VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Navigace skupiny agentů

Semestrální projekt

Vypracoval:
Pavel Drábek

Vedoucí práce:
Ing. Martin Němec, Ph.D.

2017

Obsah

1 Úvod

Umělá inteligence se v herním průmyslu využívá už od jeho vzniku. Ať už se jedná o propracovaný strategický systém, navigace po světě nebo detekce a přizpůsobení obtížnosti hráči. Oblast navigace by se dala rozdělit na navigaci samotného jedince a řízení více agentů. Tato práce se bude zabývat právě navigací více agentů.

Simulace davu má za cíl věrně napodobit chování velké množiny objektů. Zdánlivě složité chování však může být dosaženo definováním několika základních pravidel každého jedince. Chování samotného jedince se může jevit jako zmatené, avšak velká skupina stejně naprogramovaných jedinců může vykazovat známky inteligentního chování.

Takové chování uplatňujeme nejen ve filmovém a herním průmyslu, ale i v architektuře, nácviku vojenských strategií, navrhování evakuačních plánů, simulování požárních poplachů či chování robotů. [1] Díky moderním technologiím můžeme napodobit chování davu, které by nás stálo nejen velké množství času, ale i finančních prostředků. Například filmová bitva velkých armád, stadion plný fanoušků apod. Můžeme vytvořit i takové chování, se kterým bychom se v reálném životě nikdy nesetkali - ve filmu Drákula: Neznámá legenda (2014) ovládá hlavní postava desetitisíce netopýrů.

Takovéto systémy většinou vykazují chaotické chování, protože nepatrná změna hodnoty jednoho parametru, byť jen zaokrouhlením, bude mít dopad na pozdější výsledek. Nicméně dá se zajistit i chování, které pro stejnou situaci chaotické chování nevykazují.

1.1 Cíl projektu

Cílem projektu je vytvořit aplikaci, která bude využívat bioinspirované algoritmy pro umělou inteligenci. Aplikace bude simulovat chování vybraných objektů z reálného světa pomocí různých algoritmů. Cílem bude využít existující možnosti profesionálních herních enginů.

V projektu vytvoříme prostředí, kde bude možné jednotlivé chování simulovat a provedeme ve vytvořeném prostředí jednotlivé simulace.

Takto implementovaný algoritmus najde využití v oblasti virtuální reality, rozšířené reality, či herním průmyslu.

2 Mravenčí kolonie

Algoritmus popsaný začátkem 90. let známý pod jménem Optimalizace mravenčí kolonií patří do kategorie hejnových algoritmů. [2] Jedná se o heuristickou metodu hledání nejkratší cesty. Velká výhoda tohoto algoritmu spočívá v jeho adaptabilitě. Dá se využít v prostředí, kde v průběhu času možné cesty zaníkají a vznikají nové.

Přestože má každý jedinec definované základní možnosti, chování celého mraveniště je velmi strukturované. Jedná se o výsledek koordinovaných iterací. Tato práce zkouší využít algoritmus pro navigaci agentů místo pro optimalizaci nejkratší cesty.

2.1 Popis metody

Jednotliví agenti (mravenci) putují nahodile po okolí mraveniště. Jakmile naleznou potravu, vrací se do mraveniště stejnou cestou jakou k potravě dorazili. Po cestě zpátky za sebou zanechávají feromonovou stopu. Pokud jiný mravenec tuto stopu zachytí, je větší pravděpodobnost, že se vydá cestou feromonu než náhodnou. Mravenec k ve vrcholu i, grafu $G = (N, A)$ se přesune do sousedního vrcholu j s pravděpodobností p_{ij}^k:

$$p_{ij}(t) = \frac{[\tau_{ij}(t)^\alpha][\eta_{ij}]\beta}{\sum_{j=1}^{n} [\tau_{ij}(t)^\alpha][\eta_{ij}]\beta},$$

kde N_i^k představuje množinu všech vrcholů dostupných mravenci k. Koeficienty α, β a ρ jsou obecnými parametry algoritmu.

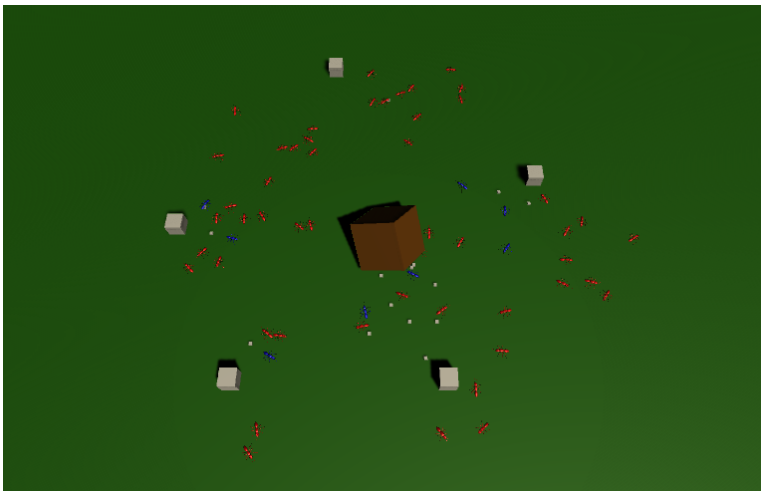
Feromon postupně vyprchává, takže jakmile potrava zanikne, postupně zanikne i feromonová stopa a mravenci začnou zase náhodně prohledávat. Každou iteraci se každá feromonová stopa upraví podle vztahu:

$$\tau_{ij} = (1 - \rho)\tau_{ij}.$$

2.2 Alternativní využití

V rámci projektu byl tento algoritmus lehce upraven pro potřebu navigace agentů. Nejprve byla odebrána nutnost grafu. Mravenci se pohybovali zcela náhodně po mapě, kdy každý mravenec ušel náhodnou vzdálenost v definovaném intervalu a poté se náhodně rozhodl kterým směrem půjde příště. Směr, který si mohl mravenec vybrat byl omezen intervalem úhlu jeho vidění.

Každý mravenec měl definováno jak dlouho vydrží hledat potravu či feromonovou stopu. Po uplynutí této doby se musí vrátit do mraveniště a začít hledat znovu. Tímto bylo zamezeno, aby se mravenec vzdálil od mraveniště příliž daleko a aby jeho cesta k potravě byla co možná nejkratší.



Obrázek 1: Implementovaný model mravenčí kolonie v průběhu simulace

Na obrázku 1 lze vidět uprostřed mraveniště, kolem kterého je rozmístěno 5 bílých krychlí představující potravu. Mravenci, kteří potravu nalezli a vracejí se do mraveniště jsou označeni modrou barvou, zatímco červení mravenci potravu teprve hledají.

3 Model Boid

Počítačový model pro řízení hejna zvířat jako jsou ptáci a ryby popsal Craig Reynolds v roce 1986. Jednotlivé jedince nazval Boidy. Samotný model pak prezentoval na konferenci SIGGRAPH o rok později.

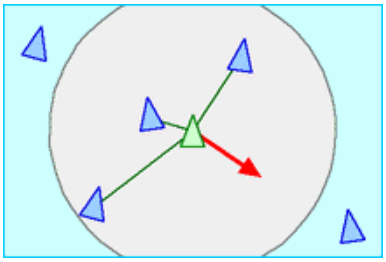
Od roku 1987 byl model Boid použit pro velké množství simulovaného chování hejna. Prvním, veřejně známým použitím byl film Batman se vrací (1992) od Tima Burtona. Obsahuje počítačově simulované chování hejna netopýrů a skupiny tučňáků, které bylo vytvořeno upravenou verzí modelu Boid. [4]

3.1 Popis modelu

Boid pracuje s agenty reprezentovanými pozicí a rychlostním vektorem. Model lze aplikovat pro pohyb ve dvou i třech rozměrech.

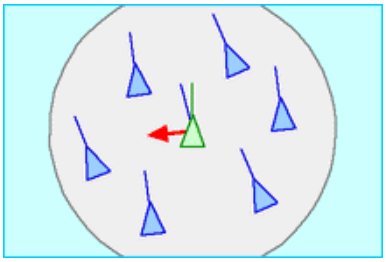
Každý agent se řídí třemi základními pravidly, podle kterých upraví svůj rychlostní vektor každý časový úsek. Jedná se o Separaci, Vyrovnání a Kohezi.

1. **Separace,**
   Agent se snaží nesrazit s jinými agenty. Upravuje proto svůj směrový vektor, aby se dostal od příliš blízkých agentů.



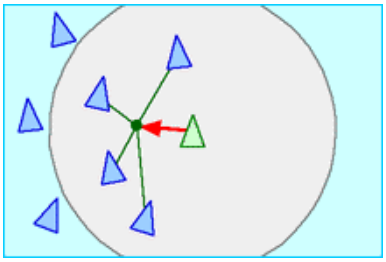
Obrázek 2: Separace (Separation) [4]

2. **Vyrovnání,**
   Agent se chce pohybovat stejným směrem a rychlostí jako jeho sousedé. Upravuje svůj vektor podle průměru z jeho okolí.



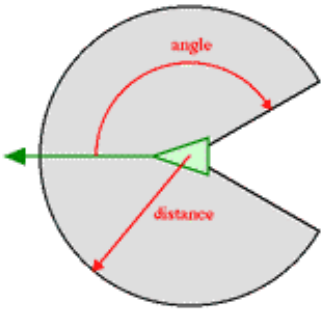
Obrázek 3: Vyrovnání (Alignment) [4]

3. **Koheze,**
   Agent upravuje svůj rychlostnní vektor tak, aby se dostal do pomyslného středu všech sousedních agentů.



Obrázek 4: Koheze (Cohesion) [4]

Každý boid má přímý přístup ke všem objektům ve scéně, nicméně tento model vyžaduje, aby reagoval pouze na své okolí. To je definováno maximální vzdáleností a zorným úhlem agenta. Můžeme přidávat i další pravidla jako třeba minimální vzdálenost nebo úhel mezi dvěmi agenty apod. Pravidla, podle kterého jedinec rozlišuje okolí, si můžeme představit jako smysly daného zvířete.



Obrázek 5: Okolí agenta [4]

Jak bylo na začátku kapitoly naznačeno, model můžeme různě zozšiřovat. Sám Reynolds přišel rok po představení svého modelu na stejné koferenci s rozšířením pro výhýbání se překážkám. Detailněji vše vysvětluje ve své práci Not bumping into things [5]. V podstatě se jedná o úpravu výsledného vektoru rychlosti pokud se před jedincem nachází nějaký objekt.

Pokud jednotlivým pravidlům přiřadíme váhy, můžeme následnými úpravami těchto vah měnit chování konkrétního jedince či celého hejna. Boidi tak můžou uzpůsobovat své chování v závislosti na prostředí. Například je-li v okolí dravec, budou mít mezi sebou větší rozestup, hlídat si větší okolí, rychleji reagovat na změnu směru atd.

4 Implementace modelu Boid

Implementace probíhala v jazyce $C\#$ a vývojovém prostředí Unity3D [6].

4.1 Základní model Boid

Jednotlivé vektory, které budou upravovat aktuální vektor rychlosti podle pravidel Koheze $\vec{v}_c$, Vyrovnání $\vec{v}_a$, Separace $\vec{v}_s$ byly realizovány těmito vzorci, kde n je počet agentů v okolí, $\vec{v}$ rychlost a P pozice aktuálního agenta, $\vec{v}_{Bi}$ rychlostní vektor a P_Bi pozice agenta v okolí, R_c určuje radius, od kterého se agent snaží vyhýbat kolizím:

$$\vec{v}_c = \frac{\sum_{i=1}^{n} P_{Bi}}{n} - P, \qquad \vec{v}_a = \frac{\sum_{i=1}^{n} \vec{v}_{Bi}}{n} - \vec{v}, \qquad \vec{v}_s = -\sum_{i=1}^{n} \frac{(\overrightarrow{PP}_{Bi}) * R_c}{\mid \overrightarrow{PP}_{Bi} \mid}.$$

Vektory pro kohezi a vyrovnání počítáme pouze pro agenty v okolí. Agent se nachází v okolí jiného agenta pokud $\mid \overrightarrow{PP}_{Bi} \mid \leq R_v \wedge acos(\overrightarrow{PP}_{Bi} \cdot \vec{v}) \leq \alpha$, kde R_v je radius dohledu a α zorný úhel agenta.

Reynolds ve svém článku neuvádí jakým stylem jednotlivé pravidla řídit ani jaké váhy použít. Byla tedy vytvořena váha pro každé pravidlo, díky kterému můžeme řídit prioritu těchto pravidel. Nový rychlostní vektor $\vec{v}_{t+1}$ získáme přičtením všech pravidel k aktuálnímu rychlostnímu vektoru $\vec{v_t}$:

$$\vec{v}_{t+1} = \sum v_i \cdot w_i \qquad v = \{\vec{v}_c, \vec{v}_a, \vec{v}_s\}, \qquad w = \{\vec{w}_c, \vec{w}_a, \vec{w}_s\}.$$

Pro lepší vizuální představu jak model Boid funguje byl vytvořen jednoduchý editor, ve kterém je možné v reálném čase nastavit upravit váhy jednotlivým pravidlům a vidět změnu v chování hejna, aniž by bylo potřeba pouštět celou simulaci znovu. To umožnilo rychleji dosáhnout požadovaného chování hejna. Pro správné nastavení vah by teoreticky šel použít i evoluční algoritmus, avšak na tak jednoduchém příkladě není jasné jakým způsobem určovat fitness.

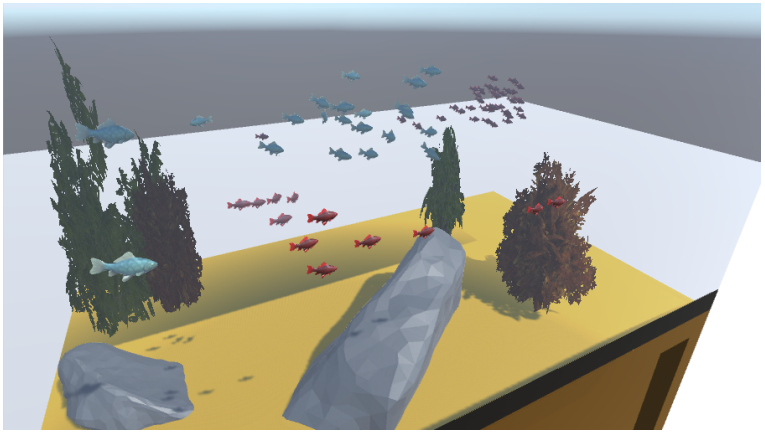
Podařilo se dosáhnout fungujícího modelu hejna, které však nekontrolovaně cestovalo po světě.

4.2 Rozšíření o pravidlo Cíl

Model Boid popisuje pouze chování jednotlivce v hejnu. Pro potřebu kontrolovat pohyb celého hejna, byl agent rozšířen o další pravidlo. Každý agent se snaží dostat do cíle, který je reprezentován pozicí bodu ve světě.

Od toho pravidla vyžadujeme, ať přitahuje agenta k cíli, ale neznehodnotí zbylé 3 pravidla s narůstající vzdáleností od cíle. Normalizování směrového vektoru k cíli se může zdát jako krok správným směrem. V praxi se však ukázalo, že když je takový agent velmi blízko svého cíle, normalizace vektor naopak zvětší. To má za následek příliš agresivní snahu jedince dostat se do cíle. Proto velikost směrového vektoru od jedince k cíli pouze ořízneme na hodnotu vzdálenosti, od které má být agent méně agresivní.

Pomocí nového pravidla je možné kontrolovat pozici hejna a pozicovat ho dle svých potřeb. Pro názornou ukázku vznikla scéna s akváriem a rybičkami. V určitém časovém intervalu nastavuji všem agentům (rybám) náhodně vygenerovaný cíl uvnitř akvária.



Obrázek 6: Hejno 20 modrých a 50 červených ryb

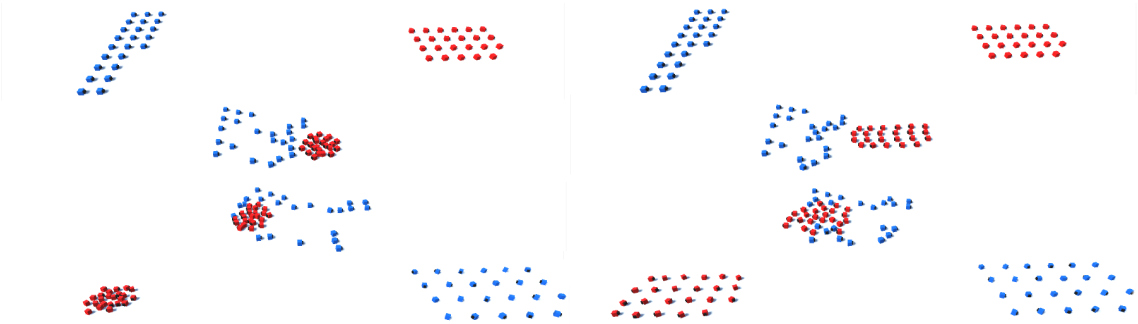
Na obrázku 6 můžeme vidět 2 různě početné a nastavené skupiny agentů. První je skupina 20 velkých modrých ryb, které jsou pomalé, drží se dál od sebe (velká váha u separace) a pomalu mění svůj směr. Druhá skupina obsahuje 50 malých červených ryb, které jsou rychlejší, agilnější, ale více se drží pohromadě (velká váha u koheze a vyrovnání).

Stejnými pravidly se dá simulovat i dav lidí, kteří se potřebují dostat z jednoho místa na druhé. Pro tyto účely byl každému jedinci nastaven odlišný cíl. Obrázek 7 znázorňuje srovnání chování 2 skupin poslaných proti sobě. Červená skupina na levém obrázku má nastavené váhy pro Kohezi a Vyrovnání převyšující váhu pravidla pro Cíl, červená skupina napravo ne. Přesné nastavení váhy lze vidět v tabulce:

Tabulka 1: Nastavení vah pro srovnání skupin

skupina	w_c	w_a	w_s	w_t
modrá	0.2	0.2	1.0	0.6
červená vlevo	0.0	0.0	1.0	0.6
červená vpravo	0.5	0.5	1.0	0.6

Každý agent má za úkol dostat se na druhou stranu a zaujmout pozici v pomyslné mřížce. Obrázky ve směru dolů jsou stavem s narůstajícím časem simulace.

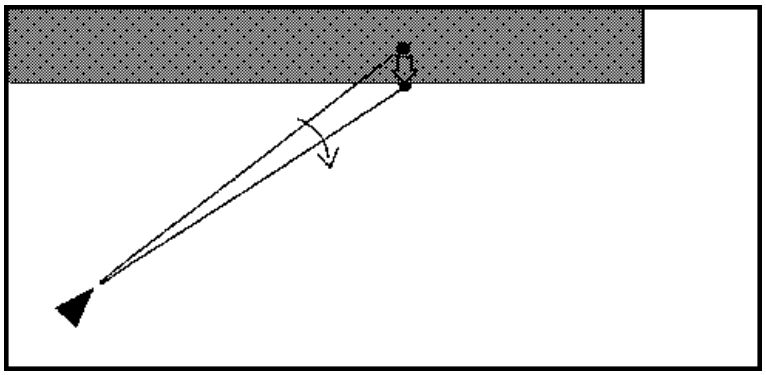


Obrázek 7: Srovnání vah v časech $t \in \{0, 20, 30, 70\}$

Z obrázku 7 můžeme usoudit, že poměr mezi váhou Cíle a váhami Koheze a Vyrovnání nemá na hejno jako celek vliv. V obou případech se hejno dostalo na druhou stranu. Avšak při malé váze pravidla Cíl nedokáže jedinec zaujmout svou koncovou pozici.

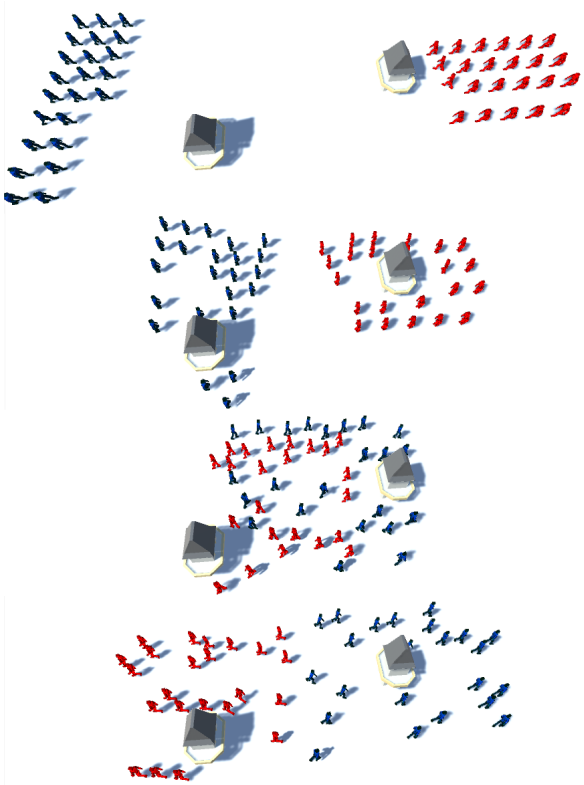
4.3 Rozšíření o pravidlo Vyhni se překážkám

Dalším přidaným pravidlem, díky kterému bude chování hejna více připomínat realitu je uhýbání před překážkami. Každý agent si v závislosti na své rychlosti kontroluje prostor před sebou. Pokud detekuje překážku, vychýlí svůj vektor rychlosti tak, aby se pokusil překážce vyhnout.



Obrázek 8: Vychýlení vektoru rychlosti při detekci překážky [5]

Pro test byly do scény přidány 2 překážky a puštěna obvyklá simulace dvou skupin, která si mají vyměnit místa.

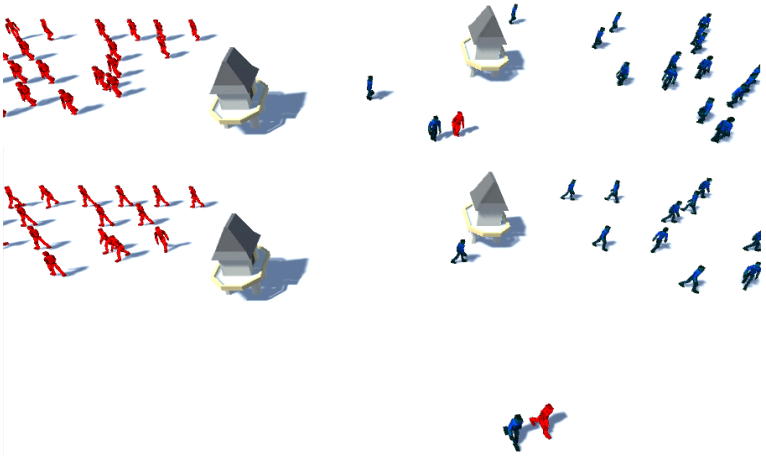


Obrázek 9: 2 skupiny agentů s detekcí překážek v časech $t \in \{0, 10, 20, 30\}$

Přidání tohoto pravidla se dav před překážkou rozdělí a v závislosti na dalších parametrech se může za překážkou zase spojit. Správným nastavením vzdálenosti, ve které agent hledá překážku můžeme docílit věrohodného chování celého davu.

4.4 Omezení modelu

Přestože jednotliví agenti vykazují známky složitého chování, stále se řídí vesměs triviálními pravidly, které mohou zapříčinit nechtěné odchylky. Jednou z nich je tzv. semknutí dvou agentů, kteří se navzájem přetlačují a jeden druhému nedokáží ustoupit. To má za následek nechtěnné chování, kdy tito agenti většinou opustí hejno a může zapříčinit, že začnou procházet překážkami.



Obrázek 10: Srážka a semknutí dvou agentů v časech $t \in \{100, 115\}$

Jednotlivá pravidla modelu jsou nastavena tak, aby se model pokusil vyhýbat kolizím, ale nedokážou zajistit 100% úspěšnost. Do jisté míry se tento problém dá oddálit správným nastavením vah. První testy chování Boid modelu dosahovaly přibližně 200 kolizí/s. Postupně se podařilo chování vyladit až na verzi bez žádné kolize.

Rozšíření o pravidlo vyhýbaní se překážkám, které Reynolds popsal ve své práci [5] pracuje pouze se statickými objekty. Abhinav Golas, Rahul Narain a Ming Lin publikovali v roce 2014 článek Hybrid Long-Range Collision Avoidance for Crowd Simulation [7]. Ve své práci popisují agenta, který umí predikovat pohyb jiných agentů a vyhnout se tak kolizi daleko dříve.

5 Závěr

Pomocí modelu Boid se podařilo simulovat pohyb velké skupiny agentů. Systém byl využit pro simulaci ve dvou i tří rozměrném prostředí. Díky rozšíření o detekci překážek se agenti dokázali dynamicky přizpůsobit prostředí a jevit inteligentní chování.

Výhoda modelu Boid spočívá v absenci detekce kolizí pomocí fyzikálního enginu. Pokud mají agenti nastavenou správnou váhu, udržují si od sebe navzájem dostatečně velkou vzdálenost a není potřeba kolize počítat.

Model Boid je velmi nenáročný na výpočetní výkon. Další možnosti vývoje může být například paralelizace. Algoritmus je možné rozdělit do více vláken nebo počítat na grafické kartě. Dále je možné rozšířit model o predikci pohybu nebo navigační systém.

Reference

[1] Crowd and Multi-agent Simulation. *Gamma: Geometric Algorithms for Modeling, Motion, and Animation* [online]. [cit. 2017-05-04]. Dostupné z: http://gamma.cs.unc.edu/research/crowds/

[2] COLORNI, Alberto, Marco DORIGO a Vittorio MANIEZZO. *Distributed Optimization by Ant Colonies* [online]. ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D), 1991, , 12 [cit. 2017-05-04]. Dostupné z: https://scholar.google.com/citations?view_op=view_citation&hl=cs&user=PwYT6EMAAAAJ&citation_for_

[3] BLUM, Christian. Ant colony optimization: Introduction and recent trends. In: *Https://www.sciencedirect.com* [online]. 2005 [cit. 2017-05-05]. Dostupné z: https://www.ics.uci.edu/ welling/teaching/271fall09/antcolonyopt.pdf

[4] REYNOLDS, Craig. Boids: Background and Update. In: *Reynolds Engineering & Design* [online]. 1995 [cit. 2017-05-03]. Dostupné z: http://www.red3d.com/cwr/boids/

[5] REYNOLDS, Craig. Not Bumping Into Things. In: *Reynolds Engineering & Design* [online]. 1988 [cit. 2017-05-03]. Dostupné z: http://www.red3d.com/cwr/nobump/nobump.html

[6] *Unity 3D* [online]. [cit. 2017-05-04]. Dostupné z: https://unity3d.com

[7] GOLAS, Abhinav, Rahul NARAIN a Ming LIN. *Hybrid Long-Range Collision Avoidance for Crowd Simulation* [online]. ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D), 2013, , 7 [cit. 2017-05-04]. Dostupné z: http://gamma.cs.unc.edu/lookahead/golas-2013-hybridcrowd.pdf

Seznam obrázků