TARTU ÜLIKOOL

Arvutiteaduse instituut

Informaatika õppekava

Erik Tarelkin

Vastaste liikumise arendamine

Blastronaut mängule

Bakalaureusetöö (9 EAP)

Juhendaja: Jaanus Jaggo, MSc

Tartu 2021

Vastaste liikumise arendamine Blastronaut mängule

Lühikokkuvõte:

Käesolevas töös kirjeldatakse erinevaid vastaste liikumise viise mängudes ja luuakse kolm erinevat tüüpi vastase liikumist mängule Blastronaut. Seejärel hinnatakse valminud vastaste liikumise jõudlust ja vastaste sobivuse hindamiseks viiakse läbi kasutajakogemuse testimine. Saadud tulemuste põhjal selgitatakse välja, millistele liikumistüüpidele tuleb rohkem tähelepanu pöörata, et mäng oleks võimalikult huvitav.

Võtmesõnad:

arvutimängud, Godot Engine, vastaste liikumine, tehisintellekt, mängu testimine

CERCS: P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria)

Developing enemy movement for Blastronaut game

Abstract:

This thesis describes different enemy movement methods in video games. Three different enemy moving behaviours are created for Blastronaut game. Then the movement performance is evaluated and user experience is conducted to verify the suitability of created enemies. The types of movement, that need the most attetion are determined based on the results in order make the game as interesting as possible.

Keywords:

computer games, Godot Engine, enemy movement, artificial intelligence, game testing

CERCS: P170 Computer science, numerical analysis, systems, control

Sisukord

1. Sissejuhatus

Arvutimängude üheks eesmärgiks on pakkuda mängijale väljakutset. Üheks levinud viisiks, millega mängijale väljakutset pakkuda, on luua arvuti poolt juhitud vastased. Vastaste loomine jaguneb mitmeks osaks: liikumine, maailmaga interakteerumine, mängijaga interakteerumine jne. Käesolevas bakalaureusetöös uuritakse lähemalt erinevate vastaste liikumise meetodeid mängudes, seejärel kirjeldatakse kuidas Blastronaudile vastased loodi ja lõpuks hinnatakse vastaste sobivust sellesse mängu.

Blastronaut[1] on kõrvalt vaates kaevandamismäng, mille tegevus toimub protseduuriliselt genereeritud tulnukate maailmas. Blastronauti maailma on kujutatud joonisel 1. Mängija avastab tundmatut planeeti, kaevandab ressursse ja teenib nende müügiga raha, et uuendada oma varustust. Selle abil saab ta kaevata sügavamale maa alla, kus teda ootavad erinevad ohud, milleks on näiteks vastased. Mängule on arendatud koostööl põhinev mitmikosa, mis võimaldab seda võrgu teel koos sõpradega mängida. Blastronaut on loodud kasutades mängumootorit Godot Engine[2] versiooni 3.2. Hetkel arendavad mängu erinavaid osi kolm inimest.

Joonis 1. Blastronaut mängu maailm.

[1]Blastronaut https://store.steampowered.com/app/1392650/BLASTRONAUT/

[2]Godot Engine https://godotengine.org/

Käesoleva lõputöö eesmärgiks on luua mängule kolm erinevat vastase liikumist ja hinnata nende sobivust Blastronaut mängu. Nendeks liikumise tüüpideks on lendamine, ronimine ja kõndimine.

Teises peatükis kirjeldatakse erinevaid vastaste liikumise meetodeid mängudes. Kolmandas peatükis antakse ülevaade komponentide põhisest arhitektuurist, seejärel kirjeldatakse vastaste liikumise komponentide loomist mängus Blastronaut. Neljandas peatükis hinnatakse vastaste sobivust Blastronaut mängule ja analüüsitakse kasutajate testimise tulemusi.

2. Vastaste liikumiste tüübid mängudes

Käesolevas peatükis antakse ülevaade enamlevinud liikumiste tüüpidest mängudes. Nendeks tüüpideks on:

1) füüsikamootoril põhinev liikumine,
2) navigatsiooni võrestikul põhinev liikumine ,
3) ruudustikul põhinev liikumine,
4) tüürimiskäitumisel põhinev liikumine,
5) pöördkineetikal põhinev liikumine.

Järgnevates alapeatükkides kirjeldatakse igat liikumise tüüpi lähemalt ning tuuakse näiteid mängudes, kus sellist liikumist rakendatud on.

2.1 Füüsikamootoril põhinev liikumine

Iga inimene puutub kokku füüsikaga igapäevaselt, olgu selleks kõndimine, jooksmine või palliga mängimine. Enamik tänapäevaseid mängumootoreid sisaldab füüsikamootorit, mida mänguarendajad saavad oma mängudes kasutada. Näiteks Godot mängumootor pakub 3D mängude jaoks Bullet[3] füüsikamootorit ning 2D mängude jaoks oma enda loodud füüsikamootorit. Arendaja saab oma mängule määrata füüsikareegleid, mille vaikimisi seadistus kujutab endast päris maailmaga sarnanevat füüsikat, tuntud kui Newtoni mehaanika.

Slime Rancher[4] on tuleviku talupidamise mäng, kus taluloomade asemel peab mängija lima loomi. Nende loomade liikumine on simuleeritud füüsika abil. Mängija saab neid oma vaakum-tööriistaga kinni püüda ja hiljem uuesti välja tulistada. Tänu füüsikamootori kasutamisele põrkavad loomad maailmas ringi ja võivad tekitada kaootilisi olukordi. Mängu on lisatud ka mehaanikaid, mis neid olukordi võimendavad, näiteks mõned lima loomad võivad teistega kokkupuutel plahvatada. Selliste situatsioonide lahendamine muudab mängu põnevaks. Joonisel 2 on pilt Slime Rancher

[3]Bullet https://godotengine.org/article/godot-30-switches-bullet-3-physics
[4]Slime Rancher https://store.steampowered.com/app/433340/Slime_Rancher/

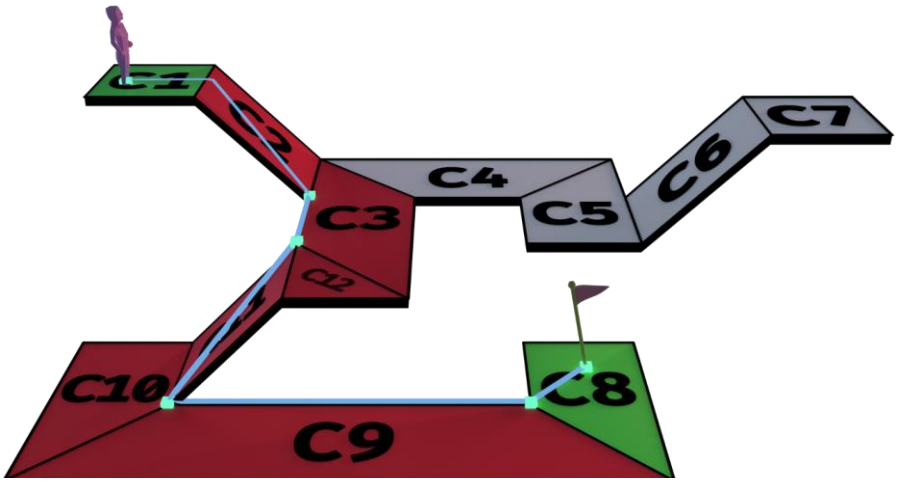
mängust, kus on mängija on lima loomad aedikusse kinni pannud, kuid füüsika abil saavad nad sealt vahel välja hüpata. [1]

Joonis 2. Slime Rancher [1].

2.2 Navigatsiooni võrestikul põhinev liikumine

Navigatsiooni võrestik on ruumi jaotus võreks, mille tahkudeks on kumerad hulknurgad. Iga tahk määrab graafi tipu, mis on omavahel ühenduses, kui need tahud külgnevad teineteisega. [2]

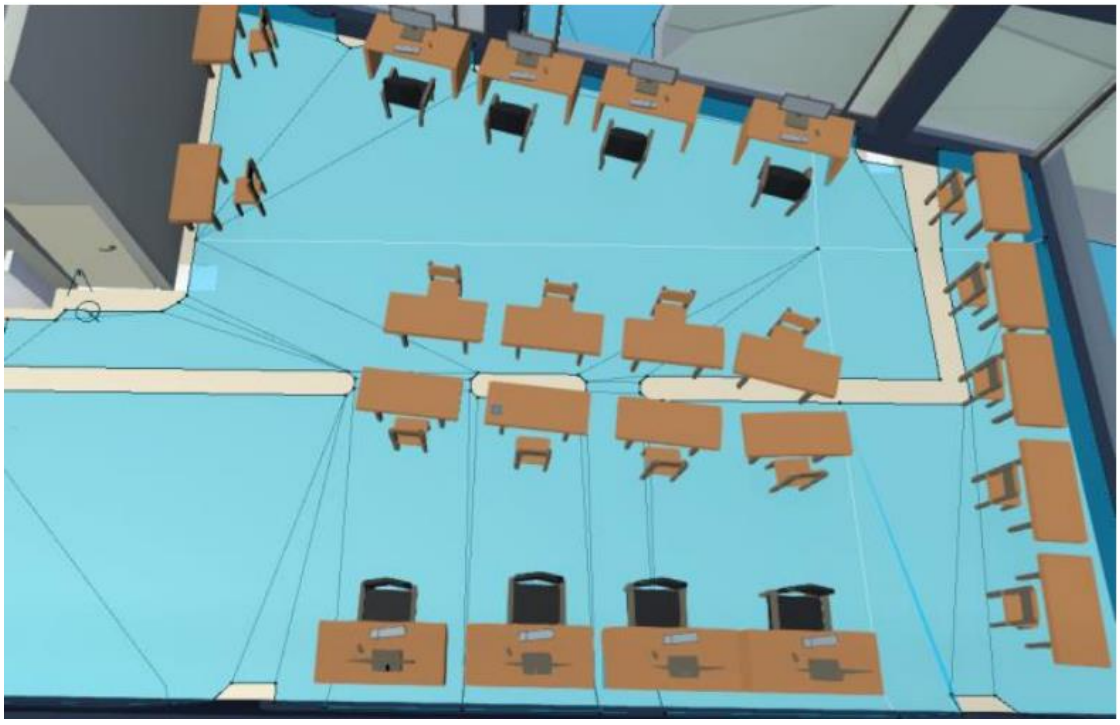
Joonisel 3 on kujutatud kolmemõõtmeline navigatsiooni võrestik. Joonisel on näha, kuidas ruum on jaotatud hulknurkadeks, rohelisega on tähistatud algus- ja lõpp-tahud ja punasega on tähistatud tahud, mis jäävad lühima tee peale.



Joonis 3. Kolmemõõtmeline navigatsiooni võrestik [3].

Lühima tee saab leida otsimisalgoritmi abil, näiteks Dijkstra või A* otsimisalgoritmiga. Kuna kõik hulknurgad on kumerad, saab lihtsalt leida ka lühima teekonna iga tahu sees, mida on joonisel kujutatud sinise joonega. Sellise otsimisalgoritmi kasutamine on väga tõhus ja efektiivne eelkõige mängudele, mille maailm on eelgenereeritud, sest neile saab navigatsiooni võrestiku eelnevalt välja arvutada. Dünaamiliselt genereeritud maailmadele, nagu on mängus Blastronaut, ei ole navigatsiooni võrestiku kasutamine mängumootori poolt toetatud.

Joonisel 4 on kujutatud Meelis Perli lõputöös "Delta õppehoone visualisatsioon – agentide loogika" [4] toodud pilt navigatsiooni võrestiku kasutamisest. Meelis kasutas navigatsiooni võrestikku, et simuleerida agentide liikumist Delta õppehoones. Ruum on jaotatud erineva kuju ja suurusega sinisteks hulknurkadeks, mis vastavad navigatsiooni võrestiku tahkudele ja mille peal saavad agendid liikuda.



Joonis 4. Navigatsiooni võrestik Delta õppehoones [4].

2.3 Ruudustikul põhinev liikumine

Paljudes mängudes on mängumaailma võimalik kujutada ruudustikuna. Blastronaut mängus olev maailm koosneb ruudustikku paigutatud plokkidest. Kui mängu tegelased saavad liikuda ruutude vahel, ühe ruudu keskpunktist teise keskpunkti, saab seda liikumist käsitleda graafina. Graafi

tippudeks on ruutude keskpunktid ja kahe külgneva ruudu tipu vahel on kaar, kui nende vahel saab liikuda. Lühima tee leidmiseks saab taaskord kasutada otsimisalgoritme nagu Dijkstra või A*.

Ruudustiku põhjal liikumist on kasutatud näiteks mängus Diablo 1, mida on näidatud joonisel 5. Tegemist on klassikalise *hack and slash* tüüpi mänguga, mis tuli välja aastal 1997 ja pani aluse uuele mängužanrile. Mäng koosneb tasemetest, kus iga tase koosneb 40x40 tükist koosnev isomeetriline ruudustik. Ühes ruudus saab korraga olla üks tegelane ja tegelased saavad liikuda ainult ühest ruudu keskpunktist teise. [5]

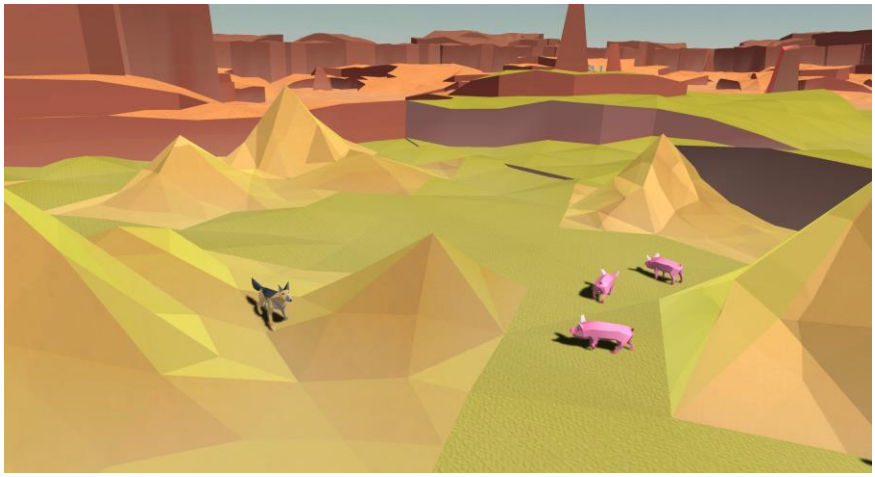
Joonis 5. Diablo mängu koobas [5].

2.4 Tüürimiskäitumisel põhinev liikumine

Tüürimiskäitumine kasutab tegelaste liigutamiseks üksteist mõjutavate jõudude simulatsiooni. Selle eesmärgiks on luua loomuliku välimusega liikumine, kasutades lihtsaid strateegiaid. Selle kõige olulisemaks elemendiks on liikumisvektor, mis määrab liikumise suuna ja kiiruse ning erinevad välised jõud saavad seda mõjutada. Mängudes rakendatakse tüürimiskäitumist enamasti kolmel viisil:

1. Otsimine on selline käitumine, kus on teada sihtpunkt ja liikumisvektorit mõjutatakse niiviisi, et see joondaks ennast sihtpunkti suunas.
2. Jälitamine on otsimisega analoogine käitumine, kus tegelase sihtpunktiks on teine liikuv tegelane. Samuti võib tegelane ennustada oma sihtpunkti positisooni tulevikus ja liikuda vastavalt ennustusele eeldades, et teine tegelane ei muuda vahepeal enda suunda ega kiirust.
3. Takistuste vältimine on vastandlik käitumine jälitamisele. Kui tegelane tuvastab läheduses oleva objekti, millega ta tahab kokkupuudet vältida, siis ta hoidub sellest objektist eemale. [6]

Tüürimiskäitumisel põhinevat liikumist on kasutatud näiteks mängus Herding Dog[5], mida on näidatud joonisel 6. Herding Dog on mäng, kus mängija tegelaseks on karjakoer, kelle eesmärgiks on avastada erinevaid põllumaid ja kaitsta oma karja maailmas leiduvate ohtude eest. Joonisel on näha mängu kahte erinevat sorti tegelast: koer ja siga. Siga alustab liikumist juhuslikult valitud suunas. Kui siga leiab enda lähedalt teise sea, siis hakkavad nad teineteise poole liikuma. Kui mitu siga niimoodi lähestikku satuvad, siis moodustavad nad karja. Juhul kui sea lähedusse satub sein või mingisugune oht, näiteks koer, siis üritab siga antud objekti vältida.



Joonis 6. Herding Dog.

2.5 Pöördkineetikal põhinev liikumine

Edasi kinemaatik[6] (*forward kinematics*) on matemaatiline meetod, millega arvutatakse objekti jäsemete lõpp-positsiooni, nii et on teada jäsemete segmentide pikkused ja nende vahelised nurgad.

[5]Herding Dog https://store.steampowered.com/app/409290/Herding_Dog/
[6]Edasi kinemaatika https://en.wikipedia.org/wiki/Forward_kinematics

Pöördkineetikas (*inverse kinematics*) tehakse võrreldes edasise kinemaatikaga vastupidised arvutuseda, ehk kui on teada jäseme lõpp-positsioon ja segmentide pikkused, siis arvutatakse nende põhjal segmentide vahelised nurgad. [7]

Pöördkineetikal põhinevat liikumist on kasutatud näiteks seiklusmängus Shadow of the Colossus[7], mida on kujutatud joonisel 7. Joonisel on näha, kuidas tegelane hoiab ühe käega kinni kolossuse vööst. Sellisel juhul on fikseeritud käega kinni hoidmise positsioon ja ülejäänud keha liigutatakse selle suhtes. Segmentide vaheliste rotatsioonide arvutamiseks kasutatakse pöördkineetikat.

Joonis 7. Shadow of the Colossus.

2.6 Blastronaut mängule sobiva liikumismeetodi valimine

Blastronaut mängule sobiva liikumismeetodi valimisel arvestati järgmiseid nõudeid:

1. Vastased peavad olema suutelised arvutama oma liikumist mängu käigus, sest vastased liiguvad dünaamiliselt genereeritud maailmas ja see tähendab, et nad ei saa kasutada eelgenereeritud navigatsiooni andmeid.
2. Vastaste komponendid ei tohi mängu jõudlust negatiivselt mõjutada. See tähendab, et kui stseenis olevate vastaste koguarv on 200, siis ei tohi mängu renderdusaeg olla aeglasem kui 20 millisekundit.

[7]Shadow of the Colossus https://en.wikipedia.org/wiki/Shadow_of_the_Colossus

3. Vastaste liikumine peab olema sünkroniseeritav mängijate vahel, sest sellel mängul on ka mitmikmängu võimalus. Selline lähenemine piirab füüsika rakendamist, sest füüsikast tunduvalt rohkem sõltuvate objektide sünkroniseerimine erinevate mängijate vahel on keeruline ja raskendatud.
4. Vastaste liikumine ei tohi välja näha robustne ja see peab peegeldama võimalikult hästi päris loomade liikumist.

Eelnevalt kirjeldatud liikumise meetoditest sobivad Blastronaut mängule kõige paremini tüürimiskäitumisel ja pöördkineetikal põhinevad meetodid, sest just need meetodid vastavad kõige rohkem nõuetele. Liikumise aluseks sai implementeeritud tüürimiskäitumisel põhinev süsteem, sest see võimaldab luua hästi loomulikku liikumist. Maa peal liikuvatele vastastele lisati hiljem pöördkineetikast inspireerituna jalgadel põhinev liikumissüsteem.

3. Vastaste arendamine

Vastaseid arendati mängumootoris Godot Engine (versioon 3.2). Godot mootori arhitektuur on üles ehitatud tippude (*Node*) puu kontseptsiooni ümber. Mängu stseenid koosnevad tippudest, kus igale tipule saab külge panna skripti, mis defineerib tema käitumise ja vastab objektorienteeritud programmeerimise klassile. Tippe defineerides saab luua komponentide põhise süsteemi. [8]

Peatükis 3.1 Komponentide põhine süsteem esitatakse lühike ülevaade komponentide põhisest süsteemist Blastronaut mängus. Peatükkides 3.2 ,3.3 ja 3.4 kirjeldatakse lõputöö raames loodud komponente, millest vastased koosnevad.

3.1 Komponentide põhine süsteem

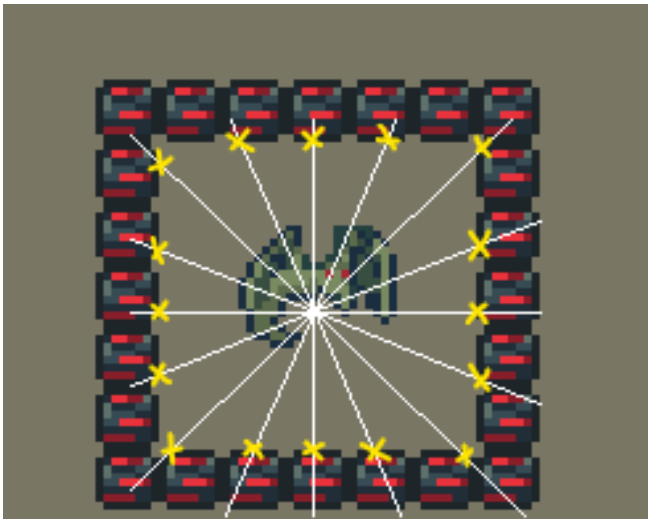
Objektorienteeritud programmeerimises jagatakse programm mitmeteks väikesteks tükkideks - komponentideks. Komponent on korduvkasutatav programmi osa, mis ei ole teistest komponentidest sõltuv. Komponente kombineerides saab luua komponentide põhise süsteemi. Sellise süsteemi mõte seisneb selles, et igal väiksemal osal on oma ülesanne, mis teisi osasid ei mõjuta. Kui objekt on jaotatud mitmeks väikeseks komponendiks, siis on vaja need omavahel siduda. Neid komponente juhib vastase peaklass, mis juhib oma alamkomponentide tööd. [9]

Joonisel 8 on näha roniva vastase klassi. Peamised komponendid on siin liikumise komponent (*MovementComp*), jala komponendid (Leg1, Leg2, Leg3), kiirte jälitamise komponent (*QueryComp*) ja neid siduv peaklass *CrawlingEnemy*.

Joonis 8. Roniva vastase peaklass.

3.2 Kiirte jälitamise komponent

Enne lendava vastase liikumise implementeerimist, tuli luua süsteem, mis aitab vastasel tuvastada kindlas raadiuses olevaid takistusi. Takistuste kuvamiseks kiirgab vastane enda ümber igas suunas kiiri, kattes nii 360-kraadise välja. Joonisel 9 on näidatud 16 kiirega kaetud ala. Neid kiiri



Joonis 9. Lendava vastase 360-kraadine väli.

kiiratakse ühekaupa, üks kiir ühes kaadris. Kollase ristiga on kujutatud kiirte kokkupõrkekohad seinaga. Kokkupõrkel objektiga salvestatakse kokkupõrke asukoht, suund ja takistuse tüüp puhvrisse, kust seda hiljem küsida saab.

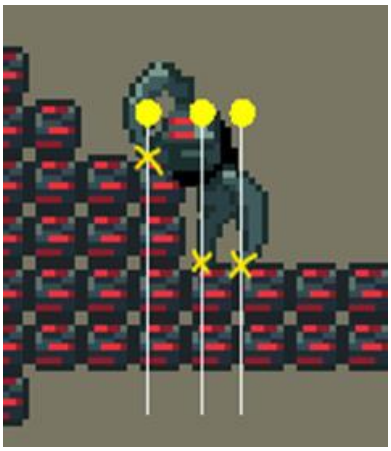
Selleks, et vastane oskaks takistusi vältida, loodi päringu meetod. See meetod vaatab läbi kõik kiired ja võtab arvesse need kiired, mille kokkupõrkepunkt on vastasele lähemal kui ette antud vältimiskaugus. Nende kiirte kaalud summeeritakse ja saadud tulemust kasutatakse selleks, et mõjutada liikumisvektorit vastassuunas.

3.3 Jala komponent

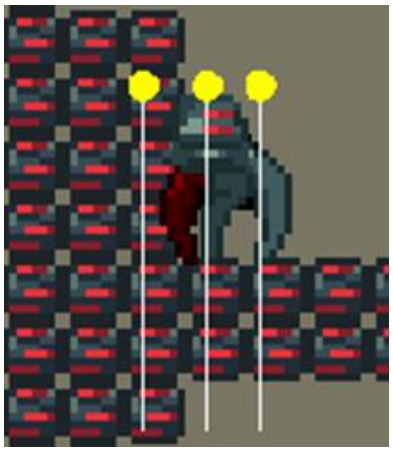
Üheks peamiseks komponendiks on jala komponent, mis vastutab jala liikumise eest. Komponendis on uuendamise meetod, mida kutsutakse iga kaader. See meetod saab ette liikumise vektori ja liigutab jalga liikumise vektori võrra tagasi. Kuna vastane liigub samal ajal liikumise vektori võrra edasi, siis selle tulemusena jääb jala asukoht maailma suhtes paigale. Kui jalg on liikunud piisavalt palju tagasi, siis tõstetakse see jalg sammu võrra ettepoole. Selle käigus tehakse ka kiire jälitus allasuunas, et teha kindlaks kui kõrgel on maapind ja tõstetakse jalg sellele kõrgusele. Seda kiire jälitamist on kujutatud joonisel 10, kus valge joonega on tähistatud kiir, kollane täpp on selle kiire alguspunkt ja kollane rist tähistab selle lõikepunkti maapinnaga. Lisaks on jalgadel erinevad seisundid: maas, õhus, kinni.

1. Kui jala kiire jälitamine õnnestus edukalt, siis märgitakse jala seisundiks maas.
2. Kui jala kiire jälitamise tulemusena lõikepunkte ei leitud, siis märgitakse jala seisundiks õhus.

3. Kui jala kiire jälitamise tulemusena saadud lõikepunkt on kohe kiire alguspunktis, siis järelikult on kiire alguspunkt maapinna sees ja jala seisundiks märgitakse kinni. Kinnises seisundis jalga on kujutatud joonisel 11.



Joonis 10. Jala kiire jälitamine.



Joonis 11. Jala seisundiks on kinni.

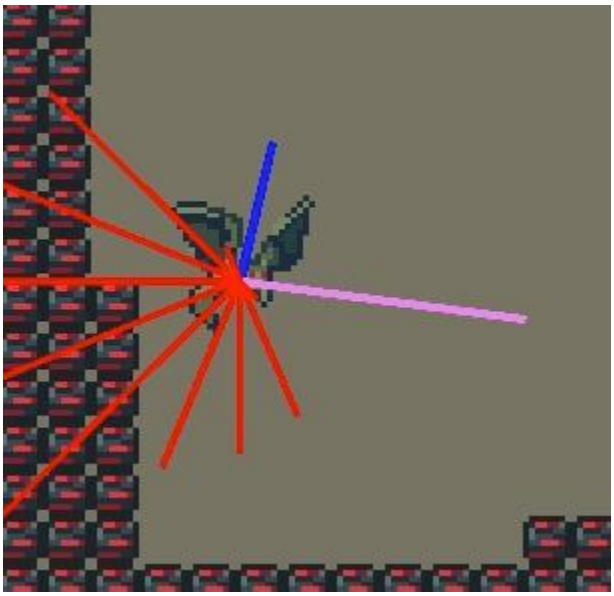
Vastase liigutamine toimub vastase liikumise komponendis, millest antakse ülevaade järgmises peatükis.

3.4 Liikumise komponent

Peamiseks vastase komponendiks on tema liikumise komponent. Liikumise komponent vastutab igasuguse liikumise eest maailmas. Sarnaselt eelmises peatükis kirjeldatud komponendile on ka liikumise komponendile implementeeritud liikumise suund.

3.4.1 Lendava vastase liikumise komponent

Lendava vastase liikumine kasutab kiirte jälitamise komponenti, et määrata kuhu suunas liikuda. Liikumiseks on kaks peamist vektorit, liikumise vektor, mida on joonisel 12 kujutatud sinise joonega ja sihtvektor, mida on kujutatud roosa joonega. Lisaks on joonisel kujutatud punaste joontega vektorid, mis osutavad lähima seina poole. Need vektorid on arvutatud nii, et mida lähemal on sein vastasele, seda pikemad need vektorid on. Iga fikseeritud aja tagant genereeritakse uus sihtvektor nii, et esmalt valitakse juhuslik suund ja seejärel lahutatakse sellest punaste vektorite summa. See tagab, et lõplik suund hoiab alati punastest vektoritest eemale ja vastane väldib kokkupõrget seinaga.



Joonis 12. Lendava vastase liikumine.

Vastane ise liigub oma liikumise vektori suunas ja seda liikumise vektorit pööratakse omakorda sujuvalt sihtvektori suunas. Sel viisil jääb vastase liikumine alati sujuvaks, isegi kui sihtvektor äkitselt muutub.

Lisaks takistuste vältimisele on implementeeritud ka jälitamise strateegia. Juhul kui kiire jälituse komponent leidis mõne objekti, millel on jälitamise prioriteet, siis sihtvektoriks märgitakse hoopis kõrgeima jälitamise prioriteediga objekti suund. Sellisteks objektideks võivad näiteks olla vastased, mille tulemusena koonduvad need vastased gruppidesse. Joonisel 13 on kujutatud lendavate vastaste grupp.

Joonis 13. Lendavate vastaste grupp.

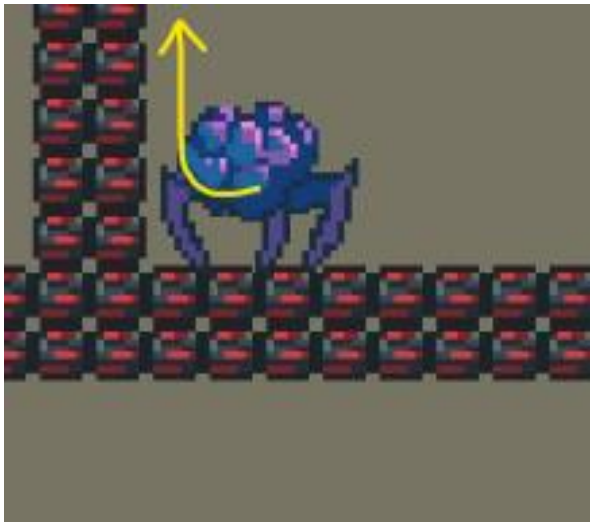
3.4.2 Jalgadega vastaste liikumise komponent

Nii kõndival kui ka ronival vastasel on ühine liikumise komponent. See komponent teab liikumise suunda horisontaalselt ning vertikaalne liikumine arvutatakse jalgade põhjal. Selleks arvutatakse kõigi jalgade keskmine vertikaalne nihe. Seejärel liigutatakse vastast vertikaalselt nii, et seda nihet minimeerida. Selle tulemusena liigub keha üles või alla, vastavalt sellele kas jalad astuvad üles või alla. Niimoodi saab vastane kõndida üles näiteks treppidest, sest esimesed jalad astuvad järjest kõrgemale ja nihke minimeerimiseks tõmbab keha ennast sellega kaasa.

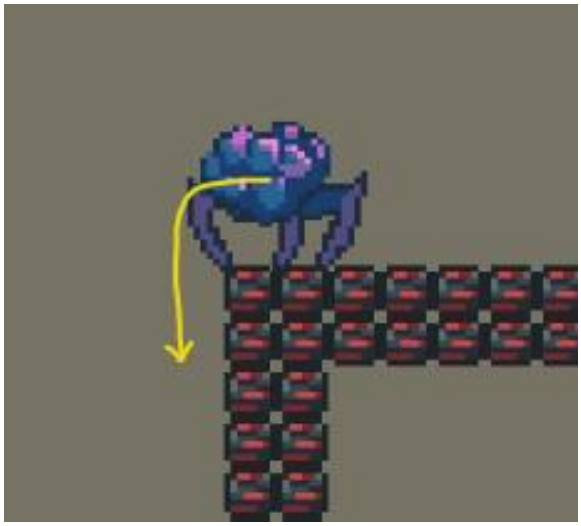
Samuti kontrollitakse igal sammul kõikide jalgade seisundit. Kui mõni jalgadest on õhus või seinas kinni, siis meetod jaguneb kaheks ja tuleb otsustada, kuidas seda probleemi lahendada. Selleks on kaks võimalust. Esimene võimalus on vastase suunda muuta ja hakata tuldud teed pidi tagasi

kõndima. Selline käitumine sobib kõndivale vastasele. Teiseks võimaluseks on pöörata tegelast 90 kraadi võrra ja ronida takistusest üle. Selline käitumine sobib ronivale vastasele.

Kui tegemist on roniva vastasega, siis algoritm kontrollib, kas vastane on seinas kinni või õhus. Juhul kui vastase jalg on seina sees kinni, siis vastane pöörab ennast 90 kraadi päripäeva ja jätkab liikumist. Kui vastase jala seisundiks on õhus, siis ta pöörab ennast 90 kraadi vastupäeva ja jätkab liikumist. Neid olukordi on kujutatud joonisel 14 ja joonisel 15.



Joonis 14. Roniv vastane pöörab 90 kraadi päripäeva.



Joonis 15. Roniv vastane pöörab 90 kraadi vastupäeva.

Kuna vastaseid genereeritakse maailma juhuslikult, peab kuidagi genereerimist stabiliseerima. Arendamise käigus avastati üks programmi viga: kui vastane genereerida õhku, siis ta satub tsüklisse ja jääb ennast lõpmatult keerama. Selleks implementeeriti vastasele meetod, mis kukutab vastase maa peale, juhul kui kõik vastase jalad on õhus.

4. Testimine ja tagasiside

Käesolev peatükk kirjeldab testimist ja kasutajate tagasisidet. Peatükis 4.1 kontrollitakse vastaste jõudlust vastavalt peatükis 2.6 püstitatud nõudele 2. Peatükis 4.2 antakse ülevaade kasutajate tagasisidest ja analüüsitakse saadud tulemusi.

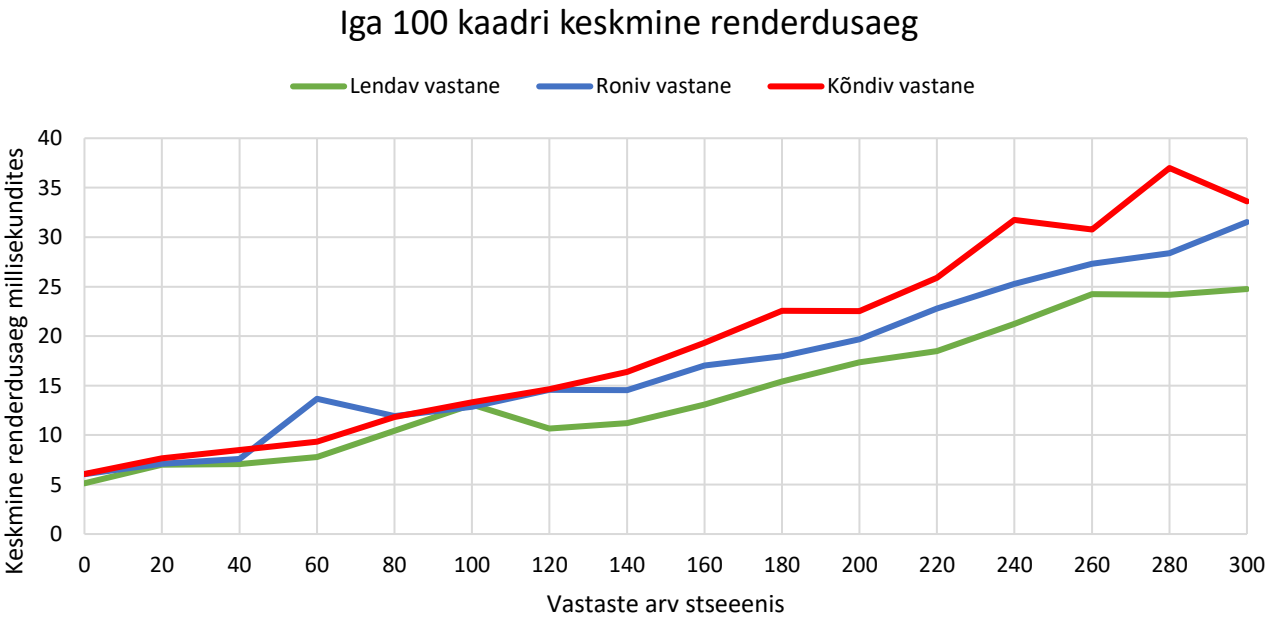
4.1 Jõudluse testimine

Peatükis 2.6 seati nõuded, mida vastaste liikumist arendades järgiti. Selles peatükis testitakse vastaste mõju jõudlusele ja kontrollitakse, et see rahuldaks neid nõudeid. Kõik jõudlustestid on tehtud kasutades järgmist riistvara:

- Protsessor: Ryzen 7 3700X
- Videokaart: NVIDIA GeForce RTX 2080 SUPER
- RAM: 16 GB

Nõude 2 kontrollimiseks loodi mängumaailma simuleeriv test stseen. Testimisel lisati vastaseid 20 kaupa iga 100 kaadri järel. Test kestis seni kuni stseenis olevate vastaste koguarvuks sai 300.

Testide tulemusi, erinevate vastaste lõikes, on näidatud joonisel 16. Lendava vastase keskmine renderdusaeg on märgitud roheliselt, roniva vastase oma siniselt ja kõndiva vastase oma punaselt.



Joonis 16. Vastaste jõudluse testide tulemused.

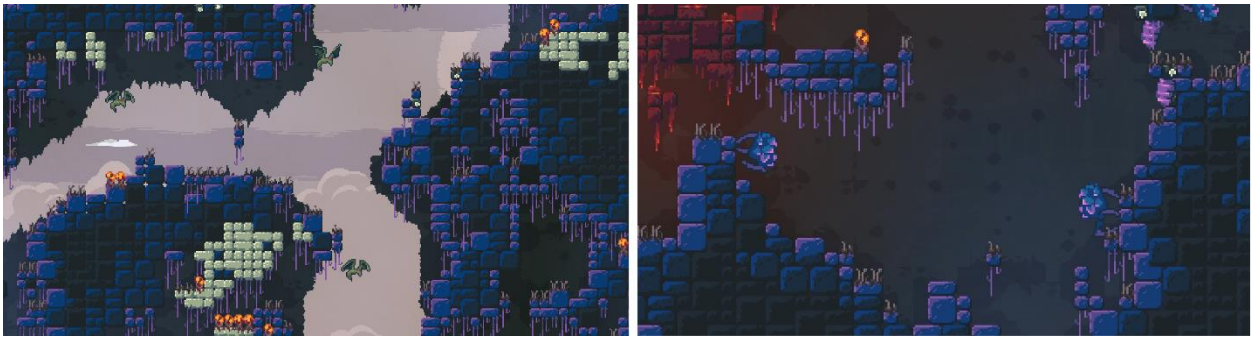
Võrreldes kolme joont olid tulemused üsna sarnased kuni stseenis oli kokku 100 vastast, see aeg oli umbes 13 millisekundit ± 0.5 ms. See tähendab, et nõue 2 on täidetud kõigi vastaste poolt.

Rohkemate vastaste korral tekib kaadrisageduses väike erinevus. Kõige parem tulemuse andsid lendavad vastased ja kõige halvema kõndivad vastased.

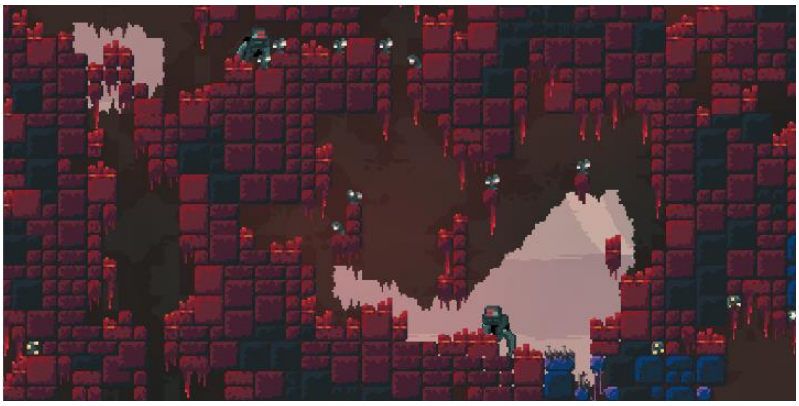
4.2 Kasutajate tagasiside

Kasutajate arvamuse väljaselgitamiseks loodi vastaste liikumiste kohta mängust kolm erinevat demo rakendust. Igas demo rakenduses sai jälgida ühte kindlat tüüpi vastast. Demo rakenduste järjekord oli järgmine: lendav vastane, roniv vastane, kõndiv vastane. Joonistel 17 ja 18 on kujutatud kuvatõmmised demo rakendustest. Kasutajate tagasiside saamiseks loodi küsimustik, mis on toodud Lisas II. Küsimustik sisaldas vajalikke faile ja juhiseid demo rakenduste kasutamiseks. Küsimustiku valimiks oli 12 tudengit. Kasutajad hindasid igat vastast eraldi neljas kategoorias:

1. liikumise loomulikkus,
2. liikumise huvitavus,
3. vastase suurus ja
4. vastase kiirus.



Joonis 17. Lendava ja roniva vastase demo ekraanitõmmised.



Joonis 18. Kõndiva vastase demo rakenduse ekraanitõmmis.

Esimesed kaks küsimust oli hinnangulised skaalal üks kuni neli. Esimeses küsimuses tähendasid numbrilised hinnangud järgmiseid omadusi: 1 - väga robustne, 2 - robustne, 3 - loomulik, 4 - väga loomulik. Pärast numbrilist hindamist paluti jagada oma muljeid, mis liikumise juures meeldis ja mis häiris. Teises küsimuses tähendasid numbrilised hinnangud järgmisi omadusi: 1 - väga igav, 2 - igav, 3 - huvitav, 4 - väga huvitav. Pärast numbrilist hindamist paluti põhjendada oma valikut. Lisaks paluti kasutajal kirjutada, kui ta märkas liikumise juures mingit viga. Vastase suurust hinnati kolme valikuga: liiga väike, paras või liiga suur. Vastase kiirust hinnati samuti kolme valikuga: liiga aeglane, paras või liiga kiire.

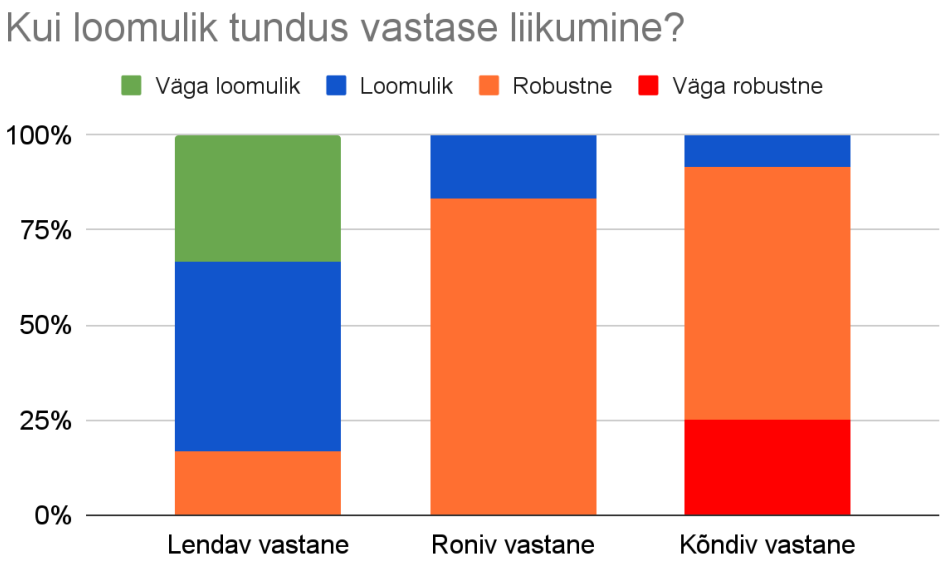
Küsimustiku lõpus küsiti veel kaks küsimust: mida kasutajad täiendaks vastase liikumise juures ja mis tüüpi vastaseid sooviksid nad veel näha.

Vastaste loomulikkuse hinnanguid on kujutatud joonisel 19. Kõige enam oldi rahul lendava vastase loomulikkusega, umbes üks kolmandik vastanutest hindas selle väga loomulikuks. Roniva ja kõndiva vastase hinnangud olid veidi kehvemad kui lendava vastase omad, enamus vastanutest arvas, et see on robustne või väga robustne.

Lendava vastase juures meeldis kasutajatele see, et vastane liikus sujuvalt ringi ja liikumine oli ettearvamatu. Üksikud kasutajad mainisid, et lendava vastase suuna ümberpööramine tundus veidi kohmakas.

Kasutajatele meeldis roniva vastase kontseptsioon ja tema astmeline astumine. Peaaegu kõik kasutajad nägid midagi ebaloomulikku vastase juures. Enim väljatoodud probleemid olid järgmised: üksteise peal ronimine, vastane ei saanud kitsastes nurkades hakkama, väiksemad

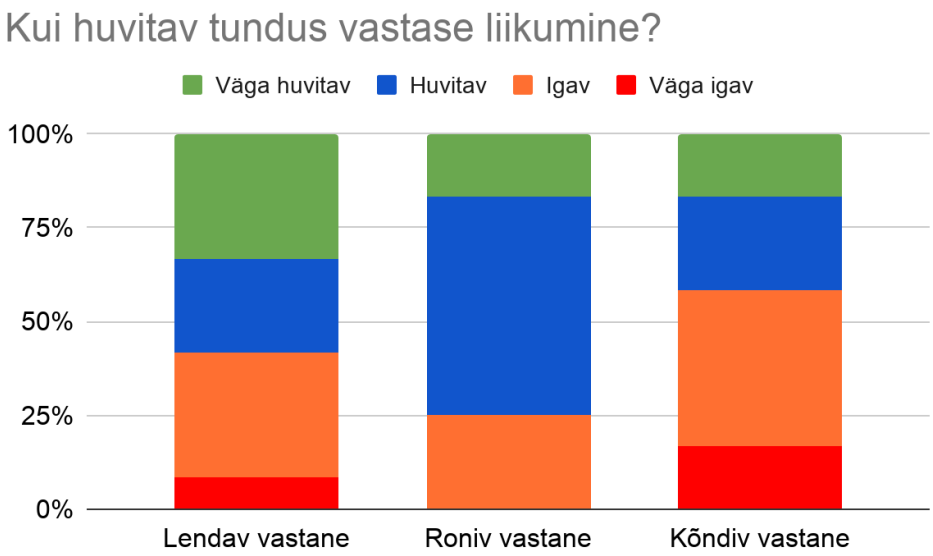
platvormid olid tülikad, ebaloomulik keha pööramine ja harva juhtus, et jalad olid kehast eraldatud.



Joonis 19. Vastaste loomulikkuse hinded.

Kõige negatiivsema tagasiside nende loomuliku liikumise kohta said kõndivad vastased. Mõned vastajad kirjutasid, et neile meeldis siledal platsil liikumine. Enamik kasutajaid märkisid, et kõndivate vastaste liikumine oli kolme vastase seast kõige robustsem, sest nende liikumises esines kõige rohkem vigast liikumist. Näiteks jäsemed tulid otsast ära ja mõned vastased liikusid õhus. Mitmed vastajad soovitasid lisada kõndivale vastasele hüppamise mehaanikat.

Vastaste liikumise huvitavuse hinnanguid on kujutatud joonisel 20. Kasutajad pidasid kõige huvitavamaks roniva vastase liikumist ja kõige igavamaks kõndiva vastase liikumist. Lendava vastase liikumine tundus kasutajate jaoks huvitav, sest nad olid ettearvamatud ja nad ei liikunud maailmas ainult kindlas piirkonnas, vaid läbisid ka suuremaid vahemaid. Samas mõne vastaja jaoks ei olnud lendava vastase liikumine huvitav, nad arvasid, et liikumine oli ühekülgne ja selles ei olnud midagi erilist. Vastane ei teinud kordagi pausi, vaid liikus pidevalt. Soovitati kasutada erinevamaid liikumismustreid. Üksikud kasutajad tõid veana välja ka selle, et lendav vastane lendas vahel seinte sisse.

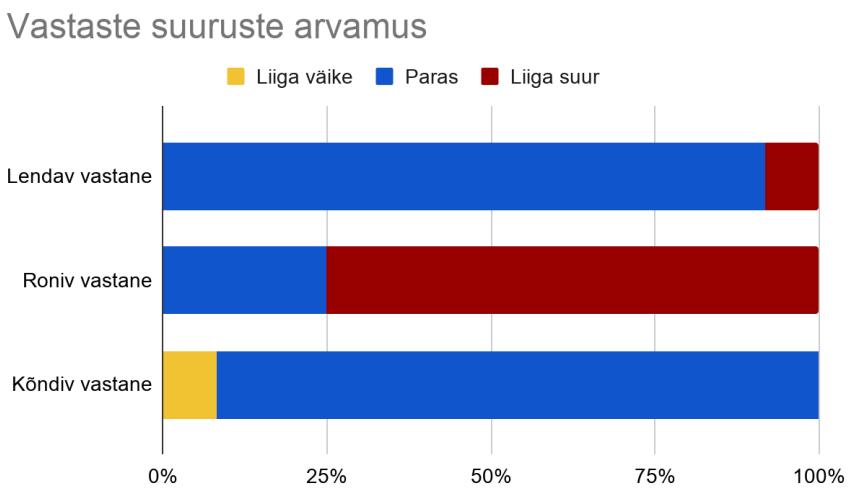


Joonis 20. Vastaste huvitavuse hinded.

Kasutajate arvates on roniva vastase liikumine huvitav ning vastase jalgadega liikumine ja taseme disain sobivad hästi kokku. Vastajad tõid välja eelmainitud vigast liikumist ja see mõjutas ka nende arvamust.

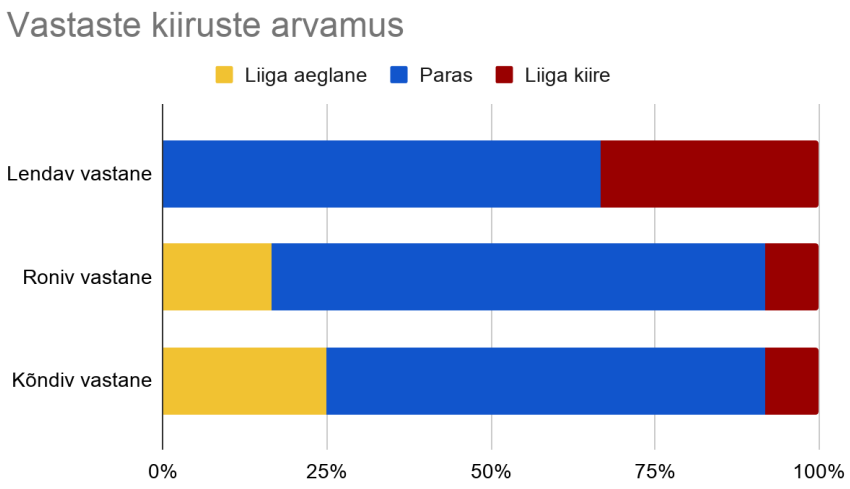
Kuna roniva ja kõndiva vastase liikumine on väga sarnane, siis hiljem vaadeldud kõndiv vastane ei tundunud nii huvitav oli roniv vastane. Põhjendusteks tõid vastajad välja samuti eelnevalt mainitud probleeme: jäsemed liikusid ebaloomulikult, teistega kokkupõrgetes liikusid ülespoole või liikumine sattus tsüklisse, millest välja ei saa ja nad ei saanud hakkama kitsaste kohtadega maailmas.

Järgmiseks küsiti kasutajatelt arvamust vastaste suuruste kohta. Tulemused on näidatud joonisel 21. Hindajate arvates oli lendava vastase suurus paras, kõigest üks inimene arvas, et lendava vastase suurus on liiga suur. Kõige suurema fluktatsiooniga hinnangud olid roniva vastase puhul, kus üheksa vastanut arvasid, et roniva vastase suurus on liiga suur. Enamus vastanutest arvas, et kõndiva vastase suurus on paras. Ainult ühe vastanu jaoks oli kõndiva vastase suurus liiga väike.



Joonis 21. Vastajate arvamus vastaste suuruste kohta.

Järgmiseks küsiti kasutajatelt arvamust vastaste kiiruste kohta. Tulemused on näidatud joonisel 22. Kaks kolmandikku vastajatest arvas, et lendava vastase kiirus oli paras ja üks kolmandik arvas, et kiirus on liiga kiire. Kaks vastajat arvasid, et roniv vastane liigub liiga aeglaselt ja üks vastaja arvas, et vastane liigub liiga kiiresti. Ülejäänud kasutajad arvasid, et vastane liigub paraja kiirusega. Kõndiva vastase ja roniva vastase hinnangud olid sarnased.



Joonis 22.Vastajate arvamus vastaste suuruste kohta.

Kasutajad soovitasid täiendamisel panna rõhku jalgadega vastastele. Teha liikumist sujuvamaks ja parandada kõik olemasolevad vead ning võimalusel lisada ka hüppamist.

Viimasena lasti vastajatel vastata küsimusele, mis tüüpi vastaseid nad sooviksid veel näha.

Esiteks soovitasid vastajad teha kaevur vastased, kes hüppaksid suvalisel hetkel läbi kivide. Teiseks soovitati teha väiksemaid, aga kiiremat tüüpi vastaseid. Kolmandaks pakuti, et mängus võiksid olla ussid, kes saaksid liikuda läbi seinte. Viimaseks soovisid paljud vastajad näha vastaseid, kes saaksid rünnata mängijat kaugemalt.

5. Kokkuvõte

Käesolevas töös tutvustati enamlevinud vastase liikumise tüüpe mängudes. Neist valiti kaks tehnoloogiat, tüürimiskäitumine ja pöördkineetika, mis sobivad kõige paremini Blastronaut mängule vastaste liikumise loomiseks. Töö eesmärgiks oli luua mängule kolm erinevat vastaste liikumist, mis oleksid võimalikult sujuvad ja sobiksid protseduuriliselt genereeritud mängu. Nendeks vastaste liikumise tüüpideks sai loodud lendav, roniv ja kõndiv liikumine. Lendav vastane kasutab tüürimiskäitumisel põhinevat algoritmi, millega ta hoidub seintest ja hoiab kokku teiste vastastega, moodustades gruppe. Roniv ja kõndiv vastane kasutab kõndimiseks jalgu ning see sarnaneb veidi pöördkineetika kasutamisele, kuna vastase keha liigutatakse fikseeritud jala asukohtade põhjal maailmas ringi.

Loodud vastaste liikumise sobivuse hindamiseks viidi läbi jõudluse testimine ja kasutajakogemuse testimine. Jõudluse testimise tulemus näitas, et kõik kolm liikumisviisi on piisavalt kiired, et mängu stseenis saaks olla vähemalt 100 vastast korraga. Kasutajakogemuse testimisest selgus, et lendava vastase liikumist väga loomulikuks, aga nii roniva kui ka kõndiva liikumist hinnati pigem robustseks, mis tähendab, et neid tuleb veel täiustada. Kõige huvitavamaks hinnati roniva vastase liikumist ja kõige igavamaks kõndiva vastase liikumist. Samuti jagasid vastajad häid soovitusi nende probleemide lahendamiseks. Kõige rohkem soovitati teha liikumist sujuvamaks ja lisada hüppamise võimekust. Lisaks pakuti erinevaid ideid uute vastaste loomiseks, näiteks vastased, kes saaksid kaevata läbi seinte või rünnata mängijat kaugemalt.

Praeguseks hetkeks on vastased juba Blastronaut mängu integreeritud. Järgnevatel kuudel on plaanis mängu edasi arendada ja võimaldada mängijal vastastega võidelda. Pärast seda on plaanis mäng avaldada veebipoes Steam, et mängijad saaksid seda mängida.

Kasutatud kirjandus

[1] N. Popovich. A Thousand Tiny Tales: Emergent Storytelling in ’Slime Rancher’. https://www.gdcvault.com/play/1024627/A-Thousand-Tiny-Tales-Emergent (07.05.2021)

[2] D. Mount ja R. Eastman. CMSC 425: Lecture 15 Motion Panning: Navigation Meshes. https://www.cs.umd.edu/class/spring2018/cmsc425/Lects/lect15-nav-mesh.pdf (07.05.2021)

[3] Navigation Meshes and Pathfinding https://www.gamedev.net/tutorials/programming/artificial-intelligence/navigation-meshes-and-pathfinding-r4880/ (07.05.2021)

[4] M. Perli. Delta Building Visualization – Agent Logic. Bachelor’s thesis, Tartu Ülikool, 2018.

[5] BorisTheBrave. Dungeon Generation in Diablo 1. https://sudonull.com/post/25412-Dungeon-Generation-in-Diablo-1 (07.05.2021)

[6] C. W. Reynolds. Steering Behaviors For Autonomous Characters https://www.red3d.com/cwr/steer/gdc99/ (07.05.2021)

[7] J. L. Andreas Aristidou. Forward And Backward Reaching Inverse Kinematics http://andreasaristidou.com/FABRIK.html (07.05.2021)

[8] Godot Docs https://docs.godotengine.org/en/stable/index.html. (07.05.2021)

[9] R. Nystrom, Game Programming Patterns, Genever Benning, 2014. (07.05.2021)

Lisad

I. Lähtekood ja muud failid

Lähtekoodi ja sellega seonduvaid faile leiab repositooriumist: https://gitlab.com/perfoon/blastronaut. Vastaste lähtekood asub kaustas Blastronaut\Objects\Enemy. Lähtekoodiga tutvumiseks võtke ühendust meiliga eriktare20@gmail.com.

Muud failid on tööga kaasas olevas arhiivis „Blastronaut_lisa.zip“. Kaasas on demo rakendused, mida leiab kaustast „Demo rakendused“.

II. Kasutaja tagasiside küsimustik

Tagasiside küsimustikku leiab manuse zip-failist nimega “küsimustik.pdf”.

III. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Erik Tarelkin,

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

   Vastaste liikumise arendamine Blastronaut mängule,

   mille juhendaja on Jaanus Jaago,

   reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 3.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.
3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.
4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Erik Tarelkin

07.05.2021