Tartu Ülikool
Arvutiteaduse instituut
Informaatika õppekava

Arlo Tammekun

Vee simulatsiooni loomine Blastronaut mängule

Bakalaureusetöö (9 EAP)

Juhendaja Jaanus Jaggo, MSc

Tartu 2022

Vee simulatsiooni loomine Blastronaut mängule

Lühikokkuvõte:

Uurimistööd on näidanud, et interaktiivsus mängib olulist rolli videomängu nautimises ja kaasahaarava mängukeskkonna pakkumine motiveerib mängijaid hiljem tagasi tulema. Käesolevas bakalaureusetöös kirjeldatakse videomängudes kasutusel olevaid vee simulatsiooni meetodeid ning luuakse vee simulatsioon Godot mängumootoris arendatud videomängule Blastronaut. Töö lõpus viiakse läbi jõudlustestimine simulatsiooni jõudluse hindamiseks ning kasutajatestimine, et hinnata simulatsiooni visuaalset ühtivust, füüsikalist käitumist ja mõju mängule.

Võtmesõnad:

arvutimäng, vedeliku simulatsioon, arvutigraafika, rakuautomaat

CERCS: P175 Informaatika, süsteemiteooria

Water Simulation for the Game Blastronaut

Abstract:

Research has shown that interactivity plays an important role in video game enjoyment and that providing players with an immersive game environment motivates them to later return to the game. This Bachelor's thesis describes various methods of water simulation used in video games. A water simulation is created for the video game Blastronaut, which is developed in the Godot game engine. Performance testing is conducted to evaluate the performance of the simulation. User testing is conducted to evaluate its physical behavior, its effect on the game and how well the simulation blends in with the visual style of the game.

Keywords:

computer game, fluid simulation, computer graphics, cellular automaton

CERCS: P175 Informatics, systems theory

Sisukord

Sissejuhatus

Paljude arvutimängude tähtis osa on mängijatele kaasahaarava mängukeskkonna pakkumine. Üheks viisiks kuidas mängukeskkonda mitmekesisemaks muuta, on lisada sellele vee simulatsioon. Dünaamiline vesi, mis voolab ja reageerib maailma muutustele, teeb muidu staatilise ja igavana tunduva keskkonna elavamaks. Samuti lisab see mängija kogemusele variatsiooni ning annab mängijale uue mänguelemendi.

Blastronaut on kõrvalt vaates seiklusmäng, mida on kujutatud joonisel 1 oleval ekraanitõmmisel. Mängijal on kasutada reaktiivranits (ing. k. *jetpack)*, millega saab ta maailmas ringi lennata ja erinevad plahvatavaid tööriistad, millega on võimalik keskkonda hävitada. Viimase abil saab mängija kaevandada mineraale, mida saab raha eest maha müüa, et endale paremat varustust osta. Parem varustus lubab mängijal kaevandada efektiivsemalt, liikuda kiiremini, üle elada rohkem vigastusi ning kaasas kanda rohkem kütust. Tegevus toimub protseduuriliselt genereeritud lõpmatus maailmas, milles on hulk erinevaid bioome ning tulnukaid. Blastronaut on arendatud Godot[1] mängumootoriga.

Joonis 1. Ekraanitõmmis mängust Blastronaut.

[1] https://godotengine.org/

Bakalaureusetöö eesmärk on luua Blastronaut mängule sobiv vee simulatsioon, mis vastaks järgmistele nõuetele:

1. Töötab mängu olemasoleva protseduurilise generatsiooniga.
2. Reageerib maailma muudatustele reaalajas.
3. Ühtib ülejäänud mängu visuaalse stiiliga.

Esimeses peatükis kirjeldatakse süsteeme, mille abil mängudes vedelikke simuleeritakse. Teises peatükis kirjeldatakse simulatsiooni implementeerimise protsessi. Kolmandas peatükis viiakse läbi kasutajakogemuse testimine, jõudlustestimine ja analüüsitakse vastavate eesmärkide täitmist.

1. Vedeliku simuleerimine mängudes

Videomängudes on olulisel kohal interaktiivsus. Mängijad ootavad, et nad saaksid mängumaailma ise võimalikult palju mõjutada. Näiteks leidsid Klimmt jt [1] oma uurimistöös, et interaktiivsus mängib olulist rolli videomängu nautimises. Enamikes mängudes ei saa mängijad keskkonda ise palju mõjutada, ning see on kunstniku poolt eelnevalt loodud. Sellised keskkonnad võivad tunduda mängija jaoks elutud, sest nad ei saa sellega eriti interakteeruda. Selle probleemi üheks võimalikuks lahenduseks on lisada mängu vedelike simulatsioon, mida mängija saab mõjutada ja mis seeläbi suurendab kaasahaaravust (ing. k. *immersion*). Tanskanen [2] kirjutab oma lõputöös, et kaasahaaravuse suurendamine motiveerib neid hiljem tagasi tulema.

1.1 Simulatsiooni meetodid

Vedelike simuleerimiseks on olemas varieeruva keerukuse ja täpsusega meetodeid. Juba 19. sajandist saati on vedelike liikumiseks kasutatud Navier-Stokesi võrrandeid. Kellomäki [3] leiab, et enamik meetodeid nende lahendamiseks pole ette nähtud reaalajalisteks eesmärkideks ning on väga aeglased. Samuti väidab Kellomäki, et enamikel videomängudel ei ole vaja nende võrrandite poolt pakutud realismi taset.

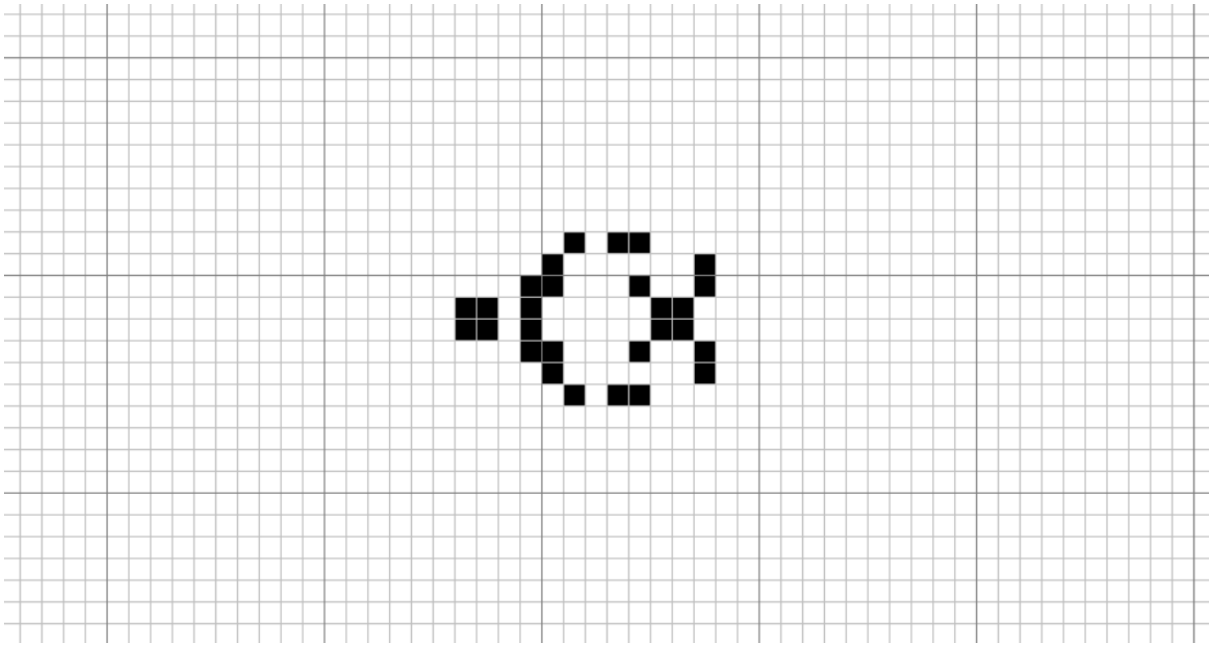
Amada, Kellomäki ning Sandu jt [3 - 5] leiavad, et kõiki vedeliku simulatsioone saab jaotada kahte kategooriasse: võrel (ing. k. *grid*) põhinevad ehk Euleri laadsed (ing. k. *Eulerian*) ja osakestel põhinevad ehk Lagrange'i laadsed (ing. k. *Lagrangian*). Amada kirjeldab Euleri laadset simulatsiooni järgnevalt. Vedeliku füüsikalised omadused on talletatud punktides, mis on fikseeritud taustsüsteemis ja ei liigu koos vedelikuga. Vedeliku liikumise puhul omadused antud punktis muutuvad. Amada ütleb, et Lagrange'i laadses simulatsioonis vedeliku omadused esindavad mingit osa kogu vedelikust ning liiguvad koos selle osaga.

Järgnevalt vaadatakse lähemalt kolme mängudes enamlevinud vedeliku simuleerimise viisi: rakuautomaate, osakestel põhinevaid süsteeme ja võrestikupõhist süsteemi. Neist viimane ei simuleeri vedelikke reaalajas.

1.1.1 Rakuautomaadid

Forsyth [6] kirjeldab rakuautomaati kui süsteemi, mille puhul on maailm jagatud ruudustikuks, mis koosneb kindla suurusega rakkudest. Selles ruudustikus on igal rakul arvuliste väärtustega omadused nagu rõhk, temperatuur, vee kogus jne. Iga iteratsioon võrreldakse igat rakku tema naaberrakkudega ning tehakse muudatusi sõltuvalt nende olekust. Muudatuse olemus sõltub kehtestatud reeglitest.

Rakuautomaadid on Euleri laadsed simulatsioonid. Amada [4] leiab, et Euleri laadse simulatsiooni eelised on teostamise ja sujuvate tulemuste saavutamise lihtsus. Ta väidab, et taolise simulatsiooniga on raske saavutada head täpsust. Forsyth [6] leiab, et sellise süsteemi täpsus on suhteliselt madal kuid näeb enamikel juhtudel piisavalt hea välja ning töötab kiiresti. Ta toob välja, et rakuautomaadi abil simuleeritud vee puhul on võimalik veerõhku simuleerida, kui see teha pisut kokkusurutavaks. See tähendab, et veerakus sisalduva vee kogus on suurem kui sellest kõrgemal oleva raku vee kogus. Maksimaalse vee suurendamine ainult 1% jagu iga raku kohta lubab simuleeritud veel näidata tavalisi vee omadusi. Näiteks U-kujulise toru sisse valatud vee kõrgus võrdsustub mõlema haru sees samale tasemele.



Joonis 2. Veebiversioon Conway elumängust.[2]

Üks tuntumaid rakuautomaate on joonisel 2 kujutatud Conway elumäng. Elumäng koosneb lõpmatust kahemõõtmelisest ruudustikust. Iga ruut (edaspidi rakk) on kas elus või surnud.

[2] https://conwaylife.com/

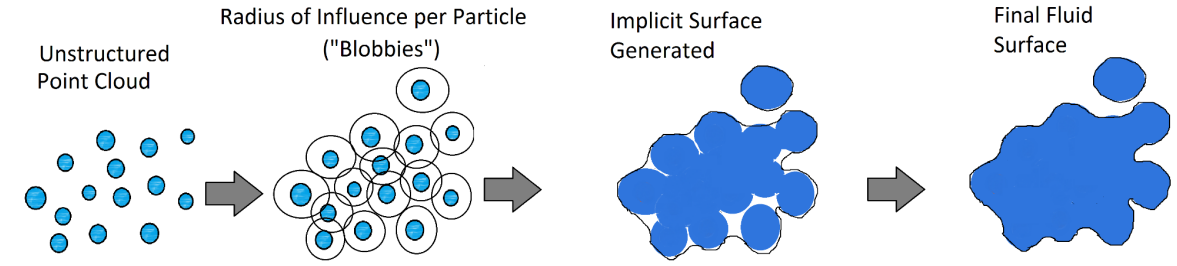
Joonisel 2 on elusad rakud kujutatud mustana ning surnud rakud kujutatud valgena. Kõikidele rakkudele rakendatakse kindlaid reegleid. Gardneri [7] sõnul on need reeglid järgmised:

1. Iga elus rakk, millel on 2 või 3 elusat naaberrakku jääb ellu.
2. Iga elus rakk, millel on rohkem kui 3 või vähem kui 2 elusat naaberrakku muutub surnud rakuks.
3. Iga surnud rakk millel on täpselt 3 elusat naaberrakku muutub elavaks rakuks.

Reegleid rakendatakse kõikidele rakkudele korraga. Igat reeglite rakendamist loetakse üheks põlvkonnaks.

1.1.2 Osakestel põhinev süsteem

Osakestel põhinevad süsteemid on Lagrange'i laadsed simulatsioonid. Sandu jt [5] kirjeldavad osakestel põhinevat süsteemi kui struktureerimata osakeste pilv. Nende väitel on igal osakesel selles pilves hulk kindlad omadusi milleks võib olla asukoht, kiirus, mass, tihedus, rõhk, jõuvektor jt. Selles süsteemis tehakse osakestele muudatusi sõltuvalt osakeste omaduste väärtustest ning nende lähedusest teistele osakestele. Veeosakeste realistlikuks käitumiseks kasutatakse nende liikumise arvutamiseks tihti lihtsustatud versiooni Navier-Stokes'i võrranditest.



Joonis 3. Osakestest koosneva vedeliku veepinna koostamine. [5]

Sellise süsteemi puhul osakeste kogum vee moodi välja ei näe. Kellomäki [3] sõnul peab osakeste kogumit siluma, et vältida individuaalsete osakeste nägemist. Selline protsess on kujutatud joonisel 3. Kellomäki väitel on üks viis selle saavutamiseks leida tulemuse pideva tihedusega pind (isopind) ning see trianguleerida. Näiteks kasutades piiravate kuubikute (ing. k. *marching cubes*) või piiravate ruutude (ing. k. *marching squares*) algoritmi, mida kasutas ka Janno [8] protseduuriliste olendite loomiseks.

1.1.3 Võrestikul põhinev süsteem

Osades mängudes on kaasatud vesi ilma reaalajalise simulatsioonita. Selle asemel kasutatakse eelnevalt koostatud võrestikku (ing. k. *mesh*), mis on eelarvutatud liikumisega või ilma liikumiseta. Vee dünaamilise liikumise välimust simuleeritakse erinevate võtete abil. Näiteks veepinna üles-alla liigutamine, et tekitada lainetamise välimus. Seda on võimalik saavutada kasutades kõrgusvälja (ing. k *heightfield*).

Selline lähenemine ei luba mängijal vee asukohta oluliselt mõjutada. Veele liikumise lisamiseks on see enamasti ette arvutatud. Näiteks selleks, et veealune koobas tühjaks voolaks kui mängija nupule vajutab, saab eelnevalt vee liikumise animatsiooni kõik kaadrid välja arvutada. Mängu käigus vahetatakse iga kaader võrestik välja eelnevalt simuleeritud võrestiku vastu.

1.2 Jõudlus

Meeldiva mängukogemuse jaoks peab mäng töötama piisavalt kiiresti, et tagada sujuv mängukogemus. Hagström [9] leiab, et minimaalne aktsepteeritav kaadrisagedus videomängudes on 60 kaadrit sekundis. Selle järjepidevaks saavutamiseks ei tohi ühe kaadri arvutamine võtta rohkem kui 16.6ms. Juhul kui mängu peafookus ei ole vee simulatsioon on vee liikumise arvutamiseks on saadaval väike osa sellest.

Vedeliku liikumise simuleerimine võib olla väga arvutuslikult kallis protsess. Lundelli [10] sõnul on enamus seniseni tehtud vedeliku simulatsiooni uurimusi keskendunud täpsetele füüsilistele simulatsioonidele või visuaalsetele animatsioonidele. Ta väidab ka, et rakupõhistes liivakastimängudes pole realism oluline. See lubab meil asendada keerulised arvutused lihtsamate kuid vähem realistlike tulemustega arvutustega.

Kellomäki [3] sõnul kasutatakse tihti võre põhistes simulatsioonides madala vee võrrandeid (ing. k. *shallow water equations*) ja osakeste põhistes simulatsioonides siledate osakeste hüdrodünaamikat (ing. k. *smooth particle hydrodynamics*). Mõlemad meetodid põhinevad lihtsustatud Navier-Stokes'i võrranditel.

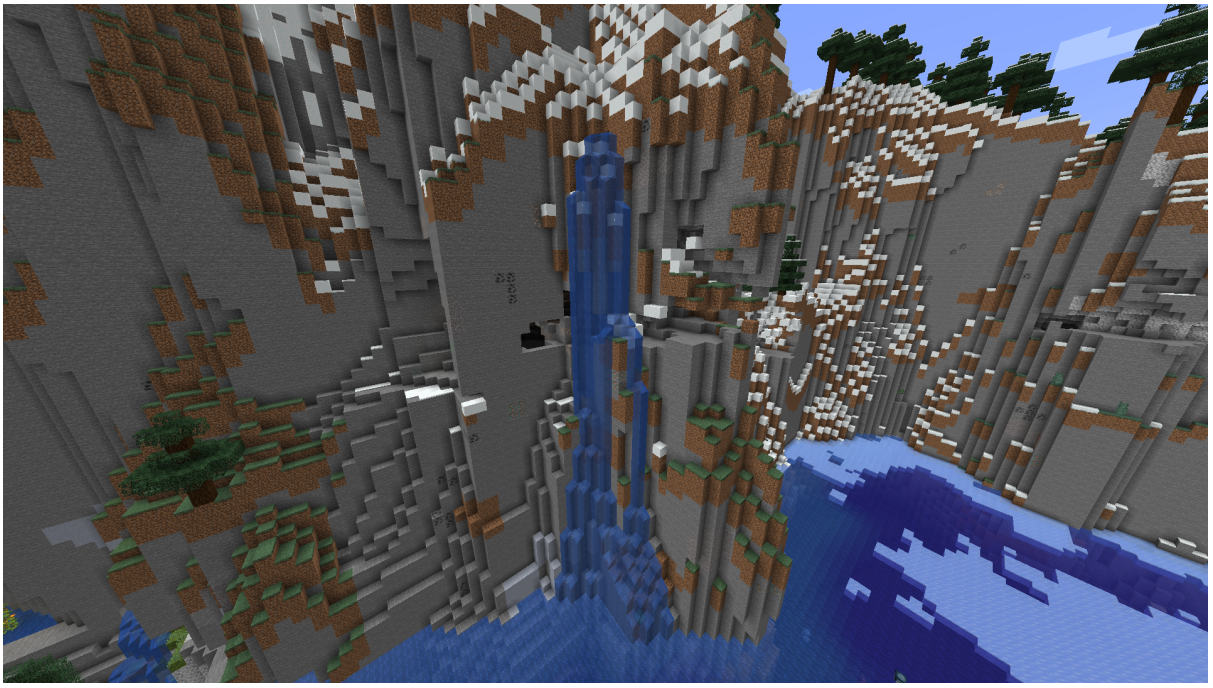
1.3 Näited vee simulatsioonidest mängudes

Vee simulatsioonid on mänginud erinevaid rolle paljudes populaarsetes mängudes. Vee simulatsioon võib olla kogu mängu keskne idee või peaaegu tähtsusetu visuaalne detail.

Selles alapeatükis antakse ülevaade mõnest populaarsest mängust, mis sisaldavad vett ning kirjeldatakse nende vee simulatsioone.

1.3.1 Minecraft[3]

Minecraft on kolmemõõtmeline liivakastimäng, mis on teeninud endale kõigi aegade enimmüüdud videomängu tiitli. Mängijad saavad avastada protseduuriliselt genereeritud maailma, koguda ressursse, meisterdada tööriistu ja ehitada ehitisi. Mängumaailm on peaaegu piiritu ning sisaldab suurt arvu erinevaid bioome ning ehitisi. Mängu võib kogeda üksi või koos teiste mängijatega. Vesi võimaldab mängijatel selles ujuda ja sellel paadiga sõita. Veest võib leida erinevaid kalu, koletisi, taimi ja struktuure.



Joonis 4. Ekraanitõmmis mängust Minecraft.

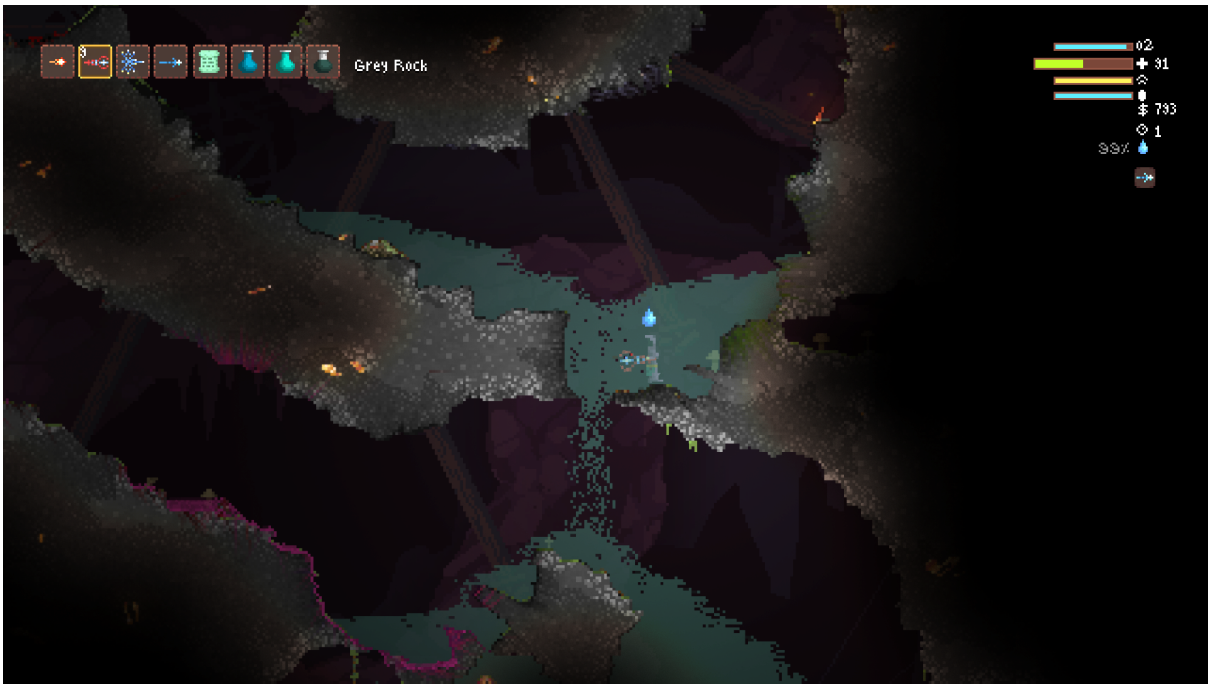
Kasutusel rakuautomaadi põhine süsteem. Joonisel 4 on kujutatud Minecraftis esinev vesi. Vee tuumaks on allikaplokid. Nendest levib vesi horisontaalselt külgedele ja alla. Külgedele levinud veeploki kõrgus on väiksem kui veeplokk, millest see levis. Alla voolates on vesi täisploki kõrgusega. Tasasel pinnal saab vesi levida täisplokist maksimaalselt 7 plokki horisontaalselt. Allikaploki eemaldamisel vesi, mis levis eemaldatud allikaplokist ning ei saa levida ühestki teisest allikaplokist, kaob. Näiteks joonisel 4 kujutatud kosk kaoks, kui

[3] https://www.minecraft.net/

eemaldada allikaplokk, millest see algab. Allikaplokid asukohta ei muuda, kuid tühi plokk muutub allikaplokiks, kui selle kahel horisontaalsel küljel on allikaplokk.

1.3.2 Noita[4]

Noita on kahemõõtmeline kõrvaltvaates seiklusmäng. Mäng võtab aset protseduuriliselt genereeritud maailmas, kus iga piksel on füüsikaliselt simuleeritud. Eesmärk on laskuda sügavamale ja sügavamale maa alla, et tappa boss vastane. Mängija saab koguda võlukeppe, loitse ja nõiajooke, et eesmärki täita. Surma puhul tuleb järgmises maailmas uuesti proovida.



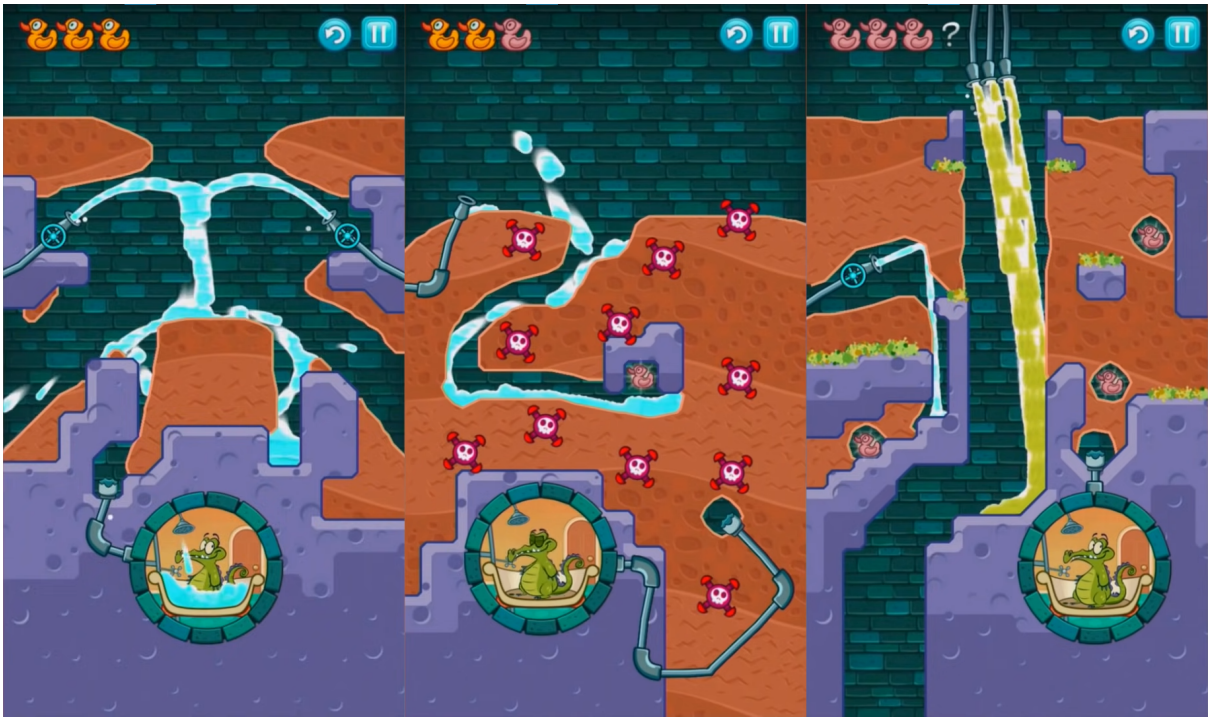
Joonis 5. Noita.

Kogu mängumaailm on rakuautomaadipõhine. Joonisel 5 on kujutatud kuidas vedelikud voolavad alla ja külgedele laiali. Mängus on kümneid erinevaid vedelike, mis võivad kombineerudes muudatusi põhjustada. Näiteks vee ja laava kokkusaamisel tekib kivi. Vedelikke saab pudelites kaasas kanda ning kasutada tule kustutamiseks. Igal vedelikul on kindel tihedus, mis lubab neil vajuda kihtidesse. Mängus on simuleeritud elekter, mis levib läbi vedelike ja metallide. Simuleeritud gaasid takistavad hingamist ja aurud kondenseeruvad tagasi vedelikeks.

[4] https://noitagame.com/

1.3.3 Where's My Water?[5]

Where's My Water on kahemõõtmeline kõrvaltvaates mobiilimäng. Mängu eesmärk on kaevata tee läbi mulla, et juhatada vesi alligaatori dušini. Kaevatud teedes saab liikuda puhas vesi, must vesi, mürgine vesi, aur ja löga.



Joonis 6. Ekraanitõmmised[6] mängust Where's My Water.

Joonisel 6 on näha mängus osakeste abil simuleeritud vedelikke. Osakeste liikumine on piiratud maastiku poolt. Liikumist mõjutab gravitatsioon, osakeste kiirus, ning lähedus teistele osakestele. Kui mitu vedelikuosakest on üksteisele piisavalt lähedal, sulandatakse nende välimus kokku, et anda ühtse vedelikumassi välimus.

1.3.4 New Super Mario Bros. U[7]

New Super Mario Bros. U on kahemõõtmeline kõrvaltvaates platvormimäng. Mängu eesmärk on vastaseid ning ohte vältides taseme lõpus leitava lipuni jõudmine.

[5] https://play.google.com/store/apps/details?id=com.disney.WMW
[6] TapGameplay YouTube'i kanal – Where's My Water? Free - Gameplay Walkthrough - All Levels (iOS, Android) https://www.youtube.com/watch?v=nkMdCkMLyf4
[7] https://www.nintendo.com/store/products/new-super-mario-bros-u-deluxe-switch/

Joonis 7. Ekraanitõmmis[8] mängust New Super Mario Bros. U.

Mängus on kasutusel võrestikul põhinev vesi. Vesi on kindlates tasemetes fikseeritud asukohal. Lisaks on kindlates asukohtades veepursked, mis perioodiliselt ilmuvad ja kaovad. Joonisel 7 on kujutatud veepinnale realistlikuma välimuse andmiseks rakendatud lainetamist. Joonisel on ka näha mängija tekitatud suuremad lained ning plärtsatuse efekt.

[8] IGN – 3 Star Coin Walkthrough - Sparkling Waters-1: Waterspout Beach https://www.ign.com/videos/2012/11/17/new-super-mario-bros-u-3-star-coin-walkthrough-sparkling-waters-1-waterspout-beach

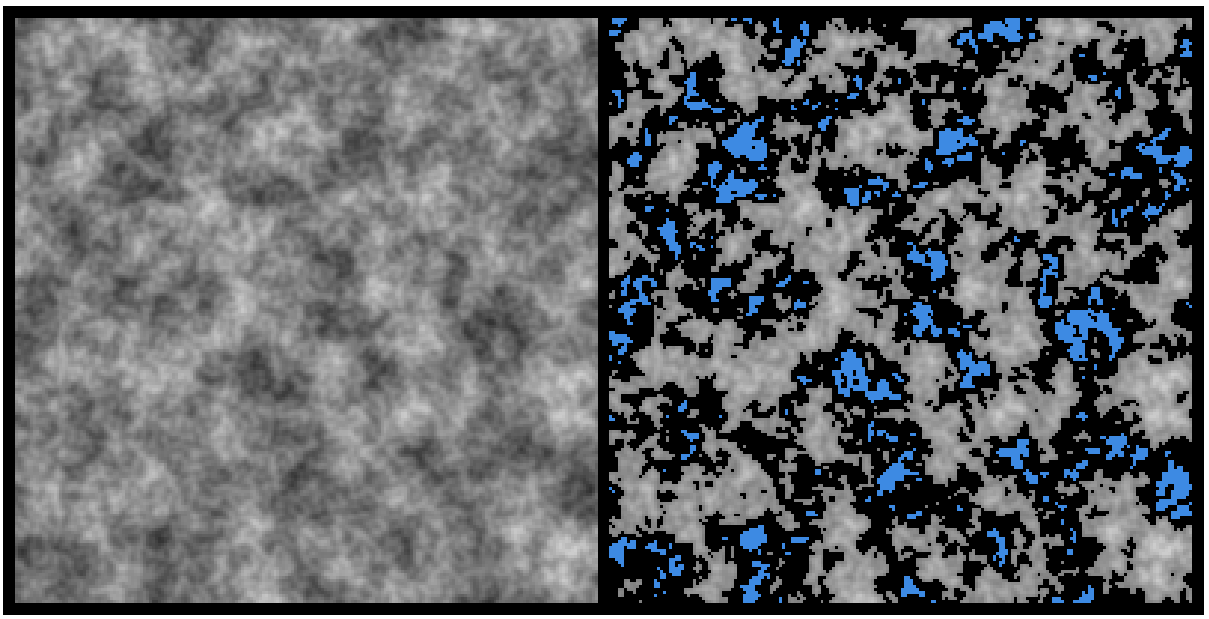
2. Implementeerimine

Kuna Blastronaudi maailm on protseduuriliselt genereeritud ja lõpmatu, ei ole võimalik vett mängumaailma käsitsi paigutada. Vee genereerimine peab toimuma ka protseduuriliselt. Selleks täiendati Blastronaudi olemasolevat maailma generaatorit selliselt, et tühjadesse kohtadesse genereeritakse vett ning simulatsiooni tulemusena valgub see ise maailmas laiali.

2.1 Vee generatsioon

Joonisel 8 on kujutatud Blastronaudi mängumaailma komponendid. Maailm koosneb tükkidest, mis koosnevad omakorda 8x8 plokkide ruudustikust. Mängumaailma maastik genereeritakse kasutades OpenSimplex müra. Müra on piiratud vahemikku $[-1, 1]$. OpenSimplex müra on kujutatud joonise 9 vasakul poolel, kus tumedamalt on tähistatud väiksemad väärtused ning heledamalt on tähistatud suuremad väärtused. Iga ploki genereerimisel määratakse OpenSimplex müra abil, millised plokid on tahked ning millised on tühjad. Kui mingil plokikoordinaadil on müra väärtus vähem kui kindel arv, siis on plokk tühi ja vastasel juhul on plokk tahke. Joonise 9 paremal poolel on tühjad plokid tähistatud mustalt ning tahked plokid hallides toonides.

Joonis 8. Maailma komponendid. Vasakus ruudus plokk. Paremas ruudus tükk. [11]



Joonis 9. Vasakul OpenSimplex müra. Paremal OpenSimplex müra, kus on tühi ruum ja vesi teise värviga kujutatud.

Vee plokid paigutatakse samamoodi müra väärtuste järgi, selliselt, et nad alati satuksid tühjadesse plokkidesse. Näiteks joonisel 9 on veeplokid tähistatud siniselt, paigutatult plokkidesse, kus müra väärtus oli väiksem kui -0,3. Genereeritud veeplokkide maailmakoordinaadid talletatakse sõnastikus võtmetena. Sõnastiku elementide väärtuseks on vee andmed. Igal 8 x 8 tükil on oma sõnastik selles sisalduva vee talletamiseks, see lihtsustab hiljem vee visualiseerimist ja praakimist.

2.2 Vee käitumine

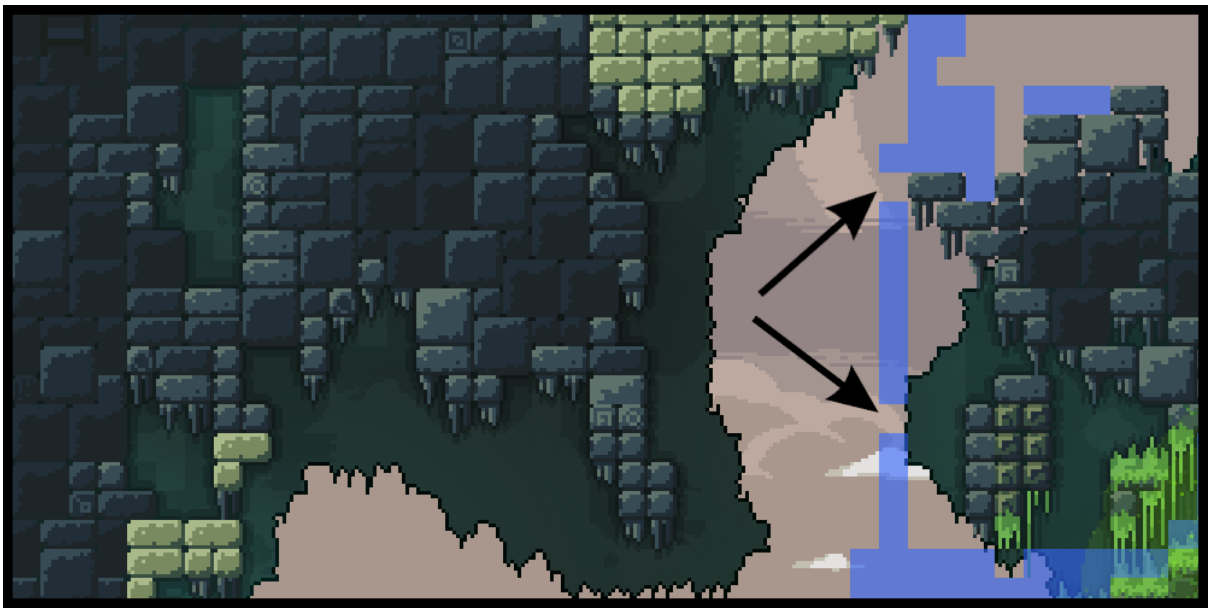
Vee käitumise koostamiseks tuleb valida üks eelnevalt kirjeldatud meetoditest. Võrestiku koostamine ja maailma paigutamine on välistatud, sest see ei tööta koos protseduurilise generatsiooniga. Nii rakuautomaati kui ka osakestel põhinevat süsteemi on võimalik rakendada piisavalt hea jõudlusega. Rakuautomaadil põhineva simulatsiooni eelis on selle ühtivus Blastronaudi rakkudest koosneva maailmaga. Osakestel põhinev süsteem annaks pisut realistlikuma käitumise. Nagu eelnevalt välja toodud, pole selliste mängude puhul vedelike realistlik käitumine oluline. Seetõttu koostatakse töö käigus rakuautomaadil põhinev süsteem.

2.2.1 Esimene iteratsioon - binaarne rakuautomaat

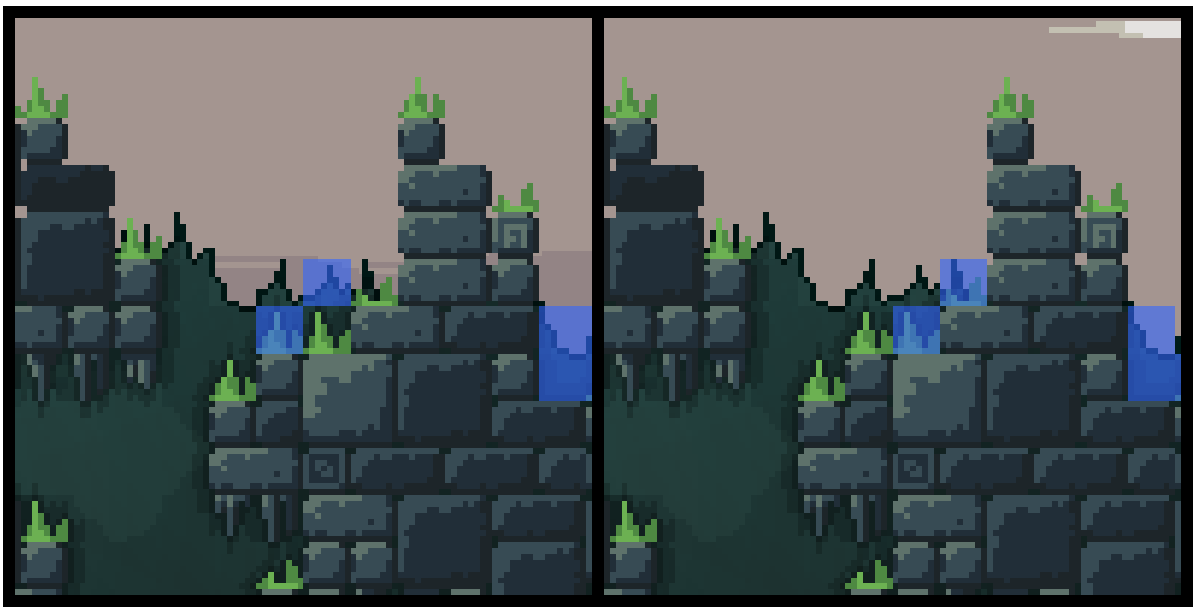
Esialgses versioonis kujutati vett alati täisplokina. Vee liigutamist tehakse kindla ajavahemiku järel. Iteratsioonide sageduseks valiti arbitraarselt 10 korda sekundis. Igal iteratsioonil liigutatakse järgemööda kõikides tükkides olevaid veeplokke. Ühe tüki siseselt liigutatakse vett järgmiste tingimuste põhjal:

1. Kui alumine plokk on tühi, siis liiguta vesi üks plokk alla.
2. Kui eelnev pole võimalik, aga parempoolne plokk on tühi, siis liiguta vesi üks plokk paremale.
3. Kui eelnev pole võimalik, aga vasakpoolne plokk on tühi, siis liiguta vesi üks plokk vasakule.

Liigutades veeplokki ühest tükist teise tuli veeandmed salvestada uue tüki sõnastikku ning eemaldada vanast sõnastikust. Juhul kui ülemise tüki veeplokke liigutati enne alumise tüki uuendamist, tekkis olukord, kus sama veeplokk võis ühe iteratsiooni jooksul liikuda kaks korda. See juhtus siis kui ülemise tüki veeplokk liigutati alumisse tükki ja seejärel alumise tüki uuendamisel liigutati veel korra. Käitumine on illustreeritud joonisel 10. Selle probleemi lahendamiseks määrati just liigutatud vee väärtuseks -1 ning takistati vee liigutamist kui selle väärtus on negatiivne. Vee väärtus muudeti tagasi positiivseks iteratsiooni lõpus vee ekraanile joonistamise ajal.



Joonis 10. Kaks korda liikunud veeplokid maailma tükkide piiridel.



Joonis 11. Võnkuvad veeplokid.

Antud loogika puhul võisid veeplokid sattuda võnkuvasse olekusse. Joonisel 11 on välja toodud kaheastmeline võnkumine. Esimesel iteratsioonil liigub alumine veeplokk paremale ning seejärel liigub ka ülemine veeplokk paremale. Teisel iteratsioonil liiguvad mõlemad veeplokid vasakule ning on samas olekus, milles nad olid enne esimest iteratsiooni. Taolisi võnkumiskäitumisi ilmnes paljudel erinevatel kujudel. Potentsiaalne lahendus sellisele käitumisele on suvalise liikumissuuna valimine. Seda lahendust prooviti kuid ei jäetud alles, sest planeeritud varieeruva tasemega vee puhul sellist käitumist ei esineks.

2.2.2 Teine iteratsioon - astmeline rakuautomaat

Teise iteratsiooni vesi oli varieeruva pinnatasemega. Vee väärtus oli talletatud arvuna vahemikus 1 kuni 8. Maksimaalseks arvuks valiti 8, sest see ühtis ülejäänud mängu visuaalse stiiliga, kus üks plokk on 8 pikslit pika küljega ruut. Iteratsioonide sagedus tõsteti 60 iteratsioonini sekundis.

Selle iteratsiooni vesi liikus järgneva loogika järgi:

1. Anna võimalikult palju oma vett alumisse plokki.
2. Anna ¼ oma vett paremasse plokki.
3. Anna ⅓ oma vett vasakusse plokki.

Alumisse plokki antud vee kogus sõltus praeguse veeploki vee kogusest ning alumise ploki tühjast ruumist. Edasi anti neist väiksema arvu jagu vett. Paremale ja vasakule antava vee

kogus erines, et ühtlustada vee valgumist mõlemale küljele. Näiteks veeplokk, mille väärtus on 8 annaks paremasse plokki oma 8-st ühikust veest 2 ühikut ning seejärel vasakusse plokki oma allesjäänud 6-st ühikust 2 ühikut vett. Lõpptulemuseks oleks võrdselt jaotunud vesi.

Gallant [12] leiab, et varieeruva pinnatasemega vesi näeks hea välja on vaja seda kukkumise ajal joonistada täisplokina. Selle saavutamiseks lisati just vett juurde saanud ploki vee väärtusele arv 0,1. Vee joonistamise käigus joonistati selliseid plokke täisplokina sõltumata selle vee väärtusest ning murdosa eemaldati põrandafunktsiooni abil.

Joonis 12. Vee tasapind.

Sellise süsteemi rakendamisel kadus võnkuv käitumine kuid ilmnes joonisel 12 nähtav käitumine, kus vee tasapind ei ühtlustu täielikult. Probleemi lahendamiseks prooviti suurendada maksimaalset vee väärtust 80-ni ning kehtestada reegel, et veeplokk kustutatakse kui selle väärtus langeb alla kolme. Rakendatud meetmed leevendasid probleemi kuid ei eemaldanud seda.

2.2.3 Kolmas iteratsioon - ujukomaarvuline rakuautomaat

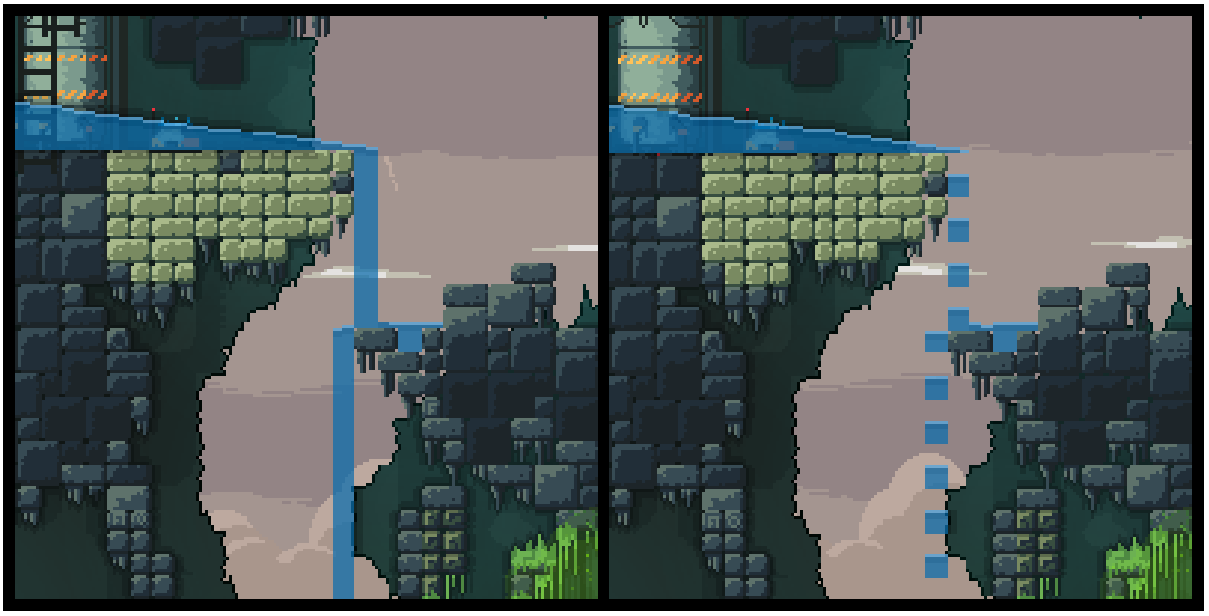
Kolmas ja viimane iteratsioon vee loogikast on samuti varieeruva veepinna tasemega. Vee väärtust talletatakse ujukomaarvuna vahemikus (0, 1]. Kuna vee väärtuse murdosa ei saa enam kukkumise jälgimiseks kasutada on veeploki väärtus nüüd järjend, mille esimene element on vee kogus, teine element on tõeväärtus, mis jälgib kukkumist ning kolmas on viimati plokki horisontaalselt liigutatud vee kogus. Neist viimast kasutatakse hiljem, et anda

voolava vee pinnale kaldus välimus. Vee genereerimise müratingimus langetati arvuni -0,4, et vähendada maailmas ilmuvat vett. See tagab parema jõudluse ning vähendab vee negatiivset mõju mängija kogemusele.

Selle iteratsiooni vesi järgib järgmist loogikat:

1. Anna 99% liigutatavast veest alumisse plokki.
2. Anna ½ oma vett paremasse plokki.
3. Anna ½ oma vett vasakusse plokki.

Joonis 13 illustreerib, kuidas alla voolamise 99% peale piiramine aitab anda kukkuvale veele ühtse joa välimuse. Paremale ja vasakule antava vee koguse erinevus eemaldati, kuna võrdne jaotus andis visuaalselt parema välimuse. Sellise vee liikumise puhul võis tekkida olukord, kus lapikule pinnale jäi õhuke kiht vett kinni. Käitumise eemaldamiseks kontrollitakse iga veeploki liigutamise alguses selle vee kogust. Juhul kui vee kogus on < 0,01, veeplokk kustutatakse.



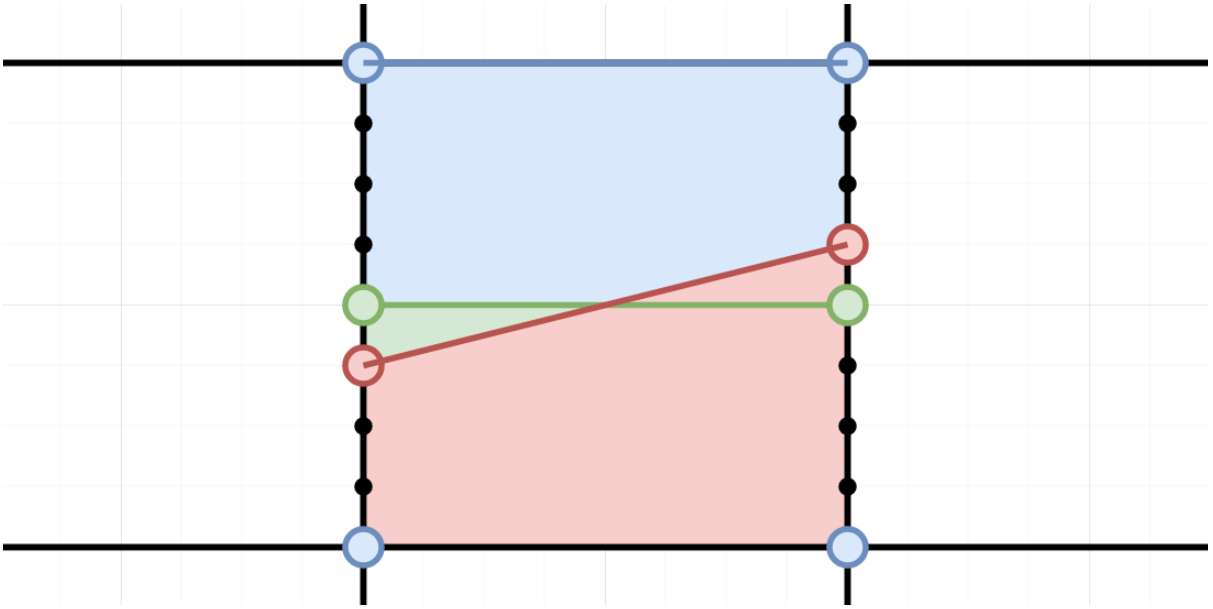
Joonis 13. Vasakul on 99% piiratud kukkumine. Paremal on piiramata kukkumine.

Kuna Blastronut annab mängijale võime mängumaailma hävitada ja muuta, peab vee liikumine sellega arvestama. Genereeritud plokid salvestatakse sõnastikku sarnaselt veeplokkidele. Vee liikumise loogika kontrollib lähedal olevaid plokke kasutades seda sõnastikku. Iga ploki hävitamise korral tuleb plokkide sõnastikku uuendada. Osade plokkide hävitamisel võib tekkida killustikuplokk, mis kukub otse alla. Ka selle ploki tekkimise korral

peab uuendama plokkide sõnastikku. On võimalik, et selline killustikuplokk kukub vee sisse. Selleks juhuks on veeloogikal lisatingimus, mis kustutab veeploki kui sellega samas asukohas on tahke plokk.

2.3 Visualiseerimine

Vee võrestik luuakse Godot ArrayMesh struktuuri abil. Esmalt leitakse veeploki nelja nurga koordinaadid maailmas. Need nurgad on joonisel 14 kujutatud sinisena. Kui vesi on kukkuvas olekus joonistatakse täisplokk ehk nurkade koordinaadid jäävad samaks. Kui vesi ei ole kukkuvas olekus kohandatakse selle ülemisi nurkasid, et anda sellele sobiv välimus. Selleks liigutatakse nurgad allapoole sõltuvalt plokis oleva vee kogusest. Näiteks kui ploki vee kogus on 0,5 liigutatakse ülemised nurgad nelja piksli jagu allapoole, et need jääksid vertikaalselt ploki keskele. Seda on tähistatud joonisel 14 roheliselt.



Joonis 14. Veepinna moodustamist illustreeriv diagramm.

Selleks, et anda voolavale veele kaldus välimus, liigutatakse ploki ühte ülemist nurka kõrgemale ning teist allapoole. Näiteks kui viimase iteratsiooni jooksul oli sisse voolanud vee koguse väärtus on -0,25, siis liigutatakse vasakut nurka ühe piksli jagu allapoole ning paremat nurka ühe piksli jagu ülespoole. Seda on joonisel 14 tähistatud punasena.

Kui ploki nurkade koordinaadid on välja arvutatud, siis lisatakse need kahe kolmnurgana võrestikku. Seda protsessi korratakse kõigi tüki sees olevate plokkidega ja koondatakse need

üheks võrestikuks. Võrestik määratakse tüki andmestikus talletatud MeshInstance2D objektile, et seda oleks mängus näha.

Joonis 15. Veele rakendatud varjutaja.

Veele parema välimuse andmiseks paigutatakse kõikide aktiivsete tükkide MeshInstance2D objektid esmalt eraldi Viewport objektile. See võimaldab kogu ekraanil olevale veele rakendada ühtset varjutajat (ing. k. *shader*). Ülejäänud mänguga ühtse visuaalse stiili saamiseks rakendatakse veele varjutajat, mis kasutab sama valgusarvutuse mudelit, mida on rakendatud ka ülejäänud plokkidel. Joonisel 13 on näha kuidas vee ülemine pind asendatakse heledama värviga, samuti varjutatakse selle all olev vesi tumedamaks. Ka taustasein vee lähedal värvitakse pisut tumedamaks, et tekitada kaudvarju (ing. k. *ambient occlusion*) meenutav efekt. Kaudvarju on näha joonisel 13 veepinna lähedal. Joonisel 15 on illustreeritud kuidas heleda valguse olemasolu korral värvitakse lähedal olevate plokkide lähim külg vastavat värvi. Vee puhul on see efekt piiratud ülemisele pinnale.

2.4 Interaktsioonid mängijaga

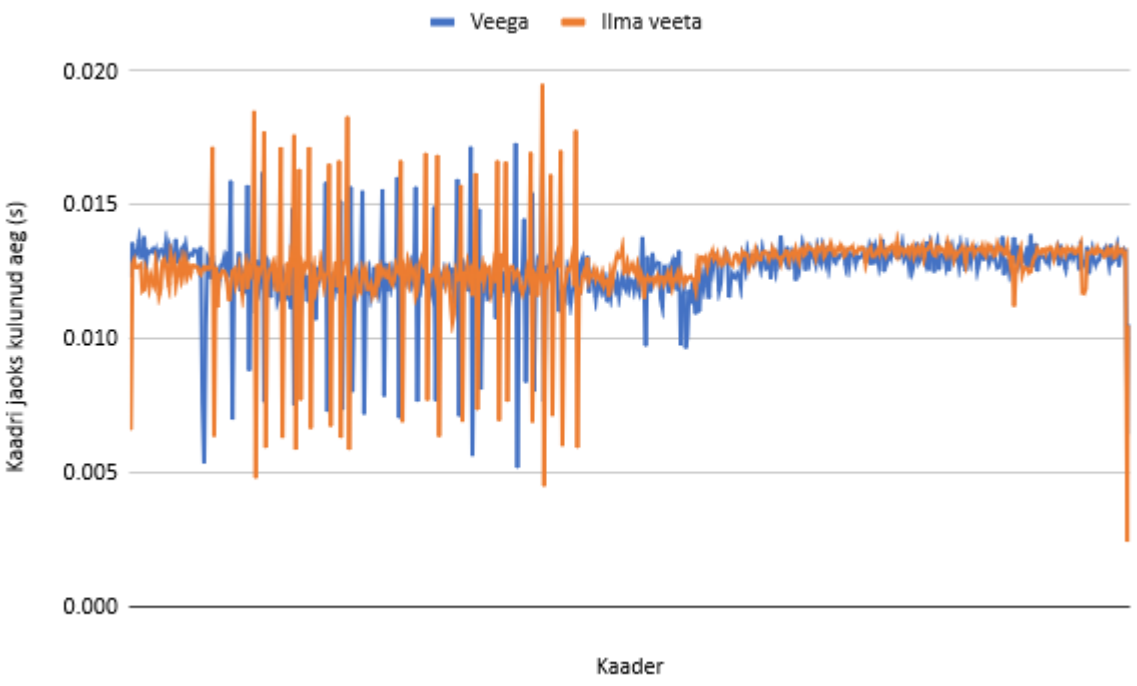
Vee realismi paremaks tegemiseks lisatakse heli mängija interaktsioonidele. Selleks jälgitakse, kas mängija asukohal on vesi. Mängija liikumisel kontrollitakse, kas tema asukohas on vesi. Mängija vette liikumisel mängitakse heli. Kuniks mängija on vee all on tema horisontaalse liikumise kiirus ning talle mõjuva gravitatsiooni tugevus tavalisest poole väiksemad. Sedasi saab mängija hüpata kõrgemale ning vajub aeglasemalt.

3. Testimine

Selles peatükis kirjeldatakse jõudlustestimist ning kasutajate tagasisidet. Alapeatükis 3.1 kontrollitakse sissejuhatuses püstitatud 2. eesmärgi täitmist. Alapeatükis 3.2 kontrollitakse sissejuhatuses püstitatud 3. eesmärgi täitmist.

3.1 Jõudlustestimine

Jõudlustestimine viidi läbi kasutades Godot sisseehitatud profileerijat. Testiti masinal, millel oli NVIDIA GeForce - GTX 1070 videokaart ning Intel® Core™ i5-8400 protsessor. Godot dokumentatsiooni [13] sõnul on profileerija jõudlusmahukas. Kuna profileerija on ainult saadaval Godot redaktoris tuli mängu testida kompileerimata kujul. See tähendab, et mängu jõudlus testimise ajal oli madalam kui normaalse kasutuse ajal. Testimist sooritati 60 Hz ja 144 Hz kaadrisagedusega monitoridega.

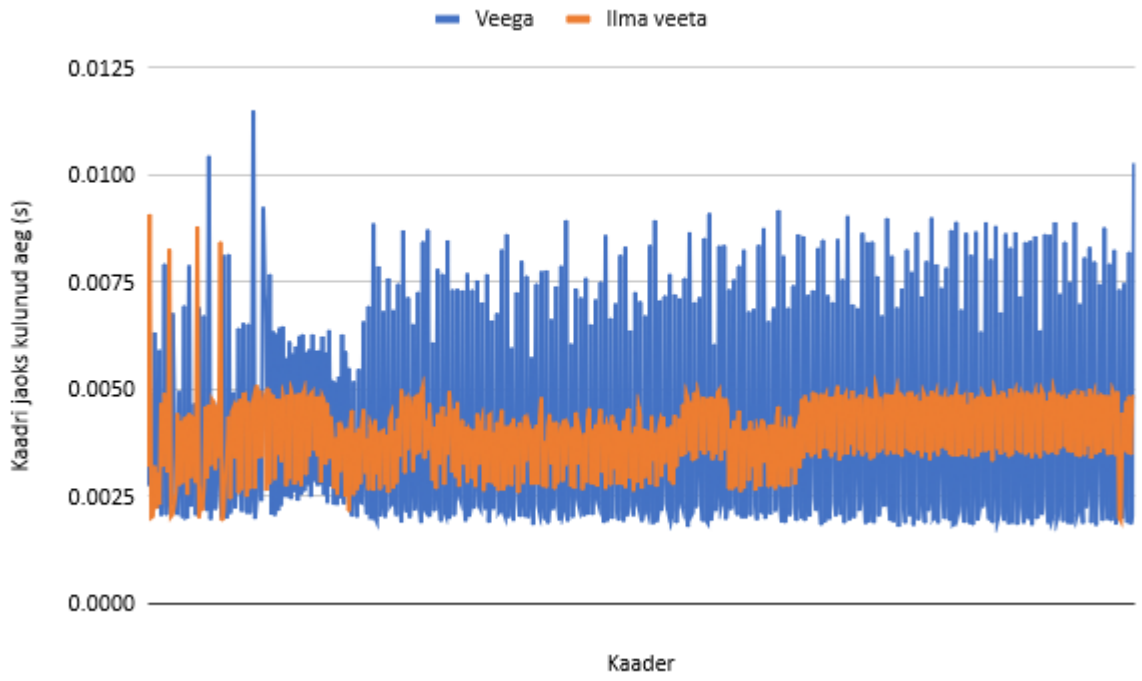


Joonis 16. Testimise tulemused 60 Hz monitoriga.

Joonisel 16 on toodud diagramm, mis võrdleb mängu kaadrite visualiseerimiseks kulunud aega 60 Hz monitoril. Ilma veeta mängu kaadrite visualiseerimiseks kulus keskmiselt 12,64

ms. Mediaanaeg oli 12,76 ms ning standardhälve oli 1,64 ms. Koos veega oli keskmine kaadriaeg 12,5 ms, mediaankaadriaeg 12,72 ms ning standardhälve 1,34 ms.

Joonisel 17 on toodud diagramm, mis võrdleb mängu kaadrite visualiseerimiseks kulunud aega 144 Hz monitoril. Testides 144 Hz monitoriga oli ilma veeta keskmine kaadriaeg 3,94 ms, mediaankaadriaeg 4,06 ms ning standardhälve 0,91 ms. Koos veega oli keskmine kaadriaeg 4,07 ms, mediaankaadriaeg 2,44 ms ning standardhälve 2,56 ms.



Joonis 17. Testimise tulemused 144 Hz monitoriga.

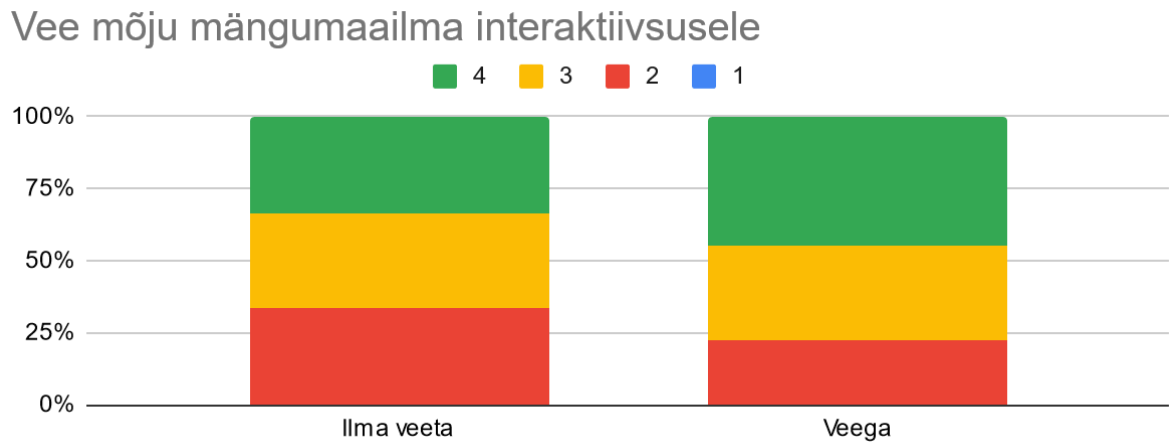
Tulemuste põhjal võib väita, et vee implementatsioon ei mõjuta märkimisväärselt mängu jõudlust ning teine eesmärk on täidetud. Kõrgematel kaadrisagedustel on märgatavalt suurem variatsioon kaadrite visualiseerimiseks kuluval ajal. Seda põhjustab vee iteratsioonisagedus. Vett uuendatakse 60 korda sekundis. See tähendab, et vee uuendamine toimub igal teisel või kolmandal kaadril.

3.2 Kasutajatestimine

Töö käigus viidi läbi kasutajatestimine, eesmärgiga hinnata vee mõju mängu interaktiivsusele, vee visuaalset sobimist ülejäänud mänguga ja vee füüsikalise käitumise headust. Testimine sooritati Google Forms platvormiga. Küsimustik koostati inglise keeles, et

laiendada potentsiaalsete kasutajate arvu. Küsimustik ning testitavad mänguversioonid on toodud lisas II.

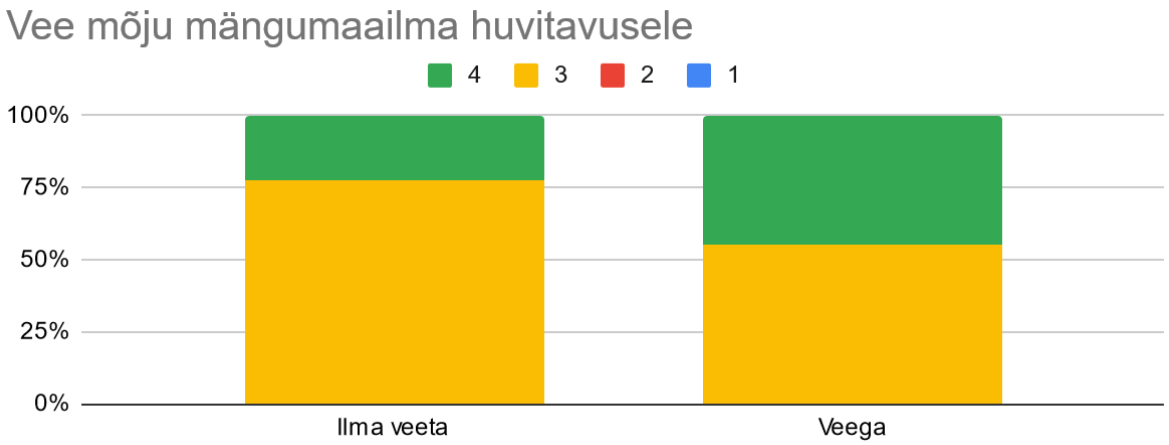
Küsimustikule vastas 9 inimest. Kasutajatelt paluti mängida kahte versiooni mängust. Esimene versioon oli ilma veeta. Teine versioon oli kõige uuema vee versiooniga. Peale mingi versiooni katsetamist paluti kasutajal hinnata 4-punkti skaalal mängumaailma interaktiivsust ja huvitavust. Hinne 1 tähendab vastavalt "väga ebainteraktiivne" ja "pole üldse huvitav". Hinne 4 tähendab "väga interaktiivne" või "väga huvitav". Lõpus paluti kasutajal hinnata vee ühtivust ülejäänud mängu visuaalse stiiliga ning vee füüsikalise käitumise headust. Hinne 1 tähendab vastavalt "ei ühti üldse" ja "väga halb" ning hinne 4 tähendab vastavalt "ühtib täiuslikult" ja "suurepärane".



Joonis 18. Vee mõju mängumaailma interaktiivsusele.

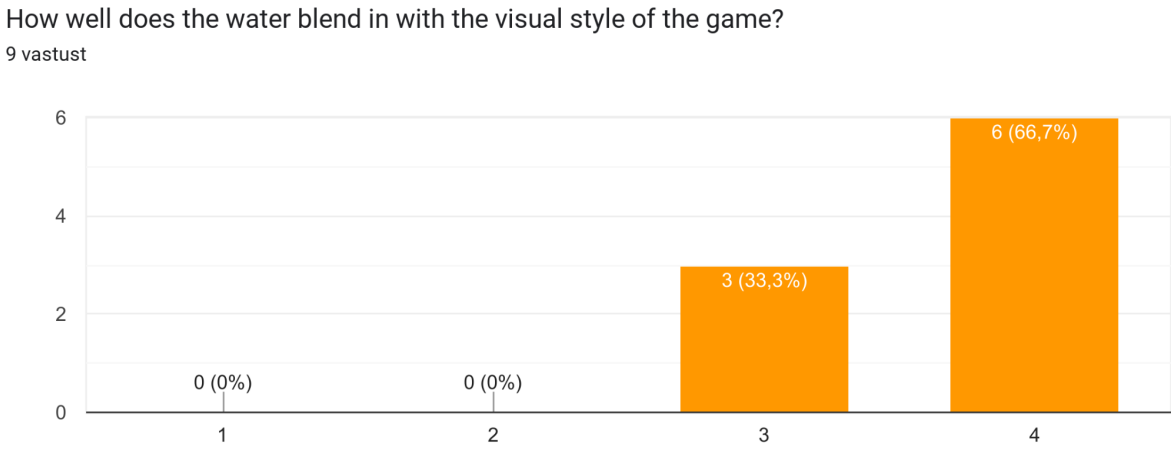
Joonisel 18 on näha kuidas vesi mõjutas kasutajate taju mängumaailma interaktiivsusest. Üheksa vastaja seast leidsid kaks vastajat, et veega versiooni mängumaailma interaktiivsus kasvas. Ülejäänud seitse vastajat leidsid, et mõlema versiooni interaktiivsus oli sama.

Joonis 19 illustreerib kuidas vee lisamine mõjutas kasutajate meelest mängumaailma huvitavust. Üheksa vastaja seast arvasid kaks, et veega versioon mängust oli huvitavama mängumaailmaga. Ülejäänud seitsme vastaja meelest oli maailma huvitavus mõlema versiooni puhul sama.

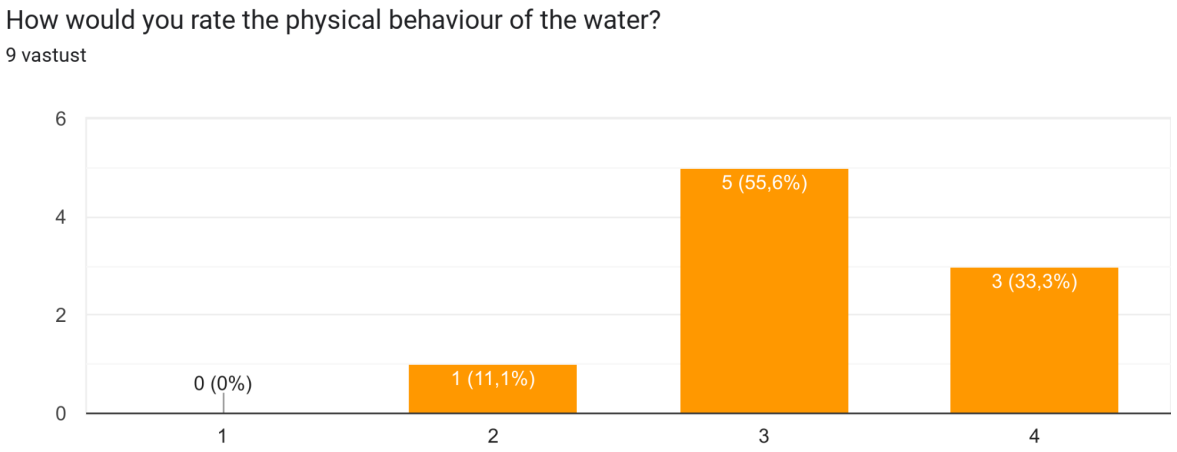


Joonis 19. Vee mõju mängumaailma huvitavusele.

Joonisel 20 on kujutatud kasutajate hinnangud vee ülejäänud mängu visuaalse stiiliga ühtimise kohta. Üheksast vastajast kuus arvasid, et vesi ühtib ülejäänud mängu visuaalse stiiliga täiuslikult. Joonisel 21 on näha kasutajate hinnangud vee füüsikalisele käitumisele. Viis kasutajat andsid hinde 3 ning ainult üks andis alla keskmise hinde 2.



Joonis 20. Vee ülejäänud mängu visuaalse stiiliga ühtimise hinnangud.



Joonis 21. Vee füüsikalise käitumise hinnangud.

Kasutajatestimise põhjal võib järeldada, et vee mõju mängumaailma interaktiivsusele ja huvitavusele ei ole märkimisväärne. Samas pole põhjust arvata, et vee simulatsioon avaldab negatiivset mõju. Hinnangute põhjal võib kindlalt väita, et vesi ühtib ülejäänud mängu visuaalse stiiliga väga hästi. Järelikult kolmas eesmärk on täidetud. Vee füüsikaline käitumine on hinnangute järgi hea. Üks kasutajatest andis tagasisidet, et vee liikumine tundus liiga kiire.

3.3 Edasiarenduse võimalused

Vee simulatsiooni täiustamiseks on mitu võimalust. Esiteks saab muuta simulatsiooni käitumist muutes väärtuseid nagu iteratsioonide sagedus, genereeritava vee kogus ja voolukiirus. On võimalik, et leiduvad sellised väärtused, mille korral käitub simulatsioon realistlikumalt. Teiseks saab lisada uusi vedelikke nagu õli või laava, millel on teistsugused omadused ja seega voolaksid nad teistmoodi. Kolmandaks saab lisada uusi süsteeme, näiteks peatükis 1.1.1 kirjeldatud veerõhu simulatsioon või põhjalikumad mängija interaktsioonid veega. Neljandaks saab täiendada vee välimust, näiteks lisades sellele erinevaid efekte nagu lained ja peegeldused.

Kokkuvõte

Bakalaureusetöös tutvustati vee simuleerimise viise mängudes ning toodi näiteid populaarsetes mängudes kasutusel olevatest meetoditest. Töö eesmärgiks oli Blastronaut mängule luua vee simulatsioon, mis töötaks mängu olemasoleva protseduurilise generatsiooniga, reageeriks maailma muudatustele reaalajas ja ühtiks ülejäänud mängu visuaalse stiiliga.

Loodud vee simulatsiooni meetodiks valiti rakuautomaadil põhinev süsteem. Simulatsiooni loomiseks tehti mitu iteratsiooni, kus kõigepealt simuleeriti vett täisplokkide kaupa ning seejärel plokis sisalduva vee koguse järgi. Lisaks muudeti veepind sujuvamaks muutes selle kallakut siseneva vee koguse põhjal.

Vee simulatsioon integreeriti mängu protseduurilise generatsiooniga. Ülejäänud eesmärkide täitmise hindamiseks viidi läbi kasutajatestimine ja jõudlustestimine. Kasutajatestimise käigus jõuti järelduseni, et vesi ei avalda mängumaailma interaktiivsusele ja huvitavusele negatiivset mõju, vesi ühtib ülejäänud mängu visuaalse stiiliga väga hästi ja vee füüsikaline käitumine on hinnangute järgi hea kuid sellel on arenguruumi. Jõudlustestimise abil leiti, et vee simulatsiooni mõju mängu jõudlusele on minimaalne ning sobib reaalajas kasutamiseks. Kõik töö eesmärgid said täidetud. Lisaks pakuti välja erinevaid lahendusi tulevikus simulatsiooni käitumise paremaks muutmiseks.

Viidatud kirjandus

[1] Klimmt C., Hartmann T., Frey A. Effectance and control as determinants of video game enjoyment. *Cyberpsychology & behavior*. Dec 2007, Volume 10, Issue 6, 845-848. DOI: http://doi.org/10.1089/cpb.2007.9942

[2] Tanskanen S. Player immersion in video games: Designing an immersive game project. South-Eastern Finland University of Applied Sciences. Bachelor's thesis. 2018. https://www.theseus.fi/bitstream/handle/10024/147016/tanskanen_selja.pdf

[3] Kellomäki T. Fast Water Simulation Methods for Games. ACM Comput. Computers in Entertainment Volume 16, Issue 1, Article 2, December 2017, 14 pages. DOI: https://doi.org/10.1145/2700533

[4] Amada T. Cellular Automata for Physical Modelling. *Game Programming Gems 6*. Boston, Massachusetts: Charles River Media, 2002, 189-205.

[5] Braley C., & Sandu A. Fluid Simulation For Computer Graphics: A Tutorial in Grid Based and Particle Based Methods. 2009. https://cg.informatik.uni-freiburg.de/intern/seminar/gridFluids_fluid-EulerParticle.pdf

[6] Forsyth T. Cellular Automata for Physical Modelling. *Game Programming Gems 3*. Hingham, Massachusetts: Charles River Media, 2002, 200-214.

[7] Gardner M. Mathematical Games: The fantastic combinations of John Conway's new solitaire game "life". *Scientific American*. October, 1970. Vol. 223, no. 4. pp. 120-123. DOI: https://doi.org/10.1038/scientificamerican1070-120

[8] Janno M. Procedural Generation of 2D Creatures. University of Tartu Institute of Computer Science. Bachelor's Thesis. 2018.

[9] Hagström R. Frames That Matter: The Importance of Frames per Second in Games. Uppsala Universitet, Department of Game Design. Bachelor's Thesis in Game Design. 2015. http://uu.diva-portal.org/smash/record.jsf?dswid=3817

[10] Lundell, C. Water simulation for cell based sandbox games. Linköpings universitet, Department of Electrical Engineering, Thesis 2014. http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-108828

[11] Spitsõn S. Mallide lisamine Blastronaut mängu protseduurilisele generaatorile. Tartu Ülikooli arvutiteaduse instituudi bakalaureusetöö. 2015. https://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=72121

[12] Gallant J. 2D Liquid Simulator With Cellular Automaton in Unity. January 30th, 2017. http://www.jgallant.com/2d-liquid-simulator-with-cellular-automaton-in-unity/ (08.05.2022)

[13] Godot Docs https://docs.godotengine.org/en/latest/tutorials/scripting/debug/the_profiler.html. (08.05.2022)

Lisad

I. Programmi lähtekood

Programmi lähtekood asub repositooriumis aadressil https://gitlab.com/perfoon/blastronaut. Bakalaureusetöö käigus tehtud töö on leitav water_simulation harus. Ligipääsu saamiseks võtke ühendust meiliga arlo.tammekun@ut.ee.

II. Küsitlus ja testitavad mänguversioonid

Tööle lisatud failis on küsimustik nimega “Blastronaut_Water_Assessment.pdf”, vastused küsimustikule failis nimega “Blastronaut_Water_Assessment.csv” ja kaust nimega “Blastronaut versions”, mis sisaldab küsitluse käigus kasutatud mängu versioone.

III. Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Arlo Tammekun

1. Annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose Vee simulatsiooni loomine Blastronaut mängule, mille juhendaja on Jaanus Jaggo 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 4.0, 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.

Arlo Tammekun

10.05.2022