Univerza
v Ljubljani

Fakulteta *za računalništvo*
in informatiko

Dominik Pangeršič

RAČUNALNIŠKI MODEL USKLAJENEGA PRISTAJANJA IN VZLETANJA

DIPLOMSKO DELO
VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE
STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

doc. dr. *Iztok Lebar Bajec*
MENTOR

Ljubljana, 2012

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00261/2012
Datum: 10.04.2012

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **DOMINIK PANGERŠIČ**

Naslov: **RAČUNALNIŠKI MODEL USKLAJENEGA PRISTAJANJA IN VZLETANJA**

COMPUTER MODEL OF COORDINATED LANDING AND TAKEOFF

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

V sredini 80. let prejšnjega stoletja se je z uvedbo t.i. vedenjske animacije v svetu računalniške animacije zgodil velik korak naprej. Vedenjska animacija je segment proceduralne animacije, s pomočjo katere animirani liki postanejo avtonomni in svoje vedenje, vsaj do neke mere, določajo sami. Animator vedenja animiranega lika ne določa več do zadnje potankosti, ampak le v grobem opiše, kako lik zaznava svoje okolje ter kakšne nagone skuša lik izpolnjevati. Naprej pa animacijo prepusti interakciji med liki samimi.

V diplomski nalogi preučite odprtokodno ogrodje OpenSteer, ki s pomočjo baze osnovnih vedenjskih nagonov omogoča gradnjo avtonomnih likov. Metodologijo gradnje in animacije avtonomnih likov prikažite na primeru računalniškega modela, namenjenega animaciji usklajenega pristajanja in vzletanja jat ptic z žice. Dinamiko pristajanja in vzletanja analizirajte in kritično komentirajte.

Mentor:

doc. dr. Iztok Lebar Bajec

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani Dominik Pangeršič, z vpisno številko **63080318**, sem avtor diplomskega dela z naslovom:

Računalniški model usklajenega pristajanja in vzletanja

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Iztoka Lebarja Bajca,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 26. junija 2012 Podpis avtorja:

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Dominik Pangeršič

Računalniški model usklajenega pristajanja in vzletanja

POVZETEK

Naloga predstavlja rešitev za eno izmed mnogih zahtev avtomatizirane (vedenjske) animacije, vzletanje in pristajanje jat avtonomnih agentov oziroma v našem primeru skupin ptic. Uporabljena je osnovna logika vedenja avtonomnih agentov, ki jo je implementiral in opisal Craig Reynolds. Želja pri tem je bila prikazati njihovo gibanje po navideznem svetu, čim bolj življenjsko a nekako na improvizacijski način.

V sami osnovi smo se bolj osredotočili na algoritem prihajanja z omejenim zaznavanjem prostora avtonomnih agentov, v katerem se nahaja enostavno področje za pristajanje. Samo delovanje oziroma osnovna logika posameznega agenta se je priredila za delovanje z uporabo več stanj tako, da se agenti vedejo oziroma delujejo skupaj na podlagi trenutnih želja in enako "mislečih" sosednjih agentov.

Ključne besede: prihajanje, zaznavanje okolja, igre, simulacija, skupinsko vedenje, umetno življenje

University of Ljubljana
Faculty of Computer and Information Science

Dominik Pangeršič

Computer model of coordinated landing and takeoff

ABSTRACT

This work presents a solution to one of the many requirements of autonomous character animation, landing and take offs of autonomous agents (characters), in our case groups of birds. The underlying logic we used was the behavior of autonomous agents, which was implemented and described by Craig Reynolds. Our desire was to show their navigation around their virtual world in a somehow improvised life-like manner.

Our principal focus was on the arrival algorithm and a limited perception of nearby autonomous agents and a simple landing area (cabels). The behaviour or basic logic of an individual agent was modifyed to operate in diferent states, so that agents behave or better work together based on their current desires and same-minded neighbour agents.

Key words: arrival, sensing the environment, games, simulations, group behavior, artificial life

ZAHVALA

Zahvaljujem se svojim staršem in tudi vsem ostalim, ki so mi na kakršen koli način pomagali pri izdelavi pričujočega dela. Še posebej pa bi se rad zahvalil mentorju Iztoku Lebarju Bajcu za njegovo pomoč in potrpljenje.

— Dominik Pangeršič, Ljubljana, junij 2012.

KAZALO

1 Uvod

Obnašanje živih bitij, vzrokov in posledic določenega obnašanja, je že stoletja deležno velike pozornosti. Posebej zanimiva so na primer bitja, ki se povezujejo v večje skupine in delujejo kot eno, to so: ptice, ribe, mravlje, itd. Eden vodilnih korakov k boljšemu razumevanju obnašanja ptic je bilo prav opazovanje načina letenja ptic v jatah. Računalniška simulacija vedenja le teh pa omogoča številna preigravanja različnih situacij in problemov.

1.1 Letenje v jatah - tipično vedenje pri pticah

Letenje v jatah (angl. *flocking*) kot vedenje je vidno, ko npr. skupina ptic z značilnostmi jate (angl. *flock*) leti, išče hrano, oziroma v našem primeru pristaja k počitku. Osnovna značilnost jate je usklajenost v: gibanju, razmikih, hitrosti, usmerjenosti in času vzleta ali pristanka. Zelo podobna vedenja, kot so plavanje v trumah (angl. *schoaling*) pri ribah, zbiranje v rojih (angl. *swarming*) pri žuželkah in zbiranje v čredah (angl. *herd*) pri kopenskih živalih. Zaradi želje po ugotovitvi ali so to edina letenju v jatah podobna vedenja, je bila v Nemčiji oziroma natančneje v Kölnu leta 2008 narejena raziskava na temo črednega obnašanja bitij, ki svoje odločitve sprejemajo na podlagi dejanj ostalih.

Raziskavo sta naredila in izvedla biologa Jens Krause in John Dyer [1] z Univerze v Leedsu, ki je prikazala čredno obnašanje pri ljudeh. Rezultati raziskave so pokazali zelo podoben vedenjski vzorec letenju jat ptic, pri čemer se je izkazalo, da če bi 5% populacije spremenilo smer, bi ji ostala populacija (ne da bi se tega zavedala) sledila.

1.2 Računalniške simulacije

Zaradi lažjega razumevanja in bolj natančnega prikaza delovanja vedenja živih bitij, je bilo razvitih že veliko vrst računalniških simulacij in matematičnih modelov, ki simulirajo letenje v jatah. Eden izmed prvih je tako vedenje ptic v jatah simuliral Craig Reynolds [2, 3] .Leta 1986 ga je prikazal s svojo implementacijo programa *Boids*, ki vedenje enostavnih agentov imenovanih *boids*[1] prikazuje z gibanjem na bazi določenih osnovnih pravilih:

- ločevanje (angl. *separation*): preprečitev prevelikega približevanja ostalim agentom v skupini,
- poravnava (angl. *alignment*): poizkus usklajevanja hitrosti in/ali smeri najbližji skupini agentov,
- kohezija (angl. *cohesion*): poizkus zbliževanja najbližji skupini agentov.

Kot v večini simulacij umetnega življenja, so tudi v programu *Boids* prisotna pojavljajoča se vedenja (angl. *emergent*), katerih zapletenost presega vplive medsebojnih dejavnikov preprostih pravil. Čeprav smo se zaradi enostavne in elegantne rešitve bolj osredotočili na Reynoldsov pristop oziroma njegovo implementacijo, za katero obstaja odprto-kodno ogrodje OpenSteer[2], je prav tako zanimiv tudi pristop Franka Heppnerja in Ulfa Grenderja [5], ki sta leta 1990 predlagala drugačen model simulacije letenja ptic v jatah, in sicer na bazi ne-tako zelo drugačnih pravilih vedenja:

- domovanje (angl. *homing*): poizkus ostajanja v bližini skupne točke (doma),
- uravnavanje hitrosti (angl. *velocity regulation*): poizkus letenja v okviru predefinirane hitrosti,
- interakcija (angl. *interaction*): ob prevelikem približevanju razdruževanje; ob prevelikem razdruževanju ni vpliva na druge, drugače se poizkušajo združevati.

[1] Izraz naj bi bil okrajšava izraza "ptici podoben" (*angl. bird-like*) predmet oziroma "birdoid" - I. Lebar Bajec [4]

[2] Izvorna koda dostopna na (2012) http://opensteer.sourceforge.net/

1.3 Uporabnost in razširitve

Reynoldsov pristop je prikazal velik korak naprej v primerjavi s tradicionalnimi tehnikami, ki so se v 80ih letih prejšnjega stoletja uporabljali v filmski industriji oziroma natančneje pri vseh vrstah računalniških animacij. Prva animacija s tem modelom je bila ustvarjena leta 1987 z naslovom Stanley in Stella v *Breaking the Ice*[3]. Tej je leta 1992 sledil vrhunec s celovečernim filmom Tima Burtona *Batman Returns*, ki je vseboval računalniško ustvarjene roje netopirjev in vojsko pingvinov. Temu so sledili še mnogi, kot npr. leta 1994 Disneyjev risani film *The Lion King*, ki je vseboval črede divjih živali itd. Zaradi tega se "Boid" knjižnice v računalniški grafiki dandanes uporabljajo vse pogosteje v želji, da bi bile animacije vedenja živih bitij na pogled čim bolj realistične.

Osnovni Reynoldsov model je bil tudi že večkrat razširjen na več različnih načinov. Na primer Delgado-Mata [6] je razširil osnovni model in vključil upoštevanje posledic strahu, Hartman in Benes [7] sta predstavila komplementarno (angl. *complementary*) silo pri algoritmu za poravnavo, ki sta jo poimenovala zamenjava vodstva; ta sila omogoča možnost nekemu agentu, da postane vodilen in pri tem poizkuša pobegniti, itd.

1.4 Pregled naloge

V osnovi implementacije smo se osredotočili na Reynoldsov osnovni algoritem, pri čemer smo si tudi pomagali z deli Lebarja Bajca [4, 8]. Izdelana je bila razširitev osnovnega algoritma, ki je podrobneje predstavljen v drugem poglavju. V tretjem poglavju je tako predstavljen prilagojen model letečega agenta, vedenjske funkcije z upoštevanjem režimov delovanja, pravilno zaznavanje pristajalnega polja, algoritem izbiranja pozicije pristanka in najpomembneje algoritem prihajanja. V četrtem poglavju je izvedena analiza s prikazom rezultatov opravljenih eksperimentov. V zaključku so predstavljene možnosti izboljšav s smernicami nadaljnjega dela.

[3] Več dostopno na (2012) http://en.wikipedia.org/wiki/Stanley_and_Stella_in:_Breaking_the_Ice

2 OpenSteer

OpenSteer je odprto-kodni program napisan v programskem jeziku C++ z grafičnimi knjižnicami OpenGL-a. Implementacija omogoča nadgrajevanje že obstoječih vedenjskih animacij z novimi, ki se nato lahko uporabljajo v igrah ali v kakršnih koli naprednih simulacijah z veliko agenti. Ti agenti so lahko živa bitja (ljudje, živali, itd.), prevozna sredstva (avtomobili, letala, itd.) ali kar koli drugega s podobnimi "nagoni".

Zaradi možnosti nadgrajevanja in simulacij vedenj različnih živih bitij, je logika agenta razdeljena na dva dela, ki je v primeru simulacije letanja v jatah:

- osnovni model avtonomnega agenta (vsebuje algoritme potrebne za avtonomno gibanje po prostoru),
- model letečega agenta (vsebuje algoritme osnovnega modela avtonomnega agenta in algoritme potrebne za združevanje v jate).

Taka hierarhija omogoča, da ob dodajanju novih algoritmov in funkcij, ne vplivamo na logiko gibanja ali združevanja agentov. Že implementiranih je nekaj najpomembnejših vedenjskih funkcij, potrebnih za simulacijo vedenj različnih vrst agentov:

- tavanje (angl. *wander*) neurejeno premikanje,
- iskanje (angl. *seek*) približevanje določeni točki,
- bežanje (angl. *flee*) izogibanje določeni točki, kar je ravno nasprotje nagona iskanja,
- sledenje poti (angl. *path following*) sledenje poljubni pred-definirani poti,
- izogibanje oviram (angl. *obstacle avoidance*) izogibanje trku z nepremičnim objektom,
- izogibanje neurejenim tokom (angl. *unaligned collision avoidance*) izogibanje trku z naključno premikajočim se objektom s predčasnim predvidevanjem trka,
- ločevanje (angl. *separation*) odmikanje v skupini,
- poravnava (angl. *alignment*) usklajevanje smeri v skupini,
- kohezija (angl. *cohesion*) združevanje v skupini,
- zasledovanje (angl. *pursuit*) približevanje premikajoči se točki, kar je zelo podobno iskanju,
- izogibanje (angl. *evasion*) bežanje pred premikajočo se točko, kar je ravno nasprotje nagonu zasledovanja.

Izmed vseh implementiranih vedenjskih funkcij se ciljna usmeritev naloge osredotoča na ločevanje, poravnavo in kohezijo, katerih vsota tvori vedenjsko silo letenja v jatah.

2.1 Osnovni model avtonomnega agenta

Osnovni model avtonomnega agenta je le delček lastnosti, ki daje agentom možnost premikanja po navideznem prostoru in je definiran s parametri:

- maso (*število*): večinoma 1,
- radijem (*število*): večinoma 0,5,
- pozicijo (*vektor*): uvodoma naključno generirana,
- hitrostjo (*vektor*): uvodoma 2,7 (maksimalna hitrost $*$ 0,3),
- maksimalno silo (*število*): v primeru letenja 27,
- maksimalno hitrostjo (*število*): v primeru letenja 9,
- orientacijo (*enotski vektorji naprej* $\mathbf{f}$*, gor* $\mathbf{u}$ *in stranski* $\mathbf{s}$): uvodoma naključno generirana.

Premikanje po navideznem prostoru se računa z enačbami (2.1) s tem, da se v vsaki iteraciji izračuna pospešek $\mathbf{a}$, na podlagi želene sile $\mathbf{F}$ (vedenjske sile) in mase m, kar na koncu privede do nove hitrosti $\mathbf{v}'$ in pozicije $\mathbf{p}'$. Zaradi spremembe smeri in pozicije se

popravi tudi orientacija, tako da se s smerjo hitrosti spremeni enotski vektor naprej **f** in temu primerno tudi stranski enotski vektor **s** in enotski vektor gor **u**.

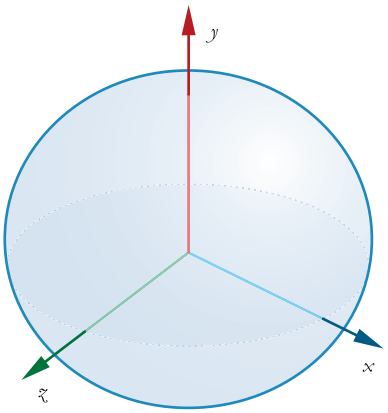
$$\begin{aligned} \mathbf{a} &= \frac{\mathbf{F}}{m} \\ \mathbf{v}' &= \mathbf{v} + \mathbf{a} * \Delta t \\ \mathbf{p}' &= \mathbf{p} + \mathbf{v}' * \Delta t \end{aligned} \tag{2.1}$$

2.2 Model letečega agenta

Model letečega agenta je enostavna nadgradnja osnovnega modela avtonomnega agenta. Kar pomeni, da poleg parametrov potrebnih za gibanje po prostoru, sedaj v vsakem koraku simulacije izlušči še "množico agentov" vidnih v omejenem vidnem polju. Ti so potrebni za izračun vedenjskih sil, ki agenta prisilijo k letenju v jati.

2.2.1 Zaznavanje okolja

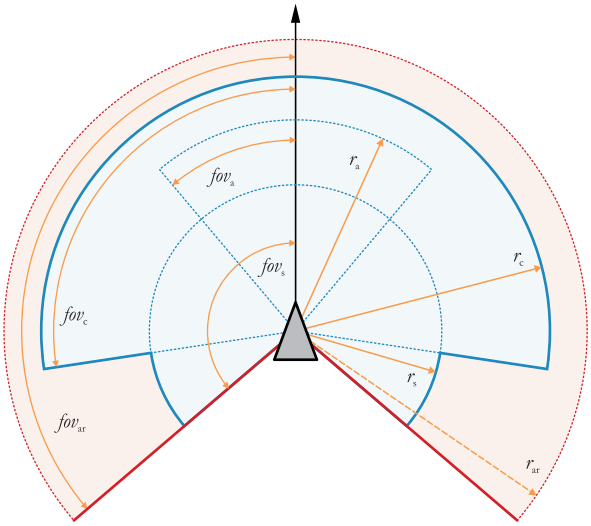
Samo okolje je definirano v obliki sfere, čigar središče je tudi središče t.i. navideznega sveta v katerem se gibljejo agenti. Taka omejitev v obliki sfere agentom onemogoči, da se porazgubijo po neskončnem prostoru ter jih prisili k vračanju nazaj proti središču navideznega sveta.



Slika 2.1 Prikaz sfere, ki obdaja koordinatni sistem navideznega sveta.

Zaznavanje okolja agenta je zaradi poenostavitve preiskovanja velike množice razdeljeno na dva dela:

- upoštevanje polja zaznavanja v obliki sfere (z največjim radijem izmed vseh vidnih polij za izračun posamezne vedenjske sile), ki obdaja agenta; videni agenti so zabeleženi v "množici agentov",
- pri računanju posamezne vedenjske sile, se "množico agentov" dodatno omeji z radijem (če je vidno polije za izračun vedenjske sile manjše) in površino, ki jo agent ne vidi, tako kot je prikazano na sliki 2.2.



Slika 2.2 Osnovni model zaznavanja okolice. Črna puščica prikazuje smer gibanja agenta, zasenčeno modro polje prikazuje polja zaznavanja ločevanja (r_s, fov_s), poravnave (r_a, fov_a) in kohezije (r_c, fov_c). Z rdečo je prikazano neomejeno polje prihajanja (r_{ar}, fov_{ar}).

2.3 Nagon letenja v jati

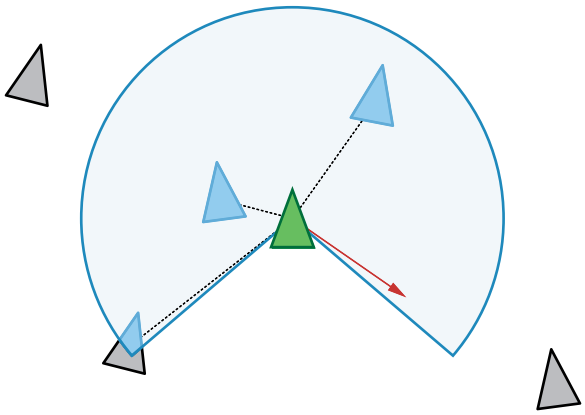
Za izračun nagona letenja v jati, so pomembne tri vedenjske funkcije: *ločevanje*, *kohezija* in *poravnava*, ki se odzivajo na večje skupine agentov in določijo kako bo agent reagiral glede odločitve sosednjih agentov v svoji okolici.

2.3.1 Ločevanje

Ločevanje (angl. *separation*) kot vedenje daje agentu možnost vzdrževanja določene varnostne razdalje do ostalih agentov. Tako obnašanje se lahko uporabi za preprečitev kolizij oziroma prevelikega prerivanja. Osnovno delovanje ločevanja, ki ga lahko podamo z enačbo

$$\mathbf{F_s} = \left[\frac{1}{n} \sum_{i=0}^{n} \frac{(\mathbf{p} - \mathbf{p_i})}{\|\mathbf{p} - \mathbf{p_i}\|} \right]^0, \tag{2.2}$$

je, da se najprej poišče vse najbližje agente. Za vsakega izmed teh agentov se izračunana odbijajoča (angl. *repulsive*) sila, z razliko med pozicijo $\mathbf{p}$ samega sebe in pozicijo posameznega agenta $\mathbf{p_i}$, ki se deli z razdaljo med njima. Vse te izračunane sile se tako seštejejo in delijo s številom agentov in nato pretvorijo v enotski vektor. Ločevanje je prikazano na sliki 2.3.



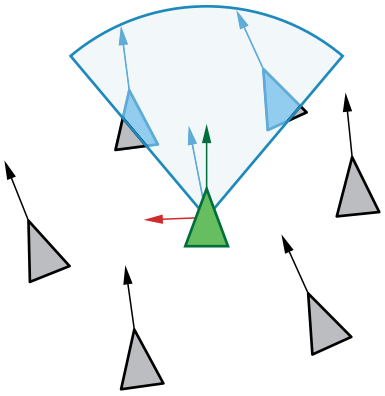
Slika 2.3 Prikaz delovanja osnovnega algoritma ločevanja. Zasenčeno modro polje predstavlja polje zaznavanja (zeleno obarvanega agenta), svetlo modri agenti pa predstavljajo del obravnavane "množice agentov". Rdeča puščica predstavlja vedenjsko silo ločevanja.

2.3.2 Poravnava

Poravnavanje (angl. *alignment*) kot vedenje daje agentu možnost poravnave (enako usmerjenost in/ali hitrost) z ostalimi bližnjimi agenti, kot je prikazano na sliki 2.4. Osnovno delovanje poravnave, ki ga lahko podamo z enačbo

$$\mathbf{F_a} = \left[\left(\frac{1}{n} \sum_{i=0}^{n} \mathbf{f_i} \right) - \mathbf{f} \right]^0, \tag{2.3}$$

je, da se najprej poišče vse najbližje agente in izračuna povprečje hitrosti najbližjih agentov $\mathbf{f_i}$. Vedenjska sila je enotski vektor razlike med tem povprečjem, ki je t.i. "želena hitrost" in hitrostjo agenta samega $\mathbf{f}$. Ta vedenjska sila potem poizkuša agenta usmerjati tako, da ga poravnava s svojimi sosedi.

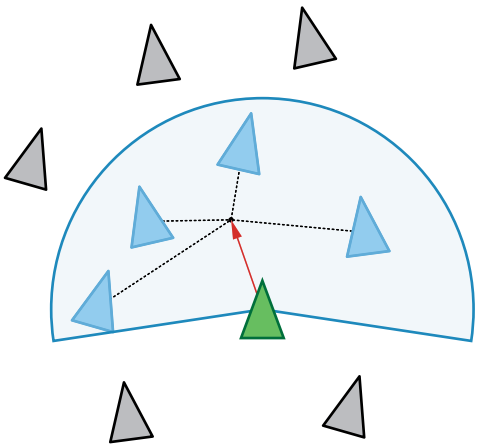


Slika 2.4 Prikaz delovanja osnovnega algoritma poravnavanja. Zasenčeno modro polje predstavlja polje zaznavanja (zeleno obarvanega agenta), svetlo modri agenti pa predstavljajo del obravnavane "množice agentov". Rdeča puščica predstavlja vedenjsko silo poravnave.

2.3.3 Kohezija

Kohezija (angl. *cohesion*) kot vedenje daje agentu možnost združevanja (približevanje in formiranje skupine) z ostalimi mimoidočimi agenti, kot je vidno na sliki 2.5. Osnovno delovanje kohezije, ki ga lahko podamo z enačbo

$$\mathbf{F_c} = \left[\left(\frac{1}{n} \sum_{i=0}^{n} \mathbf{p_i} \right) - \mathbf{p} \right]^0 , \tag{2.4}$$



Slika 2.5 Prikaz delovanja osnovnega algoritma združevanja oziroma kohezije. Zasenčeno modro polje predstavlja polje zaznavanja (zeleno obarvanega agenta), svetlo modri agenti pa predstavljajo del obravnavane "množice agentov". Rdeča puščica predstavlja vedenjsko silo kohezije.

je, da se najprej poišče vse najbližje agente in izračuna povprečno pozicijo najbližjih sosedov $\mathbf{p_i}$. Vedenjska sila je usmerjena v smer povprečne pozicije (enotski vektor razlike med pozicijo $\mathbf{p}$ samega sebe s povprečno pozicijo).

2.4 Združevanje nagonov

Ključnega pomena je, da so vedenjske sile (oziroma nagoni) le delčki, ki pripomorejo k izgradnji bolj kompleksnega vzorca vedenj. Ker vedenjska sila vsaka zase, ne podaja nič več kot le željo po spremembi smeri, je potrebno poiskati take, ki najbolje opisujejo vedenje našega agenta. Za doseg čim bolj naravnega obnašanja, se jih združi v eno samo. Najbolj enostavna združitev, ki se uporablja, je preprost izračun vsote posameznih uteženih vedenjskih sil, kjer uteži prikazujejo pomembnost posamezne sile. Taka združitev deluje dobro, ampak ima ključno slabost, da se lahko vedenjske sile kljub prilagoditvi uteži začnejo med seboj izničevati ravno v neprimernih trenutkih.

3 Od ideje do realizacije

Za nalogo smo si zastavili algoritem, ki bi simuliral pristajanje in vzletanje skupine avtonomnih agentov (ti v našem primeru predstavljajo model jate ptic). Pristop k problemu smo razdelili na več faz, ki se lahko tudi brez večjega problema enostavno združijo v celoto. V našem primeru model vedenja jate ptic pri pristajanju in vzletanju s področja počivanja.

Problem smo razdelili na naslednje faze:

- prilagoditev osnovne logike agenta (stanja, želje, razdraženost, ...),
- izboljšava vedenjskih funkcij (pravilno delovanje v primerih različnih možnih stanj agenta),
- zaznavanje pristajalnega polja (upoštevanje mrtvega kota),
- izbiranje pristajalne pozicije, končne točke,
- implementacija vedenja prihajanja (angl. *arrival*).

Pri izdelavi naloge smo se odločili, da bomo agente v 2D prostoru prikazali s trikotniki,

v 3D prostoru pa s piramido, kot je to običaj v večini primerov take vrste simulacijskih algoritmov.

3.1 Prilagojen model letečega agenta

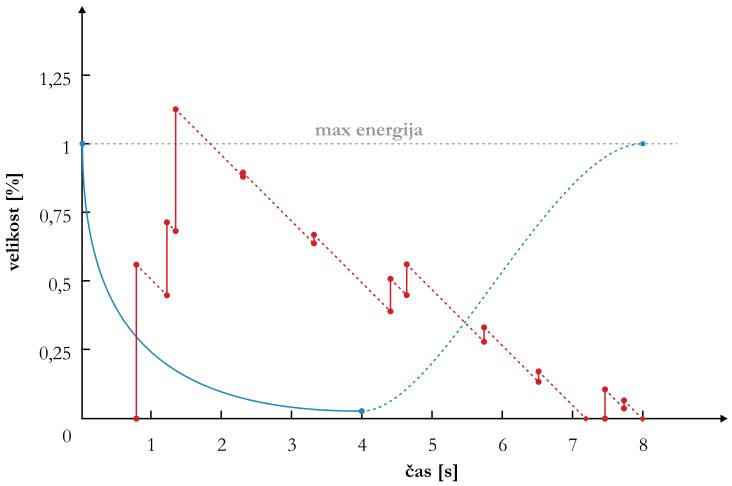
Osnovnemu modelu agenta smo dodali parameter, ki beleži razpoložljivo energijo e. Ta določa sposobnost agenta za opravljanje svojega trenutnega dela. Določili smo tudi parameter, ki beleži razdraženost a. Ta preprečuje nastanek prevelike gneče med agenti, ko ti počivajo. V času letenja se energija zmanjšuje in v času počivanja povečuje, a obenem se ob prerivanju povečuje tudi razdraženost. Zato smo temu primerno definirali spreminjanje energije in sicer, s pomočjo hitrosti premikanja agenta $||\mathbf{v}||$ (dolžina vektorja hitrosti), maksimalne možne dovoljene hitrosti $v_{\max}$ in razdraženosti a, v odvisnosti s časom Δt.

$$\begin{aligned} e\uparrow &= e + (e * 0{,}05 + \frac{a}{2}) * \Delta t \\ e\downarrow &= e - e * \left(\frac{||\mathbf{v}||}{v_{\max}}\right)^{v_{\max}} * \Delta t \end{aligned} \tag{3.1}$$

V nasprotju s spreminjanjem energije, se razdraženost povečuje v odvisnosti od pojavitve hitrosti $||\mathbf{v}||$ (ob sunkovitem premiku), velikosti vedenjske sile $||\mathbf{F}||$ in na podlagi trenutne energije e (zainteresiranost k večjemu ali manjšemu razdraženju).

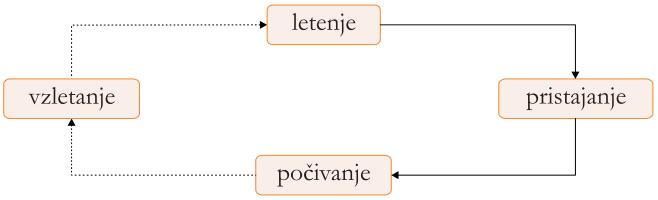
$$\begin{aligned} a\uparrow &= a + ||\mathbf{v}|| * ||\mathbf{F}|| * e * \Delta t \\ a\downarrow &= a - a * \Delta t \end{aligned} \tag{3.2}$$

Oba parametra sta omejena na svojo maksimalno velikost, kar pove ali si je agent dokončno opomogel oziroma v primeru razdraženosti, da ima dovolj prerivanja in temu primerno reagira. V našem primeru smo maksimum energije nastavili na 1 tako, da se spreminja v intervalu $(0, 1]$. Maksimalna razdraženost v nasprotju z maksimumom energije je v našem primeru 5 oziroma se ga po potrebi spremeni, kar pri velikem številu agentov v navideznem svetu, privede do spreminjanja tolerance povzročiteljev razdraženosti. Dinamično spreminjanje energije in razdraženosti je prikazana na sliki 3.1.



Slika 3.1 Prikaz spreminjanja energije in razdraženosti v odvisnosti s časom. Modra neprekinjena črta prikazuje zmanjševanje energije, prekinjena črta pa povečevanje, pri čemer nikoli ne preide intervala $(0, 1]$. Z rdečo črto je zaradi boljše preglednosti prikazano spreminjanje razdraženosti v intervalu $[0, \infty)$.

Prilagojen agent deluje v treh (štirih) režimih delovanja, kot je prikazano na sliki 3.2. Slednje omogoča bolj pregledno delovanje algoritma oziroma združevanja že obstoječih segmentov, kot so npr. teženj, katerih končen rezultat je letenje v jati.



Slika 3.2 Prikaz prehodov med različnimi režimi delovanja. Prekinjena črta prikazuje le trenutni režim delovanja, v katerem se agent ne zadržuje.

Prehod med režimi delovanja opredeljujejo tri enačbe, ki opisujejo agentovo željo po letenju D_f, željo po pristajanju D_l in željo po počivanju D_r:

$$\begin{aligned} D_\mathrm{f} &= D_\mathrm{f} + \Delta t * \left(e - F_\mathrm{tL} + \left(0,2 * \frac{F}{N} \right) \right) \\ D_\mathrm{l} &= D_\mathrm{l} + \Delta t * \left(F_\mathrm{tL} - e + \left(0,2 * \frac{R+L}{N} \right) \right) \\ D_\mathrm{r} &= D_\mathrm{r} + \Delta t * \left(S_\mathrm{r} * (1 - e/R_\mathrm{tF}) + \left(0,2 * \frac{R}{N} \right) \right) \end{aligned} \tag{3.3}$$

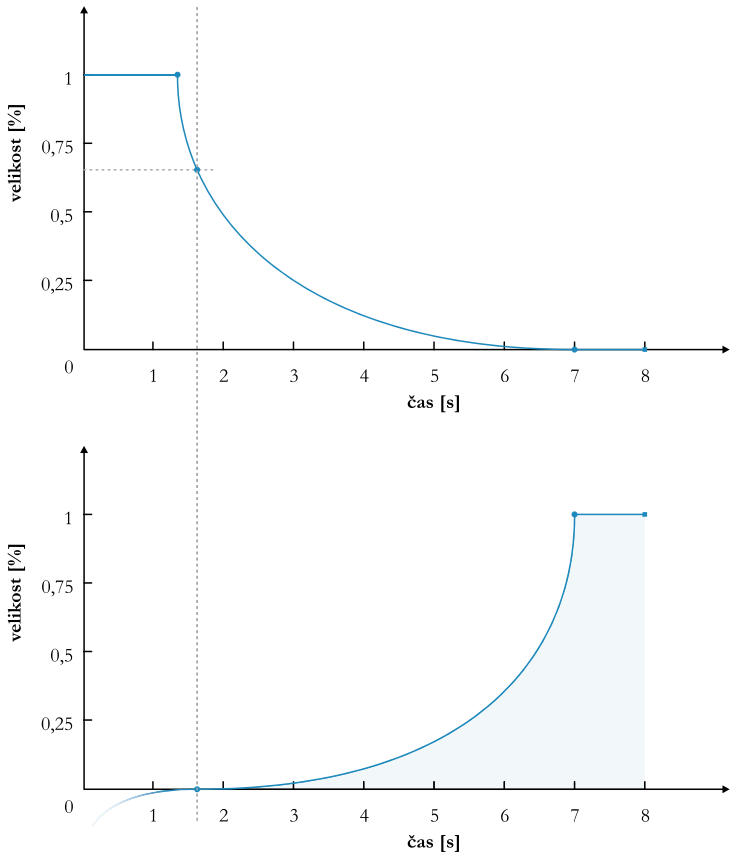
kjer e predstavlja trenutno energijo agenta, F število sosednjih agentov v režimu delovanja letenje, L število sosednjih agentov v režimu delovanja pristajanje, R število sosednjih agentov v režimu delovanja počivanje in N število vseh sosednjih agentov. F_{tL} in R_{tF} sta fiksna parametra, ki predstavljata začetek prehajanja iz stanja letenja v stanje pristajanja in začetek prehajanja iz stanja počivanja v stanje letenja (v našem primeru je $F_{\mathrm{tL}} = 0{,}65$ in $R_{\mathrm{tF}} = 0{,}85$). Pri izračunu želje po počivanju je pomemben parameter S_{r}, ki predstavlja pogojni parameter režima počivanja, kateri ob počivanju postane 1, v nasprotnem primeru 0. Rezultati vseh treh enačb so omejeni na interval $[0, 1]$, tako da imajo vsi parametri enak minimum/maksimum oziroma so enakovredni in lažje primerljivi.

Režim delovanja se določi v odvisnosti od izračunanih želja agenta po prehodu v določen režim delovanja. Algoritem je izpisan v zaporedju *če-potem-sicer* (angl. *if-then-else*) stavkov in je temelj modela pristajanja in vzletanja agentov s počivalne površine:

če *stanje* **je** *letenje* **potem**
  če $D_{\mathrm{f}} < F_{\mathrm{tL}}$ **in** razdraženost $= 0$ **potem**
    če naključno_št$[0, 1) < 1 - (D_{\mathrm{f}}/F_{\mathrm{tL}})$ **potem** stanje $\rightarrow$ pristajanje
sicer če *stanje* **je** *pristajanje* **potem**
  izbira_točke_počivanja
  če izbrana_točka_počivanja **in** hitrost $< \epsilon$ **potem** stanje $\rightarrow$ počivanje
sicer če *stanje* **je** *počivanje* **potem**
  če $(D_{\mathrm{f}} = 1$ **in** $D_{\mathrm{r}} = 0)$ **ali** razdraženost > 5 **potem**
    stanje $\rightarrow$ letenje
    $\mathbf{v}'' \rightarrow f * v_{\max} * 0{,}3$
  sicer potem $\mathbf{v}' \rightarrow proj_{\mathbf{v}'}$

Natančneje delovanje prehoda iz režima letenja v režim pristajanja omogoča agentu letenje dlje časa, kljub izpolnjenim pogojem za pristanek, kot je prikazano na sliki 3.3. Prehod iz režima pristajanja v režim počivanja nastopi, ko ima agent izbrano pristajalno točko in hitrost manjšo od ϵ (pomeni da se je agent ustavil). V nasprotnem primeru se v vsaki iteraciji znova poizkuša najti nova oziroma boljša točka počivanja. Za prehod nazaj v režim letenja mora biti (a) želja po letenju D_{f} maksimalna, želja po počivanju D_{r} pa minimalna; (b) razdraženost večja od maksimalne dovoljene. Zaradi velike verjetnosti,

da agent pade s pristajalne površine, je potrebno tudi preveriti ali je na njej, drugače prav tako preide v režim letenja. V nasprotnem primeru se izračunava nova hitrost $\mathbf{v}''$, ki je projekcija želene hitrosti $\mathbf{v}'$ na pristajalno površino $\mathbf{D}$ (smer pristajalne površine), kar omogoči da se ob razdraženosti agenti premikajo zgolj vzdolž pristajalne površine.



Slika 3.3 Prikaz grafa spreminjanja želje po letenju D_{f} (zgoraj) in grafa $1 - (D_{\mathrm{f}}/F_{\mathrm{tL}})$ (spodaj) za zavlačevanje prehoda v naslednji režim. Črtkana črta predstavlja $D_{\mathrm{f}} < F_{\mathrm{tL}}$, zatemnjeni del grafa $1 - (D_{\mathrm{f}}/F_{\mathrm{tL}})$, pa predstavlja polje iskanja naključnega_št[0,1). Ob pojavitvi pravilnega števila, agentu dovolimo prehod v naslednje stanje.

3.2 Vedenjske funkcije z upoštevanjem režima delovanja

Logika vedenjskih funkcij *ločevanje*, *kohezija* in *poravnava* v osnovnem delovanju ostaja nespremenjena. Vse tri se še vedno odzivajo na večje skupine agentov in določijo kako se bo agent odzval na odločitve agentov v svoji okolici.

Za prikaz želenega vedenja agentov, smo v našem primeru sili poravnave in kohe-

zije priredili, da se osnoven izračun razdeli na režime delovanja posameznih sosedov in upošteva željo po vedenju v tem režimu. Vedenjska sila ločevanja ostane nespremenjena, tako da omogoča preprečitev trkov med agenti ne glede na režim delovanja. V režimu počivanja oziroma v nasprotju z režimom letenja in pristajanja smo upoštevali radij zaznavanja sosedov v velikosti radija agenta, kar pripomore k temu, da se v času počivanja, na pristajalni površini agenti ob prerivanju ustavijo tesno eden zraven drugega.

Priredba vedenja poravnavanja z upoštevanjem stanj:

$$\begin{aligned}
\mathbf{F_{a_f}} &= \left[\left(\frac{1}{n_f}\sum_{i=0}^{n}\mathbf{f_i} * D_{\mathrm{f}}\right) - \mathbf{f}\right]^0 \\
\mathbf{F_{a_l}} &= \left[\left(\frac{1}{n_l}\sum_{i=0}^{n}\mathbf{f_i} * D_{\mathrm{l}}\right) - \mathbf{f}\right]^0 \\
\mathbf{F_{a_r}} &= \left[\left(\frac{1}{n_r}\sum_{i=0}^{n}\mathbf{f_i} * D_{\mathrm{r}}\right) - \mathbf{f}\right]^0 \\
\mathbf{F_a} &= \mathbf{F_{a_f}} + \mathbf{F_{a_l}} + \mathbf{F_{a_r}}
\end{aligned} \tag{3.4}$$

Priredba vedenja kohezije z upoštevanjem stanj:

$$\begin{aligned}
\mathbf{F_{c_f}} &= \left[\left(\frac{1}{n_f}\sum_{i=0}^{n}\mathbf{p_i} * D_{\mathrm{f}}\right) - \mathbf{p}\right]^0 \\
\mathbf{F_{c_l}} &= \left[\left(\frac{1}{n_l}\sum_{i=0}^{n}\mathbf{p_i} * D_{\mathrm{l}}\right) - \mathbf{p}\right]^0 \\
\mathbf{F_{c_r}} &= \left[\left(\frac{1}{n_r}\sum_{i=0}^{n}\mathbf{p_i} * D_{\mathrm{r}}\right) - \mathbf{p}\right]^0 \\
\mathbf{F_c} &= \mathbf{F_{c_f}} + \mathbf{F_{c_l}} + \mathbf{F_{c_r}}
\end{aligned} \tag{3.5}$$

S prirejenimi vedenjskimi silami smo našemu agentu tako omogočili čim bolj pravilno vedenje pri prehodih med režimi delovanja. Zaradi problematike prehajanja k zgoščevanju in prerivanju je potrebno v režimu počivanja in pristajanja vsoto teh sil prilagoditi tako, da to preprečimo:

- pri počivanju je potrebno vektor nove smeri, ki predstavlja tudi želeno hitrost projicirati na pristajalno površino in poskrbeti, da se agenti gibljejo v tej novi smeri,
- pri pristajanju je potrebno končno vsoto vedenjskih sil zmanjševati tako, da bolj kot se agent približuje končni točki, manj ga zanimajo ostali agenti.

3.3 Zaznavanje pristajalnega polja

Pristajalno polje smo v našem primeru definirali kot navadno daljico (začetno točko in končno točko). Ta je lahko poljubno postavljena kjerkoli v navideznem svetu. Za lažjo izdelavo algoritma zaznavanja pristajalne površine smo si problem razdelili na dveh fazi:

- pravilno zaznavanje intervala točk ob videnju celotne pristajalne površine (s predpostavko, da se mrtev kot nahaja v smeri $-\mathbf{f}$),
- pravilno upoštevanje površine mrtvega kota (namesto piramidne oblike, uporabimo obliko stožca).

Definicija 3.3.1. *Sférni* ali *krógelni koordinátni sistém*[a] je krivočrtni sistem koordinat v trirazsežnem prostoru, s pomočjo katerega enolično določimo lego točk na krogli ali sferoidu. Za določanje lege točke v prostoru vedno potrebujemo tri koordinate.

Koordinate s točko zapišemo na naslednji način: (r, θ, φ), kjer

- krajevni vektor r zavzame poljubno veliko vrednost,
- polarni kot θ zavzame vrednosti med 0 in π (med 0 in 180°),
- azimuth φ zavzame vrednosti med 0 in 2π (med 0 in 360°), merjeno v nasprotni smeri od gibanja urinega kazalca.

Pretvorba iz kartezičnih koordinat v sferne:

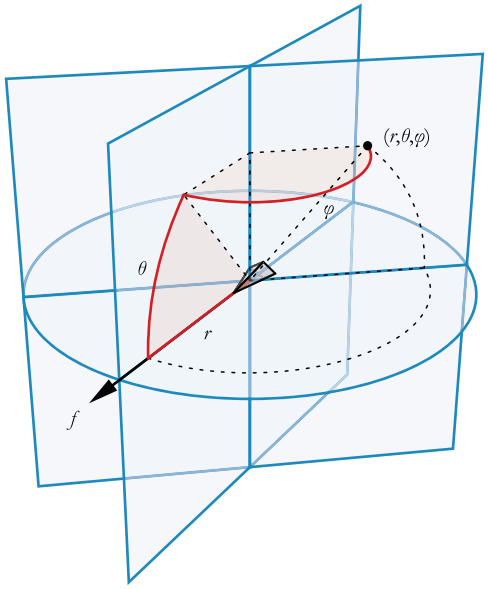
$$r = \sqrt{x^2 + y^2 + z^2}, \tag{3.6}$$

$$\theta = \arctan\left(\frac{y}{x}\right), \tag{3.7}$$

$$\varphi = \arccos\left(\frac{z}{r}\right). \tag{3.8}$$

[a] Dostopno na (2012) http://en.wikipedia.org/wiki/Spherical_coordinate_system

Zaradi potrebe po poznavanju, kje v našem navideznem svetu se nahaja določena točka (glede pozicije in usmeritve agenta), smo si pomagali s sfernim koordinatnim sistemom.



Slika 3.4 Zaznavanje točke v sfernih koordinatah, kjer je prikazana projicirana točka v koordinatni sistem agenta, pri čemer r predstavlja razdaljo do točke, θ polarni kot in φ azimut.

Osnovno enačbo za izračun azimut kota (3.8) smo priredili zato, da agent lažje loči med pogledi levo, desno, zgoraj in spodaj. Prirejena enačba sedaj omogoča, da ko agent vidi točko z leve strani zabeleži negativen kot, z desne pa pozitiven, tako kot je prikazano z enačbo

$$\varphi = \begin{cases} \arccos\left(\frac{z}{r}\right), & x < 0 \\ -\arccos\left(\frac{z}{r}\right), & \text{drugače} \end{cases} \tag{3.9}$$

kjer je x abscisa ne-pretvorjene točke (x, y, z).

3.3.1 Osnovno zaznavanje pristajalne površine

Pri osnovnem zaznavanju pristajalne površine je predvideno, da agent vidi pristajalno površino ne glede na oddaljenost od nje (dolžinsko neomejeno vidno polje). Ključnega pomena je izračun začetne točke **vS** in končne točke **vE**. Ti dve točki po vrsti predstavljata projekcijo začetne in končne točke pristajalne površine, v koordinatni sistem

agenta in od tod v sferni koordinatni sistem. Na podlagi teh lahko ugotovimo usmeritev oziroma lego agenta v prostoru glede pristajalne površine. Delovanje je razdeljeno na dva dela:

- pogled spredaj ali zadaj (odvisno od velikosti vidnega polja fov_{ar}),
- bočni pogled, ki je razdeljen na pogled desno, levo (pojavi se lahko tudi možnost zaznavanja celotne pristajalne površine).

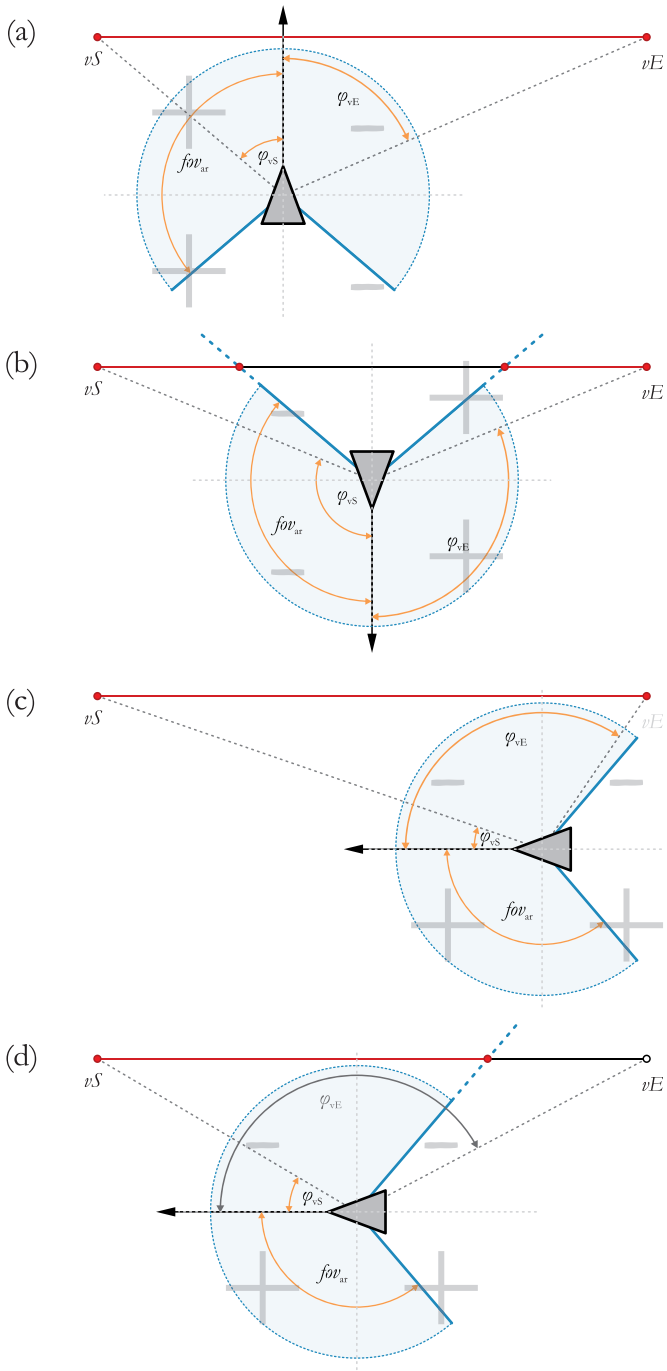
Delovanje se lahko tudi prikaže s pomočjo *če-potem-sicer* stavkov:

začetna točka pristajalne površine ← začetna točka daljice
končna točka pristajalne površine ← končna točka daljice

če ($\varphi_{\mathrm{vS}} > 0$ **in** $\varphi_{\mathrm{vE}} < 0$) **ali** ($\varphi_{\mathrm{vS}} < 0$ **in** $\varphi_{\mathrm{vE}} > 0$) **potem**
  če $|\varphi_{\mathrm{vS}}| < |fov_{\mathrm{ar}}|$ **potem**
    množica točk ← *začetna točka pristajalne površine*
  množica točk ← *izračun_točk_mrtvega_kota*
  če $|\varphi_{\mathrm{vE}}| < |fov_{\mathrm{ar}}|$ **potem**
    množica točk ← *končna točka pristajalne površine*
sicer potem
  če $\varphi_{\mathrm{vS}} > |fov_{\mathrm{ar}}|$ **in** $\varphi_{\mathrm{vE}} < |fov_{\mathrm{ar}}|$ **potem**
    množica točk ← *izračun_točk_mrtvega_kota*
    množica točk ← *končna točka pristajalne površine*
  sicer če $\varphi_{\mathrm{vS}} < |fov_{\mathrm{ar}}|$ **in** $\varphi_{\mathrm{vE}} > |fov_{\mathrm{ar}}|$ **potem**
    množica točk ← *začetna točka pristajalne površine*
    množica točk ← *izračun_točk_mrtvega_kota*
  sicer $\varphi_{\mathrm{vS}} < |fov_{\mathrm{ar}}|$ **in** $\varphi_{\mathrm{vE}} < |fov_{\mathrm{ar}}|$ **potem**
    množica točk ← *začetna točka pristajalne površine*
    množica točk ← *končna točka pristajalne površine*

Ker nobeno živo bitje ni zmožno videti v neskončnost, smo zaznavanja neskončnega vidnega polja razširili tako, da se radij zaznavanja lahko poljubno spremi (dolžinsko omejeno vidno polje). Pomagali smo si z enostavno implementacijo izračuna presečišča daljice in sfere[1], pri čemer nam izračun poda novo začetno in končno točko pristajalne površine, ki se prav tako pretvorita v koordinatni sistem agenta in od tod v sferni koordinatni

[1] Dostopno na (2012) http://en.wikipedia.org/wiki/Line%E2%80%93sphere_intersection



Slika 3.5 Prikazani so štirje pogledi na pristajalno površino (s sferno začetno **vS** in končno **vE** točko). Črna puščica pomeni smer agenta, rdeča črta prikazuje viden segment in siv križ s prekinjenimi črtami (središče v agentu) prikazuje kvadrante predznakov kotov sfernih koordinat φ_{vS} in φ_{vE}. Neodvisno od predznaka je prikazana tudi velikost vidnega polja fov_{ar}. (a) pogled spredaj, (b) pogled zadaj, (c) pogled bočno z desne strani, ko je viden celoten segment in (d) pogled bočno z desne strani

sistem. Za izračuna presečišča je potrebno vedeti parametre enačbe daljice (pristajalne površine) $\mathbf{X}(t) = \mathbf{P}+t\mathbf{D}$, $t \in \mathbb{R}$, kjer je $\mathbf{P}$ začetna točka daljice, $\mathbf{D}$ smerni vektor daljice in t dolžina daljice. Sfero okoli agenta, ki predstavlja vidno polje definirata pozicija agenta $\mathbf{p}$ (središče sfere) in radij r (omejeno vidno polje) tako, da lahko presečišče izračunamo z enostavnim zaporedjem enačb:

$$\begin{aligned} \mathbf{d_f} &= \mathbf{P} - \mathbf{p}, \\ t_0 &= \mathbf{d_f} \cdot \mathbf{d_f} - r^2, \\ t_1 &= \mathbf{D} \cdot \mathbf{d_f}, \\ d &= {t_1}^2 - t_0, \end{aligned} \tag{3.10}$$

kjer $\mathbf{d_f}$ predstavlja smerni vektor in ob enem tudi dolžino od pozicije agenta $\mathbf{p}$ do začetne točke pristajalne površine $\mathbf{P}$, t_0 predstavlja razliko med ne-korenjeno dolžino smernega vektorja $\mathbf{d_f}$ in kvadriranim radijem sfere, ter t_1 predstavlja razdaljo projekcije med enotskim smernim vektorjem daljice $\mathbf{D}$ in smernim vektorjem $\mathbf{d_f}$. Za ugotovitev presečišča izračunamo pogojni parameter d, s katerim si lahko postavimo tri vprašanja, ki nam povedo:

- ali ni presečišča ($d < 0$),
- ali je presečišče z eno točko ($d = 0$),
- ali je z dvema točkama ($d > 0$).

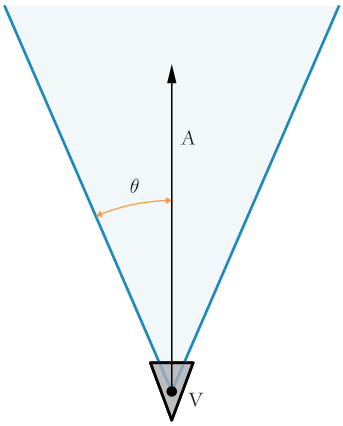
V našem primeru smo pri izračunu vidnega dela pristajalnega polja upoštevali le primer presečišča z dvema točkama, ki se izračuna z enačbo $\mathbf{I_{p_{1,2}}} = \mathbf{P} + (-t_1 \mp \sqrt{d}) * \mathbf{D}$ (potrebno je poskrbeti, da se točki nahajata na daljici) in primer, ko ni presečišča (agent ne vidi pristajalnega polja). Za upoštevanjem dolžinsko omejenega vidnega polja smo tako zamenjali prvi dve vrstici prej opisanega algoritma:

izračun_presečišča_sfere_in_daljice
začetna točka pristajalne površine $\leftarrow \mathbf{I_{p_1}}$
končna točka pristajalne površine $\leftarrow \mathbf{I_{p_2}}$

3.3.2 Upoštevanje mrtvega kota

Osnovna logika algoritma zaznavanja vidne površine zaradi svoje enostavne implementacije delovanja predvideva mrtev kot v obliki piramide, ki se enostavno prikaže v 2D

prostoru in zato zaradi ostrih robov vizualno ni primeren v 3D prostoru. Ta problem smo elegantno rešili s pomočjo algoritma Davida Eberlya [9], ki izračuna presečišče med daljico in stožcem. Za pravilno delovanje algoritma je ponovno potrebno vedeti parametre pristajalne površine, ki je definirana z enačbo daljice (pristajalna površina) $\mathbf{X}(t) = \mathbf{P} + t\mathbf{D}$, $t \in \mathbb{R}$, kjer je $\mathbf{P}$ začetna točka daljice, $\mathbf{D}$ smerni vektor daljice in t njena dolžina. Za mrtev kot, ki je definiran v obliki stožca pa vrhnjo točko $\mathbf{p}$ (pozicijo agenta), smerni vektor $\mathbf{A}$ (npr. predstavlja inverzni enotski smerni vektor naprej $-\mathbf{f}$) in kot θ med smernim vektorjem in zunanjo stranico, kot je prikazano na sliki 3.6. V implementaciji se upošteva, da je stožec akuten[2]. S pravilnim upoštevanjem kota θ in smernega vektorja stožca $\mathbf{A}$ je možno doseči dinamično spremenljivo velikost mrtvega kota v intervalu $[0, \pi]$. Z enostavnim pogojem, če je $\theta > \pi/2$ pravilno popravimo smerni vektor stožca $\mathbf{A}$, tako da je stožec vedno akuten.



Slika 3.6 Prikazan je prerezan stožec v 2D prostoru, kjer je zatemnjen prostor njegova notranjost. V našem primeru predstavlja mrtev kot agenta, ki se v tem primeru nahaja za njim.

Za pravilno delovanje je potrebno izračunati zaporedje enačb:

$$\begin{aligned} A_{\mathrm{D}} &= \mathbf{A} \cdot \mathbf{D}, \\ \mathbf{E} &= \mathbf{P} - \mathbf{p}, \\ A_{\mathrm{E}} &= \mathbf{A} \cdot \mathbf{E}, \\ D_{\mathrm{E}} &= \mathbf{D} \cdot \mathbf{E}, \\ E_{\mathrm{E}} &= \mathbf{E} \cdot \mathbf{E}, \end{aligned} \tag{3.11}$$

[2]Stožec je akuten, ko je kot $\theta \in (0, \pi/2)$.

kjer A_{D} predstavlja kot med enotskim smernim vektorjem stožca **A** in daljice **D**. **E** predstavlja smerni vektor in ob enem tudi dolžino od pozicije agenta do začetne točke pristajalne površine. Parametra A_{E} in D_{E} predstavljata razdaljo projekcije med enotskim smernim vektorjem (stožca **A** in daljice **D**) in vektorjem **E**. Za ugotovitev presečišča je potrebno izračunati pogojne parametre c_0, c_1 in c_2 z enačbami

$$\begin{aligned} c_2 &= {A_{\mathrm{D}}}^2 - (\cos\alpha)^2, \\ c_1 &= A_{\mathrm{D}} * A_{\mathrm{E}} - (\cos\alpha)^2 * D_{\mathrm{E}}, \\ c_0 &= {A_{\mathrm{E}}}^2 - (\cos\alpha)^2 * E_{\mathrm{E}}. \end{aligned} \tag{3.12}$$

Osnovna logika implementacije omogoča ugotovitev štirih različnih situacij presečišča:

- presečišče z dvema točkama (segment),
- presečišče z eno točko (žarek),
- presečišče z eno točko (točka),
- brez presečišča.

V našem primeru smo pri izračunu točk upoštevali tako rezultate presečišča z dvema točkama, kot tudi presečišče z eno točko kot žarek (predvidevamo, da agent vidi eno od mejnih točk pristajalne površine) in primer brez presečišča. Kompleksnost izračuna presečišča stožca je v primerjavi s sfero bolj kompleksna, zato je delovanje predstavljeno s *če-potem-sicer* stavki (implementacija upošteva možnost, da množica točk ni prazna):

če $|c_2| \geq \epsilon$ **potem**
  $d = {c_1}^2 - c_0 * c_2$
  če $d < 0$ **potem** *ni presečišča*
  drugače če $d > \epsilon$ **potem**
    $\mathbf{I_p} \leftarrow \mathbf{P} + ((-c_1 + \sqrt{d}) * \frac{1}{c_2}) * \mathbf{D}$
    če $(\mathbf{I_p} - \mathbf{p}) \cdot \mathbf{A} > 0$ **in** $\mathbf{I_p} \in \mathbf{X}(t)$ **potem** *množica točk* $\leftarrow \mathbf{I_p}$
    $\mathbf{I_p} \leftarrow \mathbf{P} + ((-c_1 - \sqrt{d}) * \frac{1}{c_2}) * \mathbf{D}$
    če $(\mathbf{I_p} - \mathbf{p}) \cdot \mathbf{A} > 0$ **in** $\mathbf{I_p} \in \mathbf{X}(t)$ **potem** *množica točk* $\leftarrow \mathbf{I_p}$
drugače če $|c_1| \geq \epsilon$ **potem**
  $\mathbf{I_p} \leftarrow \mathbf{P} + (\frac{c_0}{2c_1}) * \mathbf{D}$
  če $(\mathbf{I_p} - \mathbf{p}) \cdot \mathbf{A} > 0$ **potem** *množica točk* $\leftarrow \mathbf{I_p}$

drugače če $|c_0| \geq \epsilon$ **potem** *ni presečišča*
drugače potem *množica točk* $\leftarrow \mathbf{p}$

Natančneje izračun presečišča stožca z daljico deluje v treh nivojih, razporejenih po možnosti pridobitve čim več rešitev, s parametri c_2, c_1 in c_0. V primeru ko je c_2 dovolj velik, je v tem nivoju možno pridobiti vse štiri načine presekov, zato se izračuna podpogojni parameter d. V našem primeru preverimo samo, če je $d < 0$ in izračunamo ali je presečišče z dvema točkama ali presečišče z eno točko (žarek) $\mathbf{I_{p_{1,2}}}$. V primeru ko je c_1 dovolj velik, je v tem nivoju možno pridobiti dve rešitvi, ki sta: (a) presečišče z eno točko (žarek) $\mathbf{I_p}$ in (b) brez presečišča. Zato v našem primeru ko je razdalja projekcije, vektorja med pozicijo in točko kandidata za presek $\mathbf{I_p}$ na smerni vektor stožca $\mathbf{A}$ večja od nič, pomeni da je v tej točki prišlo do presečišča (žarek). V zadnjem primeru ko je $c_0 = 0$ in $c_0 < \epsilon$, pridobimo presečišče z eno točko (žarek), ki je vrh stožca oziroma v našem primeru pozicija agenta $\mathbf{I_p} = \mathbf{p}$.

Združitev osnovnega zaznavanja vidne površine z upoštevanjem mrtvega kota, omogoča možnost pridobitve enega ali dveh segmentov pristajalne površine (odvisnih od pogleda in usmeritve agenta) določenih z množico maksimalno štirih točk $\mathbf{t_1} \ldots \mathbf{t_4}$.

3.4 Izbiranje pozicije pristanka

Pri izbiri pozicije oziroma točke pristanka smo skušali agenta prisiliti k pravokotni poravnavi na pristajalno površino, kar mu omogoči zaznavanje sosedov na levi in na desni strani. Postopek smo razdelili na štiri dele:

1. izbira smeri in/ali najbolj vidni del pristajalne površine,
2. prilagoditev vidnega dela pristajalne površine (če se v segmentu vidnega polja nahaja projekcija pozicije agenta, se segment zmanjša),
3. izbira naključne točke v prilagojenem vidnem delu pristajalne površine,
4. ob že izbrani točki, iskanje boljše točke, ki zagotavlja hitrejši pristanek.

3.4.1 Izbira smeri pristajanja

Izbiro smeri pristajanja oziroma najbolj vidnega dela pristajalne površine (segmenta) smo dosegli s pomočjo mejnih točk pristajalnega polja pretvorjenih v sferične koordinate. Izbiro najbolj optimalne, najmanj potratne smeri lahko zapišemo s pogojno enačbo

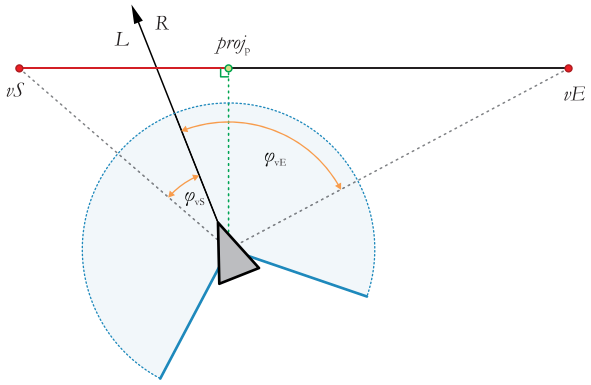
$$L_{\mathrm{oR}} = \begin{cases} 1, & |\varphi_{\mathrm{vS}}| < |\varphi_{\mathrm{vE}}| \\ 0, & \text{drugače} \end{cases}, \tag{3.13}$$

pri čemer 1 pomeni DESNO in 0 pomeni LEVO. Situacijo, ko agent vidi celotno pristajalno polje prikazuje slika 3.7. Situacijo, ko vidi dva segmenta za seboj pa prikazuje slika 3.8.

3.4.2 Prilagoditev velikosti vidnega dela pristajalne površine

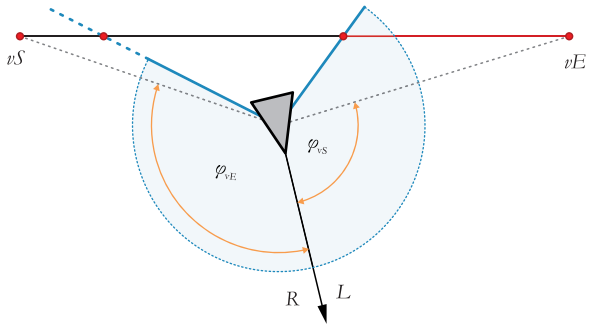
Celoten segment kot tak, je neprimeren pri zaznavanju agenta v primeru, ko vidi pristajalno površino direktno pred seboj, kar prikazuje tudi slika 3.7. Zato segment zmanjšamo tako, da s pomočjo ugotovljene smeri pristajanja zamenjamo eno od mejnih točk vidnega polja s točko projekcije agenta $proj_{\mathbf{p}}$ na pristajalno površino oziroma natančneje:

- če si agent izbere smer pristajanja LEVO, zamenjamo mejno točko intervala **vE**,
- če si agent izbere smer pristajanja DESNO, zamenjamo mejno točko intervala **vS**.



Slika 3.7 Prikazano je dolžinsko neomejeno vidno polje agenta obrnjenega proti pristajalni površini omejeni s točko **vS** in točko **vE**, kjer pretvorba v sferne koordinate vsebuje kota φ_{vS} in φ_{vE}. Ob upoštevanju pogojne enčbe (3.13) si agent v tem primeru izbere svojo LEVO oziroma L smer, ki je tudi najbolj optimalna.

Izvedena prilagoditev segmenta, kot je vidna na sliki 3.7 ni dokončna oziroma optimalna, ker v osnovi še vedno omogoča, da si agent lahko začetno točko izbere veliko bolj stran od najkrajše poti (ciljna točka $proj_{\mathbf{p}}$) oziroma optimalne poti (ciljna točka, ki nastane ob sekanju enotskega vektorja agenta naprej, $\mathbf{f}$, in pristajalne površine). Optimalno točko poiščemo s pomočjo presečišča daljice (pristajalne površine) in površine



Slika 3.8 Prikazano je dolžinsko neomejeno vidno polje agenta obrnjenega stran od pristajalne površine omejene s točko **vS** in **vE**, kjer pretvorba v sferne koordinate vsebuje kota φ_{vS} in φ_{vE}. Ob upoštevanju pogojne enčbe (3.13) si agent v tem primeru izbere svojo LEVO oziroma L smer, ki je tudi najbolj optimalna.

(definira usmeritev agenta v prostoru; pogled agenta naprej **f** in gor **u**)[3]. Za pravilen izračun potrebujemo parametre enačbe $\mathbf{X}(t) = \mathbf{P} + t\mathbf{D}$, $t \in \mathbb{R}$ (pristajalna površina), kjer je **P** začetna točka daljice, **D** smerni vektor daljice in t njena dolžina. Površina je definirana z normalo **s** (v našem primeru stranski enotski smerni vektor agenta) in točko na njej **p** (pozicija agenta)

$$\begin{aligned} D_{\mathrm{dN}} &= \mathbf{D} \cdot \mathbf{s}, \\ D_{\mathrm{s}} &= (\mathbf{P} - \mathbf{p}) \cdot \mathbf{s}, \end{aligned} \tag{3.14}$$

kjer izračunan parameter D_{dN} pomeni kot med smernim vektorjem daljice in stranskim enotskim vektorjem agenta, D_{s} pomeni razdaljo projekcije vektorja $(\mathbf{P}-\mathbf{p})$ (smerni vektor z dolžino od pozicije agenta **p** do začetne točke pristajalne površine **P**) na stranski enotski vektor agenta.

Izračun enačb tako omogoča, da se lahko vprašamo:

- ali je točkovno presečišče $|D_{\mathrm{dN}}| > 0$,
- ali pa je daljica paralelna na površino $D_{\mathrm{s}} <= 0$.

V našem primeru smo pri izračunu upoštevali le primer točkovnega presečišča, ki ob potrjenem pogoju s pomočjo enačbe $\mathbf{I_p} = \mathbf{P} + (-D_{\mathrm{s}}/D_{\mathrm{dN}}) * \mathbf{D}$ izračuna presečiščno točko, ki se lahko uporabi za zadnjo prilagoditev segmenta vidnega polja. Segment se zmanjša tako, da s pomočjo ugotovljene smeri pristajanja zamenjamo eno od mejnih točk

[3] Dostopno na (2012) http://en.wikipedia.org/wiki/Line-plane_intersection

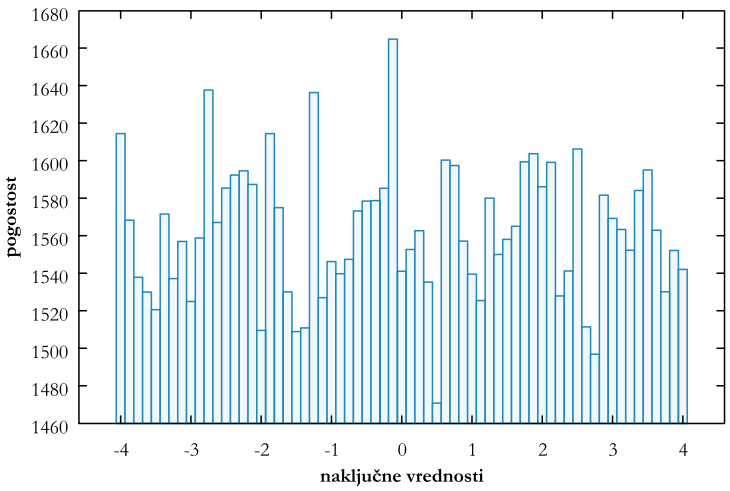
z novo izračunano točko $\mathbf{I_p}$ samo, če se ta nahaja v že prilagojenem segmentu oziroma natančneje:

- če si agent izbere smer pristajanja LEVO, zamenjamo mejno točko **vS**,
- če si agent izbere smer pristajanja DESNO, zamenjamo mejno točko **vE**.

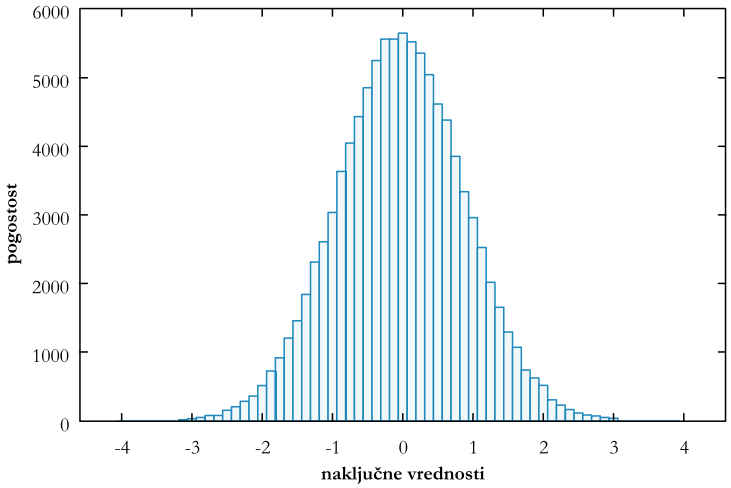
3.4.3 Izbira točke v prilagojenem vidnem delu pristajalne površine

Ko smo izbrali pravilno smer in/ali poleg tega še segment, smo na koncu logiko priredili tako, da si agent dokler ne prispe do cilja izbira novo bližjo oziroma boljšo točko, kar na koncu privede do pravilne usmerite pri ustavitvi. Ob pridobitvi novega segmenta, začetno točko izberemo z generatorjem naključnih števil, kar agentu omogoči bolj dinamično odločitev glede svoje končne točke (nepredvidljiva končna pozicija). Implementacija v osnovi ni zelo zahtevna, ker so generatorji naključnih števil na voljo v vseh standardnih matematičnih knjižnicah, pri čemer osnovna implementacija deluje v intervalu $[0, 1)$. A delovanje osnovnega generatorja deluje po principu naključne verjetnosti preko celotnega intervala (glej sliko 3.9). Slednje posledično povzroči precejšnje skakanje izbrane končne točke. Za izbiro nove točke smo zato uporabili drugačen generator naključnih števil, ki porazdeljuje števila s pogostostjo v obliki Gaussove krivulje[4], čigar delovanje je prikazano na sliki 3.10. Z njegovo izbiro je vsaka naslednja naključna točka, z veliko verjetnostjo blizu trenutni.

[4]Dostopno na (2012) http://www.taygeta.com/random/gaussian.html



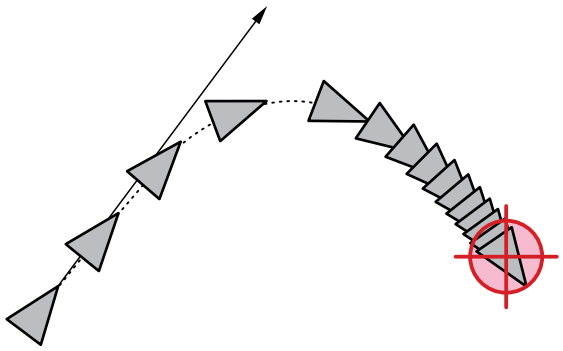
Slika 3.9 Prikazana je pogostost dobljenih naključnih števil standardnega generatorja, pri 100.000 naključno generiranih številih v intervalu $[-4, 4)$.



Slika 3.10 Prikazana je pogostost dobljenih naključnih števil Gaussovega generatorja, pri 100.000 naključno generiranih številih v intervalu (-4,4).

3.5 Algoritem prihajanja

Implementacija vedenja prihajanja je v osnovni logiki zelo podobna vedenju iskanja (angl. *seek*). Razlika je predvsem v tem, da se agent ne premika le z maksimalno hitrostjo proti svojemu cilju ampak, ko se mu dovolj približa, začne ustavljati in se na cilju tudi dokončno ustavi, kot je prikazano na sliki 3.11. Zato je razdalja na kateri se agent začne ustavljati zelo pomemben parameter, ki prepreči, da bi agent prekoračil točko in se nenehno poizkušal vračati in jo zadeti.



Slika 3.11 Prikaz delovanja algoritma vedenja prihajanja kjer je vidno, da agent poizkuša najti čim hitrejšo pot do svojega cilja.

Delovanje algoritma je sledeče:

1. poišči razdaljo med svojo pozicijo in ciljno točko,
2. poišči željeno hitrost (ravna linija med pozicijo in ciljno točko),
3. če agent ni znotraj radija ustavljanja pospešuj,
4. če je agent znotraj radija ustavljanja začni zavirati do ustavitve,
5. vrednosti vzdržuj v mejah maksimalne dovoljene.

Logika algoritma kot taka, je primerna npr. za simulacijo bejzbolskega igralca, ki se ob teku ustavi na bazi; ali vožnje avtomobila proti križišču, ki se ustavi pred rdečo lučjo.

Zaradi prevelike agresivnosti algoritma prihajanja smo se odločili poiskati boljšo rešitev. Pri pregledu več možnih rešitev[5,6] smo ugotovili, da v grobem vse delujejo na zelo podoben princip pospeševanja do cilja, kar v določenih primerih lahko privede tudi do prekoračitve ciljne točke. Zaradi tega smo se odločili izvesti priredbo osnovnega algoritma in sicer tako, da smo pri zgoraj opisanem delovanju zanemarili točko tri in dosegli, da agent potuje do svojega cilja z nespremenjeno hitrostjo in ko prispe do območja zaviranja se temu primerno tudi ustavi. Izkaže se, da je pri algoritmih, ki agresivno pospešujejo do svojega cilja potrebno upoštevati radij ustavljanja in utež sile, ki sta dovolj velika, to je vsaj $r = 3{,}5$ in $w = 8{,}5$, zato da se omogoči ustavitev. Za prirejen algoritem, ki ne uporablja pospeševanja lahko uporabimo manjši radij ustavljanja in utež sile, ki sta vsaj $r = 1{,}3$ in $w = 3{,}5$. Zaradi manjšega radija ustavljanja in ravno prav velike uteži se agent lahko primerno in ne prehitro ustavi na svojem cilju.

[5]Dostopno na (2012) http://rocketmandevelopment.com/2010/06/18/steering-behaviors-arrival/
[6]Dostopno na (2012) http://www.shiffman.net/teaching/nature/steering/

4 Analiza delovanja

Implementacija algoritma za način delovanja vsebuje veliko parametrov, ki omogočijo prilagajanje. Slednje lahko privede simulacijo do zelo različnih zaključkov. Z analizo smo skušali ugotoviti ali vidno polje vpliva na način pristajanja oziroma ali lahko omejitev pogleda privede k temu, da pristane veliko manj agentov (ki ne letijo v jatah).

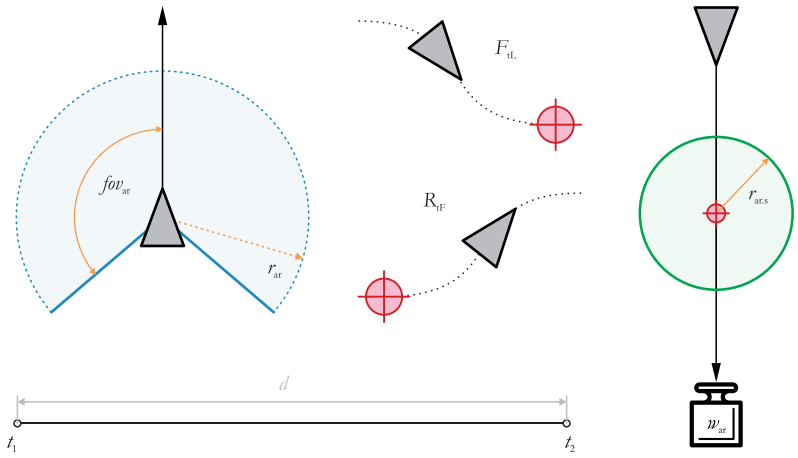
Analizo delovanja smo razdelili na tri dele:

- prikaz normalnega delovanja (vsi agenti lahko preidejo k počivanju),
- prikaz mejnega delovanja (toliko kot premore pristajalna površina),
- prikaz delovanja v ekstremnih pogojih (vidno nemogoč pristanek vseh agentov k počitku).

Spremenljivi parametri:

- pozicija mejnih točk ($\mathbf{t_1}$ in $\mathbf{t_2}$) pristajalne površine v prostoru (s postavitvijo definira dolžino d oziroma velikost),
- vidno polje oziroma upoštevanje mrtvega kota fov_{ar} in radijem r_{ar},
- faktor pričetka pridobivanja želje po pristajanju F_{tL},

- faktor pričetka pridobivanja želje po letenju L_{tF},
- velikost radija ustavitve $r_{\mathrm{ar.s}}$,
- velikost uteži oziroma privlačnost ciljne točke w_{ar}.
- število agentov,
- maksimum razdraženost $a_{\max}$.



Slika 4.1 Vizualen prikaz uporabe spremenljivih parametrov.

Pred začetkom izvajanja smo se odločili, da postavimo v navidezni svet pristajalno površino z začetno točko $\mathbf{t_1}(-25, 0, 25)$ in končno točko $\mathbf{t_2}(25, 0, -25)$, katere velikost (dolžina) je 70,710 7. Kar pomeni, da maksimalno premore približno 70 agentov z radijem 0,5 oziroma premerom 1. Kot vidnega polja zaznavanja pristajalne površine nastavimo identično kotu vidnega polja izračuna vedenjske sile ločevanja (glej sliko 2.2). V primeru omejenega vidnega polja, radij nastavimo enak, kot je pri vidnem polju izračuna vedenjske sile kohezije. Analizo delovanja v vseh primerih izvajamo maksimalno do 5 minut oziroma pridobitve maksimuma energije pri vseh agentih. Nastavljenih parametrov tekom celotne analize ne spreminjamo zaradi lažje primerjave le teh.

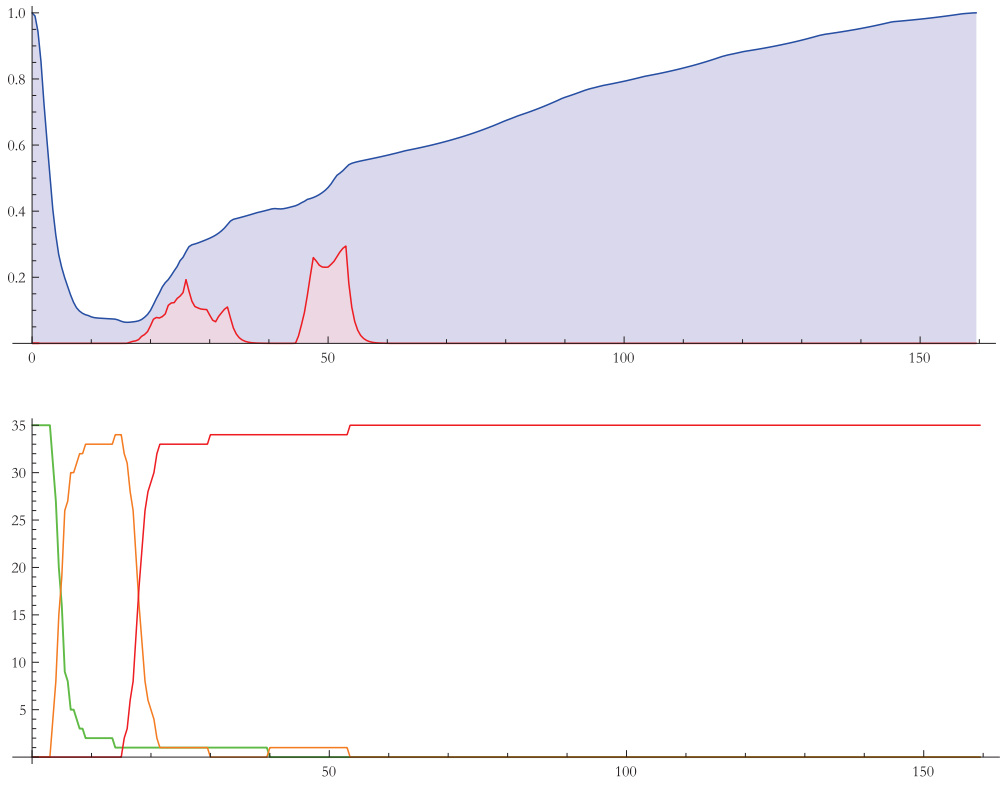
4.1 Analiza delovanja v normalnih pogojih

Analizo delovanja algoritma v primeru z normalnimi pogoji pristajanja smo izvedli tako, da se je v svet naključno pozicioniralo in usmerilo 35 agentov, kar ob zastavljeni velikosti pristajalne površine omogoča več kot dovolj manevrskega prostora (približno polovico

manj, kot premore pristajalna površina). Pri tem smo se osredotočili na opazovanje spreminjanja energije v odvisnosti od razdraženosti. Zaradi želje po boljšem razumevanju le tega, smo beležili tudi spreminjanje režimov delovanja v odvisnosti števila agentov. Pri tem smo prav tako tekom simulacije beležili kolikšno je povprečno zanimanje za določeno mesto pristajalne površine (v korakih velikosti radija agenta oziroma 0,5) v odvisnosti od povprečnega števila agentov.

4.1.1 Upoštevanje dolžinsko neomejenega vidnega polja

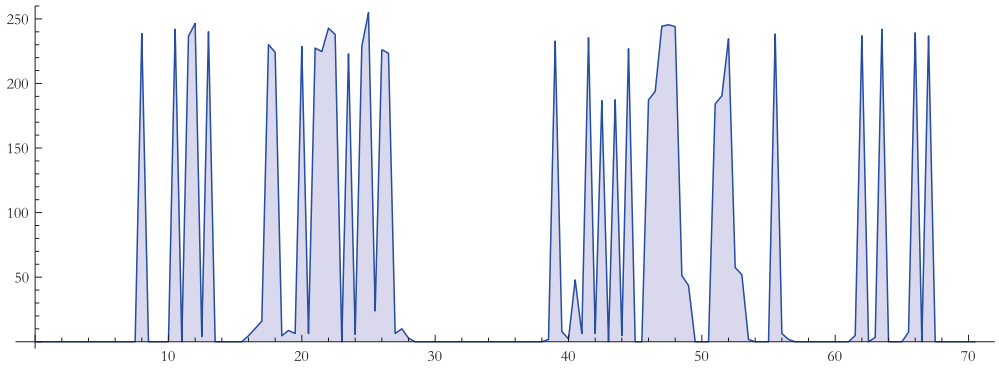
Iz rezultatov je viden idealen pristanek vseh agentov, ki se ob pristanku umirijo in preidejo k počivanju (energija naraste do maksimuma). Ob pristanku oziroma postavitvi agentov je prisotno zelo malo razdraženosti. To je možno zaradi skupinske usmeritve, ki agente privede do tega, da se pravilno med seboj poravnajo, še preden pristanejo k počitku.



Slika 4.2 Zgoraj je prikazano spreminjanje energije (modra) in razdraženosti (rdeča) skozi čas. Spodaj je prikazano spreminjanje režima delovanja (letenje - zelena, pristajanje - oranžna, počivanje - rdeča) v odvisnosti števila agentov skozi čas.

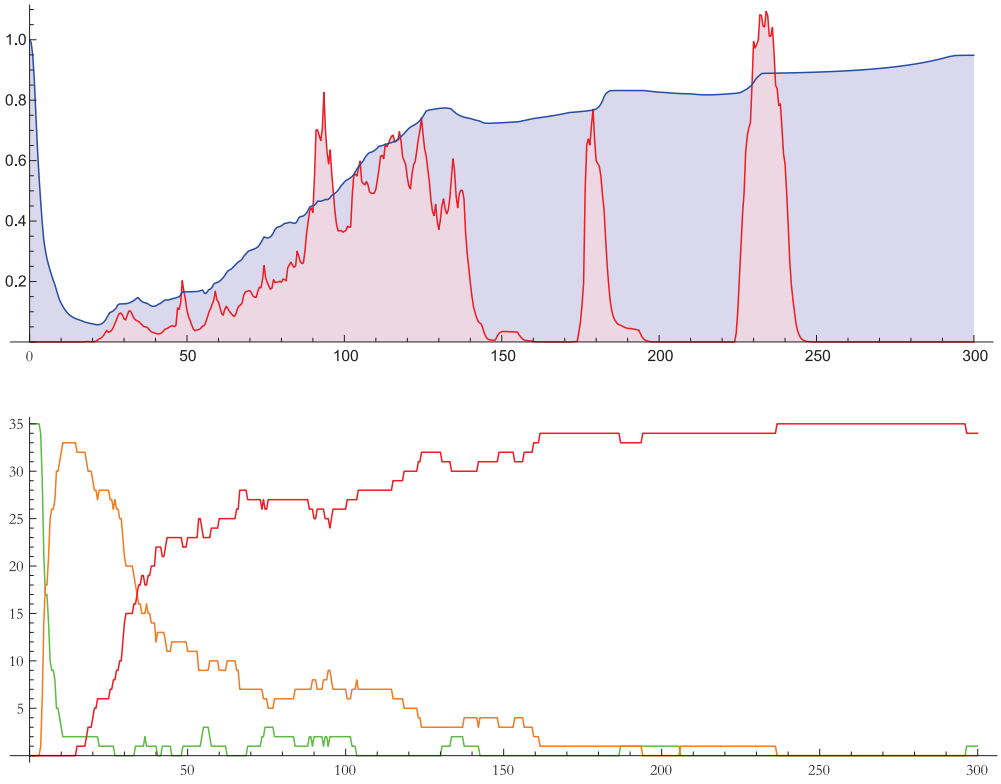
Zaradi hitre ustalitve, premeščanj v tem primeru ni prisotnih. Zato so najbolj želene

pozicije zabeležene kar s prvo točko, ki si jo agent izbere (točka na kateri počiva).



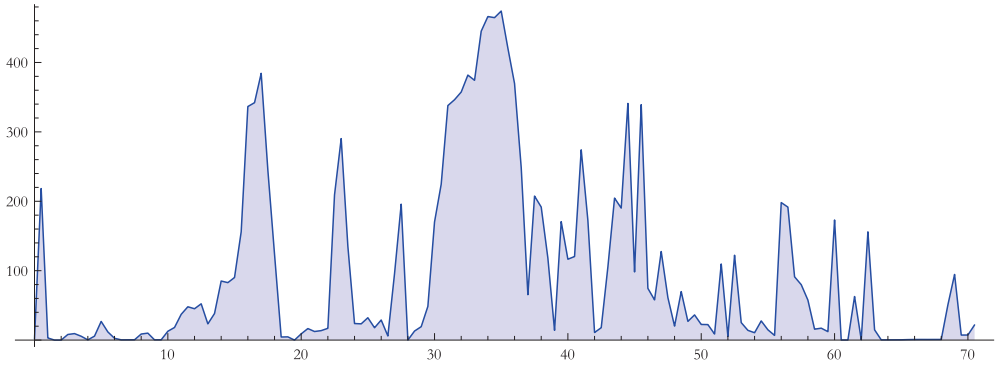
Slika 4.3 Prikaz verjetnosti pojavitve agentov na pristajalni površini (dolžina pristajalne površine v odvisnosti števila agentov).

4.1.2 Upoštevanje dolžinsko omejenega vidnega polja



Slika 4.4 Zgoraj je prikazano spreminjanje energije (modra) in razdraženosti (rdeča) skozi čas. Spodaj je prikazano spreminjanje režima delovanja (letenje - zelena, pristajanje - oranžna, počivanje - rdeča) v odvisnosti števila agentov skozi čas.

Ob upoštevanju dolžinsko omejenega vidnega polja je pristajanje in vzletanje zelo dinamično. Zaradi pogostega vrivanja (tesno postavljeni agenti imajo zelo malo možnosti za ponovno preureditev) je prisotno veliko razdraženosti, kar privede k temu, da veliko agentov vzleti in težko ponovno najde pristajalno površino.



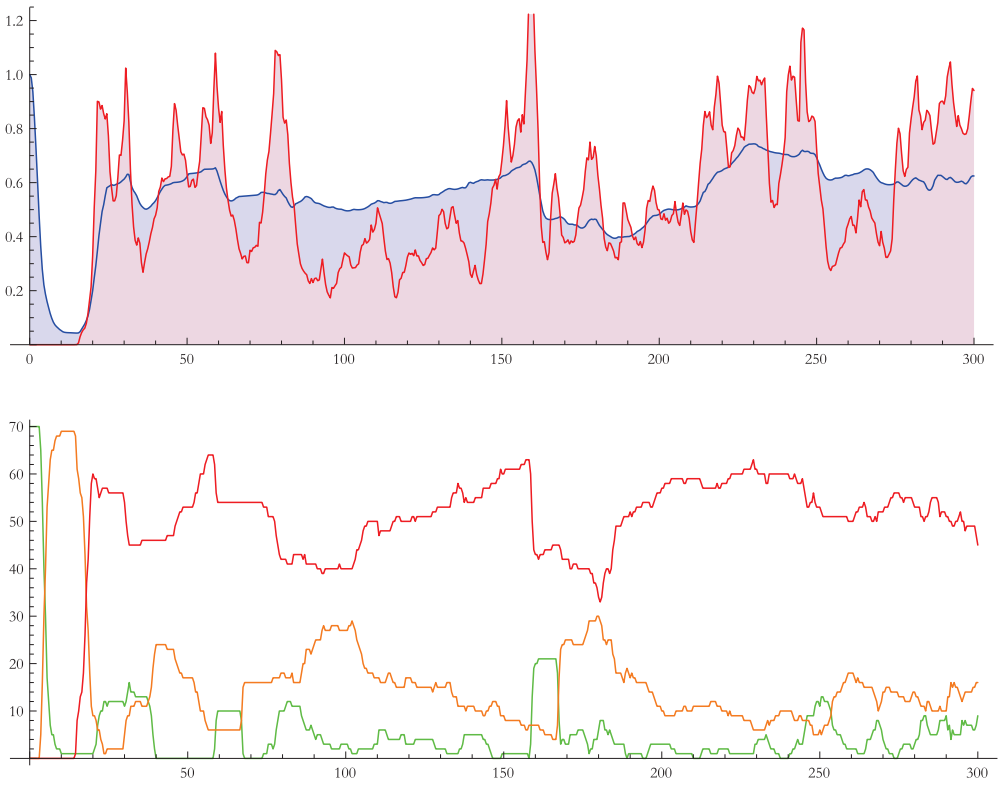
Slika 4.5 Prikaz verjetnosti pojavitve agentov na pristajalni površini (dolžina pristajalne površine v odvisnosti števila agentov).

4.2 Analiza delovanja v mejnih pogojih

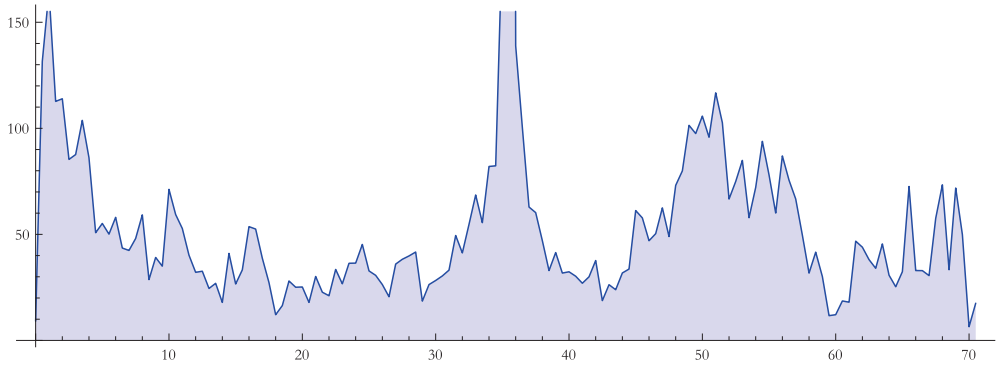
Analizo delovanja algoritma v primeru z normalnimi pogoji pristajanja smo izvedli tako, da se je v svet naključno pozicioniralo in usmerilo 70 agentov, kar ob zastavljeni velikosti pristajalne površine omogoča zelo tesno postavitev. Pri tem smo se osredotočili na opazovanje spreminjanja energije v odvisnosti od razdraženosti. Zaradi želje po boljšem razumevanju le tega, smo beležili tudi spreminjanje režimov delovanja v odvisnosti števila agentov. Pri tem smo prav tako tekom simulacije beležili kolikšno je povprečno zanimanje za določeno mesto pristajalne površine (v korakih velikosti radija agenta oziroma 0,5) v odvisnosti od povprečnega števila agentov.

4.2.1 Upoštevanje dolžinsko neomejenega vidnega polja

Z dobljenih rezultatov se vidi, da kljub razdraženosti pristane in počiva vsaj 85% agentov. Med vsemi upoštevanimi (tako tistimi, ki na novo pristanejo kot tistimi, ki že počivajo) se večina zadržuje dlje časa ob straneh pristajalne površine, če se zanemari pojav (odstopanje v sredini), ki nastane ob avtomatiziranem vračanju agentov nazaj oziroma proti središču sveta 2.1.



Slika 4.6 Zgoraj je prikazano spreminjanje energije (modra) in razdraženosti (rdeča) skozi čas. Spodaj je prikazano spreminjanje režima delovanja (letenje - zelena, pristajanje - oranžna, počivanje - rdeča) v odvisnosti števila agentov skozi čas.

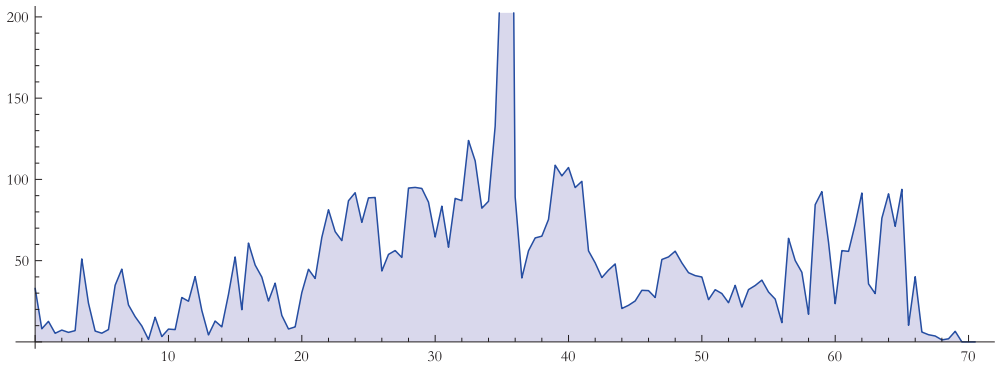


Slika 4.7 Prikaz verjetnosti pojavitve agentov na pristajalni površini (dolžina pristajalne površine v odvisnosti števila agentov).

4.2.2 Upoštevanje dolžinsko omejenega vidnega polja

V nasprotju z upoštevanjem dolžinsko neomejenega vidnega polja analiza prikazuje daljše letenje po prostoru, čeprav večina agentov prestopi v režim pristajanja (omogočeno dodatno združevanje v jate). Prav tako ob prvem videnju pristajalne površine agenti zaradi omejenega pogleda postopoma in v manjših skupinah prehajajo k počivanju. Anomalija zaradi vračanja v središče navideznega sveta je še vedno prisotna z razliko v tem, da se agenti bolj orientirajo na druge v svoji bližini, kar privede še do dodatnega, večjega, pristajanja v središče pristajalne površine.

Slika 4.8 Zgoraj je prikazano spreminjanja energije (modra) in razdraženosti (rdeča) skozi čas. Spodaj je prikazano spreminjanje režima delovanja (letenje - zelena, pristajanje - oranžna, počivanje - rdeča) v odvisnosti števila agentov skozi čas.

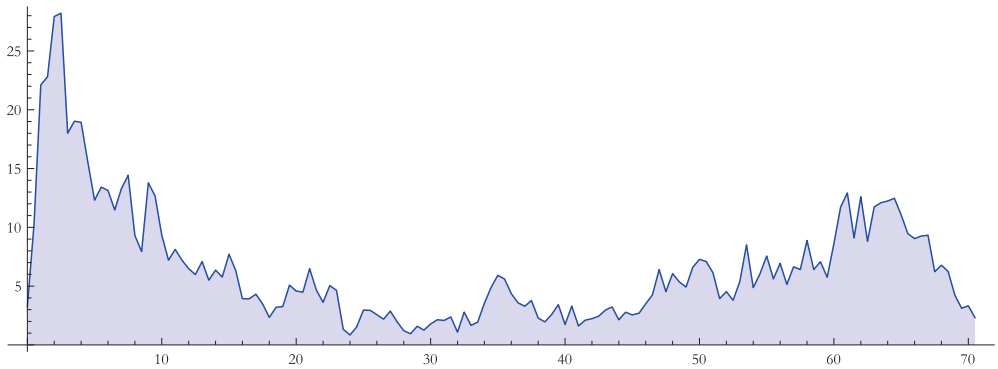


Slika 4.9 Prikaz verjetnosti pojavitve agentov na pristajalni površini (dolžina pristajalne površine v odvisnosti števila agentov).

4.3 Analiza delovanja v ekstremnih pogojih

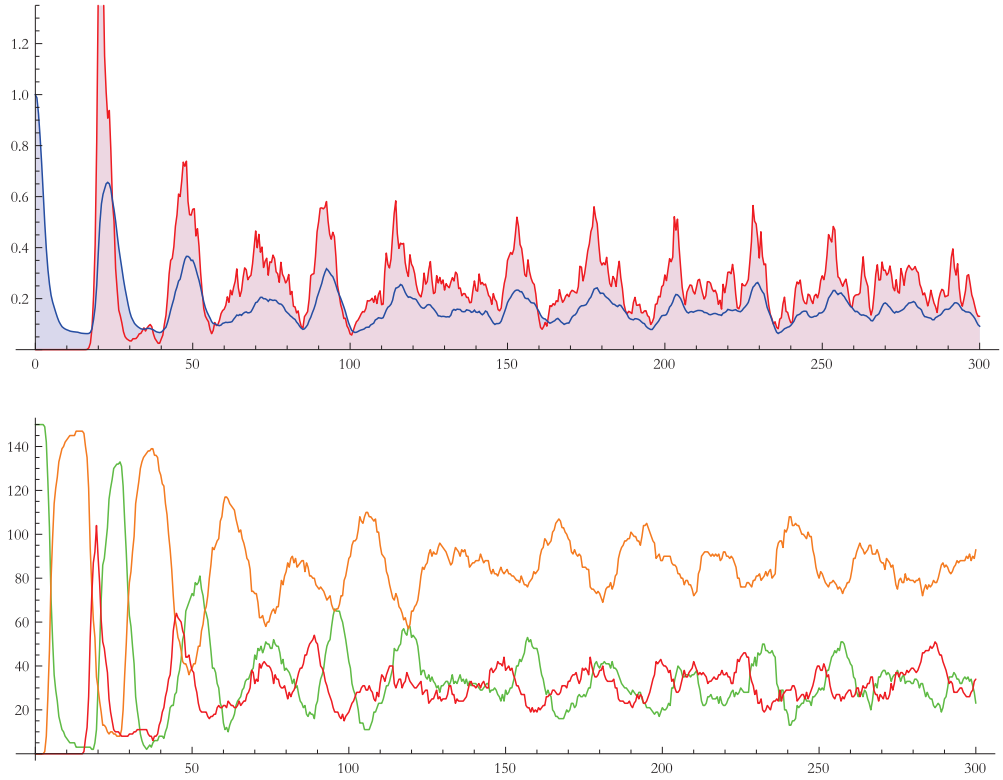
Analizo delovanja algoritma v ekstremnih oziroma vidno nemogočih pogojih smo izvedli tako, da se je v svet naključno pozicioniralo in usmerilo 150 agentov, kar ob zastavljeni velikosti pristajalne površine pomeni približno 80 agentov več kolikor jih premore. Osredotočili smo se na opazovanje spreminjanja energije v odvisnosti od razdraženosti. Poleg tega zaradi lažjega razumevanja le tega, tudi spremljamo spreminjanje režimov delovanja v odvisnosti od števila agentov. Pri tem pa smo tekom simulacije beležili kolikšno je povprečno zanimanje za določeno mesto pristajalne površine (v korakih velikosti radija agenta oziroma 0,5) v odvisnosti od povprečnega števila agentov.

4.3.1 Upoštevanje dolžinsko neomejenega vidnega polja



Slika 4.10 Prikaz verjetnosti pojavitve agentov na pristajalni površini (dolžina pristajalne površine v odvisnosti števila agentov).

V nasprotju z dobljenimi rezultati manjše populacije agentov, se v istem času pojavi veliko več razdraženosti. Zelo pa je tudi vidno, da namesto množičnega počivanja nastopa več pristajanja, kar je približno 85% agentov.



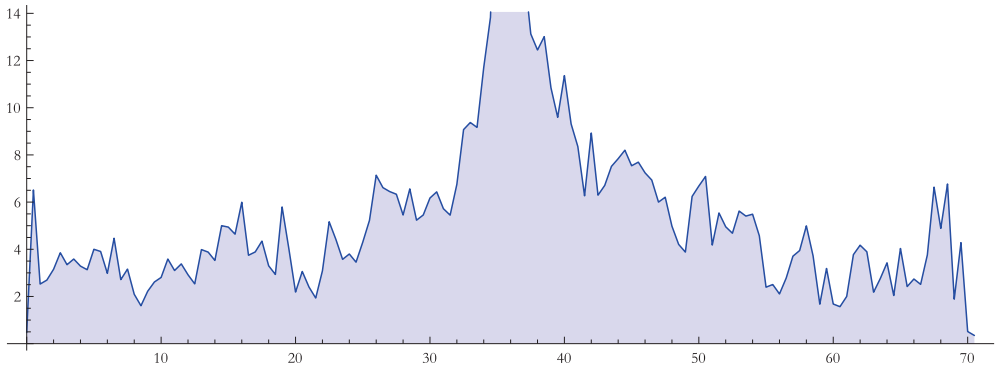
Slika 4.11 Zgoraj je prikazano spreminjanje energije (modra) in razdraženosti (rdeča) skozi čas. Spodaj je prikazano spreminjanje režima delovanja (letenje - zelena, pristajanje - oranžna, počivanje - rdeča) v odvisnosti števila agentov skozi čas.

Zaradi večje populacije agentov je v tem primeru veliko bolj vidno, da je želja po pristajanju na robovih pristajalne površine veliko večja, kot nekje v središču (anomalije vračanja v središče sveta v tem primeru niso prisotne, zaradi hitrejšega vzleta in nato hitrega pristanka, oziroma agent je zmožen takoj videti novo točko za pristanek).

4.3.2 Upoštevanje dolžinsko omejenega vidnega polja

Primer velike populacije agentov z dolžinsko omejenim vidnim poljem privede le do daljšega pristajanja k počitku, zaradi česa se poveča intenzivnost razdraženosti. Poleg vsega pa je vidno veliko večje skupinsko pristajanje in vzletanje, kar v primeru dolžinsko neomejenega vidnega polja ni prisotno tako zelo intenzivno.

Slika 4.12 Zgoraj je prikazano spreminjanje energije (modra) in razdraženosti (rdeča) skozi čas. Spodaj je prikazano spreminjanje režima delovanja (letenje - zelena, pristajanje - oranžna, počivanje - rdeča) v odvisnosti števila agentov skozi čas.



Slika 4.13 Prikaz verjetnosti pojavitve agentov na pristajalni površini (dolžina pristajalne površine v odvisnosti števila agentov).

Verjetnost pojavitve agentov na točno določeni točki pristajalne površine so v nasprotju ostalih primerov zelo uravnotežena, če se zanemari pojavitve anomalije vračanja v središče navideznega sveta.

5 Zaključek

Z uporabo odprto-kodnega programa OpenSteer, avtorja Craiga Reynoldsa smo realizirali algoritem vzletanja in pristajanja jat ptic. Za prikaz delovanja smo izvedli nekaj primerov, v katerih smo opazovali reakcijo agentov z uporabo dolžinsko omejenega in dolžinsko neomejenega (neskončnega) vidnega polja na dveh množicah agentov.

Delovanje samega algoritma smo realizirali s prilagoditvijo modela letečega agenta, kateremu smo dodali parametre beleženja:

- energije,
- razdraženosti,
- režimov delovanja,
- želj.

Zaradi načina pristajanja in ideje po nenehnem združevanju v jate, smo preuredili vedenjske funkcije tako, da delujejo z upoštevanjem želj in režimov delovanja. Na koncu smo tudi izdelali zaznavanja pristajalnega polja oziroma površine, enake oblike kot pri

upoštevanja sosedov za izračun vedenjskih sil. Ob izračunu videnega polja in točke, smo implementirali še ne-agresiven algoritem prihajanja.

Ob dobljenem končnem izdelku se je pojavilo tudi nekaj ključnih smernic za nadaljnjo izboljšavo delovanja celotnega algoritma. Najbolj ključna je pohitritev, ki se pojavi ob testiranju večje množice agentov pri izbiranju nove oziroma boljše točke na pristajalni površini (uporabljen algoritem za generiranje naključnih števil z verjetnostjo v obliki Gaussove krivulje je veliko počasnejši od standardnega generatorja naključnih števil). Poleg tega, tudi implementacija dodatnega preprečevanja nepotrebnih trkov, ki se pojavljajo ob prerivanju agentov na pristajalni površini; dati mora možnost agentu, da se lahko odloči predčasno ali bo pristanek privedel do čim manjšega prerivanja ob svojem pristanku. Zaradi samega prikaza delovanja simulacije je tudi zelo pomembno, da bi se agenti med letenjem ustavljali enako usmerjeni navzgor za približno 15°, ampak še vedno čim bolj pravokotno na pristajalno površino.

LITERATURA

[1] J. R. G. Dyer, C. C. Ioannou, L. J. Morrell, D. P. Croft, I. D. Couzin, D. A. Waters, J. Krause, Consensus decision making in human crowds, Animal Behaviour 75 (2) (2008) 461–470.

[2] C. W. Reynolds, Flocks, herds and schools: A distributed behavioral model, ACM SIGGRAPH Computer Graphics 21 (4) (1987) 25–34.

[3] C. W. Reynolds, Steering behaviors for autonomous characters, in: Proc. of GDC 1999, Miller Freeman Game Group, San Francisco, CA, 1999, pp. 763–782.

[4] I. Lebar Bajec, Računalniški model letenja ptic v jatah, MSc thesis, Fakulteta za računalništvo in informatiko (2002).

[5] F. Heppner, U. Grenander, A stochastic nonlinear model coordinated bird flocks, in: S. Krasner (Ed.), The Ubiquity of Chaos, AAAS Publications, 1990, Ch. 19, pp. 233–238.

[6] C. Delgado-Mata, J. Martinez, S. Bee, On the use of virtual animals with artificial fear in virtual environments, New Generation (2007) 1–10.

[7] C. Hartman, N. G. Street, W. Lafayette, Autonomous Boids, Computer Animation and Virtual Worlds (2006) 1–26.
url: http://www.tech.purdue.edu/CGT/Applied-Research/documents/benes_boids.pdf

[8] I. Lebar Bajec, Fuzzy Model for a Computer Simulation of Bird Flocking, PhD dissertation, Fakulteta za računalništvo in informatiko (2005).

[9] D. Eberly, Intersection of a Line and a Cone, Geometric Tools, LLC (2008) 1–4.
url: http://www.geometrictools.com/Documentation/IntersectionLineCone.pdf