TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Mihkel Riik 193967IADB

DEMUT – Videomängude tugirakendus suhtlemisraskustega inimestele

Bakalaureusetöö

Juhendaja: Kaido Kikkas
Tehnikateaduste doktor

Tallinn 2023

Autorideklaratsioon

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

Autor: Mihkel Riik

30.11.2022

Annotatsioon

Käesoleva bakalaureusetöö eesmärk on arendada suhtlemisraskustega inimestele tugirakendus, mis võimaldaks kommunikatsiooni rehabilitatsiooni videomängude mängimiseks ning mis leevendaks takistatud suhtlusest tulenevate negatiivsete vaimsete häirete mõju.

Tugirakendus on sihitud isikutele, kellel ei ole tekkinud suhtlusraskused kuulmishäirete tagajärjel, vaid kellel on raskusi häälelise suhtlusega, kuid on võimelised kuulma. Rakendus hakkab töötama „tekst hääleks“-põhimõttel ja simuleeritud füüsilise mikrofoni kasutuse põhimõttel, mitte hääl tekstiks põhimõttel.

Lõpprakendus annab lõppkasutajatele võimaluse kõrgendada tekstipõhist suhtlust nii mängude siseselt ning ka VoIP-tarkvarades, mis toetavad mängude suhtlust mängude väliselt.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 36 leheküljel, 7 peatükki, 18 joonist, 4 tabelit.

Abstract

DEMUT – a Videogame Support Application for People with Communication Difficulties

The aim of this bachelor's thesis is to develop a support application for people with communication difficulties, which would offer communication rehabilitation for playing video games and which would alleviate the effects of negative mental disorders resulting from hindered communication.

The support application is aimed at individuals who do not have communication difficulties as a result of hearing loss, but who have difficulties with vocal communication and who are able to still hear. The application will work on the basis of text-to-speech and simulated use of a physical microphone, not on the basis of voice-to-text.

The end application gives its end users the ability to enhance text-based communication both within games and in VoIP software that supports game communication outside of video games.

The thesis is in Estoniain and contains 36 pages of text, 7 chapters, 18 figures, 4 tables.

Lühendite ja mõistete sõnastik

API	*Application Programming Interface*, Rakendustarkvara liides
ADPCM	*Adaptive Differential Pulse Code Modulation structure format*, Kadudega tihendatud heli faili struktuuri formaat
COM	*Component Object Model*, Komponentobjektide mudel
DLL	*Dynamic Link library*, Rakendustega kompaktseks korduskasutuseks dünaamiliselt lingitav teek
Git	Versioonihaldustööriist
HID	*Human Interface Device*, Arvutiseade, mis võtab inimestelt sisendi ja tagastab inimestele väljundi
hook	Mehhanism, mille abil rakendus saab pealt kuulata sündmusi, nagu sõnumid, hiiretoimingud ja klahvivajutused
HTML	*HyperText Markup Language*, Hüperteksti märgistuskeel
IDE	*Integrated Development Environment*, Integreeritud programmeerimiskeskkond
IPC	*Inter-process-communication*, Protsessidevaheline suhtlus
MMDevice	*Multimedia Device*, Multimeediumiseade
MVVM	*Model-view-viewmodel*, Arhitektuuriline mall, mis eraldab kasutajaliidese graafilise nähtava osa rakenduse äriloogikast
NPM	*Node package manager*, Node teekide haldur
PCM	*Pulse-code modulation*, Impulsskoodmodulatsioon
RAWINPUTDEVICE	Andmestruktuur, mis defineerib töötlemata sisendseadmete informatsiooni struktuuri
RIFF	*Resource Interchange File Format*, Heli failide andmestruktuuri standard. Hoiustab andmeid viidetega andmetükkides
SAPI	*Microsoft Speech API*, Microsoft Windows kõnerakenduste programmeerimisliides
TTS	*Text-To-Speech*. Tekst kõneks
WAVE/WAV	*Waveform Audio File Format*, Pakkimata helifaili salvestusvorming
WAVEFORMATEX	Microsoft Windows kasutatav WAVE failide programmaatiline andmestruktuur

WAVEFORMAT-EXTENSIBLE	Laiendatud WAVEFORMATEX andmestruktuur. Rohkem helikanaleid või kõrgem helikvaliteet
Win32 API	*Windows application programming interface*, Microsoft Windows rakendustarkvara liides. Lubab kasutada iga versiooni jaoks ainulaadsed funktsioone ja võimalusi

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Mängude mängimine maailmas on populaarne hobi, mis ei sõltu vanusest. Mänge mängib maailmas umbes 3,24 miljardit inimest ning suur osa nendest mängijatest mängib mänge, milles on olemas mitme valikuline kommunikatsioon [1]. Mitme valikuline kommunikatsioon koosneb peamiselt tekstipõhilisest suhtlusest, häälepõhisest suhtlusest või ette määratud suhtluskäskudest.

On tehtud mitmeid uuringuid, et määrata ära seostatus vaimsete häirete tekkimisel, nagu depressioon, ärevus ja madal enesehinnang ning kommunikatsiooni raskuste vahel [2]. Nii nagu mängimine kui hobi, võivad esineda vaimsed häired igal vanusegrupil, kuid varieeruva võimaluse ja mõjuga [3]. Uuringute põhjal võib järeldada, et suhtlus on tähtis osa vaimsele tervise heaolule.

Suhtlus on väga tähtis kiiretempolistes ühismängudes, kus informatsiooni kiire ja lakooniline vahetus on vajalik sirgjooneliseks ja tõhusaks mängukogemuseks. Kuigi suhtluseks mängudes on mitmeid valikuid, ülistatakse mängijate poolt häälelist suhtlusviisi selle efektiivsuse tõttu [4].

Nagu eelnevalt mainitud, on suhtlus tähtis vaimsele tervisele ning tähtis kiiretempolistes ühismängudes. Nendele faktidele tuginedes on lõputöö eesmärk luua arendatav tugirakendus suhtlusraskustega lõppkasutajatele, kes tänu rehabilitatsiooni võimalusele saavad oma suhtlust teistega muuta efektiivsemaks.

Eelkõige annab rakendus võimaluse muuta mängusiseselt tipitud tekst hääleliseks sünteesitud kõneks. Sekundaarseks võimaluseks on võimaldada siduda kindlaid klaviatuuri/hiire liigutusi eelnevalt ära määratud WAV helifailide tagasimängimiseks. Mõlemad valikud suunavad heli sisendina rakendustesse, kus simuleeritakse heli kui füüsilist mikrofoni kasutamist.

Töö esimene osa räägib üldisemast lõputöö arenduse analüüsist. Peatükk kirjeldab ära, milline on lahendatav probleem, kuidas probleemi lahendus on välja mõtestatud ning mida on lahenduse läbi viimiseks vaja silmas pidada. Töö teises osas on ära kirjeldatud

tehnoloogiad, mis on valitud, et saavutada eelnevas peatükis ära määratud lahenduse praktiline vorm. Teine osa kirjeldab, milliste tehnoloogiatega on tegu ning milline on nende otstarve praktilises osa arenduses. Kolmandas osas kirjeldatakse lahti praktilise töö funktsionaalsused, mis panustavad soovitud lahenduse saavutamisesse. Eelviimases osas kompileeritakse ja pakitake kokku lõpplahendus, et luua rakenduse paigaldaja, mille abil lihtsustada rakenduse paigaldamist erinevatesse süsteemidesse. Viimaseks osaks jääb lõpliku rakenduse testimine ning testimise järgne tulemuste analüüs ja järeldused.

2 Arenduse eelanalüüs

Järgnevas peatükis tuuakse välja lahendatava probleemi, sarnased rakendused ning seatakse paika funktsionaalsused.

2.1 Lahendatav probleem

Paljudes mängudes ei ole lisatud TTS (*text-to-speech*) võimalusi.

Kaasmängijatel on kõrge kontsentratsiooniga olukordades raske panna tähele teksti akent selle olemasolul või ette määratud suhtluskäsklusi. See probleem on ka mänguvälistel suhtlusrakendustel nagu Discord – ei ole võimalik lugeda kõrvalist teksti teiste tegevuste ajal.

Kui suhtlusraskustega isik annab endast parima teistega suhtlemisel, et jõuda tiimipõhises mängus parima lõpptulemusi

Kui suhtlusraskustega isik üritab anda endast parim, et suhelda tiimipõhises mängus kaasmängijatega parema lõpptulemuse saavutamiseks, võib tekkida reaalne situatsioon, kus suhtlus jääb ühepoolseks. Sellises olukorras võib sihtgrupi esindaja seda tõlgendada kui isiku olemasolu ignoreerimiseks kaasmängijate poolt. Tekstipõhist kommunikatsiooni kasutades, võib siginema hakata või süveneda olemasolevad vaimsed häired. See on eelkõige tingitud inimese hääle kaotamisest asjakohaste mõtete ja tähelepanude esitamiseks.

2.2 Lahenduse kitsendused

Arendusele eelnev analüüs näitas, et töö maht on suurem kui selle tegemiseks antud võimalik ajaline maht. Kuna autorile on suurem osa kasutatud tehnoloogiatest võõrad, on vaja arenduse käigus arvestada ka õppe ja katsetuste perioodidega. Et vähendada töö mahukust ja koormust on võetud kasutusele kolm kitsendust.

Esmalt tehakse otsus keskenduda Microsoft Windows platvormile. See lubab võtta maksimumi Microsofti dokumentatsiooni kogumist ning leevendab ajakulu tänu paralleelse multiplatvormse arenduse loobumisele.

Teiseks tehakse otsus kasutada kolmanda osapoole lahendust, et viia rakenduse väljund valitud rakendustesse. Analüüsi käigus nähti, et virtuaalse seadme arendamine võib mahu poolest olla lõputöö raames arendatud lahendusega samaväärne või isegi suurem. Arendusel on kasutatud kolmanda osapoole lahendust nimega VB-CABLE, mis kasutab annetustarkvara litsentsi.

Kolmandaks tehakse otsus kasutada rakenduse funktsionaalsuse saavutamiseks lahendusi, mis hoiduvad mängude otsesest graafilisest manipulatsioonist ja sisendite simuleerimisest. Analüüsi käigus nähti, et tänapäeva mängudes on olemas sohivastased tehnoloogiad, mis takistavad lihtsamate lahenduste kasutamist. Et aga lahendada probleeme, mis võivad lõputöö raames esineda, oleks vaja autoril kasutada ebaeetilisi võtteid, et sohivastastest tehnoloogiatest läbi murda. Sellisel juhul on suur võimalus lõppkasutajatel saada mängimiskeelu nendes mängudes, kus lõputöö lõpplahendust kasutada.

2.3 Lahendus

Probleemi lahenduseks on välja töötatud rakenduse idee, mille eesmärk on anda suhtlusraskustega isikutele rohkem valikuid ja kontrolli oma suhtluse üle mängudes ja mängudega seostuvates mänguvälistes suhtlusrakendustes.

Rakenduse peamine ülesanne on simuleerida tavapärast mikrofoni kasutust mängudes. Selle läbi viimiseks hakkab rakendus kasutama spetsiifilisi helifaile, mida kasutaja saab ise määrata. Helifailide andmed loeb rakendus mällu ning mängib seda heli tagasi kui mikrofoni sisend tahetud rakendustes. Selleks, et oleks võimalik mängida heli kui mikrofoni sisendit, oli vaja kasutusele võtta VB-CABLE nimeline annetustarkvara. VB-CABLE hakkab võtma vastu sisse loetud helifaili andmeid ning hakkab seda väljastama igas seadmes, kus VB-CABLE virtuaalse seadme väljund on kasutusel.

Lisaks helifailide kasutusele hakkab rakendus pakkuma rakendusi katvat nurgas olevat teksti sisendi akent. Aknasse tipitud kirjed sünteesitakse otse robotisarnaseks hääleks

ning mängitakse sarnaselt helifailidele tagasi. See lahendab mängudes tekst kõneks funktsionaalsuse puudulikkust.

Helifailide tagasimängimise valik hakkab käima läbi hiire ringvaliku süsteemi. Kasutades hiire rulliknupu alla vajutust on võimalik järgida, kus rulliknuppu alla vajutati ning kus rulliknupu alla vajutus lahti lasti. See võimaldab arvutada sektori, kus hiir seisma jäi ning millise ringvaliku kasutaja tegi. Esialgselt toetab see lahendus eelkõige mänge, kus on ettemääratud suhtlusvalikut juba olemas koos visuaalse graafilise ringvaliku süsteemiga. Teistes mängudes jääb hiirvalik alles, kuid sõltub kasutaja mälu ja harjumusel, et valikuid teha ilma visuaalse abita.

2.4 Sarnased eksisteerivad rakendused

Osaliselt sarnased funktsionaalsused:

- iMyFone MagicMic
- EXP Soundboard
- Jingle Palette
- Soundboard
- Speechify
- DCSB
- GOXLR

Eelnimetatud on osa olemasolevatest rakendustest, mis jagavad osalist sarnast funktsionaalsust. Ükski rakendus ei täida tervenisti mängudepõhilist kommunikatsiooni rehabilitatsiooni võimaluse rolli. Lõputöö raames valmiv rakendus keskendub nii tekst kõneks funktsionaalsusele ning ka *soundboard* seadmetüüpide funktsionaalsusele. Samuti on lõputöö lahendus tasuta ning pürgib erinevalt eelnimetatud rakendustest parema kasutajaliidese disaini ja mugavuse poole.

2.5 Funktsionaalsed nõuded

Lühikesed laused, mis kirjeldavad ära mida rakendus peab suuteline olema tegema lõppkasutaja käes [5].

Loetelu arendatava rakenduse funktsionaalsetest nõuetest:

- Rakenduse kasutatavad helifailid peavad olema kasutaja poolt lisatavad.
- Rakendus peab suutma luua helifaile, kasutades kasutaja teksti sisendit.
- Kasutajad peavad saama sätestada hiire ringvalikus kasutatavaid helifaile.
- Kasutajad peavad saama lisada helifaile igas koguses.
- Kasutajad peavad saama kiirklahve sätestada.
- Kasutajad peavad saama valida heliväljundi seadet.

2.6 Mittefunktsionaalsed nõuded

Laused, mis kirjeldavad ära kriteeriumid ja kvaliteedi atribuudid, mida rakenduse funktsionaalsused peavad jälgima [6].

Loetelu arendatava rakenduse funktsionaalsetest nõuetest:

- Rakenduse kasutatav muutmälu ei piira arvuti tavakasutust.
- Rakenduse kasutatav protsessori kasutus ei piira arvuti tavakasutust.
- Rakendust on lihtne paigaldada ja kasutada – hallatavus.
- Rakendus töötab järjepidevalt ilma tõrgeteta – töökindlus.
- Rakenduse tegevused toimuvad viivitamatult.
- Rakendus töötab ainult Microsoft Windows platvormil.
- Kasutab multiplatvormseid lahendusi, et tulevikus areneda multiplatvormseks rakendusek

3 Arendustehnoloogiad

Järgnevas peatükis tuuakse välja arenduse käigus kasutatud tehnoloogiad ning nende kirjeldused.

3.1 Integreeritud programmeerimiskeskkond

Arendusel kasutati mitut IDE-d (*Integrated development environment*) . Rakenduse ja kasutajaliidese arenduseks kasutati Visual Studio Code keskkonda. C++ moodulite arenduseks kasutati nii Visual Studio 2022 kui ka Visual Studio Code keskkonda.

Visual Studio 2022 keskkond võimaldas arendada C++ funktsionaalsust, andes sellele vajalikud arendustööriistad, mis andsid märku vigadest, toetas paremat vigade otsimist ning pakkus suures osas teeke kohe pärast keskkonna ülesse seadmist.

Visual Studio Code keskkond oli ideaalne kasutajaliidese ja rakenduse arenduseks. Visual Studio Code võimaldas lisada arenduskeskkonna lisasid, mis lihtsustavad arendaja tööd läbi tuhandete lisatavate tööriistade ja isikupärastamise. Visual Studio Code oli kasutusel ka C++ arendusel, et liidestada Node-API teek JavaScriptiga. See andis võimaluse koheselt ehitada C++ koodist vajalikud Node keskkonna failid ning nende töökäiku testida.

3.2 Microsoft Windows

Rakendus on arendatud Microsoft Windows operatsioonisüsteemi keskkonnas ning on mõeldud kasutuseks Microsoft Windows operatsioonisüsteemidega arvutites. Microsofti poolt on loodud Microsoft Learn keskkond, mis koosneb Microsoft Windows API dokumentatsioonidest. Eelnimetatud dokumentatsioonid annavad palju abistavat informatsiooni, et valitud operatsioonisüsteemi keskkonnas arendada töölauarakendus, mis suudab suhelda madala taseme süsteemi funktsionaalsustega.

Microsoft Windowsil on olemas API-d, mis võimaldavad luua tekstist robot häälelist heliandmeid, kuulata pealt klaviatuuri ja hiire kasutust, loendada arvutiga ühendatud seadmeid ning neid kasutada ja rohkemgi.

3.3 Git ja GitHub

Git on tasuta ja avatud lähtekoodiga hajutatud versioonihaldussüsteem. Git lubab juurutada projekti eraldiseisvateks paralleelseteks oksadeks, mis üksteisest ei sõltu. Nii on võimalik jätta töötav peamine oks muutumatuks ja teha kõrvaline oks, kus saab muudatusi testida ja paigaldada, mida hiljem mestida pea oksaga. Git muudab arendamise turvalisemaks ja kiiremaks, kuna muudatusi on võimalik vajadusel tagasi võtta. [7]

Joonis 1. Illustreeriv pilt Git oksadest ja mestimistest

Projektid lisati versioonihalduseks GitHub koodihoidlatesse. GitHub võimaldab kasutada Git-i, et laadida pilve koodibaasi muudatused. Koodihoidlatest on koodibaase lihtne alla laadida ükskõik millistesse seadmetesse, mis seda võimaldavad. Samuti saab koodihoidlatest lihtsal graafilisel kujul ülevaate eraldiseisvatest oksadest koos nende muudatustega. Koodihoidlas olev koodibaas on ainuõige tõe allikas, mis omab kõige uuemaid ja tähtsamaid muudatusi. Kohalikus masinas olev koodibaas on eraldiseisev pilve omast. Et neid sünkroniseerida, on vaja kas kohalikust masinast muudatused ülesse lükata (*push*) või koodihoidlast kohalikku masinasse alla tõmmata (*pull*). [8]

3.4 C++

Madalatasemeliste keelte valikutes oli nii C, C++ ja Rust. Autor valis projekti läbiviimiseks C++ keele, millega on autoril olnud varasem kogemus. Samuti on C++ keel, mida Microsoft Windows suuresti toetab, kasutades dokumentatsioonides C++ näiteid ning pakkudes Win32 API-t, mis lubab otsest ligipääsu Microsoft Windows funktsioonidele ja sellega ühendatud riistvarale. Win32 API on programmeerimise vahelüli, mida kasutades on võimalik arendada C++ funktsionaalsust, mis on võimeline suhtlema operatsioonisüsteemi C keele teekidega. [9]

Projekt hõlmab suuremas ulatuses heli tehnoloogiatega tegelemist. See tähendab, et C++ on tähtis osa, et heli tagasimängimine ja sünteesimine oleks jõudluse poolest lõppkasutajale otsekohene.

C++ on kompileeritud keel, mis tähendab, et valmis kirjutatud rakenduse kood kompileeritakse masinkoodiks, mida on arvutil väga lihtne käitada. Kuna koodi tõlgendatakse läbi kompileerimise ainult ühe korra, on lõpprakendust võimalik lihtsasti ilma lisa tegevuste sisse laadida ja käitada. [10]

3.5 Electron

Electron on tasuta avatud lähtekoodiga tarkvararaamistik, mille on välja töötanud ja hooldab GitHub. Raamistik on loodud töölauarakenduste loomiseks, kasutades veebitehnoloogiaid nagu HTML-i, CSS-i ja JavaScript-i(Typescript, erinevad JavaScripti raamistikud). [11] Kasutatud veebitehnoloogiad visualiseeritakse Chromium brauseri mootoriga Node käituskeskkonnas. See tähendab, et Electron pakib kaasa brauseri mootori, mis tagab selle, et kasutatud veebitehnoloogiad kindlasti tööle jääks. Electron tagab töökindlust läbi kaasa pakitud Chromium brauseri versiooni, mis ei uuene, isegi kui Google poolt arendatav Google Chrome brauser uueneb. Samuti ei ole vaja muretseda, kas veebitehnoloogiatega kirjutatud kasutajaliides töötab erinevate veebibrauserite peal nagu oodatud. [12]

Electron kasutab erinevaid API-sid, mis toetab integreerimist Node.js teenustega ja turvalist Electron protsesside vahelist suhtlusmoodulit. Node.js keskkonna olemasolu on projektis äärmuslikult tähtis, kuna Node.js võimaldab JavaScripti liidestada madalataseme keelega. Selle tulemusel on võimalik JavaScriptis kasutada suurtes

kogustes olemasolevaid C/C++ teeke, mis võimaldab JavaScriptil kasutada funktsionaalsust, milleks JavaScript ise suuteline ei ole. [13]

Electron alternatiivideks on väiksema mälukasutusega ning uuemad Tauri ja NeutralinoJS raamistikud. Töö jaoks valiti valikutest Electron. Autoril on olemas varasem kogemus Electron raamistikuga ning Electron on valikutest ainuke, mis pakib kaasa Chromium brauseri. Samuti on Electroni vanuse tõttu olemas suurem dokumentatsiooni kogum ja teekide arv.

3.6 Node

Node on avatud lähtekoodiga multiplatvormne JavaScripti käituskeskkond ja teek veebirakenduste käitamiseks väljaspool kasutaja brauserit. Node-i kasutatakse serveripoolsete veebirakenduste loomiseks ja see sobib suurepäraselt andmemahukate rakenduste jaoks. Node kasutab asünkroonset juhitavate sündmuste arhitektuuri, mis tähendab, et Node on ühe lõimeline keskkond. Et Node keskkond suudaks teenindada mitmeid päringuid ja funktsionaalsuste kutseid korraga, hoitakse funktsioonide ohjureid kogumis, mida hiljem kutsutakse. [14]

Ohjurid pannakse kirja kui Node keskkonnas on kasutatud funktsioone, mis ei blokeeri ülejäänud koodi käsitlust. Kui esialgselt kutsutud funktsioon saab valmis, saadetakse selle poolt valmis saamise sõnum, mis pannakse kollektsiooni kirja. Seda kollektsiooni hakkab tsükliga läbi käima kollektsiooni jälgija. Kui kollektsiooni tekib sõnum, siis jälgija kutsub välja sellega seotud ohjuri, mis esialgse koodi kutsumise punktist lõpetab koodi jooksutamist. Sellel põhimõttel töötab ka Electron IPC, millega kasutajaliidese protsess suhtleb põhiprotsessis oleva äriloogikaga. Projektis on Electron IPC tähtis osa, millega JavaScript ja C++ omavahel suhtlema panna. [15]

3.7 Node-API

Node-API on tööriistakomplekt, mis toimib vahendajana C/C++ koodi ja Node JavaScripti mootori vahel. See võimaldab C/C++ koodil JavaScripti objektidele juurde pääseda, neid luua ja neid muuta, nagu oleksid need JavaScripti koodiga loodud. Node-API on sisse ehitatud Node'i 8.0.0 ja uuematesse versioonidesse ning ei vaja täiendavat paigaldamist. [13]

Et kasutada Node-API C/C++ projektis, on vaja selleks importida „node_api.h“ nimeline C/C++ päise (*header*) fail. Järgnevalt on vaja hakata kasutama napi päise failist andmetüüpe ja korrektseid funktsioone, et luua ja manipuleerida JavaScripti objekte kasutades Node keskkonda, mis saadakse kui C/C++ mooduli funktsionaalsust kutsutakse JavaScripti koodist, Node keskkonnast. [16]

3.8 Node-GYP

Node-gyp võimaldab konstrueerida C/C++ koodist kompileeritud Node laiendiga faili, mida suudab Node keskkond importida ja kasutada kui tavalist JavaScripti koodi. Node-GYP võimaldab luua C/C++ keeles projektiks vajalikke heli funktsionaalsust, mida saab kasutada otse JavaScriptis. [17]

3.9 Vue

Vue on JavaScripti raamistik kasutajaliideste loomiseks. See tugineb standardsele HTML-ile, CSS-ile ja JavaScriptile ning pakub deklaratiivset ja komponendipõhist programmeerimismudelit, mis aitab kasutajaliideseid tõhusalt arendada, olgu need lihtsad või keerulised. [18]

Rakenduse kasutajaliidese disain sai loodud Vue raamistikuga. Kasutajaliidese loomise valikute hulgas olid Angular, React, Vue või tavaline JavaScript. Kuna autoril oli kõige rohkem kogemust Vue raamistikuga ning Vue pakkus efektiivset ja intuitiivset arenduskogemust, siis valiti see rakenduse visualiseerimiseks.

3.10 Vite

Vite on uut tüüpi kasutajaliideste ehitustööriist, mis parandab oluliselt kasutajaliideste arenduskogemust. Alternatiive on olemas, millest kõige tuntum on Webpack. Kuigi Webpack on eksisteerinud mitmeid aastaid, mille tõttu on sellel ka rohkem toetust ja dokumentatsiooni, jääb ta mugavuse ja jõudluse poolest kõvasti hätta. [19]

Eelmainitud ehitustööriist pakub arendusserverit, mis teenindab projekti lähtefaile üle ES-moodulite, millel on rikkalikud sisse ehitatud funktsioonid ja kiire HMR(*Hot Module Replacement*). Vite kasutusel olevad lisandite API lisab laiemat tugevalt tüübitud

TypeScript toetust Vue raamistikule, mis tagab koodi parema hallatavuse ja töökindluse. [20]

3.11 Inno Setup

Inno Setup on tasuta, skriptipõhine paigaldussüsteem, mis on kirjutatud Delphi programmeerimiskeeles. Inno Setup võimaldab kaasa pakkida paigalduseks vajalikke faile, mida on võimalik läbi kirjutatud skripti kasutaja arvutisse paigaldada. [21] Inno Setup on olemas olnud alates 1997 aastast, mis tähendab, et selle funktsionaalsus ja töö võime on tasuliste paigaldussüsteemidega sarnasel tasemel [22].

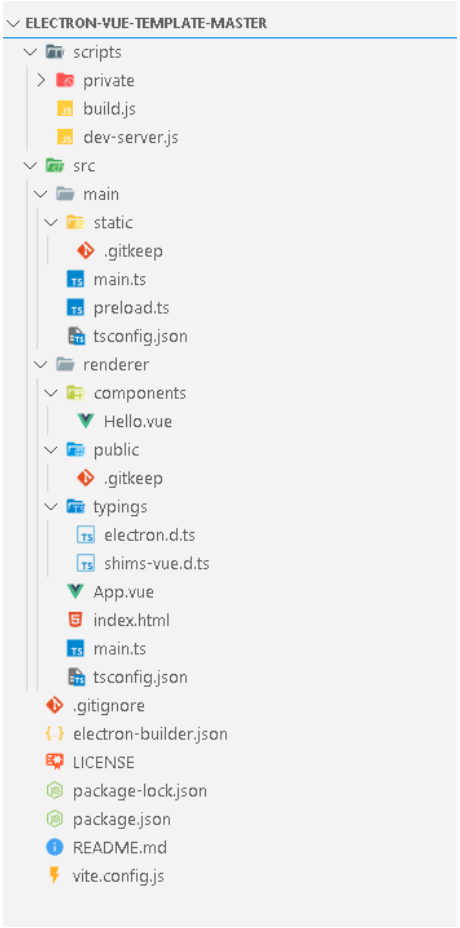
Kuna Inno Setup ja Delphi programmeerimiskeel on autorile tuttav ning autor on eelnimetatud tehnoloogiatega varem paigaldusskripte loonud, valiti Inno Setup paigaldusüsteem rakenduse paigaldaja loomiseks.

4 Arendus

Järgnevas peatükis räägitakse detailsemalt, kuidas erinevad rakenduse osad arendati.

4.1 Töölauarakenduse mall

Rakenduse arenduseks võeti aluseks „electron-vue-template“ vabavaraline mall, mis pakub valmis sätestatud Electron, Vue ja Vite arenduskeskkonda. Mallis on olemas HMR võimalusega arendusserver ning valmis struktureeritud failisüsteem. [23]



Joonis 2. Valmis struktureeritud Vue + Electron failisüsteemi.

4.2 Kasutajaliides

Kasutajaliidese kujunduse ja struktuuri kirjeldused ning selgitused.

4.2.1 Kujundus

Kasutajaliidese kujunduse eeskujuks võeti populaarsemad Electron põhjaga rakendused, nagu Visual Studio Code, Spotify, Steam.

Rakenduse värvi temaatikaks valiti varieeruva heledusega tume taust, mille peal jääb silma valget värvi tekst. Kujunduse kvaliteedi kõrgemaks tõstmiseks ning tähtsamate elementide esile kutsumiseks, otsustati kasutada helendavat oranži värvi. Oranž värv sobis kasutuseks ümber blokk elementide ja aktiivsete nuppude, et neid paremini eraldada ja esile kutsuda.

Kujundus koosneb staatilisest päis elemendist ja vasakupoolsest valikute rippmenüüst. Päisest allpool olev parempoolne element kuvab erinevaid lehti, mida lõppkasutaja saab sirvida kasutades rippmenüüd. Rippmenüü valikud muudavad paremapoolse elemendi sisu vastavalt valitud lehe valikule.

4.2.2 Struktuur

Valitud veebitehnoloogia raamistik Vue võimaldab kasutada MVVM(model-view-viewmodel) arhitektuurilist mustrit, et muuta rakenduse kujundus dünaamiliseks ja interaktiivseks. Muster seab üles andmete mõlema suunalise suhtluse, mis tähendab, et visualiseeritud HTML funktsionaalsus suudab muuta programmaatilisel kujul andmeid ning programmaatiliselt muudetud andmed suudavad muuta hüpertekst kuva, visualiseerides uuendatud programmaatilisi andmeid, kui uuendatud HTML-i. Kõik see on reaktiivne ning andmete muutmisel ja uuendamisel visualiseeritakse leht osaliselt uuesti, et muudatusi näidata ning jõudlust parandada. [24]

Vue kasutab SPA(single page application) mudelit, mis tähendab, et kujundust on võimalik lahku lüüa väiksemateks komponentideks, mida on võimalik välja vahetada või taaskasutada. See parandab kõvasti lähtekoodi loetavust, hallatavust ja edasi arendamist. Samuti muudab see kasutaja kasutuskogemust meeldivamaks, kuna rakenduse kasutus voog on sujuvam. [25]

Komponentide kooslusi hakati nimetama vaadeteks. Vaade on väiksematest komponentidest koostatud kindla funktsionaalsuse ja temaatikaga struktuurid, mis midagi spetsiifilisemat teevad. Vaadete vahel liikumine käis läbi rippmenüü valikutega, mille funktsionaalsusega tegeleb Vue Router moodul. Rippmenüü nupud käituvad sarnaselt hüperlinkidega, mis suunavad teisele dokumendile.

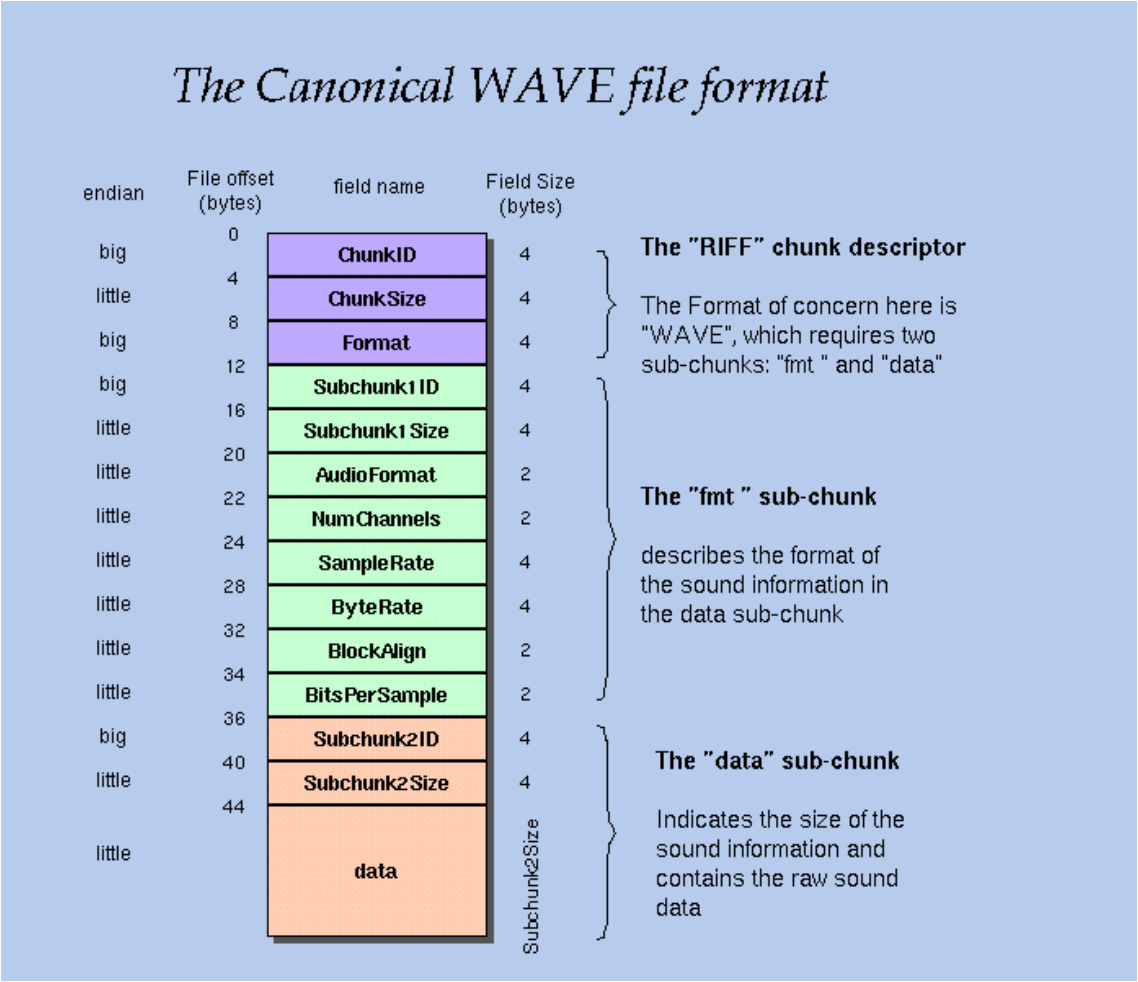
Vue Router moodul on eraldiseisev moodul, mida on võimalik Vue raamistikule lisada läbi NPM moodulihalduri. Vue Router võimaldab lisada rakendusele lõuendi, millele Vue Router hakkab laadima erinevaid kuvasid, mis on eelnevalt ära kaardistatud. [26]

4.3 WAV failide lugemine

Helifailide kasutamiseks võeti kasutusele WAVE faili formaat. WAVE formaat seostub väga hästi Microsoft Windows platvormiga ning lihtsustab Win32 API funktsionaalsuste kasutamist. [27]

WAVE formaat allub Microsoft RIFF (*Resource Interchange File Format*) spetsifikatsioonile. RIFF spetsifikatsiooni järgi koosnevad RIFF helifaili tüübid kolmest osast: “RIFF“, „fmt“ ja „data“. „RIFF“ osa kirjeldab ära helifaili formaadi. „fmt“ osa sisaldab helifaili vormingu andmeid. Vorminguteks võib olla WAVEFORMATEX, WAVEFORMATEXTENSIBLE või ADPCMWAVEFORMAT. Vormingu andmeteks on üldiselt heliformaat, helikanalite arv, diskreetimissagedus, jne. „data“ osa RIFF-st sisaldab töötlemata ja tihendamata heli andmeid. WAVE formaat kasutab heliandmete formaadiks PCM-i (*pulse code modulation*), mis tähendab, et heli on ujukomade massiivi kujul. [27][28]

Helifailid laetakse muutmällu ning töötlemata baidid loetakse massiivi. Massiivist on võimalik lugeda mitmete baitide kaupa andmeid, et neid töödelda vastavateks andmetüüpideks. Nagu Joonis 3 näha, on igal andme real olemas baidi suurus. Nende baitide suuruste järgi liiguti loetud baitide massiivis edasi ning muudeti vastavalt kas sõneks, ujukomaks või numbriks.



Joonis 3. Standard WAVE failiformaadi ehitus.

Heli mängimiseks oli vaja läbi töödelda terves suuruses „data“ RIFF blokk. Projektis käideldud WAVE formaadi failid olid PCM 16 kujul, ehk iga ujukoma oli suuruses 16 bitti/2 baiti. See tähendas, et massiivis liiguti 2 indeksi koha kaupa edasi, muutes igat 16 bitti ujukoma arvuks. [27]

Andmeblokid võivad olla miljonites baitides pikad, olenevalt helifailide ajalisest pikkusest. See tähendab, et tavaline massiiv ei suudaks efektiivselt hoida tervet andmeblokki korraga mälus. Lahenduseks valis autor vektor andmetüübi kasutuse. Vektor andmetüüp on dünaamilise mäluga ning suudab mälus hoida suurtes kogustes andmeid. Vektor on taga taustal automaatne massiivide haldaja, mis suudab efektiivselt hallata massiivide muutmist. Kasutades vektorit on võimalik laadida mällu kuni 4 gigabaidiseid helifaile. Selle lahenduse puhul peab meeles pidama seda, et sel puhul kasutatakse suurtes kogustes arvutis olevat muutmälu ning helifaile tuleks piirata

väiksemate suuruste juurde. Probleemide vältimiseks on rakenduses mõeldud kasutamiseks lühikesi paari sekundilisi heliklippe. [29]

4.4 Heli esitamine C++-ga

Heli esitamisega seotud funktsionaalsuste arendamise kirjeldused, mis on kirjutatud C++ arenduskeeles.

4.4.1 Heli esitamine

Heli esitamine valitud seadmes viidi ellu C++ keeles, kasutades Microsoft Windows Win32 API COM (*Component Object Model*) teenust. Kasutades seda teenust oli võimalik seadmestada operatsioonisüsteemi küljes olevate seadmete loendaja ja heli seadme programmaatiline objekt. Heli seadme objekt andis ligipääsu Microsoft Windowsi heli mootorile, mis tegeleb heli esitamisega ja sellega seotud tegevustega.

Heli andmete kasutuseks loodi klass nimega *MyAudioSource*, mis tegeleb esitatava heli formaadi andmete hoidmisega kujul WAVEFORMATEX, kasutades selleks RIFF struktuuri. Seda on vaja sama klassi puhul esitatava helipuhvri täitmisel, kasutades RIFF struktuuriga salvestatud heliandmete jada. Klass koosnes *init* meetodist, mis initsialiseeri objekti, *SetFormat* meetodist, mis salvestas väljundi heliformaadi ja *LoadData* meetodist, mis laeb objekti siseselt hoitud heliandme jadast ujukoma kujul andmeid väljundis kasutatavasse kindla suurusega puhvrisse. Viimaks omas klass *deconstructor* omadust, mille eesmärk oli vabastada heliandmete talletamiseks vajaminevat mälu kui objekt manuaalselt hävitati tagasimängimise lõpul. [30]

Heli esitamise funktsionaalsust juhtis *PlayAudioStream* nimeline funktsioon, mille eesmärk oli initsialiseerida vastavad COM teenuse objektid, hankida vajalikud andmed ning läbi viia tsükkel, mis pärast heli esitamise algamist kirjutas igas tsükli iteratsioonis pool jagatud kasutuses oleva helimootori puhvri andmed üle, et toimuks järjepidev andmete esitus. Samal ajal kui rakendus kirjutas iteratsiooni käigus ühe poole puhvrist üle, käis helimootoris puhvri teise poole esitlus – samaaegne andmete kirjutamine ja lugemine.

Vajalike andmete ja seadmete ligi pääsemiseks on vaja kasutusele võtta olemasoleva Microsoft Windows API – MMDevice (*Multimedia Device)* API. MMDevice API

võimaldab luua COM instantsi seadmete väärtustikust, kasutades selleks CLSID_MMDeviceEnumerator ja IID_IMMDeviceEnumerator liideseid (*interface*). [30][31]

Loodud väärtustiku objektiga on võimalik valida olemasolevates arvutiga ühendatud heliseadetest spetsiifiline heliseadme objekt. Heliseadme objektiga aktiveeritakse heli kliendi objekt, millega päritakse heliseadme heliformaat ning millega initsialiseeritakse tagasimängimiseks mõeldud sätted. Heliformaat suunatakse eelnevalt mainitud *MyAudioSource* objekti. [30]

Heli mängimiseks kasutatakse heli kliendi objekti edasi, et pärida arvuti operatsioonisüsteemi käest heli ette kandva teenuse, millega on võimalik kätte saada puhvri viide. Päritud puhvri viitest hakkab operatsioonisüsteemi heli mootor heliandmeid järjestikult lugema. Eelnevalt heli mängimise tsükli käivitamisele, laetakse kätte saadud puhver esialgse andmete hulgaga ning arvutatakse puhvri tagasimängimise aeg. [30]

Heli esitamine algab heli kliendi objekti *Start* meetodiga, millele järgneb tsükkel, mis lõppeb, kui puhver on lõplikult tühi või rakenduseväliselt saadetakse peatumise käsk. Tsükkel ootab pool puhvri läbimängimise kestvust ning seejärel arvutatakse puhvri tühja ala suurus. Seejärel päritakse puhvri eelnevalt arvutatud suurusega vaba osa kuhu kirjutatakse uued andmed, millele järgneb päritud puhvri vabastamine, et puhvrit järgmise tsükli iteratsioonis uuesti pärida. [30]

4.4.2 TTS

Tekst kõneks funktsionaalsust kasutab rakendus, et muuta mängusiseselt kirjutatud tekst koheselt esitatavaks robot kõneks. Sarnaselt kasutab rakendus seda funktsionaalsust, et luua WAV heli faile, mida kasutaja saab sätestada hiire ringvaliku kiirklahviks.

Kuna rakenduse töökeskkonnaks on valitud ainuüksi Microsoft Windows operatsioonisüsteem, siis otsustas autor kasutusele võtta Microsoft Windows olemasoleva SAPI (*Microsoft Speech API)*. SAPI töötab hästi koos ülejäänud C++ keeles kirjutatud funktsionaalsusega, jõudlus on suur ja genereeritud hääle kvaliteet on parem kui olemasolevad TTS arenduse teegid.

Vajalike SAPI objektide initaliseerimiseks kasutatakse *CComPtr* klassi, et mugavdada COM objektide käsitlust.

Tekst kõneks sünteesiks on esmalt vajalik luua ISpVoice liidese objekt, mille kaudu on võimalik saavutada sõnest tagasimängitav ujukomade massiiv. ISpVoice objekti hakatakse kasutama koos kasutajaliideses oleva funktsionaalsusega, kus kasutaja teksti sisend saadetakse objektile sünteesiks.

Kuna ISpVoice tavapärane käitumine on kasutada süsteemi heli vaike seadet, on vaja sünteesitud andmete kättesaamiseks luua ISpStream COM objekt, mis sätestatakse ISpVoice objekti väljundiks, kasutades ISpVoice objekti SetOutput meetodit. Sedaviisi on võimalik suunata sünteesitud heliandmed ligipääsetavasse puhvrisse. [32]

Et ISpStream oleks vajalikul kujul, on vaja luua IStream COM objekt, mis pakendab (*wrapper*) ISpStream objekti enda sisse, mis võimaldab muuta sünteesitud andmete kuju projekti kontekstis õigeks formaadiks. Kõigepealt seatakse IStream objektile õige formaat, kasutades selleks *SpConvertStreamFormatEnum* funktsiooni. Selle funktsiooni kaudu on võimalik sätestada IStream objekt WAVE helifaili formaadiks. Seejärel lisatakse ISpStream objektile IStream objekt andmete pakendajaks. Sünteesi lõppedes tuleb siinjuures olla ettevaatlik, kuna puhver kuhu andmed kirjutati, on oma järjelt puhvri lõpu indeksil. See tähendab, et puhvri indeks tuleb viia uuesti selle alguspunkti. [32][33]

Edaspidine töö on sarnane WAV failide lugemisele, kus tõlgendatakse IStream objekti puhvrist andmed üle bait tüüpi massiivi. Bait tüüpi massiivist loetakse kahe baidi kaupa andmeid, mis tõlgendatakse ujukoma arvuks ja lisatakse heliandmete ujukoma tüüpi vektorisse. Kuna helikanaleid on kaks, siis loetakse ja tõlgendatakse ujukomasid kordamööda. Lõpuks kutsutakse heli esitamiseks *PlayAudioStream* funktsiooni.

4.5 Optiliste seadmete kuulamine

Rakenduse põhiline funktsionaalsus tuleneb mikrofoni kasutuse simuleerimisest, mida hakkab käivitama nii hiire ringmenüü kasutus ning ka mängimise ajal kiirklahvi vajutamisest tingitud kirjutamine. Win32 API pakub erinevaid haake võimalusi, mis lubavad pealt kuulata arvuti ja optiliste seadmete informatsiooni vahetust.

Haakimine võimaldab rakenduses igat klahvivajutust või hiire sisendit pealt kuulata. Kasutades haakeprotseduure on võimalik iga kuulatud sõnumi peale mingisugust funktsionaalsust läbi viia. Haakeprotseduur on funktsioon, mis käivitub iga haakest läbi läinud sõnumi peale.

Alternatiiviks on võimalik luua nähtamatu graafiline aken, mille eesmärk on, tänu lokaalse sõnumi tsüklile, kuulata pealt hiire kasutust. Aken luuakse uue lõime (*thread*) sees põhjusega, et sõnumi tsükkel oleks eraldatud põhi ja paralleel lõimedest. Eraldiseisval lõimel on võimalik ilma mürata kuulata funktsionaalsusele tähtsaid sõnumeid ning põhiline lõim võib takistamata edasi joosta.

4.5.1 Hiire kuulamine

Kuna mängud võivad võtta täieliku kontrolli üle hiire liikumise ei ole võimalik kätte saada hiire liikumist nii füüsiliselt ega virtuaalselt. Kui kasutada tavapärast haakimist, et saada töödeldud virtuaalseid hiire andmeid, on hiire koordinaadid alati arvutikuvari keskel. Näitena võib tuua Apex Legends tiitli. Mängu žanriks on kiire tempoga tulistamine, mille tagajärjel lukustab mäng hiire positsiooni arvutikuvari keskele, et kasutaja vaatepunkt oleks alati keskel.

Lahenduseks on võimalik hakata kuulama hiire töötlemata andmeid, ehk andmed otse füüsilise optilise hiire sensorist. Sedaviisi on võimalik saada füüsilist absoluutset hiire liikumist deltade kujul ning hiire nuppude vajutusi numbriliste koodidena.

Selleks on vaja luua uus lõim ja loodud lõime sisse on vaja luua virtuaalne nähtamatu rakenduse aken. Järgmisena on vaja paigaldada akna külge HID (*Human Interface Device*) seade, milleks kasutatakse RAWINPUTDEVICE andmestruktuuri kuju.

```
    contextData->nativeThread = std::thread([]
                                            {
    // Why even bother with WinMain?
    HINSTANCE instance = GetModuleHandle(0);

    // Create message-only window:
    const char *class_name = "SimpleEngine Class";


    window_class.lpfnWndProc = EventHandler;
    window_class.hInstance = instance;
    window_class.lpszClassName = L"SimpleEngineClass";

    if (!RegisterClass(&window_class))
    {
        std::cout << "registering class went wrong" << std::endl;
    }

    window = CreateWindow(L"SimpleEngineClass", L"SimpleEngine", 0, 0, 0, 0, 0, HWND_MESSAGE, 0, 0, 0);

    if (window == nullptr)
    {
        std::cout << "window is nullptr" << std::endl;
        // return -1;
    }
// End of creating window.
```

Joonis 4. Lõime ja programmi graafilise akna loomine ning kasutamine koodis.

RAWINPUTDEVICE struktuuris määratakse ära seadme tüüp, seadme andmete interpreteerimise kuju ning millist seadmete kaardistatud kollektsiooni kasutada. Andmed on määratud kuueteistkümnend arvulisel kujul. [34]

```
// Registering raw input devices
#ifndef HID_USAGE_PAGE_GENERIC
#define HID_USAGE_PAGE_GENERIC ((unsigned short)0x01)
#endif
#ifndef HID_USAGE_GENERIC_MOUSE
#define HID_USAGE_GENERIC_MOUSE ((unsigned short)0x02)
#endif

    // We're configuring just one RAWINPUTDEVICE, the mouse,
    // so it's a single-element array (a pointer).
    RAWINPUTDEVICE rid[1];
    rid[0].usUsagePage = HID_USAGE_PAGE_GENERIC;
    rid[0].usUsage = HID_USAGE_GENERIC_MOUSE;
    rid[0].dwFlags = RIDEV_INPUTSINK;
    rid[0].hwndTarget = window;
    RegisterRawInputDevices(rid, 1, sizeof(rid[0]));
    // End of registering.
```

Joonis 5. Optilise töötlemata seadme registreerimine graafilise akna külge.

Usage ID	Usage Name	*hidusage.h* constant
0x01	Pointer	HID_USAGE_GENERIC_POINTER
0x02	Mouse	HID_USAGE_GENERIC_MOUSE
0x04	Joystick	HID_USAGE_GENERIC_JOYSTICK
0x05	Game Pad	HID_USAGE_GENERIC_GAMEPAD
0x06	Keyboard	HID_USAGE_GENERIC_KEYBOARD
0x07	Keypad	HID_USAGE_GENERIC_KEYPAD
0x08	Multi-axis Controller	HID_USAGE_GENERIC_MULTI_AXIS_CONTROLLER

Joonis 6. Microsoft Windows ära määratud kaardistus seadmete ja id vahel.

Paigaldatud HID hiire seade hakkab saatma loodud aknale töötlemata hiire andmete sõnumeid. Sõnumite vastu võtmiseks on vaja luua kahe tasemeline tsükkel, mille eesmärk on kuulata aknale saadetuid sõnumeid kasutades *GetMessage* funktsiooni. Hiire andmete töötlemiseks on vaja luua CALLBACK makro sümbolit kasutatav sündmuse vastu võttev funktsioon, mis lisatakse akna andmestruktuurile selle loomise käigus. Sõnumi tsüklis kutsutakse välja *DispatchMessage* funktsioon, mis suunab kätte saadud sõnumid eelnevalt määratud CALLBACK tüüpi funktsioonile. [35]

```
// Main loop:
MSG event;
bool quit = false;
while (!stopMouseListener && !quit)
{
    while (GetMessage(&event, 0, 0, 0))
    {
        if (event.message == WM_QUIT || stopMouseListener)
        {
            quit = true;
            ReleaseTSFN();
            break;
        }

        // Does some Windows magic and sends the message to EventHandler()
        // because it's associated with the window we created.
        TranslateMessage(&event);
        DispatchMessage(&event);
    }
} });
// Stop();
```

Joonis 7. Kahe tsükli kasutus sõnumite kätte saamiseks sees paiknevast lõimest.

CALLBACK funktsioon tegeleb hiire andmete töötlemise ja kasutamisega. Kasutatava hiire informatsiooni kättesaamiseks on vaja luua RAWINPUT tüüpi töötlemata hiire andmestiku muutuja ning seejärel kasutada *GetRawInputData* funktsiooni, et kirjutada

äsja loetud hiire andmed loodud RAWINPUT muutujasse. Järgnevalt on võimalik ligi pääsed absoluutsetele delta x ja y koordinaatidele ning vajutatud hiire nuppudele.

Eelnevad sammud on vajalikud, et kasutada hiire ringvalikut mängudes ja väljapool mänge. Rakendus märgib ära millal hakati alla vajutama hiire rullik nupule ning millal see lahti lasti. Selle ajavahemiku vahel millal rullik nuppu all hoiti, salvestati liikumiste koordinaatide deltade arvutused. Vastavalt liikumisele koordinaatidele kas lahutati või liideti juurde. Kui rullik nupp lahti lastakse, hakatakse arvutama sektorit, kus hiir seisma jäi.

Sektori arvutuskäik käis tänu matemaatiliste vektorite pikkuse ning acos ehk arkuskoosinus valemile, et saada hiire trajektoori nurk y teljestiku suhtes. Silmas tuli pidada, et hiire liikumisel üles on y koordinaadid miinuses, ülejäänud teljestik sarnaneb standard matemaatilisele kraadijoonise struktuurile. Kuna arvutus käib y teljestiku suhtes on vaja järge pidada millises koordinaat sektoris hiir parasjagu oli. Vastavalt sellele oli vaja lisada 180 kraadi arvutuskäigule juurde. Viimaseks on vaja roteerida visualiseeritud ringi sektorid 22.5 kraadi vasakule. 22.5 kraadi tuleneb sellest, Apex Legends hiire ringvalik süsteem koosneb 8st pakutavast mängusisestest märguandest ehk sektorist ning esimene sektor on kraadide vahemikus 337.5-22.5, mis tähendab, et esimese sektori keskmine osa on tavapärase ringi viimase sektori ja esimese sektori vaheline ala.

```
int CalculateDegreeFromVectors(signed int x, signed int y)
{
    double vectorOneLength = y;
    double vectorTwoLength = sqrt(x * x + y * y);
    // VectorOne / VectorTwo
    double vectorLengthDivision = ((double)vectorOneLength) / ((double)vectorTwoLength);
    // Get degree
    double degree = acos(vectorLengthDivision);
    double sector = ((degree * 180 / 3.1415) + 22.5) / 45.0;
    double ceilSector = floor(sector);
    if ((x < 0 && y < 0) || (x < 0 && y > 0))
    {
        double reversalDegree = (180.0 + (180.0 - (degree * 180 / 3.1415)));
        if (reversalDegree > 360)
        {
            double reversalSector = reversalDegree / 45.0;
            return ceil(reversalSector);
        }
        else
        {
            double reversalSector = reversalDegree / 45.0;
            return round(reversalSector);
        }
    }
    return floor(ceilSector);
}
```

Joonis 8. Sektori arvutamine programmaatiliselt.

4.5.2 Klaviatuuri kuulamine

Klaviatuuri klahvide vajutuste informatsiooni oli võimalik kinni pidada kasutades eelnevalt kirjeldatud haakimismeetodit. Klaviatuuri kuulamise seadistamine on üsna sarnane hiire meetodiga. Esmalt luuakse selleks uus lõim, et vältida peamise lõime tegevuste takistamist. Lõime sees kirjeldatakse ära nii haakeprotseduur ning ka sõnumite tsükkel, mis kontrollib kas klahvistikku on kasutatud. Lõime siseselt on klaviatuuri haakimiseks vaja kasutada *SetWindowsHookEx* funktsiooni. Funktsioon võtab neli parameetrit. [36][37]

Esimeseks parameetriks võtab funktsioon haake id. Haake id määrab ära, millist seadet soovitakse haakida lõime külge. Id on võimalik kirjeldada inimkeeles loetava nimega, mis kaardistatakse numbriks või otse numbrilise väärtusena. [37]

Teise parameetrina ootab funktsioon haakeprotseduuri viidet. Viide on C/C++ maailmas mälu aadress, mis suunab õige väärtuseni. Viited võimaldavad kasutada vähem mälu ning pakuvad suuremat jõudlust ja arendus võimalusi. Suuremate andmemahtude transportimise ja kopeerimise asemel võib seda teha väiksemat jalajälge omava viitega. [37][38]

Kolmandaks ootab funktsioon DLL-i(*Dynamic Link Library*) ohjurit, mille sees on ära kirjeldatud haakeprotseduur, millele teises parameetris viidati. DLL-i ohjur on abstraktne viide, mille tüüp ei oma tähtsust. Samas lõimes või protsessis olev DLL ohjur suudetakse ära kaardistada õige DLL-s kirjeldatud funktsiooniga kui seda on samas lõimes või protsessis kasutusel. [37][39]

Neljandaks parameetriks ootab funktsioon lõime id-d, millega haakimist seostada. Kui selle väärtus on 0, kasutatakse lõimeks seda funktsiooni kutsuvat lõime [37].

```
// Set the hook and set it to use the callback function above
// WH_KEYBOARD_LL means it will set a low level keyboard hook. More
// information about it at MSDN. The last 2 parameters are NULL, 0 because
// the callback function is in the same thread and window as the function
// that sets and releases the hook.
if (!(_hook = SetWindowsHookEx(WH_KEYBOARD_LL, HookCallback, NULL, 0)))
{
    std::wcout << "Failed to install hook!" << std::endl;
}
```

Joonis 9. Klaviatuuri seadme haakimine selle kutsuva lõime külge.

Haake registreerimisel, lisatakse see haakeahelasse. Siinjuures tuleb olla teadlik ja ettevaatlik, mida soovitakse teha kätte saadud sõnumiga. Kuna haakimisi võib olla ahelas mitmeid, siis kõige uuem ahelasse lisatu saab sõnumi saamise prioriteedi. Üks sõnum käib potentsiaalselt läbi kõik ahelas olevad registreeritud haakimised. Et kõik töötaks arvutis edasi normaalselt, on vaja kasutada haakimist registreeritud lõimes *CallNextHookEx* funktsiooni. Kasutades seda funktsiooni saadetakse ahelas sõnum edasi järgmisele. Vastasel juhul sõnum jääb peatuma selles ahelapunktis, kus puudub *CallNextHookEx* funktsioon. [40]

```
// call the next hook in the hook chain. This is nessecary or your hook
// chain will break and the hook stops
// Passes hook event to other installed hooks, without this other programs will not receive
the event.
return CallNextHookEx(_hook, nCode, wParam, lParam);
```

Joonis 10. Haagitud sõnumi edasi saatmine haagete ahelas edasi.

4.6 C++ ja JavaScripti liidestus

Electron koosneb kahest samaaegselt jooksvast protsessist. Main ja Renderer, see järgib Google Chrome rakenduse tehnoloogiat, kus iga uus sakk on uus protsess, mis tagab turvalisuse ja jõudluse ja eraldiseisvuse, kus ühe saki erindid ja pikad laadimisajad ei peata terve veebilehitseja tegevusi. [41]

Kasutajaliides ehk Renderer protsess suhtleb alumise põhi protessiga läbi Electroni välja töötatud API-ga, milleks on IPC. IPC tagab turvalisuse, mida on vaja veebitehnoloogiate arendamisel. Electron Main protsess on võimeline kasutama operatsioonisüsteemi funktsionaalsust tänu Electroni kaasapakitud või hiljem lisatud C/C++ moodulite. Kui

Renderer protsessis on tehtud tüüpiline veebirakendus, mis näitena võib anda võimaluse käivitada pahatahtlikku JavaScript koodi, tekib suur turvarisk. [42][43]

Vite, kui arenduskeskkonna tööriist, on ES-moodulite peal üles ehitatud, et see oleks väikse mälu mahuga, kiire ning standarditel põhinev. Kuna C++-s kirjutatud heliloogika on kompileeritud node api ja node-gyp abiga Node tüüpi failiks, saab seda importida ainult tänu Node käituskeskkonna funktsionaalsusega. Vite ei toeta sedaviisi importimist ning kasutajaliidese protsess jääb turvaliselt eraldatuks äriloogikast. Kasutajaliides hakkab kasutama äriloogika funktsionaalsust läbi loodud Electron IPC projekti jaoks kohandatud API kaudu. Kohandatud API ligipääs avaldati Renderer protsessile tänu eel käitatava skripti, mis süstib (*inject*) Renderer protsessi viite JavaScripti, API sisenemispunkti. [42][44]

Suhtlemine käib sõnumite vahendusel. Sõnumid olid kohandatud API siseselt ära kirjeldatud tüüpidena, millel on sõnumi nimetus ning transporditav andmestruktuur. Sõnum saadeti kasutajaliidese nupu või muu tegevuse käigus Renderer protsessist Main protsessi, kus sõnum vastu võeti ning millele vastavalt kutsuti C++ mooduli funktsionaalsus. Main protsess, Renderer protsessi soovi täitmisel, saatis sarnase sõnumiga vastuse tagasi.

```
export type PlayClip = { type: 'playClip'; payload: string }
```

Joonis 11. Kohandatud IPC API sõnumi tüüp.

C++ mooduli kasutamiseks oli vaja see konstrueerida tänu Node-API ning Node-GYP tööriista kasutuse. C++ kood jäi enamasti muutumatuks ning ainult üksikutesse kohtadesse lisati Node-API liidestus JavaScripti objektide kasutuseks. Enamasti kasutasid need funktsioonid JavaScript objektide modifitseerimist, mis olid ära kaardistatud kui JavaScripti funktsioonidena. Kaardistatud funktsioone oli võimalik kasutada pärast C++ kompileerimist Node laiendiga mooduliks. [16]

5 Rakenduse paigaldaja loomine

Rakenduse paigaldaja loomiseks võeti kasutusele Inno Setup kompilaator, mis suudab vastavalt ette antud kirjelduste ja definitsioonide, moodustada Microsoft Windows süsteemidele täieliku paigaldaja. Inno Setup on abistav ja mugavdav tööriist ning enamus tööst on ära tehtud kompilaatori poolt, mis tähendab, et rakendusega seotud toimingud paigaldamisel ja eemaldamisel on ülesse seatud Inno Setup poolt.

Lõputöö projekt koosneb kahest koodibaasi hoidlast. Eelnevalt Inno Setup kasutusele on vaja nende projektide programmikood kompileerida, linkida ja komplekteerida kasutuskõlblikule või täidetavale kujule. Esiteks oli vaja seda teha heli funktsionaalsuste projektis, mis kirjutati C++ keeles. Selle jaoks kasutati Node-Gyp nimelist teeki, mis võimaldas kompileerida C++ koodi Node laiendiga kasutuskõlblikuks komplektiks, mida sai Node keskkonnas kasutada. Teiseks oli vaja kompileerida ja komplekteerida kasutajaliidese ja töölauarakenduse projekt. Omakorda oli see projekt veel tehtud kaheks: Vite.js, mis tegeles Vue.js raamistikuga loodud nähtava hüpertekstiga ja Electron.js, mis tegeles töölauarakenduse funktsionaalsustega, mis serveeris kasutajale seda hüperteksti. Seega lihtsustati ja komplekteeriti kõigepealt kasutajaliides ja selle funktsionaalsused. Seejärel ehitati ja pakiti kokku Electron.js rakendus ning sellega seonduvad failid ja andmed.

Komplekteeritud töölauarakenduse programm oli mahukas ning vajas erinevaid kaasapakituid faile, mis olid programmivälised. Need välised sõltuvused on võimalik liigutada töölauarakenduse programmi failide hulka ning seejärel rakendus kokku pakkida ja edasi saata. Kuid see ei taga kasutajamugavust selle paigaldamiseks. Inno Setup loob ühtse rakenduse, mis tegeleb paigaldamisega ise, küsides võimalikke kasutajasisendeid paigaldamiseks, luues operatsioonisüsteemi registrisse rakenduse haldamiseks vajalikud andmed, et seda oleks kergemini hallata eemaldamisel ja rakenduse jooksutamisel.

Paigaldusskript koosnes erinevatest sektsioonidest, mille nimed olid vastavalt soovitud operatsioonidele. Skript algab eelkõige rakenduse nime ning selle versiooni defineerimisest. Neid defineerituid muutujaid hakatakse hilisemates sektsioonides kasutama.

```
#define APPNAME "Demut"
#define VERSION GetFileVersion("..\demut-electron-vue-template\dist\win-unpacked\" + APPNAME + ".exe")
```

Joonis 12. Paigaldusskripti rakenduse nime ja versiooni defineerimine.

Setup nimelises sektsioonis määratakse ära paigaldaja ja eemaldaja nimed ning versioonid. Seadistatakse vaikimisi kasutatav paigalduskausta asukoht ning kompilaatori valikud nagu, millist tihendust kasutada, kas paigaldajaga tuleb kaasa ka eemaldamise võimalus, kas paigaldajal tekib logifail, mitu tuuma kompilaator võib kasutada, et protsessi kiirendada jne. [45]

```
[Setup]
AppName={#APPNAME}
AppVersion={#VERSION}
DefaultDirName={commonpf}\{#APPNAME}
OutputBaseFilename={#APPNAME}_{#VERSION}
SetupLogging=yes
Uninstallable=yes
SolidCompression=yes
LZMANumBlockThreads=4
```

Joonis 13. Skripti sektsioon paigaldaja ja kompilaatori seadistusteks.

Tasks sektsioon võimaldab kirjeldada paigaldus viisardis tehtavaid operatsioone, mida paigaldaja kasutaja saab aktiveerida raadionupu või märkeruuduga. Selles skriptis on kirjeldatud ära töö, mis küsib kasutajalt, kas soovitakse, et paigalduse käigus luuakse arvuti töölauale programmi otsetee. Tegu on Inno Setup sisseehitatud funktsiooniga, mis oskab ise leida rakenduse täitmisprogrammi ning sellest luua töölauale rakenduse otsetee. [46]

```
[Tasks]
Name: "desktopicon"; Description: "{cm:CreateDesktopIcon}"; GroupDescription: "{cm:AdditionalIcons}"; Flags: unchecked
```

Joonis 14. Tasks sektsioonis kirjeldatud rakenduse otsetee loomine töölauale.

Files sektsioon defineerib ära kõik failid, mida tahetakse, et paigaldusskript pakib kaasa ning mis paigalduse käigus lisatakse määratud failisüsteemi asukohale. [47] Rakenduse skriptile on defineeritud ära Electron.js pakendatud töölauarakenduse kausta sisu, rakenduse konfiguratsiooni fail, töölauarakenduse kasutatav ikoon ning helifunktsioonide kompileeritud fail.

```
[Files]
Source: ..\demut-electron-vue-template\dist\win-unpacked\*; DestDir: "{app}"; Flags: recursesubdirs; BeforeInstall: TaskKill('{#APPNAME}.exe')
Source: ..\demut-electron-vue-template\config\app-config.json; DestDir: "{commonappdata}\{#APPNAME}"; DestName: config.json;
Source: ..\demut-electron-vue-template\src\icons\icon.ico; DestDir: "{app}"
Source: ..\..\final-thesis-audio\build\Release\AudioEndpoints.node; DestDir: "{app}"
```

Joonis 15. Files sektsioonis kirjeldatud kaasapakitavad failid.

Dirs sektsioon on kasulik kui tahetakse luua tühjasid kaustasid või paigaldatavale programmile alamkaustasid. [48]

```
[Dirs]
Name: {commonappdata}\{#APPNAME}\DEMUT_WAV_CLIPS
```

Joonis 16. Dirs sektsioonis kirjeldatud tühja kausta loomine nimega DEMUT_WAV_CLIPS

Icons sektsioonis defineeritakse rakenduse täitmisprogrammi otseteede asukohad. *Tasks* sektsioonis olev funktsioon kasutab neid definitsioone, et luua otseteed õigesti. [49]

```
[Icons]
Name: "{commondesktop}\{#APPNAME}"; FileName: "{app}\{#APPNAME}.exe"
Name: "{commonstartup}\{#APPNAME}"; Filename: "{app}\{#APPNAME}.exe"
```

Joonis 17. Icons sektsioonis defineeritud töölaua ja kiirkäivitus otseteed.

Code sektsioon hõlmab Pascal keeles kirjutatud funktsionaalsusi, mis muudavad kuidas rakenduse paigaldus või eemaldus käitub. Skriptis on kirjutatud protseduur nimega *TaskKill*, mille eesmärk on kinni panna töötavad protsessid, millel on paigaldatava rakendusega sama nimi. See funktsioon on kasutusel *Files* sektsioonis, kui paigaldatakse Electron.js töölauarakenduse sisu.

```
[Code]
procedure TaskKill(FileName: String);
var
  ResultCode: Integer;
begin
  Exec(ExpandConstant('{sys}\taskkill.exe'), '/f /im ' + '"' + FileName + '"', '', SW_HIDE,
    ewWaitUntilTerminated, ResultCode);
end;
```

Joonis 18. Code sektsioonis kirjutatud Pascal keele protseduur rakenduse protsesside kinni panemiseks.

6 Rakenduse testimine ja järeldused

Rakenduse funktsionaalsusi testiti järjepidevalt rakenduse arenduse käigus, kuid selle vältel ei testitud kogurakendust kui paigaldatud töölauarakendust. Arenduse lõpus toimus kogurakenduse testimine, mille käigus paigaldati rakendus erinevatesse arvutitesse, kus testiti rakenduse funktsionaalsusi, rakenduse töökindlust ning ressursside kasutust. Rakenduse eesmärgiks loodud funktsionaalsust testis autor ning suhtlusraskustega isik. Anonüümsed isikud testisid rakenduse töökindlust, kasutaja kogemust ning ressursside kasutust.

6.1 Testimine

Testimisel on tähtsateks osadeks muutmälu kogus, protsessori mudel ja operatsioonisüsteemi versioon. Järgnevas tabelis on kirjeldatud seadmed, mis olid kasutusel testimisel. Graafika kaart ei mõjuta rakenduse tööd. Vt Tabel 1.

Tabel 1. Testimisel kasutatud arvutisüsteemid.

Tüüp	**Mudel**	**Protsessor**	**RAM**	**OS**
Lauaarvuti	-	Ryzen 7 2700	16GB	MS WIN 11
Lauaarvuti	-	Intel 7 10700k	16GB	MS WIN 11
Sülearvuti	Lenovo IdeaPad Flex 5	Ryzen 3 5300U	8GB	MS WIN 11
Sülearvuti	Acer Aspire 5	Intel 5 8250U	8GB	MS WIN 10
Sülearvuti	ASUS ROG Zephyrus G14	Ryzen 7 5800HS	16GB	MS WIN 10

Testimisel jälgiti tegumihaldurist rakenduse muutmälu kasutust ja protsessori jõudluse kasutust. Märgiti minimaalne, maksimaalne ja keskmine muutmälu ja protsessori kasutus. Järgnevad tabelid annavad ülevaate muutmälu ja protsessori kasutusest. Tulemused on vastavalt Tabel 1 järjestusele.

Muutmälu miinimum märgiti kui rakendus oli kasutamata vähemalt tund aega. Maksimumi jaoks märgiti kasutuse käigus kõige kõrgem väärtus. Keskmine tuletati järjepidevast kasutusest 5 minuti jooksul. Arvud on ümardatud allapoole. Vt Tabel 2.

Tabel 2. Muutmälu kasutuse miinimum, maksimum ja keskmised väärtused.

Miinimum (MB)	**Maksimum (MB)**	**Keskmine (MB)**
63	187	144
48	104	76
55	139	110
68	163	142
51	115	87

Protsessori kasutuse miinimum märgiti kui rakendus oli kasutamata vähemalt tund aega. Maksimum märgiti kui oli kasutusel nii optilise hiire kuulamine ja faili või otsese sünteesi töötlemine ning esitamine. Keskmine kasutus tuletati järjepidevast kasutusest 5 minuti jooksul. Vt Tabel 3.

Tabel 3. Protsessori kasutuse miinimum, maksimum ja keskmised protsendilised väärtused.

Miinimum (%)	**Maksimum (%)**	**Keskmine (%)**
0	4,7	3,5
0	1,8	0,8
0	4,3	3,2
0	11,3	5,8
0	2,8	1,5

Lisaks arvulistele mõõtmetele vaadati rakenduse kasutamise mugavust, selle töökindlust, intuitiivsust ja kasulikkust. Töökindlust ning intuitiivsust testiti anonüümsete isikute, autori ja sihtgrupi esindaja. Sihtgrupi esindaja testis peamiselt kasutusmugavust ning kasulikkust, mille all võrreldakse rehabilitatsiooni võimaldavat tugirakendust olemasolevate kasutatud lahendustega. Kasulikkuse hinnang annab autorile teada, kas tugirakendust vastab töö alguses püstitatud eesmärkidele. Tagasiside on võetud kokku üheks tabeliks, mille kriteeriumite väärtused 100 palli süsteemis. Vt Tabel 4.

Tabel 4. Rakenduse töökindluse, intuitiivsuse, kasutusmugavuse ja kasulikkuse tagasiside.

Töökindlus	Intuitiivsus	Kasutusmugavus	Kasulikkus
95	70	87	90

6.2 Tulemuste analüüs

Järgnevas peatükis analüüsitakse testimise tulemused ning sõnastatakse järeldused.

6.2.1 Muutmälu kasutus

Muutmälu kasutuse mõõtmise eesmärk oli hinnata rakenduse survet arvuti muutmälu ressursile. Vaadates Tabel 2 mõõtmisi on võimalik näha, et rakendus ei koorma arvuti muutmälu vabasid ressursse. Arvutades maksimumide keskmine ning keskmiste keskmine, saab tulemusteks vastavalt 141,6 MB ja 111,8 MB. Arvestades, et keskmiselt võtab Microsoft Windows 4 GB muutmälu, võib eeldada, et süsteemides jääb vastavalt 8 GB ja 16 GB muutmäluga süsteemides vabaks 4 GB ja 12 GB muutmälu. Selle põhjal on keskmiste keskmine 111,8 MB ainult 2,72% 4-st GB-st ning 0.90% 12-st GB-st. Neid arvutusi korrates, näitavad arvutused, et maksimumide keskmine 141,6 MB on 4-st GB-st 3,45% ning 12-st GB-st 1,15%. [50]

6.2.2 Protsessori kasutus

Protsessori kasutuse mõõtmiste eesmärk oli hinnata, kas rakendus on võimeline töötama mängudega samaaegselt, ilma, et see võiks mõjutada mängude jõudlust. Sekundaarne eesmärk oli hinnata jõudlust arvutides, kus mänge mängida ei ole võimalik, vaid kus oleks võimalik kasutada rakendust igapäevastes suhtlusrakendustes. Uurides Tabel 1 süsteeme ning vastavalt Tabel 3 protsessori kasutust, on võimalik järeldada, et mängimist võimaldavates süsteemides nagu Ryzen 7 2700, Intel 7 10700k ja Ryzen 7 5800HS saab ilma muredeta mängida samaaegselt mänge ning kasutada rakenduse sünteesitud mikrofoni kasutust. Nendes süsteemides jääb nii keskmine kui ka maksimaalne kasutus alla 5%, mis sobib autori jõudluse kriteeriumite alla. Süsteemides nagu Lenovo IdeaPad Flex 5 ning Acer Aspire 5 on protsessori kasutus kõrgem kui teistel süsteemidel. Nimetatud süsteemid on mõeldud igapäevaseks interneti surfamiseks ning lihtsa kontoritöö tegemiseks, mis tähendab, et mängude jõudlust rakendus nendes süsteemides ei segaks. Kõrgenenud kasutus tuleneb protsessorite tuumade ning kella kiiruste

madalamatest väärtustest. Sellegipoolest ei sega kõrgemad protsessori kasutuse näitajad rakendust kasutamast suhtlusrakendustes nagu Discord ja Teams.

6.2.3 Töökindlus, intuitiivsus

Töökindlust ja intuitiivsust hindasid suhtlemisraskustega isik, anonüümsed isikud ja autor, kes testisid rakendust mängudesiseselt, kui võimalik, ning üleüldiselt kui töölauarakendust. Töökindlus on hinnatud kõrgelt, kuna terve testimisperioodi jooksul ei lõpetanud rakendus töötamist. Harva juhtus ajutisi hangumisi, kuid see tuleneb vananenud riistvarast.

Intuitiivsust hinnati kõige madalamalt. Rakenduse pealehel on kirjeldatud funktsionaalsused ja nende kasutus, kuid sellele vaatamata tekkis segadus, kuidas mingit funktsionaalsust kasutada või milliseid samme on vaja sooritada antud järjekorras.

6.2.4 Kasutusmugavus ja kasulikkus

Kasutusmugavus ning kasulikkus oli üldiselt väga hea. Kuigi rakendus pakub lisaks otsese teksti sünteesile ka hiirega valitavat ringvalikut ettemääratud heliklippide esitamiseks ning seda toetavat heliklippide tekst kõneks sünteesi, jäi kõige enam kasutatavaks funktsionaalsuseks otsene mikrofoni simuleerimine. Ringvalik ei olnud piisavalt sätestatav ega mugav kasutada määratud hiire sisendiga. Küll aga on funktsioon kasulik kui kasutada seda kiirvastuseks nagu „Jah“ või „Ei“. Kõige enim kasutatud funktsionaalsus oli otsese teksti sünteesimine ning selle kasutamine füüsilise mikrofoni simuleerimiseks. Vajutades klahvi F9 oli võimalik esile tuua rakendusi kattev kuva, mille nurgas oli teksti sisendit nõudev lahter. Võrreldes seda eelnevalt kasutatud Steam rakenduse funktsionaalsusega või Discordi rakenduse kasutamisega oli tugirakendus mugavam ning kiirem. Steam rakenduse funktsionaalsus oli jõudluse poolest kehv. Selle esile kutsumine võttis aega ning võis tihti ka pikemaks ajaks hanguda. Discordi puhul pidi aga mängu rakendusest väljuma, et võtta fookusesse Discord rakendus ning seejärel sinna tippida sõnum. Selle infovahetus oli kõige aeglasem, kuna sarnaselt suhtlemisraskustega isikuga, pidi suhtluse sihtisik selle lugemiseks tegema sarnaseid samme. Sihtgrupi esindaja testis rakendust mängudes ja testis rakendust mängimise ajal suhtlemisrakenduste kõnedes. Lisaks kasutati rakendust vabal ajal sõpradega rääkimiseks erinevate kõnedes. Üldiselt on kommunikatsioon sihtgrupi esindajal paranenud. Inimestega, kellega suhtlus käib, ei pea enam lugema tekstipõhist suhtlust ning kuna

rakenduse mikrofoni simuleerimise kiirus on otsekohene on saavutatud lakoonilisem ja ajakohasem suhtlus. Samuti on toonud rakendus kasu mängude jõudlusele nii suhtlemisraskustega isikule, kuid ka temaga mängivatele isikutele. Enam ei ole vaja osapooltel kasutada näiteks Steam-i rakendust katvat lahendust, mille kasutamisel tekkisid märgatavad jõudluse hädad.

Lisaks suhtlemisraskustega isikutele rehabilitatsiooni võimaluse loomisele oli eesmärk ka leevendada vaimsete häirete mõju vaimsele tervisele. Seega oli tähtis ka hinnata, kas rakenduse olemasolu ning kasutus on mingil määral vaimset tervist mõjutanud positiivselt. Suhtlemisraskustega isiku jaoks oli juba see fakt, et keegi mõtleb inimestele, kellel sellised raskused on, väga rõõmu pakkuv. Samuti toob positiivset mõju lihtsalt ka see, et on olemas lisa võimalus, mida on võimalik kasutada lihtsustatud ja kiiremaks suhtlemiseks teiste osapooltega, mis tagab, et sihtgrupi esindaja häält kuulatakse tulevikus rohkem, mitte ei unustata tekstipõhiselt lugemata.

6.2.5 Järeldus

Antud testide väljundite põhjal on näha, et rakendus on ressursside kasutamisega tagasihoidlik ning on võimeline töötama nii kaasaegse kui ka aegunud riistvaraga süsteemides ilma kokku jooksmata. Suhtlemisraskustega isik andis hea ülevaate sellest, kuidas valmis rakendus täidab lõputöö alguses formuleerituid eesmärke. Kindlasti pakub rakendus rehabilitatsiooni võimalust tänu lisatud suhtlemisvõimalustega, mida on suhtlemisraskustega isikul lihtne ning mugav kasutada. Tänu sellele mõjutab tugirakendus vaimset tervist positiivselt, andes nendele isikutele teada, et nende peale mõeldakse ning üritatakse toetada.

7 Kokkuvõte

Käesoleva lõputöö eesmärk oli luua rakendus, mis pakub suhtlemisraskustega inimestele rehabilitatsiooni võimalust kiiretempolistes ühismängudes ning selle kaudu ka leevendada võimalike vaimsete häirete mõju vaimsele tervisele, mis võivad esineda suhtlemisraskuste olemasolust.

Eesmärgi saavutamiseks läbiti mitmed sammud. Esimeseks sammuks oli arendusele eelnev analüüs, mille käigus pandi paika kitsendused ja funktsionaalsused ning sõnastati lahti probleem ja selle lahendus. Järgmise sammu vältel tehti vajalikud tehnoloogia valikud ning kirjeldati nende otstarvet lahenduse arendamisel. Sellele järgnes rakenduse töötamiseks loodud koodi lahti seletamine ning rakenduse paigaldaja loomise kirjeldus. Viimase etapina testiti rakendust, analüüsiti tulemust ning konstrueeriti järeldus, mille käigus on kinnitatud, et rakendus on täitnud suhtlemise rehabilitatsiooni tugirakenduse eesmärgi ning vaimse tervise heaolu tõstmise eesmärgi.

Rakenduse edasi arendamiseks on plaanis viia rakendus täielikult iseseisvaks, mis võimaldaks lõpetada kolmanda osapoole lahenduse kasutust. Lisaks on plaan tõsta rakenduse jõudlust ja töökindlust, kasutades tuntuid arenduses kasutusel olevaid paradigmasid ning parimaid tavasid. Samuti arendatakse lisatud funktsionaalsusi ning kasutaja sõbralikumat kasutajaliidest, et saavutada parem tugirakendus.

Kasutatud kirjandus

[1] J. Clement (2022). „Number of video gamers worldwide 2021, by region“, Statista [Online], Available: https://www.statista.com/statistics/293304/number-video-gamers/ [Külastatud: 23.11.2022]

[2] YC. Lee, V.CH. Chen, YH. Yang (2020). Association Between Emotional Disorders and Speech and Language Impairments: A National Population-Based Study. Child Psychiatry and Human Development 51, 355–365. https://doi.org/10.1007/s10578-019-00947-9 [Külastatud: 23.11.2022].

[3] BL. Zhong, W. Luo, YM. Xu, WX. Li, WC. Chen, LF. Liu et al (2020). Major depressive disorder in Chinese persons with speech disability: High rates of prevalence and perceived need for mental health care but extremely low rate of use of mental health services, Journal of Affective Disorders, Volume 263, Pages 25-30, ISSN 0165-0327, doi: https://doi.org/10.1016/j.jad.2019.11.123. [külastatud: 23.11.2022]

[4] F. Spyridonis, D. Daylamani-Zad, M. O'brien. (2018). „Efficient In-Game Communication in Collaborative Online Multiplayer Games,“ in VS-Games At: Würzburg, Germany, 2018, doi: 10.1109/VS-Games.2018.8493420. [Külastatud: 24.11.2022]

[5] V. Vorteil, J. Laanpere, „Tarkvara analüüs ja testimine“, M. Laanpere, Tallinn, Estonia: Tallinna Ülikool (ilmumisaasta puudub). Available: https://web.htk.tlu.ee/digitaru/testimine/ [Külastatud: 23.11.2022]

[6] K. E. Wiegers, „Software requirements“, Redmond, Washington, USA: Microsoft Press, 2013, pp. 10. [online]. Available: https://archive.org/details/softwarerequirem0000wieg/mode/2up?q=functional [Külastatud: 23.11.2022]

[7] „About“. Git SCM. [Online]. Available: https://git-scm.com/about [Külastatud: 25.11.2022]

[8] J. Juviler. „What is GitHub? (And What Is It Used For?)“. (2022). HubSpot. [online]. Available: https://blog.hubspot.com/website/what-is-github-used-for [Külastatud: 25.11.2022]

[9] „Get started with desktop Windows apps that use the Win32 API“. (2021). Microsoft. [Online]. Available: https://learn.microsoft.com/en-us/windows/win32/desktop-programming [Külastatud: 24.11.2022]

[10] „Compiled versus interpreted languages“, [online]. Available: https://www.ibm.com/docs/en/zos-basic-skills?topic=zos-compiled-versus-interpreted-languages [Külastatud: 24.11.2022]

[11] Electron 20.1.0. (2022). GitHub. [online]. Available: https://github.com/electron/electron [Külastatud: 24.11.2022]

[12] „What is Electron?“. Electron. [online]. Available: https://www.electronjs.org/docs/latest/ [Külastatud: 24.11.2022]

[13] „What is Node-API“. Node-API resource. [online]. Available: https://nodejs.github.io/node-addon-examples/about/what/ [Külastatud: 30.11.2022]

[14] T. Sufiyan. „What is Node.js: A Comprehensive Guide“. Simplilearn. [online]. Available: https://www.simplilearn.com/tutorials/nodejs-tutorial/what-is-nodejs#what_is_nodejs [Külastatud: 24.11.2022]

[15] „Explain Event-Driven Programming in Node.js“. GeeksforFeeks. [online]. Available: https://www.geeksforgeeks.org/explain-event-driven-programming-in-node-js/ [Külastatud: 24.11.2022]

[16] „C/C++ addons with Node-API“. Node.js. [online]. Available: https://nodejs.org/dist/latest-v18.x/docs/api/n-api.html#node-gyp [Külastatud: 24.11.2022]

[17] „Node-gyp“. Node-API resource. [online]. Available: https://nodejs.github.io/node-addon-examples/build-tools/node-gyp [Külastatud: 30.11.2022]

[18] „Introduction“. Vue.js. [online]. Available: https://vuejs.org/guide/introduction.html [Külastatud: 24.11.2022]

[19] Vite 3.0.9. (2022). Vite. [online]. Available: https://github.com/vitejs/vite [Külastatud: 24.11.2022]

[20] „Features“. Vite. [online]. Available: https://vitejs.dev/guide/features.html [Külastatud: 24.11.2022]

[21] R. Kumar. „What is Inno Setup?“. DevOpsSchool. [online]. Available: https://www.devopsschool.com/blog/what-is-inno-setup/ [Külastatud: 24.11.2022]

[22] „Inno Setup“. Jrsoftware. [online]. Available: https://jrsoftware.org/isinfo.php [Külastatud: 24.11.2022]

[23] C. Bontecou. „Building an Electron App with VueJS and Vite“. Cody Bontecou. [online]. Available: https://codybontecou.com/electron-app-with-vuejs-and-vite.html [Külastatud: 25.11.2022]

[24] „Getting Started“. Vue.js. [online]. Available: https://012.vuejs.org/guide/ [Külastatud: 24.11.2022]

[25] „What is a single page application? A simple definition.“. (2022). AppCheck. [online]. Available: https://appcheck-ng.com/single-page-applications [Külastatud: 24.11.2022]

[26] „Getting Started“. Vue Router. [online]. Available: https://router.vuejs.org/guide/ [Külastatud: 24.11.2022]

[27] C. Hwuang. „WAV Files: File Structure, Case Analysis and PCM Explained“. (2022). VideoProc. [online]. Available: https://www.videoproc.com/resource/wav-file.htm [Külastatud: 25.11.2022]

[28] „Resource Interchange File Format (RIFF)“ (2021). Microsoft. [Online]. Available: https://learn.microsoft.com/en-us/windows/win32/xaudio2/resource-interchange-file-format--riff- [Külastatud: 25.11.2022]

[29] „Vector“. Cppreference. [Online]. Available: https://en.cppreference.com/book/intro/vector [Külastatud: 30.11.2022]

[30] „Rendering a Stream“. (2021). Microsoft. [online]. Available: https://learn.microsoft.com/en-us/windows/win32/coreaudio/rendering-a-stream [Külastatud: 29.11.2022]

[31] „About MMDevice API“ (2021). Microsoft. [online]. Available: https://learn.microsoft.com/en-us/windows/win32/coreaudio/mmdevice-api [Külastatud: 29.11.2022]

[32] „Microsoft Speech Platform ISpVoice“ (2012). Microsoft. [online]. Available: https://learn.microsoft.com/en-us/previous-versions/office/developer/speech-technologies/jj127823(v=msdn.10) [Külastatud: 30.11.2022]

[33] „ISpStreamFormatConverter::SetBaseStream (SAPI 5.3)“ (2017). Microsoft. Available: https://learn.microsoft.com/en-us/previous-versions/windows/desktop/ms719457(v=vs.85) [Külastatud: 30.11.2022]

[34] „RAWINPUTDEVICE structure (winuser.h)“ (2021). Microsoft. [online]. Available: https://learn.microsoft.com/en-us/windows/win32/api/winuser/ns-winuser-rawinputdevice [Külastatud: 25.11.2022]

[35] „DispatchMessage function (winuser.h)“ (2021). Microsoft. [online]. Available: https://learn.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-dispatchmessage [Külastatud: 25.11.2022]

[36] „Using Hooks“ (2021). Microsoft. [online]. Available: https://learn.microsoft.com/en-us/windows/win32/winmsg/using-hooks [Külastatud: 25.11.2022]

[37] „SetWindowsHookExA function (winuser.h)“ (2022). Microsoft. [online]. Available: https://learn.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-setwindowshookexa [Külastatud: 25.11.2022]

[38] U. Kirch-Prinz & P. Prinz, „Pointers“ in A Complete Guide to Programming in C++. Sudbury: Jones & Bartlett Learning, 2001. pp. 231.

[39] „Pushing the Limits of Windows: Handles“ (2009). Microsoft. [Online]. Available: https://learn.microsoft.com/en-us/archive/blogs/markrussinovich/pushing-the-limits-of-windows-handles [Külastatud: 28.11.2022]

[40] „CallNextHookEx function (winuser.h)“ (2021). Microsoft. [Online]. Available: https://learn.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-callnexthookex [Külastatud: 28.11.2022]

[41] „Process Model“. Electron. [online]. Available: https://www.electronjs.org/docs/latest/tutorial/process-model [Külastatud: 30.11.2022]

[42] „Security“. Electron. [online]. Available: https://www.electronjs.org/docs/latest/tutorial/security#2-do-not-enable-nodejs-integration-for-remote-content [Külastatud: 25.11.2022]

[43] „Inter-Process Communication“. Electron. [online]. Available: https://www.electronjs.org/docs/latest/tutorial/ipc [Külastatud: 30.11.2022]

[44] „Features“. Vite. [online]. Available: https://vitejs.dev/guide/features.html [Külastatud: 30.11.2022]

[45] „[Setup] section“. Jrsoftware. [online]. Available: https://jrsoftware.org/ishelp/index.php?topic=setupsection [Külastatud: 01.01.2023]

[46] „[Tasks] section“. Jrsoftware. [online]. Available: https://jrsoftware.org/ishelp/index.php?topic=taskssection [Külastatud: 01.01.2023]

[47] „[Files] section“. Jrsoftware. [online]. Available: https://jrsoftware.org/ishelp/index.php?topic=taskssection [Külastatud: 01.01.2023]

[48] „[Dirs] section“. Jrsoftware. [online]. Available: https://jrsoftware.org/ishelp/index.php?topic=taskssection [Külastatud: 01.01.2023]

[49] „[Icons] section“. Jrsoftware. [online]. Available: https://jrsoftware.org/ishelp/index.php?topic=taskssection [Külastatud: 01.01.2023]

[50] „Windows 11 System Requirements“. Microsoft. [online]. Available: https://support.microsoft.com/en-us/windows/windows-11-system-requirements-86c11283-ea52-4782-9efd-7674389a7ba3 [Külastatud: 02.01.2023]

[51] VB-CABLE (12), VB-AUDIO. [Application]. https://vb-audio.com/Services/licensing.htm

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

Mina, Mihkel Riik

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „DEMUT – Videomängude tugirakendus suhtlemisraskustega inimestele“, mille juhendaja on Kaido Kikkas
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

01.12.2022

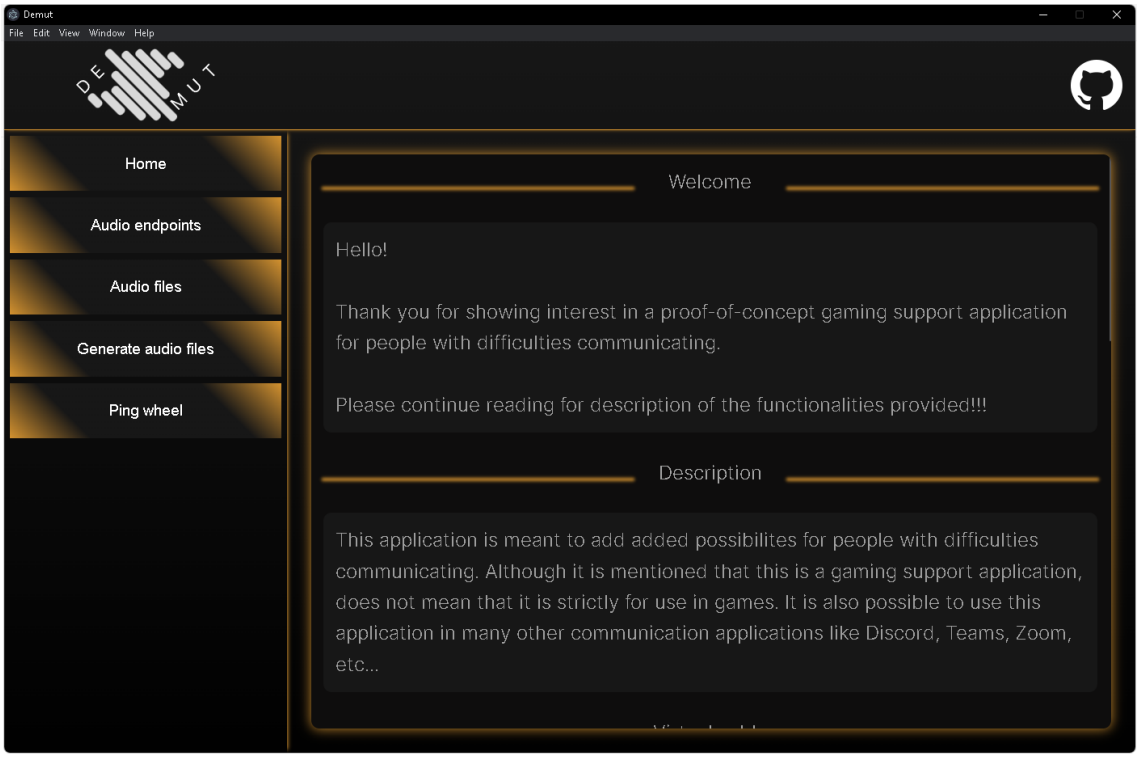
1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või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

Lisa 2 – Rakenduse lähtekood

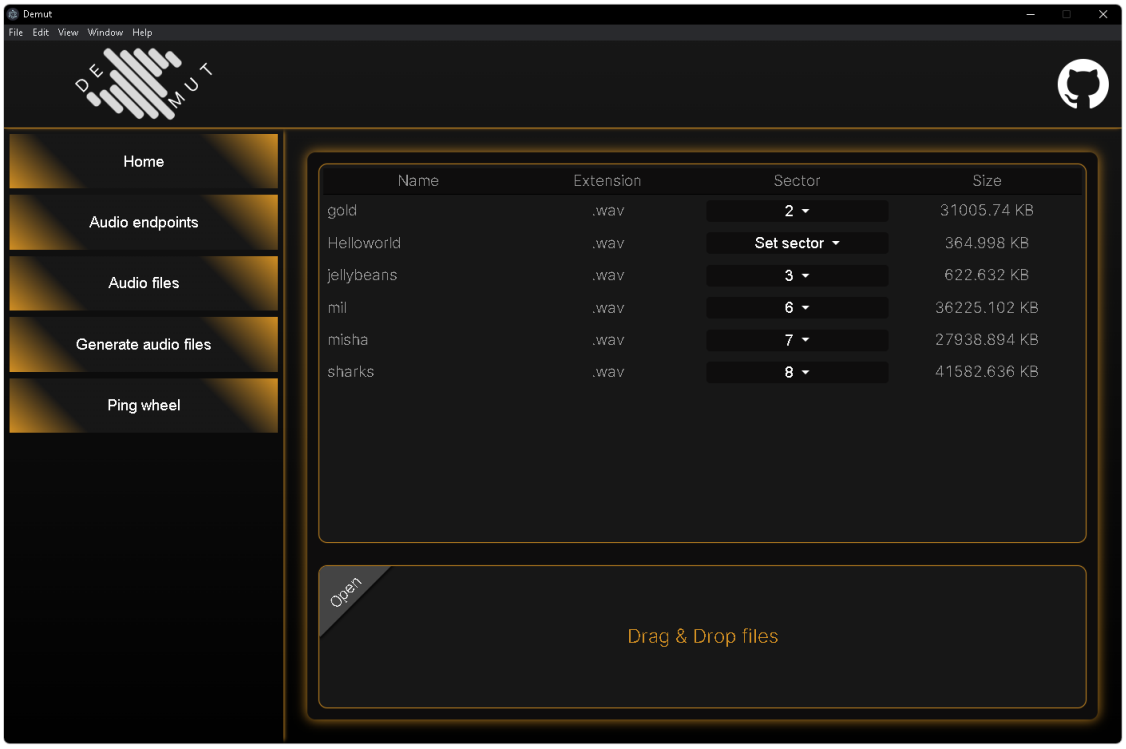
Kasutajaliides ja töölauarakendus – https://github.com/MikadoMiku/final-thesis-electron-ui

Madalatasemelise masina loogika kasutamise JavaScripti liides C++ keeles – https://github.com/MikadoMiku/final-thesis-audio

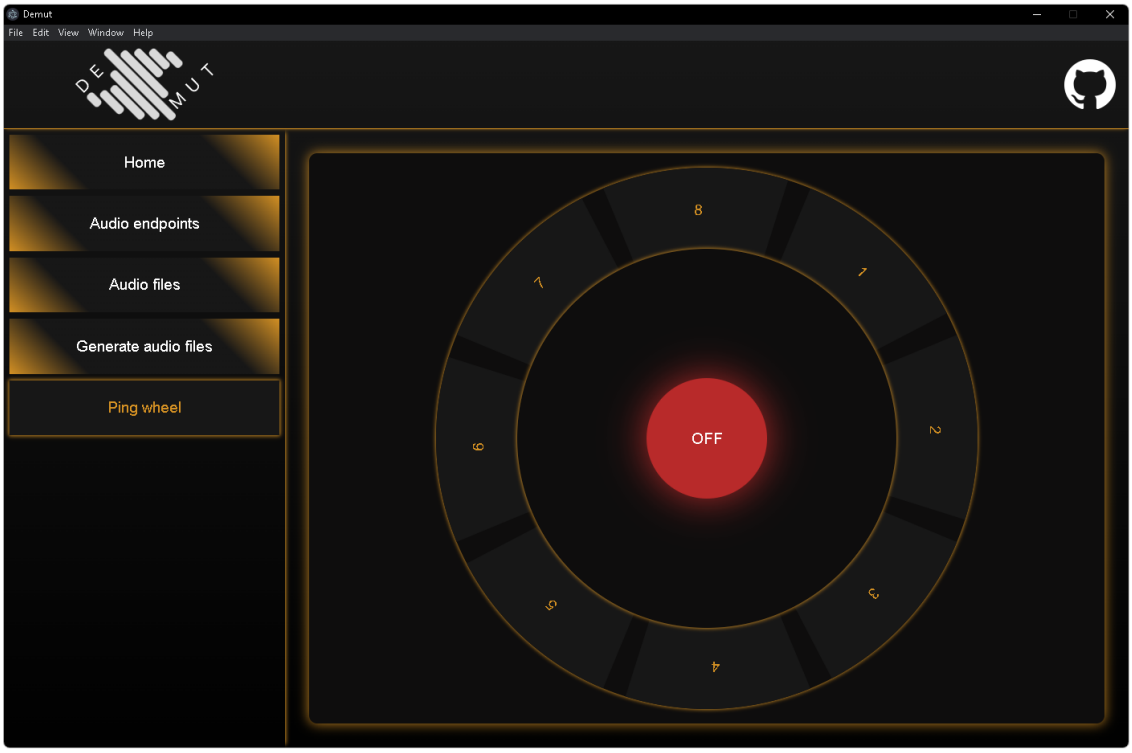
Lisa 3 – Kasutajaliidese esileht

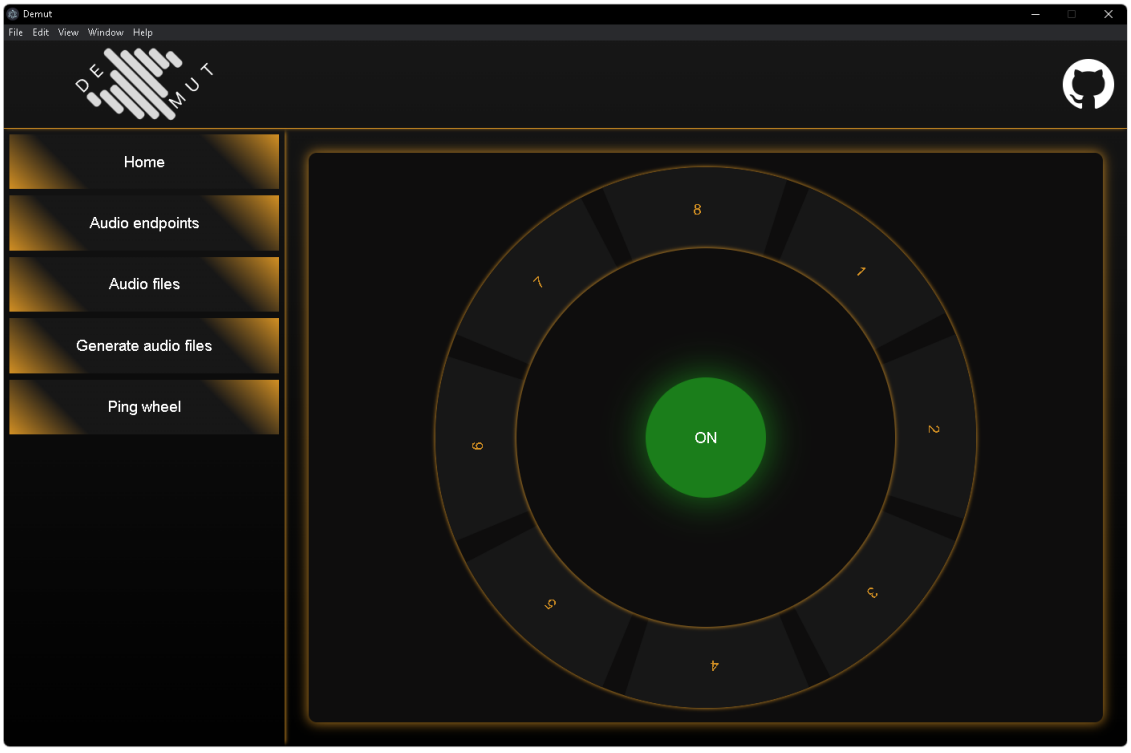


Lisa 4 – Kasutajaliidese helifailide leht

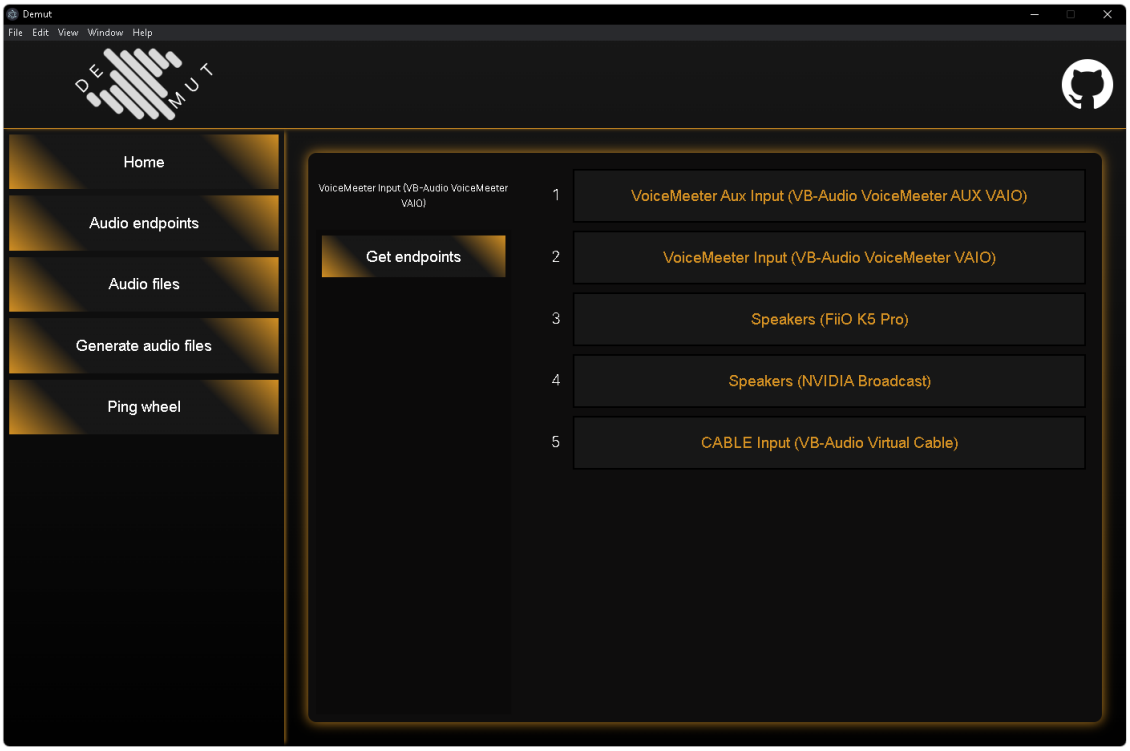


Lisa 5 – Kasutajaliidese hiire ringvaliku leht

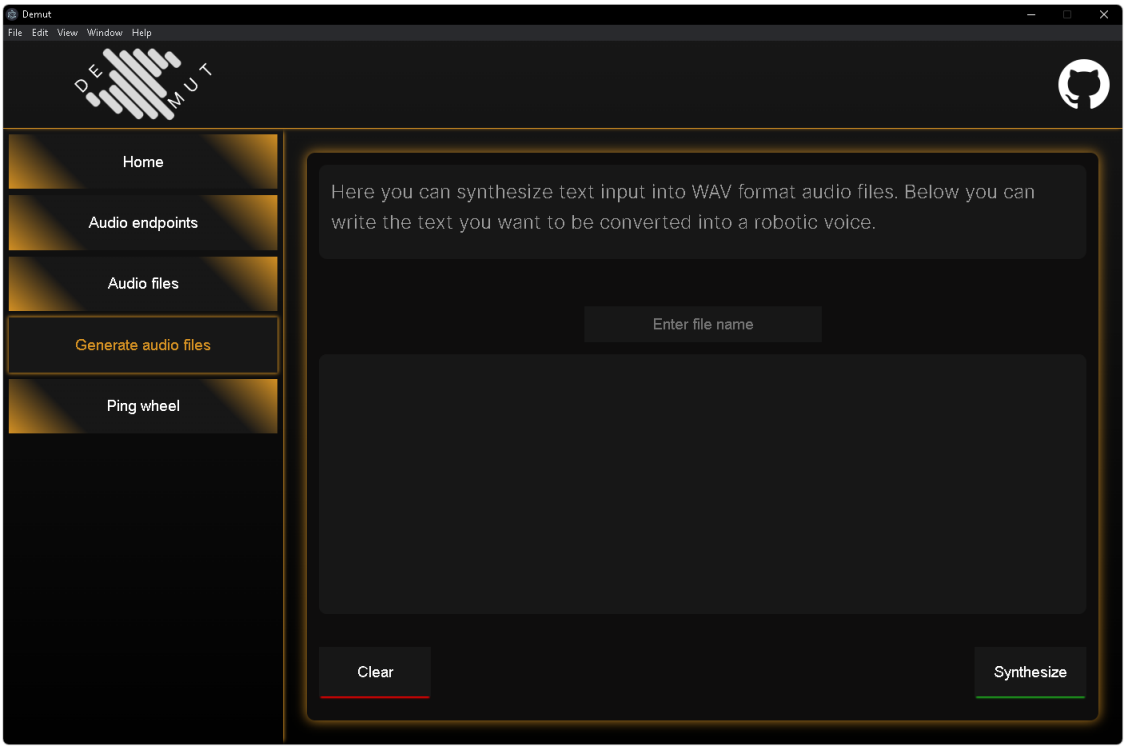




Lisa 6 – Kasutajaliidese heliseadme valik



Lisa 7 – Kasutajaliidese WAV helifailide süntees tekstist



Lisa 8 – Kasutajaliidese akent kattev teksti sisend otsese tekstisünteesi esitamiseks läbi mikrofoni simuleerimise