TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Arvutiteaduse instituut

Mitmikmängu „Snake“ juhtimine brauseripõhiselt

Bakalaureusetöö

Üliõpilane: Oliver Savi
Üliõpilaskood: 120717 IAPB
Juhendaja: Jaagup Irve

Tallinn
2016

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.

(*kuupäev*) (*allkiri*)

Annotatsioon

Bakalaureusetöö eesmärgiks on uurida *WebSocket* tehnoloogia valmisolekut ja sobivust veebipõhise mitmikmängu loomiseks. Töö käigus valmib mäng, mida on võimalik juhtida võrguühendust kasutades läbi kasutaja nutiseadme või arvuti. Mängu põhjal teeb autor järeldusi, kas valitud tehnoloogia oli sobilik ja hindab selle kohasust rakenduse arendamisel.

Mängu valmistamise vajadus tugineb antud valdkonnas vähese näidete olemasolule. Seoses massilise nutiseadmete levikuga on vajalik luua ka sellest lähtuvaid uutele põhimõtetele toetuvaid rakendusi. Töös on katsetatud ühte võimaliku uut suunda, kuidas veebipõhist mängu on võimalik juhtida. Lõputöös vaadeldakse erinevate juhtpuldi kasutajaliideste eripärasid ja sellest lähtuvalt valitakse mängu jaoks sobivaim. Lisaks on kirjeldatud terve töökäik alustades mängu planeerimisest kuni realiseerimiseni.

Tulemuseks valmis veebipõhine mäng, mis kasutab „Snake“ mängust tulenevat põhimõtet. Rakenduse kasutamiseks ei ole vaja lisatarkvara installeerida ja kõik toimib veebibrauseris. Juhtpuldi erinevate kasutajaliideste analüüsi tulemusena valis autor välja ekraanil viipamise meetodi (*swipe*), mis osutus kõige kasutajasõbralikumaks. Arenduskäik toimus vastavalt planeeritud etappidele ja implementeeritud said kõik eelkirjeldatud nõuded. Lõpptulemuseks pidas autor mängu õnnestunuks, sest WebSocket tehnoloogia sobis kokku valitud rakendusega ja uudne mängu juhtimisviis tõestas enda atraktiivust testkasutajate hulgas.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 41 leheküljel, 6 peatükki, 11 joonist.

Abstract

The purpose of the thesis is to research WebSocket technology and its suitability for creating web-based multiplayer games. The outcome of the work is a game that can be controlled over network by other smart devices or computers. In the end, the author will decide whether the chosen technology was appropriate and assesses the relevance of the topic.

Need for the game is based on the lack of written applications on that topic. With the mass rollout of smart devices it is necessary to build new type of games based on the new technologies. The game uses one new possible way to control a web-based game. The thesis examines various remote control user interfaces and chooses the best one for the project based on the results. In addition, the entire workflow is described starting from the planning to the implementation of the game.

The end result is a web-based game that uses classical “snake” game principles. There is no need to install any extra software to play the game as it all works in a web browser. During the analysis of different gamepad user interfaces the author chose swipe technique, which provides the best user experience. The development process took place according to the earlier planned stages and all the defined requirements were implemented. The author of the game considered the game a success, because the technology chosen for the project suited well with the written application and a different way to control the game proved its attractiveness among the users.

The thesis is written in Estonian and contains 41 pages of text, 6 chapters, 11 figures.

Lühendite ja mõistete sõnastik

HTML	***HyperText Markup Language*** Keel, milles märgendatakse veebilehti
TCP	***Transmission Control Protocol*** Levinuim transpordikihi võrguprotokoll
API	***Application programming interface*** Määratud reeglistik, mille alusel rakendusprogramm kasutab mõnda operatsioonisüsteemi või programmi teenuseid
MVC	***Model-view-controller*** Tarkvaraarenduse arhitektuuri liik
GIT	***Git*** Versioonihaldus tarkvara
WebSocket	***WebSocket*** Protokoll, mis lubab kahe poolset ühendust seadmete vahel üle ühe TCP ühenduse
QR-kood	***Quick Responsive Code*** QR-kood ehk ruutkood on Jaapanis loodud kodeeritud kahemõõtmeline maatrikskood, mis võimaldab skaneerida infot mobiiltelefoni, kus mobiilirakendus selle dekodeerib [1]
UX	***User Experience*** UX ehk kasutajakogemus on emotsioonid, mida inimene kogeb toote, süsteemi või teenuse kasutamisel [2]

Jooniste nimekiri

Sisukord

1. Sissejuhatus

Videomängu žanre ja loomisviise on maailmas palju ja igaühel neist on oma tava kuidas teistest erineda ning seeläbi ennast mängijate jaoks atraktiivseks muuta. Iga loodud mäng proovib endas kujutada midagi uut ja teistsugust, kuid vähesed suudavad luua innovatsiooni. Tehnoloogia kiire areng on kaasa toonud endaga mitmeid uusi võimalusi, mis lubavad luua senisest erinäolisemaid ja kasutajaid veelgi rohkem paeluma panevaid mänge.

Käesoleva bakalaureusetöö ideeks on luua veebipõhine mäng, mis erineb teistest turul olevatest mängudest tunduvalt oma juhtimisviisi ja lihtsuse poolest. Lisaks uuritakse töö käigus erinevate puuteekraanide juhtpuldi loomise loogikaid ja WebSocket tehnoloogia kasutust. See teema on võrdlemisi uus ja seega pole selle kohta eelnevalt palju kirjutatud ja teemakohaseid rakendusi loodud.

Valdav osa inimesi Eestis omab mõnda nutiseadet, milleks on näiteks mobiiltelefon või tahvelarvuti. Töös valmiva mängu üheks keskseks osaks ongi vajalik nutiseadme olemasolu. Projekti sihiks on luua ussi (*snake*) mitmikmäng, mida saab juhtida üle võrguühenduse läbi oma nutiseadme või teise arvutiga. Eriliseks teeb selle kõik see, et enne mängima asumist pole vaja midagi lisaks alla laadida ja mänguga ühinemine toimub hetkega. Põnevust lisab asjaolu, et mängida saab samaaegselt koos mitme teise kasutajaga.

Lõputöö jaotatud erinevateks loogilisteks jaotusteks, millest igaüks seletab lahti töökäigus toimunud protsessi või probleemi. Töö raames valmis mäng „SnakeWar“, mille põhjal on autori bakalaureusetöö kirjutatud.

Antud mäng on tehtud Tallinna Tehnikaülikooli informaatika eriala bakalaureuse õppeastme lõputööna. Bakalaureusetöö esitatakse Arvutiteaduse instituuti ja on valminud 2016 aastal Tallinnas.

1.1 Taust ja probleem

WebSocket on alles 2011 aastal standardiseeritud protokoll [3] ja see teeb sellest võrdlemisi uue tehnoloogia. WebSocket lubab lihtsamini ja kiiremini dünaamilisi võrgurakendusi programmeerida.

Antud bakalaureusetöö on vajalik, sest sel teemal pole enne palju sarnast praktikat kasutades rakendusi loodud. Seetõttu pakkus teema autorile huvi ja tema isiklik ambitsioon oli valmistada midagi sama tehnikat kasutades.

1.2 Ülesande püstitus

Bakalaureuse lõputöö eesmärk on luua mitmikmäng (*multiplayer game*), mida on võimalik juhtida üle võrguühenduse arvuti, mobiili või tahvelarvutiga. Mäng peab olema veebipõhine, mis tähendab seda, et nii *host* kui ka juhtpuldiga mängiv kasutaja ei pea endale midagi lisaks alla laadima. Piisab õige veebilehe avamisest ja kõik ülejäänud töö tehakse ära veebibrauseri poolt. Üheks tähtsaks osaks on puuteekraani kasutajaliidese loomine, mis vajab põhjaliku analüüsi erinevate viiside vahel. Mängu valmimisel saab teha järeldusi, kas valitud mäng on sellise tehnoloogia jaoks liiga kiireloomuline või on see piisavalt valmis, et veelgi keerukaimaid mänge sarnaselt implementeerida.

1.3 Metoodika

Eesmärkideni jõudmiseks tuleb autoril vaadelda juba samas valdkonnas tehtud töid ja õppida tundma erinevate lisateekide kasutamist. Enne programmeerima asumist on vaja töö jagada kindlatesse etappidesse ning teha selgeks, mis järjekorras need kõige arukam lahendada on. Olles omandanud vastavad teadmised ja teinud valmis plaani, tuleb autoril mäng valmis programmeerida. Arendamise käigus tuleb kasutada erinevaid lisakomponente, mis abistavad koodi kirjutamisel.

Üheks kõige tähtsamaks koodi osaks on WebSocket protokolli kasutamine, mille teeb lihtsamaks NodeJS-l põhinev raamistik Socket.io. See võimaldab luua ühenduse *host* masina ja juhtpuldi seadme vahel nii, et nad oleksid pidevalt omavahel suhtluses.

Veebirakendus, mida jooksutab *host* seadme brauser, on programmeeritud enamasti *JavaScript* programmeerimiskeeles. Rakenduse visuaalset poolt aitab lahendada HTML5 Canvas, mis on madala ressursinõudlusega ja samas võimas viis *JavaScripti* põhiselt mängu graafikat kuvada.

1.4 Ülevaade tööst

Töö annab ülevaate, kuidas toimib mitmikmäng „SnakeWar“ ja kuidas see on valminud. Selgitatakse lähemalt kasutatud tehnoloogiate otstarvet ja tööpõhimõtteid. Lisaks analüüsitakse lähemalt erinevate juhtpuldi disaini meetodite funktsionaalsust ja tehakse selle põhjal valik, millist liidest antud mängus kasutada. Järgnevalt võib leida kõigi peatükkide lühitutvustused.

2. Ülevaade mängust

Peatükis räägitakse mängu üldistest põhimõtetest ja reeglitest. Lisaks antakse ülevaade juba selles valdkonnas tehtud tööde kohta ja tuuakse olemasolevatest lahendustest mõned näited.

3. Kasutatud tehnoloogiad

Mainitud osas tutvustatakse lähemalt erinevaid tehnoloogiaid, mida mängu arendamisel kasutati. Täiendavalt on ära selgitatud nende töötamise põhimõtted ning põhjendatud valikut neid mängu valmistamisel kasutada.

4. Juhtpuldi kasutajaliidese arendus

Juhtpuldi kasutajaliidese arenduse peatükis keskendutakse erinevate juhtimisviiside kirjeldusele ja sellest põhinevalt nende analüüsile. Autor põhjendab, miks on lõputöös just kasutatud sellist juhtpuldi liidest ning kuidas ta selle otsuseni jõudis.

5. Mängu „SnakeWar“ loomine

Antud peatükis vaadeldakse mängu arendusprotsessi algusest lõpuni. Tuuakse välja ka mõni lihtsam koodinäide, mis annab lugejale ülevaate, kuidas toimib ühendus *host* seadme ja juhtpuldi vahel. Lõpetuseks analüüsitakse valminud mängu.

2. Ülevaade mängust

Järgnevalt luuakse ülevaade mängu olemusest ning vaadeldakse ka selles valdkonnas varem valminud sarnaseid töid. Ühtlasi tuuakse välja, miks on see teema aktuaalne ja kas sellel on ka tulevikku.

2.1 Mängu tutvustus

Bakalaureuse töö käigus realiseeritakse „Snake“ tüüpi mitmikmäng (*multiplayer*), kus eesmärgiks on koguda võimalikult suur punktisumma ja edastada seega oma konkurente. Tegemist ei ole tüüpilise „Snake“ mänguga, sest korraga võib mängida ühes ruumis kuni kümme kasutajat.

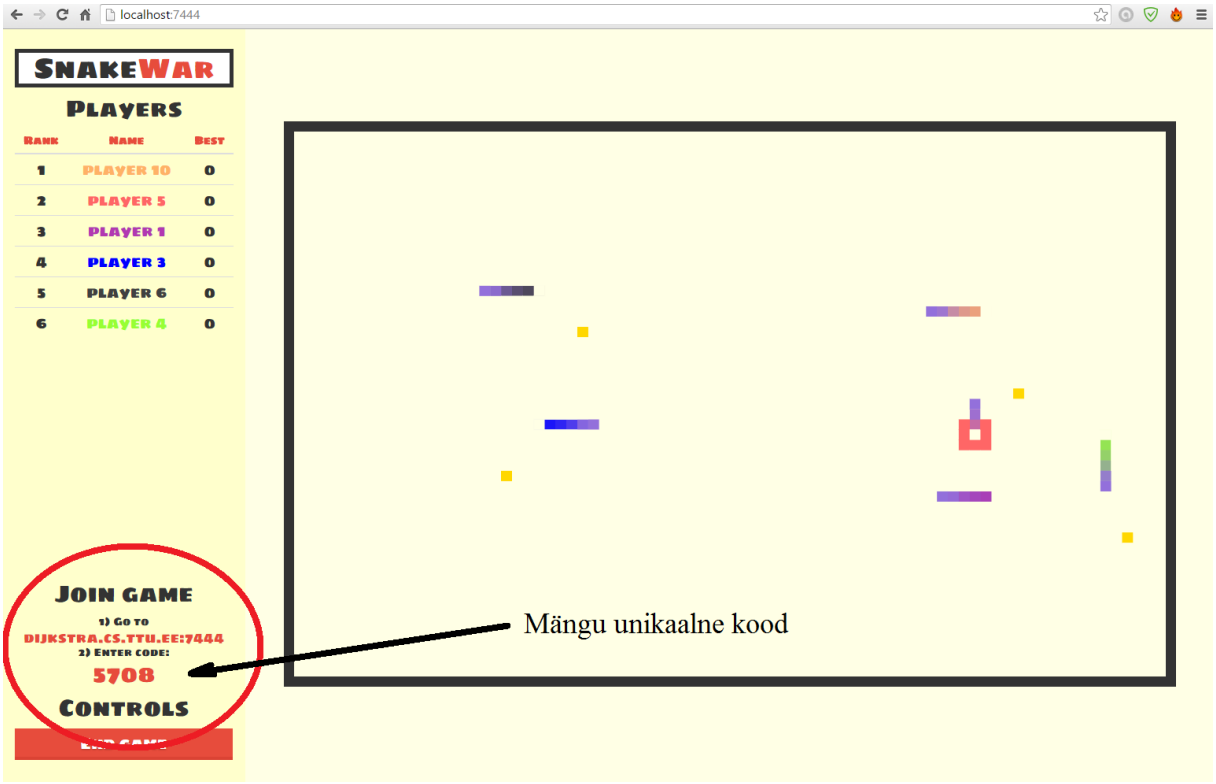
Mängu põhiideeks on põigelda mööda teistest ussidest vältides seina vastu liikumist. Sealjuures peab koguma kollaseid kastikesi, mis annavad punkte. Iga kollase kasti söömise järel kasvab ka ussi pikkus. See tähendab seda, et mida pikem on uss, seda rohkem on ta käimasolevas mängus punkte kogunud. Kui uss põrkab vastu seina, teist ussi või tabab omaenda saba, siis mäng lõppeb ja tulemus salvestatakse. Mängu *host* seadme ekraanil kuvatakse ka edetabelit kõigist ruumis olevatest mängijatest. Edetabel on sorteeritud nii, et üleval on kõige suurema punktisumma saavutanud kasutajad.

Tegelaskuju juhtimiseks kasutakse *host* masinast erinevat seadet, milleks võib olla puutetundliku ekraaniga mobiiltelefon, tahvelarvuti või tavaline arvuti. Seadmed, mis toetavad puuteekraani saavad kasutada mängimiseks ekraanil viipamist (*swipe*). Eelneva funktsiooni puudumisel saab mängida ka klaviatuuril asetsevaid noolenuppe kasutades.

Uue mängu loomiseks ehk *host* seadmeks saamiseks peab olema ekraani laius resolutsioon vähemalt 960 pikslit. See väldib olukordi, kus mänguga ühinevatel inimestel on raske jälgida mängus toimuvat liiga väikselt ekraanilt.

Juba loodud mänguga ühinemiseks on vajalik *host* seadme ekraanilt leida üles mängu unikaalne kood. See asub ekraani vasakul all osas ja koosneb neljast naturaalarvust. Kood tuleb sisestada juhtpuldi liidese avalehel asuvale vormiväljale.

Joonis 1 Mängu unikaalne kood



2.2 Ülevaade valdkonnast

Autori eesmärk on luua mäng, mis kasutab ühe tähtsa osana WebSocket protokolli. WebSocket-i näol on tegemist üpris uue tehnoloogiaga ning seega on antud valdkonnas sarnase ideega rakendusi vähe. Samuti on kasinalt erinevaid õppetekste ja -videoid, mis aitaksid sel teemal teadmisi hankida ja kiiremini edasi areneda.

WebSocket-i kasutamine on muutnud üha populaarsemaks ja igapäevaselt lisandub huvitavaid uusi lahendusi. [4] Usutavasti jätkub ka populaarsuse tõus tulevikus, sest tegu on ühe kõige parema viisiga luua kiiret kahe suunalist ühendust vajavaid veebirakendusi. Ka autor näeb suurt potentsiaali, sest maailm liigub selles suunas, kus kõik infovahetus veebilehtedel peab toimuma koheselt ja viivitusaeg on lubamatu.

2.3 Olemasolevad lahendused

Järgnevalt tutvustatakse paari käesoleva bakalaureusetöö teemaga kokku minevat mängu, mis on valmistatud kellegi teise poolt. Täiesti samal teemal tööd ei ole varem tehtud, kuid on

olemas osaliselt samu tehnoloogiad kasutades ning sarnast loogikat jälgides loodud veebipõhiseid mänge.

2.3.1 Multeor

Multeor on veebipõhine mitmikmäng, kus tuleb maale langenud meteoriiti juhtides võimalikult palju kahju tekitada purustades erinevaid hooneid ja objekte. Mängu on loodud kolme autori poolt, kelleks on Rajan de Vries, Filidor Wiese ja Arthur van 't Hoog. [5]

Mängu visuaalne osa kuvatakse arvutis, kuid seal toimuvat kontrollitakse nutitelefoniga. Maksimaalselt 8 inimest saavad korraga ühes mängus mängida. Mängijatel on võimalus oma meteoriiti ka isikupärastada logides rakendusse sisse oma „Facebook“ kontoga. Mängida saab ka üksinda, kuid autorid soovitavad parema elamuse saamiseks siiski mitmekesi mängimist. Mida rohkem on mängijaid, seda rohkem on hävingut, värve ja kaost.

Mängu arendamisel lähtuti põhimõttest, et rakendus ei pea olema laialdaselt ühilduv ning seetõttu oli neil võimalik kasutada uuemaid tehnoloogiaid. Mäng on kirjutatud *JavaScript* keeles kasutades HTML Canvas-t visuaalse poole kuvamiseks. Mäng toetub NodeJS serverile, mis omakorda aitab nutiseadmel ja *host* seadmel omavahel suhelda kasutades WebSocket-it.

Autori idee luua bakalaureusetööna sarnast põhimõtet jälgiv mäng põhineb just sellel mängul. Antud töö raames valminud mäng kasutab samu tehnoloogiaid ja on kasutanud Multeor mängu inspiratsioonina.

2.3.2 Chrome Super Sync Sports

Chrome Super Sync Sport on veebirakendus, mis lubab kasutajal ühendada oma nutiseadme või tahvelarvuti tavalise arvutiga, et juhtida kolme eriliiki spordivõistlust. See projekt on valmistatud „Google Creative Lab“ poolt koostööna Rafaël Rozendaaliga, WEIR+WONG-ga ja Rami Niemiga. [6]

Mäng koosneb kolmest spordialast, milleks on jooksmine, rattasõit ja ujumine. Kasutaja saab alguses valida, millisest võistlusest soovib osa võtta. Lisaks on võimalik selekteerida endale sobiv tegelaskuju viiekümne erineva sportlase vahel. Korraga saab ühest mängust osa võtta kuni neli kasutajat.

Mängu idee on olla võimalikult kiire ja teisi edestada. Selleks, et olla kiireim, tuleb mobiilseadme ekraanil teha erinevaid liigutusi. Mida rutemini ja täpsemini neid teed, seda vähem aega võtab tegelaskuju finišisse jõudmine.

Rakendus kasutab töötamiseks samuti WebSocket tehnoloogiat, mis võimaldab reaalajas erinevate seadmete vahel koostööd teha. Ilma selleta, ei oleks antud rakendust võimalik sarnaselt realiseerida. Tööd ei ole võimalik valmistada kasutades näiteks AJAX-it, sest ta ei suuda nii kiiresti infot vahetada. AJAX peab alati serverilt küsima, kas on saabunud uusi andmeid, aga WebSocket-ile saadab server ise automaatselt teate uue teabe kohta. Lisaks kasutab programm ära HTML5 Audio teeki, mis lubab kerge vaevaga mängule heli lisada. Mängu visuaalne pool töötab HTML Canvas peal, sest see lubas neil ülimalt täpselt kontrollida seda mida kasutaja näeb. Kuna tegu on Google poolt valmistatud mänguga, siis kasutavad nad veel lisaks Google-i enda poolt valmistatud tarkvara nagu „Google Web Fonts“, „Google Cloud Platform“ ja „Google App Engine“.

2.3.3 Duck Shoot

Duck Shoot on unikaalne 3D mäng, mis on paigutatud veebilehe reklaambännerisse, kust on võimalik seda koheselt mängida kasutades nutiseadet. Mängu eesmärgiks on tulistada parte ümber ja saada võimalikult suur punktisumma. Mäng on valmistatud Toaster Ltd poolt 2013 aastal. [7]

Mängu alustamiseks tuleb nutiseadmega skaneerida sisse QR-kood, mis automaatselt teostab seadme ühenduse arvutiga ja lubab hakata mängima. Tulistamiseks tuleb ekraani sõrmega libistada (*swipe*) sinna suunas, kus part asub. Samuti mängib olulist rolli nutiseadme asend, mis määrab ära kuuli lendamise kõrguse.

Mäng kasutab ühenduse loomiseks WebSocket tehnoloogiat, sest see võimaldab kiiret kahe suunalist ühendust arvuti ja nutiseadme vahel. Mängu autor on ära kasutanud ka suuremas osas nutiseadmetes olevat kiirendusmõõturit (*accelerometer*), mis annab informatsiooni seadme liigutamise ja asendi kohta. 3D graafika jaoks on implementeeritud WebGL tehnoloogia läbi lisateegi „Three.js“.

Selles rakenduses kasutatud tehnikad näitavad sarnaste mängude võimaliku potentsiaali ka reklaamivaldkonnas, kus on väga tähtis millegi uudsega silma jääda. Lisaks on see hea näide sellest, kuidas on võimalik ära kasutada nutiseadme erinevaid võimalusi mängu juhtimiseks.

3. Kasutatud tehnoloogiad

Antud mängu väljatöötamisel on kasutatud mitut erinevat lisateeki ja raamistiku (*framework*), mis on mängu valmistamise võimalikuks teinud või seda oluliselt lihtsustanud. Mõned neist on olnud hädavajalikud – ilma nendeta oleks mängu sellisel kujul valmistamine raskendatud või võimatu. Sellisteks lisadeks on näiteks socket.io, NodeJS ja HTML5 koos jQuery-ga. Täiendavalt on kasutatud ka mõnda programmeermist lihtsustavat teeki, milleks on Express, jQuery Mobile, Bootstrap ja Modernizr. Järgnevalt on lähemalt seletatud, miks mainitud tehnoloogiaid on kasutatud ning mis on nende funktsioon loodud mängus.

3.1 WebSocket

WebSocket on protokoll, mis lubab luua täisdupleks ühendusi üle ühe TCP ühenduse. WebSocket-i protokoll standardiseeriti IETF poolt 2011 aastal, mis teeb sellest üsnagi uudse tehnoloogia. [3] WebSocket on iseseisev TCP-l baasseeruv protokoll ja teda ühendab HTTP-ga ainult see, et tema *handshake* on interpreteeritud HTTP serveri poolt *Upgrade request-ina*. WebSocket lubab pidevat ühendust veebibrauseri ja serveri vahel, mistõttu on võimalik infot reaalajas väikse viibega (*latency*) vahetada. [8] Tänu sellele on võimalik WebSocket-it kasutades luua kiiresti infot vahetava iseloomuga rakendusi ja mänge.

Sarnast probleemi, mis WebSocket proovib lahendada, on üritatud teha ka teisiti ja sarnaseid tehnoloogiaid on loodud ka varem. Nendest kuulsamad on näiteks *long polling*, „Push“ ja „Comet“. [9] Neil kõigil on üks ühine probleem, millest WebSocket neid edastab – nad on kõik tihedalt seotud HTTP-ga, mis teeb nad sobimatuks reaalaja rakenduste loomiseks oma suure viibe (*latency***)** tõttu.

3.1.1 Socket.io

Socket.io on *JavaScripti* teek, mis lubab luua reaalajas veebirakendusi ning on ehitatud Engine.io peale. See võimaldab kiiret kahepoolset ühendust veebibrauseri ja serveri vahel. Socket.io töötab sündmuste põhiselt (*event-based*) ja töötab paljudel erinevatel platvormidel. [10] Socket.io on jagatud kaheks põhiliseks osaks. Esimene on kliendi pool (*client-side*), mis töötab kliendi enda veebibrauseris. Teine on serveripoolne osa, mis baseerub ühel teisel *JavaScripti* teegil NodeJS. See *framework* on lihtsa API-ga, mis teeb arendamise kiireks ja

arusaadavaks. Põhilisteks kasutusaladeks on reaalajas infot vajavad ning saatvad rakendused (*real-time analytics)*. Socket.io teeki kasutavad väga erinevad programmid, milleks on näiteks Microsoft Office, Yammer ja Zendesk. [11]

Käesolevas töös on just socket.io üks kõige tähtsamaid komponente ilma milleta see mäng ei saaks toimida. See raamistik (*framework*) võeti kasutusele just sellepärast, et oleks võimalikult kiirelt võimalik saata infot erinevate seadmete vahel. Antud mängus on vaja saata mängijatel oma seadmest kiiresti infot *host* seadmele, kus see omakorda teeb muudatusi mängus ning saadab vajadusel infot tagasi.

3.2 NodeJS

NodeJS on *JavaScripti* põhine avatudlähtekoodiga teek, mis on ehitatud *Chrome V8 JavaScript engine* peale ja on mõeldud veebis serveripoolse osa programmeerimiseks. [12] NodeJS valmis 2009 aastal Ryan Dahl poolt ja algselt toetas see ainult Linux operatsioonisüsteemi. [13] Tänapäevaks töötab see enamikel platvormidel ja on kasutusel paljude inimeste poolt. Node aitab tegeleda veebisuhtluse ja *back-end* poolega kasutades *JavaScripti* keelt. Node hoiab endas ka mitmeid mooduleid, mis muudavad arendamise lihtsamaks. Moodulid kasutavad hästi struktureeritud ja dokumenteeritud API-t, et lihtsustada serveri poolsete rakenduste programmeerimist.

NPM on NodeJS-ga kaasatulev *package manager*, mis lubab NodeJS programme käsurealt lihtsa vaevaga installeerida. [14] *NPM-ga* on uute teekide alla laadimine kergendatud, sest ühe käsuga saab käsurealt lisada oma projektile palju erinevaid kasulike teeke ja *frameworke*.

Valminud mängus kasutati NodeJS-i, sest see võimaldab *WebSocketitega* lihtsasti töötada kasutades eelmainitud teeki Socket.io. NodeJS võimaldab luua serveri, mis juhib tervet mängu. Ilma serverita ei ole võimalik mängu käivitada, sest viimane on sõltuvuses NodeJS-st ja Socket.io-st.

3.3 Express.js

Express.js on minimalistlik ja paindlik NodeJS veebirakenduse raamistik (*framework*), mis annab võimaluse lihtsamalt arendada ja struktureerida veebi- ja mobiilirakendusi. [15] See võimaldab programmeerijal kergesti organiseerida rakenduse MVC arhitektuuri kasutama. Express aitab korraldada paljusi hädavajalike asju, mis muidu võtavad palju aega. Nendeks on

näiteks veebilehtedel marsruutimisega tegelemine, erinevate vaadete (*views)* loomine, päringute (*requests)* käsitlemine ja palju muud. [16] Tegemist on ühtlasi NodeJS raamistiku kõige populaarsema lisaga.

Antud töös kasutati Expressi põhiliselt failide paremaks struktureerimiseks ja marsruutimiseks. Otseselt see mängu jaoks hädavajalik polnud, kuid tegi arendamise mõne võrra lihtsamaks ja vähem aega nõudvamaks.

3.4 HTML5, jQuery, Bootstrap

Mängu *front-end* pool on ehitatud üles jälgides HTML5 standardeid ja kasutatud on ka abistavaid lisaraamistike. Põhiosa tugineb kõige tavalisemale veebilehe arendus meetodile, kus kasutatakse HTML, CSS ja *JavaScripti*. HTML on vajalik veebilehe struktureerimiseks ja kuvamiseks veebibrauseris. CSS kasutatakse kirjeldamaks, kuidas teatud elemendid veebilehel välja nägema peaksid. CSS kirjutades on silmas peetud CSS3 standardile vastavaid aspekte ja süntaksit.

Rakenduse visuaalne pool töötab HTML5 Canvas peal, mis on madala ressursinõudlusega ja samas tõhus viis *JavaScripti* põhiselt mängu graafikat kuvada.

Vähendamaks töö kujundamis- ja programmeerimisprotsessi, kasutab veebirakendus lisateeki Bootstrap, mis on ilmselt maailma kõige populaarsem *front-end framework*. [17] See sisaldab endas hulganisti valmis programmeeritud veebilehe osi ja funktsioone, mida on kerge arendajal implementeerida. Installeerimine on lihtne ja sisaldab endas ühte CSS faili ja ühte *JavaScript* faili. Mängus aitab Bootstrap muuta disaini kohanduvaks ja abistab kirjastiilidega tegutsemist.

Terve mängu dünaamiline osa ja loogika põhineb *JavaScripti* peal, mis on veebipõhine brauseris töötav programmeerimiskeel. *JavaScript* on kõige tuntum *ECMAScripti* osa ja seda toetavad kõik modernsed veebibrauserid. [18] Lihtsustamaks *JavaScripti* kirjutamist on mängus kasutatud ka *JavaScriptil* põhinevat teeki jQuery. jQuery on suur hulk valmis kirjutatud *JavaScripti* funktsionaalsust, mis võimaldab kergemini programmeerijal dünaamilisi veebe luua. Peale lihtsustatud funktsioonide sisaldab jQuery endas ka algelisi animatsioone, millest antud töös on kasutatud näiteks *fade in* ja *fade out*. Esimene nendest muudab nähtamatu elemendi sujuvalt nähtavaks ja teine nähtava elemendi sujuvalt nähtamatuks.

Mobiilse kasutajaliidese valmistamisel on kasutusel jQuery Mobile, mis on puuteekraanide jaoks optimeeritud *framework* jQuery peal. See on samuti kirjutatud *JavaScript* keeles ja vajab töötamiseks juba eelnevalt lisatud jQuery teeki. jQuery Mobile sisaldab endas palju funktsionaalsust, mis on just puutetundlikute ekraanidega seadmetele mõeldud. Mängu *controller* mobiilil kasutab seda teeki ekraanil erinevate liigutuste ära tundmiseks. Nendeks on näiteks *swipe up* ja *swipe down*, mis tähendavad ekraanil näpuga libistamist ülesse või alla. Just selliste sündmuste tuvastamiseks on jQuery Mobile antud projektis vajalik.

3.5 Modernizr

Modernizr on *JavaScripti* teek, mille eesmärk on tuvastada HTML5 ja CSS3 funktsioonide töötavust erinevates veebibrauserites. See teek aitab vältida olukordi, kus veebilehele tulev inimene kasutab vana versiooniga brauserit, mis ei suuda tõlgendada ja kuvada uute funktsioonidega koodi. Modernizr aitab kaasa, et kasutajale kuvataks talle sobivat sisu vastavalt tema brauseri võimetele.

Modernizr tööpõhimõte on lihtne ja arusaadav ning seetõttu saab selle kasutusel võtta ilma suurema õppeprotsessita. Modernizr-it kasutades saab valida, kas kontrollida kõiki funktsioone või ainult osaliselt. Valikute tegemine on kasulik, sest enamasti ei ole vaja kõikide funktsioonide olemasolu testida ja nii on võimalik säästa mälu.

Kui veebilehele on installeeritud Modernizr raamistik, siis lehe avamisel lisab see html teegi klassi atribuudile uusi väärtusi. Kui soovitud funktsioon on olemas, lisatakse see uue klassi väärtusena, aga kui brauser seda ei toeta, siis lisatakse eelliide „no“. Järgnevalt saab kontrollida *JavaScripti* abil, millised klassid lisati ja kas need on eesliitega „no“ või mitte. Vastavalt sellele saab teha veebilehel muudatusi.

Arendatud mängus avamisel testib Modernizr, kas brauser toetab puutetundlikku ekraani. Vastavalt sellele tehakse disainis muudatusi ning kuvatakse sobilikud tekstid ja pildid.

4. Juhtpuldi kasutajaliidese arendus

Projekti üheks tähtsaimaks osaks oli luua kasutajaliides, millega mängijad saaksid oma tegelasi liigutada. Liidese planeerimisel oli vaja keskenduda paljudele aspektidele, mis kõik pidid olema hästi läbi mõeldud, et kogu tervik töötaks. Järgnevalt tuuakse välja täpsemalt erinevad nõuded liidesele ning näidatakse, miks on tehtud teatud valikud.

4.1 Nõuded kasutajaliidesele

Juhtpuldi valmistamisel tuli algselt palju mõelda ja katsetada ning alles siis panna paika kindlad nõuded, mille suunas liikuda. Sarnaseid liideseid pole palju loodud ja seega tuli võtta ideid nendest vähestest. Lisaks oli vaja ise proovida erinevad variandid läbi ja nende hulgast selgitada välja parim. Katsetusprotsessi käigus jäid kehtima all toodud nõuded.

Üldised nõuded:

- Juhtpulti peab saama kasutada nii arvutist, tahvelarvutist kui ka mobiililt
- Juhtpult peab töötama erinevate rohkem levinud operatsioonisüsteemidega ja uuemate veebibrauseritega
- Kasutajasõbralik – juhtpuldi käima saamine peab olema lihtne ja kiire.
- Skaleeruv disain
- Punktisummat kuvatakse liideses jooksvalt
- Igal mängijal on oma unikaalne värv ja nimi
- Surres vilgub ekraan punase värviga
- Punktisummat suurendades vilgub ekraan rohelise värviga
- Ruumi sulgemisel kuvatakse vastav sõnum ja suunatakse avalehele
- Ühes ruumis võib maksimaalselt mängida 10 mängijat

Tavaekraaniga liidese lisanõuded:

- Mängus sees olles saab uuesti sündida vajutades nuppu *enter* või *space*
- Mängu juhitakse klaviatuuriga kasutades noolenuppe
- Kuvatakse juhised nooltega mängimiseks

Puutetundliku ekraaniga liidese lisanõuded:

- Mängu juhitakse sõrmega ekraanil üles, alla, paremale või vasakule libistades (*swipe*)
- Nupud peavad olema paigutatud nii, et ekraanil *swipe-des* need ei aktiveeruks
- Kuvatakse juhised puuteekraanil mängimiseks

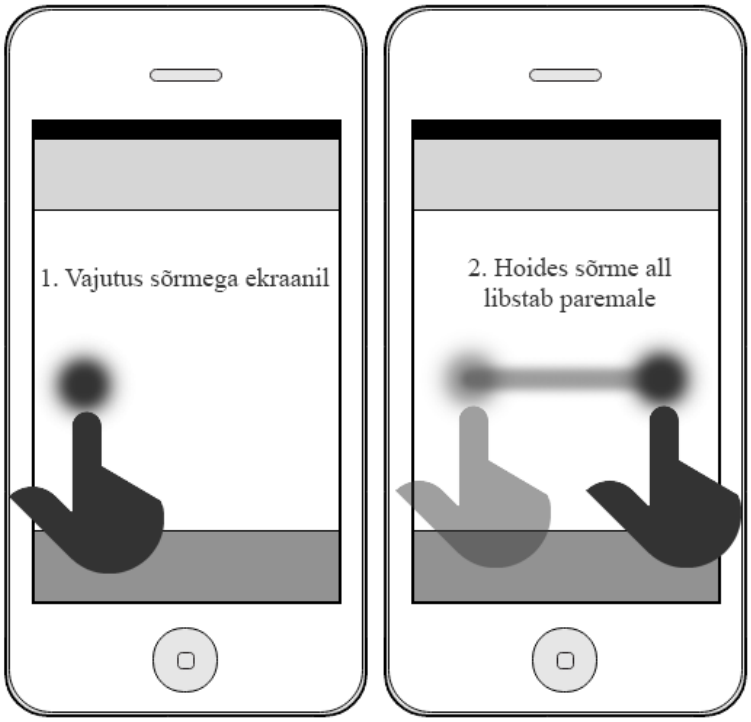
4.2 Puutetundliku juhtpuldi tüübi valikud

Järgnevalt on välja toodud mõned levinumad võimalused, kuidas luua puutetundlikku juhtpulti (*controller).* Puutetundliku ekraani peal on võimalik disainida väga erinevaid liideseid, mis kontrollivad mängus olevaid tegelasi. Iga mängu jaoks on vaja eraldi läbi analüüsida kõik võimalused ning teha selgeks, milline viis on rakenduse jaoks parim.

Käesoleva mängu juhtpuldi disaini ja loogika väljatöötamisel mõeldi läbi erinevaid situatsioone, kuidas *controllerit* mängus kasutatakse. Seejärel implementeeriti neist kaks ja testiti, kumb on selle mängu jaoks sobivaim. Esimeseks oli *swipe* liigutust kasutav liides ja teiseks füüsilisi nuppe imiteeriv juhtpult.

4.2.1 Swipe

Swipe ehk ekraani peal näpuga libistamine on võrdlemisi uus liigutus ja inimesed on selle kiiresti omaks võtnud. *Swipe* kujutab endast ette liigutust, kus sõrm asetatakse puutetundliku ekraani peale ja libistatakse punktist A kuhugi punkti B. (Joonis 2) *Swipe-i* pikkuse punktist A punkti B määrab iga programmeerija ise. See tähendab, et iga väikest liigutust ekraanil ei ole vaja registreerita *swipe-ina* ja iga rakendus kasutab seda vastavalt vajadusele.



Joonis 2 Swipe liigutus

***Swipe* eelised:**

- Ei pea ekraani vaatama
- Kiiresti lihasmällu talletuv

***Swipe* miinused:**

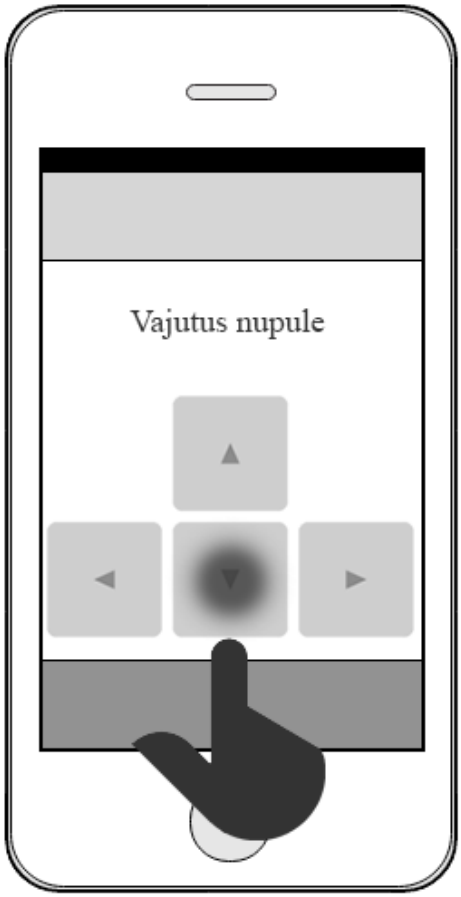
- Liigutus võtab aega
- Programm ei pruugi alati liigutusest aru saada
- Mõned mobiilibrauserid kasutavad *swipe-i* veebilehe navigeerimiseks edasi ja tagasi, mistõttu võib paremale ja vasakule viipamine tuua kaasa vale tegevuse

4.2.2 Imiteeritud füüsilised nupud ekraanil

Kõige tüüpilisem viis mängu juhtida on ekraanile kuvada nupud, mis imiteerivad füüsilisi nuppe. Seda viisi kasutatakse mängudes väga palju ja tegu on ka ühe äärmiselt efektiivse

lahendusega. Kasutajate jaoks on selline juhtpult loogiline ja arusaadav. Väikestes seadetes (mobiiltelefonid) on väga oluline nuppude paigutus ja suurus hästi läbi mõelda, sest valesti paigutatud või liiga väikesed nupud võivad kasutajale hulgaliselt pahameelt tekitada. Nuppudele vajutamine toimub lühikese puutega ekraani vastu. (Joonis 3)

Joonis 3 Nuppudega liidese kasutamine



Nuppude eelised:

- Kasutaja jaoks lihtsasti arusaadav
- Lihtne implementeerida mängu

Nuppude miinused:

- Kasutaja peab vaatama ekraani, et tabada õiget nuppu
- Võib tabada juhuslikult vale nuppu

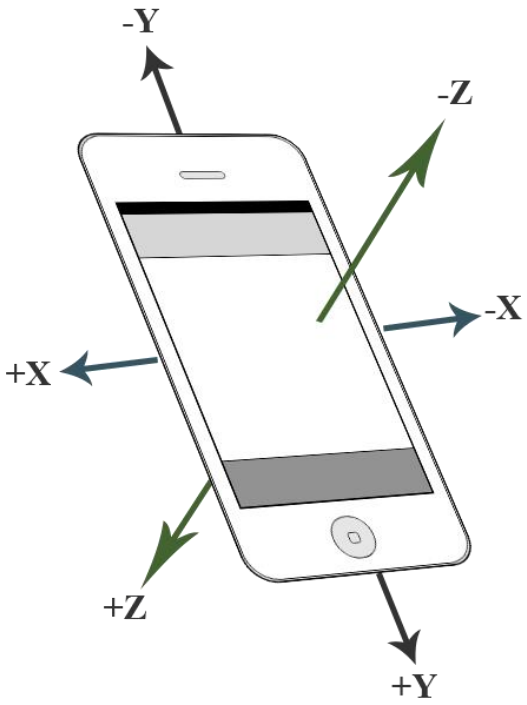
- Puudub füüsiline tagasiside

4.2.3 Kiirendusmõõtur

Kiirendusmõõtur ehk aktseleromeeter (*accelerometer*) on andur, mille abil saab mõõta seadme kiirendust ja gravitatsioonijõudu. [19] Aktseleromeeter annab võimaluse kätte saada info nutiseadme ruumilise liikumise kohta. Enamustes uuemates nutitelefonides ja tahvelarvutites on kiirendusmõõtur sisseehitatud ja seega on tegu ühe väga tõsiselt võetava viisiga mängus tegevust juhtida. Tänu HTML5 API-le on lihtne brauserist kätte saada seadme liikumise X, Y ja Z koordinaadid, mille tulemuste põhjal saab programmis teha vastavaid muudatusi. (Joonis 4)

Accelerometer-ga mängu üks kõige tüüpilisemaid juhtimisviise on seadme kallutamine kuhugi suunas ja sellest lähtuvalt tegelase liigutamine. Sellele lisaks on võimalik programmi tuvastama panna ükskõik mis liigutusi. Näiteks kasutavad paljud rakendused seadme raputamise (*shake*) tuvastamist ja selle registreerimisel viiakse läbi mingi funktsioon, milleks võib olla näiteks info värskendamine (*refresh*).

Joonis 4 Kiirendusmõõturi tööpõhimõte



Kiirendusmõõturi eelised:

- Ei vaja ekraani jälgimist (kui kasutada juhtpuldina)
- Kasutajale lihtsasti arusaadav

Kiirendusmõõturi miinused:

- Liigutades seadet on raske telefoniekraani jälgida (kui ekraanil on vajaliku infot)
- Kontrolli säilitada on raske (iga väike liigutus registreeritakse)
- Ei sobi kiireloomulistele mängudele
- Aeglane

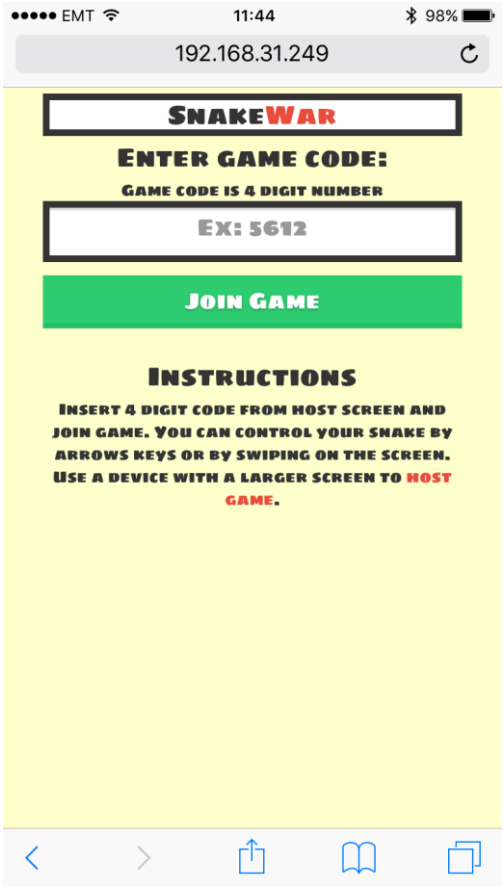
4.3 Valminud juhtpuldi liides

Erinevate meetodite läbitöötamise käigus valmis nõuetele vastav kasutajaliides, mis on ühelt poolt kasutajasõbralik ja samas hea välimusega. (Joonis 5) Juhtpuldi tähelepanu on suunatud mängu unikaalse koodi sisestamisele, mille korrektsuse korral saab ühineda soovitud mänguga. Lisaks on kohe avalehel välja kirjutatud lühike instruktsioon mängu kohta. See aitab kerge vaevaga inimestel, kes pole mängu varem mänginud, aru saada, kuidas mänguga ühinemine toimub. Ebakorrektse sisestuse puhul kuvatakse sisestusvormi ülesse veateade, mis torkab kasutajale kohe silma. Juhtpuldiga on võimalik mängida nii arvutis kui ka puutetundliku ekraaniga seadmetel. Arvutis mängides toimub tegelase juhtimine klaviatuuri nuppudega (Joonis 7), aga puuteekraanil kasutades *swipe* tehnikat. (Joonis 6)

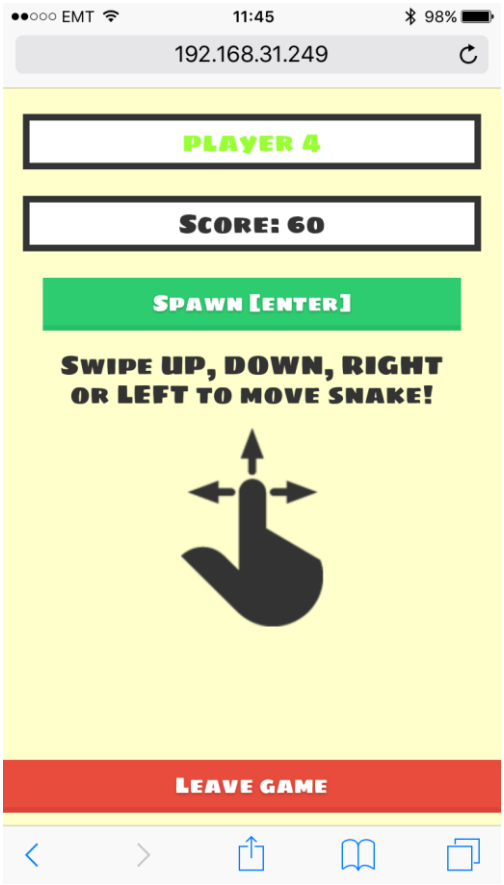
Mängu autor otsustas projekti raames kasutada juhtpuldi loogika implementeerimiseks *swipe* ehk ekraani peal näpuga libistamist oma tegelaskuju liigutamiseks. Mängu seisukohalt oli tähtis, et kasutaja ei peaks tegelase juhtimisel vaatama pidevalt mobiiliekraani. Lisaks on „*Snake*“ tüüpi mängus vaja teha kiireid liigutusi ja kõige mugavamalt saab neid teha ekraanil näppu libistades.

Imiteeritud füüsiliste nuppude variant sai välistatud kohe peale esimesi teste, sest see ei sobi kokku mängu kontseptsiooniga. Kasutaja jaoks ei ole mugav pidevalt mobiiliekraani vaadata ja mõelda, et kuhu täpselt ta vajutama peab. Pidevalt tuleb jälgida *host* seadme ekraani, sest kõik mängus toimuv on näha ainult sealt. Edasi-tagasi vaadates muutub selline viis äärmiselt tülikaks ja ebamugavaks.

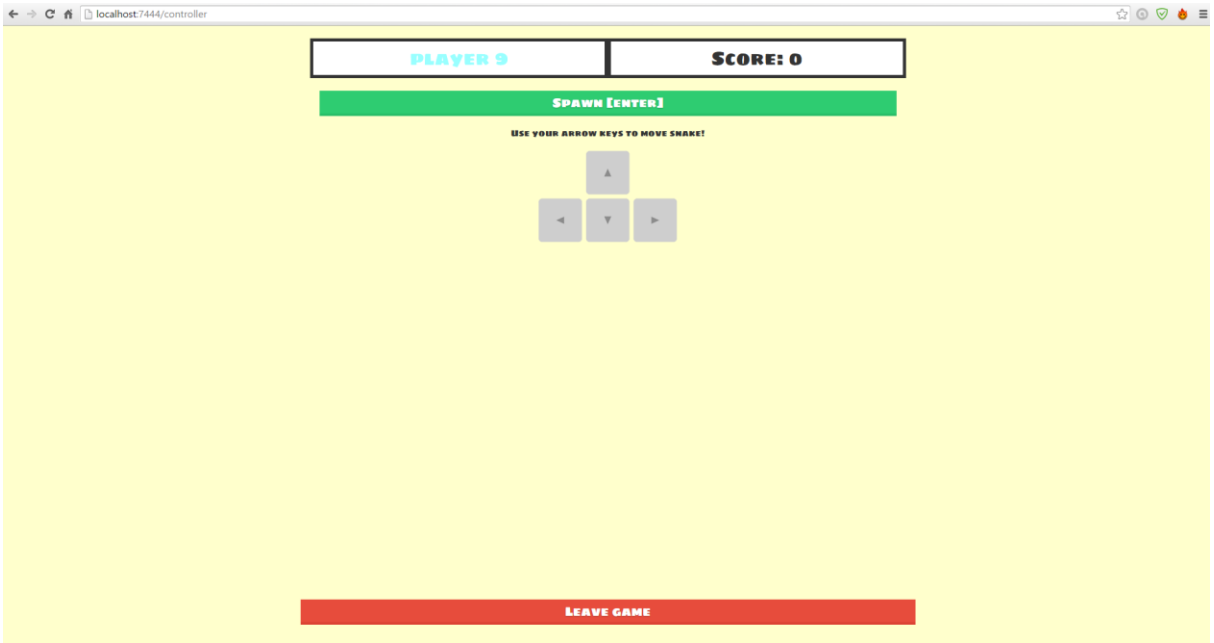
Aktseleromeetri kasutamine oleks sobinud käesoleva mängu juurde ainult sellel põhjusel, et see ei vajanud kasutaja pidevat ekraani jälgimist. Kiirendusmõõturi kasutuselevõtt jäi suuresti selle taha, et selle liigutuste täpsus ja aeglane kiirus on mängu jaoks vastuvõtmatu.



Joonis 5 Puuteekraani juhtpuldi liidese avaleht



Joonis 6 Aktiivne puuteekraani juhtpult



Joonis 7 Aktiivne tavaekraani juhtpult

5. Mängu „SnakeWar“ loomine

5.1 Mängu arendus

Mängu arenduskäik koosnes erinevatest etappidest alustades vajalike teadmiste ja info kogumisest. Programmeerimistööd jaotati sisu järgi osadeks, mis kõik olid omavahel seotud. Järgnevas peatükis on need etapid eraldi lahti seletatud ja põhjendatud.

Arendusprotsessis kasutati versioonikontrolli süsteemi GIT, mis pakub võimalust salvestada, hallata ja levitada failide erinevaid versioone. [20] GIT-i kasutamine andis autorile meelerahu, et töö on pidevalt säilitatud ja lokaalse masina vigade korral on võimalik koodi taastada.

5.1.1 Arenduskäik

Arenduskäik koosnes kaheksast etapist:

1. **Tutvumine valdkonnaga**

   Teema oli algselt autorile võõras ja seega oli vaja enne arenduse alustamist omandada põhiteadmised antud valdkonnas. Lisaks uuriti juba tehtud töid ja nende lähtekoode, mis andsid ülevaate mängu arhitektuurist ja keerukusest.

2. **Mängu lihtsustatud versiooni implementeerimine**

   Olles omandanud põhiteadmised algas koheselt programmeerimine. Siin kohal proovis autor luua võimalikult lihtsa, kuid samas sarnast tehnoloogiat ja põhimõtet jälgiva mängu demoversiooni. See oli vajalik, et kinnitada omandatud teadmisi ja testida, kas valitud teema on autorile jõukohane.

3. **Vahetulemuste tegemine ja kindla plaani paika panemine**

   Peale demomängu valmimist oli õige hetk panna paika kindlad plaanid töö valmistamise osas. Siinkohal töötati valmis ekraanikuvandid, kuidas lõpp-produkt välja nägema peaks. Lisaks sõnastati mängureeglid ja nõuded. Viimaks määrati erinevate arendusetappide kirjeldused ja tähtajad.

4. Mängu valmis programmeerimine

Programmeerimistöid alustati mängu valmistamisest, mis on ka projekti üheks kõige tähtsamaks etapiks. Autor implementeeris „*snake*“ tüüpi mängu, mida oli võimalik mängida klaviatuuril samas arvutis noolenuppude abil. Siinkohal arvestati seda, et kõik objektid, mida tulevikus tuli duplitseerida, oleksid õigesti programmeeritud ja vajadusel kergesti laiendatavad.

5. *Host* seadme liidese disain

Mängu valmides oli tegu suure *canvas* objektiga, mis oli terve ekraani suurune. Selle ümber tuli ehitada liides, mis aitab mänguga ühineda, uut mängu luua ja ka mänge sulgeda. Liidese loomisel oli tähtis, et see oleks skaleeruv erinevate ekraanide peal ning kasutajasõbralik.

6. Juhtpuldi liidese disain

Järgnevalt loodi liides mänguga ühineda soovivate seadmete jaoks. Rohkem juhtpuldi liidese kohta saab lugeda peatükist 4.

7. Serveri lisamine

Olles valmis saanud nii *host* kui ka juhtpuldi liidese, oli vaja see omavahel suhtlema panna. Selleks kasutati NodeJS-l baseeruvat *frameworki* socket.io. Tegu oli autori jaoks ühe raskeima etapiga, sest vaja oli ühendada kaks täiesti eraldiseisvat veebilehte koos töötama ja infot vahetama. Andmeid tuli saata omavahel ülimalt kiiresti ja seega oli tähtsal kohal koodi hoolikalt planeerimine ja optimeerimine. Selle etapi lõppedes olid mängus kõik põhifunktsioonid implementeeritud ja oli võimalik juhtpuldiga teisest seadmest mängu juhtida.

8. Vigade parandamine ja testimine

Viimaseks arendusfaasiks oli valminud mängu testimine ja selle käigus tulenevate vigade (*bug*) elimineerimine. Selle tsükli käigus katsetati kõiki võimalike mängu situatsioone ja vaadeldi, kuidas rakendus neile reageerib. Testimise käigus avastati paar viga, mis said parandatud. Mängu lasti proovida ka tavakasutajal, kes polnud

enne mängust midagi kuulnud ning ei teadnud mis neid ees ootab. Tänu sellele tuli välja enne avastamata vigu ja kitsaskohti kasutajakogemuses (*UX*).

5.1.2 Juhtpuldi ja *host* seadme omavaheline suhtlus

Projekti võib-olla et kõige olulisemaks osaks on kahe eraldiseisva veebirakenduse omavahel suhtlema panek võimalikult väikse viite ajaga (*latency*). Selleks otstarbeks kasutati NodeJS-il põhinevat veebiraamistiku socket.io, mis omakorda põhineb WebSocket protokollil.

Socket.io API kohaselt saab sõnumeid saata mitut erinevat viisi. [21] Esiteks on võimalik saata sõnum kõigile serveris olevatele seadmetele kaasaarvatud saatjale endale. Teiseks on võimalus edastada andmeid kõigile serveris olevatele seadetele, aga mitte iseendale (*broadcast*). Kolmandaks on võimalik sõnumeid edastada ainult teatud ruumis olevatele inimestele. Kõiki variante on töös kasutatud ja erinevatel juhtudel on kõigil omad eelised.

Juhtpuldi ja *host* seadme omavaheline suhtlema panek on tehtud kergeks ja järgib lihtsat loogikat. Järgnevalt on näitena toodud üks juhtum mängust, kus juhtpuldil tehakse liigutus paremale ning seejärel näidatakse, kuidas see info lõpuks *host* seadmeni jõuab.

1. **Juhtpult edastab sõnumi serverile**

   Tundes ära viipe paremale edastab programm sõnumi serverile objektiga, mis sisaldab erinevaid andmeid liigutuse teinud kasutaja kohta. Sellisel juhul kasutab programm funktsiooni *socket.emit(data)*. Juhtpult saadab serverile sõnumi sisuga ’move’.

```
$(document).on("swiperight",function(){
    if(room != null) {
        socket.emit('move', {
            user: socket.id,
            room: room,
            host: host,
            move: 39
        });
    }
});
```

2. **Server registreerib sisse tulnud sõnumi ja edastab selle**

   Server kuulab konstantselt kõiki sõnumeid, mis edastatakse erinevate seadmete poolt, ja tegutseb kuuldes õiget märksõna. Kasutaja andmed, kellele sõnum saadetakse, on ainuke info, mida serveril antud juhul vaja on. Nüüd kasutatakse kindlasse ruumi

saatmise funktsiooni *io.to(room).emit(data)*. Server saadab andmed sõnumiga ’newmove’ mängu *host* seadmele, mis asub teatud ruumis.

```
socket.on('move', function (data) {
    io.to(data.room).emit('newmove', data);
});
```

3. **Mängu *host* saab kätte info ja liigutab ussi**

Mängu *host* seade kuulab samuti koguaeg serveris (antud juhul ruumis) toimuvat ja reageerib teatud sõnumitele. Kuuldes sõnumit ’newmove’ saab programm aru, et tuleb liigutada ussi vastavas suunas. Seejärel käivitatakse juba lokaalses mängus funktsioon, mis teeb vastava muudatuse.

```
socket.on('newmove', function (data) {
    moveSnake(data);
});

function moveSnake(data) {
    ...
    snakeList[i].keystate[data.move] = true;
    ...
}
```

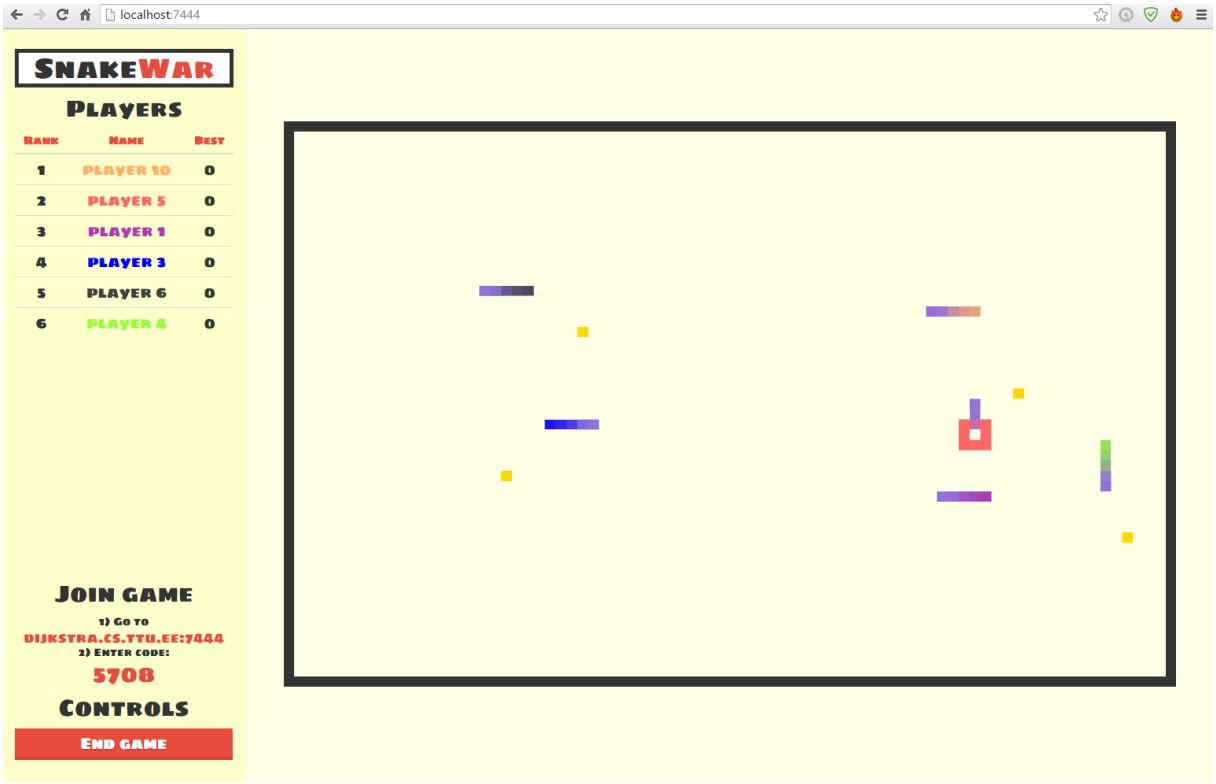
5.1.3 Valminud mäng

Lõplik versioon mängust kujutab endast mitmikmängu, mis on inspireeritud „snake“ mängu olemusest ja reeglitest. Mängu teeb eriliseks võimalus juhtida oma tegelaskuju teise seadme pealt, mis on ühendatud läbi serveri mängu looja arvutiga. Korraga saab ühes ruumis mängida maksimaalselt 10 mängijat, mis on hetkel seatud ülemiseks piiriks. Vajadusel saaks seda piirangut ka suurendada või vähendada. (Joonis 8)

Mängu eesmärgiks on koguda võimalikult palju punkte süües kollaseid kastikesi ja seeläbi edastada enda konkurente. Tuleb vältida kokkupõrget seinte ja teiste tegelaskujudega. Kokkupõrke järel saab kasutaja uuesti mänguga ühineda ja uuesti alustamisel pole mingit piirangut seatud. Kollisiooni vältimiseks tuleb leida kõige optimaalsem tee kollaste objektideni ning sellest üle minna, misjärel kasvab ka tegelaskuju pikkus.

Ruumidega ühinemine toimub neljakohalise numbrikombinatsiooni sisestamisel juhtpuldi liideses. (Joonis 1) Kui mängija soovib uue tegelaskuju mänguväljale luua, siis tuleb vajutada selleks „spawn“ nuppu või klaviatuuriga seadmetel klahvi „enter“.

Joonis 8 Valminud mäng host seadmes



5.1.4 Tähtsamad situatsioonid mängust

5.1.4.1 Situatsioonid "host" seadmes

1. **Mänguga ühinemine**

   Teise mänguga ühinemiseks tuleb vajutada nupule „Join Game", mis suunab kasutaja edasi juhtpuldi kasutajaliidesesse. Seal edasi toimuv on kirjeldatud punktis 5.1.4.2.

2. **Uue mängu loomine**

   Uue mängu loomiseks tuleb avalehel vajutada nuppu „Host Game", mis tekitab serverisse uue ruumi, kus *host* seade lisatakse automaatselt. *JavaScript* genereerib neljakohalise ruumi koodi, mida kuvatakse uue akna vasakul all nurgas. See on unikaalne kood, mille järgi saavad teised kasutajad selle ruumiga ühineda.

3. **Mängu lõpetamine**

   Mängu lõpetamiseks peab *host* kasutaja vajutama vasakul all nurgas olevat nuppu „End Game". Seejärel ruum kustutakse serverist ja kõiki mängijaid informeeritakse veateatega (vaata peatükki 5.1.4.3).

5.1.4.2 Situatsioonid “controller” seadmes

1. **Mänguga ühinemine**

   Mänguga ühinemiseks peab sisestama neljakohalise koodi, mille leiab *host* seadme ekraani vasakult alt nurgast. Ilma koodita ei ole võimalik ruumiga ühineda. Õige koodi sisestamise järel tuleb kinnituseks vajutada nuppu „Join Game“, mis saadab päringu serverisse. Serverist tulev vastus ütleb, kas on võimalik ruumiga ühineda või mitte. Ühinemise õnnestumisel kuvatakse kasutajale juhtpuldi vaade, millega on võimalik mängus ussi liigutada.

2. **Uue tegelaskuju loomine**

   Uue tegelaskuju loomiseks mänguväljale on vaja vajutada juhtpuldil asetsevat nuppu „Spawn“. Selle tulemusel ilmub *host* masina ekraanile uus ussi (*snake*) objekt.

3. **Ruumist lahkumine**

   Ruumist lahkumiseks tuleb vajutada liidese allosas olevat nuppu „Leave Game“. Seejärel eemaldatakse kasutaja ruumist ja kuvatakse uuesti avaleht.

4. **Skoori suurendamine**

   Kui tegelaskuju läheb üle kollase kastikese siis punktisumma muutub ja seda kuvatakse koheselt ka juhtpuldi liideses. Lisaks illustreerib seda tervet ekraani hõlmav kiire roheline välgatus.

5. **Tegelaskuju sureb**

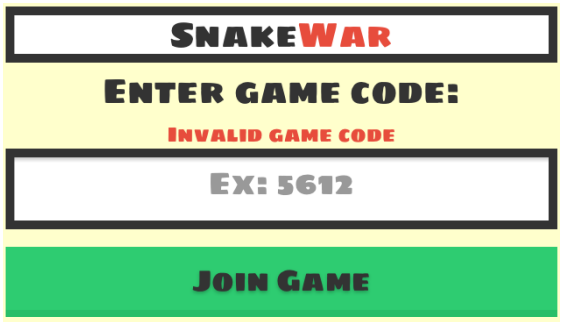
   Kui tegelaskuju põrkab kokku seina, mõne teise ussi või iseendaga, siis selle tegelaskujuga enam mängida ei saa. Juhtseadmel toimub tervet ekraani hõlmav kiire punane välgatus. Juhtpuldi liideses jääb viimase mängu punktisumma nähtavaks kuni uue mängu alustamiseni.

5.1.4.3 Veateated

Veateateid kuvatakse kasutaja informeerimiseks ette tulnud veast või muutusest mängus. Veateated on kõik seotud juhtpuldi liidesega ja järgnevalt on lähedamalt räägitud kolmest.

1. **Puudub sisestatud koodiga ruum**

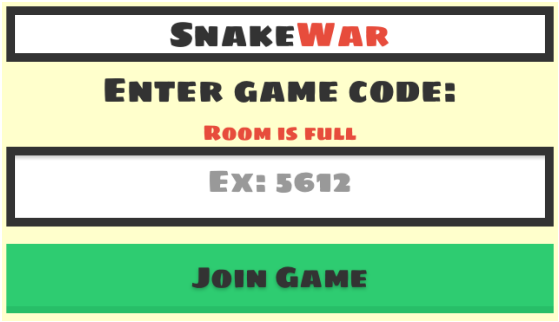
   Kasutaja sisestab juhtpuldi liidese vormiväljale ruumi koodi, mida ei ole serveris veel loodud. Kasutajale kuvatakse veateade tekstiga: „Invalid game code“. (Joonis 9)



Joonis 9 Veateade: "Invalid game code"

2. **Ruumis ei ole vabu kohti**

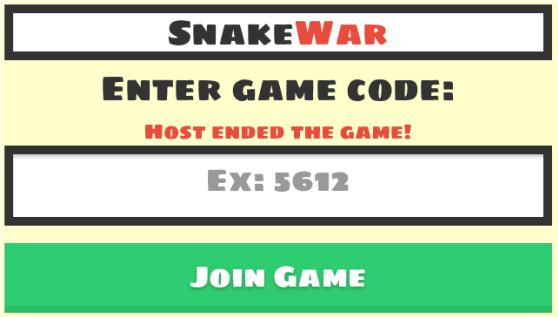
   Igas ruumis saab olla maksimaalselt 10 mängijat ja kui see summa on täis saanud, siis uutel mängijatel ühineda ei lubata. Vastasel korral kuvatakse kasutajale veateade sisuga: „Room is full“. (Joonis 10)



Joonis 10 Veateade: "Room is full"

3. ***Host* katkestas serveri**

   Mänguruum, kus kasutaja mängis, katkestati ruumi *host* seadme poolt. Mängijale kuvatakse veateade tekstiga: „Host ended the game!“ (Joonis 11)



Joonis 11 Veateade: "Host ended the game"

5.2 Valminud mängu analüüs

Mängu kavandades pani autor paika nõuded ja mänguga seotud reeglid. Püstitatud nõuded ja reeglid said kõik realiseeritud. Mäng valmis algselt planeeritud etappide järgi, kuid võttis plaanitust rohkem aega. Vale ajaplaneering oli tingitud sellest, et autor ei olnud antud teemaga enne palju kokku puutunud ja ei osanud korrektselt arvestada mõne osa keerukust.

Programmeerimise käigus ilmnesid mitmed keerukused, mis nõudsid palju pühendumist ja aega. Üheks selliseks osaks oli *host* ja juhtpuldi seadme omavahel ühilduma panek. Välja võib veel tuua algse mängu programmeerimise, sest autor ei olnud enne graafikaga veebipõhiseid mänge loonud. HTML5 Canvas-e peale rakenduse ehitamine oli arvatust raskem.

Mängu käigus kasutatud programmeerimiskeeled ja lisateegid osutusid sellise mängu arendamisel otstarbekaks. *JavaScripti* põhise mängu suurte laiendusvõimaluste tõttu oli programmeerimine lihtne, sest kasutada sai palju abistavaid lisateeke. Veebipõhise rakenduse testimiseks ja vigade otsimiseks oli suureks abiks „Google Chrome-i“ sisse ehitatud „Chrome Developer Tools“.

Valminud mäng töötab erinevates modernsetes veebibrauserites ja on seega paljude erinevate seadmete peal võimalik käitada. Probleemid võivad esineda vanemate veebilehitsejatega, mis ei toeta uuemaid lisateeke. Selliste probleemide tuvastamist hetkel veebirakendus teha ei oska.

Mängu testiti nelja inimese poolt, kes igaüks andis autorile tagasisidet. Mõned tekkinud probleemid olid korduvad ning neid pidas autor ka kõige tähtsamaks parandada. Üheks probleemiks oli juhtpuldi kasutajaliidese liiga väike nupp tegelaskuju uuesti sündimiseks. Seda oli testijate sõnul raske tabada. Viga sai parandatud ja see nupp muutus suuremaks. Testgrupile meeldis, et punktisumma suurendades või surres väljendub see ka juhtpuldi

seadme ekraanil. See tegi nende hinnangul mängimise nauditavamaks ja kaasas neid rohkem mängu. Kõige rohkem meeldis neile idee, et nad saavad mängu juhtida oma nutitelefonilt ja ei pea selleks midagi lisaks installeerima.

6. Kokkuvõte

Töö eesmärgiks oli realiseerida mitmikmäng (*multiplayer game*), mida on võimalik juhtida üle võrguühenduse mõne teise seadmega. Rakenduse kasutamiseks ei ole vaja midagi lisaks installeerida, sest mäng on veebipõhine ja töötab kõigis uuemates veebibrauserites. Lisaks uuriti kasutatud tehnoloogia sobivust valitud teemal ja selgitati välja erinvate juhtpuldi kasutajaliideste omadusi.

Töö tulemusena valmis mäng „SnakeWar“, mis realiseeriti kasutades erinevaid raamistike ja lisateeke. Suurem osa tööst on kirjutatud *JavaScript* programmeerimiskeeles ja pideva ühenduse loomiseks kasutaja ja *host* masina vahel on kasutatud WebSocket tehnoloogiat. Valminud rakendus võimaldab kuni kümnel mängijal korraga ühes ruumis mängida klassikalist „Snake“ tüüpi mängu. Töö käigus selgitati välja erinevate juhtpuldi kasutajaliideste positiivsed ja negatiivsed omadused ning nendest kõige sobilikumaks selle töö raames osutus näpuga ekraanil viipamise tehnika (*swipe*).

Autor jõudis järeldusele, et valitud tehnoloogiad olid sobilikud sellise mängu realiseerimiseks ja näeb WebSocket tehnoloogias suurt potentsiaali. Rakenduse hoogne mängukäik pani tõsiselt proovile WebSocket-i toimimiskiiruse, mis oli hädavajalik kahe seadme omavahel suhtlemiseks. Korraliku veebiühenduse olemasolul ei tähendanud autor suuremaid latentsus (*latency*) probleeme ja seega võib öelda, et selline viis mängude loomiseks on kohane. Mängu looja on kindel, et samu tehnoloogiaid kasutades on võimalik luua veel keerukamaid ja kõrgema graafilise tasemega mänge.

Lisaks mängudele, võib see tehnika kasutusviis otstarvet leida ka kommertsaladel. Töö kirjutaja arvates võib see väga edukaks osutuda just reklaaminduses, kus on tähtis luua kasutajaga personaalne suhe.

Viimaks tahab autor välja tuua tõsiasja, et mängude valmistamisel on eriti tähtis osa selle testimisel just teiste kasutajate peal. Sel viisil tulevad välja probleemid ja kitsaskohad, mida programmi autor ise ei pruugi algselt näha. Tänu kasutajatestidele sai töös parandatud nii mõnigi loogikaviga ja täiustatud kasutajakogemust (*UX*).

Olles uurinud eeltoodud teemasid ja väljaarendanud valmis mängu, jõudis autor oma püstitatud probleemide lahenduseni. Lõputöö edasiarendus võimalusteks on mängu

programmikoodi mõni optimeerimisalgoritm implementeerida ja seega mängu jõudlust parandada. Täiendavalt võib lisada heliefekte ja täiustada graafikat.

Summary

The purpose of the thesis was to implement a multiplayer game that can be controlled over the network by other devices. There is no need to install any extra software to play the game, because it is web-based and works in all modern web browsers. Furthermore, the author investigated different gamepad user interfaces and evaluated the chosen technology suitability for the game.

The result is a game called “SnakeWar”, which was programmed using different frameworks and libraries. Most of the program is written in JavaScript programming language and WebSocket technology was used for making continues connection between user and the host device. Completed application allows up to ten players in the same room to simultaneously play a classic game of “Snake”. From analysis of the different gamepad user interfaces, the author chose to use the swipe move.

The author came to conclusion, that chosen technologies were suitable for implementing the game and sees a great potential in WebSocket technology. Application fast-paced nature tested the abilities of the WebSocket. The author did not observe any major latency problems while using a decent internet connection. Game creator is confident to say, that more complex and higher-level graphical games can be programmed using the same technologies.

In addition to games, this technique can also be used in commercial areas. The author considers advertising to be the field that can make use of this kind of technology.

Finally, the author wants to highlight the fact that testing the game among the users is very important part of the game development. It is one of the best ways to find problems and bottlenecks of the program, which programmer will not initially notice. Thanks to the user testing many bugs were removed and the user interface (UX) was improved.

Having examined the following topics and programmed the game, the author fulfilled all of his previously set requirements. Further development opportunities may include improving the game performance by adding some sort of optimization algorithm to the code. Additionally, audio and higher-level graphics can be added.

Kasutatud kirjandus

[1] „QR-kood,“ [Võrgumaterjal]. Available: https://et.wikipedia.org/wiki/QR-kood. [Kasutatud 09 01 2016].

[2] „Kasutajakogemus,“ [Võrgumaterjal]. Available: https://et.wikipedia.org/wiki/Kasutajakogemus. [Kasutatud 09 01 2016].

[3] Wikipedia, „WebSocket,“ [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/WebSocket. [Kasutatud 02 01 2016].

[4] J. Freeman, „9 killer uses for WebSockets,“ [Võrgumaterjal]. Available: http://www.infoworld.com/article/2609720/application-development/9-killer-uses-for-websockets.html. [Kasutatud 03 01 2016].

[5] „Multeor,“ [Võrgumaterjal]. Available: http://multeor.com/about/. [Kasutatud 06 01 2016].

[6] Google, „Chrome Super Sync Sports,“ [Võrgumaterjal]. Available: https://chrome.com/supersyncsports/#/en-US/about. [Kasutatud 06 01 2016].

[7] T. Ltd, „Duck Shoot Game,“ [Võrgumaterjal]. Available: https://www.chromeexperiments.com/experiment/websockets-duckshoot-game. [Kasutatud 06 01 2016].

[8] I. Melnikov ja I. Fette, „The Websocket Protocol,“ [Võrgumaterjal]. Available: https://tools.ietf.org/html/rfc6455#section-1.2. [Kasutatud 04 01 2016].

[9] M. Ubl ja E. Kitamura, „The Problem: Low Latency Client-Server and Server-Client Connections,“ 20 10 2010. [Võrgumaterjal]. Available: http://www.html5rocks.com/en/tutorials/websockets/basics/. [Kasutatud 05 01 2016].

[10] „Socket.io,“ [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/Socket.IO. [Kasutatud 05 01 2016].

[11] „Socket.io,“ [Võrgumaterjal]. Available: http://socket.io/. [Kasutatud 05 01 2016].

[12] „About Node.js,“ [Võrgumaterjal]. Available: https://nodejs.org/en/about/. [Kasutatud 08 01 2016].

[13] „Node.js,“ [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/Node.js. [Kasutatud 07 01 2016].

[14] „What is NPM?,“ [Võrgumaterjal]. Available: https://docs.npmjs.com/getting-started/what-is-npm. [Kasutatud 07 01 2016].

[15] „Express,“ [Võrgumaterjal]. Available: http://expressjs.com/. [Kasutatud 01 01 2016].

[16] [Võrgumaterjal]. Available: http://www.tutorialspoint.com/nodejs/nodejs_express_framework.htm. [Kasutatud 02 01 2016].

[17] „Bootstrap,“ [Võrgumaterjal]. Available: http://getbootstrap.com/. [Kasutatud 04 01 2016].

[18] „JavaScript,“ [Võrgumaterjal]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript. [Kasutatud 01 01 2016].

[19] „PhoneGap - Kiirendusmõõtur,“ [Võrgumaterjal]. Available:

http://metshein.com/index.php/veeb/phonegap-huebriidrakendused/769-14-phonegap-kiirendusmootur. [Kasutatud 03 01 2016].

[20] „Cervisia käsiraamat - Peatükk 1. Sissejuhatus,“ [Võrgumaterjal]. Available: https://docs.kde.org/trunk4/et/kdesdk/cervisia/introduction.html. [Kasutatud 06 01 2016].

[21] „Socket.io - Server API,“ [Võrgumaterjal]. Available: http://socket.io/docs/server-api/. [Kasutatud 08 01 2016].