TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Mattias Lass

Tehisintellekti loomine käigupõhisele strateegiamängule L-put

Bakalaureusetöö (9 EAP)

Juhendaja: Jaanus Jaggo, MSc

Tartu 2017

Tehisintellekti loomine käigupõhisele strateegiamängule L-put

Lühikokkuvõte:
Käigupõhised strateegiamängud on sellised strateegiamängud, milles mängijad langetavad otsuseid käikude kaupa. Need mängud pakuvad suurepärast võimalust rakendada erinevaid tehisintellektide jaoks mõeldud algoritme. Käesoleva töö eesmärk on realiseerida sellist tüüpi mängule tehisintellekt, kasutades minimax ja Monte Carlo algoritme. Töös antakse hinnang nende algoritmide efektiivsusele ning katsetatakse nende võimekust töö raames loodud käigupõhisel strateegiamängul L-put. Samuti annab töö ülevaate kuusnurksete väljadega mänguväljade võimalikest koordinaatsüsteemidest ja algoritmidest, mida nendel mänguväljadel kasutada saab.

Võtmesõnad:
käigupõhine strateegiamäng, kuusnurksete väljadega mängulaud, minimax, alfa-beeta kärpimine, monte carlo

Creating an Artificial Intelligence for a Turn-Based Strategy Game L-put

Abstract:
Turn-based strategy games are a subset of strategy games, where players make decisions by turns. These type of game provide an excellent opportuniy for testing different Artificial Intelligence algorithms. The aim of this thesis is to create AI systems using minimax and Monte Carlo algorithms and to test their performance on L-put, which is a turn-based strategy game that was made for this thesis. The thesis also provides an overview of different coordinate systems and algorithms for hex-based playing fields.

Keywords:
turn-based strategy game, hexagonal tiles, minimax, alpha-beta pruning, monte carlo

Sisukord

1 Sissejuhatus

L-put on käesoleva töö käigus loodud käigupõhine strateegiamäng, mis sarnaneb oma reeglitelt arvutimängu *Heroes of Might and Magic III* ühele osale. Töö eesmärgiks on L-puti implementeerimine Unity mängumootoril ja sellele algelise tehisintellekti loomine. Kuna L-puti mängitakse kuusnurksete väljadega mängulaual, on töö üheks osaks sellise mänguvälja omaduste kirjeldamine. Töö on abiks kuusnurksete väljadega mängulaua implementeerimisel ja see on abiks kõigile, kes soovivad ise implementeerida mõnele käigupõhisele strateegiamängule algelist tehisintellekti.

Esimeses peatükis (L-puti reeglid) kirjeldatakse, millised on töös loodud mängu L-put reeglid.

Teises peatükis (Sarnased käigupõhised strateegiamängud) antakse lühiülevaade kahest mängust, millest lähtudes L-put on loodud.

Kolmandas peatükis (Kuusnurksete väljadega mängulaud), kirjeldatakse iseärasusi, mis kaasnevad mängulaudadega, mille väljad on ruutude asemel kuusnurgad. Samuti antakse ülevaade selliste mänguväljade võimalikest koordinaatsüsteemidest ja algoritmidest, mida nendel väljadel kasutada saab.

Neljandas peatükis (L-puti tehisintellekt), selgitatakse minimax ja Monte Carlo algoritme, mida on võimalik kasutada, et luua tehisintellekti käigupõhisele strateegiamängule. Samuti võrreldakse erinevaid tehisintellekti lahendusi, mida L-put kasutab.

Lisana on kaasas töö käigus valminud programm L-put mängule, mis sisaldab nii mängu ennast, kui ka loodud tehisintellekti selle mängimiseks.

2 L-puti reeglid

L-put on kahe mängijaga käigupõhine strateegiamäng. Mõlemad mängijad kontrollivad võrdse tugevusega armeed. Armee koosneb viiest mängunupust, mis esindavad erinevate omadustega üksusi. Mängijad saavad oma käikude ajal nuppe liigutada ja nendega vastase nuppe rünnata. Mängu võitmiseks tuleb mängulaualt eemaldada kõik vastase nupud.

2.1 Nupud

L-putis kujutatud mängunupp esindab üksust, mis koosneb mitmest ühesuguste omadustega võitlejast. Näiteks üks nupp kujutab vibulaskjaid ja teine rüütleid. Vibulaskjad on aeglased ja kergesti haavatavad, kuid kuna neil on vibud saavad nad rünnata ka nuppe, mis ei ole vahetult nende kõrval. Rüütlid on kiired ratsanikud, kellel on kõrge ründevõime, kuid kuna nende relvaks on mõõk saavad nad rünnata ainult vahetult enda kõrval seisvaid nuppe.

2.1.1 Nuppude omadused

Erinevatel nuppudel on erinevad omadused, mis määravad muuhulgas näiteks selle kui kaugele nupp ühe käigu jooksul liikuda saab, kui suurt kahju suudab nupp vastase nuppudele tekitada ja kui palju kahju peab nupp saama, et ta hävineks. Selles alampeatükis defineeritakse kõik nuppude omadused ja kirjeldatakse valemit, mille abil leitakse kahju mida nupp ründamisel tekitab.

Liikumiskiirus määrab kui palju nupp saab ühe käigu jooksul liikuda.

Initsiatiiv määrab, mis järjekorras käigu sees nuppe liigutatakse. Suurema initsiatiiviga nupud liiguvad esimesena.

Võitlejate arv näitab, mitu võitlejat on üksuses, mida nupp esindab. See on ainus mängu jooksul muutuv omadus. Kui nuppu rünnatakse, siis piisava kahju tekitamisel hukkuvad mõned võitlejad. Kui võitlejate arv langeb nullini, siis eemaldatakse nupp mängulaualt. Võitlejate arv määrab ka nupu võime tekitada vastasele kahju. Kõik nupud alustavad mängu 16 võitlejaga.

Elupunktid esindavad ühe võitleja võimet saada kahju enne kui ta hukkub. Kui nuppu rünnatakse leitakse erinevate parameetrite abil kahju, mis nupp saab. Seejärel leitakse kahju ja elupunktide jagatise täisosa ja jääk. Täisosa lahutatakse võitlejate arvust, see näitab mitu võitlejat hukkus. Jääki kasutakse ühe parameetrina, et leida kahju järgmisel korral kui nuppu rünnatakse.

Ründe tugevus esindab jõudu millega üks võitleja üksuses vastast ründab. See on täis-

arvuline väärtus mida kasutatakse kahju tekitamise valemis.

Ründamisoskus esindab võitleja võimekust vastasele pihta saada. See on täisarvuline väärtus mida kasutatakse kahju tekitamise valemis.

Kaitsemisoskus esindab võitleja turvist ja oskust vastase lööke tõrjuda. See on täisarvuline väärtus, mida kasutatakse kahju tekititamise valemis.

Kui üks nupp teist nuppu ründab, siis leitakse tekitatav kahju kasutades **kahju tekitamise valemit**. Kahju tekitamise valem on järgnev:

$$K = a * t * (1 + 0,05 * (r - k)) + j \tag{1}$$

Valemis 1 tähistavad muutujad järgnevaid väärtusi:

K - kahju,

a - ründaja võitlejate arv,

t - ründaja ründamistugevus,

r - ründaja ründamisoskus,

k - kaitseja kaitsmisoskus,

j - jääk eelmisest kaitseja vastu sooritatud rünnakust.

2.1.2 Nuppude tüübid

Selles alampeatükis kirjeldatakse kõiki erinevaid nuppe. Nende omadustele antakse täpsed väärtused ja lisaks kirjeldatakse nende nuppude tugevusi ja nõrkusi.

Piigimees on väga tugeva kaitsmisoskusega nupp, kuid sellel on kõige madalam ründamistugevus. See on tugev rüütli, kuid nõrk mõõgamehe vastu.

Ründamisoskus	10	**Kaitsemisoskus**	16	**Ründe tugevus**	8
Elupunktid	108	**Liikumiskiirus**	3	**Initsiatiiv**	3

Tabel 1. piigimehe omadused

Mõõgamees on väga tugeva ründamisoskusega nupp, kuid sellel on keskmisest madalamad elupunktid. See on tugev mõõgamehe, kuid nõrk rüütli vastu.

Ründamisoskus	14	**Kaitsemisoskus**	10	**Ründe tugevus**	10
Elupunktid	98	**Liikumiskiirus**	3	**Initsiatiiv**	4

Tabel 2. mõõgamehe omadused

Rüütel on kõrgete elupunktide ja kõrge ründamistugevusega nupp. Lisaks on see liikumiskiiruse poolest teine nupp. Tasakaalustamiseks on sellel väga madal kaitsmisoskus ja madal initsiatiiv. Rüütel on tugev mõõgamehe, kuid nõrk piigimehe vastu.

Ründamisoskus	10	**Kaitsemisoskus**	4	**Ründe tugevus**	12
Elupunktid	128	**Liikumiskiirus**	5	**Initsiatiiv**	2

Tabel 3. rüütli omadused

Vibumees on omaduste arvulise väärtuste poolest kõige nõrgem nupp. Selle ainus tugev omadus on ründamistugevus. Vibumehega on aga võimalik rünnata ka nuppe, mis ei asu selle naaberväljades. Vibumehega saab rünnata kõiki nuppe, mis on selle nägemisulatuses. Nägemisulatuse leidmist kirjeldatakse neljandas peatükis. Lisaks ei ole võimalik vibumehe rünnakutele sooritada vasturünnakuid ja ka vibumees ise ei soorita vasturünnakuid. Vasturünnakute sooritamise reegleid on lähemalt seletatud käigu alampeatükis.

Ründamisoskus	8	**Kaitsemisoskus**	4	**Ründe tugevus**	12
Elupunktid	90	**Liikumiskiirus**	4	**Initsiatiiv**	1

Tabel 4. vibumehe omadused

Lendaja on keskpäraste võitlusomadustega nupp, kuid selle kõrge kiirus ja parim initsiatiiv teevad lendajast olulise abinupu. Kuna lendaja liigub ja võitleb lennates ei avalda väljade tüübid lendajale mõju. Lendaja saab üle liikuda kaljudest, sügavast veest ja väljadest kus asuvad teised nupud, kuid ei saa neil oma käiku lõpetada.

Ründamisoskus	10	**Kaitsemisoskus**	10	**Ründe tugevus**	10
Elupunktid	100	**Liikumiskiirus**	8	**Initsiatiiv**	6

Tabel 5. lendaja omadused

2.2 Mängulaud

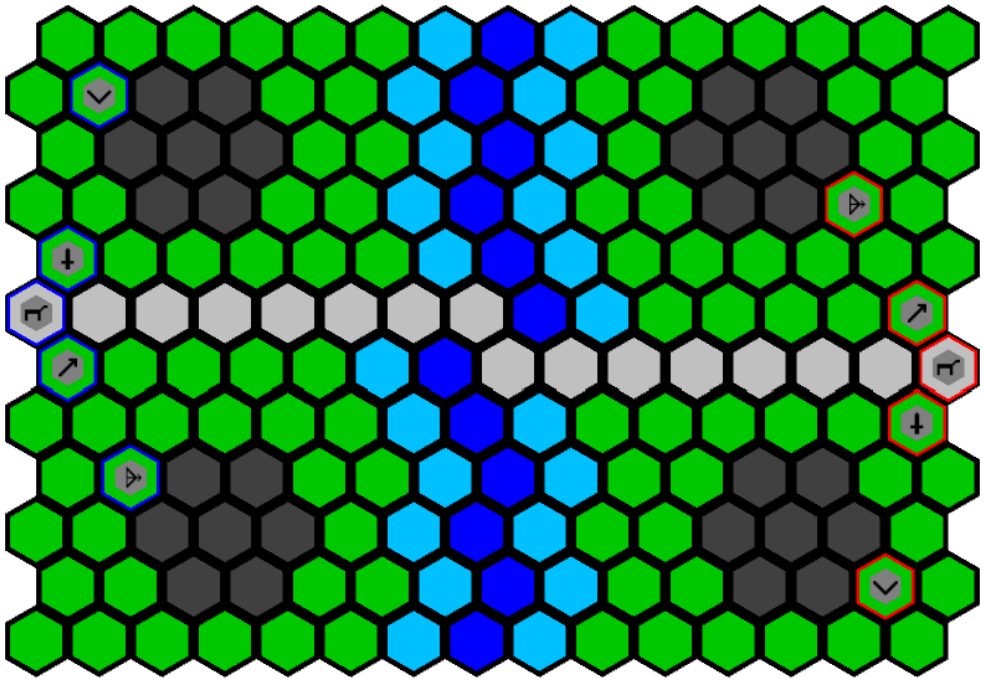
L-puti mängulaud, mida on kujutatud joonisel 1, koosneb võrdkülgsete kuusnurkade kujulistest väljadest. Mängulaual on 12 rida, mis koosnevad 15 kuusnurgast. Sellised mõõtmed annavad lauale ristkülikule sarnaneva kuju, mille tõttu on mängijate nuppude vahel palju ruumi. Mängunuppude alguskohad on lauale määratud selliselt, et esimesel käigul ei ole võimalik vastase nuppe rünnata.

Mängulaua mõõtmete ja kuusnurksete väljade tõttu ei ole võimalik saavutada selle sümmeetrilisust x- või y-telje suhtes. Teatud sümmetria on mängu tasakaalu jaoks aga va-

jalik, et tagada mõlemale mängijale võimalikult võrdne alguspositsioon. Seega on laual väljad ja nuppude alguspositsioonid määratud nii, et kui esialgse seisuga lauda peegeldada nii x- kui ka y-teljest saadakse täpselt mängulaud, kus mängijad on vahetanud pooled, kuid väljade ja nuppude asukohad säilivad.

Mängulaud esindab lahinguvälja, mille keskelt jookseb vertikaalselt läbi jõgi. Jõega risti poolitab välja tee, mis ületab sillana ka jõe. Mõlemal pool jõge asub kaks kaljunukki, mis takistavad nuppude liikumist ja pakuvad kaitset viburünnakute eest.

Joonisel 1 on kujutatud ka kõiki erinevaid välju ja nuppe. Roheline väli on muru, helehall on tee, tumesinine on sügav vesi, helesinine on madal vesi ja tumehall on kalju. Sinise mängija nupud liikudes ülevalt alla on lendaja, mõõgamees, rüütel, piigimees ja vibumees.



Joonis 1. mängulaud ja nuppude esialgne paigutus

2.2.1 Väljade tüübid

Väli on kuusnurk, millel nupud asetsevad. Igal väljal saab korraga olla ainult üks nupp. Kokku on viit erinavat tüüpi välju:

1. **muru** on tavaline väli, millest läbi liikumine vajab ühte liikumispunkti;

2. **tee** vajab sellest läbi liikumiseks poolt liikumispunkti ja suurendab sellel seisva nupu ründeoskust ühe võrra;

3. **madal vesi** vajab sellest läbi liikumiseks kahte liikumispunkti ja vähendab sellel seisva nupu kaitsmisoskust ühe võrra;

4. **sügavast veest** ei ole võimalik nuppudel läbi liikuda;

5. **kaljust** ei ole võimalik nuppudel läbi liikuda ja lisaks piirab see vibumehe nägemisulatust.

2.3 Käik

L-putis peavad ühe käigu jooksul mõlemad mängijad liigutama kõiki oma nuppe. Ühe nupu liigutamist kutsutakse osakäiguks. Osakäikude järjekord on määratud vastavalt liigutatava nupu initsiatiivile. Esimesena tuleb sooritada osakäik kõige kõrgema initsiatiiviga nupuga. Kuna mõlemal mängijal on võrdse initsiatiiviga nuppe, kasutatakse nende liigutamise järjekorra leidmiseks käigu paarsust. Paarituarvulistel käikudel liigutab oma nuppu esimesena sinine mängija ja paarisarvulistel käikudel punane mängija.

Ühel osakäigul võib nupp kulutada liikumispunkte vastavalt sellele, kui suur on nupu liikumiskiirus. Väljalt A naaberväljale B liikumine kulutab teatud arvu liikumispunkte vastavalt välja A tüübile. Täpsed liikumispunktide hinnad on välja toodud väljade tüüpide alapeatükis. Kui väljalt A ära liikumine nõuab rohkem liikumispunkte, kui nupul veel kulutada on, ei saa nupp enam liikuda. Ei ole võimalik liikuda väljale, kus asub mõni teine nupp.

Pärast nupuga liikumist saab sellega rünnata kõiki vastaste nuppe, mis asuvad selle välja naaberväljadel kuhu nupp liikus. Erandiks on vibumees, kellega saab rünnata ainult juhul, kui seda ei liigutata. Pärast vastase nupu ründamist sooritab vastase nupp automaatse vasturünnaku, kuid seda ainult ühe korra käigu jooksul. Nupp mis on sooritanud vasturünnaku kaotab kuni käigu lõpuni ühe punkti kaitseoskusest. Kui mängija otsustab nuppu mitte liigutada ja sellega ka mitte rünnata, siis saab see nupp ühe punkti oma kaitseoskusele boonuseks. See boonus kestab kuni selle nupu liigutamiseni.

Mängija saab ka nupuga osakäigu sooritamise asemel selle nupu osakäiguga oodata. Ühe nupuga saab oodata ainult üks kord käigu jooksul. Kui mängija otsustab nupu osakäiguga oodata, sooritatab ta selle nupuga osakäigu alles siis, kui on sooritatud osakäigud kõikide madalama initsiatiiviga nuppudega. Seega näiteks kui mõlemad mängijad otsustavad kõikide oma nuppudega oodata, siis sisuliselt pööratakse osakäikude järjekord ümber.

3 Sarnased käigupõhised strateegiamängud

L-puti disainimisel on kasutatud ideid mitmestest teistest sarnastest käigupõhistest strateegiamängudest. Selles peatükis on toodud kirjeldused neist kahele põhilisele *Heroes of Might and Magic* ja *Age of Wonders* 3.

3.1 *Heroes of Might and Magic*

Heroes of Might and Magic ehk HoMM on käigupõhiste strateegiamängude sari, millel on aastatel 1995–2015 ilmunud seitse erinevat mängu. HoMM-us on mängija ülesanne juhtida ühte mitmest omavahel sõjas olevatest riikidest. Mängu edukaks läbimiseks peab mängija haldama linnu ja juhtima armeesid, et hävitada kõik vastased. Armeedega on võimalik vallutada vastaste linnu ja rünnata nende armeesid. Armee või linna ründamisel sisenetakse mängu lahingu faasi, kus kaks mängijat annavad käigupõhiselt käske oma armees olevatele üksustele.[wik17b]

HoMM III käsiraamatu [her99] põhjal on näha, et L-put ja HoMM-u lahingufaas on väga sarnased mängud. Mõlemas mängus üksustel samasugused omadused ja ka käikude struktuur on samasugune. HoMM-us kontrollib mängija aga kuni seitset üksust ja erinevate võimalike üksuste hulk on märkimisväärselt suurem kui L-putis. Lisaks sellele sisaldab HoMM mitmeid mittedeterministlike elemente:

1. üksuste ründamistugevus on täisarvude vahemik, millest valitakse ründamisel üks juhuslik arv, mida kasutatakse tekitatud kahju arvutamiseks;

2. üksustel on lisaomadus moraal, mis annab teatud tõenäosusega võimaluse sooritada üksusega lisakäik;

3. üksustel on lisaomadus õnn, mis võimaldab teatud tõenäosusega teha rünnates rohkem kahju.

3.2 *Age of Wonders* 3

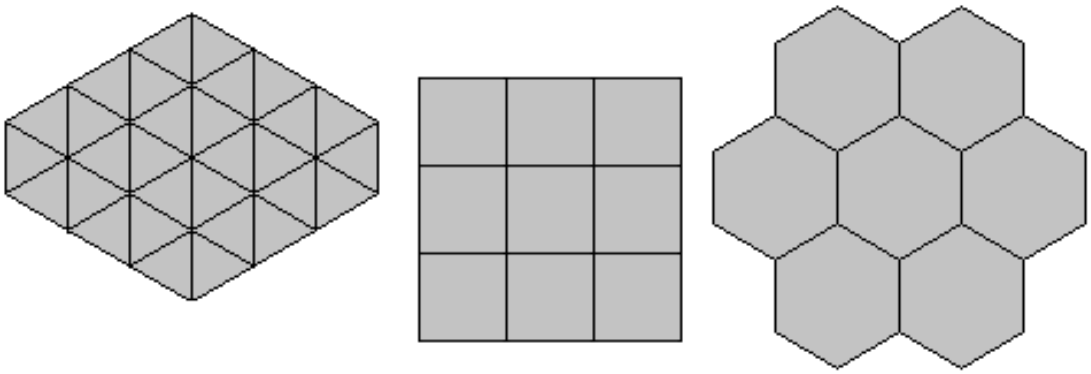
Age of Wonders 3 ehk AoW on 2014. aastal väljalastud käigupõhine strateegiamäng. Sarnaselt HoMM-ule juhib mängija AoW-is ühte sõjasolevatest riikidest. AoW-is on mängijal ulatuslikum kontroll erinevate riigijuhtimise aspektide üle. Näiteks pakub linnade haldamine mängijale rohkem võimalusi ning oma riigi laiendamiseks on mängijatel ka võimalik ise linnu ehitada, erinevalt HoMM-ust kus linnu oli võimalik juurde saada ainult vallutamise teel. Ka AoW-is on käigupõhine lahingufaas.[wik17a]

AoW-i lahingufaas põhineb samuti kuusnurksel mänguväljakul, kus mängijad saavad liigutada oma üksusi. Erinevalt L-putist ja HoMM-ust on AoW-is üksus üks võitleja, mis tähendab et üksuse ründe tugevus ei ole sõltuvuses sellest kui palju kahju üksus lahingu

käigus saanud on. AoW teine peamine eripära on see, et üksustel on iga käik teatud arv tegevuspunkte, mida üksus saab kulutada liikumiseks, ründamiseks ja erivõimete kasutamiseks. Erivõimed mõjutavad teiste üksuste või mänguväljade omadusi. Kuna AoW-i üheks elemendiks on takistused lahinguväljal, siis on selles implementeeritud ka vibulaskjate nägemisulatuse arvutamine, mida kasutatakse ka L-putis. [wik17a]

4 Kuusnurksete väljadega mängulaud

Mosaiik on tasand või kahemõõtmeline kujund, mis on jagatud mitmeks kahemõõtmelisteks kujundiks. Korrapärane mosaiik on mosaiik, mis koosneb ainult ühesugustest võrdkülgsetest hulnknurkadest. Korrapäraseid mosaiike, nagu on kujutatud joonisel 2, saab luua ainult kolmnurkadest, ruutudest ja kuusnurkadest[Wei]. Üritades tasandit mosaiigiks jagada teiste võrdkülgsete hulknurkadega pole võimalik, sest tekivad katmata alad, kuhu ei saa vastavat hulknurka mahutada.



Joonis 2. korrapärased mosaiigid

Sisuliselt on kõik erinevad mängulauad, mis on jagatud väiksemateks väljadeks mosaiigid. Ilmselt on enimlevinud mosaiik mängulaudade katmiseks ruutmosaiik ehk mosaiik mis koosnev ühesuguste mõõtmetega ruutudest. Sellist mosaiiki kasutatakse näiteks malelaual. Ruutmosaiigil on kaks peamist eelist:

1. ruutmosaiik võimaldab täielikult katta nelinurkse pinna;
2. ruutmosaiigi jaoks on võimalik luua lihtne koordinaatsüsteem kasutades x- ja y-telge.

L-putis on mängulaud võrdkülgsetest kuusnurkadest koosnev mosaiik. Kuigi kuusnurksete väljadega mängulaua kasutamine vajab keerukamat koordinaatsüsteemi ja on sellisel mänguväljal üks peamine eelis. Korrapärastes mosaiikides on igal väljaga külgnevate väljade arv võrdeline välja külgede arvuga. Väljaga külgnevaid välju nimetatakse naaberväljadeks. Kasutades nuppude liigutamiseks süsteemi, kus nupud liiguvad sammhaaval ja samme saab sooritada naaberväljadele võimaldab kuusnurksete väljadega mosaiik korrapärastest mosaiikidest suurima hulga liikumissuundi. Seega võib väita

et välja jaotamine kuusnurkadeks annab täpseima lihtsustuse reaalsele ruumile, kus liikumissuundade arv on lõpmatu.

Kuusnurkade tasandile paigutamiseks on kaks peamist viisi, mida on kujutatud joonisel 3. Esimesel viisil on kuusnurga üks külgede paar paralleelne tasandi y-teljega teisel jugul on üks külgede paar paralleelne x-teljega. Kuna erinevad kuusnurkade paigutused saadakse teineteisest tasandi pööramise teel võib erinevaid paigutusi lugeda isomorfideks. Käesolevas töös kasutatakse näidete kirjeldamiseks paigutust, kus üks külgede paar on paralleelne y-teljega.

Joonis 3. y-teljeline ja x-teljeline viis kuusnurkade tasandile paigutamiseks

4.1 Koordinaatsüsteemid kuusnurksete väljadega mängulauale

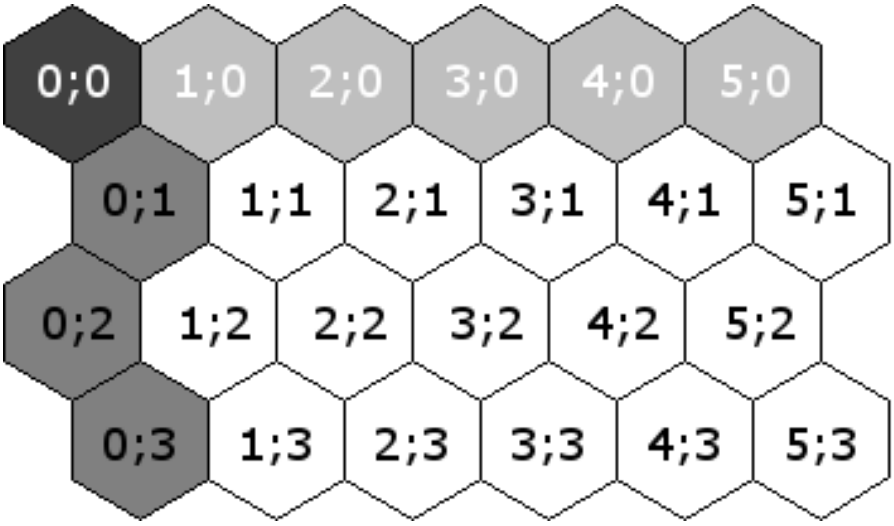
Üheks suurimaks probleemiks kuusnurksete väljadega mängulaua kasutamisel on sobiva koordinaatsüsteemi leidmine. Kordinaatsüsteem peab olema sobilik nii mänguvälja arvuti mälus hoidmiseks kui ka kirjeldama reaalseid seoseid erinevate väljade vahel, näiteks kui kaugel kaks välja teineteisest asuvad. L-puti implementeerimisel kasutati kahte koordinaatsüsteemi, mille kombineerimine võimaldas lahendada mõlemad probleemid.

4.1.1 Nihkega koordinaatsüsteem

Jooniselt 4 on näha, et kuusnurksete väljadega mänguväljal on paaris ja paarituarvulised read omavahel nihkes. Nihkega koordinaatsüsteemis võetakse seda omadust arvesse ja seatakse nihkesse ka y-telg, mida on joonisel 4 kujutatud tumehalli tooniga. Hellehall toon kujutab sellel joonisel x-telge.

Nihkega koordinaatsüsteem eelis on see, et see võimaldab kergesti hoida infot väljade kohta kahemõõtmelises massiivis, kuna väljade koordinaadid kattuvad täielikult massiivi indeksitega. Näiteks saab hoida joonisel 4 kujutatud mänguvälja sellises kahemõõtmelises massiivis, kus välimise massiivi suurus on neli ja sisemiste massiivide suurus on 6 elementi. Kuna nihkega koordinaatsüsteem sarnaneb ruudustikule, siis on see inimese jaoks kõige intuitiivsem.

Nihkega koordinaatsüsteemi puhul on puuduseks see, et teades ainult kahe välja koordinaate, ei ole võimalik anda hinnangut kuidas need väljad teineteise suhtes paiknevad.



Joonis 4. nihkega koordinaatsüsteem, helehalliga on tähistatud x-telg ja tumehalliga y-telg

Selleks on vaja ka teada kas väljad asuvad nihkega või nihketa reas. Samuti ei ole võimalik koordinaatide abil defineerida ühesuguseid liikumissuundi. Nagu on näha tabelist 6 erinevad osade liikumissuunade vektorid vastavalt sellele, millises reas asutakse.

Liikumissuund	**Nihketa rida**	**Nihkega rida**
Vasak	-;0	-;0
Parem	+;0	+;0
Ülevale ja vasakule	-;-	0;-
Ülevale ja paremale	0;-	+;-
Alla ja vasakule	-;+	0;+
Alla ja paremale	0;+	+;+

Tabel 6. suunavektorid nihkega koordinaatsüsteemis, miinusmärk kujutab nihet negatiivses suunas ja pluss märk nihet positiivses suunas

4.1.2 Nihketa koordinaatsüsteem

Teine kuusnurksete väljadega mängulaua koordinaatsüsteem on nihketa koordinaatsüsteem. Erinevalt nihkega koordinaatsüsteemist on selles süsteemis y-telg sirge nagu on näha jooniselt 5, kus y-telge on kujutatud tumehalli tooniga.

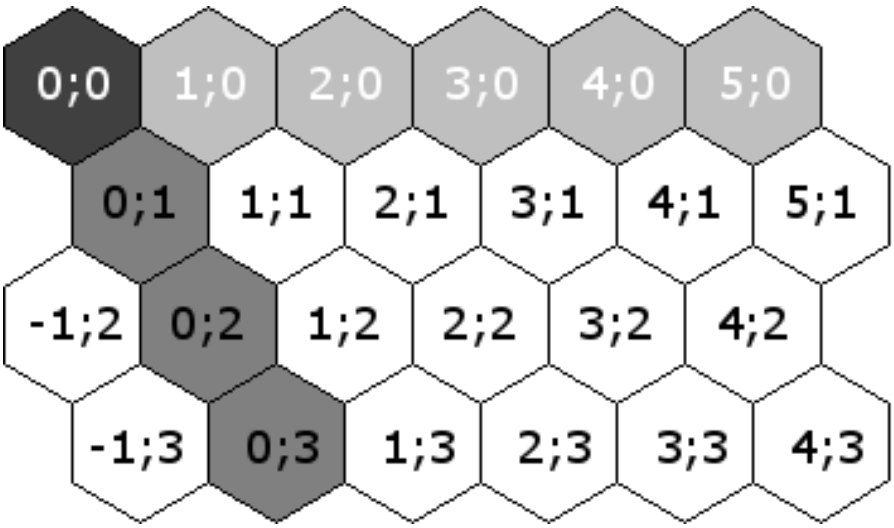
Nihketa koordinaatsüsteemi on keerulisem hoida kahemõõtmelises massiivis otse, kuna

väljade koordinaadid ja massiivi indeksid ei kattu kõikide väljade korral. Näiteks joonise 5 põhjal on näha, et alates kolmandast reast on koordinaatide ja indeksite vahel erinevus.

Nihketa koordinaatsüsteemi eeliseks on see, et selles on liikumissuunad ehk suunavektorid üheselt defineeritud, need on kõikide väljade jaoks samasugused. Tabelis 7 on kujutatud suundade ja suunavektorite seosed. Selles koordinaatsüsteemis on võimalik määrata kahe välja vahelist kaugust, teades ainult nende väljade koordinaate ning ka teiste algoritmide rakendamine on lihtsam, kui nihkega koordinaatsüsteemi puhul.

Liikumissuund	**Vektor**
Vasak	-;0
Parem	+;0
Ülevale ja vasakule	0;-
Ülevale ja paremale	+;-
Alla ja vasakule	-;-
Alla ja paremale	0;-

Tabel 7. suunavektorid nihkega koordinaatsüsteemis, miinusmärk kujutab nihet negatiivses suunas ja pluss märk nihet positiivses suunas



Joonis 5. nihkega koordinaatsüsteem, helehalliga on tähistatud x-telg ja tumehalliga y-telg

4.1.3 Nihkega ja nihketa koordinaatsüsteemide vaheline teisendamine

Kuna nihketa ja nihkega koordinaatsüsteem täiendavad teineteist, ühe süsteemi nõrkus on teise tugevus, siis on oluline nende süsteemide vaheline teisendamine. Teisendada on vaja ainult x-koordinaati, sest mõlemad süsteemid jagavad samasugust x-telge. Valemid 2 ja 3, mis kirjeldavad vastavaid teisendusi, põhinevad A. Pateli [Pat15] kokkukogutud materjalidel võrdkülgsete kuusnurksete mängulaudade kohta.

$$x_2 = x_1 - (y + (1 + y)\%2)/2; \tag{2}$$

$$x_1 = x_2 + (y + (1 + y)\%2)/2; \tag{3}$$

Valemites 2 ja 3 tähistavad sümbolid järgnevaid väärtusi:

x_1 - nihkega süsteemi x-koordinaat,

x_2 - nihketa süsteemi x-koordinaat,

y - mõlema süsteemi y-koordinaat,

$\%$ - jagatise jääk.

4.2 Algoritmid kuusnurksete väljadega mänguväljadele

Järgnevas alampeatükis kirjeldatakse erinevaid kuusnurksete väljade algoritme, mida on kasutatud L-puti implementeerimisel. Järgnevalt väljatoodud algoritmid kasutavad nihketa koordinaatsüsteemi.

4.2.1 Kahe välja vahelise kauguse leidmine

Kahe välja vaheliseks kauguseks loetakse vähimat vajalikke sammude arv, et liikuda ühelt väljalt teisele. Üks samm on liikumine väljalt mõnele selle naaberväljale. Pateli [Pat15] kogutud materjalide põhjal on võimalik kahe välja vahelist kaugust arvutada valemiga 4.

$$d = 1 - (abs(x_1 - x_2) + abs(y_1 - y_2) + abs(x_1 + y_1 - x_2 - y_2))/2 \tag{4}$$

Valemis 4 tähistavad sümbolid järgnevaid väärtusi:

d - kaugus,

x_1 - algusvälja x-koordinaat,

x_2 - lõppvälja x-koordinaat,

y_1 - algusvälja y-koordinaat,

y_2 - lõppvälja y-koordinaat,

abs - absoluutväärtuse funktsioon.

4.2.2 Kahe välja vahelise joone leidmine

Arvutigraafikast on tuntud probleem kuidas joonistada ruutmosaiigile ehk ruudustikule sirglõiku. Samasugust probleemi saab lahendada ka kuusnurkadest koosneval mosaiigil. Pateli [Pat15] materjalides on kirjeldatud järgnevat lineaarsel interpoleerimisel põhinevat algoritmi.

1. Leia algusvälja A ja lõppvälja B vaheline kaugus N.
2. Lisa mõlema välja koordinaatidele kolmas mõõde $z = -x - y$.
3. Leia $N + 1$ koordinaatide kolmikut. Selleks itereeri indeksiga i üle valemi $A + (B - A)/N * i$, kus i on vahemikus 0 kuni N.
4. Ümarda koordinaadid täisväärtustele kasutades koordinaatide ümardamise algoritmi.
5. Eemalda koordinaatidelt kolmas mõõde z, ning saadud koordinaadid esindavad kõiki välju, mis on joone joonistamiseks vajalikud.

4.2.3 Koordinaatide ümardamise algoritm

Kahe välja vahelise joone leidmiseks on vajalik koordinaatide ümardamise algoritm. Algoritm saab sisendiks kolm ratsionaalarvu: x, y ja z ning tagastab kolm täisarvu: x_r, y_r ja z_r. Patel [Pat15] kirjeldab algoritmi järgnevalt.

1. Leia sisendiks saadavate arvude x, y ja z ümardatud väärtused, mis on vastavalt x_r, y_r ja z_r.
2. Leia iga sisendi ja sellele vastava ümardatud väärtuse erinevus ehk vahe absoluutväärtus.

3. Väärtusta ümber suurima erinevusega ümardatud väärtus. Vastavalt kas $x_r = -y_r - z_r$, $y_r = -x_r - z_r$ või $z_r = -x_r - y_r$. Kui z ja x või y jagavad suurimat erinevust, väärtusta ümber z_r. Kui x ja y jagavad suurimat erinevust ja z erinevus on väiksem väärtusta ümber y_r.

4. Tagasta leitud x_r, y_r ja z_r.

4.2.4 Nägemisulatuse arvutamine

L-putis on üheks lahendatavaks probleemiks küsimus, kas väljalt A on näha välja B. Leidub kahte erinevaid tüüpi välju: nägemisulatust piiravad väljad ja väljad mis nägemusulatust ei piira. Ühelt väljalt on võimalik näha teisele, kui nende vaheline joon ei läbi ühtegi nägemisulatust piiravat välja. Selleks et teada saada, kas väljalt A on näha välja B leitakse kõik väljad, mis asuvad nendevahelisel joonel, kasutades selleks kahe välja vahelise joone leidmise algoritmi. Kui nende väljade seas ei ole ühtegi nägemisulatust piiravat välja, siis on väljalt A näha välja B, vastasel juhul nähtavus puudub.

5 L-puti tehisintellekt

L-putis implementeeritud tehisintellektid põhinevad kahel erineval algoritmil: minimax ja Monte Carlo. Lisaks on implementeeritud neli erinevat hinnangufunktsiooni, mida erinevad tehisintellekti variatsioonid kasutavad. Järgnevas peatükis kirjeldatakse esiteks neid hinnangufunktsioone, seejärel antakse ülevaade minimax ja Monte Carlo algoritmidest ning lõpuks võrreldakse erinevate algoritmide ja hinnangufunktsioonide kombineerimisel saadavate tehisintellektide sooritust L-putis.

5.1 Hinnangufunktsioonid

Hinnangufuntksiooni eesmärgiks on teha võimalikuks erinevate mängulaua seisude võrdlemine. Hinnangufuntsioon saab sisendiks mõne laua seisu ning tagastab reaalarvu. See reaalarv on hinnang mängijatevahelisele tasakaalule. Võrdset seisu tähistab null, negatiivsed seisud tähistavad ühe mängija eelist ja positiivsed seisud tähistavad teise mängija eelist. Eelise suurust näitab seisu väärtuse absoluutväärtus.[Asu15]

L-putis on implementeeritud neli erinevat hinnangufunktsiooni. Neist lihtsaim on **elupunktide hinnang**, mis arvestab ainult erinevate nuppude täis-elupunkte. Täis-elupunktid on korrutis nupu elupunktidest ja võitlejate arvust, millest on maha lahutatud eelmise rünnaku jääkkahju. Hinnangufunktsioon liidab kokku kõikide ühe mängija nuppude täis-elupunktid ja lahutab sellest kõikide vastasmängijate nuppude täis-elupunktid. Saadav väärtus ongi hinnang laua seisule.

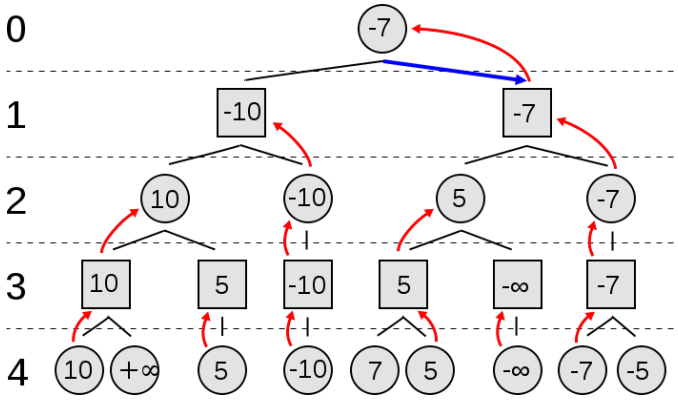
Teiseks hinnangufunktsiooniks on **kahjuvõime hinnang**. Igale nupule arvutatakse selle keskmine kahjuvõime. Selleks kasutatakse kahju tekitamise valemit ehk valemit 1, kus kaitseja kaitsemisoksuseks võetakse kõikide vastaste nuppude keskmine kaitseoskus. Sarnaselt elupunktide hinnangule liidetakse kokku kõikide ühe mängija nuppude keskmine kahjuvõime ja lahutatakse sellest kõikide vastasmängija nuppude keskmine kahjuvõime. Saadavat väärtust kasutatakse laua seisu hinnanguna.

Nii elupunktide hinnangule kui kahjuvõime hinnangule on implementeeritud ka variant mis võtab lisaks arvesse nuppude paigutust mängulaual. Seda hinnangufunktsiooni variatsiooni kutsutakse **positsiooni arvestavaks hinnanguks**. Selle hinnangufunktsiooni puhul väärtustatakse esiteks seis laual tugevushinnanguga, milleks kasutatakse ka elupunktide või kahjuvõime hinnangut. Seejärel lisatakse hinnangule positsioonihinnang, mis muudab ainult võrdsete tugevushinnangutega seisude hinnangute järjestusi. Positsioonihinnangu puhul eeldatakse, et parimad kohad mängulaual asuvad laua keskel, sest need võimaldavad liikumiseks suurima hulga erinevaid käike. Seega kasutatakse positsioonihinnangu arvutamiseks mängija nuppude keskmist kaugust väljaku keskosast.

5.2 Minimax algotitm

Käigupõhise mängu erinevate võimalike käikude põhjal on võimalik konstrueerida selle mängu otsuste puu. Juurtipuks esindab mängu algseisu. Juurtipp ühendatakse uute tippudega, mis esindavad kõiki võimalikke seise peale mõne legaalse käigu sooritamist. Uusi tippe lisatakse, kuni leidub mõni tipp, millel ei ole esindatud mõnda legaalset käikudest. Saadud puu lehted esindavad mängu lõppseise. [Asu15]

Saadud otsuse puu peal on võimalik kasutada minimax algoritmi, et igal võimalikul laua seisul teha parim võimalik käik. Selleks väärtustatakse esiteks puu lehed vastavalt sellele kumma mängija võitu leht esindab. Ühte mängijat kutsutakse minimeerijaks ja tema võitu tähistavad lehed väärtustatakse negatiivse väärtusega. Teist mängijat kutsutakse maksimeerijaks ja tema võitu tähistavad lehed väärtustatakse positiivse väärtusega. Viiki tähistavad lehed väärtustatakse nulliga. Seejärel väärtustatakse mõni tipp A millest kõik väljuvad tipud, ehk tipud mis on tipu A-ga ühendatud ja asuvad juurtipust kaugemal kui tipp A, on juba väärtustatud. Väärtustamisel valitakse kas väljuvate tippude vähim või väljuvate tippude suurim väärtus vastavalt sellele, tipp A esindab minimeerija või maksimeerija käiku. Tippude väärtustamist jätkatakse, kuni kõik tipud on väärtustatud. [Asu15]



Joonis 6. piiratud sügavusega minimax algoritmi visualisatsioon [Nog06]

Eelnevas lõigus kirjeldatud minimax algoritmi rakendamine L-putil ei ole tänapäevase arvutusvõimsuse juures võimalik, sest L-putis on keskmine võimalike käikude arv liiga suur. Seega tuleb rakendada piiratud sügavusega minimax algoritmi. Piiratud sügavusega minimax algoritmi jaoks kontsrueeritakse otsuste puu osaliselt. Sügavus määrab mitu käiku juurtipust edasi läbi vaadatakse. Näiteks joonisel 6 kujutatud piiratud sügavusega minimax algoritmi puhul on otsuste puu konstrueeritud sügavuseni neli. Piiratud sügavuse puhul tekivad lehttipud, mis ei esinda mängu lõppseise. Nendele tippudele

hinnangu andmiseks kasutatakse hinnangufunktsioone.

Jägnevalt kirjeldatakse L-putis kasutatavat piiratud sügavusega minimax algoritmi implementatsiooni. Algoritm saab sisendiks seisu laual ja täisarvu d, mis tähistab otsingu sügavust.

1. Kui d on võrdne nulliga, siis kasutades hinnangufunktsiooni, väärtusta seis laual ja tagasta see.
2. Väärtusta kõik legaalsete käikude tulemusel tekkivad uued laua seisud. Selleks kasuta rekursiivselt igal uuel laua seisul minimax algoritmi, kus sügavust on ühe võrra vähendatud.
3. Tee kindlaks, kas parasjagu sooritab käiku minimeerija või maksimeerija.
4. Kui hetkel sooritab käiku minimeerija, tagasta käik, mis annab laua seisu mis väärtustati vähima väärtusega.
5. Kui hetkel sooritab käiku maksimeerija, tagasta käik, mis annab laua seisu mis väärtustati suurima väärtusega.

Minimax algoritmi keskmine ajaline keerukus on $O(b^m)$, kus b on keskmine legaalsete käikude arv ja m on sügavus millel minimax algoritmi rakendatakse[Asu15]. L-putis on legaalsete käikude arvu varieeruvus suur, kuid hinnanguliselt on see umbes 50. Sellise käikude arvu puhul on juba sügavusel kolm konstrueeritavas otsuste puus üle saja tuhande tipu. Seega ei paku minimax algoritm piisavat võimekust, et langetada strateegilisi otsuseid.

5.2.1 Alfa-beeta kärpimine

Üheks viisiks minimax algoritmi optimaliseerida on alfa-beeta kärpimine. Alfa-beeta kärpimisel lisatakse minimax funktsiooni parameetriteks kaks väärtust: alfa ja beeta, mis tähistavad vastavalt hetkel suurima ja vähima väärtustuega tippu juurtipu ja hetkel kontrollitava tipu vahel. Alfa ja beeta väärtuste võrdlemine võimaldab lõpetada otsingu mõnes puu harus, mis ei saa mõjutada juurtipu väärtustamist. [Asu15]

Alfa-beeta kärpimine vähendab parimal juhul minimax algoritmi keerukust ruutjuure võrra vastavalt sellele kui mitmetes harudes otsing lõpetati. Harude kärpimise efektiivsemaks muutmiseks on võimalik käigud enne nende kontrollimist paremusjärjestusse seada. Paremusjärjestusse seadmiseks on võimalik kasutada hinnangufunktsiooni, sest kvaliteetne hinnangufunktsioon annab sarnaseid tulemusi minimax algoritmile.[Asu15]

Järgnevalt kirjeldatakse L-puti jaoks implementeeritud alfa-beeta kärpimisega minimax algoritmi. Algoritm saab sisendiks seisu laual, täisarvu d, mis tähistab otsingu sügavust ja väärtused alfa ja beeta milleks on esialgu minimaalne ja maksimaalne 32-bitise täisarvu väärtus.

1. Kui sügavus on 0 väärtusta seis laual kasutades hinnangufunktsiooni ja tagasta see.
2. Tee kindlaks, kas parasjagu sooritab käiku minimeerija või maksimeerija.
3. Järjesta kõik legaalsed käigud vastavalt hinnangufunktsioonile. Kui käiku sooritab minimeerija sea käigud kasvavasse järjekorda, kui maksimeerija siis kahanevasse järjekorda.
4. Itereeri üle kõikide võimalike legaalsete käikude.
   (a) Väärtusta itereeritav käik rekursiivselt. Edasta heltkel kehtivad alfa ja beeta väärtused.
   (b) Kui käiku sooritab maksimeerija, siis kui käigu väärtus on suurem, kui alfa, muuda alfa väärtus võrdseks käigu väärtusega.
   (c) Kui käiku sooritab minimeerija, siis kui käigu väärtus on väiksem, kui beeta, muuda beeta väärtus võrdseks käigu väärtusega.
   (d) Kui beeta väärtus on väiksem või võrdne alfaga, siis lõpeta itereerimine, vastasel juhul jätka itereerimist.
5. Kui hetkel sooritab käiku minimeerija, tagasta käik, mis annab laua seisu mis väärtustati vähima väärtusega.
6. Kui hetkel sooritab käiku maksimeerija, tagasta käik, mis annab laua seisu mis väärtustati suurima väärtusega.

5.3 Monte Carlo algoritm

Monte Carlo algoritm on mittedeterministlik alternatiiv minimax algoritmile. Monte Carlo algoritmi mittedeterminislikud elemendid võimaldavad selles sooritada sügavamaid otsinguid suure hargnevusega otsuste puudele. Samas ei pruugi algoritm tagastada parimat käiku, kuid on tõestatud et Monte Carlo algoritm koondub minimax algoritmiks. [wik17c]

Monte Carlo algoritmi saab sisendiks mõne mängulaua seisu, millest jooksutatakse suure sügavusega juhuslike käikudega mänge ehk simulatsioone. Kõikide võimalike algkäikude kohta peetakse statistikat, mitu mängu on vastav algkäik viinud võiduni ja mitu kaotuseni. Simulatsioonide jooksutamisel eelistatakse algkäiku, millel on seni suurim võitude osakaal. Lõpuks tagastataksegi käik, millel on parim võiduvõimalus. [Bra15]

Järgnevalt kirjeldatakse L-putis kasutatavat Monte Carlo algoritmi, mis saab sisendiks arvulised väärtused k ja n, mis kirjeldavad vastavalt simulatsiooni sügavust ja simulatsioonide arvu.

1. Soorita kõikide võimalike algkäikudega simulatsioon ja salvesta kas need viisid võidu või kaotuseni.

2. Simulatsiooni sooritamiseks tee k juhuslikku käiku ja kasuta hinnangufunktsiooni, et selgitada kumb mängija simulatsiooni võitis.

3. Soorita n simulatsiooni kasutades algkäiguks seni parima võitude osakaaluga leitud algkäiku.

4. Tagasta parima võitude osakaaluga algkäik.

Monte Carlo algoritmi ajaline keerukus on $O(n * k)$, kus n on simulatsioonide arv ja k on simulatsioonide sügavus. Seega ei sõltu selle keerukus otsingupuu hargnevusest, küll aga vähendavad suure hargnevusega otsingupuud selle algoritmi täpsust.

5.4 L-putis implementeeritud tehisintellektide võrdlus

L-putis implementeeritud tehisintellektide eesmärgiks on see, et nad sooritaks oma käigu mõistliku aja jooksul, see tähendab vähem kui kümne sekundiga. Selline ajapiirang on valitud selleks, et kui tehisintellekt mängib inimese vastu ei peaks inimene ootama. Alfa-beeta kärpimisega minimax algoritm, millel üks tehisintellektidest põhineb on seega implementeeritud sügavusel kaks.

Minimax algoritmist on implementeeritud kaks versiooni. Neist esimene kasutab hinnangufunktsioonina positsiooni arvestavat elupunktide hinnangut ning teine keerulisemat positsiooni arvestavat kahjuvõime hinnangut.

Tabelis 8 on võrreldud kahe minimax algoritmil põhineva tehisintellekti sooritusi teineteise vastu. Tabeli ridades olev algoritm on sinine mängija ja veergudes olev algoritm on punane mägija. Null tähistab punase mängija võitu ja üks tähistab sinise mängija võitu. Nagu tabelist näha, siis edukam on lihtsam elupunktide hinnangufunktsioon. See on mõnevõrra üllatav tulemus, kuid sellest võib järeldada, et keerulisem hinnangufunktsioon ei pruugi alati anda paremat tulemust.

	elupunktide hinnang	**kahjuvõime hinnang**
elupunktide hinnang	0	1
kahjuvõime hinnang	0	0

Tabel 8. Minimax algoritmil põhinevate tehisintellektide võrdlus

L-putis implementeeritud Monte Carlo algoritm kasutab hinnangufunktsioonina ainult positsiooni arvestavat elupunktide hinnangut, kuid lisaks on implementeeritud ahne Monte Carlo algoritm, mis esiteks väärtustab hinnangufunktsiooniga kõik algkäikudega

saadavad laua positsioonid ja seejärel käivtab simulatsioone neist kuni kümnel parimal. Mõlemad versioonid Monte Carlo algoritmist simuleerivad 100 mängu sügavuseni 10.

Tabelis 9 on võrreldud omavahel erinevaid positsiooni arvestava elupunktide hinnangufunktsiooni kasutavaid tehisintellekte. Kõikide algoritmide paaridega tehti läbi 10 mängu. Tabeli ridades olev algoritm on sinine mängija ja veergudes olev algoritm on punane mägija. Tabeli väljades olevad arvud tähistavad sinise mängija võitude osakaalu ehk arvu 1 korral võitis alati sinine ning arvu 0 korral alatu punane mängija.

	minimax	**Monte Carlo**	**ahne Monte Carlo**
minimax	0	1	1
Monte Carlo	0	0,5	0,4
ahne Monte Carlo	0	0,6	0,6

Tabel 9. Minimax algoritmi ja Monte Carlo algoritmide võrdlus

Selgelt on näha, et minimax algoritm on tugevam kui Monte Carlo algoritm. Minimax võitis kõik mängud mõlema Monte Carlo algoritmi vastu. Kuna katsete hulk on väike, ei saa teha väga kindlaid järeldusi. Sellele vaatamata võib oletada, et kuna Monte Carlo algoritmid sooritasid teineteise vastu küllaltki võrdsed tulemused, olenemata sellest kas mängiti siniste või punaste nuppudega, et algoritmid mängivad peaaegu juhuslike käikudega.

Monte Carlo algoritmide juhuslikke käigud tulenevad ilmselt väikesest simulatsioonide arvust. Simulatsioonide arvu suurendamine ei ole aga ajapiirangu tõttu võimalik. Piiravaks teguriks saab legaalsete käikude leidmine suvalisel laua positsioonil, mille keerukust võib lugeda küll konstantseks, kuid see on siiski liiga suur, et piiratud ajaga leida käigud tuhandele erinevale laua positsioonile.

6 Kokkuvõte

Töö tulemuseks valmis Unity mängumootoril põhinev käigupõhine strateegiamäng L-put, mida saavad omavahel mängida nii kaks kasutajat kui ka kasutaja ja tehisintellekt. Samuti on üks osa valminud tööst kokkuvõte kuusnurksetele väljadele mõeldud koordinaatsüsteemidest ja neid kasutatavatest algoritmidest.

Töö käigus võrreldi ka minimax ja Monte Carlo algoritmi. Töö käigus läbiviidud testimine näitas, et kumbki algoritm L-puti laadsele mängule tehisintellekti loomiseks ei sobi. Peamisteks takistavateks teguriteks on suur käikude hargnevus ja kõikide võimalike korrektsete käikude leidmise keerukus.

L-puti jaoks on võimalik tehisintellekti edasi arendada peamiselt hinnangufunktsiooni täiendamise kaudu. Põhiline hinnangufunktsiooni nõrkus millele saab keskenduda on nuppude omavahelise positsiooni hindamine. Teiseks on võimalik edasi uurida erinevaid Monte Carlo algoritmide variatsioone, mis võivad anda paremaid tulemusi, kui töö käigus implementeeritud variatsioon. Näiteks üheks selliseks Monte Carlo variatsioonist on *Upper Confidence Bound* meetodit kasutav Monte Carlo algoritm.

Viidatud kirjandus

[Asu15] Marit Asula. Strateegia ja mängupositsiooni hindamine juhuslikkuse elementi sisaldavas kahe isiku mängus. Bakalaureusetöö, Mai 2015.

[Bra15] Jeff Bradberry. Introduction to monte carlo tree search. https://jeffbradberry.com/posts/2015/09/intro-to-monte-carlo-tree-search/, Spetember 2015. Vaadatud: 2017.05.11.

[her99] Heroes of might and magic iii player manual. http://heroes3wog.net/download/[Heroes%203]%20Restoration%20of%20Erathia%20Manual.pdf, Veebruar 1999. Vaadatud: 2017.05.11.

[Nog06] Nuno Nogueira. Minimax algorithm. https://commons.wikimedia.org/wiki/File:Minimax.svg, Detsember 2006. Vaadatud: 2017.05.11.

[Pat15] Amid Patel. Hexagonal grids. http://www.redblobgames.com/grids/hexagons/, Märts 2015. Vaadatud: 2017.05.11.

[Wei] Eric Weisstein. Regular tessellation. http://mathworld.wolfram.com/RegularTessellation.html. Vaadatud: 2017.05.11.

[wik17a] Age of wonders iii. https://en.wikipedia.org/wiki/Age_of_Wonders_III, Märts 2017. Vaadatud: 2017.05.11.

[wik17b] Heroes of might and magic. https://en.wikipedia.org/wiki/Heroes_of_Might_and_Magic, Mai 2017. Vaadatud: 2017.05.11.

[wik17c] Monte carlo tree search. https://en.wikipedia.org/wiki/Monte_Carlo_tree_search, April 2017. Vaadatud: 2017.05.11.

Lisad

I. L-puti mänguprogramm

Töö käigus loodud programm, mille abil on võimalik mängida L-puti teise inimese või mõne loodud tehisintellekti vastu.

II. L-puti Unity projekt

L-puti programmi lähtekood ja abifailid.

III. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, **Mattias Lass**,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose

   Tehisintellekti loomine käigupõhisele strateegiamängule L-put

   mille juhendaja on Jaanus Jaggo

   1.1 reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace-is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;

   1.2 üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace´i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.

3. 3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tartus, 11.05.2017