TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Harold Otsus 213425IACB

Arvuti poolt juhitud vastaste käitumine arvutimängudes ja ellujäämise arvutimängu arendus

Bakalaureusetöö

Juhendaja: Lembit Jürimägi
Lektor

Tallinn 2024

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Harold Otsus

11.05.2024

Annotatsioon

Käesoleva töö eesmärgiks oli uurida arvuti poolt juhitud vastaste käitumise kohta, arendada valmis ellujäämise arvutimäng, kus on võimalik valida erinevate vastaste käitumise algoritmide vahel ning seejärel viia läbi arvutimängu testimine, kus peamine fookus oli mängija kogemus.

Töös kirjeldatakse loodud arvutimängu arendusprotsessi ja kuidas olemasolevaid vastase käitumise algoritme mängus realiseeriti.

Testimise tulemusel selgus, et olemasoleva materjali põhjal realiseeritud oleku algoritm osutus kõige populaarsemaks ja pakkus kõige meeldivamat kogemust mängijatele, kuigi algoritmi arenduse keerukus ei olnud kõige kõrgem.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 59 leheküljel, 8 peatükki, 23 joonist, 8 tabelit.

Abstract

Computer Controlled Enemy Behavior in Computer Games and Survival Game Development

The aim of this thesis was to investigate the behavior algorithms of computer controlled enemies, develop a survival computer game, where it is possible to choose between different enemy behavior algorithms, and then carry out the testing of the game, where the main focus was on the player experience.

This thesis describes the development process of the created computer game and how existing enemy behavior algorithms were implemented in the game.

As a result of the testing, it was concluded that the state algorithm implemented based on existing material turned out to be the most popular and offered the most pleasant experience to the players, even though the development complexity of the algorithm was not the highest.

The thesis is in Estonian and contains 59 pages of text, 8 chapters, 23 figures, 8 tables.

Lühendite ja mõistete sõnastik

Spawn point	Ilmumiskoht, mis tähistab kohta mängumaailmas, kus mingi objekt või vastane tekib.
Grid	Unity mängumootori komponent, millega saab moodustada võrgustikke.
Box Collider 2D	Ristküliku kujuline Unity mängumootori füüsikakomponent, mis võimaldab määrata mänguobjekti mõjutavaid kokkupõrkeid.
Rigidbody 2D	Unity mängumootori füüsikakomponent, mis võimaldab mänguobjektil liikuda vastavalt füüsikaseadustele.
PlayerPrefs	Unity mängumootori klass, mis salvestab mängija eelistusi.
Prefab	Unity mängumootoris eelvalmistatud mänguobjekt.
Animator	Unity mängumootori komponent, millega saab mänguobjektidele määrata animatsioone.
Raycasting	Unity mängumootori tööriist, mida saab kasutada selleks, et arvuti poolt juhitud vastased saaksid mängumaailmast informatsiooni.
Layer	Kiht, millega saab erinevaid mänguobjekte eristada Unity mängumootoris.
Sprite	Unity mängumootori visuaalne tekstuur.
Trigger	Unity mängumootoris kasutusel olev käiviti, mis ei takista mänguobjektide omavahelist liikumist.
NavMesh	Unity mängumootoris kasutusel olev tööriist, mis võimaldab arvuti poolt juhitud vastasel mängumaailmas intelligentselt liikuda.

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Arvutimängude maailm on pidevalt arenev valdkond, kus mängijate ootused ja nõudmised tihti muutuvad. Seda arvesse võttes on tähtis uurida erinevaid aspekte arvutimängu arenduses, et saavutada lõppkasutajale kõige meeldivam kogemus. Üks tähtis aspekt, mis suuresti mõjutab mängija kogemust on arvuti poolt juhitud vastaste käitumine. Antud töö keskendub sellele teemale ning üritab selguse luua, mis algoritmid võivad pakkuda mängijatele rohkem meelelahutust.

Lõputöö esimene osa toob välja arvuti poolt juhitud vastaste käitumise tähtsuse ning kuidas see võib mõjutada kasutaja kogemust ja arvutimängu müüki. Lisaks, käsitletakse kahte populaarsemat vastase algoritmi ning tuuakse välja võrdlus nende vahel. Viimaseks, uurib autor, milline võiks välja näha arendusprotsess enda arvuti poolt juhitud vastase algoritmi loomiseks ning kuidas võivad valitud arvutimängu tüüp ja mängijate sihtgrupp seda protsessi mõjutada.

Teine osa käsitleb ellujäämise arvutimängu arendust. Peatükis kirjeldatakse arvutimängu arendusprotsessi ning näidatakse, kuidas on võimalik olemasolevaid populaarsemaid algoritme realiseerida enda loodud arvutimängus. Lisaks, näitab autor ära, kuidas proovis realiseerida adaptiivset algoritmi loodud arvutimängus.

Viimases osas kirjeldab autor ellujäämise arvutimängu testimist, mille ta viis läbi peale arvutimängu valmimist. Testimisel mängisid arvutimängu autori poolt valitud inimesed ning seejärel esitati neile küsimusi valitud küsimustiku põhjal, mille tulemusi peatükis analüüsitakse.

Töö pakub ülevaadet arvuti poolt juhitud vastaste käitumise algoritmide kohta arvutimängudes, pakkudes praktilisi näiteid ja analüüsides testimise tulemusi, mis loodetavasti aitavad tulevastel mänguarendajatel luua nauditavamaid mängukogemusi.

2 Arvuti poolt juhitud vastaste käitumise tähtsus

Arvutimängudes on väga palju erinevaid aspekte, mis teevad mängu kasutaja jaoks meeldivaks ja annab mängijale hea kogemuse. Üks tähtsatest aspektidest on vastaste käitumine mängija suhtes, mis mõjutab suuresti kasutaja kogemust ja võib ka positiivselt mõjutada arvutimängu müüki.

2.1 Kasutaja kogemus

Vastaste käitumine arvutimängudes mõjutab mängija üldist suhtumist mängu atmosfääri ja selle mängumaailma usutavust. Vastaste käitumine kujundab oluliselt ka mängu kulgu, arengut, raskusastet ja mängija kaasatust arvutimängu. Kui mängu jaoks on arendatud hea ja realistlik vastase käitumise algoritm, siis võib üldiselt eeldada, et mängija kogemus selles mängus on meeldivam ja lõbusam, kui mängus, kus ei ole suurt rõhku pandud vastaste käitumise arendamisele. Mitmekülgsed ja huvitavad vastased arvutimängus võib ka olla peamine müügiargument, millega uusi kliente ja mängijaid enda mängu mängima meelitada. [1]

Arvuti poolt juhitud vastased, kellel on mitmekesine käitumine vastavalt olukordadele, pakuvad teatud tüüpi mängijatele rohkem väljakutset, mis suurendab mängu põnevust ja emotsionaalset kaasatust. Arvutimäng nimega „F.E.A.R“., mis avaldati 2005. aastal, on saanud väga palju positiivset tagasisidet mängijatelt just sellel põhjusel, et antud mängu vastaste käitumine on väga realistlik ja organiseeritud. Eelmainitud mängus suhtlevad vastased omavahel ja teevad palju koostööd mängija vastu, mis annab mängijale mulje nagu ta võitleks väga tarkade vastastega. Sellised muljed suurendavad mängija immersiooni ja mängija kasutaja kogemus tõuseb märkimisväärselt. [2]

2.2 Arvutimängu müük

Kuigi edukamates ja populaarsemates mängudes enamasti ei ole peamine müügiargument vastaste käitumine, võib see siiski oluliselt mõjutada mängija kogemust ja anda lisaväärtust mängudünaamikale. Selline lisaväärtus ajendab mängijaid postitama

positiivseid arvustusi foorumitesse ja arvutimängude müügiplatvormidel, mis omakorda tõmbab ligi uusi mängijaid, kes võivad ka endale mängu soetada.

Seiklusmängus „Middle-earth: Shadow of Mordor“, mis avaldati aastal 2014, tutvustati uut *Nemesis System* [3] mehaanikat, mis võimaldab arvuti poolt juhitud vastastel mäletada mängija varasemaid tegevusi ja sellele vastavalt järgneval korral muuta oma strateegiaid mängija suhtes. Mängu mängijatel on olnud mitmeid personaalseid kogemusi tänu vastase käitumise süsteemile, kus vastased on näiteks kasutusele võtnud kilpe ja tugevaid peakaitsmeid, kuna mängija peamine strateegia on varasemalt olnud vibu või mõne muu relva kasutus, mis võimaldab kaugelt vastastele kahju teha. Seiklusmäng müüs esimese aastaga 2,45 miljonit ühikut [4], millega mängu tegijad saavutasid kasumi, mis ületas 18 miljonit dollarit [5]. Kuigi mängu edukust ei saa omistada üksnes *Nemesis System* mehaanikale, on mängijate tagasiside ja kogemused näidanud, et see oli kindlasti oluline aspekt.

Ellujäämisõudusmäng „Alien: Isolation“ avaldati samuti aastal 2014, kus oli ainult üks peamine arvuti poolt juhitud vastane, kes jahtis mängitavat karakterit terve mängu vältel. Mängu mängides jääb mängijatel tunne nagu neid jahiks ainult üks intelligentne vastane, aga tegelikult tegutseb mängus kaks erinevat vastase algoritmi. Üks algoritm on nähtamatu ja ei saa mängijale kahju teha, aga ta saab koguda informatsiooni mängija ja tema tegevuste kohta. Teine algoritm on tegelik vastane mängus, kes saab mängijat takistada ja talle kahju tekitada. Nähtamatu vastane varustab tegelikku vastast mängus informatsiooniga ja vastavalt sellele juhib tegelik vastane oma tegevusi [6]. Selline piltlikult nähtamatu koostoime teeb mängu mängijate jaoks väga unikaalseks ja soodustab mängu taasläbimängitavust. Vähem kui aastaga müüs õudusmäng 2,1 miljonit ühikut [7], millega teenisid mängu tegijad kasumit üle 21 miljoni dollari [8]. Mängijate kogemusi lugedes saab eeldada, et vastase käitumine mängus oli oluline tegur mängu müügis.

3 Populaarsemad vastaste käitumise algoritmid

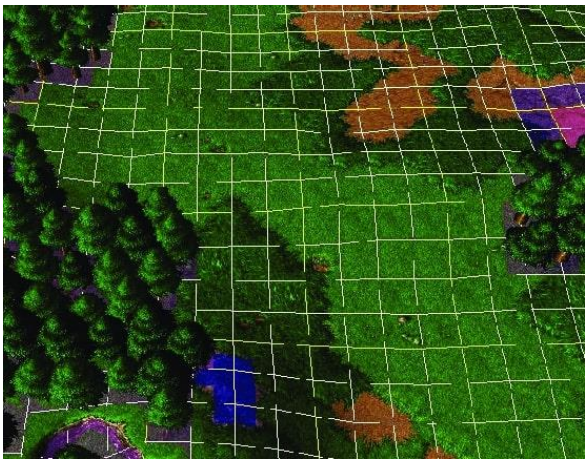
Antud peatükis on analüüsi ja võrdluse alla võetud kaks populaarset arvuti poolt juhitud vastase käitumise algoritmi, mida lõputöö autor realiseerib enda arendatud ellujäämise arvutimängus. Populaarseid vastase käitumise algoritme on palju, aga välja on valitud A* graafi läbimise ja teekonnaotsingu algoritm ning oleku põhine otsustusalgoritm.

3.1 A* graafi läbimise ja teekonnaotsingu algoritm

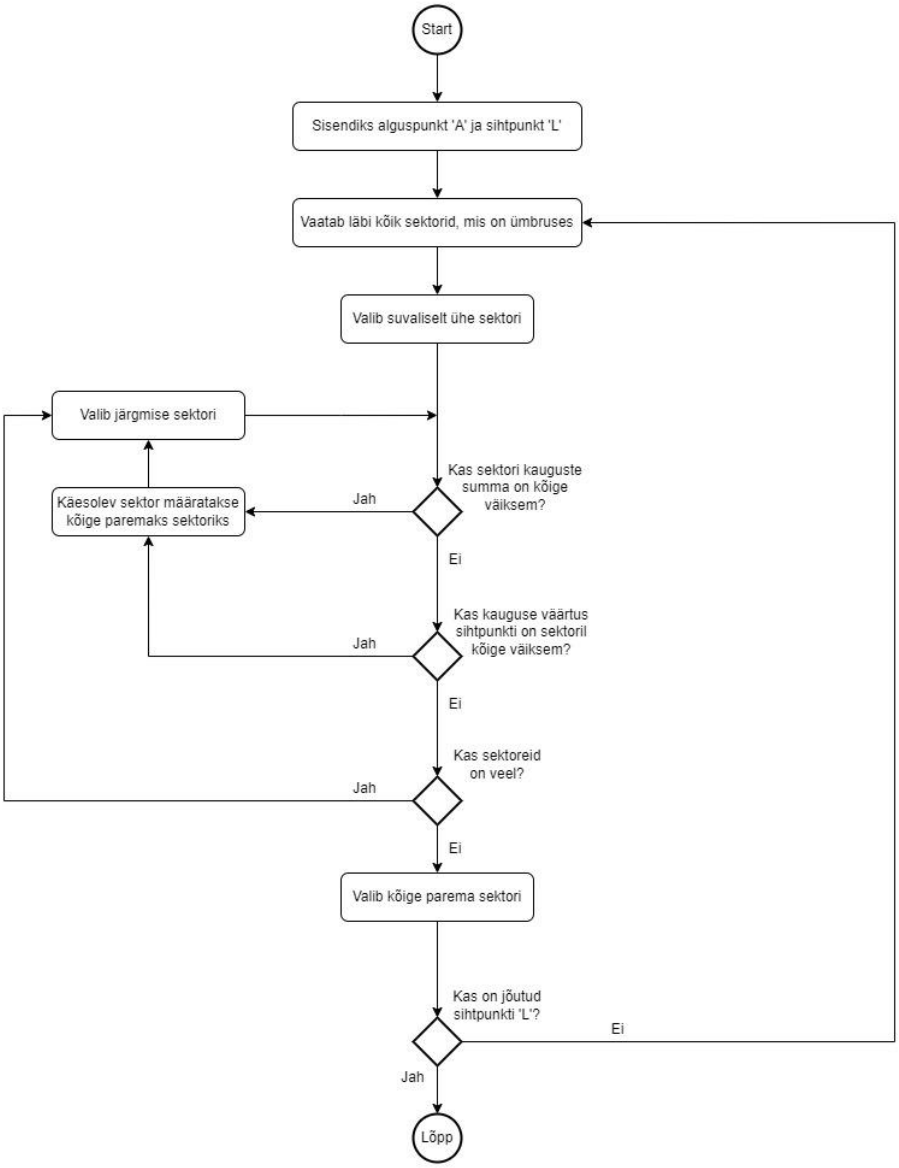
Antud algoritmi rakendatakse paljudes mängudes, et võimaldada arvuti poolt juhitud vastastel leida mängumaailmas mängitav karakter või mingi oluline sihtkoht kõige kiiremal viisil. Selline vastase käitumine on mängudes tavaliselt kõige lihtsam, kuna vastane suudab mängitava karakteri kiiresti leida ja seejärel püüab teda mingil moel rünnata. Üldjuhul intelligentse vastase vastu mängimine parandab mängija kogemust, kuid antud vastase lihtne käitumise algoritm sellist muljet ei jäta.

A* algoritm otsib välja lühima teekonna algsest olekust lõppolekusse. Algne olek üldiselt määratakse mängu tegija poolt ja selleks on tavaliselt nii-öelda vastase *spawn point*, ehk ilmumiskoht, mis määrab mängumaailmas ära, kuhu vastane tekkima peab. Lõppolek on tavaliselt mängitava karakteri objekt või siis mingi tähtsam sihtobjekt mängus, mida vastane näiteks kaitsma peab.

Algoritm enamasti töötab eeldusel, et mängumaailma *walkable*, ehk kõnnitav ala on jaotatud võrdseteks sektoriteks (Joonis 1), kus vastastel on antud luba liikuda. Kiireima teekonna leidmiseks kasutab algoritm kahte kaugust, milleks on kaugus valitud sektorist alguspunkti ning kaugus valitud sektorist sihtpunkti. Lisaks, liidab algoritm mõlemad kaugused kokku ja saab kauguste summa. Mida väiksem on kauguste summa sektoris, seda parem valik antud sektor on, aga kui kauguste summa on mitmel sektoril sama väärtusega, siis valib algoritm selle sektori, mille kaugus sihtpunkti on kõige väiksem. [9] (Joonis 2)



Joonis 1. Sektoriteks jaotatud maapind Warcraft III arvutimängus.



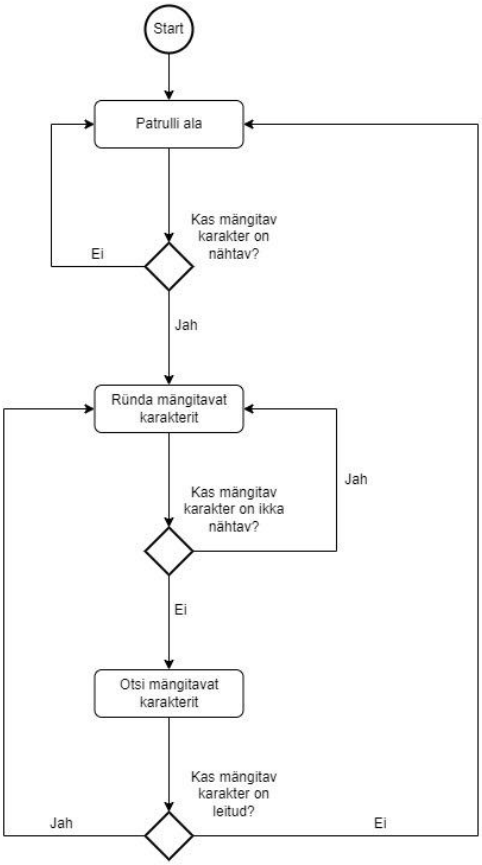
Joonis 2. Lihtne A* algoritmi diagramm.

Sageli kasutatakse A* algoritmi reaalaja strateegiamängudes nagu „StarCraft“, „Age of Empires“ ja „Warcraft III“, kuna sellistes mängudes on vajalik teekondade läbimine algpunktist sihtpunkti kiiresti, võttes arvesse erinevaid takistusi ja maastiku omadusi. [10]

3.2 Oleku põhine otsustusalgoritm

Antud algoritm on väga sarnane lõplikule juhtautomaadi mudelile, mis koosneb olekutest, üleminekutest ja käitumistest olekutes. Algoritm leiab kasutust mitmetes mängudes, kus on tahetud anda mängijale huvitavamaid kogemusi arvuti poolt juhitud vastaste võitlemise vastu. Kuna see algoritm võimaldab vastastel olla paljudes erinevates olekutes, tunduvad vastased mängijale intelligentsemad. See loob mulje, et vastased reageerivad unikaalselt igale toimuvale sündmusele. Oleku põhise algoritmi kasutamine on lihtne, kui olekuid on vähe. Kuid mida rohkem olekuid lisatakse, seda keerulisemaks muutub protsess. Mängu loojad ei pruugi suuta iga sündmuse jaoks unikaalset olekut luua. Kuigi, rohkemate olekute kasutamine võimaldab vastase käitumist unikaalsemaks ja huvitavamaks muuta, suurendades seeläbi mängija huvi mängu vastu. [11]

Oleku põhine algoritm suunab vastase käitumist vastavalt tema olekule või mängu olekule. Tavaliselt on vastastel algolekuks mingi mängus oleva ala patrullimine, koha peal seismine või mingi tähtsama objekti suunas liikumine. Kasutades sellist algoritmi, saab vastasele anda lisaülesandeid, mida ta peab mängu jooksul täitma. Näiteks, kui vastane ei ole mängijat mängumaailmas tuvastanud, siis tema peamine ülesanne on kedagi valvata või parandustööde tegemine vms. Kuid niipea, kui vastane on tuvastanud mängija, muutub tema peamiseks ülesandeks mängitava karakteri kinnipidamine või ründamine. Lisaks, saab anda erinevatele vastastele erinevaid olekuid, mis tõstab mängu mitmekesisust. Sellist algoritmi kasutades peab kõiki olekuid ja olekute üleminekuid testima, et vastased käituksid soovitud viisil. [12] (Joonis 3)



Joonis 3. Lihtne oleku põhise otsustusalgoritmi diagramm.

Oleku põhiseid algoritme kasutatakse palju rollimängudes, kus mängijal on väga palju erinevaid võimalusi mängus ja mängija saab ülesandeid täita mitmel erineval viisil. Kuna mängijal on suur tegevuste valik, siis vastased peavad igale võimalikule tegevusele vastama realistlikult ja selleks on vastastel väga palju võimalike olekuid ja üleminekuid. Autor enda kogemuse põhjal väidab, et rollimängud, mis rakendavad oleku põhist algoritmi on „Deus Ex“ ja „The Elder Scrolls V: Skyrim“.

3.3 Algoritmide võrdlus

Mõlemal algoritmil on oma eelised ja puudused, mis mõjutavad mängu arendusprotsessi ja lõppkasutaja kogemust arvutimängus. (Tabel 1)

A* teekonnaotsingu algoritmi puhul on võimalik luua lihtne vastase algoritm kiiresti, mis leiab sihtobjekti või mängitava karakteri asukoha ja asub seda ründama. Mängu loojal on lihtsam rakendada A* algoritmi, kuna siis on võimalik keskenduda rohkem teistele aspektidele mängus. Lisaks, ei pea mängu looja väga palju muretsema erinevate

võimalike olekute peale, kuna sisuliselt on vastasel ainult üks olek, milleks on mängitava karakteri või mingi muu sihtobjekti leidmine ning selle hävitamine/kaitsmine vms. Kuigi algoritm soodustab arendusprotsessi kiirust, ei soodusta see tingimata kasutaja kogemust mängus. Kuna vastasel on väga limiteeritud käitumine, siis võib mängu kulg kujuneda üksluiseks ja igavaks, aga seda ei saa öelda iga mängija kohta, kuna teatud tüüpi mängijad leiavad meelelahutust ka ühedimensioonilistes vastastes.

Oleku põhist algoritmi kasutades saab realiseerida väga mitmekülgset ja huvitava käitumisega vastast, mis teatud mängijate jaoks teeb mängu väga huvitavaks ja taasläbimängitavaks. Mängu arendaja jaoks, teeb see aga vastase algoritmi loomise palju keerukamaks ja ajakulukamaks. Mängu loojad peavad algoritmi loomisel arvestama kõikide võimalike tegevustega ja olekutega. Lisaks peavad nad läbi proovima ja testima võimalikud mängija tegevused vastase vastu ja vastasel peavad olema olekud kõikide nende tegevuste vastu, et muuta mäng väga huvitavaks mängija jaoks ja vältida üksluisuse tekkimist.

Tabel 1. A* teekonnaotsingu ja oleku põhise algoritmide võrdlus.

Algoritm	**A* teekonnaotsingu algoritm**	**Oleku põhine otsustusalgoritm**
Algoritmi realiseerimise kiirus	Kiire	Aeganõudev
Algoritmi keerukus	Lihtne	Keeruline
Testimisele kuluv aeg	Lühike	Pikk
Kasutaja kogemus	Üksluine	Mitmekesine

4 Protsess vastase käitumise algoritmi loomiseks/valimiseks

Õige arendatava arvutimängu vastase käitumise algoritmi loomine või valimine võib olla keerukas protsess ning sõltub väga palju subjektiivsetest asjaoludest, aga antud peatükis proovib autor seda protsessi lihtsustada. Peatükis tuuakse välja erinevad arvutimängude tüübid ja enamlevinud vastase käitumised nendes mängudes, ning ka mängijate sihtgruppide valimise, mis hõlbustab vastase käitumise algoritmi loomist või valikut.

4.1 Arvutimängu tüüp

Enne vastase käitumise algoritmi loomist tuleb arvestada, millise kategooria või stiili alla arendatav mäng langeb. Kui mäng langeb näiteks strateegiamängude kategooria alla, siis tuleb arvestada sellise algoritmiga, mis oskaks ressursse hallata, reageerida vastavalt mänguseisule ning juhtida enda loodud üksusi mängija baasi või üksuste vastu. Näiteks, strateegiamängus „Warcraft III“ koosneb vastase algoritm mitmetest erinevates skriptidest ja käivitusmehhanismidest, millega saab vastane reageerida mänguseisule ja mängija tegevustele. Käivitusmehhanismist näide oleks, kui mängija otsustab rünnata vastase baasi, siis lõpetab vastane kõik oma muud tegevused ja asub oma baasi kaitsma mängija eest nii kaua, kuni rünnak kestab [13] [14]. Strateegiamängu loomiseks tuleks suunata rohkelt arendusressurssi vastase käitumise algoritmile, kuna skriptide ja käivitusmehhanismide väljamõtlemiseks ning rakendamiseks kulub ohtralt aega. Lisaks, tuleb algoritmi ka palju testida, et vastane käituks soovitud viisil.

Arvutimängud, kus vastaste käitumise algoritmi peale nii palju ressursse ei pea kulutama on näiteks tulistamismängud ja platvormiseiklusmängud. Sellistes mängudes on vastased enamasti lihtsamad ja teevad mingit ühte tegevust nii kaua, kuni mängija neid tuvastab või vastane tuvastab mängijat. Lihtsamateks tegevusteks on patrullimine või lihtsalt koha peal seismine mängumaailmas. Arvutimängus „Half-Life“, mis on esimese isiku tulistamismäng, tekitatakse suurem osa vastaseid mängumaailma ära ja vastased üldjuhul seisavad koha peal ja ootavad, kuni mängija satub nende lähedusse [15]. Kui mängija lõpuks saabub vastase juurde, siis vastane hakkab mängitavat karakterit taga ajama ja

üritab teda rünnata. Sellise mängu loomise puhul tuleb arvestada suurema arvu vastastega, kui selleks soovi on, aga väga palju arendamist selline vastase käitumine ei nõua.

4.2 Mängijate sihtgrupi valimine

Lisaks arvutimängu tüübile, peab ka arvestama, kes kuuluvad mängu peamisse sihtgruppi, ehk kes on need arvutimängude mängijad, kellele on soov mäng luua. Tänapäevaks on välja kujunenud väga palju erinevaid arvutimängude mängijate gruppe, keda saab kategoriseerida nende mängimisstiili ja mängude eelistuste järgi [16]. Antud alapeatükis on ära kirjeldatud autori arvamusel kolm peamist mängija sihtgruppi.

Esimene tüüp mängijaid, kellega saaks arvestada oleks nii-öelda pühendunud mängurid, kes on väga palju aega veetnud arvutimänge mängides ning selle tõttu on neil tekkinud suur vilumustase mängude läbimisel. Selliste mängijate sihtgrupiks valimine tähendaks keerukamate vastaste käitumise algoritmide arendamist, kuna pühendunud mänguritele meeldib väljakutse ning nad on huvitatud kompleksetest ja aeganõudvatest mängudest. [17]

Teine tüüp mängijaid on nii-öelda harrastajad mängurid, kellele meeldivad arvutimängud, aga nad enamjaolt ei pühenda väga palju aega mängudes paremaks saamisele. Harrastajad mängurid eelistavad arvutimänge mängida lõbu pärast ja enamasti ei ole nõus väga palju aega veetma keerulisemaid mänge mängides. Antud mängijate sihtgrupiks valimine eeldaks seda, et vastase käitumise algoritmid ei oleks liiga keerukad ja rasked, ning annaksid mängijale kergema väljakutse mängu läbimisel. [17]

Kolmas tüüp mängijad on mängurid, kes eelistavad arvutimängu lugu, kunsti või muusikat. Sellistele mängijatele suunatud arvutimängu arendamisel tuleb arvestada, et väga palju arendust vastase käitumise algoritmi jaoks ei ole vaja, kuna seda tüüpi mängijad naudivad mängu loo kulgu ja visuaalseid aspekte rohkem. [18]

Arvestades eeltoodud mängurite tüüpe arvab autor, et kõige tulusam oleks valida sihtgrupiks harrastajad mängurid. Sihtgrupi valimisel ei ole väga keerukaid algoritme vaja välja mõelda, vaid saab keskenduda muudele aspektidele. Arvestades, et harrastajate mängurite sihtgrupp on üldiselt suurem kui pühendunud mängurite sihtgrupp, jõuaks loodud arvutimäng potentsiaalselt rohkemate inimesteni. [19]

5 Ellujäämise arvutimängu arendus

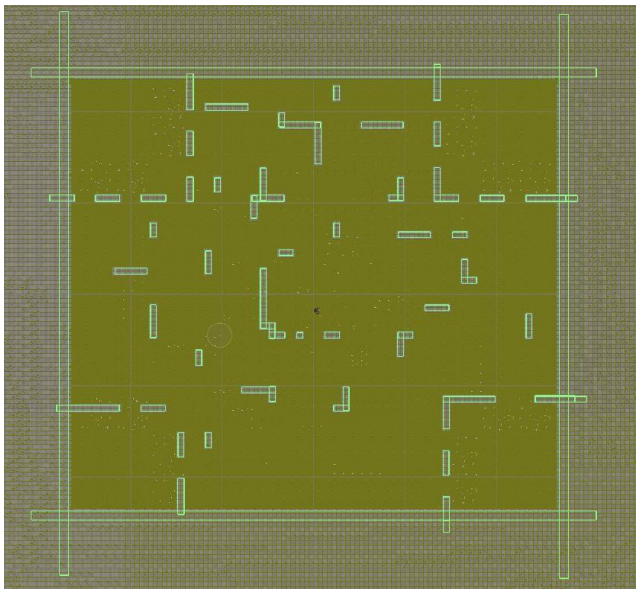
Selles peatükis käsitletakse ellujäämise arvutimängu arendusprotsessi. Peatükis tuuakse välja mängu arendusvahendid ja nõuded. Lisaks, kirjeldatakse ära ka mängumaailm, vastased ja arvuti poolt juhitud vastaste algoritmid. Loodud arvutimäng on mõeldud ennekõige Windows'i operatsioonisüsteemiga personaalarvutitele, aga mäng võib töötada ka teiste operatsioonisüsteemide peal.

5.1 Arvutimängu nõuded

Autori poolt loodud ellujäämise arvutimäng on mäng, mille käigus peab mängija mängitavat karakterit võimalikult kaua elus või töövõimelisena hoidma. Mängija jaoks raskendatakse seda ülesannet arvuti poolt juhitud vastaste ning mängumaailma piiratuse näol. Arendatud arvutimängus otsivad vastased mängumaailmast mängitava karakteri üles ja üritavad mängija ellujäämist takistada.

Arvestades ellujäämise arvutimängu olemust, peab mängus kindlasti olema mängitav karakter, keda mängija saab juhtida ning arvuti poolt juhitud vastased, kellel peab olema vähemalt selline algoritm, et vastane leiaks mängitava karakteri asukoha mängumaailmas. Ühtlasi, peab vastastel olema võimalus mängitavat karakterit takistada ning mängijal endal peab olema võimalus ennast kaitsta või kuidagi vastata arvuti poolt juhitud vastaste tegevustele. Selleks võimaluseks üldiselt taolistes arvutimängudes on tulirelvad ning lähivõitlusrelvad.

Kuna mängitav karakter ja vastased peavad kuskil ringi liikuma, et saavutada oma ülesandeid, siis selleks peab olema loodud ka vastav mängumaailm, mis oleks teatud piirangutega. Mängumaailm peab andma piisavalt vabadust, et mängija oleks suuteline põgeneda vastaste eest, aga peab olema piiratud seintega, et vältida olukorda, kus mängijal oleks võimalik põgeneda lõpmatult. Täiendavalt, peab mängumaailma seintega piiratud ala olema ka sisemiselt struktureeritud, luues labürindi sarnase keskkonna. See on vajalik, et mängijal ja vastastel oleks võimalik kasutada katet üksteise vastu ja teeb mänguelamuse huvitavamaks (Joonis 4).



Joonis 4. Mängumaailm nähtavate kokkupõrkekehadega.

Mängus peab ka olema võimalik valida kolme erineva vastase algoritmi vahel ning selleks peab looma menüü, kus on võimalik valida algoritmi ja siis valitud algoritmiga mängu alustada.

5.2 Mängu taust ja ressurssid

Loodud arvutimängu mõte on mängijal võimalikult kaua võitlusvõimelisena püsida tulistamismängus. Mängija juhib mängitavat karakterit, kellel on kokku kolm peamist ründamis- ja kaitsmisvahendit. Mängitavat karakterit ründavad vastasmeeskonnas olevad arvuti poolt juhitud vastased, kellel on samad vahendid, mis on mängijal.

Mängus kasutatud graafikavara on võetud www.itch.io veebilehelt, kus artistid on jaganud enda tehtud mänguressursse tasuta. Mängumaastiku ja selle *tilemap*'i, ehk plaatkaardi, loomiseks on kasutatud Cainos'e „Pixel Art Top Down Basic" tekstuure. Mängitava karakteri ja vastaste välimused ja ressurssid on võetud ChrisJulch'i „Top Down Shooter Character" varapakist [20] [21]. Mängitava karakteri ja vastaste võimetuks tegemise, relvade ja kilbi graafikavara ning tekstuurid on loodud Liina Taluste poolt [22].

5.3 Mängumootor ja versioonihaldus

Arvutimängu arendusplatvormiks sai valituks Unity Real-Time Development Platform, mis kasutab Unity Engine mängumootorit. Lisaks, kasutab Unity arendusplatvorm koodi kirjutamiseks Microsoft'i Visual Studio integreeritud arenduskeskkonda. Unity's saab koodi kirjutada C# programmeerimiskeeles. Autor valis Unity, kuna on olnud varasem kokkupuude arendusplatvormiga ning ka põhjusel, et C# programmeerimiskeel on autorile mugav kasutada. [23]

Versioonihalduseks sai valituks GitHub'i arendajate platvorm, kuhu autor pidevalt laadis arvutimängu arendamise projekti vältel üles mänguressursse, programmikoodi ning muid teisi projektiks vajaminevaid faile. [24] (Lisa 2 – Ellujäämise arvutimängu repositoorium)

5.4 Vastased

Arvutimängu jaoks loodi 3 erinevat vastast (Joonis 5), kelle ülesandeks on mängitava karakteri ründamine ja tema takistamine mängumaailmas. Esimene loodud vastane saab kaugelt rünnata mängitavat karakterit püstoliga, mis tulistab kuule. Teine vastane saab lähemalt rünnata mängitavat karakterit, kasutades selleks kurikat. Kolmas vastane on kilbiga ning saab kaitsta ennast ja teisi vastaseid mängitava karakteri rünnakute eest. Kõikidest kolmest vastase tüüpist on moodustatud *prefab*'id [33] ehk eelvalmistatud mänguobjektid, et nende ilmumine mängumaailma oleks lihtsam. Lisaks, on kõikidel vastaste tüüpidel mänguobjektide *Box Collider 2D* ja *Rigidbody 2D* komponendid. *Rigidbody 2D* komponent võimaldab mänguobjektil liikuda ja käituda mängumaailmas vastavalt füüsikaseadustele. Mõlemad komponendid on vajalikud, et tuvastada kokkupõrkeid näiteks vastase mänguobjekti ja mängija poolt tulistatavate kuulide vahel.

Joonis 5. Püstoliga vastane (vasakul), kurikaga vastane (keskel), kilbiga vastane (paremal).

5.4.1 Vastase tüübid

Püstoliga vastase mänguobjekt koosneb püstoli alamobjektist ning sellele lisaks tulistamispunkti alamobjektist. Tulistamispunkti objekti on vaja selleks, et tulistatav kuul ei alustaks lendu vastase mänguobjekti keha keskelt, vaid tulistamispunktist, mis asub vastasest natukene eespool. (Lisa 4, Joonis 19)

Kurikaga vastane koosneb kurika alamobjektist, millele on lisatud *Box Collider 2D* komponent, et oleks võimalik tuvastada olukorda, kui kurikaga vastane lööb mängitavat karakterit. Lisaks, erinevalt teistest vastaste tüüpidest, on kurikaga vastasel ka animatsiooni komponent *Animator* [34], millega saab näidata kurikaga löömise animatsiooni. (Lisa 4, Joonis 20)

Kilbiga vastane koosneb kilbi alamobjektist, mis annab visuaalselt märku, et vastasel on käes kilp. Kilbile ei ole lisatud *Box Collider 2D* komponenti, mis tuvastaks kokkupõrkeid, vaid pigem on kilp visuaalse tähtsusega, mis eristab vastast teistest vastaste tüüpidest. (Lisa 4, Joonis 21)

Vastase ilmumisel mängumaailma, määratakse talle elu punktid, mis sõltuvad vastase tüübist. Elu punktide otsa saamisel on vastane kõrvaldatud ning ei saa enam mängitavale karakterile kahju tekitada. Püstoliga vastasele määratakse 20 elu punkti, kuna vastane saab rünnata kaugemalt, kui teised vastased; kurikaga vastasele määratakse 40 elu punkti, kuna tema peab mängijale tunduvalt lähedamale minema ning kilbiga vastasele määratakse 60 elu punkti, kuna temal on kilp, millega saab ennast kaitsta.

Peale elu punktide, määratakse vastasele ka liikumise kiirus vastavalt vastase tüübile. Kõige kiirem on kurikaga vastane ja kõige aeglasem on püstoliga vastane, mis on tingitud sellest, et kurikaga vastane ei oleks liiga aeglane, kuna muidu oleks teda liiga lihtne kõrvaldada ning samuti, et püstoliga vastane ei oleks liiga kiire, kuna sel juhul oleks teda keerulisem kõrvaldada.

5.4.2 Takistuse tuvastus, mängija ründamine ja liikumine

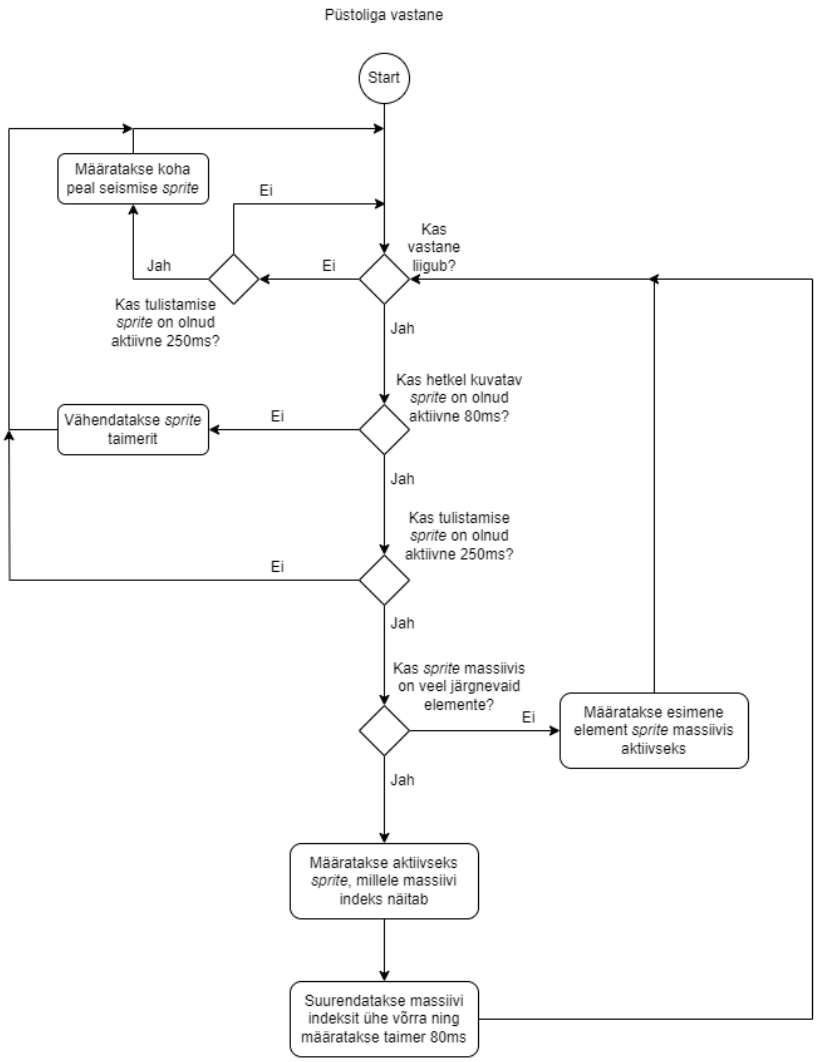
Kuna mängumaailmas on väga palju seinu, mis on takistuseks vastastele, siis nad peavad nägema enda ette, et mitte kõndida vastu seina ja mitte tulistada seina, kui nad tuvastavad, et mängitav karakter on teisel pool seina vms. Selleks on kasutusele võetud *Raycasting*,

mis on nähtamatute kiirte kuvamine soovitud suunas mängumaailmas ja sellest informatsiooni saamine [35]. Kui vastane tahab rünnata mängijat, siis selleks peab ta enne kuvama *Raycast* kiire selles suunas, kuhu vastane hetkel vaatab. Mängitavale karakterile ja mängumaailmas asuvatel seintele on määratud erinevad *layer*'id, ehk kihid, mida vastane saab näha *Raycasting*'uga. Kui vastane saab tagasi informatsiooni, et *Raycasting* tabas kihti, mis vastab mängitavale karakterile, siis saab kindel olla, et vastasel ei ole takistusi ees selles suunas, kuhu ta hetkel vaatab. (Lisa 4, Joonis 22)

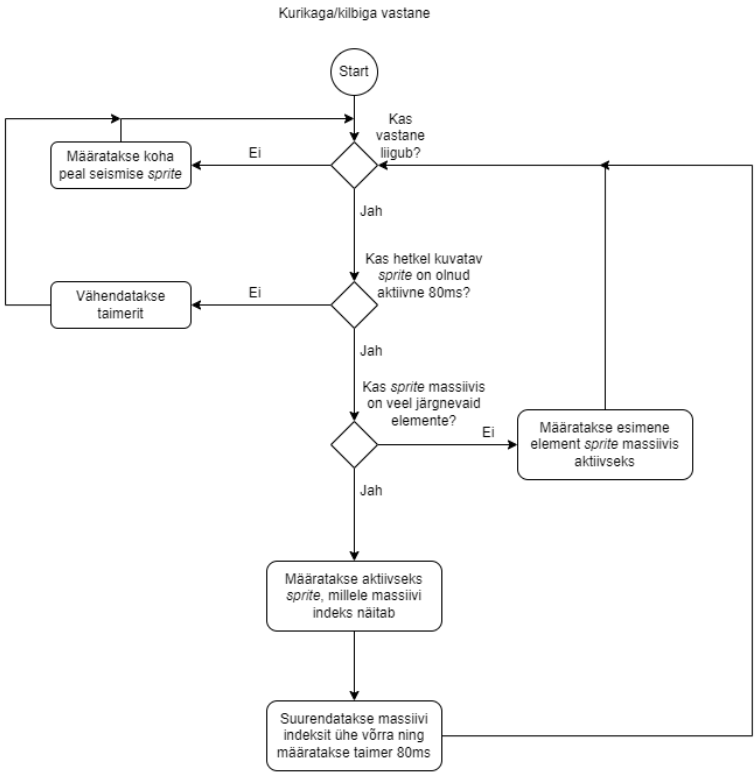
Enne mängija ründamist kontrollitakse ka mängitava karakteri kaugust vastasest. Kui kaugus on väiksem või võrdne ette antud väärtusest, mis on erinev igal vastase tüübil, siis alustab vastane mängija ründamisega. Püstoliga vastasel on ette antud väärtus kõige suurem, kuna temal peab olema võimalik kõige kaugemalt rünnata mängijat. Kurikaga ja kilbiga vastastel on see väärtus väiksem, kuna nende ülesanne on just mängijale lähedale saada.

Vastase liikumise kasutajale selgemaks tegemiseks on vastaste mänguobjekti *sprite*'id, ehk visuaalsed tekstuurid, pidevalt muutuses, kui vastane liigub. See annab mängijale hea arusaama, et vastane liigub ja näeb loomulikum välja. Selle teostamiseks on igal vastasel *sprite*'idest koosnev massiiv, mis kujutavad vastase liikumist. (Lisa 4, Joonis 23)

Igat *sprite*'i kuvatakse 80ms ning peale seda vahetatakse *sprite* järgnevaga massiivis. Kui on jõutud massiivi viimase *sprite*'ini, siis algväärtustatakse massiivi indeks tagasi esimese elemendini. Püstoliga vastasel on veel lisaks eraldi tulistamise *sprite*, mida kuvatakse, kui püstoliga vastane tulistab mängija suunas. Tulistamise *sprite*'i kuvatakse 250ms ning lisaks arvestatakse, et kui tulistamise *sprite* ei ole aktiivne olnud talle määratud aja, siis enne selle lõppemist liikumise *sprite*'si kuvama ei hakata. (Joonis 6, Joonis 7) [36]



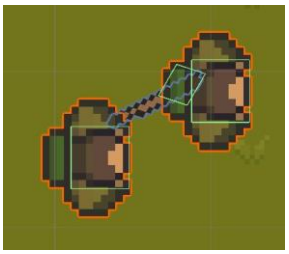
Joonis 6. Püstoliga vastase liikumise *sprite*'ide vahetamise algoritm.



Joonis 7. Kurikaga/kilbiga vastase liikumise *sprite*'ide vahetamise algoritm.

5.4.3 Vastase mänguobjekti hävitamine ning vastaste ilmumine mängumaailma

Vastase elu punkte saab vähendada kahel viisil mängija poolt: püstoliga kuulide tulistamisega ja kurikaga ründamisega. Tänu *Rigidbody 2D* komponendile, saab tuvastada kokkupõrkeid vastase mänguobjekti ja muude võimalike mänguobjektide vahel. Üheks mänguobjektiks võib olla kuul, mis kokkupõrke tagajärjel vastasega vähendab antud vastase elu punkte 10-e võrra. Lisaks, vähendab vastase elu punkte ka mängitava karakteri kurikas, mille tipus on *Box Collider 2D* komponent, millega saab kontrollida, kas vastase mänguobjektiga puutus kokku teatud tüüpi *trigger*, ehk käiviti, mis vähendab vastase elu punkte 20-e võrra. Kokkupõrke ja *trigger*'i vahe seisneb selles, et kokkupõrke puhul hävitatakse lendav kuul, aga *trigger*'i puhul jääb kurikas alles ning registreeritakse kokkupuude mänguobjektiga. (Joonis 8)



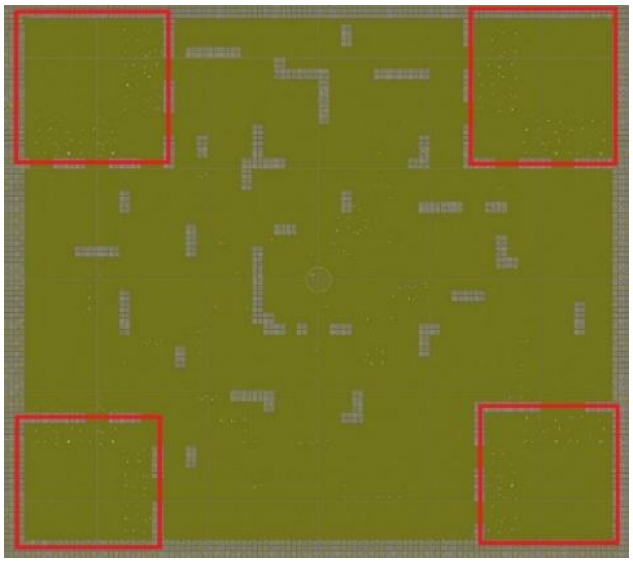
Joonis 8. Mängija (vasakul) kurikas puutub kokku vastase (paremal) mänguobjekti *Box Collider 2D* komponendiga.

Kui vastasel saavad elu punktid otsa, siis kuvatakse *sprite*, mis iseloomustab vastase kõrvaldamist mängust ja näitab mängijale, et antud vastane enam võitlusvõimeline ei ole (Joonis 9). Lisaks, hävitatakse vastase *Rigidbody 2D* ja *Box Collider 2D* komponendid, et vastase mänguobjekt ei reageeriks muutustele, mis võivad olla põhjustatud mängija või teiste vastaste liikumiste pärast. Vastase mänguobjekti lõplik hävitamine toimub 60 sekundi pärast peale kõrvaldatud *sprite*'i kuvamist, kus terve vastase mänguobjekt kustutatakse mängumaailmast, et hävitatuid vastaseid liialt palju ei koguneks.

Joonis 9. Mängust kõrvaldatud vastane.

Erinevalt teistest vastastest, kui kilbiga vastasel saavad elu punktid otsa, siis tekib tema asukohale püstoliga vastane ning kilbiga vastase kõik komponendid ja *sprite*'id kustutatakse koheselt. See on tehtud nii sellepärast, et mängijale jääks mulje, et ta hävitas ära vastase kilbi ja nüüd kui vastasel kilpi enam ei ole, siis hakkab vastane mängijat ründama püstoliga.

Vastaste ilmumise eest mängumaailma vastutab mänguobjekt, millel on nimekiri kõikidest erinevatest vastaste mänguobjektide *prefab*'idest. Lisaks, on mänguobjektil massiiv, kus asuvad vastaste ilmumis asukohad mängumaailmas, et ilmumise mänguobjekt teaks, kuhu ta saab vastaseid tekitada. (Joonis 10)

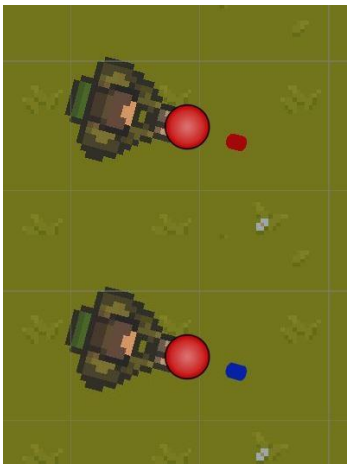


Joonis 10. Alad, kus saavad vastased ilmuda (punaste ristkülikutega tähistatud alad).

5.5 Tulistamine ja lähivõitlus

Püstoliga vastasele on määratud tulistamise kiiruse taimer, mis määrab ära kui tihti saab vastane tulistada. Kui tulistamise kiiruse taimer on null või väiksem, siis saab vastane tulistada mängitava karakteri suunas. Tulistades tekib vastase ees olevast tulistamispunktist kuuli *prefab* mänguobjekt, mis lendab vastase vaatamise suunas. (Joonis 11) [37]

Kui mängija soovib tulistada, siis esmalt kontrollitakse, kas mängija on üldse elus, mis on võimalik ainult siis, kui mängitava karakteri elu punktide kogus on suurem nullist. Kuuli tulistamiseks peab mängija vajutama vasakut hiire klõpsu ning seejärel kontrollitakse, kas tulistamise kiiruse taimer on null või väiksem, et mängija ei saaks liiga kiiresti kuule tulistada. Sarnaselt püstoliga vastasele, tekitatakse kuul mängija tulistamispunktist, aga ainuke erinevus on, et tekitatakse teise sildiga *prefab* kuulist, mis vastab ainult mängija poolt tulistatud kuulile. (Joonis 11) [37]



Joonis 11. Vastase kuul (punane) ja mängija kuul (sinine), ning suured punased tulistamispunktid.

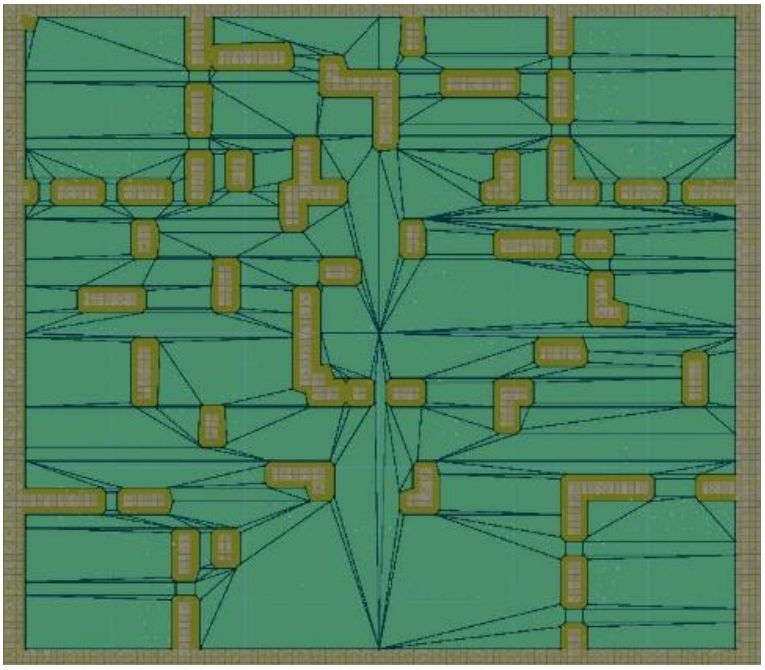
Kuulile on määratud kuuli kiirus ja lendamise aeg, mis määrab kui kaua püsib kuuli mänguobjekt mängumaailmas, enne kui ta hävitatakse. See on tehtud põhjusel, et vältida lõputult või kaamera vaateväljast välja lendavaid kuule. Kui kuul tekib, siis hakkab taimer käima, mis nulli jõudes hävitab kuuli mänguobjekti. Kui kuul põrkab kokku ükskõik mis mänguobjektiga, mis ei ole sildiga *Enemy*, ehk vastane, siis hävitatakse kuuli mänguobjekt. Kuna on olemas kaks erinevat kuuli *prefab*'i, siis kui vastase kuul põrkab kokku *Enemy* sildiga mänugobjektiga, siis kuuli ei hävitata, aga kui kuuli enda silt on *Bullet*, siis see saab olla ainult mängija poolt lastud kuul, mis kindlalt hävib kui puutub kokku ükskõik mis mänguobjektiga. Erinevad kuuli *prefab*'id loodi, et püstoliga vastased saaksid lasta läbi kilbiga vastaste, kuna adaptiivses algoritmis kaitsevad kilbiga vastased püstoliga vastaseid ja sel juhul ei teki olukorda, kus püstoliga vastane ei saa tulistada mängijale pihta, kuna tema ees seisab kilbiga vastane. [37]

Lähivõitluses on kurikatega vastastel samuti taimer, mis kontrollib, kas vastane tohib kurikaga lüüa. Selle mõte on jätkuvalt see, et vastane ei saaks liiga kiiresti rünnata mängijat ning tema elu punkte selle tagajärjel väga kiiresti vähendada. Kui taimer on null või väiksem, siis käivitatakse ründamise animatsioon antud vastase mänguobjekti peal, kes ründamist sooritab. Mängja saab sisuliselt samal viisil rünnata vastaseid, aga mängija peab selleks elus olema ja vajutama vasakut hiire klõpsu. Samuti kontrollitakse ka mängija puhul taimerit, et tema ei saaks ka liiga kiiresti vastaseid rünnata, kuigi mängijale on määratud kiirem löömise sagedus kui vastastel, et mängijat julgustada kurikat kasutama. Mängija puhul samuti käivitatakse ründamise animatsioon, mis näitab mängijale, et ta just lõi kurikaga. [38]

5.6 Lihtne algoritm

Lihtne algoritm põhineb eelpool peatükis mainitud A* teekonnaotsingu algoritmi põhimõtetest ning algoritm kasutab ainult teekonna otsingut mängija üles leidmisel. Muu keerukus algoritmil puudub ja selle tõttu on ka algoritmil selline nimetus. Antud algoritmi arendamiseks kulus ka võrreldavalt vähem aega, kui teiste keerulisemate algoritmide puhul.

Teekonnaotsingu algoritmi on Unity mängumootoris võimalik realiseerida läbi *NavMesh* tööriista, mis võimaldab arvuti poolt juhitud vastastel mängumaailmas intelligentselt liikuda. Tööriist võimaldab vastastel leida parima tee punktide vahel mängumaailmas ning võtab arvesse võimalike takistusi, mida vastane ei tohiks saada läbida. Selleks tuleb mängumaailmas ära määrata, mis on nii-öelda kõnnitav ala, kus vastased tohivad liikuda ning samuti peab ära määrama ka takistused, kus vastane ei tohi kõndida, milleks antud mängus on seinad. (Joonis 12) [41]

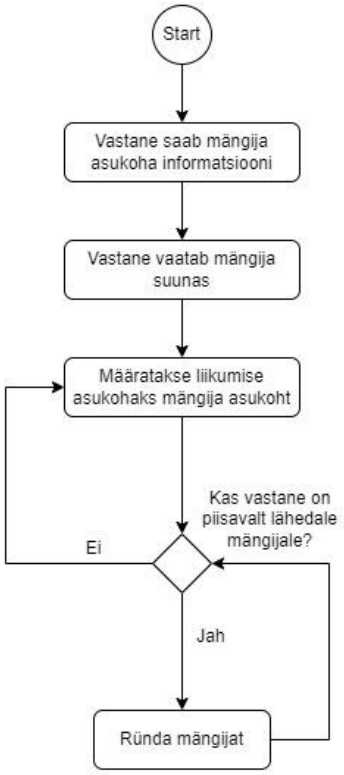


Joonis 12. Mängumaailma loodud *NavMesh* pind, kus helerohelise värviga on märgitud kõnnitav ala.

Antud algoritmi puhul tekitatakse vastaseid mängumaailma staatiliste lainetena. Iga teatud aja tagant hakkab ilmuma mängumaailma teatud kogus vastaseid. Esialgse laine vastaste arv on 10 ning iga järgnev laine lisatakse juurde 1 püstoliga ja 1 kurikaga vastane. Iga teine laine lisatakse juurde ka 1 kilbiga vastane. Vähem kilbiga vastaseid on oluline alguses selle jaoks, et ei oleks liiga palju nii-öelda tugevaid vastaseid, kelle elu punktid

kõige kõrgemad on. Peale esimest lainet, tekib iga järgnev laine peale 40 sekundit eelnevast ning lained tekivad lõpmatult.

Kui vastased tekivad mängumaailma, siis hakkavad nad kohe liikuma mängija suunas, kuna neile on teada mängija asukoht. Püstoliga vastased liikuvad teatud kaugusele mängijast ning siis asuvad teda tulistama; kurikaga vastased liiguvad mängijale väga lähedale ja samuti hakkavad ründama teda; kilbiga vastased liiguvad mängitavale karakterile ka väga lähedale ja lükkavad mängijat. (Joonis 13)



Joonis 13. Lihtsa algoritmi diagramm.

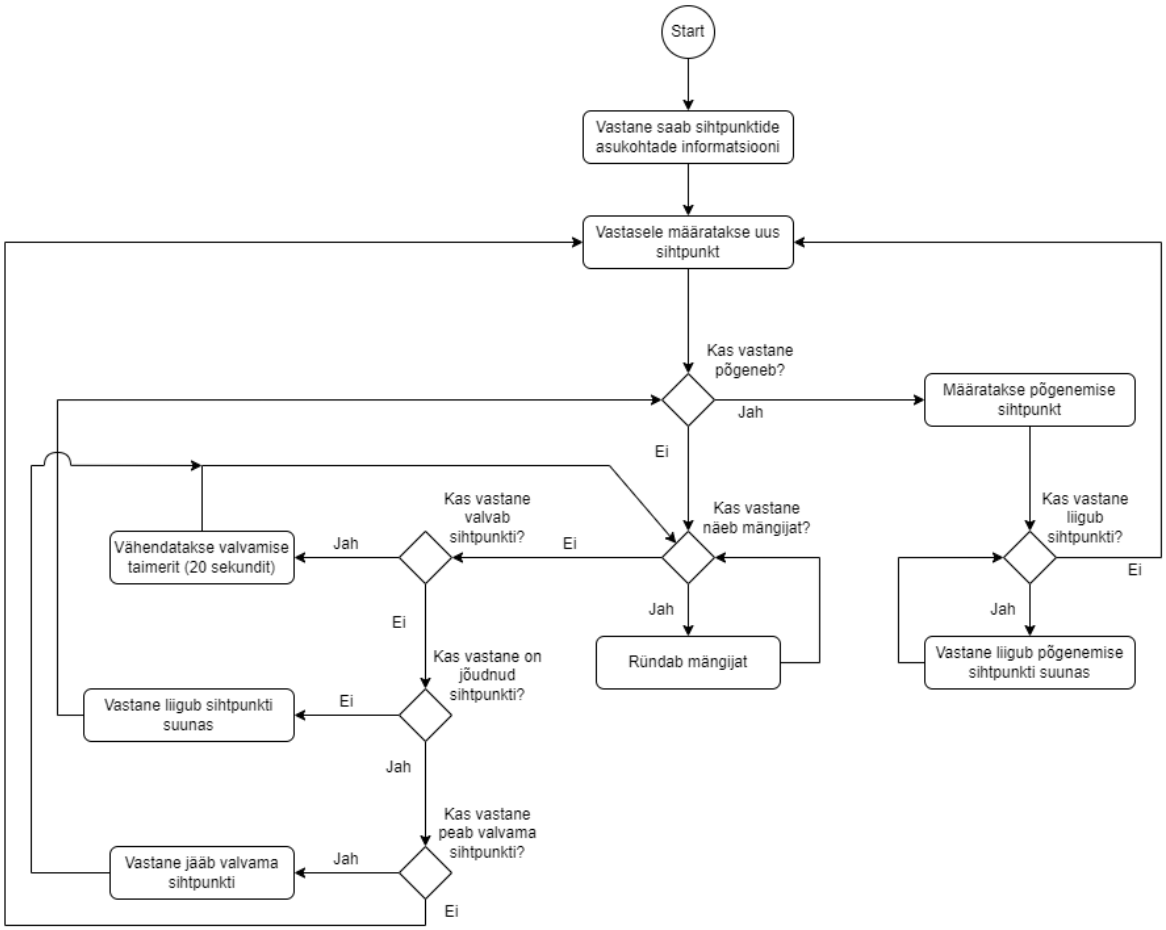
5.7 Oleku algoritm

Oleku algoritm põhineb eelpool peatükis mainitud oleku põhise otsustusalgoritmi põhimõtetest ja on proovitud antud arvutimängus realiseerida erinevaid olekuid, milles vastased olla saavad. Erinevate olekute väljamõtlemine ning realiseerimine võttis rohkem aega, kui lihtsa algoritmi arenduse puhul, aga arendusprotsess ei olnud nii keerukas, kui adaptiivse algoritmi arendamisel.

Oleku algoritmi puhul tekitatakse vastaseid mängumaailma iga 3 sekundi järel ning vastase tüüp, mis tekitatakse on juhuslik. Sarnaselt eelneva algoritmiga, tekitatakse oleku algoritmi puhul vastaseid lõpmatult, aga puudub lainega vastaste ilmumine.

Erinevate olekute realiseerimiseks on mängumaailma paigutatud nähtamatud sihtpunktid, kuhu vastased liiguvad. See tekitab mängijale efekti, et vastased patrullivad ala. Lisaks, on 30% tõenäosus, et vastane jääb sihtpunkti valvama 20 sekundiks, mis samuti annab mängijale mulje nagu vastased valvaksid teatud sihtpunkte.

Erinevalt teistest algoritmidest, ei ole vastastel kogu aeg teada mängija asukoht. Vastased hakkavad mängijat ründama ainult siis, kui mängija on liikunud piisavalt lähedale. Täiendavalt, kui vastase elu punktid langevad teatud piirist allapoole, siis vastase ründamise olek muutub hoopis põgenemiseks ning vastane valib suvaliselt ühe sihtpunkti mängumaailmas ja põgeneb sinna. Selle mõte on panna mängija mõtlema, et kas tal on mõistlik jälitada vastast, kui ta põgeneb sihtpunkti, kus võib ees oodata veel rohkem vastaseid. (Joonis 14)



Joonis 14. Oleku algoritmi diagramm.

5.8 Adaptiivne algoritm

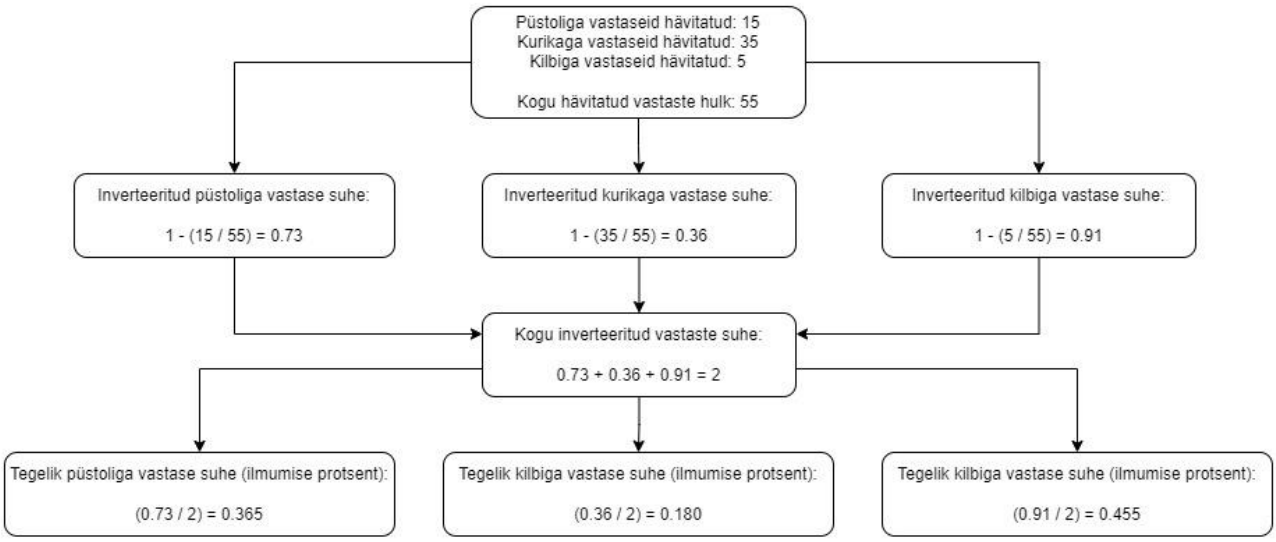
Adaptiivne algoritm on autori enda mõtetel põhinev loodud algoritm, mis proovib arvesse võtta mängija tegusi mängus ja vastavalt sellele, teha mängu kogemus dünaamilisemaks. Algoritmi arendusprotsess oli kõige keerukam ja aeganõudev teistest algoritmidest.

Vastaseid tekitatakse mängumaailma sarnaselt lihtsale algoritmile, aga peale 3. lainet, muutub vastaste ilmumine dünaamiliseks. Esimesed kolm lainet kogub mäng infot hävitatud vastaste kohta ning peale kolmandat lainet hakkab vastaste tekkimise mänguobjekt vastaseid mängumaailma tekitama vastavalt ilmumise protsendile. Vastaste tüüpide ilmumise protsente kuvatakse ka mängijale läbi kasutajaliidese. (Joonis 15, Joonis 16)

Joonis 15. Ilmumise protsendid esimese kolme laine ajal.

Joonis 16. Ilmumise protsendid peale kolmandat lainet.

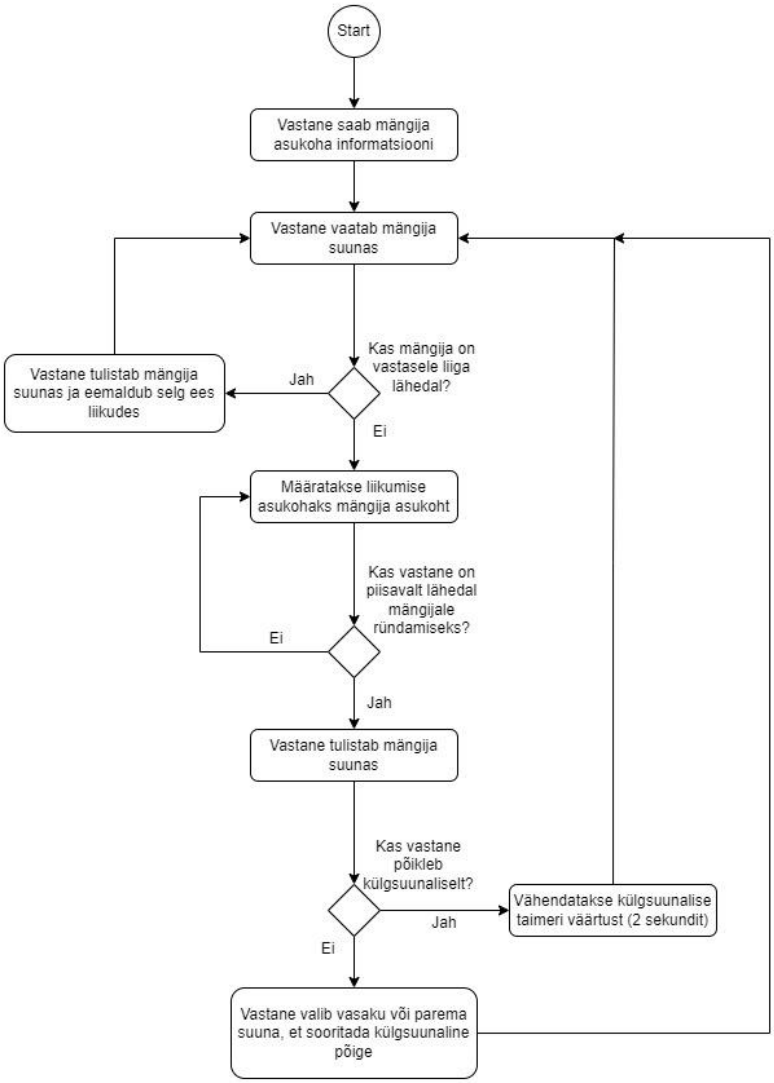
Ilmumise protsent sõltub sellest, mis vastast on kõige rohkem hävitatud. Näiteks, kui mängija otsustab, et ta hävitab kõige rohkem kurikaga vastaseid, siis kõige rohkem hävitatud vastaseid on kurikaga ning antud vastase tüübi ilmumise protsent langeb. Eksklusiivselt ühte tüüpi vastase hävitamise tulemust on ka näha Joonis 16-e peal, kus mängija on hävitanud peaaegu ainult kurikaga vastaseid. Ilmumise protsendi arvutamiseks leitakse alguses kogu vastase hulga ning siis leitakse iga vastase tüübi kohta eraldi inverteeritud vastaste hävitamise suhte. Selleks jagatakse iga vastase tüübi hävitatud kogus terve hävitatud vastase hulgaga ning tulemus lahutatakse 1-st. Seejärel liidetakse inverteeritud vastaste hävitamise suhted omavahel ning iga vastase tüübi tegelik ilmumise protsent arvutatakse nende enda inverteeritud vastaste hävitamise suhte jagamisel kogu inverteeritud suhtega. Sellega saavutatakse dünaamiline tekkimise süsteem, mille põhieesmärk on mängijale mängu kogemuse muutmine unikaalseks. (Joonis 17)



Joonis 17. Suvalise näitena välja toodud võimalik arvutuskäik ilmumise protsendi arvutamisel.

Ühtlasi, on ka antud algoritmis muudetud kahe vastase tüübi käitumist. Kurikaga vastane käitub täpselt samamoodi nagu lihtsas algoritmis, ainuke erinev asi vastase puhul on dünaamiline ilmumise protsent.

Püstoliga vastane hakkab mängumaailma ilmudes koheselt mängijat otsima ning kui piisavalt lähedale jõuab, siis alustab rünnakuga. Erinevalt teistest algoritmidest, üritab püstoliga vastane põigelda mängija kuulide eest. Selleks liigub vastane ettearvamatult iga kahe sekundi tagant külgsuunaliselt, et mängijal oleks raskem tabada vastast. Kui vastane otsustab külgsuunaliselt liikuda, siis valib ta suvaliselt kas vasaku või parema suuna. Suvalise suuna valik väldib olukorda, kus mängija suudab ennustada püstoliga vastase liikumist. Täiendavalt, liigub vastane selg ees mängija eest eemale, kui mängija otsustab talle liiga lähedale liikuda. Selline käitumine raskendab oluliselt mängija lähivõitluse rünnakut. (Joonis 18)



Joonis 18. Adaptiivse algoritmi püstoliga vastase diagramm.

Antud algoritmis on kilbiga vastane kõige erilisema käitumisega, kuna seekord teeb ta püstoliga vastasega koostööd. Nimelt, kilbiga vastane otsib kõige lähima püstoliga vastase ning hakkab teda oma kilbiga kaitsma. Kilbiga vastane liigub kõige lähima püstoli vastase ette ning suunab kilbi mängija suunas, et kaitsta püstoliga vastaseid mängija kuulide eest. Samal ajal tulistab püstoliga vastane kilbiga vastase selja tagant ja ei ole ohus mängija tulistamise vastu.

6 Arvutimängu testimine

Loodud ellujäämise arvutimängu testisid läbi autori poolt valitud 11 inimest. Testijate seas oli neid, kes väga tihti arvutimänge ei mängi ning ka inimesi, kes mängivad pidevalt arvutimänge. Testimine viidi läbi üks ühele sessioonides, kus testija mängis arvutimängu ning autor vaatles. Lisaks sellele, pani autor kirja mängimise ajal saadud tagasisidet mängu kohta. Peale mängimist küsis autor igalt testijalt küsimusi arvutimängu vastaste algoritmide kohta. (Lisa 3 – Küsitluse vastused)

Enne tulemuste analüüsimist on oluline arvestada teatud piirangutega testimisel. Kuna testijate hulk on 11 inimest, siis tulemuste põhjal tehtud järeldusi ei saa pidada absoluutseks tõeks. Lisaks, kuna testijad on autori poolt valitud ja autorile lähedased inimesed, siis tulemustes võib esineda teatud määral kallutatust, kuigi igale testijale sai öeldud, et ta avaldaks oma arvamust võimalikult ausalt. Seda kõike silmas pidades on testimise tulemused kindlasti kasulikud ning annavad olulist teavet, mis võib arvutimängude arenduses kasuks tulla.

6.1 Arvamused lihtsast algoritmist

Nagu algoritmi nimi ka ütleb, siis tegemist on lihtsamate arvuti poolt juhitud vastastega, kellel puudub keerukas käitumine. Algoritmi eesmärk oli anda kasutajale kogemus, mis oleks kõige tavapärasem ellujäämise arvutimängudes, kus vastased jooksevad mängija suunas ja ründavad. Täiendavalt, vastased pidid olema võimalikult lihtsad ja etteaimatavad.

Küsitluse esimese küsimusena uuriti testijatelt, kuidas nad hindavad enda kogemust mängides lihtsa algoritmiga skaalal 1-10, kus 1 on väga halb ja 10 on väga hea (Lisa 3, Tabel 3). Esimese küsimuse tulemuste põhjal on lihtsa algoritmi mängija kogemuse aritmeetiline keskmine 6,73 ning mediaan 7. Kõige madalam hinnang oli 3 ning kõige kõrgem oli 10.

Neljanda küsimusega soovis autor teada saada, millise algoritmiga oli kõige meeldivam mängida (Lisa 3, Tabel 4). Lihtne algoritm oli kõige meeldivam algoritm ainult ühele testijale ning põhjuseks oli algoritmi lihtsus võrreldes teistega. Lisaks, järgneva küsimusega (Lisa 3, Tabel 5) küsiti testijatelt ka kuidas erines nende valitud kõige meeldivam algoritm teistest ning lihtsa algoritmi puhul vastas Testija 5, et mängu oli lihtne läbida ja ei tekitanud paanikat.

Kuuenda küsimuse puhul vastasid testijad, mis algoritm neile kõige ebameeldivam oli (Lisa 3, Tabel 6). Neli inimest arvasid, et lihtne algoritm oli neile kõige ebameeldivam. Testijad arvasid, et algoritm on liiga lihtne, puine ja isegi igav. Täiendavalt, küsiti testijate käest kuidas erines kõige ebameeldivam algoritm teistest (Lisa 3, Tabel 7). Testijate arvates puudus lihtsal algoritmil lisa mängumehaanikad ja mõtlemisoskus vastastel. Lisaks, ei olnud vastaste vastu mängimine nauditav ning jäi tunne, et teiste algoritmide puhul ei olnud vastased nii agressiivsed.

6.2 Arvamused oleku algoritmist

Oleku algoritmi eesmärk oli anda kasutajale tunne, et vastased tunduksid targemad ning nende mängumaailmas olemise mõte ei ole ainult mängitava karakteri hävitamine. Selleks on vastastel mitu erinevat olekut, kus vastane teatud hetkedel patrullib, kaitseb või otsib teatuid sihtpunkte mängumaailmas. Lisaks, kui vastane tuvastab mängijat, siis asub teda ründama, aga vastane ei tee seda lõputult, vaid piisavalt suure kahju saamisel proovib vastane ka ära põgeneda.

Oleku algoritmi kohta küsiti teise küsimusena küsitluses testijatelt nende hinnangut enda mängu kogemusest oleku algoritmiga skaalal 1-10. Küsimuse tulemuste alusel on oleku algoritmi mängija kogemuse aritmeetiline keskmine 7,55 ning mediaan 7. Kõige madalam hinnang oli 5 ja kõige kõrgem hinnang oli 10.

Oleku algoritmiga mängimist pidasid kõige meeldivamaks 5 testijat. Testijatele meeldis asjaolu, et vastased jooksevad eest ära ning et neid pidi taga ajama. Lisaks, tundus testijatele, et see oli kõige lihtsam teistest algoritmidest ning samuti ka kõige võrdväärsem. Erinevused teiste algoritmidega olid peamiselt asjaolud, et vastased jooksid eest ära ning tundusid inimlikumad. Testijate sõnul andis see võimaluse kakelda vähemate vastastega korraga.

Ainult üks testija vastas, et kõige ebameeldivam algoritm tema jaoks oli oleku algoritm. Testijale ei meeldinud, et ta pidi vastaseid taga ajama ning see ei pakkunud talle lõbu.

6.3 Arvamused adaptiivsest algoritmist

Algoritmi eesmärk oli jätta mängijale mulje, et vastaste käitumine arvutimängus on adaptiivne, ehk vastased kohandavad või muudavad kuidagi enda käitumist mängija suhtes. Selleks sai loodud dünaamiline lainega vastaste ilmumise süsteem, mis muudab vastase tüübi tekkimise tõenäosust vastavalt sellele, mis tüüpi vastaseid on kõige rohkem hävitatud mängija poolt. Lisaks, osad vastased eemalduvad mängijast teatud kaugustel ning kasutavad rohkem meeskonnatööd kaitstes üksteist mängija rünnakute eest. Sellise käitumise eesmärk on panna mängijat mõtlema, kuidas ta enda tegudega saaks mängu kulgu mõjutada ning pakkuda mängijale unikaalset mänguelamust.

Testijate hinnangu saamiseks antud algoritmi jaoks küsiti küsitluse kolmanda küsimusega, millega sooviti teada saada nende mängu kogemuse hinnangut adaptiivse algoritmiga skaalal 1-10. Tulemuste põhjal on adaptiivse algoritmi mängija kogemuse aritmeetiline keskmine 7,73 ja mediaan 8. Kõige madalam hinnang oli 3 ja kõige kõrgem oli 10.

Kõige meeldivamaks osutus adaptiivne algoritm 5-le testijale, kuna testijatel jäi mulje, et vastastel on mängu tajumine hea ja ei jookse lihtsalt rünnates peale. Lisaks, oli algoritm testijate arvates huvitav ja tehniliselt lahedam, kuna vastased kaitsesid üksteist ja see pani mängijat ka rohkem mõtlema ning taktikaliselt käituma. Ühe testija jaoks oli ka algoritm meeldivam ainult selle tõttu, et enne seda algoritmi proovides sai harjutada teiste algoritmidega, siis adaptiivse algoritmini jõudes oli testija juba osavam. Erinevused, mida testijad algoritmi puhul märkasid olid asjaolud, et vastased kasutasid rohkem tiimitööd ja see pani testija uusi strateegiaid välja mõtlema. Lisaks, vastased ei jooksnud lihtsalt rünnates peale, mis pani testijaid ka pingutama, et kauem elus püsida.

Adaptiivne algoritm osutus kõige ebameeldivamaks 3-le testijale. Testijate arvates oli adaptiivne algoritm kõige keerulisem ning üks testija isegi väitis, et läks pisut paanikasse, kuna vastaseid oli liiga palju. Täiendavalt, ei meeldinud testijatele ka asjaolu, et kilbiga vastaseid tekkis liiga palju. Peamised erinevused teiste algoritmidega olid vastaste koostöö üksteisega ning kilbiga vastaste ette jäämine mängijale.

6.4 Muu tagasiside

Autor sai testijatelt ka väga palju muud põhjaliku tagasisidet, mis otseselt ei olnud seotud vastaste käitumise kohta, vaid üldiselt arvutimängu enda kohta. Antud tagasiside sisaldab visuaalseid elemente ja kasutajaliidesega seotud täiendusi, mis võiksid mängus esineda. Samuti, anti palju ideid, kuidas saaks mängu lisa arendustega meeldivamaks ja õiglasemaks teha. (Lisa 3, Tabel 8)

Testijad soovisid näha rohkem visuaalseid kinnitusi mängus toimuva kohta. Testijate arvates võiksid püstoliga vastased kogu aeg püstoleid väljas hoida, et nad oleksid üksteisest rohkem eristatavad ning vastast rünnates võiks olla ka visuaalne kinnitus, et vastasele on tõesti pihta saadud. Lisaks, võiks mängija enda püstol ka nähtaval olla ning vastaste elu punktid võiksid samuti kogu aeg nähtaval olla. Mängitava karakteri enda välimus võiks ka erinev olla vastastest.

Kasutajaliidese kohta sai autor kriitikat, et see on liiga üksluine ning hiire kursori asemel võiks olla mingi muu visuaalne sihtimisvahend. Lihtsa ja adaptiivse algoritmi puhul võiks olla ilmumise lainete number ka ekraanil näha, et mängija teaks, kui kaugele ta mängus jõudnud on. Täiendavalt, lisaks punktide arvule võiks mängijal näha olla tema tulistamise täpsuse protsent ja elu punktid võiksid olla ekraani üleval keskel kuvatud ribana, mitte numbritena paremal üleval nurgas.

Sisukamad edasiarenduse ideed olid testijate poolt näiteks uute teistsuguste relvade lisamine mängu, mis tulistaksid kuule kiiremini ning uus tugevam unikaalne vastane, keda oleks raskem hävitada. Testijate arvates võiks olla ka mängus erinevad mängurežiimid, näiteks aja peale mängimine või võimalus valida, missugused vastased ilmuvad mängumaailma ja millised mitte. Lisaks, kõrgemat meelelahutust pakuksid režiimid, kus oleks võimalik mängida koos teiste inimestega, kas läbi interneti või lokaalselt.

Mängu õiglasemaks muutmiseks pakkusid testijad välja elu ja kilbi punktide taastamise viisid, kus mängijal oleks võimalik elu punktide pakke üles korjata või peale mingit arvu raunde saaks mängija automaatselt mingi hulga elu või kilbi punkte juurde. Mõned testijad märkasid ka, et kurikas tundus liiga tugev rünnaku vahend ning püstol jällegi liiga nõrk. Testijate arvamusel võiks kurika tugevust vähendada ja püstoli tulistamiskiirust suurendada. Lisaks, arvasid testijad, et mängija enda tugevus ja võimekus võiks kaasas

käia vastaste kogusega, ehk mida rohkem vastaseid, seda tugevamaks saaks ka mängija, et mängijal oleks ka ellujäämise võimalus kõrgem suurema hulga vastaste vastu.

6.5 Järeldused ja võrdlus

Algoritmide võrdluseks saab tuua välja nende arvutatud testijate kogemuse hinnangute aritmeetilisi keskmisi ja mediaane. Tulemustest võib eeldada, et mida kõrgem on aritmeetiline keskmine ja mediaan, siis seda nauditavam kogemus oli mängida selle algoritmiga. Lisaks, saab hinnata ka nauditavuse taset vastavalt sellele, kui palju mainiti mingit algoritmi kõige meeldivamana ja millist kõige ebameeldivamana (Tabel 2).

Tabel 2. Algoritmide võrdlus testijate küsitluse tulemustest.

	Aritmeetiline keskmine	Mediaan	Mainiti kõige meeldivamana	Mainiti kõige ebameeldivamana
Lihtne algoritm	6,73	7	1 kord	4 korda
Oleku algoritm	7,55	7	5 korda	1 kord
Adaptiivne algoritm	7,73	8	5 korda	3 korda

Lihtsa algoritmi testijate kogemuse hinnangute aritmeetiline keskmine on kõige madalam ning algoritmi mainiti kõige ebameeldivamana kõige rohkem ja kõige meeldivamana kõige vähem võrreldes teiste algoritmidega. Antud testijatele enamasti ei pakkunud lihtne algoritm nii suurt meelelahutust, kui teised algoritmid. Selle põhjuseks on algoritmi lihtsus ja vastaste ebaintelligentne käitumine. Kuigi osadele testijatele meeldis, et lihtne algoritm ei olnud kuigi keerukas, siis enamuse arvates tegi see algoritmi igavaks ja üksluiseks.

Oleku algoritmi puhul on testijate kogemuse hinnangute aritmeetiline keskmine kõrgem lihtsa algoritmi omast ning pisut madalam adaptiivse algoritmi keskmisest. Algoritmi mainiti kõige meeldivamana 5 korda, mis oli sama ka adaptiivse algoritmi puhul, aga oleku algoritmi mainiti kõige ebameeldivamana ainult ühe korra, mis oli kõigi algoritmidega võrreldes parim tulemus. Vastaste käitumine antud algoritmi puhul pakkus suurt meelelahutust enamus testijatele, kuna vastased käitusid intelligentselt ja põgenesid mängija eest ära.

Testijate kogemuse hinnangute aritmeetiline keskmine oli kõige kõrgem adaptiivsel algoritmil. Algoritmi mainiti kõige meeldivamana 5 korda ning kõige ebameeldivamana 3 korda. Peamiselt meeldis algoritmi puhul vastaste meeskonnatöö ja asjaolu, et algoritm oli kõige keerulisem võrreldes teistega. Algoritmi keerukus oli ka põhjus, miks algoritmi mainiti 3 korda kõige ebameeldivamana.

Antud tulemuste põhjal, mis on piiratud testijate arvuga, eeldab autor, et kõige mõistlikum oleks arendada mängu oleku algoritmiga, kuna selle aritmeetiline keskmine on väga lähedal adaptiivse algoritmi tulemusele ning oleku algoritmi mainiti kõige vähem kõige ebameeldivamana. Lisaks, võtab autor arvesse ka asjaolu, et adaptiivse algoritmi arendus oli kõige keerukam teistest algoritmidest ning kui adaptiivne algoritm saavutas sarnased tulemused oleku algoritmiga, siis lisa vaev ei tasu ennast ära.

7 Kokkuvõte

Lõputöö raames sai uuritud arvutimängude arendamise üht olulist aspekti, milleks on arvuti poolt juhitud vastaste käitumine. Lisaks, uuriti kuidas on vastaste käitumine seotud mängija kogemusega ja arvutimängude müügiga.

Testimise läbiviimiseks sai loodud arvutimäng, mille mängides on võimalik valida kolme erineva algoritmi vahel. Kaks algoritmi sai valitud juba olemasolevatest materjalidest ning teiste arvutimängude näidetel. Kolmas algoritm oli autori enda mõtetel põhinev algoritm, mille eesmärk oli pakkuda mängijale unikaalset kogemust adaptiivse vastase käitumise näol.

Testimise tulemuste põhjal tuli välja, et kõige mõistlikum oleks arendada antud testijate hulga põhjal oleku algoritmi. Algoritmi arendus oli keskmise keerukuse tasemega võrreldes teistega, aga sai väga palju positiivset tagasisidet, mis tõstis algoritmi esile.

8 Resume

This thesis focuses on one important aspect of computer game development, which is the behavior of computer controlled enemies. In addition, it was investigated how the behavior of enemies is related to the player's experience and the sales of computer games.

In order to test player experience, a computer game was created, in which it is possible to choose between three algorithms. Two algorithms were chosen from existing materials and examples of other computer games. The third algorithm was based on the author's own ideas, where the aim was to provide the player with an unique experience in the form of adaptive enemy behavior.

Based on the results of the testing, it turned out that the most sensible option would be to develop the state algorithm based on the given testers. The development of the state algorithm was of medium complexity compared to other algorithms, but received a lot of positive feedback, which highlighted the algorithm.

Kasutatud kirjandus

[1] Dango, „Sometimes, the enemy makes the game,“ Destructoid, 30 September 2017. [Võrgumaterjal]. Saadaval: https://www.destructoid.com/sometimes-the-enemy-makes-the-game/. [Kasutatud 17 Veebruar 2024].

[2] S. Horti, „Why F.E.A.R.'s AI is still the best in first-person shooters,“ Rock Paper Shotgun, 3 Aprill 2017. [Võrgumaterjal]. Saadaval: https://www.rockpapershotgun.com/why-fears-ai-is-still-the-best-in-first-person-shooters. [Kasutatud 26 Veebruar 2024].

[3] R. Taljonick, „Shadow of Mordor's Nemesis system is amazing--here's how it works,“ Gamesradar+, 29 August 2014. [Võrgumaterjal]. Saadaval: https://www.gamesradar.com/shadow-mordor-nemesis-system-amazing-how-works/. [Kasutatud 28 Veebruar 2024].

[4] W. D'Angelo, „Middle-Earth: Shadow of Mordor Sales Top 2.4M Units Worldwide - PS4, Xbox One, Xbox 360, PS3, PC - News,“ VGChartz, 15 Detsember 2014. [Võrgumaterjal]. Saadaval: https://www.vgchartz.com/article/252478/middle-earth-shadow-of-mordor-sales-top-24m-units-worldwide-ps4-xbox-one-xbox-360-ps3-pc/. [Kasutatud 28 Veebruar 2024].

[5] „Middle-earth™: Shadow of Mordor™,“ Steam Revenue Calculator, [Võrgumaterjal]. Saadaval: https://steam-revenue-calculator.com/app/241930/middle-earthtm:-shadow-of-mordortm. [Kasutatud 28 Veebruar 2024].

[6] T. Thompson, „The Perfect Organism: The AI of Alien: Isolation,“ Game Developer, 31 Oktoober 2017. [Võrgumaterjal]. Saadaval: https://www.gamedeveloper.com/design/the-perfect-organism-the-ai-of-alien-isolation. [Kasutatud 28 Veebruar 2024].

[7] E. Makuch, „Alien: Isolation Sells 2.1 Million,“ Gamespot, 11 Mai 2015. [Võrgumaterjal]. Saadaval: https://www.gamespot.com/articles/alien-isolation-sells-2-1-million/1100-6427238/. [Kasutatud 28 Veebruar 2024].

[8] „Alien: Isolation,“ Steam Revenue Calculator, [Võrgumaterjal]. Saadaval: https://steam-revenue-calculator.com/app/214490/alien:-isolation. [Kasutatud 28 Veebruar 2024].

[9] Tarodev, „Pathfinding - Understanding A* (A star),“ YouTube, 16 November 2021. [Võrgumaterjal]. Saadaval: https://www.youtube.com/watch?v=i0x5fj4PqP4. [Kasutatud 6 Märts 2024].

[10] A. Mishra, „Creating your first 2D game with A* Algorithm,“ Hackerearth, 6 Oktoober 2016. [Võrgumaterjal]. Saadaval: https://www.hackerearth.com/blog/developers/creating-first-2d-game-algorithm/#:~:text=Games%20like%20Warcraft%20III%20use,a%20dragon%20on%20its%20way.. [Kasutatud 7 Märts 2024].

[11] DigiDigger, „How does videogame AI make its decisions? (FSM, Behaviour Trees, BDI, GOAP) | Bitwise,“ YouTube, 6 Jaanuar 2023. [Võrgumaterjal]. Saadaval: https://www.youtube.com/watch?v=ValJk15l_y8. [Kasutatud 6 Märts 2024].

[12] G. S. David M Bourg, „Chapter 9. Finite State Machines,“ O'REILLY, [Võrgumaterjal]. Saadaval: https://www.oreilly.com/library/view/ai-for-game/0596005555/ch09.html#:~:text=Finite%20state%20machines%20date%20back,determined%20by%20the%20player's%20actions.. [Kasutatud 6 Märts 2024].

[13] Tommi, „AI Trigger Actions Explained,“ Hiveworkshop, 8 September 2004. [Võrgumaterjal]. Saadaval: https://www.hiveworkshop.com/threads/ai-trigger-actions-explained.11244/. [Kasutatud 18 Märts 2024].

[14] XXXconanXXX, „AI - Basics and Triggers,“ World Editor Tutorials, 2005. [Võrgumaterjal]. Saadaval: https://world-editor-tutorials.thehelper.net/cat_usersubmit.php?view=15516. [Kasutatud 18 Märts 2024].

[15] T. Thompson, „History of AI in Games – HalfLife & Halo,“ Modl.ai, 11 Mai 2023. [Võrgumaterjal]. Saadaval: https://modl.ai/halflife-halo/. [Kasutatud 18 Märts 2024].

[16] juegoadmin, „Types of Gamers and the Bartle Player Taxonomy,“ Juegostudio, 19 Detsember 2022. [Võrgumaterjal]. Saadaval: https://www.juegostudio.com/blog/types-of-gamers. [Kasutatud 18 Märts 2024].

[17] b. ip, „From Casual to Core: A Statistical Mechanism for Studying Gamer Dedication,“ Game Developer, 5 Juuni 2022. [Võrgumaterjal]. Saadaval: https://www.gamedeveloper.com/business/from-casual-to-core-a-statistical-mechanism-for-studying-gamer-dedication. [Kasutatud 18 Märts 2024].

[18] A. Jack, „Understanding Game Stories: Storytelling, Interaction and Emergence in Video Games,“ Game Developer, 6 Juuni 2011. [Võrgumaterjal]. Saadaval: https://www.gamedeveloper.com/design/understanding-game-stories-storytelling-interaction-and-emergence-in-video-games. [Kasutatud 18 Märts 2024].

[19] G. Alex, „The Importance of Casual Gamers,“ YouTube, 14 September 2020. [Võrgumaterjal]. Saadaval: https://www.youtube.com/watch?v=5JMxrGNRHAY. [Kasutatud 30 Märts 2024].

[20] Cainos, „Pixel Art Top Down Basic,“ Itch.io, [Võrgumaterjal]. Saadaval: https://cainos.itch.io/pixel-art-top-down-basic. [Kasutatud 7 Veebruar 2024].

[21] ChrisJulch, „Top Down Shooter Character,“ Itch.io, [Võrgumaterjal]. Saadaval: https://chrisjulch.itch.io/top-down-shooter-pixel-art. [Kasutatud 10 Veebruar 2024].

[22] L. Taluste, Artist, *Graafikavara ja tekstuurid Ellujäämise Arvutimängu jaoks.* [Art]. 2024.

[23] U. Technologies, Unity Technologies, [Võrgumaterjal]. Saadaval: https://unity.com/. [Kasutatud 16 Märts 2024].

[24] GitHub, GitHub, Inc., [Võrgumaterjal]. Saadaval: https://github.com/. [Kasutatud 16 Märts 2024].

[25] Brackeys, „START MENU in Unity,“ YouTube, 29 November 2017. [Võrgumaterjal]. Saadaval: https://www.youtube.com/watch?v=zc8ac_qUXQY. [Kasutatud 15 Veebruar 2024].

[26] U. Technologies, „PlayerPrefs,“ Unity Documentation, 24 Aprill 2024. [Võrgumaterjal]. Saadaval: https://docs.unity3d.com/ScriptReference/PlayerPrefs.html. [Kasutatud 25 Aprill 2024].

[27] U. Technologies, „Box Collider 2D,“ Unity Documentation, 16 Märts 2024. [Võrgumaterjal]. Saadaval: https://docs.unity3d.com/Manual/class-BoxCollider2D.html. [Kasutatud 17 Märts 2024].

[28] U. Technologies, „Grid component reference,“ Unity Documentation, 16 Märts 2024. [Võrgumaterjal]. Saadaval: https://docs.unity3d.com/Manual/class-Grid.html. [Kasutatud 17 Märts 2024].

[29] U. Technologies, „Tilemap component reference,“ Unity Documentation, 16 Märts 2024. [Võrgumaterjal]. Saadaval: https://docs.unity3d.com/Manual/class-Tilemap.html. [Kasutatud 17 Märts 2024].

[30] U. Technologies, „Tilemap Renderer,“ Unity Documentation, 16 Märts 2024. [Võrgumaterjal]. Saadaval: https://docs.unity3d.com/Manual/class-TilemapRenderer.html. [Kasutatud 17 Märts 2024].

[31] U. Technologies, „Tile Palette,“ Unity Documentation, 19 August 2019. [Võrgumaterjal]. Saadaval: https://docs.unity3d.com/2019.1/Documentation/Manual/Tilemap-Palette.html. [Kasutatud 17 Märts 2024].

[32] Brackeys, „TILEMAPS in Unity,“ YouTube, 31 Jaanuar 2018. [Võrgumaterjal]. Saadaval: https://www.youtube.com/watch?v=ryISV_nH8qw. [Kasutatud 10 Veebruar 2024].

[33] U. Technologies, „Prefabs,“ Unity Documentation, 24 Aprill 2024. [Võrgumaterjal]. Saadaval: https://docs.unity3d.com/Manual/Prefabs.html. [Kasutatud 25 Aprill 2024].

[34] U. Technologies, „Animator component,“ Unity documentation, 24 Aprill 2024. [Võrgumaterjal]. Saadaval: https://docs.unity3d.com/Manual/class-Animator.html. [Kasutatud 25 Aprill 2024].

[35] U. Technologies, „Physics2D.Raycast,“ Unity Documentation, 25 Aprill 2024. [Võrgumaterjal]. Saadaval: https://docs.unity3d.com/ScriptReference/Physics2D.Raycast.html. [Kasutatud 25 Aprill 2024].

[36] J. French, „How to change a Sprite from a script in Unity (with examples),“ YouTube, 28 Juuni 2023. [Võrgumaterjal]. Saadaval: https://gamedevbeginner.com/how-to-change-a-sprite-from-a-script-in-unity-with-examples/. [Kasutatud 2024 Veebruar 2024].

[37] M. Wolf, „2D Shooting in Unity - 2022 Tutorial,“ YouTube, 3 November 2022. [Võrgumaterjal]. Saadaval: https://www.youtube.com/watch?v=ApLXuMeSVOI. [Kasutatud 10 Veebruar 2024].

[38] M. Wolf, „2D Melee in Unity Tutorial,“ YouTube, 8 Jaanuar 2023. [Võrgumaterjal]. Saadaval: https://www.youtube.com/watch?v=giJKCl-GVrU. [Kasutatud 26 Veebruar 2024].

[39] J. M. Games, „TOP DOWN MOVEMENT in UNDER 1 MINUTE! Unity 2D Tutorial,“ YouTube, 22 Detsember 2021. [Võrgumaterjal]. Saadaval: https://www.youtube.com/watch?v=tFblCEFQoTs. [Kasutatud 10 Veebruar 2024].

[40] PitilT, „Unity Tutorial: Top down shooter movement (mouse and keyboard),“ YouTube, 22 Detsember 2021. [Võrgumaterjal]. Saadaval: https://www.youtube.com/watch?v=sBmsqVwGd2E. [Kasutatud 10 Veebruar 2024].

[41] R. Studio, „Unity 2D Pathfinding with NavMesh tutorial in 5 minutes,“ YouTube, 21 Juuli 2023. [Võrgumaterjal]. Saadaval: https://www.youtube.com/watch?v=HRX0pUSucW4. [Kasutatud 21 Veebruar 2024].

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina, Harold Otsus

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Arvuti poolt juhitud vastaste käitumine arvutimängudes ja ellujäämise arvutimängu arendus", mille juhendaja on Lembit Jürimägi
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

11.05.2024

1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.

Lisa 2 – Ellujäämise arvutimängu repositoorium

Veebiaadress: https://github.com/HaroldOtsus/SurvivalGame.

Lisa 3 – Küsitluse vastused

1. Kuidas hindaksid oma kogemust mängides “Lihtne algoritm” skaalal 1-10, kus 1 on väga halb ja 10 on väga hea?

2. Kuidas hindaksid oma kogemust mängides “Oleku algoritm” skaalal 1-10, kus 1 on väga halb ja 10 on väga hea?

3. Kuidas hindaksid oma kogemust mängides “Adaptiivne algoritm” skaalal 1-10, kus 1 on väga halb ja 10 on väga hea?

Tabel 3. Küsitluse esimese, teise ja kolmanda küsimuse vastused.

	Lihtne algoritm	Oleku algoritm	Adaptiivne algoritm
Testija 1	6.	8.	7.
Testija 2	7.	10.	8.
Testija 3	7.	10.	10.
Testija 4	5.	7.	10.
Testija 5	8.	6.	3.
Testija 6	7.	7.	8.
Testija 7	9.	10.	5.
Testija 8	10.	5.	9.
Testija 9	8.	9.	10.
Testija 10	3.	5.	8.
Testija 11	4.	6.	7.

4. Millise algoritmiga oli kõige meeldivam mängida? Miks?

Tabel 4. Küsitluse neljanda küsimuse vastused.

Testija 1	Oleku algoritmiga. Vastased käitusid teistmoodi ning jooksid eest ära.
Testija 2	Oleku algoritm oli kõige huvitavam. Vastaseid pidi taga ajama.
Testija 3	Oleku algoritm. Vastased jooksevad naljakalt eest ära.
Testija 4	Adaptiivne algoritm. Vastastel on mängu tajumine hea ning ei jookse niisama rünnates peale.
Testija 5	Lihtne algoritm. See oli kõige lihtsam.
Testija 6	Adaptiivne algoritm. Kui ma selle algoritmini jõudsin, siis oskasin juba osavamalt mängida.
Testija 7	Oleku algoritm. See oli kõige lihtsam.
Testija 8	Adaptiivne algoritm. See oli huvitav, et nad üksteist kaitsesid. Pidin rohkem mõtlema ja taktikat vastaste vastu kasutama.
Testija 9	Adaptiivne algoritm. Tundus tehniliselt kõige lahedam.
Testija 10	Adaptiivne algoritm. Vastaseid ei olnud liiga palju ega liiga vähe.
Testija 11	Oleku algoritm. Kuna mul oli piiratud relvade kasutus võrreldes vastastega, siis see algoritm tundus kõige võrdväärsem.

5. Kuidas erines kõige meeldivam algoritm teistest algoritmidest?

Tabel 5. Küsitluse viienda küsimuse vastused.

Testija 1	Vastased jooksid eest ära. Erineb suuresti teistest algoritmidest.
Testija 2	Vastaseid pidi taga ajama ning tundusid inimlikumad.
Testija 3	Vastased jooksevad eest ära, mis annab võimaluse vähematega korraga kakelda.
Testija 4	Tegi mängu põnevamaks, pidin ise ka mõtlema ja strateegiat kasutama vastaste vastu.
Testija 5	Seda oli lihtne läbida ning ei tekitanud paanikat.
Testija 6	Suurema kambaga ründasid. Kasutasid rohkem tiimitööd.

Testija 7	Vastased jooksid eest ära.
Testija 8	Vastased kaitsesid üksteist.
Testija 9	Pani mind mõtlema, mida ise lisaks ja muudaks mängus.
Testija 10	Vastased ei jooksnud eest ära, aga samas ei jooksnud nad ka lihtsalt rünnates peale. Pani mind natuke pingutama ka, et elus püsida.
Testija 11	Kui vastased saavad viga, siis nad jooksevad ka eest ära.

6. Millise algoritmiga oli kõige ebameeldivam mängida? Miks?

Tabel 6. Küsitluse kuuenda küsimuse vastused.

Testija 1	Lihtne algoritm. Liiga lihtne.
Testija 2	Adaptiivne algoritm. Kilbiga vastaseid tekkis liiga palju.
Testija 3	Adaptiivne algoritm. See on kõige keerulisem.
Testija 4	Lihtne algoritm. Liiga üksluine ja igav.
Testija 5	Adaptiivne algoritm. Ma läksin paanikasse, kuna vastaseid oli liiga palju ja mind piirati ümber.
Testija 6	Ei olnud sellist algoritmi, mis ei meeldiks.
Testija 7	Kõikide algoritmidega oli ebameeldiv mängida, sest mul oli raske relvi vahetada.
Testija 8	Oleku algoritm. Pidin taga ajama vastaseid.
Testija 9	Mitte ühegagi.
Testija 10	Lihtne algoritm. Vastaseid lihtsalt ei olnud vahe peal.
Testija 11	Lihtne algoritm. Kõige puisem ja kuna see oli mu kõige esimene katse mängus, siis ei läinud kõige paremini.

7. Kuidas erines kõige ebameeldivam algoritm teistest algoritmidest?

Tabel 7. Küsitluse seitsmenda küsimuse vastused.

Testija 1	Tundus liiga lihtne ja tuim. Puudusid lisa mängumehaanikad.
Testija 2	Tegid koostööd üksteisega.
Testija 3	Kilbiga vastased jäid kogu aeg ette.
Testija 4	Lihtsalt lased vastaseid ja nad jooksevad kogu aeg peale. Vastastel puudub muu mõtlemisoskus.
Testija 5	Mulle tundus, et vastased liikusid kuidagi kiiremini.
Testija 6	-
Testija 7	Adaptiivne algoritm tundus kõige raskem olevat, kuna vastased tegid koostööd.
Testija 8	Pidin vastaseid taga ajama.
Testija 9	-
Testija 10	Vastaste vastu mängimine ei pakkunud lõbu.
Testija 11	See tundus kõige lihtsam, kuna vastased kohe ründasid. Teiste algoritmide puhul ei olnud vastased nii agressiivsed.

8. Muud kommentaarid?

Tabel 8. Küsitluse kaheksanda küsimuse vastused.

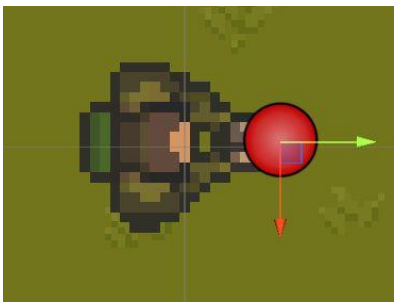
Testija 1	Vastaseid oli mõnus eristada, kuigi püstoliga vastased võiksid kogu aeg püstoleid väljas hoida, et oleks aru saada, mis tüüpi vastased nad on. Kui mängija vastasele pihta saab, siis võiks olla ka visuaalne kinnitus, et vastasele on tõesti pihta saadud. Kurikas on liiga tugev, püstol võiks tugevam olla. Normaalne kogemus, hea videomäng.
Testija 2	Püstoli tulistamiskiirus võiks kiirem olla.

	Võiks olla ka kiirema tulistamiskiirusega relva valik. Näiteks, kui oled mängus mõned voorud juba edasi saanud, siis saad selle kuskilt üles korjata. Mängija tugevus võiks kaasas käia koos vastaste kogusega. Elu punktide pank võiks suuremaks minna või mängija võiks rohkem kahju teha vastastele, mida kaugemale mängus mängija jõuab. Kui püstol on käes, siis see võiks olla nähtaval. Ilmumis lainete arv võiks ka kuskil kirjas olla ekraani peal.
Testija 3	Kui oleks võimalik kuidagi mängijat täiustada, siis oleks mäng lõbusam. Kui oleks reaalne võimalus pikemalt ellu jääda, siis mängu kogemus oleks parem.
Testija 4	Iga viies raund võiks teha väikese pausi vastaste tekitamisest, et vastaseid mängumaailma liiga palju ei koguneks. Kui kõik vastased on ühes raundis hävitatud, siis võiks järgmine raund kohe hakata. Kui 50 punkti on kogutud, siis võiks kilpide elud taastada, et saaks uuesti kilpi kasutada. Vastaste elu punkte võiks ka näha olla. Raundi number võiks ekraanil näha olla. Kui vastased ilmuvad, siis nad võiksid natuke oodata, enne kui nad mind ründama hakkavad.
Testija 5	Elu punkte võiks ilmuda maailma mingi hetk, et oleks võimalik elusi taastada.
Testija 6	Mingi tugevam vastane võiks tulla mingi hetk, nagu boss fight. Vastastel võiks relvad maha kukkuda ja mängijal võiks olla võimalus neid üles korjata. Mängija võiks ka tugevamaks saada mängu jooksul. Võiks olla ka võimalik kilbiga rünnata.

Testija 7	Värvid meeldivad. Vastased näevad liiga sarnased välja mängijaga. Maailm on ka hea, saab kasutada enda kasuks ära. Näiteks, seini kasutada varjumiseks. Maailm on piisavalt suur, aga mitte liiga suur, et see kuidagi halvalt mõjutaks mängimise kogemust. Võiks olla mingi lõpp eesmärk. Elu punktid võiksid olla ekraani keskel üleval ja ribana, mitte numbritena. Võiks olla valik, et püstoliga vastaseid ei tekiks mängu üldse. Relvade vahetamine võiks ka võimalik olla hiire rullikuga. Mängu mängimine maandas pingeid ja viis eksami mõtted mujale.
Testija 8	Võiks olla võimalik vastaseid tagasi lükata, aga mitte lähivõitluse vastaseid. Tulistamise täpsuse protsenti võiks ka kuskil näha olla. Vastased võiksid ka ennustada mängija liikumist ning üritada tema liikumis teekonda takistada. Võiks olla mingi aja peale mängurežiim. Kasutajaliides oli liiga üksluine. Püstolil võiks olla mingi laadimis süsteem, et ei saaks kogu aeg lõputult tulistada. Elu pakid võiksid olla maailmas, mis taastavad elu punkte. Võiks olla mängumaailm iga kord erinev. Kui vastaste kilp kaob ära, siis vastane saab kohe tulistada ja see on ebaõiglane. Hiire kursori asemel võiks olla midagi muud sihtimiseks. Võiks olla mingi hekt mingi tugevam vastane vastas. Nagu mingi kuulipildujaga boss.

Testija 9	Väga intrigeeriv mäng. Kuna mäng sai liiga kiiresti läbi, siis võiks olla võimalus elu punkte korjata. Kui ma olen mängus juba parem, siis tahaksin mingit raskemat versiooni. Tore oleks mängida kellegagi kõrvuti ja jagatud ekraani peal. Saaks kellegagi mängimise emotsioone jagada. Mitme inimesega mängimis versioon võiks ka olla. Vastavalt mängija tasemele saab mängu nautida erinevalt.
Testija 10	Kilbiga vastased muutuvad liiga kiiresti püstoliga vastasteks ja tulistavad liiga kiiresti.
Testija 11	Tegelasel võiks raskem kaal olla liikudes. Võiks kuidagi saada põigelda vastaste eest. Rohkem mehaanikat oleks mängus vaja, hetkel liiga pealiskaudne. Relva valik võiks olla nuppude peal, mitte aint TAB nupu peal. Mängijal võiks natuke aega võtta enne, kui ta saavutab maksimaalse liikumiskiiruse. Võiks mängus olla ka mingid helid. Püstoliga vastane võiks enne tulistamist natuke sihtida, et oleks võimalik reageerida tema tulistamise peale. Mängijal võiks olla mingi granaat suurema grupi vastaste vastu.

Lisa 4 – Ellujäämise mängu vastaste erinevad pildid



Joonis 19. Püstoliga vastane koos punase tulistamispunktiga.

Joonis 20. Kurikaga vastane ja ristküliku kujuline *Box Collider 2D* komponent kurika otsas.

Joonis 21. Kilbiga vastane koos välja toodud kilbi alamobjektiga.

Joonis 22. Püstoliga vastane kuvab *Raycasting* kiirt mängija suunas.

Joonis 23. Vastase liikumist kujutavad *sprite*'id.