,B} = \{\mathbf{v}_{A,B} \mid \lambda_{A,B} \cap \widehat{B} \neq \emptyset\}$$

kde $\mathbf{v}_{A,B}$ je relativní rychlost A vzhledem k B, $\mathbf{v}_{A,B}=\mathbf{v}_A - \mathbf{v}_B$, a $\lambda_{A,B}$ je přímka vektoru $\mathbf{v}_{A,B}$.

Tento kužel je plocha s vrcholem v $\widehat{A}$, ohraničená dvěma tangentami λ_l a λ_p z $\widehat{A}$ do $\widehat{B}$ tak, jak je to na obrázku 3.2. Kterákoliv relativní rychlost ležící mezi tangentami λ_l a λ_p, způsobí kolizi mezi agenty A a B. Jakákoliv relativní rychlost, která bude ležet mimo $CC_{A,B}$, je garantována jako bezkolizní, pokud agent B zachová současnou rychlost.

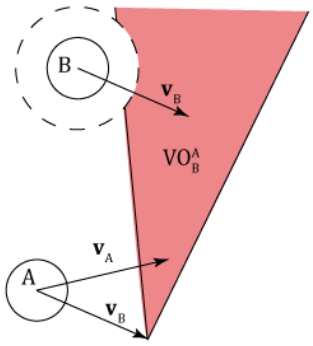


Obrázek 3.2 Kolizní kužel

Kolizní kužel je specifický pro pár agent / překážka. Při uvažování více překážek je výhodné stanovit ekvivalentní podmínku pro absolutní rychlosti agenta A. Toho je jednoduše dosaženo přičtením rychlosti agenta B ke všem rychlostem v množině $CC_{A,B}$ nebo ekvivalentně posunutím kolizního kuželu $CC_{A,B}$ o $\mathbf{v}_B$, výsledek je vidět na obrázku 3.3. Překážka rychlosti je pak definována jako:

$$VO_B^A = CC_{A,B} \oplus v_B$$

kde $\oplus$ je operátor Minkowského vektorové sumy.



Obrázek 3.3 Překážka rychlosti agenta B agentu A

Kolizní překážka rozděluje absolutní rychlosti agenta A na bezkolizní a kolizní rychlosti. Vybrání rychlosti $\mathbf{v}_A^{nová}$ mimo překážku rychlosti zajistí předejití kolizi s B, tedy:

$$A(t) \cap B(t) = \emptyset \text{ když } \mathbf{v}_A(t) \notin VO_B^A(t)$$

kde t je čas simulace.

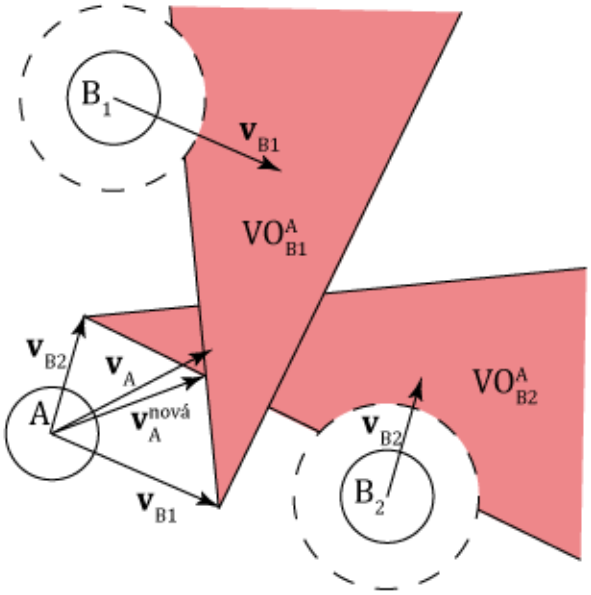
Rychlosti, které budou na hranicích překážky rychlosti, vyústí v tečné vyhnutí agenta A a agenta B. Je dobré si uvědomit, že kolizní překážka statického agenta je identická k relativnímu koliznímu kuželu, protože $\mathbf{v}_B = 0$.

Pro předcházení kolizím s více překážkami je předpokládáno sjednocení překážek rychlosti od jednotlivých agentů:

$$VO^A = \cup_{i=1}^{m} VO_{Bi}^A$$

kde m je počet překážek a VO_{Bi}^A je překážka rychlosti od agenta B_i.

Bezkolizní rychlosti jsou vybírány mimo všechny kolizní překážky. Tento příklad je vidět na obrázku 3.4, kde se agent A vyhýbá agentům B_1 a B_2. Nová rychlost agenta A $\mathbf{v}_A^{nová}$ je vypočítána tak, aby neležela v překážkách rychlosti VO_{B1}^A a VO_{B2}^A od agentů B_1 a B_2.



Obrázek 3.4 Překážky rychlosti od agentů B_1 a B_2

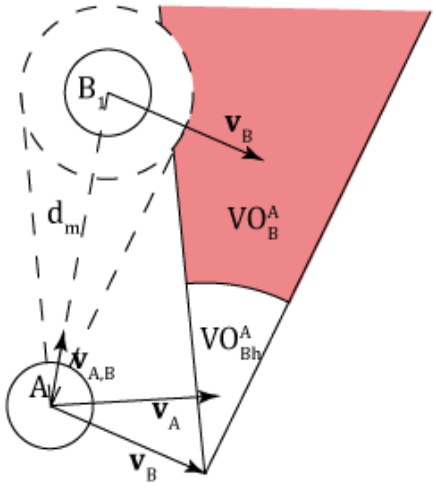
Pokud je v simulaci velké množství překážek, může být výhodné seřadit je tak, aby ty s dřívější kolizí předcházely těm s pozdější kolizí. Protože překážky rychlosti

používají lineární aproximaci pohybu překážky (překážka se pohybuje konstantní rychlostí), jejich použití pro předpověď kolizí s dlouhým časem do kolize nemusí být správná, pokud se překážka nepohybuje po přímce. Kolize mezi agentem a překážkou hrozí, pokud nastane v čase $t < T_h$, kde T_h je vhodný časový horizont, vybraný na základě dynamiky systému, trajektorií překážek a času simulačního kroku.

Pro zahrnutí pouze agentů, u kterých hrozí kolize v blízkém časovém horizontu, je od množiny překážky rychlosti odečtena množina rychlostí VO^A_{Bh} definovaná jako:

$$VO^A_{Bh} = \left\{ \mathbf{v}_A \middle| \mathbf{v}_A \in VO, \|\mathbf{v}_{A,B}\| \leq \frac{d_m}{T_h} \right\}$$

kde d_m je nejkratší vzdálenost mezi agentem a překážkou, T_h je vhodný časový horizont, $\mathbf{v}_A$ je současná rychlost agenta A a $\mathbf{v}_{A,B}$ je relativní rychlost agentů A a B. Množina VO^A_{Bh} je vidět na obrázku 3.5 a reprezentuje rychlosti, které mohou zavinit kolizi v časovém horizontu T_h.



Obrázek 3.5 Překážka rychlosti pro krátký časový horizont

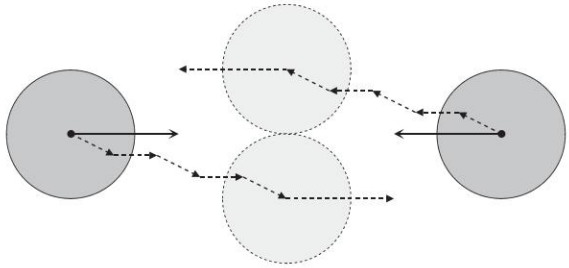
3.3 Oscilace a vzájemné překážky rychlosti

Koncept překážek rychlosti může být použitý pro navigaci více agentů, pokud každý agent bere ostatní agenty jako pohybující se překážky a vždy vybere svou rychlost tak, aby neležela v žádné překážce rychlosti jakéhokoliv jiného agenta. Tento přístup bohužel vede k oscilacím.

Oscilace je vysvětlena na následující situaci. Dva agenti A a B se pohybují rychlostmi $\mathbf{v}_A$ a $\mathbf{v}_B$ takovými, že $\mathbf{v}_A \in VO^A_B(\mathbf{v}_B)$ a $\mathbf{v}_B \in VO^B_A(\mathbf{v}_A)$. Pokud pokračují

stávajícími rychlostmi, dojde ke kolizi. Výsledkem bude, že agent A upraví svou rychlost na $\mathbf{v}_A^{\text{nová}}$ takovou, že bude mimo překážku rychlosti agenta B ($\mathbf{v}_A^{\text{nová}} \notin \text{VO}_B^A(\mathbf{v}_B)$). Zároveň agent B upraví svou rychlost na $\mathbf{v}_B^{\text{nová}}$, která je mimo překážku rychlosti agenta A ($\mathbf{v}_B^{\text{nová}} \notin \text{VO}_A^B(\mathbf{v}_A)$).

V dalším kroku simulace jsou staré rychlosti $\mathbf{v}_A$ a $\mathbf{v}_B$ mimo kolizní překážky agentů B a A ($\mathbf{v}_A \notin \text{VO}_B^A(\mathbf{v}_B^{\text{nová}})$ a $\mathbf{v}_B \notin \text{VO}_A^B(\mathbf{v}_A^{\text{nová}})$). Pokud agenti budou preferovat staré rychlosti, které je mohou vést přímo do cíle, vyberou tyto rychlosti znovu. V dalším kroku je tyto rychlosti povedou opět do kolize a agenti pravděpodobně vyberou znovu $\mathbf{v}_A^{\text{nová}}$ a $\mathbf{v}_B^{\text{nová}}$ a takto znovu a znovu. Takže agenti budou oscilovat mezi těmito dvěma rychlostmi, a to i v případě, že vyberou stejnou stranu pro vzájemné obejití. Celý příklad oscilujícího vyhnutí se je vidět na obrázku 3.6. Agenti v při vzájemném obejití vybírají v každém kroku simulace střídavě preferovanou rychlost a rychlost pro obejití druhého agenta. Jejich cesta má poté oscilující charakter.



Obrázek 3.6 Oscilující agenti při řešení kolize [Ber08]

Pro předcházení těmto oscilačním problémům lze použít vzájemné překážky rychlosti **[Ber08]** (*reciprocal velocity obstacles* – RVO). Je to jednoduchý přístup, kterým je docíleno bezpečné a plynulé navigace mezi agenty.

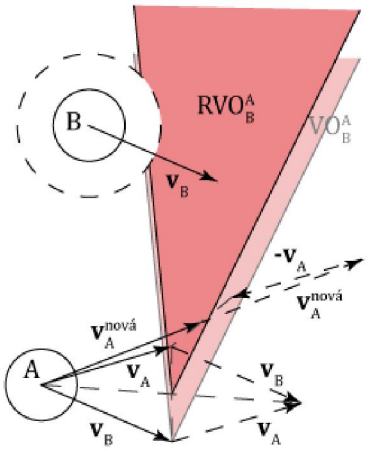
Základní myšlenka je velmi jednoduchá: místo aby agent vybral novou rychlost mimo překážku rychlosti jiného agenta, vybere rychlost, která je průměrem jeho stávající rychlosti a rychlosti, která leží mimo překážku rychlosti od druhého agenta.

Tento princip přístupu je definován jako vzájemná překážka rychlosti a jeho matematická definice je následující:

$$\text{RVO}_B^A(\mathbf{v}_B, \mathbf{v}_A) = \left\{\mathbf{v}_A^{\text{nová}} \mid 2\mathbf{v}_A^{\text{nová}} - \mathbf{v}_A \in \text{VO}_B^A(\mathbf{v}_B)\right\}$$

Vzájemná překážka rychlosti $\text{RVO}_B^A(\mathbf{v}_B, \mathbf{v}_A)$ od agenta B k agentu A obsahuje všechny rychlosti pro agenta A, které jsou průměrem současné rychlosti a rychlosti uvnitř

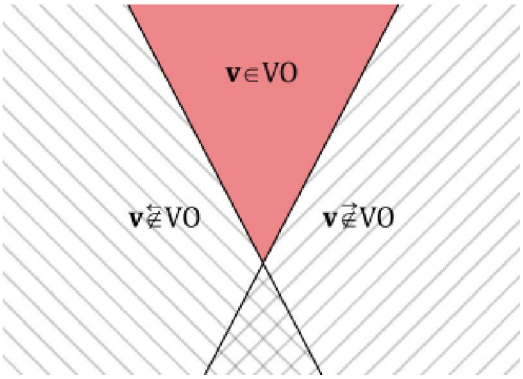
překážky rychlosti $VO_B^A(\mathbf{v}_B)$ agenta B. Toto lze geometricky vyjádřit jako překážku rychlosti $VO_B^A(\mathbf{v}_B)$ posunutou tak, že její vrchol leží na $\frac{\mathbf{v}_B+\mathbf{v}_A}{2}$. Vzájemná překážka rychlosti je vidět na obrázku 3.7.



Obrázek 3.7 Vzájemná překážka rychlosti od agenta B k agentu A

Vzájemné překážky rychlosti garantují neoscilující a bezkolizní cestu pro každého agenta. Nechť $\mathbf{v}_A$ je současná rychlost agenta A a $\mathbf{v}_B$ je současná rychlost agenta B a oba agenti A a B vyberou svou novou rychlost ($\mathbf{v}_A^{nová}$ a $\mathbf{v}_B^{nová}$) mimo vzájemnou kolizní překážku toho druhého. Taková cesta bude pak bezkolizní, pokud agenti k míjení se vyberou stejnou stranu vzájemné kolizní překážky. Že toto platí, ukazuje následující teorém (znak ´$\vec{\notin}$´, jehož význam lze vidět na obrázku 3.8, značí obcházení agentů vpravo, ale rovnici by bylo možné formulovat i pro obcházení vlevo se znakem ´$\overleftarrow{\notin}$´) :

$$\mathbf{v}_A^{nová} \vec{\notin} RVO_B^A(\mathbf{v}_B, \mathbf{v}_A) \wedge \mathbf{v}_B^{nová} \vec{\notin} RVO_A^B(\mathbf{v}_A, \mathbf{v}_B) \Rightarrow \mathbf{v}_A^{nová} \vec{\notin} VO_B^A(\mathbf{v}_B) \wedge \mathbf{v}_B^{nová} \vec{\notin} VO_A^B(\mathbf{v}_A)$$



Obrázek 3.8 Oblasti překážky rychlosti

Lze také garantovat, že oba agenti vyberou stejnou stranu pro vzájemné vyhnutí se, pokud každý z nich vybere takovou rychlost mimo vzájemnou kolizní překážku druhého agenta, která je nejblíže jeho původní rychlosti. Toto je dokázáno následujícími fakty:

1. Pokud pro agenta A je rychlost $\mathbf{v}_A^{\text{nová}} + \mathbf{u}$ nejbližší rychlost k rychlosti $\mathbf{v}_A$ mimo vzájemnou překážku rychlosti od agenta B, pak pro agenta B, $\mathbf{v}_B^{\text{nová}} - \mathbf{u}$ je rychlost nejbližší rychlosti $\mathbf{v}_B$ mimo vzájemnou překážku rychlosti od agenta A.
2. Pokud pro agenta A tato nejbližší rychlost je na pravé (levé) straně vzájemné překážky rychlosti od agenta B, pak nejbližší rychlost agenta B je také na pravé (levé) straně vzájemné překážky rychlosti od agenta A.

Obě tato fakta vyplývají z následující podmínky (znak ´∉´ může být ve všech výskytech nahrazen ´$\vec{\notin}$´, ´$\overleftarrow{\notin}$´ nebo ´∈´):

$$\mathbf{v}_A + \mathbf{u} \notin \text{RVO}_B^A(\mathbf{v}_B, \mathbf{v}_A) <=> \mathbf{v}_B - \mathbf{u} \notin \text{RVO}_A^B(\mathbf{v}_A, \mathbf{v}_B)$$

Tato podmínka tvrdí, že pokud k rychlosti $\mathbf{v}_A$ agenta A je přičten vektor $\mathbf{u}$ a výsledný vektor neleží ve vzájemné překážce rychlosti agenta A k agentu B $\text{RVO}_B^A(\mathbf{v}_B, \mathbf{v}_A)$, potom odečtením stejného vektoru $\mathbf{u}$ od rychlosti $\mathbf{v}_B$ vyjde vektor, který neleží ve vzájemné překážce rychlosti agenta B k agentu A $\text{RVO}_A^B(\mathbf{v}_A, \mathbf{v}_B)$.

Vybrání nejbližší rychlosti mimo vzájemnou překážku rychlosti druhého agenta také garantuje neoscilující vyhnutí se agentů. Toto je zajištěno následující podmínkou:

$$\mathbf{v}_A \in \text{RVO}_B^A(\mathbf{v}_B, \mathbf{v}_A) <=> \mathbf{v}_A \in \text{RVO}_B^A(\mathbf{v}_B - \mathbf{u},\ \mathbf{v}_A + \mathbf{u})$$

Tato podmínka tvrdí, že pokud současná rychlost $\mathbf{v}_A$ agenta A leží mimo vzájemnou překážku rychlosti agenta A k agentu B $\text{RVO}_B^A(\mathbf{v}_B, \mathbf{v}_A)$ a rychlosti $\mathbf{v}_A + \mathbf{u}$ a $\mathbf{v}_B - \mathbf{u}$ jsou nové rychlosti agentů A a B, pak současná rychlost $\mathbf{v}_A$ agenta A leží také mimo vzájemnou překážku rychlosti agenta A k agentu B spočítanou s novými rychlostmi $\text{RVO}_B^A(\mathbf{v}_B - \mathbf{u},\ \mathbf{v}_A + \mathbf{u})$.

Vybráním nejbližší rychlosti mimo vzájemnou překážku rychlosti pro oba agenty A a B vzájemná překážka rychlosti zůstane na stejné pozici. A nové rychlosti jsou stále nejblíže původním preferovaným rychlostem. Výsledkem je, že nebude docházet k oscilacím.

3.4 ClearPath a FVO

Autoři *ClearPath* používají jako základ svého přístupu k problematice vzájemné překážky rychlosti, ale protože vzájemné překážky rychlosti používají celý kužel, může být tento přístup velmi restriktivní ve scénách s vysokou hustotou davu. Proto je tento přístup ještě rozvinut a byl představen nový typ překážky rychlosti – FVO (tato zkratka není v původním článku vysvětlena).

FVO počítá pouze s omezeným časovým intervalem ΔT, po který zaručuje bezkolizní cestu. Tento ořez kuželu je podobný práci v původních překážkách rychlosti **[Fio98]**, ale zde je dále rozšířen z jednoho agenta pohybujícího se mezi překážkami, pro zvládnutí více agentů pohybujících se současně.

Ještě je nutné nadefinovat dva operátory, kterých je využito pro definování FVO:

- $\lambda(\mathbf{p}, \mathbf{v}) = \{\mathbf{p} + t\mathbf{v} \mid t \geq 0\}$ – definuje paprsek začínající v $\mathbf{p}$ a směřující směrem $\mathbf{v}$
- $\phi(\mathbf{v}, \mathbf{p}, \mu) = \{ (\mathbf{v} - \mathbf{p}) * \mu\}$ – definuje vzdálenost bodu $\mathbf{v}$ od $\mathbf{p}$ po přímce μ

Originální překážky rychlosti nebo vzájemné překážky rychlosti jsou definovány pouze dvěma omezeními (hrany kužele). FVO je definováno čtyřmi omezeními a celá je vidět na obrázku 3.9. První dvě omezení jsou stejná jako u vzájemných překážek rychlosti:

$$\mathrm{FVO}_{\mathrm{L}_\mathrm{B}}^{\mathrm{A}}\left(\mathbf{v}_\mathrm{A}^{\mathrm{nová}}\right) = \mathbf{v}_\mathrm{A}^{\mathrm{nová}} \overleftarrow{\in} \mathrm{RVO}_\mathrm{B}^\mathrm{A}(\mathbf{v}_\mathrm{B}, \mathbf{v}_\mathrm{A})$$

$$\mathrm{FVO}_{\mathrm{R}_\mathrm{B}}^{\mathrm{A}}\left(\mathbf{v}_\mathrm{A}^{\mathrm{nová}}\right) = \mathbf{v}_\mathrm{A}^{\mathrm{nová}} \overrightarrow{\in} \mathrm{RVO}_\mathrm{B}^\mathrm{A}(\mathbf{v}_\mathrm{B}, \mathbf{v}_\mathrm{A})$$

Omezení $\mathrm{FVO}_{\mathrm{L}_\mathrm{B}}^{\mathrm{A}}(\mathbf{v})$ je definováno tak, že nová rychlost $\mathbf{v}_\mathrm{A}^{\mathrm{nová}}$ agenta A leží na levé straně od vzájemné překážky rychlosti $\mathrm{RVO}_\mathrm{B}^\mathrm{A}(\mathbf{v}_\mathrm{B}, \mathbf{v}_\mathrm{A})$. Omezení $\mathrm{FVO}_{\mathrm{R}_\mathrm{B}}^{\mathrm{A}}(\mathbf{v})$ je definováno tak, že nová rychlost $\mathbf{v}_\mathrm{A}^{\mathrm{nová}}$ agenta A leží na pravé straně od vzájemné překážky rychlosti $\mathrm{RVO}_\mathrm{B}^\mathrm{A}(\mathbf{v}_\mathrm{B}, \mathbf{v}_\mathrm{A})$. Pro novou rychlost $\mathbf{v}_\mathrm{A}^{\mathrm{nová}}$ agenta A pak platí omezení $\mathrm{FVO}_{\mathrm{L}_\mathrm{B}}^{\mathrm{A}}\left(\mathbf{v}_\mathrm{A}^{\mathrm{nová}}\right)$ nebo $\mathrm{FVO}_{\mathrm{R}_\mathrm{B}}^{\mathrm{A}}\left(\mathbf{v}_\mathrm{A}^{\mathrm{nová}}\right)$.

Omezení Typ-I: Omezení plyne z faktu, že FVO garantuje bezkolizní cestu pouze v omezeném časovém intervalu ΔT. Je počítána konečná podmnožina RVO, která může vést ke kolizi v ΔT. Oříznutý kužel (na obrázku je vyšrafován) je ohraničen křivkou $\gamma_{\mathrm{AB}}(\mathbf{v}_\mathrm{A}^{\mathrm{nová}})$**.** Pro efektivnost je křivka $\gamma_{\mathrm{AB}}(\mathbf{v}_\mathrm{A}^{\mathrm{nová}})$ nahrazena jednoduchou lineární aproximací $\Gamma_{\mathrm{AB}}(\mathbf{v}_\mathrm{A}^{\mathrm{nová}})$. V praxi tato aproximace zvětší ořez kuželu velmi málo. Toto další omezení je reprezentováno jako:

$$FVO_{T_B}^{A}(\mathbf{v}_A^{nová}) = \Gamma_{AB}(\mathbf{v}_A^{nová}) = \lambda(M - \widehat{\mathbf{p}_{AB}^{\perp}}\eta, \mathbf{p}_{AB}^{\perp}), \text{ kde}$$

$$\eta = \tan\left(\sin^{-1}\frac{r_A+r_B}{|\mathbf{p}_{AB}|}\right)(|\mathbf{p}_{AB}| - (r_A + r_B)) \text{ a}$$

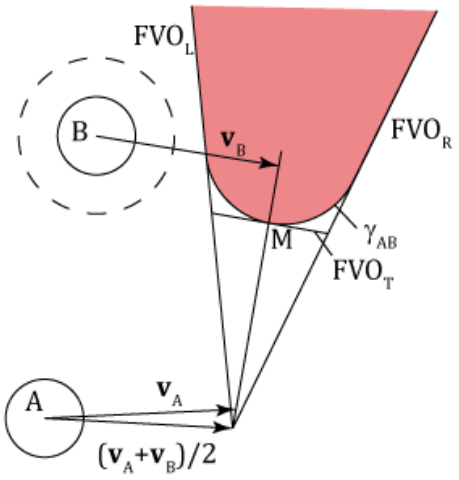
$$M = (|p_{AB}| - (r_A + r_B))\widehat{p_{AB}} + \frac{v_A+v_B}{2}$$

kde $\mathbf{p}_{AB}$ je vektor vzdálenosti mezi agenty, $\widehat{\mathbf{p}_{AB}}$ je normalizovaná verze tohoto vektoru, vektor $\mathbf{p}_{AB}^{\perp}$ je vektor kolmý na $\mathbf{p}_{AB}$, r_A a r_B jsou poloměry agentů A a B, M a η jsou použity pro mezivýpočty.

Omezení Typ-II: Omezení udává, že agent musí vybrat novou rychlost na straně FVO, která je nejblíže jeho rychlosti z minulého kroku:

$$FVO_{C_B}^{A}(v_A^{nová}) = \phi\left(v_A^{nová}, \frac{(v_A+v_B)}{2}, \phi(v_A, v_B, p_{AB}^{\perp})p_{AB}^{\perp}\right) \geq 0$$

Každé řešení splňující všechna tato omezení, formulovaná nezávisle pro každého agenta, zaručuje předcházení kolizím. *ClearPath* počítá distribuovaným způsobem novou rychlost, splňující všechna omezení a minimalizující odchylku od rychlosti $\mathbf{v}_A^{pref}$, danou globálním navigačním algoritmem.



Obrázek 3.9 ClearPath FVO

Dále je sjednocení FVO N nejbližších agentů nazýváno potenciálně kolidující region (*potentially colliding region* - PCR) a hraniční segmenty FVO všech sousedních agentů jsou nazývány mezní hrany (*boundary edges* – BE). Pokud je počítáno s N nejbližšími agenty, mezní hrany obsahují 4N hraničních segmentů – čtyři od každého

blízkého agenta. Využitím geometrického rázu problému byla odvozena následující podmínka:

Pokud $\boldsymbol{v}_A^{pref}$ je uvnitř PCR, $\boldsymbol{v}_A^{nová}$ leží na BE.

V mnoha případech existují další omezení rychlosti agenta (např. maximální rychlost nebo zrychlení). V případě, že řešení nesplňuje tato omezení, PCR je vypočítán pouze pro $N - 1$ agentů. Tato relaxace je prováděna do doby, než se nalezne řešení, které splňuje všechna omezení.

Klíčovým aspektem algoritmu *ClearPath* je mít silné podmínky pro předcházení kolizím v daném časovém intervalu. Toto je dáno následujícím teorémem.

Pokud ClearPath najde řešení pro všechny agenty, výsledné řešení bude bezkolizní.

Na základě této garance je použit následující algoritmus pro získání bezkolizní cesty pro všechny agenty.

ClearPath-1 garantované předcházení kolizím: Cílem je vypočítat bezkolizní cestu pro daný časový interval ΔT. V případě, že existuje přijatelné řešení optimalizačního algoritmu, je to bezkolizní řešení. Může se ale stát, že optimalizační algoritmus nenajde řešení, protože časový interval ΔT je moc dlouhý nebo protože omezení nové rychlosti jsou příliš restriktivní. V tomto případě je časový interval redukován $(\Delta T/2)$ a jsou znovu spočítána všechna kolizní omezení pro tento kratší interval. Tento proces redukce je opakován, dokud se nenajde přijatelné řešení.

ClearPath-2 relaxace kolizních omezení: Je možné, že algoritmus ClearPath-1 potřebuje velice malý interval, aby našel přijatelné řešení. Každý krok přístupu půlení časového intervalu vyžaduje znovu vypočítat PCR. Rovněž omezení typ-II v optimalizaci může být velice restriktivní. Proto je představen algoritmus, který uvažuje pouze omezení typ-I. Tento algoritmus tedy počítá novou rychlost $\mathbf{v}_A^{nová}$ pouze pro $3N$ omezení od každého agenta.

Při tomto přístupu je ale možné, že někteří agenti spolu budou kolidovat. V těchto případech by agent měl novou rychlost vybrat tak, aby kolizi v následném kroku vyřešil. Toto je řešeno přidáním další překážky rychlosti do existujícího PCR od ostatních sousedních agentů. Tato dodatečná překážka rychlosti je aproximací průniku dvou kruhů. První kruh je množina maximálních rychlostí dosažených v jednom časovém kroku. Druhý kruh je Minkowského rozdíl dvou agentů ($B \oplus -A$). Plocha, která leží v prvním

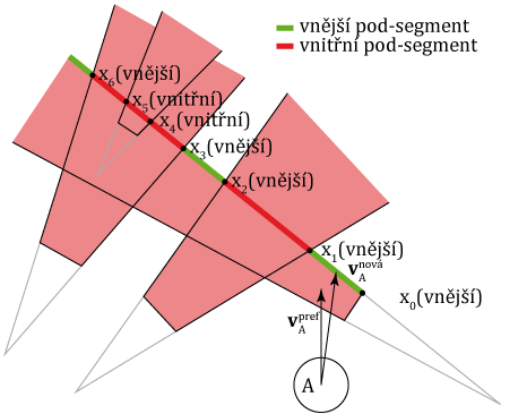
kruhu a ne v druhém, je oblast, která vyřeší kolizi během jednoho časového kroku. Plocha, která leží v obou kruzích, ale kolizi nevyřeší, je přidána do PCR (aproximovaná jako FVO). Agentovi, který je v kolizi, je nastavena preferovaná rychlost $\mathbf{v}_A^{pref}$ na nulový vektor. Agent tak vybere nejmenší možnou rychlost pro vyřešení kolize takovou, aby se předešlo oscilacím.

3.5 Implementace ClearPath

Matematická formulace vysvětlená v minulé kapitole je použitá k navržení algoritmu pro vypočítávání bezkolizní rychlosti pro každého agenta zvlášť. Pokud $\mathbf{v}_A^{pref}$ leží v PCR, $\mathbf{v}_A^{nová}$ leží na BE, proto jsou počítány průsečíky všech hraničních segmentů se všemi ostatními. Uvažujme segment X (viz obrázek 3.10). Máme k průsečíků oblasti FVO pojmenovaných jako $\mathbf{x}_1,\dots,\mathbf{x}_k$. Průsečíky jsou seřazené podle vzdálenosti od koncového bodu $\mathbf{x}_0$ vzestupně. Dále tyto průsečíky klasifikujeme jako vnitřní a vnější podle toho, zda jsou uvnitř nějaké FVO. Po klasifikaci bodů jsou klasifikovány segmenty mezi těmito body, zda jsou uvnitř nebo vně potenciálně kolidujícího regionu podle následujícího lemmatu:

První podsegment je označen jako vnější, pokud oba jeho konce byly klasifikovány jako vnější. Každý další podsegment je označen jako vnější, pokud oba jeho konce byly označeny jako vnější a podsegment před ním je označen jako vnitřní.

Pokud jsou například oba body $\mathbf{x}_0$ a $\mathbf{x}_1$označeny jako vnější, pak podsegment $\mathbf{x}_0\mathbf{x}_1$ je označen jako vnější. Podsegment $\mathbf{x}_1\mathbf{x}_2$ je označen jako uvnitř PCR, protože $\mathbf{x}_0\mathbf{x}_1$byl označen jako vnější. Bod nejbližší bodu $\mathbf{v}_A^{pref}$ ležící na vnějším podsegmentu BE je pak nová rychlost $\mathbf{v}_A^{nová}$.



Obrázek 3.10 Klasifikace mezních hran na vnitřní a vnější

ClearPath prochází následujícími kroky pro počítání nové rychlosti každého agenta:

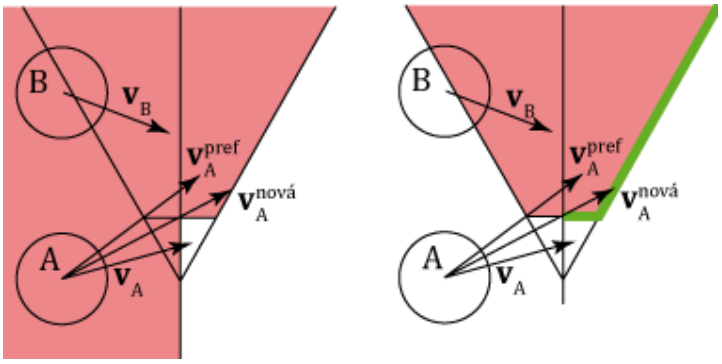
1. Pro každého agenta získá *N* nejbližších sousedních agentů z KD-strom a vypočítá FVO omezení pro mezní hrany BE.
2. Vypočítá normály segmentů v BE.
3. Vypočítá průsečíky segment s ostatními segmenty v BE.
4. Klasifikuje průsečíky jako vnější nebo vnitřní.
5. Seřadí průsečíky vzestupně, podle vzdálenosti od počátečního bodu.
6. Klasifikuje podsegmenty na každém segmentu jako vnitřní a vnější a získává nejbližší bod pro novou rychlost.
7. Pokud výsledné řešení neřeší nějaké další omezení rychlosti, odstraní se nejvzdálenější agent a algoritmus je opakován pro méně agentů.

3.6 Vlastní návrh řešení lokální navigace

Nové řešení lokální navigace vychází z konceptu *ClearPath*. Cílem úpravy je, aby výsledné řešení bylo velmi rychlé, protože výsledný algoritmus má řešit velké množství agentů. Pokud by algoritmus byl využitý v řešení Ing. Tomáše Vomáčky popsanému v kapitole 2.2.1, agenti by museli reagovat na chodce simulované kontinuálním modelem, kterých je velmi vysoký počet.

Aloritmus *ClearPath-1* chce zaručit v první řadě bezkolizní cestu. Kvůli této podmínce musí být simulační krok za určitých podmínek několikrát přepočítán. Algoritmus *ClearPath-2* odstraní nejvíce restriktivní omezení, a to že se agenti musí vždy minout na stejné straně, a tím agenty pouští do kolize, pokud vyberou každý jinou stranu minutí.

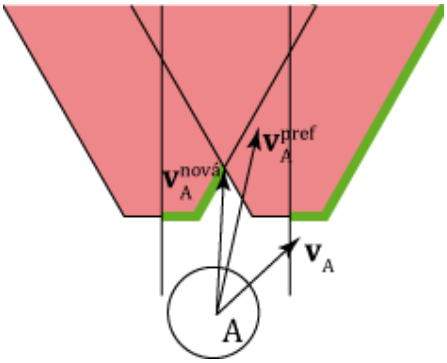
Nové řešení je navržené tak, aby se krok simulace nemusel přepočítávat, a zároveň, aby agenti vybrali stejnou stranu míjení ve většině případů. Tím že se krok nebude přepočítávat, snížíme výpočetní nároky algoritmu v místech simulace s vysokou hustotou agentů. Vybrání stejné strany míjení je docíleno tak, že segmenty FVO jsou ve fázi hledání bezkolizní cesty rozděleny na dvě části, podle strany obcházení (není rozdělena celá rovina simulace, jako je tomu v *ClearPath*) a pouze jedna část je začleněna do BE. V PCR se omezení strany nebere v úvahu. Agent při obcházení vybere pro každou překážku rychlosti část nejbližší jeho původní rychlosti. Díky tomu je možné, že pokud se vyhýbají více než dva agenti, může některý agent vybrat novou rychlost $\mathbf{v}_A^{nová}$ tak, že bude na druhé straně překážky rychlosti, než po které měl agent projít. Rozdíl proti *ClearPath* je vidět na obrázku 3.11. U aplikace omezení strany míjení na FVO novým způsobem je zelenou barvou vyznačená část BE, kterou lze použít pro obejití překážky.



(a) (b)

Obrázek 3.11 (a) FVO po aplikaci omezení strany míjení (b) FVO po omezení strany novým způsobem

Chyba, ke které může díky tomuto přístupu dojít, je vidět na obrázku 3.12. Z obrázku je vidět, že chodec měl pravou překážku v novém kroku obejít zprava, ale zvolil stranu míjení vlevo. Nicméně tyto chyby většinou nevadí, pokud agenti nejsou v bezprostřední blízkosti, protože v následujícím kroku se počítá z nové rychlosti a všechny strany minutí mohou být za určitých podmínek správně. Pokud dojde ke kolizi agentů, je řešena stejně, jako bylo popsáno v kapitole 3.4.



Obrázek 3.12 Chyba nového přístupu

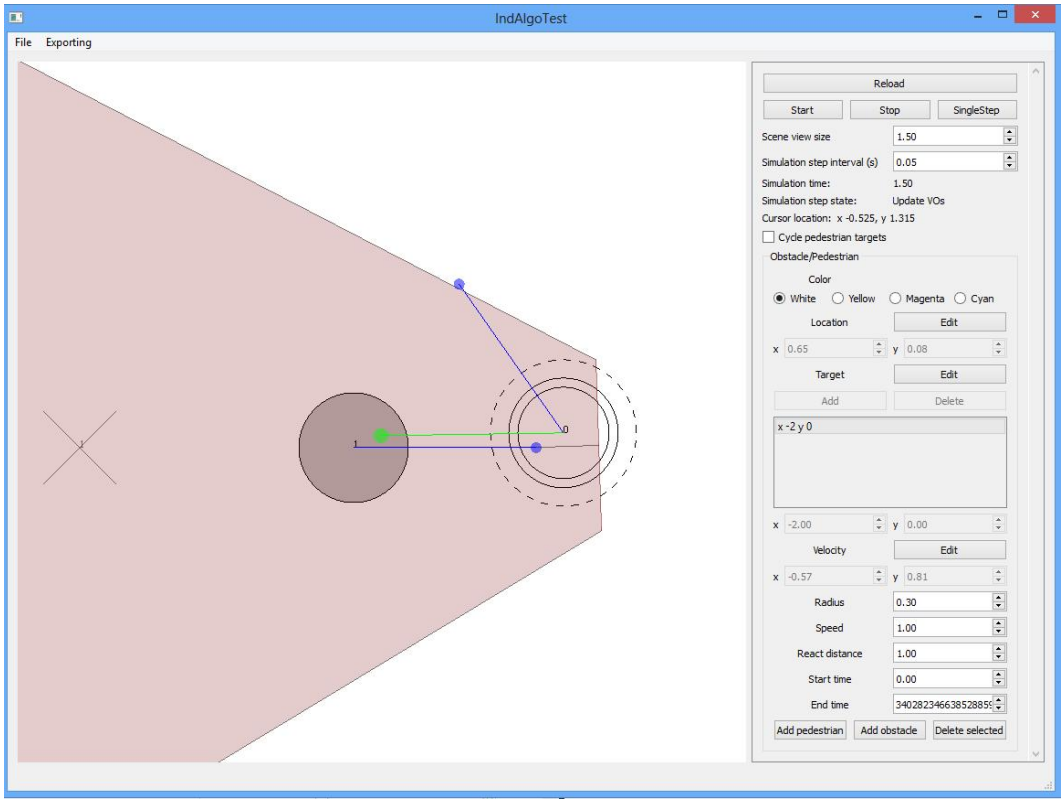
Další změnou je, že překážky rychlosti, ze kterých se bude počítat PCR, nejsou vybírány pro N nejbližších překážek, ale jsou vybrány podle vzdálenosti překážek. Důvodem je že agenti nejsou uloženi v KD-stromu jako je tomu u *ClearPath.* KD-strom nebyl implementován z časových důvodů. Agenti tedy počítají FVO pro všechny překážky do prahové (reakční) vzdálenosti. Je dobré si uvědomit, že k nastavené vzdálenosti se připočítávají poloměry agenta i překážky, aby nemohlo dojít ke kolizi z důvodu velké překážky.

Propojení s algoritmem pro globální navigaci je zpracováno tak, že každému chodci je nastavena posloupnost bodů, které musí při simulaci projít. Chodec pak v každém kroku spočítá svou preferovanou rychlost ze směru k dalšímu cíli a rychlosti chodce. Rychlost chodce lze v algoritmu měnit. Pro algoritmus není problém zvládnout dynamické vkládání a odebírání chodců ze simulace. Také lze měnit poloměr jednotlivých chodců.

Algoritmus je schopen také zvládnout interakce chodce se statickými a dynamickými překážkami ve tvaru kruhu. Jediným rozdílem oproti interakci dvou chodců je, že vrchol FVO spočítaného pro statickou nebo dynamickou překážku je posunut pouze o rychlost překážky (statická překážka má nulovou rychlost), používá se tedy přístup popsaný v kapitole 3.2.

4 Implementace

Jako praktická část diplomové práce byla napsána aplikace IndAlgoTest pro ladění a testování algoritmu popsaného v kapitole 3.6. Aplikace je napsána v jazyce C++ s využitím knihovny Qt pro napsání uživatelského rozhraní a xml komunikaci a knihovny OpenGL pro vykreslování simulované scény. Aplikace byla napsána pro operační systém Windows ve vývojovém prostředí Microsoft Visual Studio, ale protože Qt knihovna je multiplatformní, neměl by být problém přeložit aplikaci i pod operačním systémem Linux. Ukázka uživatelského rozhraní je vidět na obrázku 4.1.



Obrázek 4.1 Uživatelské rozhraní aplikace

Aplikace umožnuje uživateli následující akce:

- Spuštění a zastavení simulace
- Krokování simulace (vykonání pouze další nedělitelné časti simulačního kroku a zastavení)
- Editaci, uložení a nahrání scény simulace
- Editaci, uložení a nahrání konfigurace simulace

- Uložení sekvence obrázků ze simulace v PNG formátu
- Uložení simulace pro načtení programem pro zobrazení simulace *Animation Renderer for Crowd*

Algoritmus simulace je velice podobný algoritmu použitému v ClearPath modelu pro simulaci chodců. Nejprve jsou všem chodcům přiřazeny jejich preferované rychlosti podle směru k aktuálnímu cíli a rychlosti. Poté se pro každého chodce zjistí všechny překážky, se kterými bude chodec interagovat, a spočítají se překážky rychlosti. Pokud se chodec nemůže pohybovat preferovanou rychlostí, pomocí spočítaných překážek rychlosti se spočítá bezkolizní rychlost. Po spočítání bezkolizní rychlosti všem chodcům jsou chodci posunuti na základě časového kroku.

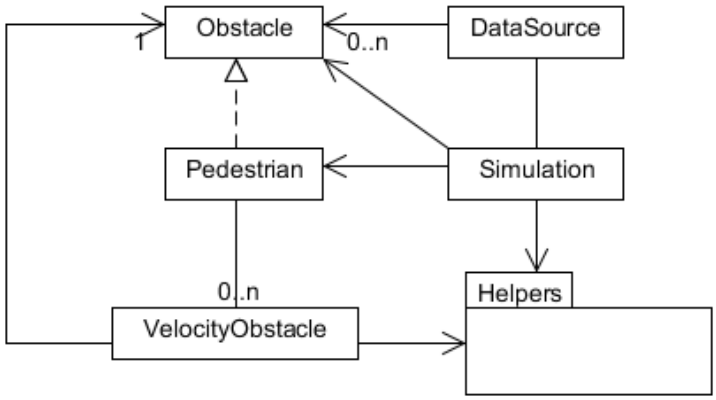
4.1 Struktura aplikace

Architektura aplikace je rozdělena na dvě části: část pro řešení simulace chodců a část uživatelského rozhraní. Část pro řešení simulace nevyužívá knihovny Qt, aby bylo možné tuto část jednoduše zakomponovat do programu SPH Fluid Simulator, a je překládána jako statická knihovna. Jako matematický základ pro počítání geometrických úloh je využito části zdrojového kódu z programu SPH Fluid Simulator. Z důvodů kompatibility je také použito třídimenzionálního vektoru na místo dvoudimenzionálního, který by byl pro potřeby algoritmu dostačující.

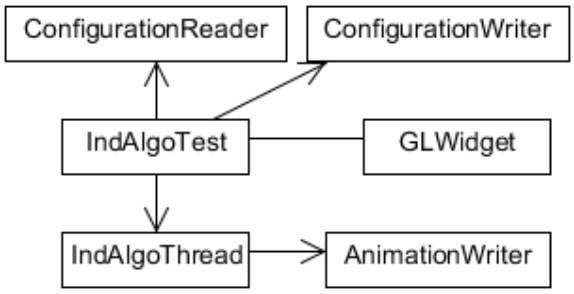
Struktura knihovny pro řešení simulace je vidět na obrázku 4.2. Třída *Simulation* poskytuje metody pro běh simulace. Třída *DataSource* je úložištěm pro všechny objekty scény, řídí jejich správu, jako je přidání a mazání objektů. Třída *Obstacle* reprezentuje základní překážku ve scéně ve tvaru kruhu. Tato překážka se může scénou pohybovat, ale žádným způsobem nereaguje na okolí, na rozdíl od třídy *Pedestrian* (rozšíření třídy *Obstacle*), která reprezentuje interaktivního chodce ve scéně. Tento chodec se pohybuje způsobem popsaným v kapitole 3.6. Třída *VelocityObstacle* reprezentuje překážku rychlosti, kterou chodec vytváří pro okolní překážky. Ve jmenném prostoru (*namespace*) *Helpers* jsou metody používané pro řešení geometrických problémů, jako je například najít průsečík dvou přímek nebo vzdálenost bodu a přímky.

Struktura aplikace využívající knihovnu pro simulaci chodců je na obrázku 4.3. Třída *IndAlgoTest* je hlavní okno aplikace a řeší logiku ovládání aplikace. Třídy *ConfigurationReader* a *ConfigurationWriter* slouží pro načítání a ukládání simulované scény a konfigurace do xml souborů. Třída *GLWidget* slouží pro vykreslování simulované

scény. Simulace běží ve vlastním vlákně spravovaném třídou *IndAlgoThread*, aby uživatelské prostředí bylo stále responzivní. Pokud je zapnuté ukládání výstupu simulace pro program *Animation Renderer for Crowd*, pak je to také prováděno vláknem simulace pomocí třídy *AnimationWriter*.



Obrázek 4.2 Struktura knihovny pro řešení simulace



Obrázek 4.3 Struktura aplikace

5 Experimenty

Všechny experimenty byly prováděny na notebooku Lenovo IdeaPad y570, který je osazený procesorem Intel Core i7 2 GHz, 8 GB RAM a grafickou kartou NVidia GForce m550. Tento počítač používá Hyper-Threading, a tím lze pro jedno vlákno využít pouze jednu osminu výkonu celého procesoru.

Detailní popis kreslené scény lze najít v uživatelském manuálu programu, který je přiložen jako příloha A. Vstupní soubory, animace a obrázky ke všem experimentům jsou přiloženy na DVD ve složce *experiments*.

U experimentů je vždy popsáno, pokud byl experiment provedený s jiným než defaultním nastavením chodců nebo překážek. Pokud je ve složce experimentu soubor konfigurace (*.iaconf) je nutné tento soubor načíst po načtení souboru scény (*.iascene), aby bylo dosaženo stejného výsledku experimentu.

V tabulce 5.1 jsou vysvětleny všechny prvky objevující se na obrázcích z experimentů.

Prvek scény	Vysvětlení
Kruhy	Kruh reprezentuje chodce nebo překážku. Překážka je vždy vyplněná šedou barvou. Chodci mohou mít barvu bílou, purpurovou, azurovou nebo žlutou. Číslo uprostřed značí index prvku.
Přerušovaná kružnice	Indikace aktivního chodce nebo překážky.
Červeně vyplněné oblasti	Překážky rychlosti od ostatních chodců k aktuálně vybranému chodci.
Šedivý křížek	Značí cíl aktivního chodce. Čísla značí indexy jednotlivých cílů.
Kružnice uvnitř některých chodců	Značí, že se chodec nemůže pohybovat svou požadovanou rychlostí a musí vypočítávat novou rychlost pomocí překážek rychlosti.
Zelené čáry	Preferovaná rychlost chodce. Konec je pro lepší viditelnost označen malým kruhem stejné barvy.
Modré čáry	Rychlost, kterou se chodec pohyboval minulý krok nebo rychlost, kterou se bude pohybovat v následující části kroku pro pohyb.

Tabulka 5.1 Vysvětlení prvků scény

5.1 Interakce malého počtu chodců

Experimenty v této kapitole detailně popisují chování malé skupiny chodců, které řídí svou lokální navigaci implementovaným algoritmem.

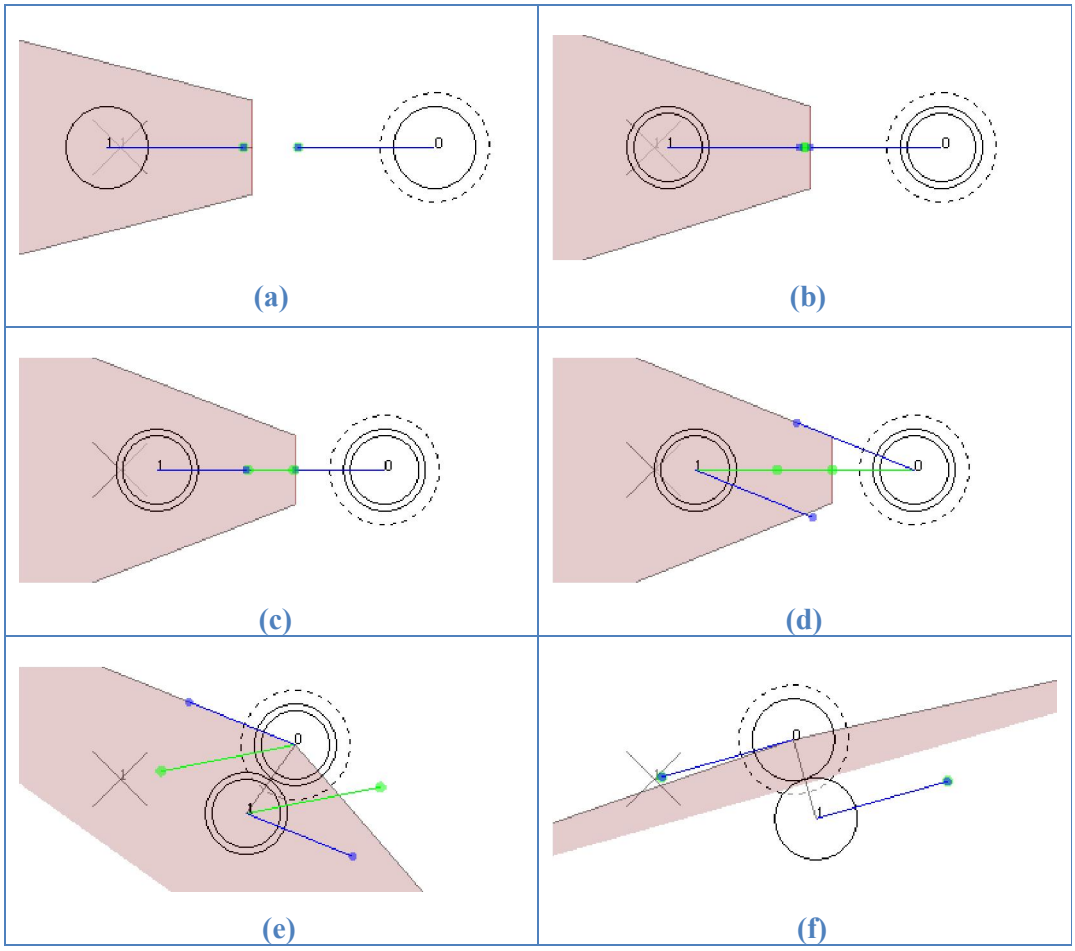
5.1.1 Interakce chodce s jiným chodcem a překážkou

První experiment na obrázcích 5.1 zobrazuje způsob vyhnutí dvou chodců jdoucích proti sobě. Oba chodci mají nastavenou dvojnásobnou reakční vzdálenost. Na obrázku (a) má chodec s indexem nula (dále jen $chodec_0$) již vypočtenou rychlostní překážku pro $chodce_1$ (dále VO[0,1]), ale nemusí přepočítávat novou rychlost, protože jeho preferovaná rychlost je mimo VO[0,1]. Na obrázku (b) již preferovaná rychlost $chodce_0$ zasahuje do VO[0,1] a proto byla jeho rychlost přepočítána. Obrázek (c) zobrazuje, že $chodec_0$ nejprve zpomalí, než začne obcházet $chodce_1$. Na obrázku (d) $chodci_0$ již vychází nová rychlost na pravou stranu VO[0,1] a začíná obcházení. Na obrázku (e) je vidět, že kužel VO[0,1] již není ořezáván, protože chodci jsou příliš blízko u sebe. Na obrázku (f) už oba chodci dokončili obcházení a mohou jít po preferovaných rychlostech. Vstupní soubory jsou v podsložce *two-pedestrians1*.

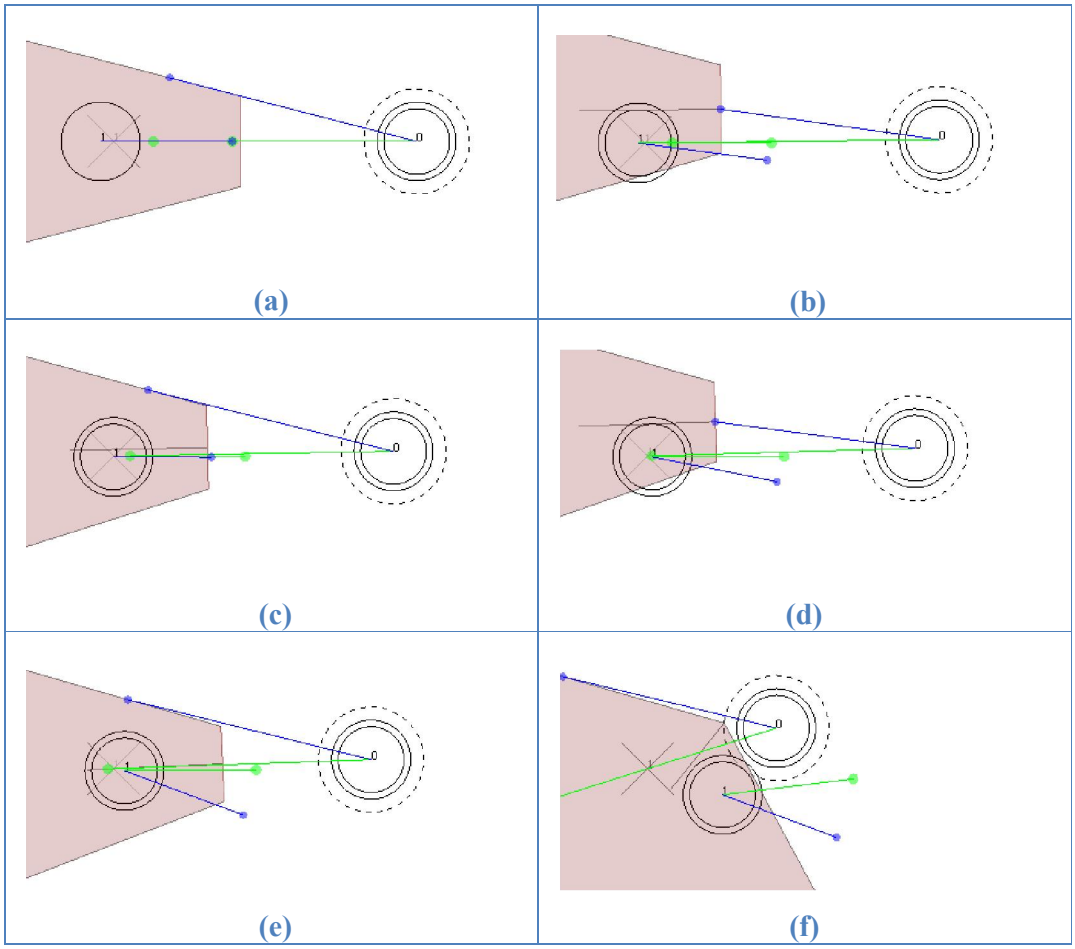
Experiment na obrázcích 5.2 zobrazuje podobnou situaci, ale $chodec_0$ se pohybuje rychlostí 2x větší než $chodec_1$. Mezi obrázky (a) až (e) je vidět kmitání rychlosti chodce. Podobné kmitání se v simulacích s více chodci objevuje často, ale není způsobené chybou popsanou v kapitole 3.3. Tento typ kmitání je způsoben ořezem vrcholu kuželu RVO. Vstupní soubory jsou v podsložce *two-pedestrians2*.

Experiment na obrázcích 5.3 zobrazuje dva chodce, kteří si vzájemně křižují trasu pod mírným úhlem. $Chodec_1$ má 1,5x větší poloměr než $chodec_1$. Minutí proběhne bez problému. Vstupní soubory jsou v podsložce *two-pedestrians3*.

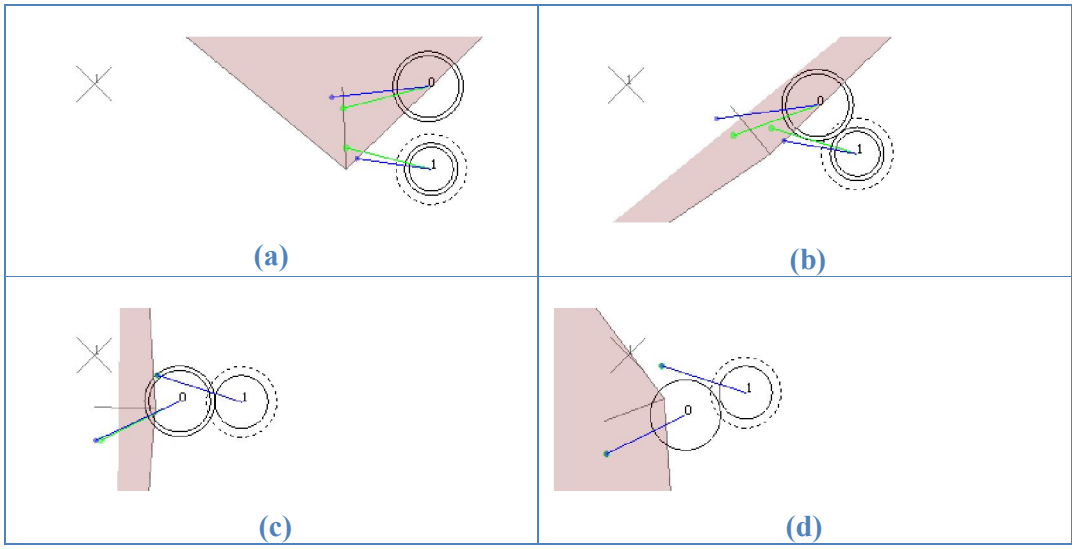
Na obrázcích 5.4 je zobrazena interakce chodce a pohybující se překážky. Protože $překážka_1$ na $chodce_0$ nijak nereaguje, musí $chodec_0$ předejít celé kolizi. Pokud se $překážka_1$ pohybuje 2x pomaleji než $chodec_0$, vyhnutí proběhne bez problému. Ale pokud se $překážka_1$ pohybuje stejnou rychlostí jako $chodec_0$, dojde ke kolizi. Tento případ lze vidět na obrázcích 5.5. Vstupní soubory obou experimentů jsou v podsložkách *pedestrian-and-obstacle1* a *pedestrian-and-obstacle2*.



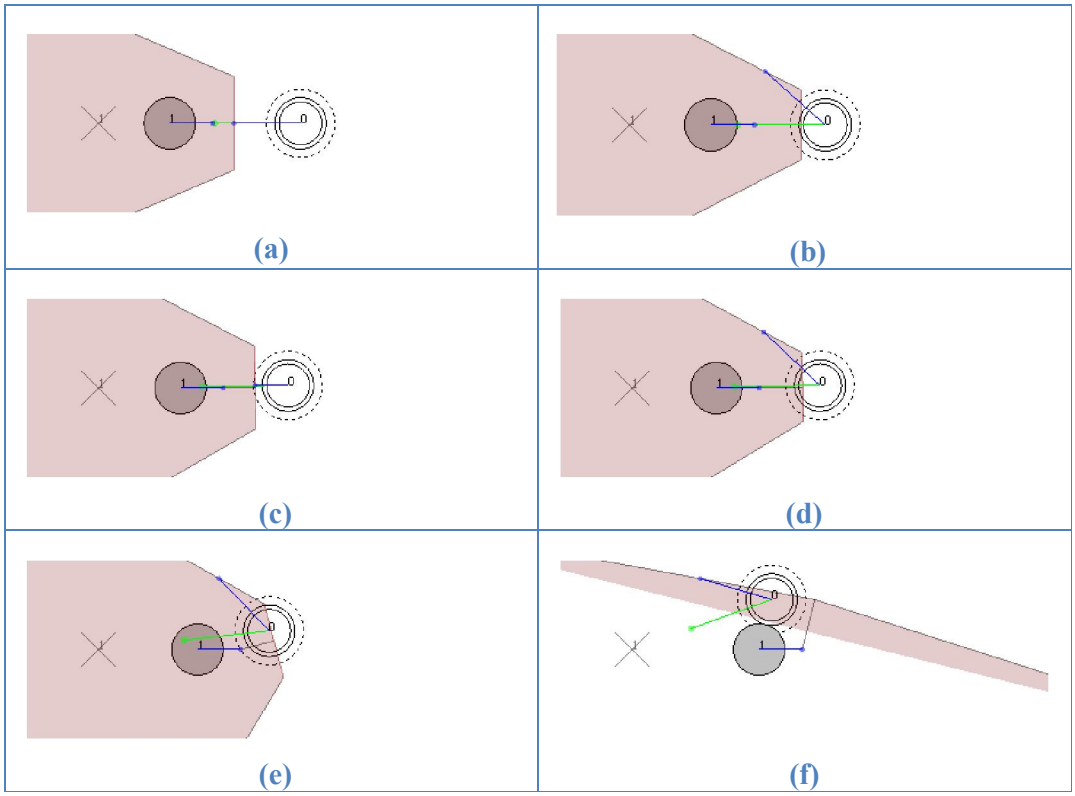
Obrázek 5.1 Dva chodci jdoucí proti sobě



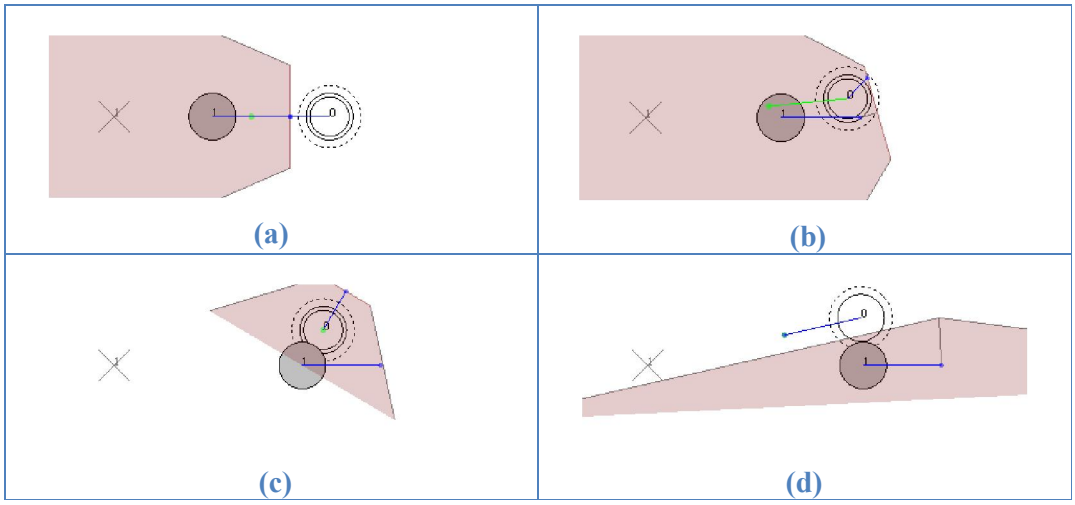
Obrázek 5.2 Dva chodci jdoucí proti sobě různou rychlostí



Obrázek 5.3 Dva chodci vzájemně křižující trasu



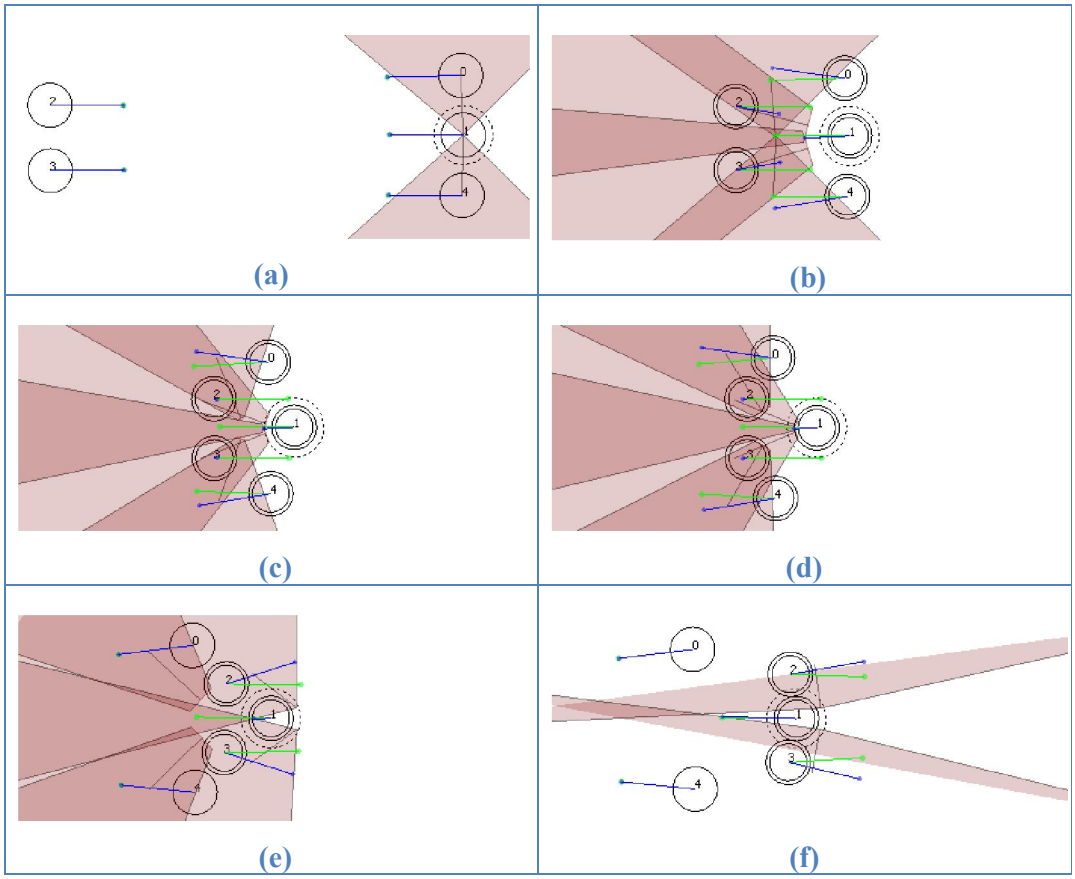
Obrázek 5.4 Chodec a pomalu se pohybující dynamická překážka



Obrázek 5.5 Chodec a rychle se pohybující dynamická překážka při kolizi

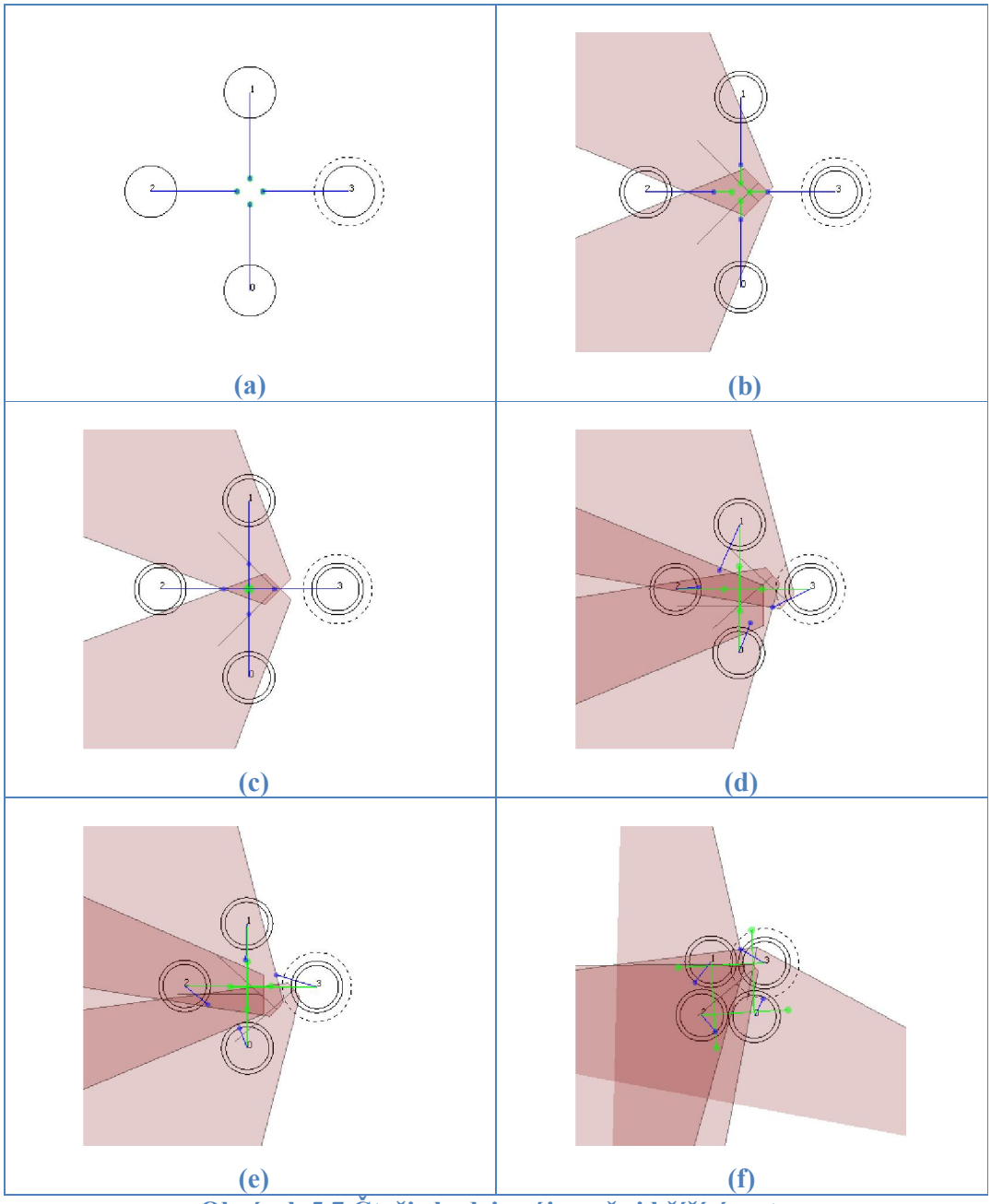
5.1.2 Více chodců v interakci

Experiment na obrázcích 5.6 zobrazuje pět chodců pohybujících se vodorovně proti sobě. Tři chodci jdou zprava a dva zleva. Na obrázcích je vidět, že nejrychleji vyřeší minutí $chodec_0$ a $chodec_4$, protože mají hodně místa pro vyhnutí. Naopak $chodec_1$ je téměř zastaven a čeká než se $chodec_2$ a $chodec_3$ dostatečně rozestoupí, aby mohl projít. Vstupní soubory jsou ve složce *five-pedestrians*.



Obrázek 5.6 Pět chodců pohybujících se vodorovně proti sobě

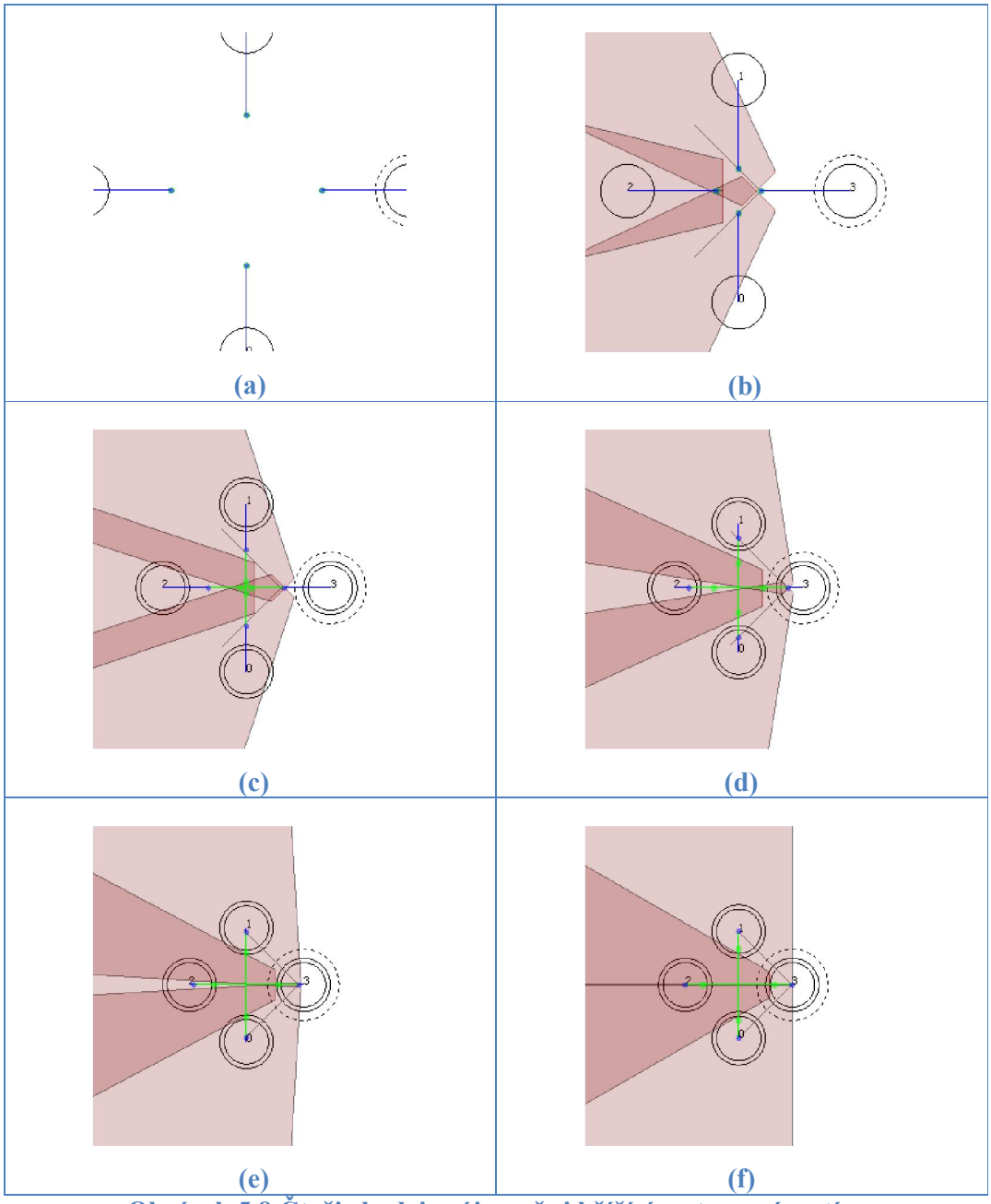
Na obrázcích 5.7 jsou zobrazeni čtyři chodci jdoucí proti sobě ze čtyř stran. Chodci se setkávají ve středu scény a vytvoří vír proti směru hodinových ručiček. Na obrázku (d) je vidět že chodec$_3$, se také pohyboval směrem opačným. V simulaci to vyústí v zakmitání. Vstupní soubory jsou ve složce *four-pedestrians1*.



Obrázek 5.7 Čtyři chodci vzájemně si křížící cesty

Na obrázcích 5.8 je zobrazena podobná situace, pouze nyní je reakční vzdálenost všech chodců nastavena na dvojnásobek. Chodci vytvoří překážky rychlosti dříve a nedojde k vypočtení rychlosti „za“ překážkami rychlostí jako je tomu na obrázku 5.7(c)

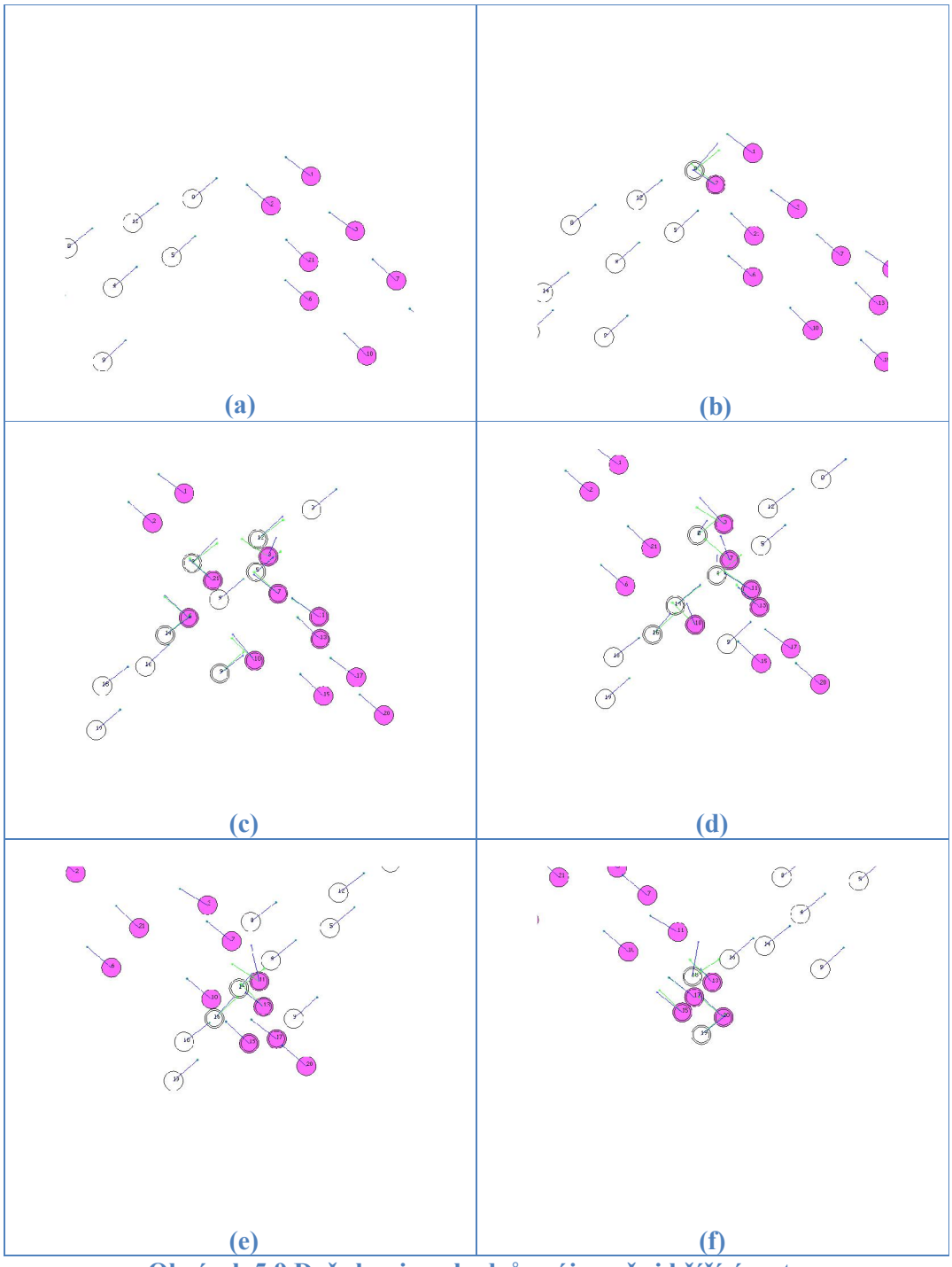
a dojde k uvíznutí všech chodců. Tomuto uvíznutí lze předejít například malým posunem cíle jednoho z chodců. Kvůli tomuto problému byl poslán email autorovi ClearPath, s dotazem na řešení, ale nedorazila odpověď. Vstupní soubory jsou ve složce *four-pedestrians2*.



Obrázek 5.8 Čtyři chodci vzájemně si křížící cesty s uvíznutím

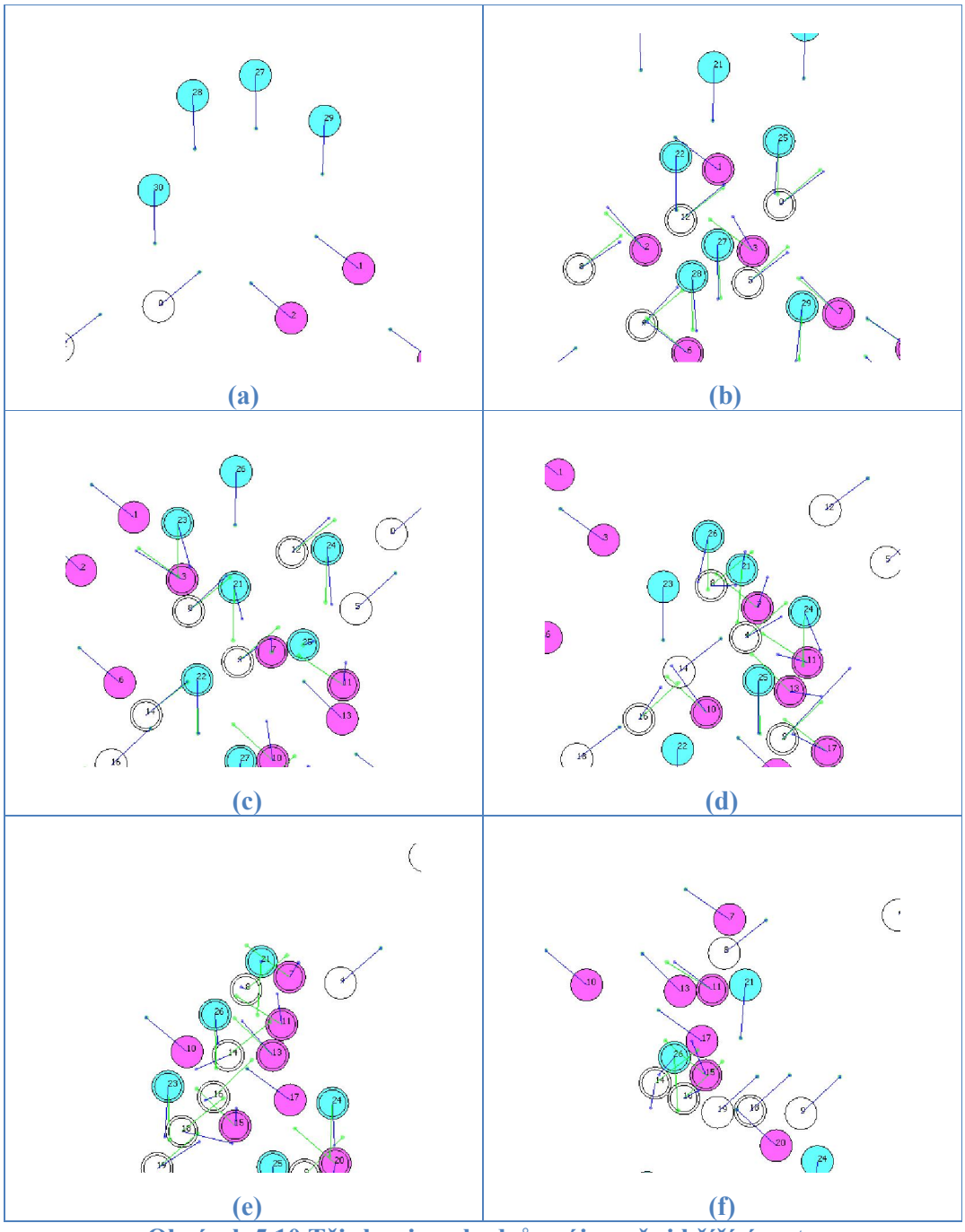
5.2 Pohyb davu

Na obrázcích 5.9 jsou zobrazeny dvě skupiny chodců, které si vzájemně křižují cestu. Chodci s purpurovou barvou směřují vlevo nahoru. Chodci s bílou barvou směřují vpravo nahoru. V průběhu simulace dochází ke kmitání rychlostí, ale celkově simulace probíhá plynule. Vstupní soubory jsou ve složce *crowd-two-groups-crossing*.



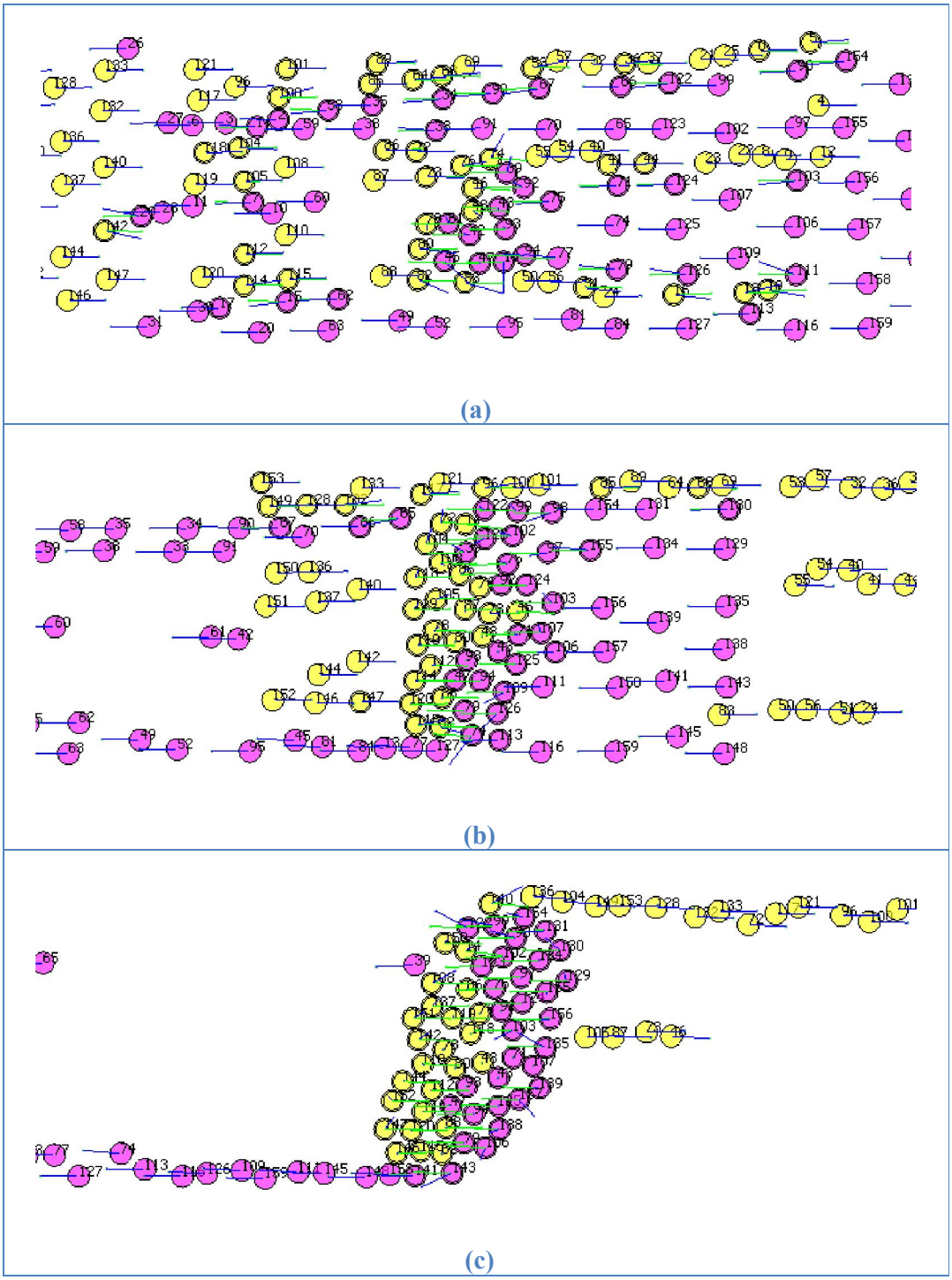
Obrázek 5.9 Dvě skupiny chodců vzájemně si křížící cesty

Na obrázcích 5.10 jsou dvě skupiny z minulého experimentu, plus skupina azurových chodců jdoucích směrem dolů. Na obrázku (e) chodci hodně zpomalí, ale vše je vyřešeno bez kolize. Vstupní soubory jsou ve složce *crowd-three-groups-crossing*.



Obrázek 5.10 Tři skupiny chodců vzájemně si křížící cesty

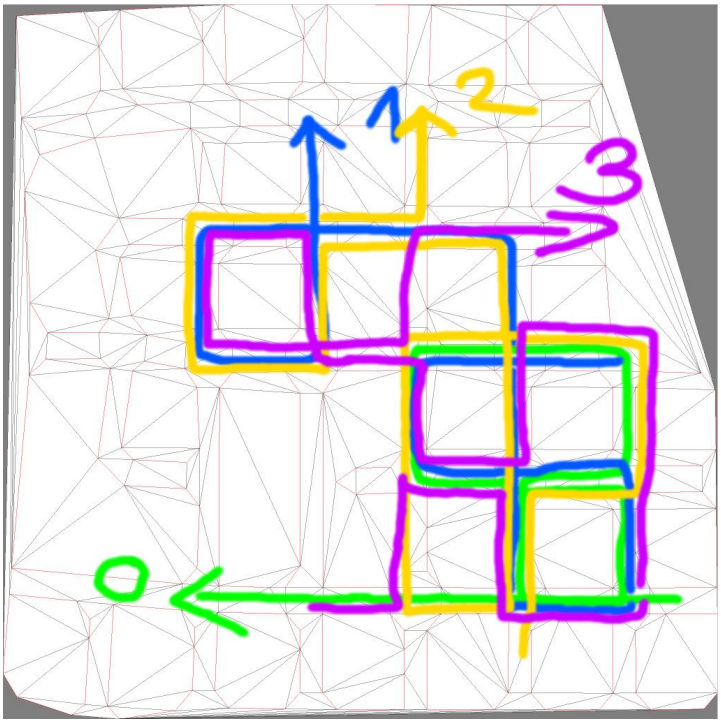
Na obrázcích 5.11 je sto šedesát chodců pohybujících se ve dvou skupinách přímo proti sobě. Při simulaci je vidět drobné trhání animace. Skupiny při simulaci vytvářejí jev pásů viděný v reálných davech. Bohužel v pokročilém stádiu simulace dojde k zahlcení toku chodců. Problém je následně vyřešen, ale jen díky tomu, že nepřicházejí další chodci. Vstupní soubory experimentu jsou ve složce *crowd-two-groups*.



Obrázek 5.11 Tři skupiny chodců vzájemně si křížící cesty

5.3 Procházení městem

Od Bc. Jakuba Szkandery byla v rámci vzájemné spolupráce obdržena výstupní data z jeho aplikace, popsané v kapitole 2.2.3. Jsou to data čtyř chodců pohybujících se virtuálním městem podle obrázku 5.12. Protože v aplikaci IndAlgoTest nejsou stejné barvy jako v obdrženém obrázku, bylo při simulaci použito mapování zobrazené v tabulce 5.2.



Obrázek 5.12 Cesty chodců ve virtuálním městě

Barva v obrázku	Barva v simulaci
oranžová	žlutá
fialová	purpurová
modrá	azurová
zelená	bílá

Tabulka 5.2 Mapování barev obrázku a simulace

Aby nebyla simulace prázdná, jsou všichni čtyři chodci do simulace přidáváni padesátkrát za sebou. Vytvoří tak skupinu ve formaci čáry. Protože algoritmus není schopen řešit překážky typu polygon, nejsou řešeny kolize s budovami města a město není vykresleno. Obrázky ze simulace nejsou přiloženy, protože simulační scéna je velmi rozlehlá. Simulaci lze spustit pomocí vstupního souboru ve složce *szkandera*.

6 Závěr

Tato diplomová práce přinesla, alespoň rámcový přehled algoritmů nyní používaných pro simulaci navigace chodců. Z tohoto přehledu byl vybrán algoritmus *ClearPath*, který se nejlépe hodil pro simulování velkého počtu chodců jako množiny agentů. Tento algoritmus byl dále upraven, aby bylo možné simulovat ještě větší počet chodců. Implementace byla ztížena vysokou složitostí studovaných materiálů.

Výsledný program je schopný simulovat lokální navigaci chodců, pokud jsou dodány cíle, kterými má agent po cestě projít. Lze současně simulovat stovky chodců v reálném čase, a program umožnuje výslednou animaci ukládat ve formě xml dat pro program, který dokáže výsledek simulace zobrazit ve 3D animaci nebo lze ukládat přímo náhled simulované scény do PNG obrázků. Proto se může výsledná animace programem předpočítat a je tak možné simulovat i více chodců. Důvodem, že simulace není rychlá, je pravděpodobně špatná optimalizace implementace a také to, že nebyla použita vhodná struktura pro ukládání chodců.

Další práce na programu by proto měly odstranit tyto dva hlavní nedostatky. Je také nutné zakomponovat do programu řešení dalších typů statických překážek (nyní lze v simulaci mít pouze překážky ve tvaru kruhu, které nejsou dostatečné pro simulaci měst). Další možností vylepšení je počítání simulace ve více vláknech.

Editor simulované scény, který je nyní v programu se výborně hodí pro navržení scén o malém počtu chodců. Pokud by byla zrychlena simulace, bylo by výhodné rozšířit editor scén o možnosti práce s velkým počtem chodců.

Reference

[Lee08] LEE H. K., LEE J., KWON T. T. S.: Group Motion Editing. *ACM SIGGRAPH* (2008)

[Tre06] COOPER S., POPOVIČ Z., TREUILLE A.: Continuum Crowds. *ACM SIGGRAPH* (2006)

[Guy09] GUY S., CHHUGANI J., KIM C., SATISH N., LIN M. C., MANOCHA D., DUBEY P.: Guy: ClearPath: Highly Parallel Collision Avoidance for Multi-Agent Simulation. *Proceedings of Symposium on Computer Animation* (2009)

[Guy10] CHHUGANI J., CURTIS S., DUBEY P., LIN M. C., MANOCHA D., GUY S.: PLEdestrians: A Least-Effort Approach to Crowd Simulation. *ACM SIGGRAPH/ Eurographics Symposium on Computer Animation* (2010)

[Guy11] GUY S., LIN M. C., MANOCHA D., BERG J.: Reciprocal n-body Collision Avoidance. *The 14th International Symposium ISRR* (2011)

[Nar09] GOLAS A., CURTIS S., LIN M. C., NARAIN R.: Aggregate Dynamics for Dense Crowd Simulation. *ACM SIGGRAPH Asia* (2009)

[Ber08] BERG J., LIN M., MANOCHA D.: Reciprocal velocity obstacles for realtime multi-agent navigation. *Proceedings of IEEE Conference on Robotics and Automation* (2008)

[Fio98] FIORINI P., SHILLER Z.: Motion Planning in Dynamic Environments using Velocity Obstacles. *International Journal on Robotics Research* (1998)

[Ben14] YOO I., VANEK J., NIZOVTSEVA M., ADAMO-VILLANI N., BENES B.: Sketching human character animations by composing sequences from large motion database. *The Visual Computer* (2014)

[Tom12] TOMICZKOVÁ S.: Minkowského operace a jejich aplikace, *http://www.amathnet.cz/Portals/0/workshopy/Beskydy%202012/dokumenty/122012odp/Min_suma.pdf* (2012)

[Hot09] HOTZLEIN R.: Fluids v.2 - A Fast Open Source Fluid Simulator (online), *http://www.rchoetzlein.com/eng/graphics/fluids.htm* (2009)

Použité zkratky

Následující tabulka obsahuje zkratky použité v této práci.

ARfC	Animation Renderer for Crowd (aplikace pro vytváření animací chodců)
BE	boundary edges (hraniční přímky)
FVO	ClearPath velocity obstacle (speciální překážka rychlosti představená v ClearPath algoritmu)
ORCA	Optimal Reciprocal Collision Avoidance (model lokální navigace chodců pro optimální vzájemné přecházení kolizím)
PCR	potentially colliding region (region potenciální kolize)
RVO	reciprocal velocity obstacle (vzájemná překážka rychlosti)
SPH	Smoothed-particle hydrodynamic (metoda výpočtu toku tekutin)
UIC	unilateral incompressibility constraint (jednostranné omezení nestlačitelnosti)
VO	velocity obstacle (překážka rychlosti)

Přílohy

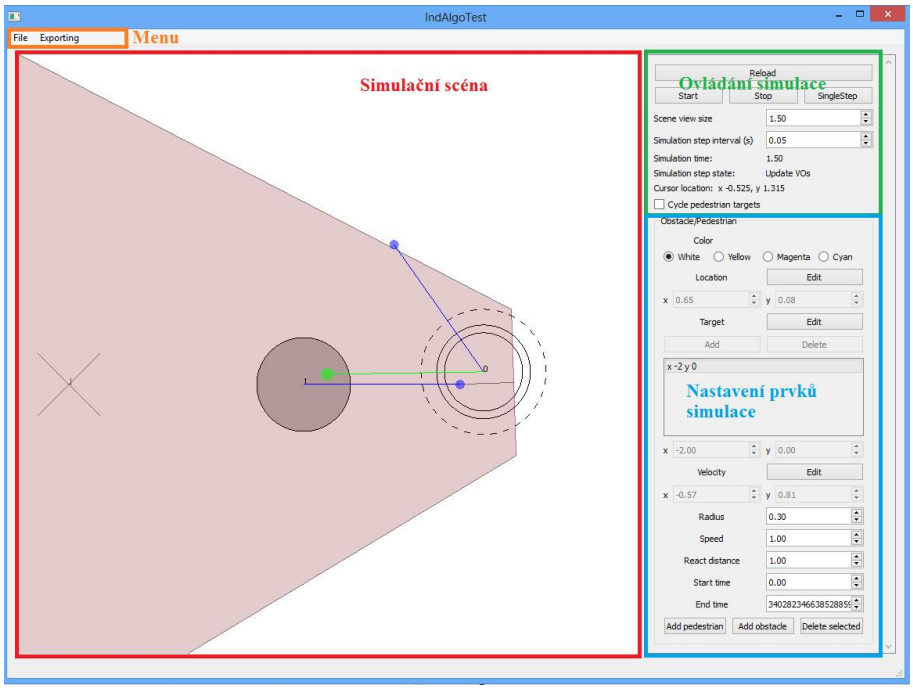
A. Uživatelská dokumentace

Aplikace *IndAlgoTest* byla vyvinuta pro předvedení možností algoritmu pro simulaci chodců. Zdrojový kód je na CD ve složce *src* a verze přeložená pro Win32 je ve složce *bin*. Příklady vstupních i výstupních souborů jsou ve složce *experiments*.

Ke spuštění v operačním systému Windows 8 není třeba instalovat žádné doplňky. Pro přeložení aplikace je potřeba mít nainstalovanou knihovnu Qt[1] ve verzi 5.0.2. Projekty aplikace jsou vytvořeny ve vývojovém prostředí Microsoft Visual Studio 2010.

Popis ovládání aplikace IndAlgoTest

Na obrázku B.1 je vidět uživatelské rozhraní aplikace IndAlgoTest. Aplikaci lze rozdělit na čtyři menší části: menu, obraz simulované scény, panel ovládání simulace a panel nastavení simulovaných prvků. Všechny tyto části budou podrobně popsány.



Obrázek B.1 Uživatelské rozhraní programu IndAlgoTest

[1] http://qt-project.org/

Funkce menu jsou vysvětleny v tabulce B.1.

File -> Load scene	Načte existující soubor simulační scény. Soubory musí mít příponu iascene.
File -> Save scene	Uloží aktuálně zobrazenou simulační scénu.
File -> Load configuration	Načte existující soubor konfigurace prvků[2]. Soubory musí mít příponu iaconf.
File -> Save configuration	Uloží aktuální nastavení jednotlivých prvků scény do souboru.
Exporting -> Start Skeleton Recoring	Spustí vytváření xml souboru pro program Animation Renderer for Crowd a zobrazí možnost pro ukončení nahrávání.
Exporting -> Start Image Recoring	Spustí ukládání obrázků ze simulace a zobrazí možnost pro ukončení a uložení výsledných souborů do formátu PNG.

Tabulka B.1 Funkce menu

Funkce panelu pro ovládání simulace jsou vysvětleny v tabulce B.2.

Reload	Načte poslední načtený soubor konfigurace a simulační scény. Pro soubor simulační scény je uložen poslední načtený soubor i mezi jednotlivými spuštěními programu IndAlgoTest
Start	Spuštění simulace.
Stop	Zastavení simulace.
SingleStep	Spuštění následující části kroku simulace.
Scene view size	Ovládání zobrazené velikosti simulační scény.
Simulation step interval	Ovládání délky kroku simulace v sekundách.
Simulation time	Ukazatel času simulace udaný v sekundách.
Simulation step state	Ukazatel aktuálního stavu kroku simulace. Možné stavy jsou Update VOs, Update velocities, Update locations. Ve stavu Update VOs jsou počítány překážky rychlosti. Ve stavu Update velocities jsou počítány rychlosti. Ve stavu Update locations jsou počítány nové pozice prvků scény.
Cursor location	Ukazatel pozice kurzoru v simulační scéně.
Cycle pedestrians targets	Pokud je tento checkbox zaškrtnutý, chodec po dosažení posledního cíle pokračuje znovu k prvnímu cíli, jinak je ze simulace smazán.

Tabulka B.2 Funkce ovládání simulace

[2] Prvek simulační scény může být chodec nebo překážka

Vysvětlení grafických prvků scény je v tabulce B.3.

Kruhy	Kruh reprezentuje chodce nebo překážku. Překážka je vždy vyplněná šedou barvou. Chodci mohou mít barvu bílou, purpurovou, azurovou nebo žlutou. Číslo uprostřed značí index prvku.
Přerušovaná kružnice	Indikace aktivního chodce nebo překážky.
Červeně vyplněné oblasti	Překážky rychlosti od ostatních chodců k aktuálně vybranému chodci.
Šedivý křížek	Značí cíl aktivního chodce. Čísla značí indexy jednotlivých cílů.
Kružnice uvnitř některých chodců	Značí, že se chodec nemůže pohybovat svou požadovanou rychlostí a musí vypočítávat novou rychlost pomocí překážek rychlosti.
Zelené čáry	Preferovaná rychlost chodce. Konec je pro lepší viditelnost označen malým kruhem stejné barvy.
Modré čáry	Rychlost, kterou se chodec pohyboval minulý krok nebo rychlost, kterou se bude pohybovat v následující části kroku pro pohyb.

Tabulka B.3 Grafické prvky scény

Funkce panelu pro nastavení prvků scény jsou vysvětleny v tabulce B.4.

Color	Barva, kterou bude vyplněn příslušný chodec.
Location	Prvky pro ovládání pozice vybraného prvku. Pokud je stisknuto tlačítko Edit, lze vybrat pozici prvku přímo ve scéně.
Target	Prvky pro nastavení cílů chodce. Lze přidávat, mazat nebo upravovat existující cíle. Existující cíle jsou zobrazeny v listu. Pokud je stisknuto tlačítko Edit, lze vybrat pozici cíle přímo ve scéně.
Velocity	Prvky pro nastavení aktuální rychlosti chodce. Pokud je stisknuto tlačítko Edit, lze vybrat rychlost prvku přímo ve scéně.
Radius	Nastavení poloměru prvku.
Speed	Nastavení rychlosti chodce.
React distance	Nastavení minimální vzdálenosti od prvků, pro které se budou počítat překážky rychlosti.
Start time	Nastavení času začátku působení prvku v simulaci.
End time	Nastavení času konce působení prvku v simulaci.
Add pedestrian	Přidání nového chodce do scény
Add obstacle	Přidání nové překážky do scény
Delete selected	Odebrání prvku scény

Tabulka B.4 Nastavení prvků scény

Ovládání scény pomocí myši je vysvětleno v tabulce B.5.

Pohyb scénou	Podržením pravého tlačítka myši.
Přiblížení/oddálení	Pomocí kolečka myši.
Vybrání prvku	Stiskem levého tlačítka myši.
Speciální mód	Pokud je v panelu nastavení prvku vybrána editace pozice, cíle nebo rychlosti, stisk levého tlačítka vybere hodnotu.
Posun prvku	Podržením levého tlačítka myši.

Tabulka B.5 Ovládání scény pomocí myši

Ovládání scény pomocí klávesových zkratek je vysvětleno v tabulce B.6:

A	Přidá do scény chodce. Pokud jsou editovány cíle, je přidán cíl.
D	Smaže prvek scény. Pokud je editován cíl, je cíl smazán.
L	Vybere editační mód pozice.
T	Vybere editační mód cíle.
N nebo Enter	Ukončí editační mód.

Tabulka B.6 Ovládání scény pomocí klávesových zkratek

B.Ukázky xml souborů použitých v aplikaci

Ukázka xml souboru pro uložení simulované scény (*.iascene)

```
<indAlgoConf><!-- kořen xml souboru -->
 <obstacles><!-- list prvků scény -->
  <pedestrian index="0" radius="0.4">
<!-- prvek chodce, jeho poloměr a index -->
   <location x="2" y="0"/><!-- pozice prvku -->
   <velocity x="0" y="0"/><!-- rychlost prvku -->
   <targets><!-- list cílů chodce -->
    <target x="-2" y="0"/><!-- cíl chodce -->
   </targets>
  </pedestrian>
  <obstacle index="1" radius="0.5"><!-- prvek překážky -->
   <location x="-2" y="0"/>
   <velocity x="1" y="0"/>
  </obstacle>
 </obstacles>
</indAlgoConf>
```

Ukázka xml souboru konfigurace simulované scény (*.iaconf)

```
<indAlgoConfiguration><!-- kořen xml souboru -->
 <obstacles><!-- list prvků scény -->
  <pedestrian index="0" reactDistance="1.7" speed="1.8">
<!-- prvek chodce, reakční vzdálenost, rychlost a index -->
   <timeSpan start="3" end="6"/>
<!-- čas působení ve scéně -->
  </pedestrian>
  <obstacle index="1"><!-- prvek překážky -->
   <timeSpan start="3" end="6"/>
  </obstacle>
 </obstacles>
</indAlgoConfiguration>
```

Ukázka xml souboru pro aplikaci Animation Renderer for Crowd

```
<crowd><!-- kořen xml souboru -->
 <timestep t="0"><!-- snímek simulace s aktuálním časem -->
  <pedestrian id="0" new="true">
<!-- chodec s indexem a indikací že je nový ve scéně -->
   <position x="1.15" y="0"/><!-- pozice chodce -->
   <direction x="-1" y="0"/><!-- směr chodce -->
   <state>agent</state><!-- typ chodce agent/fluid -->
   <movement>walking</movement>
<!-- typ pohybu chodce standing/walking/running -->
  </pedestrian>
  <pedestrian id="1" new="true">
   <position x="-1.15" y="0"/>
   <direction x="1" y="0"/>
   <state>agent</state>
   <movement>walking</movement>
  </pedestrian>
 </timestep>
</crowd>
```