TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Indrek Luts 179537IADB

MÕISTATUSMÄNGU MIGRATSIOON PAINDLIKUMALE MOBIILIMÄNGUDE ARENDAMISE PLATVORMILE

bakalaureusetöö

Juhendaja: Meelis Antoi
MSc

Tallinn 2020

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: Indrek Luts

17.05.2020

Annotatsioon

Diplomitöö eesmärgiks on luua baasprojekt mõistatusmängu rakenduse üleviimiseks vanalt platvormilt uuele, mis on rohkem keskendunud mobiilimängude arendamiseks. Migreerimisel luuakse ka laiendus reklaamide kuvamiseks, mis aitab vähendada sõltuvust kolmandatest osapooltest. Samuti suurendatakse uues projektis süsteemi osade eraldatust, mille tulemusel on rakenduse haldamine ja uuendamine oluliselt lihtsam.

Analüüsi osas leitakse parim võimalik mänguarendusplatvorm antud projekti jaoks arvestades kehtestatud nõudeid. Teostus keskendub ületoomise lihtsustamisele, reklaamilaienduse arendamisele ja mängu sees kõige rohkem kasutust leidvate süsteemide loomisele. Mängu arhitektuur, mis on seotud navigeerimise ja esemete kasutamisega tehakse täielikult ümber – abiks on mitmed disainimustrid. Valminud projektis saab olema lihtsam tasemete loomise protsess. Kvaliteedi kontrollimiseks kasutatakse profileerimist, uurimuslikku testimist ja automaatset juhuslikku klikkimist.

Migreerimise tulemusel valminud rakenduse osa saab olema lõppkasutaja jaoks välimuselt ja kasutatavuse poolest suures osas identne praeguse versiooniga. Uuel platvormil loodud projekti baassüsteeme kasutatakse hiljem teiste ülesehituselt ja žanrilt sarnaste mängude üleviimiseks.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 40 leheküljel, 6 peatükki, 21 joonist, 7 tabelit, 4 koodinäidet.

Abstract

Puzzle game migration to a more flexible mobile games development platform

The goal of thesis is to create a base project for migrating a puzzle game application from old development platform to a new one, that is more focused to create mobile games. In the process a plugin will be created to display ads. That will help to lessen dependencies from third parties. Additionally modularity is increased which makes the application easier to maintain and update.

In the analysis phase the best game development platform will be selected based on the requirements. In the development phase a plugin for displaying ads will be created, migration process will be simplified and crucial parts of the game are designed and implemented. Game architecture that is responsible for navigation and inventory interactions is done from scratch using design patterns. Completed project provides easier level creation process. Quality control will use methods like profiling, exploratory testing and automated random clicking.

Migrated application will be identical to the end-user when compared to the previous version. That means user interface and usability will not be altered to a big extent. The base systems of the completed project are used to migrate other similar applications in the future.

The thesis is in Estonian and contains 40 pages of text, 6 chapters, 21 figures, 7 tables, 4 code samples.

Lühendite ja mõistete sõnastik

SDK	*(Software Development Kit)* – hulk arendustööriistu, mis võimaldavad luua rakendusi teatud tarkvara pakile, raamistikule, platvormile või arvutisüsteemile [3].
Adobe AIR SDK	SDK, mis võimaldab luua rakendusi ja mänge erinevatele seadmetele (iOS, Android, Mac, Windows) [4].
64-bit protsessor	Protsessor, mis võimaldab teha arvutusi kuni 64-biti pikkuste integer numbritüüpidega. Samuti viitab „64-bit“ protsessori registri laiusele, mida kasutatakse andmete aadressite ja väärtuste salvestamiseks [2].
Freemium mudel	Maksustamisstrateegia, kus teenus antakse tasuta tarbijale kätte (reklaamidega või ilma) ja seejärel pakutakse lisateenuseid tasu eest [8].
singleton	Disainimuster, mille rakendamisel luuakse klassist üksainus eksemplar.
Gradle	Vabavaraline rakenduse ehitamise tööriist.
adapter	Disainimuster, mille rakendamisel mittekokkusobivad klassid saavad omavahel suhelda. Tekstis tähendab see koodi või teeki, mis aitab seda saavutada.
plugin	Tarkvara komponent, mis annab programmile lisavõimalusi.
stseen	Grupp *blokke* organiseeritud puu kujul, millel on alati üks juur ja mida saab salvestada kettale ja sealt laadida. Igal Godoti mängul peab alati olema üks kindel *stseen* alguspunktiks [34].
blokk	Fundamentaalne ehitusblokk mängu loomiseks. Iga *blokk* saab omada teisi *blokke* - niimoodi luuakse puustruktuur [34].
stseenipuu	Üks olulisemaid klasse Godoti mängumootoris, mis haldab *blokkide* hulkade hierarhiat *stseenis* ning samuti *stseene* endid [35].
callback	Igasugune käivitatav kood, mis antakse argumendina teisele koodile ning eeldatakse, et see teine kood kutsub antud koodi välja teatud ajal [36].

GDScript	Kõrgtaseme programmeerimiskeel, mis on dünaamiliselt tüübitud ja mõeldud sisu loomiseks Godotis. Meenutab süntaksilt Pythonit [37].
level	Mängu tase, mis on eraldiseisev *stseen* ja mille raskusaste peaks kasvama olenevalt selle järjekorranumbrist.
sprite sheet	Pildifail, mis sisaldab endas palju väiksemaid pilte ruudustikuna. Mitmete piltide kompileerimine ühte faili aitab mänguarenduses tihti parandada jõudlust [39].
Adobe Animate	Multifilmide, reklaamide, mängude ja muu interaktiivse sisu loomiseks mõeldud programm, mis võimaldab avaldada neid paljudel erinevatel platvormidel [41].
JavaScript	Kõrgtaseme programmeerimiskeel.
PNG	Rastergraafika failiformaat.
mediator	Disainimuster, mis kapseldab ühte objekti mingi hulga objektide vastastikuse käitumise loogika. Aitab neil objektidel säilitada sõltumatust kuna objektid ei viita üksteisele otseselt [43].
GML	GameMaker Studio 2 jaoks loodud dünaamilise tüübikontrolliga programmeerimiskeel.
SBTM	Session-Based Test Management ehk sessioonipõhine testimise haldamine. Strateegia, mis aitab viia läbi testimist struktureeritumal kujul.
exploratory testing	Uurimuslik testimine. Rõhutab testija autonoomsust, oskusi ja loovust.
profileerija	Tarkvara tööriist, mille abil mõõdetakse erinevaid käitusaja karakteristikuid ja ressursside kasutust.
Google Play Console	Rakenduste ja mängude publitseerimise platvorm, mis muuhulgas pakub arendajatele võimalust parandada oma rakenduste kvaliteeti.
adb	Android Debug Bridge. Käsurea tööriist, mis võimaldab arendusmasinast ühenduda seadmega ja sellega suhelda ning käivitada seal erinevaid käske.
monkey	Käsurea programm rakendustele koormustestide tegemiseks. Genereerib pseudo-juhuslikke puudutusi ja muid sündmusi.
logcat	Käsurea tööriist, mis kuvab nii süsteemi logisõnumeid kui ka rakendusest saadetud logisõnumeid (kui kasutada Log klassi).

NavigationDirector	Diplomitöös valminud programmi klass, mis koordineerib paljude erinevate klikitavate alade koostoimimist.
InventoryDirector	Diplomitöös valminud programmi klass, mis koordineerib inventaris olevate pesade koostoimimist
Bubble Sort	Üks lihtsamaid sorteerimismeetodeid, kus kõrvuti olevad elemendid vahetatakse ümber, kui nad pole õiges järjekorras. See jätkub nii kaua, kuni vales järjekorras olevaid element enam pole.
Drag and Drop	Visuaalne programmeerimise tööriist, kus kood genereeritakse automaatselt vastavalt seadistatud ja ühendatud graafilistele blokkidele.
In-App Purchase	Rakenduse sisene ost, mille sooritamine annab kasutajale rakenduses lisavõimalusi [5].
laadur	Ressursside laadimiseks mõeldud stseen, mis kuvab kasutajale protsente laadimise edenemisest. Rakenduses kasutati allika [54] lahendust.
observer	Disainimuster, mis võimaldab ühe objekti seisu muutumisel teavitada kõiki temast sõltuvaid objekte ning uuendada neid automaatselt.
eksport	Andmete salvestus mingi teise programmi jaoks sobivas vormingus.
Test-Driven Development	Programmeerimise praktika, kus arendaja kirjutab esiteks ebaõnnestuva testi ja seejärel koodi, mis paneb testi läbima. Test kujutab endast nõudeid programmile. Üks eesmärkidest on vähendada vigu.
Staatiline tüübikontroll	Tüüpe kontrollitakse kompileerimisel.
Skriptimiskeel	Programmeerimiskeel, millega juhitakse programmide tööd.
beta faas	Tarkvara avaldamise faas, mis järgneb *alpha* faasile.
protsess	Täitmisel olev programm või selle osa [63].
mälu	Kogu adresseeritav, käskude täitmiseks kasutatav salvestusruum [63].
protsessor, CPU	Interpreteerib ja täidab käske ning koosneb vähemalt käsuseadmest ja aritmeetika-loogikaseadmest [63].

Sisukord

Jooniste loetelu

Tabelite loetelu

Koodinäidete loetelu

1 Sissejuhatus

Diplomitöö eesmärgiks on puuduva arhitektuuriga mõistatusmängu üleviimine uuele arendusplatvormile, mis võimaldab hõlpsamat haldamist ja laiendamist. Tuleb jälgida, et kulud oleksid võimalikult madalad ja arendustööriistade ning teekide toetus püsiv ka tulevikus.

Adobe Animate, kus ümbertehtav rakendus on loodud, ei ole fokuseeritud mängude arendamiseks. See on üks põhjustest, miks otsitakse alternatiivi. Teine põhjus on see, et teek, millega rakendust Android ja iOS platvormile eksporditakse, muutus tasuliseks. Tasuline on ka Adobe Animate, millega praegune projekt on arendatud.

Üks võimalus, mis lahendaks haldamise probleemid, oleks loobuda uuenduste tegemisest. See aga tooks kaasa rahalisi kaotusi. Selleks, et rakendusi poest ei eemaldataks on tarvis täita Google ja Apple poolt esitatud nõudeid. Näiteks eemaldatakse poest rakendused, mis ei toeta 64-bit protsessoreid [1]. Enamusel ettevõtte rakendustest selline tugi puudub. Nõude täitmata jätmine tähendaks suure osa kasutajate kaotust ja sellepärast pole uuendusi võimalik vältida.

Diplomitöös on plaanis leida uus sobiv platvorm, mis täidab esitatud nõudeid. Kulud proovitakse hoida võimalikult madalad. Seejärel peab valmima sellise arhitektuuriga rakendus (muuhulgas suurema süsteemi osade eraldatusega), mis võimaldab projekti paremini hallata. Ühtlasi peab olema võimalus lihtsamini uusi tasemeid lisada.

Projekt on konkreetselt vajalik ettevõttele, mille valmimisel viiakse üle ka teised sarnased mängud. Lahendus võib pakkuda huvi ka teistele arendajatele, kes otsivad alternatiive.

1.1 Migreeritavast mängust

Mäng kuulub mõistatusmängude kategooriasse. Mängus on 15 erineva raskusega taset. Igal tasemel tuleb leida üles vajalikke esemeid, neid kasutada ja lahendada mõistatusi. Kõige selle tulemusel leitakse viimaks üles uksekaart, mis võimaldab peavaates olevast uksest väljuda. Näited mängu edenemisest on näidatud Joonisel 1, 2 ja 3.

Joonis 1 – Mängu algseis

Joonis 2 – Mängus võtme kasutamine sahtli peal, et leida sealt vihje

Joonis 3 – Mängus edukalt läbitud taseme lõppseis

2 Analüüs

2.1 Nõuete kogumine

2.1.1 Meetodi valik

Rakenduse nõuete kogumisel keskenduti põhiliselt arendaja vaatenurgale. Põhjus on selles, et kuna tegemist on migratsiooniga siis uut funktsionaalsust on plaanis lisada esimesel korral vähe. Kasutajate rahulolematus pole olnud probleemiks. Tuginedes autori eelnevale kogemusele on olnud takistavaks asjaoluks rakenduse haldamisel ja edasiarendamisel mängu arhitektuur ja arendusplatvormi vähene võimekus.

Nõuete kogumisel kasutati McCalli mudeli [6] modifikatsiooni [7], sest et see keskendub rohkem rakenduse loomisele, haldamisele ja teiste teekidega koostoimimisele. Teha mängu fundamentaalseid muudatusi pole autori arvates hetkel riski väärt ja need ei mahuks ka skoopi.

Puudub vajadus üles kirjutada paljusid funktsionaalseid nõudeid kuna need on nähtavad praeguses rakenduses ja need tuleb viia üle uude rakendusse täpselt samal kujul.

Punktis 2.1.2 toodud „Sisemise kvaliteedi nõuded“ keskenduvad enamasti arendaja vaatenurgale ja punktis 2.1.3 olevad „Välimise kvaliteedi nõuded“ lähtuvad eelkõige kasutaja vaatest. Sellist sõnakasutust on kasutatud ka allikates [7, 57].

2.1.2 Sisemise kvaliteedi nõuded

Tabel 1 – Sisemise kvaliteedi nõuded

	Nõue
Tõhusus	Rakendus ei tohi olla suurem kui keskmine suurus antud mängul võrdlusgrupis (Google Play Developer Console)
	Rakendus ei tohi kasutada telefoni akut taustal, kui selleks puudub vajadus
	Arenduskeskkonnas on kasutusel staatilise tüübikontrolliga programmeerimiskeel
Hallatavus	Arenduskeskkonnas on visuaalne stseenide monteerimise võimalus
	Arenduskeskkonnal on aktiivne kogukond
	Dokumentatsioon ei piirdu ainult API napisõnalise kirjeldamisega
	Arenduskeskkonna programmeerimiskeel on TOP 50 kõige kasutatavama keele seas
Testitavus	Arenduskeskkonnas on võimalus automaatteste kirjutada
	Arenduskeskkonnas on võimalus kasutajaliidese teste kirjutada
Paindlikkus	Arendajal on võimalus luua uusi tasemeid ilma muud mängu muutmata
Ühilduvus	IronSource laienduse olemasolu või valmidus selle loomiseks
	Mängusiseste ostude laienduse olemasolu või valmidus selle loomiseks
	Arenduskeskkonnal peab olema lihtne võimalus andmeid kasutaja seadmesse salvestada
Taaskasutatavus	Navigatsiooni süsteem on taaskasutatav
	Mängu seisu salvestamise loogika on taaskasutatav
	Mängusisene inventari süsteem on taaskasutatav
	Reklaami laienduse ja mängusiseste ostude loogika on taaskasutatav
Teisaldatavus	Mängu peab saama viia nii iOS kui Android platvormile ühe koodibaasiga

2.1.3 Välimise kvaliteedi nõuded

Tabel 2 – Välimise kvaliteedi nõuded

	Nõue
Terviklikkus	Kasutajal ei tohiks olla võimalik teha kettal muudatusi, et mängus ebaausalt järgmistele tasemetele jõuda
	Taseme seis seadmes peab säilima ka pärast rakenduse uuendamist
	Andmete õigsus peab säilima pärast mistahes operatsiooni seadme kettaga
Töökindlus	Kokku jooksmisi viimase 30 päeva jooksul ei tohi esineda rohkem kui 0.18% kasutajate päevastest sessioonidest (Google Play Developer Console info põhjal)
	Kasutajaliidese reageerimatust viimase 30 päeva jooksul ei tohi esineda rohkem kui 0.10% kasutajate päevastest sessioonidest (Google Play Developer Console info põhjal)
Kasutatavus	Kasutajaliidese navigeerimisrajad peavad jääma muutumatuks
	Navigatsioon ühest vaatest teise ei tohi olla aeglasem, kui praeguses rakenduses; hinnata mitme kasutaja tagasiside põhjal
	Praegused kasutajad ei pea õppima midagi juurde, et rakendust kasutada
	Mängu salvestamine/laadimine ei tohiks kasutajaliideses tekitada tõrkeid/jõnkse
	Vähemalt 7 populaarseima keele tugi
	Toetus peab säilima enamusele 13 270 seadmest (Google Play Developer Console info põhjal)

2.1.4 Nõuete prioritiseerimine

Kuna kõik nõuded näisid ühtmoodi tähtsad siis oli vaja leida võimalus neid prioritiseerida. Kasutati *Bubble Sort* tehnikat, mis annab võrdlemisi täpseid tulemusi ja on mõeldud kasutamiseks pigem väikese kogusega nõuete puhul [58]. Ennem tehnika kasutamist valiti välja ainult need nõuded, mis on seotud rakenduse arendamise ja haldamisega.

Pärast prioritiseerimist eraldati 7 kõige tähtsamat nõuet Tabelisse 3, mille alusel alustati diplomitöös kasutatava platvormi väljavalimist.

2.1.5 Esmane valik

Koheselt välistati väga spetsiifiliste eesmärkidega ja hobikorras loodud platvormid. Kokku valiti välja 10 erinevat arendusplatvormi. Tabelis 4 on näha kuidas need täidavad ettevõtte poolt esitatud nõudeid. Tabelis 4 tuleb tähele panna, et nõude nr. 2 täitmine eeldab, et laiendus on juba olemas ja seda arendama ei pea. Võimalust platvormidel laiendust luua analüüsitakse punktis 2.2.1.

Tabel 3 – Kõige olulisemad nõuded prioritiseeritult

Tähtsus	Nõue
1	Mängusiseste ostude laienduse olemasolu või valmidus selle loomiseks
2	IronSource laienduse olemasolu või valmidus selle loomiseks
3	Arenduskeskkonnas on visuaalne stseenide monteerimise võimalus
4	Mängu peab saama viia nii iOS kui Android platvormile ühe koodibaasiga
5	Arenduskeskkonnal peab olema võimalus andmeid kasutaja seadmesse salvestada
6	Arenduskeskkonnas on võimalus automaatteste kirjutada
7	Arenduskeskkonnas on kasutusel staatilise tüübikontrolliga programmeerimiskeel

Tabel 4 – Potentsiaalsed arenduskeskkonnad

	1	2	3	4	5	6	7	Hind
Amazon Lumberyard	JAH	EI	JAH	JAH	JAH	JAH	EI	Tasuta
Android Studio ja Xcode	JAH	JAH	JAH	EI	JAH	JAH	JAH	Tasuta
Cocos Creator	MA	EI	JAH	JAH	JAH	EI	JAH	Tasuta
Corona	JAH	JAH	EI	JAH	JAH	EI	EI	Tasuta
Defold	JAH	EI	JAH	JAH	JAH	MA	EI	Tasuta
GameMaker Studio 2	JAH	$	JAH	JAH	JAH	MA	EI	$199
Godot	JAH	EI	JAH	JAH	JAH	MA	JAH	Tasuta
libGDX	JAH	EI	EI	JAH	JAH	MA	JAH	Tasuta
Unity	JAH	JAH	JAH	JAH	JAH	JAH	JAH	$399/a
Unreal Engine	JAH	MA	JAH	JAH	JAH	JAH	JAH	5% *

MA – Mitteametlik ehk pole toetatud platvormi arendajate poolt või puudub info dokumentatsioonis.

$ – Tasuline.

* – Kui kvartalis teenitud kogutulu ületab $3000 siis tuleb 5% maksu tasuda teenitud summalt.

2.2 Arenduskeskkonna valimine

2.2.1 Reklaamivahendaja teegi kasutusvõimalus

Selgus, et mänguarendusplatvormide kõige suuremaks puuduseks oli võimalus kuvada reklaame ettevõtte poolt valitud reklaamivahendaja kaudu. Diplomitöö tegija arvates võib põhjus olla mänguarendusplatvormide suures arvus ja reklaamivahendaja huvi puudumises kõigile platvormidele laienduse arendamiseks. Kuidas ise kolmanda osapoole teeke platvormiga siduda oli Defold ja Godot puhul dokumentatsioonis kirjeldatud. Kuigi näiteks Defold puhul õnnestus autoril õpetuse järgi lihtne laiendus luua aga reklaamivahendaja teegi sidumine näis olevat tunduvalt raskem ja nõudis head C++ keele oskust. GameMaker Studio 2 poes olev lahendus maksis 46 eurot ja see oli ainuke variant. Unreal Engine jaoks leidus kahe aasta vanune laiendus, mis polnud enam sünkroonis hetkel kasutusel oleva reklaamivahendaja teegiga. Kirjeldatud hinnad ja saadavus on seisuga 6. märts 2020.

2.2.2 Automaattestide kirjutamise võimalus

Kuna algne eesmärk oli rakendada *Test-Driven Development* praktikat, mis võimaldab vähendada loodavas tarkvaras esinevate defektide arvu [59] siis oli automaattestide kirjutamise võimalus nõutud. Testide kirjutamise võimalus leidus paljudel väiksema kasutajaskonnaga arendusplatvormidel kuid esines puudujääke. Enamikel platvormidel oli küll võimalus teste kirjutada ja käitada aga palju oli selliseid lahendusi, kus testide kirjutamiseks tuli kasutada kolmanda osapoole teeke. Teegid olid tihti kirjutatud ainult ühe või kahe arendaja poolt. Diplomitöö tegija arvates on oht, et selliste teekide toetatavus võib tulevikus kergemini kaduda. Kirjutada polnud enamasti võimalik keerulisemaid teste, mis kasutavad simuleeritud objekte.

Kõige parem tugi testide kirjutamiseks oli Unity, Unreal Engine ja Amazon Lumberyard platvormil. Kõigil neil olid vajaminevad tööriistad olemas. Autori arvates oli Godoti laiendus GUT testimiseks piisavalt hea ja kindlust lisas projekti kaastöötajate suurem hulk [9].

2.2.3 Programmeerimiskeel

Samuti osutus tähtsaks nõudeks staatilise tüübikontrolliga keele toetatavus, kuna selliste keelte kasutamine aitab ennetada vigu [60].

Uuritavatel platvormidel polnud selline keel tihti toetatud.

Godotis oli toetatud C#, C++ ja GDScript. Kus esimese kahe puhul toimub tüübikontroll enne käitamist ja viimase puhul käitamise ajal. Amazon Lumberyard, Corona ja Defold on võtnud kasutusele Lua skriptimiskeele. GameMaker Studio 2 kasutab spetsiaalselt platvormile loodud skriptimiskeelt GML. Lua ja GML puhul toimub tüübikontroll käitamise ajal. Ülevaade keelte toetatavusest on seisuga 6. märts 2020.

Ühtlasi peaks autori arvates keel positsioneerima end vähemalt 50 kõige populaarsema programmeerimiskeele hulka [11] kuna see teeb arendajate leidmise lihtsamaks.

2.2.4 Ebasobivad platvormid rakendusele

Esimesena välistati Amazon Lumberyard kuna platvorm oli kirjutamise hetkel endiselt *beta* faasis ja seega ei pruugi olla stabiilne ning võib sisaldada vigu. Puudus reklaamivahendaja laiendus ja ei leidunud piisavalt dokumentatsiooni kolmanda osapoolte teekide kasutamiseks.

Samuti ei osutunud sobilikuks Unreal Engine, kuna viimane on ehitatud pigem AAA ehk suurema eelarvega mängude arendamiseks [12]. Seetõttu pole see optimaalne variant arvestades diplomitöös arendatava rakenduse lihtsust. Samuti oli reklaamivahendaja laiendus aegunud. Kui autor proovis Androidi seadme jaoks käitatavat faili eksportida siis oli failimaht tühja projekti puhul üle 50 MB, mida on liiga palju. Umbes selline peaks olema lõpliku mängu kogusuurus.

Android Studio ja XCode kasutamine kõrvaldaks väga palju muresid, mis kaasnevad Apple ja Google uuendustega. Platvormide kasuks räägib veel see, et koheselt oleks võimalik kasutada IronSource teeki ja laienduste loomine poleks vajalik.

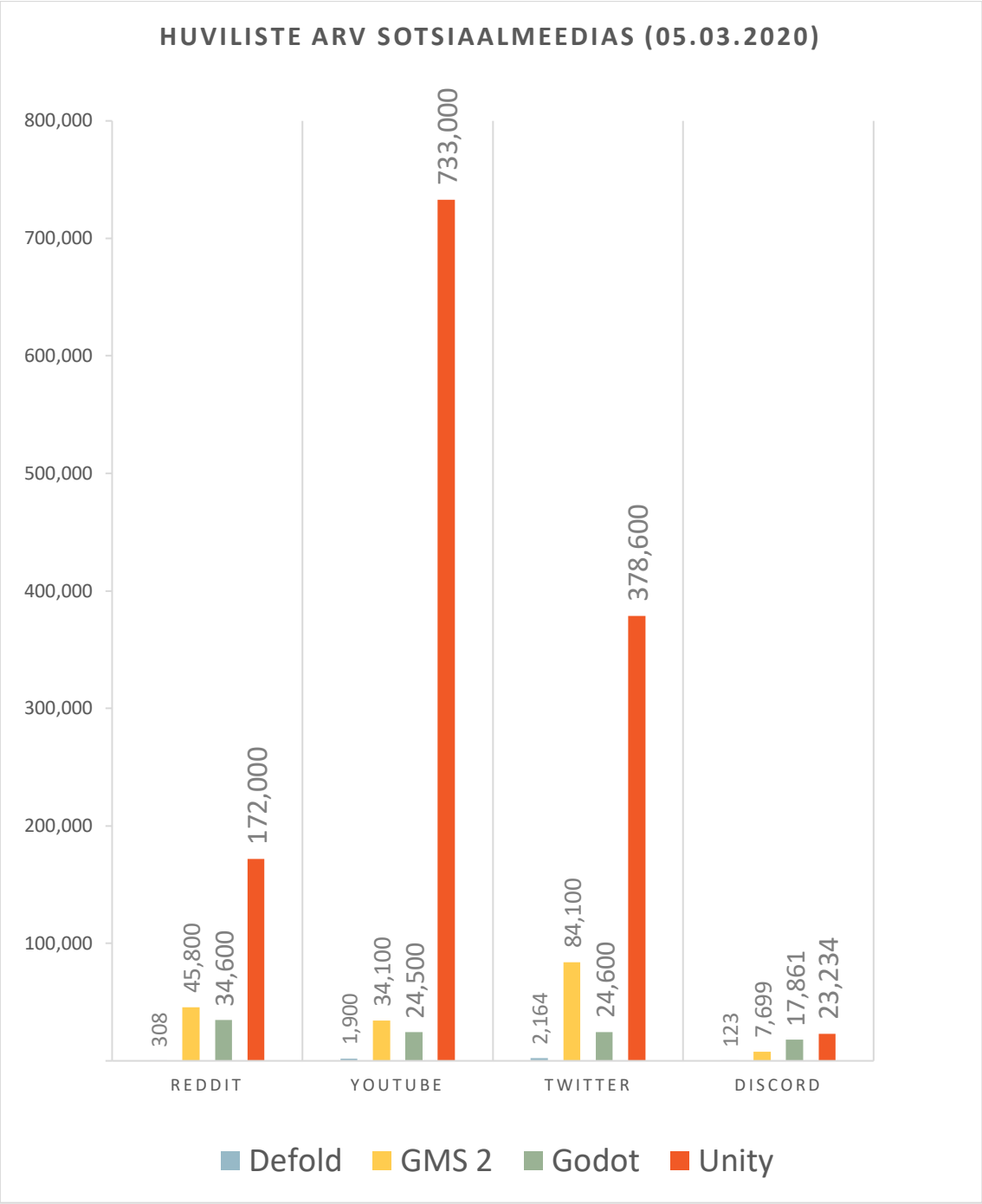
Miinusena tuleb arvestada, et antud platvormid pole mõeldud mängude arendamiseks. Samuti nõuab rohkem ressursse kahe erineva koodibaasi ülalpidamine. Selline lahendus pole mõistlik arvestades seda, et on palju platvormiüleseid alternatiive.

Cocos Creator ei pakkunud häid lahendusi esimesele kahele kõige tähtsamale nõudele. Seega jääb ka see platvorm valikust välja.
Kuna Corona on ainult teek ja integreeritud visuaalne keskkond on mängu arendamisel tähtis, ei kaaluta edasi platvormi kasutamist. Corona andis hiljuti ka teada, et nad muudavad oma mängumootori vabavaraliseks ja loobuvad ise selle haldamisest. Corona ülalhoidmisega hakkavad tegelema mõned endised töötajad hobi korras [10]. Seega Corona tulevik on pigem ebakindel.
Samamoodi ei saa edasi minna libGDX analüüsiga – kuna puudub platvormiga integreeritud visuaalne keskkond objektide paigutamiseks.

2.2.5 Antud projekti jaoks sobivad lahendused

Defold, GameMaker Studio 2, Unity ja Godot on kõik variandid, millega autori arvates oleks töö teostatav. Platvorme hinnati selles etapis suuresti selle põhjal, kui kerge on neid kasutada. Täpsemalt – kui palju lisasamme nõuab mingi nõude täitmine.
Koheselt võib valida välja Unity tema suure populaarsuse, ohtra abimaterjali ja hea sobivuse tõttu. Kuigi Unity on kõige kallim siis pakub ta ka kõige rohkem. Järgnevalt on plaanis leida ülejäänud kolme seast veel üks, mida edasi uurida.

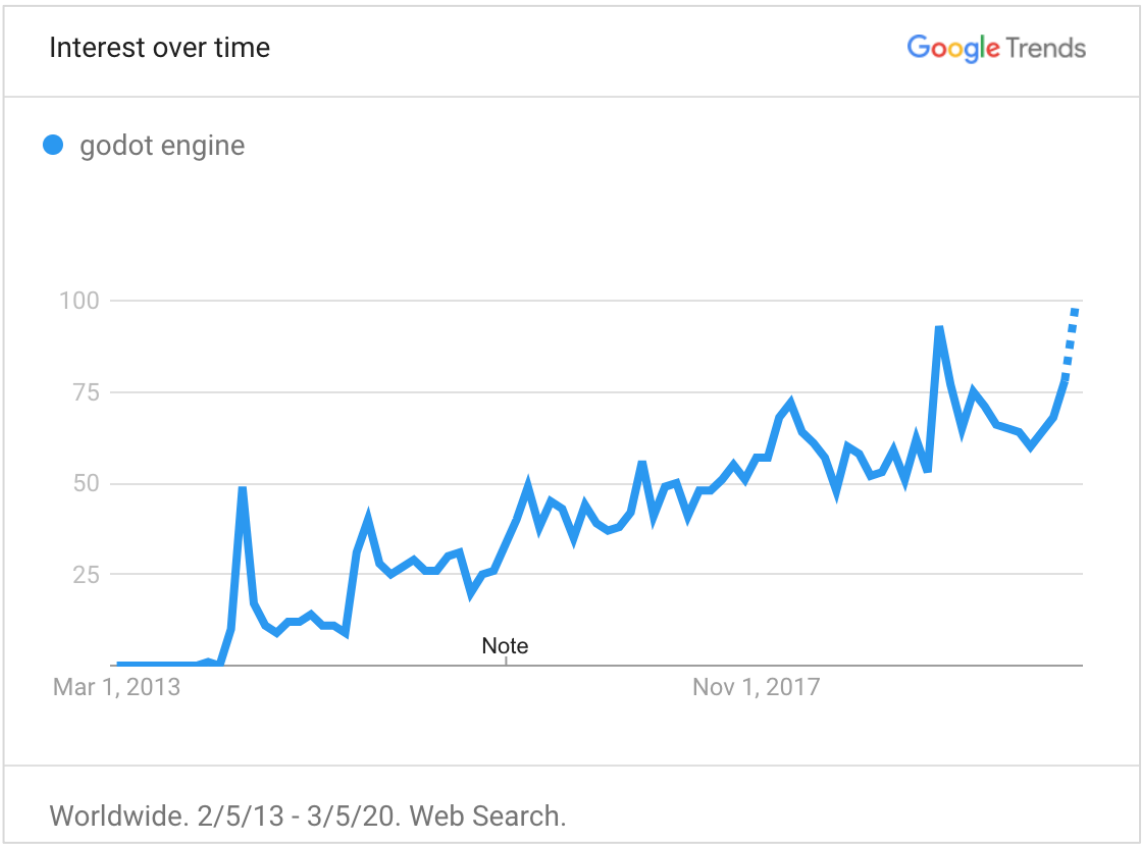
Tehes sotsiaalmeediakontode ülevaatust (Joonis 4) selgus, et kõige väiksem huviliste arv on Defold arendusplatvormil. Defoldi jaoks on olemas reklaamivahendaja laiendus, millel aga puudus reklaamitahvlite näitamise tugi seisuga 5. märts 2020 [13]. Oli olemas dokumentatsioon ise kolmanda osapoole teekide kasutamiseks. Puuduseks oli kasutusel olev programmeerimiskeel. Nõutud oli staatilise tüübikontrolliga keel aga toetus sellise keele kasutamiseks oli vähene või puudus. Seetõttu antud projektis seda platvormi pole plaanis kasutada.
Suurt kasutajate hulka omab GameMaker Studio 2, mille juured ulatuvad tagasi aastasse 1999 [15]. GameMaker Studio 2 võimaldab kasutada Drag And Drop süsteemi ja teha mänge ilma koodi kirjutamata [14]. GameMaker Studio 2 puhul oli reklaamide kuvamise laienduse hind võrdlemisi kõrge ja leidus üksainus tegija. Programmeerimiskeeleks oli ainult GML ja testimise tööriistad olid küllaltki algelised ja väheste võimalustega. Uurimisel selgus, et ükski neist ei võimaldanud kirjutada näiteks simuleeritud objektidega teste. Nimetatud puuduste tõttu ei osutunud see platvorm valituks.



Joonis 4 – Kasutajate arv erinevate arendusplatvormide sotsiaalmeedia kontodel

2.2.6 Godot ülevaade

Godot sai alguse 10 aastat tagasi kahe argentiinlase Ariel Manzur ja Juan Linietsky poolt. Hiljem sai sellest avatud lähtekoodiga mänguarendusplatvorm. Algul kasutasid platvormi loojad seda ise mängude arendamiseks. Huvi mängumootori vastu on stabiilselt kasvanud (Joonis 5).



Joonis 5 – Üleilmne huvi Godoti vastu veebiotsingute põhjal aastatel 2013 – 2020 [64]

Samuti viitab populaarsusele kõrge koht Githubi projektide pingereas, kus Godot asub 207. kohal (seisuga 06.03.2020) [16] . Aastal 2019 oli Godot 100 kõige väärtuslikuma projekti hulgas [17].
Viimase stabiilse versiooni (3.2) arendamisest võttis osa 450 kaastöötajat, kelle hulgast 300 tegid seda esmakordselt. Kümne kuuga tegid arendajad kokku 6000 sisestust [18]. Hetkel saab Godot Patreon platvormi kaudu iga kuu toetusi summas 10 998 dollarit (seisuga 06.03.2020), mis moodustab 92% põhiarendajate küsitud summast [19]. Eelneva põhjal järeldab autor, et Godot ei ole lähitulevikus kadumas ja selle arendamine jätkub ka tulevikus.

Godotiga on Androidile kokku arendatud üle 394 mängu [20].

Programmi failimaht on kõigest 74 MB (Standard versioon 3.2.1, macOS). Saadaval on *Standard* versioon, kus mängu arendada võimalik GDScript keeles. Valikus on ka *Mono* versioon, mis võimaldab C# kasutamist.
Näiliselt vähe võimalusi pakkuv kasutajaliides aitas autoril programmis lihtsasti orienteeruda ja aru saada kasutatavate piirkondade funktsioonidest.
Kõige ülemisel valikuribal on ainult viis valikut: *Scene*, *Project*, *Debug*, *Editor* ja *Help*.

Editori on sisse ehitatud koodiredaktor seega puudub otsene vajadus kasutada eraldi programmi. Koodiredaktoris oli võimalik avada dokumentatsioon kasutatavate klasside peal klikkides, mis oli mugav. Editori on sisseehitatud ka lokaliseerimise võimalus.

Suurimaks puuduseks on ehk keerulisemate näidete puudumine dokumentatsioonis selle kohta, kuidas kolmanda osapoole teeke kasutada. Samas on seda raske nõuda kuna laiendused on erinevad. Üldiselt on dokumentatsioon korralik ja sisaldab palju põhjalikke õpetusi.

Probleemide korral leiab abi Discordi jututoast ja foorumist.

2.2.7 Unity ülevaade

Unity loodi Kopenhaagenis Nicholas Francis, Joachim Ante ja David Helgasoni poolt aastal 2002 kui Francis otsis OpenGL foorumis arendajaid oma projekti tarvis [21]. Enda säästude ja tuttavate investeeringu toel tuli aastal 2005 välja mäng „GooBall“, mille eesmärgiks oli mängumootorile reklaami tegemine. Samal aastal tuli välju Unity 1.0 [22]. Aastaks 2008 oli platvorm juba rohkem arenenud. Ettevõttel tekkis võimalus programmi müügist saadud tulu kasutades palgata uusi töötajaid [23].

Üks murdepunkte oli 2008. aasta keskpaik, kui Apple avas oma rakenduste müümise keskkonna. Unity oli esimene mänguarendusplatvorm, kes suutis toetada eksportimist iPhonele juba sama aasta lõpuks. Samal aastal hakkasid kasutama Unityt ka Cartoon Network, Microsoft, Ubisoft ja Electronic Arts.

Hetkel keskendub Unity muuhulgas jõudluse suurendamisele. DOTS ehk Data-Oriented Technology Stack keskendub objektide asemel andmetele [24]. Üks mängustuudio on selle tehnoloogia kasutamisel saavutanud 2250x paremaid kiirusi suure hulga objektide kuvamisel [25].

Pool kõikidest mängudest on tehtud Unity platvormil [26]. Seega ei teki raskusi ka populaarsete mängude leidmisega. Näiteks „Heartstone“ ja „Monument Valley“ [27].

Kui välja arvata paigaldamine siis on programm igati kasutajasõbralik. Dokumentatsiooni on palju. Avaldatud on väga palju raamatuid erinevatele oskustasemetele ja leidub hulgaliselt õpetusi YouTubes ja mujal.

Üks puudus on see, et tegemist on tasulise platvormiga, kui soovitakse vältida Unity logo kuvamist ennem mängu laadimist. Selline kindel soov ettevõttel oli. Teine puudus on see, et programm nõuab arvutilt rohkem ressursse ja selle kasutamine on mõnevõrra aeglasem võrreldes Godotiga. Autori arvates nõuab Unity arendajalt rohkem süvenemist kui Godot.

2.2.8 Ressursside kasutuse hindamine

Arvuti ressursside kasutust programmide käitamise ajal aitas hinnata psrecord [62] programm, mille abil saab salvestada protsesside mälu ja protsessori aktiivsust. Arvuti andmed, kus platvorme käitati, leiab Tabelist 5.

Tabel 5 – Seadme andmed, kus programme käitati

Mudel	MacBook Pro (Retina, 15-inch, Mid 2015)
Operatsioonisüsteem	macOS Catalina (Versioon 10.15.3)
Protsessor	2,5 GHz Quad-Core Intel Core i7
Mälu	16 GB 1600 MHz DDR3
Graafikakaart	Intel Iris Pro 1536 MB

Hinnangu andmiseks loodi Unity ja Godot abil samaväärsed triviaalsed testprogrammid. Testprogrammid, mis loodi, võimaldavad nupu vajutamisel eelmise pildi peita ja järgmise pildi tuua nähtavale. Mõneti olid need sarnased sellega, mida oleks vaja luua reaalses rakenduses.

Monitooring teostati 10 minuti jooksul. Kolmandik ajast loodi eelkirjeldatud testprogrammi ja kuni sessiooni lõpuni valminud programmi käitati.

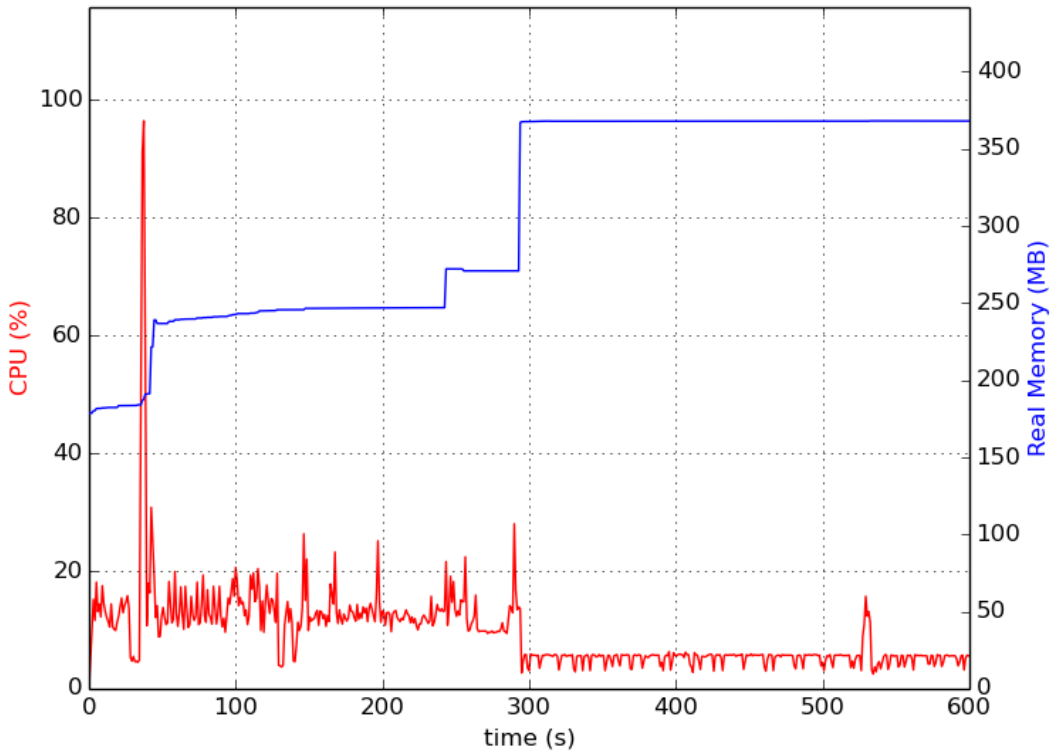
Andmeid koguti programmide põhiprotsessi ja selle alamprotsesside kohta. Intervall andmete salvestamisel oli 1 sekund. Logifailidesse salvestatu põhjal arvutati mälu kasutuse ja protsessori aktiivsuse aritmeetilised keskmised. Ennem igat sessiooni oli võrreldavatele saadaval samas koguses ressursse. Teostati mitmeid monitooringuid ja märkimisväärseid erinevusi ei esinenud.

2.2.9 Godot mälu ja protsessori kasutus

Jälgiti Godot versiooni 3.2.1 (Standard) ressursside kasutust. Testprogrammi loomiseks kulus ligikaudu 300 sekundit. Pärast valmimist testprogramm käivitati. Joonisel 6 on see hetk näha mälukasutuse suurema kasvu kujul. Keskmised väärtused on Tabelis 6.

Tabel 6 – Godot keskmine mälu ja protsessori kasutus

Programm	Godot
Mälu MB	305
Protsessor %	9



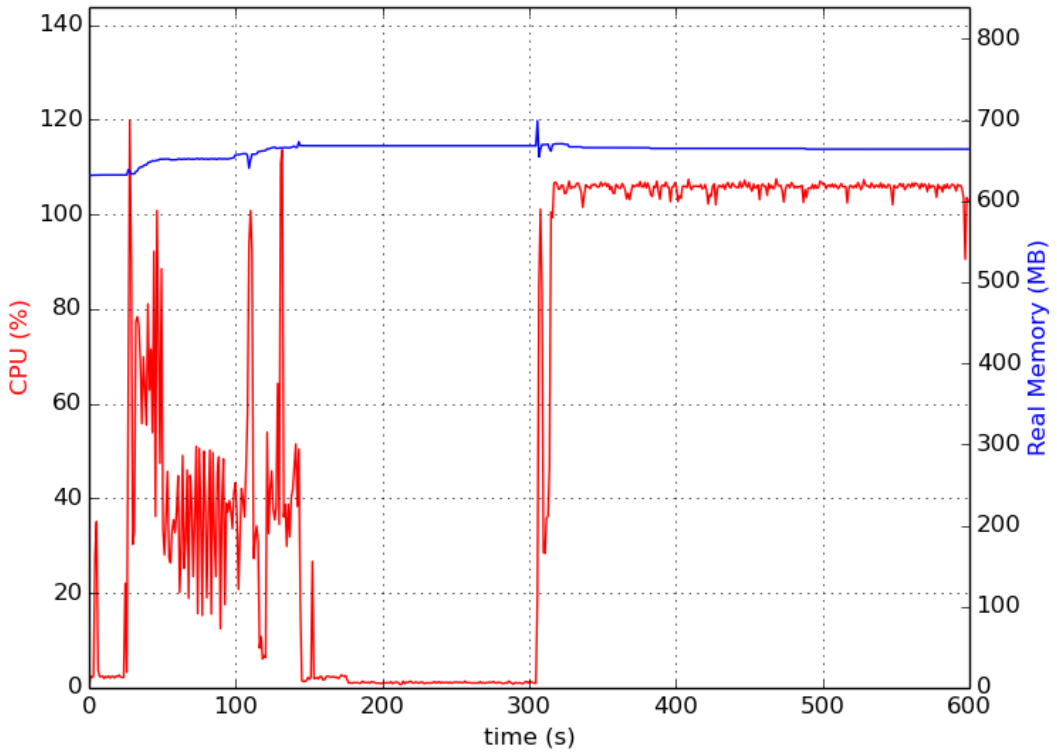
Joonis 6 – Godot mälu ja protsessori kasutus

2.2.10 Unity mälu ja protsessori kasutus

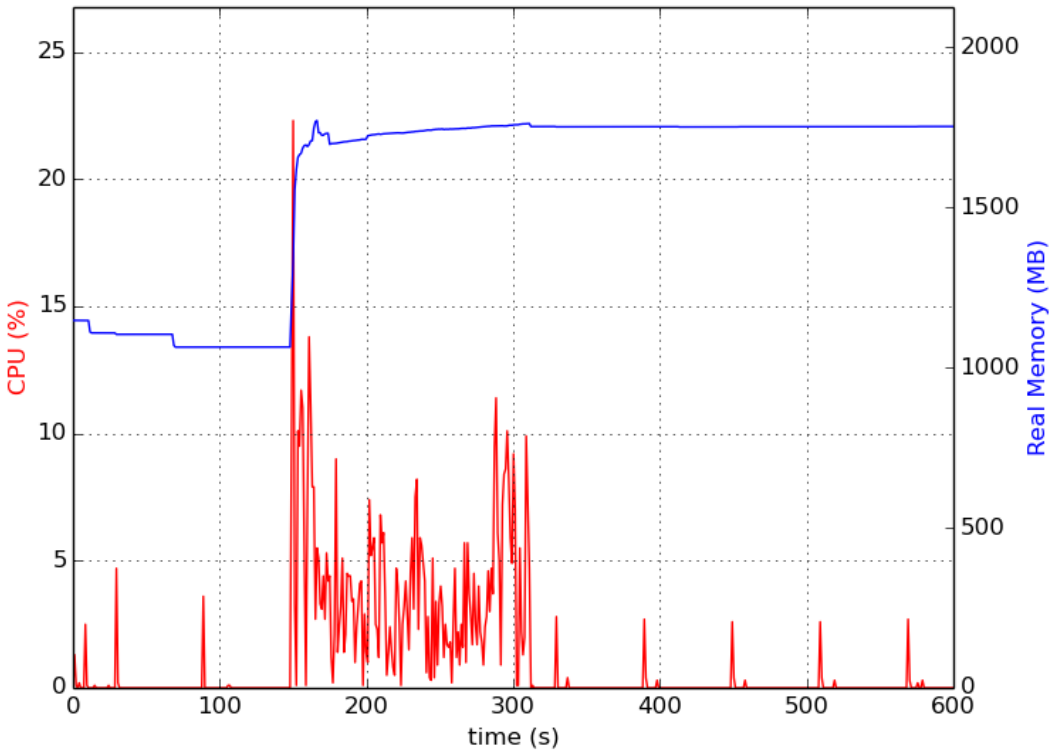
Jälgiti Unity versiooni 2019.3.13f1 ressursside kasutust. Kuna koodiredaktorit polnud Unity programmile sisse ehitatud tuli kasutada välist rakendust. Selleks valiti Visual Studio Code. Ühtlasi nõudis Unity kasutamine Unity Hub käitamist (macOS), mis on mõeldud projektide haldamiseks. Tulemused kolme eraldiseisva programmi mälu ja protsessori kasutuse kohta on toodud vastavalt lõigus esinemise järjekorrale Joonistel 7, 8 ja 9. Monitooringu perioodi keskmised väärtused on Tabelis 7.

Tabel 7 – Unity ja kaasprogrammide mälu ja protsessori kasutus

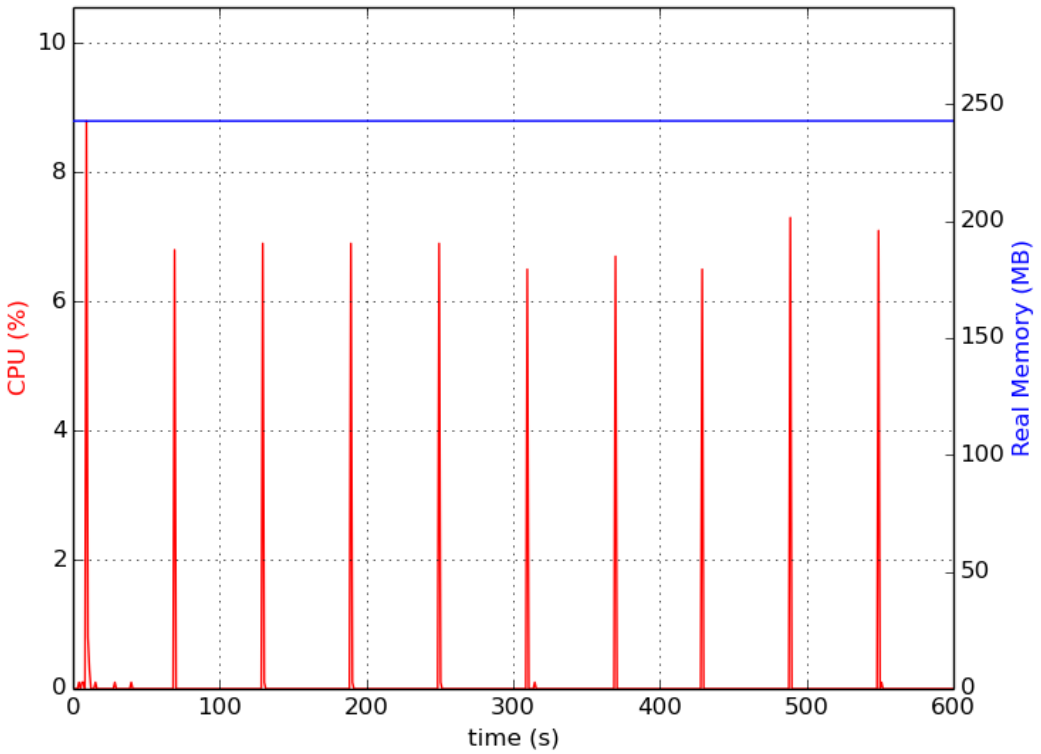
Programm	Unity	Visual Studio Code	Unity Hub
Mälu MB	662	1577	243
Protsessor %	60	1	0



Joonis 7 – Unity mälu ja protsessori kasutus



Joonis 8 – Visual Studio Code mälu ja protsessori kasutus



Joonis 9 – Unity Hub mälu ja protsessori kasutus

2.2.11 Valik ja põhjendus

Kuna rakendus, mida üle viiakse on graafiliselt väga lihtne ja ei nõua palju keerukust siis Unity järele otsene vajadus puudus, kuna see platvorm on pigem mõeldud palju rohkem jõudlust ja keerukust nõudvate mängude arendamiseks. Samuti küsib Unity programmi kasutamise eest kohatasu, mis eelmise aastal (2019) suurenes 399 dollarini aastas. On tarvis rõhutada, et tegemist on kohatasuga ja minimaalselt oleks ettevõttel tarvis kaks kohta. Kui ettevõtte otsustab tellida sisse uusi mänge või tasemeid olemasolevatele mängudele, siis tuleb litsentse juurde soetada. Summad oleksid ettevõttele ehk vastuvõetavad, kui litsentse tuleks soetada ainult mõned – sealt edasi oleksid lisakulud juba märgatavad. Godotiga sellist probleemi ei kaasneks.

Mitmete ressursside kasutuse monitooringute põhjal oli näha, et Godot kasutab oluliselt vähem arvuti mälu ja protsessorit. Ressursside kokkuhoid on tähtis, kuna üleviimine eeldab paralleelselt vähemalt kahe teise programmi kasutamist: Adobe Photoshop ja Adobe Animate.

Seega valiti Godot tema tasuta kättesaadavuse ja vähesema ressursside kasutuse tõttu programmiks, millega migratsioon teostada.

3 Lahenduse disain

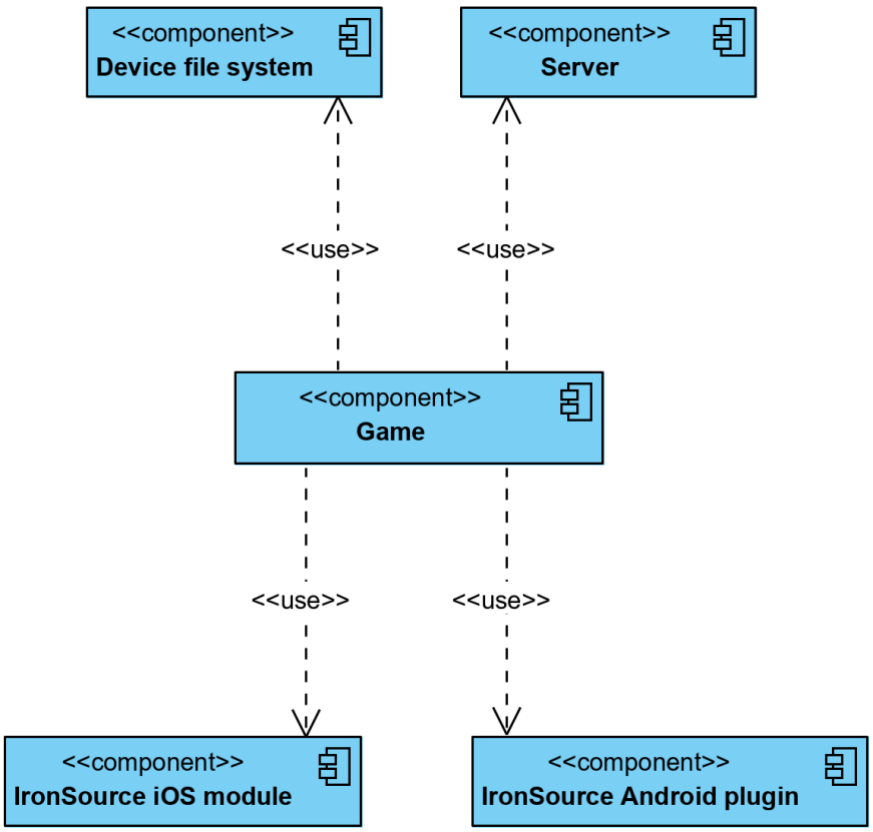
3.1 Ülevaade

Joonisel 10 on toodud peamised mängu komponendid. Järgnevalt lühikirjeldus komponentide ülesannetest.

iOS platvormi jaoks arendatakse reklaamide kuvamiseks moodul (IronSource iOS Module) [28] ja Androidi jaoks *plugin* (IronSource Android Plugin) [29].

Seadme failisüsteemi (*Device File System)* kasutatakse selleks, et salvestada sinna mängu seisu. Diplomitöös täpsemalt salvestus -ja laadimisprotsessi ei käsitleta. Server on selleks, sealt alla laadida info ja pildid mängusiseste reklaamtahvlite jaoks. Ka seda osa rohkem ei käsitleta.

Punktis 3.2 on täpsemalt kirjeldatud põhikomponenti *Game*.



Joonis 10 – Ülevaade rakenduse osadest

3.2 Mängu komponendid

Komponendid, mida diplomitöös arendatakse ja mis kokku moodustavad nõuetele vastava mängu, on kaks – mäng ja *plugin*. iOS mooduli arendus jääb skoobist välja.

3.2.1 Mäng

Kuna paljude funktsioonide järgi on globaalne vajadus siis on paigutatud vajaminev kood erinevatesse *singletonidesse*. *Singleton* on juba Godotis teostatud ja autoril polnud seda vaja teha [61]. Godoti *singletonid* on *stseenipuus* rakenduse elutsükli jooksul alati kättesaadavad. Järgnevalt on välja toodud Joonisel 11 kujutatud mängu struktuuri klassid ja nende ülesanded.

Ads - siin teostatakse IronSource *plugina* meetodid ja *callbackid GDScriptis*.

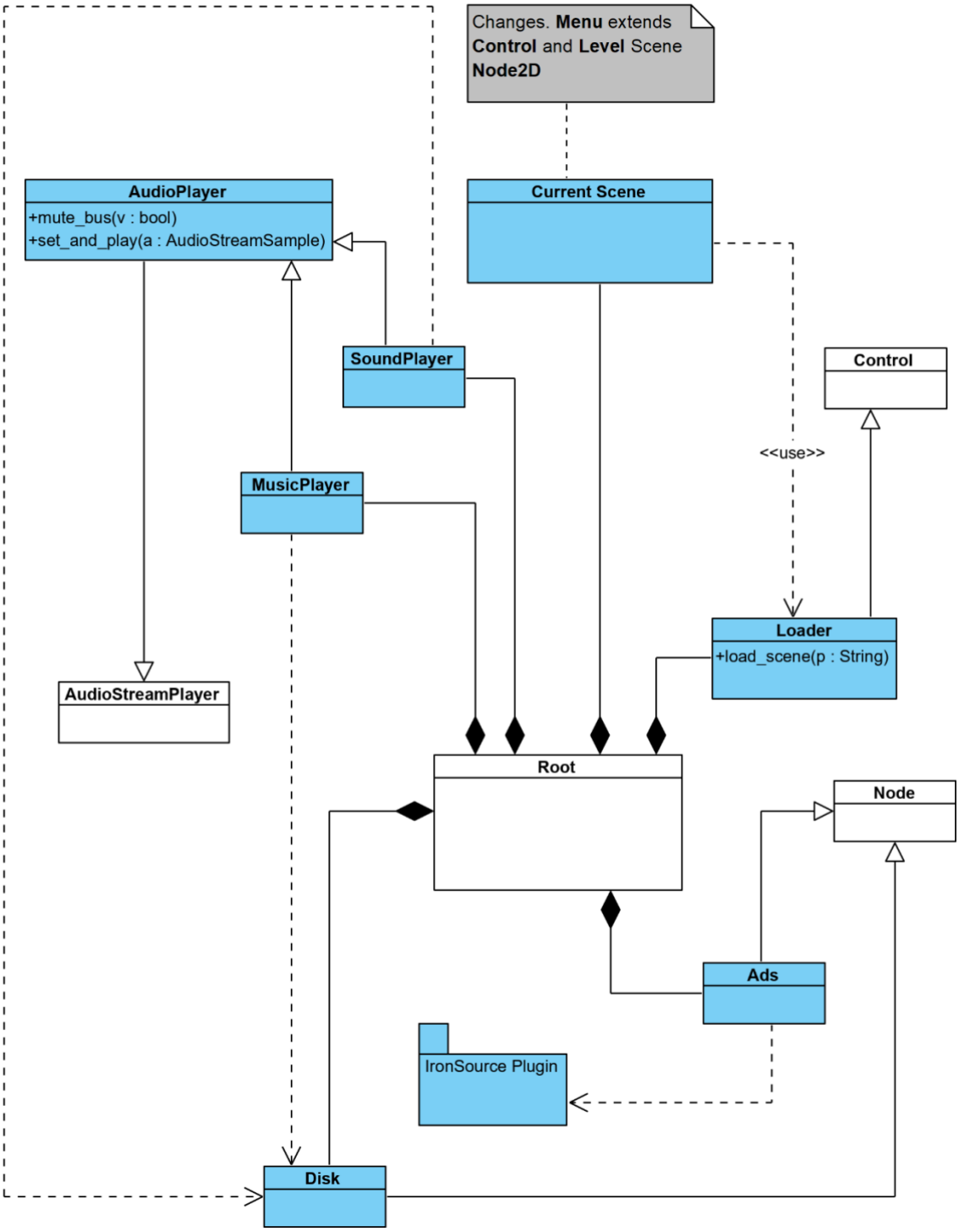
MusicPlayer - AudioPlayeri instants, mis mängib muusikat „Music“ nimelises helikanalis. Vastavalt kasutaja valikutele on võib „Music“ helikanal olla summutatud ning muusikat ei mängita.

SoundPlayer - AudioPlayeri instants, mis mängib heliefekte „Sound“ nimelises helikanalis. Ka „Sound“ helikanal võib-olla summutatud ning sel juhul ükski heli, mida SoundPlayeri *set_and_play()* meetodi abil mängida proovitakse seadme kõlaritest ei kostu.

Loader - stseen, mis laeb järgmise stseeni ressursid ette ja kuvab kasutajale progressi.

Disk - sisaldab meetodeid seadme kettale kasutaja valikute salvestamiseks (näiteks heli ja muusika seaded)

Current Scene – Stseen, mis vahetub Loaderi abil. Antud projektis võib stseen olla kas menüü või *level*.

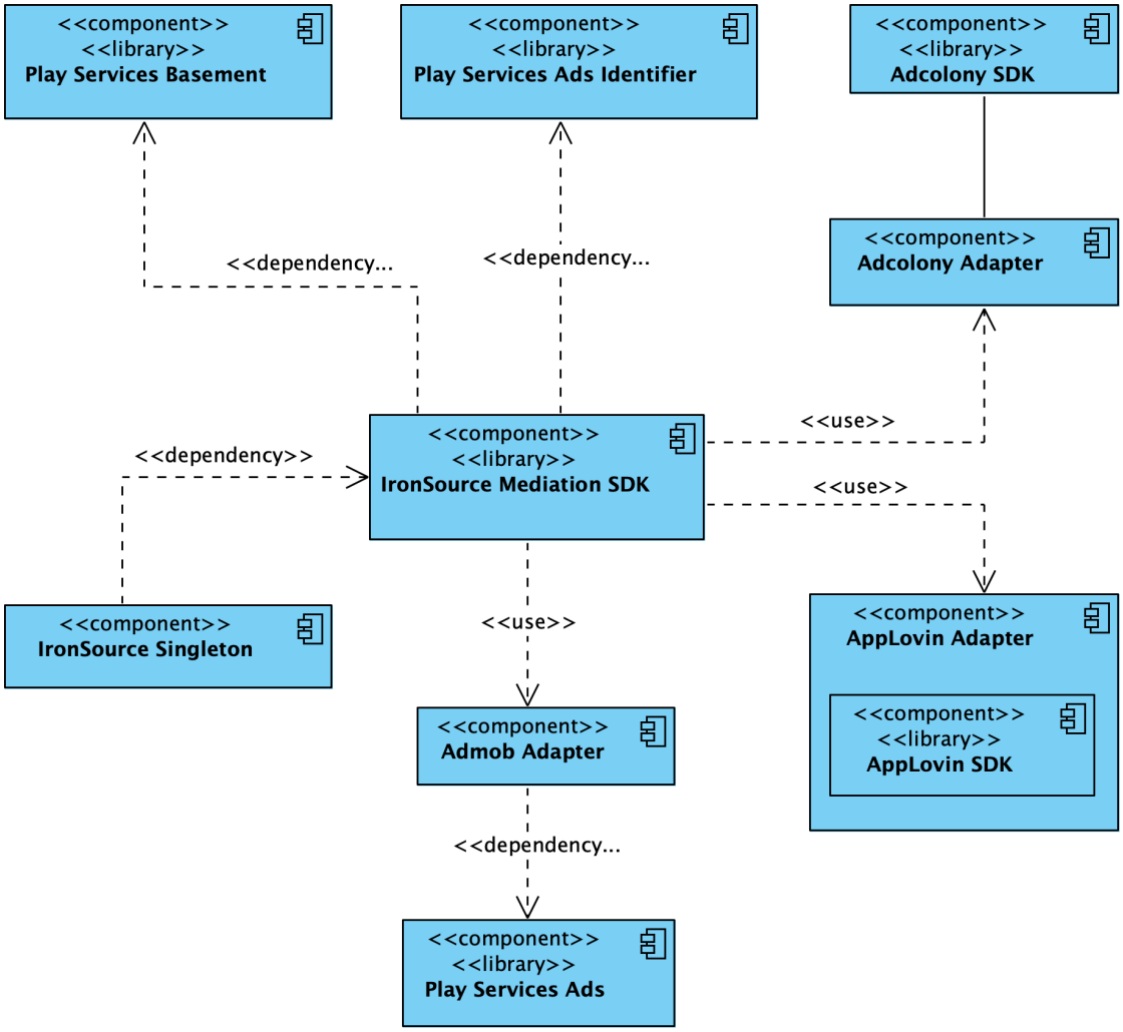


Joonis 11 – Loodava rakenduse struktuur Godot projektis

3.2.2 IronSource Androidi plugin

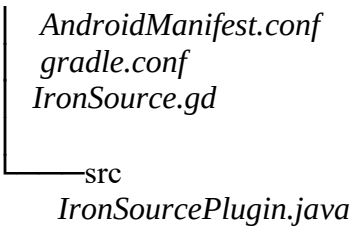
Plugin on vajalik Androidi IronSource SDK kasutamiseks Godoti mängus, et kuvada reklaame paljude erinevate reklaamivahendajate kaudu.

IronSource SDK võimaldab reklaame kuvada 14 erineva reklaamivõrgustiku vahendusel [30]. Ettevõttel on huvi paigaldada nende seast 3. Iga reklaamivahendaja teegi jaoks on IronSource arendanud *adapteri*, mis aitab IronSource Androidi teegil teiste reklaamivõrgustike teekidega suhelda [31].



Joonis 12 – Loodava Android plugini osad ja nende sõltuvused

Joonisel 12 kujutatud Androidi plugin laetakse mängu *singletonina.* Struktuuri poolest võib *plugin* olla nagu Androidi projekt, mis kasutab Gradlet. Antud projekti jaoks arendatud *plugina* puhul oli minimaalselt vajalik kõigest 3 faili. *IronSource.gd* pole plugina töötamiseks vajalik vaid on mugavus lõppkasutajale. Info laienduse arendamiseks oli allikas [29] aga *plugina* valmimise ajaks oli pakutud uus lahendus ja vana pole enam saadaval (12.05.2020).



AndroidManifest.conf – see fail lubab sisestada teksti põhiprojekti AndroidManifest.xml faili. Märgetest lähtuvalt sisestatakse tekst õigesse kohta põhiprojekti ehitamisel.

gradle.conf – see fail lubab sisestada teksti põhiprojekti build.gradle faili. Märked selles failis aitavad sisestada teksti õigesse kohta põhiprojekti build.gradle failis.

IronSourcePlugin.java – *singleton*, kus asuvad IronSource SDK teostamiseks vajalikud meetodid.

IronSource.gd – *singleton*, mida kasutatakse Godot siseselt, et *pluginat* eesmärgipäraselt kasutada.

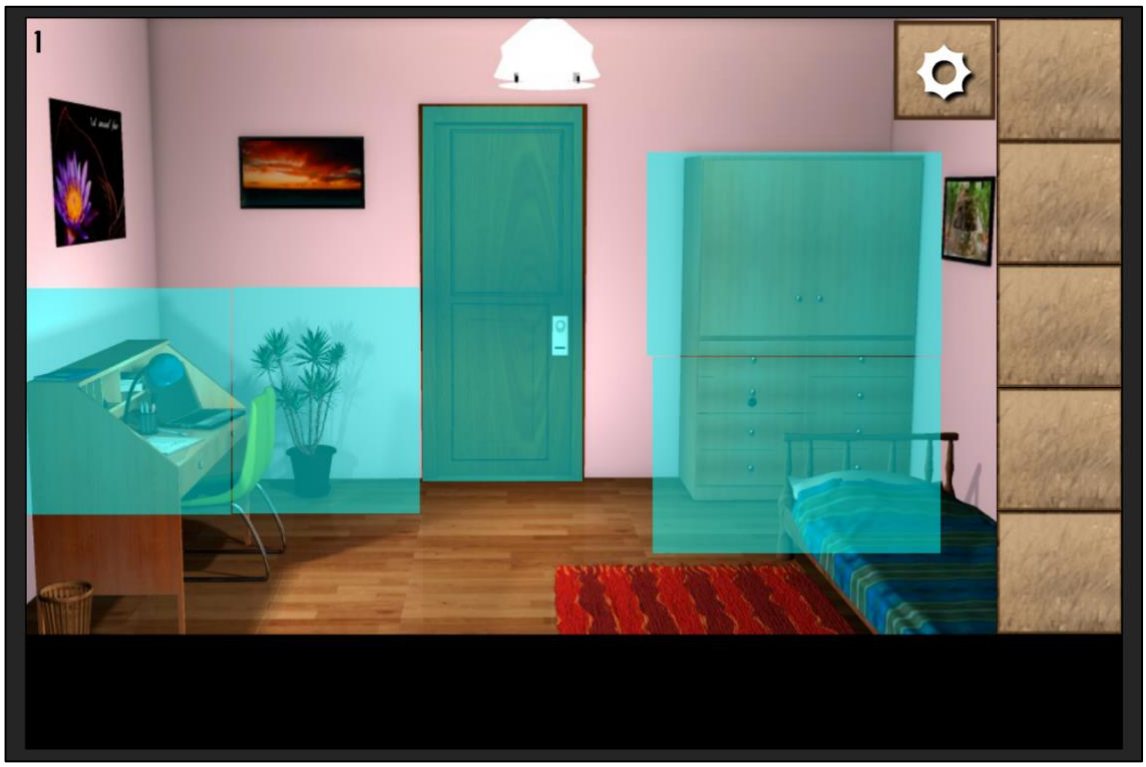
4 Teostus

4.1 Ületoomise lihtsustamine

Et mingil määral automatiseerida mängu ületoomist endisest arenduskeskkonnast valmis paar tööd hõlbustavat abivahendit. Godot pakub mitmeid võimalusi kuidas seda teha. Antud ülesannete puhul oli selleks *EditorPlugin* loomine. *EditorPluginit* kasutab arenduskeskkond, et selle funktsionaalsust laiendada. Tavaliselt kasutatakse neid elementide või ressursside modifitseerimiseks [38].

4.1.1 Klikkimisalade ületoomine

Mängu üks põhilisemaid tegevusi on aladel klikkimine või siis ekraani puudutamine, mille tulemusel jõutakse edasi teise vaatesse, avatakse mõni mõistatus või korjatakse üles mujal kasutatav ese. Joonisel 13 näha olevad rohelised alad on mängus klikitavad.

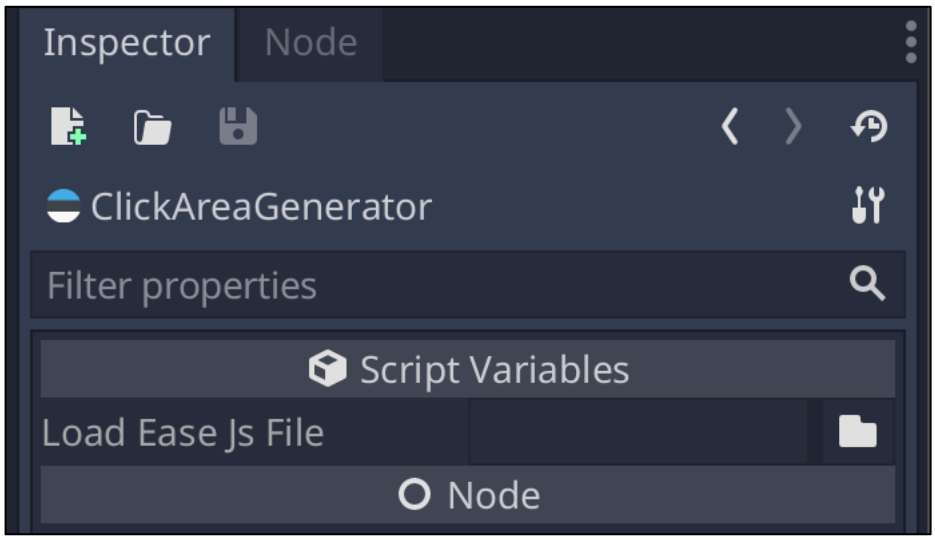


Joonis 13 – Mängus klikitavad alad

Kuna alasid on palju ja neil kõigil on oma kindel suurus ja positsioon siis ükshaaval koordinaate ja suurusi ümber trükkida näis tülikas. Arvestades ka tulevaste sarnaste mängude migreerimisvajadust otsustas autor seda korduvat tegevust veidi kiirendada.

Valminud tööriist võtab sisendina vastu vanast Adobe Animate projektist eksporditud alade *spritesheeti* EaselJS teegi kaasfaili [40]. Katsetades erinevate ekspordi sätetega selgus, et ainult sellest ekspordi tüübi väljundist, täpsemalt JavaScripti failist, võib leida ainukesena nii alade suuruse kui ka asukoha. Pärast faili töötlust Godot projektis luuakse välja otsitud info põhjal alade jaoks *blokid*, mille suurus ja positisioon on identsed Adobe Animate projektis olevate vastavate atribuutidega. *Plugin* eeldab, et nii Adobe Animate kui Godoti projekti baasresolutsioon on sama.

Valminud tööriista kasutamine on lihtne. Esiteks tuleb eksportida soovitud alad Adobe Animate projektist ning seejärel laadida JavaScripti fail tööriista kaudu nagu näha Joonisel 14. Uued elemendid luuakse automaatselt ja on valmis edasiseks toimetamiseks. Tööriist osutub kasulikuks just selliste vaadete puhul kus on palju interaktiivseid alasid.



Joonis 14 – Klikitavate alade loomise tööriist

4.1.2 Kasutatavate elementide loomine

Mängus samaväärselt suur osa on inventaris esemete kasutamine, kombineerimine ja vaatamine. Mõnel juhul saab eset kasutades luua uusi esemeid ja vahel saab eseme pealt välja lugeda vihjeid mõistatuste lahendamiseks. Pilt inventarist Joonisel 15.

Joonis 15 – Inventari näidis

Kuna inventar on kesksel kohal ja pidevalt kasutaja silme ees siis on tähtis, et esemed oleksid korrektselt neile määratud pesadesse paigutatud.

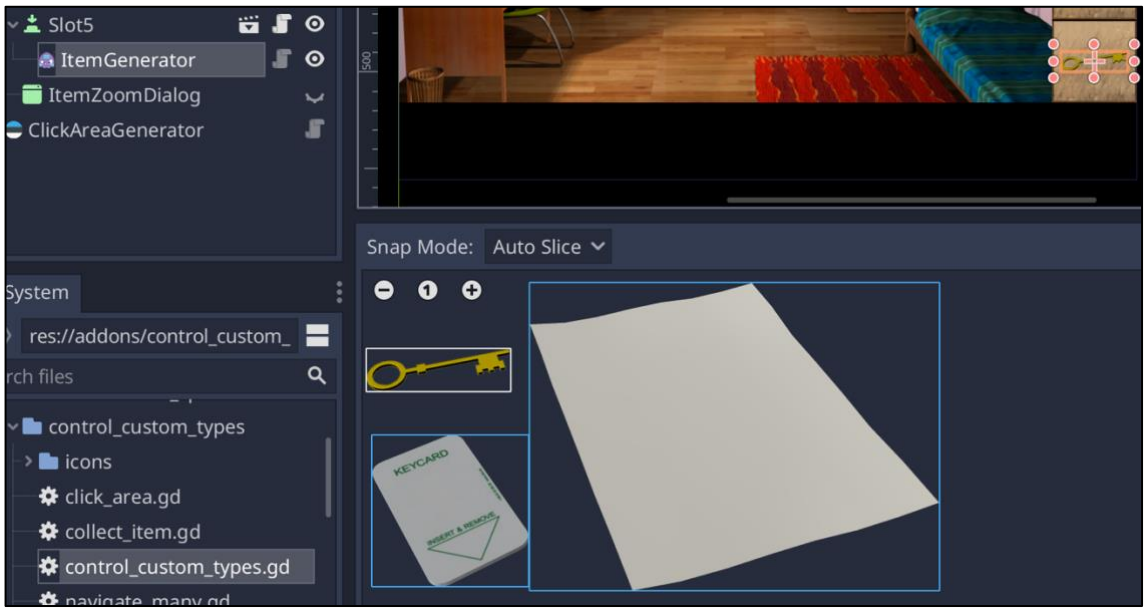
Autor leidis mängu algprojekti kaustadest igale tasemele vastava esemete üldfaili. Tegemist on PNG formaadis failiga, kus esemed on ühes vaates kõik koos. Fail on sarnane lihtsamat tüüpi *spritesheetiga* aga erinevus on selles, et pildid failis on erisuurusega [42]. See fail osutus kasulikuks, sest Godot pakub sisse ehitatud tööriista, mille abil on võimalik sellisest failist objekte välja lõigata. Kirjeldatud tegevus on näha Joonisel 16.

Joonis 16 – Godoti automaatne tükeldamise tööriist

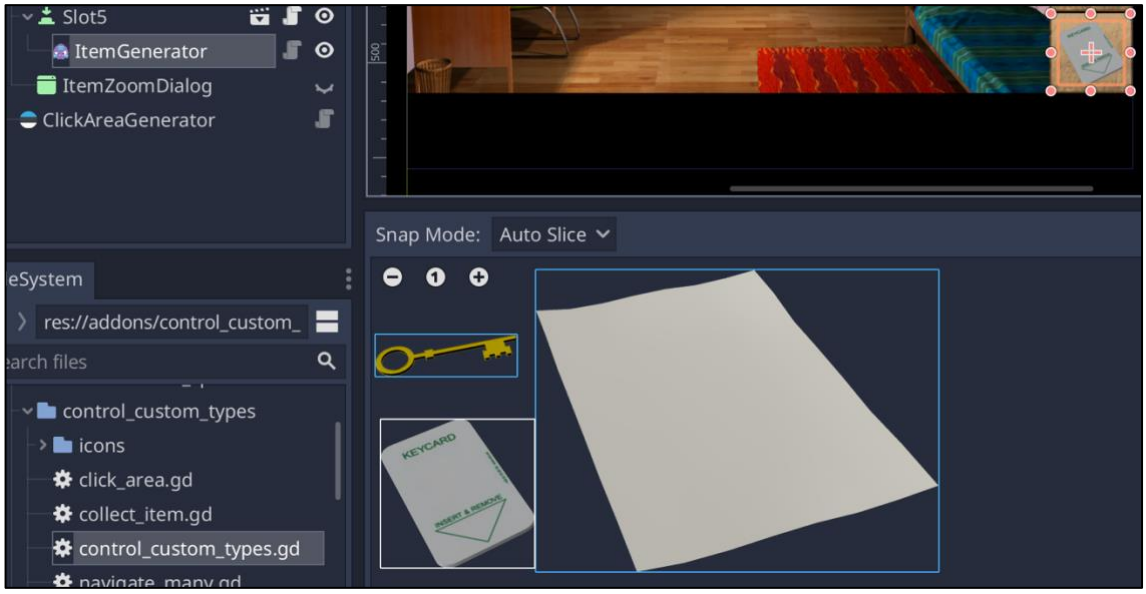
Mida sisse ehitatud polnud ja autoril vähemalt peale mõningast proovimist ei õnnestunud saavutada oli väljalõigatud eseme suuruse automaatne muutmine nii, et see paigutuks täpselt pesasse ja säilitaks oma külgede pikkuste originaalse suhte. Arvestama pidi sellega, et esemed on erisuuruses.

Selle probleemi lahendamiseks valmis veel üks *EditorPlugin*, mis protsessi lihtsustab. Tööriista kasutades toimub eseme paigutus automaatselt. Seejärel saab eseme salvestada valitud kausta, et seda rakenduse koodis hiljem kasutada. Erinevalt eelmisest alade tööriistast kiirendab see abivahend märgatavalt tehtava ülesande sooritamist ka siis, kui esemeid pildil on ainult üks või kaks. See tuleneb sellest, et ühe eseme loomine ja paigutamine nõuaks manuaalselt tehes palju väikeseid liigutusi.

Tööriista kasutamisel ei tule teha muud kui valida vastav regioon ja ese paigutatakse automaatselt pessa (*Slot5)* nagu näha Joonisel 17. Seejärel saab eseme salvestada stseenina ressursside alla. Hiljem saab eset pessa lisada koodist ja puudub vajadus paigutamiseks. Pärast eseme salvestamist saab valida järgmise eseme nagu näha Joonisel 18. Niimoodi jätkates on esemete toomine projekti väga kiire.



Joonis 17 – Esemete loomise tööriista kasutus (võti)



Joonis 18 – Esemete loomise tööriista kasutus (uksekaart)

4.2 *Leveli* baassüsteemid

Erinevate kombinatsioonide rohkuse tõttu osutus üheks keerulisemaks probleemiks mängu arendamisel *levelis* olevate vaadete vahel navigeerimine ning samuti leitud esemetega toimetamine.

Vanas projektis oli kõik valminud „käsitööna“. Iga *leveli* ülesehitus oli unikaalne ning korduvkasutatavaid osi oli vähe. Seetõttu üle tuua polnud võimalik midagi.

4.2.1 *Levelis* navigeerimine

Mängu analüüsides proovis autor leida kõik erinevad navigeerimisstsenaariumid. Selgus, et kui kasutaja klikib suvalisel alal mängu sees siis kliki tulemusel juhtub üks asi viiest. Iga erineva tegevusega ala jaoks loodi oma klass, mis pärineb *ClickArea* (Koodinäide 1) klassist, kus asub kõikide klikitavate alade ühine kood.

```
extends Control
class_name ClickArea

export(NodePath) var director_path: NodePath
export(Array, NodePath) var navigation_paths setget set_navigation_paths,
get_navigation_paths
export(NodePath) var affected_node: NodePath
var director

func _ready():
    add_to_group("persist")
    add_to_group("navigation_click_areas")
    if director_path:
        director = get_node(director_path)
        if director.has_method("area_clicked"):
            connect("gui_input", director, "area_clicked", [self])
    else:
        printerr("No director path set!")

func get_navigation_paths():
    return navigation_paths

func set_navigation_paths(arr: Array):
    navigation_paths = arr

func get_first_view() -> CanvasItem:
    return (get_node(navigation_paths.front()) as CanvasItem)

func remove_first_view() -> void:
     navigation_paths.pop_front()

func get_usable_item():
    return null

func set_usable_item(_value):
    pass

func get_class() -> String:
    return "ClickArea"

func save():
    var save_dict = {
        "node_path": get_path(),
        "class_type" : get_class(),
        "navigation_paths": navigation_paths,
    }
    return save_dict
```

Koodinäide 1 – *ClickArea* klass

Igal klassil on ka unikaalne tegevust väljendav ikoon, mis aitab arendajal neid lihtsamini eristada.

1. Liikumine järgmisesse vaatesse (kui kasutaja on põhivaates)

2. Liikumine järgmisesse vaatesse (kui kasutaja on alamvaates)

3. Liikumine järgmisesse vaatesse – millisesse, sõltub kolmandatest tegevustest

4. Eseme üles korjamine ning liikumine järgmisesse vaatesse

5. Eseme kasutamine ning liikumine järgmisesse vaatesse

Kuna klikitavate alade ja teiste *blokkide* erinevad koostoimimise viisid osutusid arvukaks ja keeruliseks siis tekkis vajadus luua keskne *blokk*, mis seda kõike juhib. Disainimustreid uurides näis sobiv lahendus olevat *mediator*, mis aitab muuhulgas grupil objektidel omavahelist suhtlust koordineerida [43]. Lisaks kasutati palju signaale, mis võimaldavad *blokkidel* välja saata teateid, mida teised *blokid* saavad kuulata ja nende vastuvõtmisel reageerida [55]. Signaalid on Godoti teostus *observer* disainimustrist [56].

Süsteemis on kõik klikitavad alad ühenduses *NavigationDirector* klassiga – see tähendab, et *NavigationDirector* kuulab pealt kõiki hiirega tehtud liigutusi, mis toimuvad klikitavate alade peal. Kui mõni liigutus toimub klikitava ala peal siis käivitatakse *NavigationDirector* klassis meetod *area_clicked(...)* (Koodinäide 2). Vastavalt meetodile parameetrina kaasa antud viitega klikitavale alale otsustatakse, millist tegevust on vaja teha eelpool kirjeldatud viiest erinevast. Sarnast lahendust on põgusalt käsitletud ka raamatus [43].

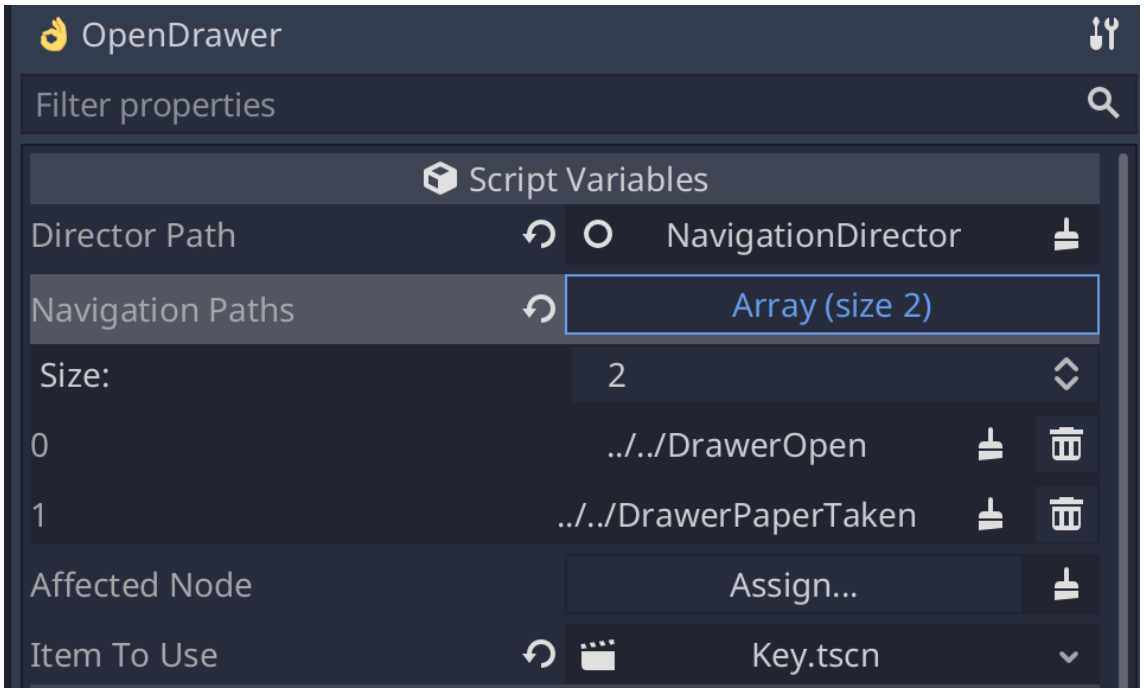
```
func area_clicked(e: InputEvent, sender: ClickArea):
    if is_screen_touched(e):
        match(sender.get_class()):
            "NavigateOne":
                navigate_one_mainview(sender)
            "NavigateOneSub":
                navigate_one_subview(sender)
            "NavigateMany":
                navigate_many_views(sender)
            "CollectItem":
                navigate_one_subview(sender)
                emit_signal("item_collected", sender.get_parent().get_path())
                if inventory_director.has_method("add_item"):
                    inventory_director.add_item(
                        (sender as CollectItem).get_item_sprite())
            "UseItem":
                if sender.get_usable_item():
                    use_item(sender)
                else:
                    navigate_many_views(sender)
```

Koodinäide 2 – *NavigationDirector* klassi meetod *area_clicked(...)*

Üks suuremaid erinevusi *mediator* raamatus soovitatud kuju ja antud projekti lahenduse vahel on see, et lahenduses klikitavad alad viitavad üksteisele. Raamatu soovitusi järgides, peaks ainult *NavigationDirector* klassis endas olema kirjeldatud objektide omavaheliste seoste tulemid [43]. Põhjus, miks autor kasutas modifitseeritud lahendust on see, et seoseid on mugavam luua klikitava ala peal. Selleks kasutati Godot võimalust klassi liikmeid eksportida [46], mis aitab seoseid luua graafilise liidese abiga. Näide, kuidas näeb arendajale välja klikitava ala ülesseadmine, asub Joonisel 19.

Seletuseks Joonisele 19. Klikitav ala võib viia mitmesse kohta olenevalt sooritatud tegevustest. Samuti võib olla määratud ese, mida seal peal kasutama peab. Kui klikitava ala peal toimub klikk siis *NavigationDirector* otsustab ja saadab teate edasi sündmusega seotud osapooltele.

Lahendus oli mugav uute alade kohandamisel erinevate *levelite* stsenaariumitega.



Joonis 19 – Klikitava ala ülesseadmine graafilise liidese abil

4.2.2 Inventari süsteem

Sarnaselt navigatsioonisüsteemile kasutab ka inventar keskset kontrollobjekti, mis juhib objektide olekuid. *InventoryDirector* omab viiteid kõikidele pesadele (kokku 5) ja kui nende peal toimub klikk siis otsus, kuidas käituda, langetatakse *InventoryDirector* klassis. Kokkuvõtvalt peab viimane otsustama, kas pärast klikki peaks eset suurendama, eseme ära kaotama (pärast selle kasutamist ruumis), eseme kombineerima teise esemega (ja ära kaotama) või mitte midagi tegema. Kui eset kasutatakse ruumis edukalt õiges kohas siis toimub lisategevus ka *NavigationDirectoris*, millele on *InventoryDirectoris* viide. Samamoodi on ka *NavigationDirectoris* viide *InventoryDirectorile* – seda läheb vaja näiteks siis, kui ese üles korjatakse ja see tuleb inventari lisada.

4.2.3 Android *plugin*

Plugina loomine ainult dokumentatsioonis oleva info põhjal oli keeruline. Kuna on palju erinevaid laiendusi, mida võib Godotile *plugina* näol luua, siis ametlikus õpetuses oli olemas ainult minimaalne informatsioon. Seda leidus siiski piisavalt. Kõige rohkem oli vaja teadmisi hoopis Android SDK-st ja Gradlest.

Kuna IronSource SDK teostus nõudis mitme reklaamivahendaja teegi kasutamist siis probleeme tekitas erinevate teekide ühildumatus.

Näiteks tekkisid konfliktid mängu ehitamisel Gradlega, mis andis veateateid kuna moodulites olid mõned klassid topelt. Esimese lahendusena, sarnaselt ühele teisele laiendusele [32], kasutas autor *exclude group* võimalust Gradles, millega saab pakke moodulist välja jätta. Algul see lahendus töötas, kuid kui autor soovis SDK versioone uuendada, tekkisid sarnased vead. Lähemal uurimisel selgus, et konflikt on vanematele Androidi versioonidele toetatavust pakkuva teegi ja uue AndroidX teegi vahel, mis sama ülesannet täidab. Lahendus oli üle minna AndroidX teegile. Seda soovitab ka Google ise [33]. Plugina arendamisel olid abiks sarnased lahendused [32] ja [44].

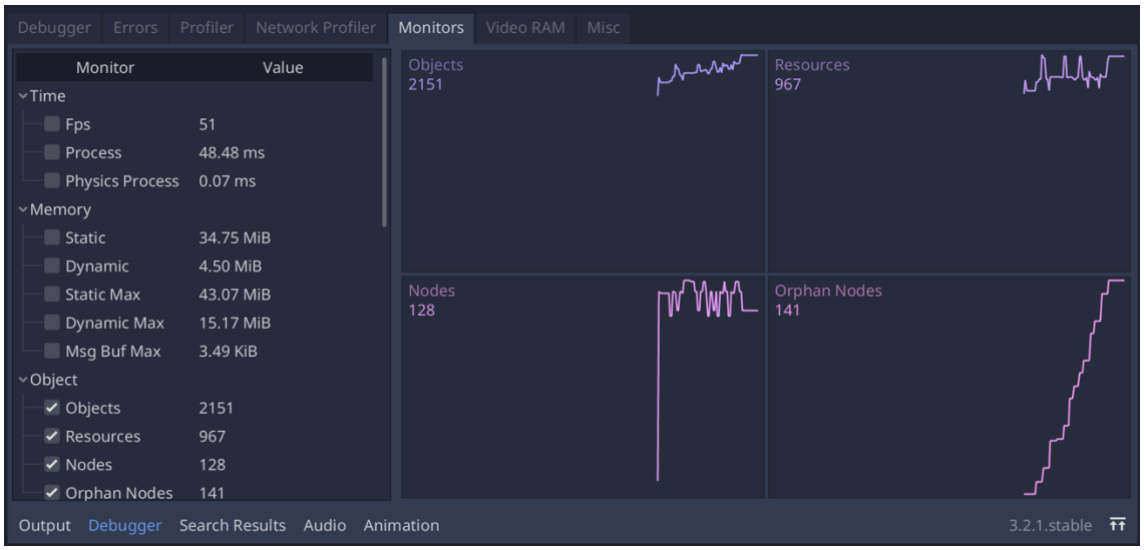
Selleks, et *plugin* oleks võimalikult mugav kasutada ka teistel arendajatel lõi autor Godot projektides kasutamiseks ühe *singletoni*, kus on kõik plugina poolt pakutavad meetodid ja *callbackid*. Võimalus on ka Godotis otse IronSource rakenduse identifikaator määrata. Samuti on võimalus määrata reklaamitüübid, mida aktiveerida. Kokku teostati kolm reklaamitüüpi. Plugin on avalikult saadaval Githubis koos õpetusega [45].

5 Kvaliteedikontroll

Kvaliteedi tagamisel keskenduti selles töös peamiselt mängusisestele klassidele *NavigationDirector* ja *InventoryDirector*. Autori kogemuse põhjal on enamus tagasisidest väljaloetud vigu seotud just sellega, et kasutajatel pole võimalik mängus edeneda mittetöötavate vaadete või esemete tõttu.

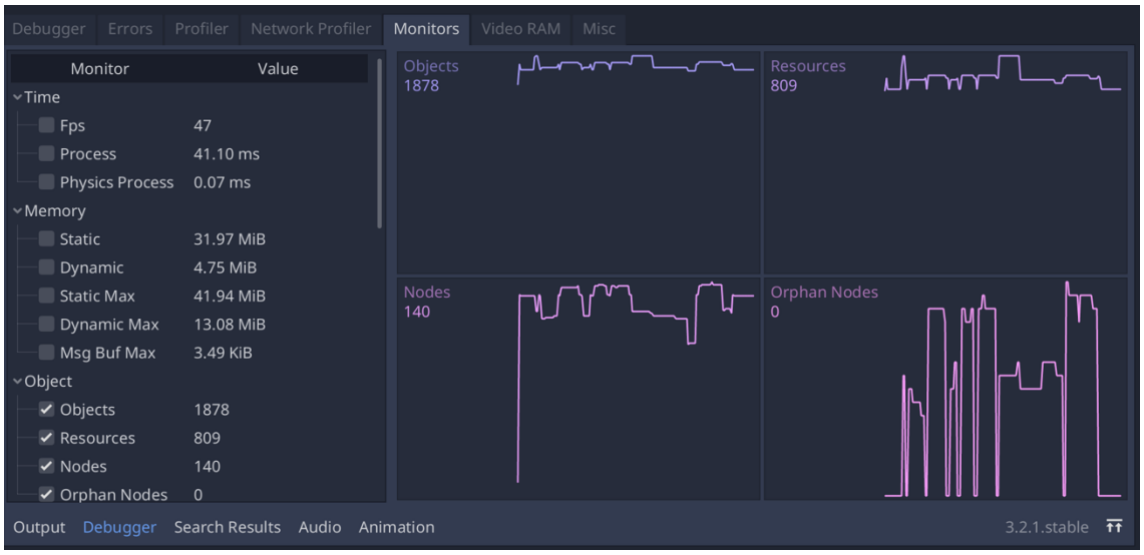
5.1 Profileerimine

Arenduse käigus kasutati regulaarselt Godot sisseehitatud profileerijat. Jälgides objekte ja orvuks jäänud *blokke* selgus, et peamenüü ja *levelite* vahel edasi tagasi liikumine suurendab nende arvu. Joonisel 20 olev *Objects* näitab loodud objektide arvu (*blokid* kaasa arvatud) ja *Orphan Nodes* näitab loodud *blokkide* arvu stseenipuus, kellel pole vanemat [52]



Joonis 20 – Godot profileerija objektide monitooring

Olukord tekkis kuna skriptis loodud *blokid*, kellele vanemat pole määratud tuleb mälust vabastada manuaalselt [53]. Üks lahendus oli iga *blokk* (mis koodis loodi) lisada ühte massiivi. Stseeni vahetamisel vabastati kõik massiivis olevad *blokid.*
Selline lahendus toimis antud projektis. Vaja on uurida edasi, kas on paremaid võimalusi *blokkide* mälust eemaldamiseks või kuidas üldse läheneda sellele probleemile antud arendusplatvormil. Muudatuse tulemusel liigseid objekte ega orbe enam ei tekkinud nagu näha Joonisel 21.



Joonis 21 – Godot profileerija objektide monitooring pärast paranduse sisseviimist

5.2 Uurimuslik testimine

Üks viis kuidas rakenduse funktsionaalseid nõudeid kontrolliti oli *exploratory testing*. See rõhutab testija autonoomsust, oskusi ja loovust ning soovitab läbi viia (põimitud kujul) sellised tegevused nagu: testimise disain, testide läbi viimine ja testide tulemuste hindamine [47].

Et protsess oleks struktuursem kasutati täpsemat strateegiat - *Session-Based Test Management* ehk SBTM, mis räägib sellest, kuidas organiseerida testimisprotsessi ennast ja kuidas sessioonidelt saadud mõõdikuid koguda ning hiljem ära kasutada [48]. Diplomitöös kasutati kirjeldatud strateegia esimest osa. Viimasena toodud allika näitel loodi dokument, mida testimisel täita tuleb. Viidi läbi mitmeid testimise sessioone. Üks nendest on näha Lisas 1.

Testimine viidi läbi kaugjuhtimisel TTÜ Smartlabis ja autori enda seadmetel teostati põhjalikum testimine, kuna niimoodi puudub viivitus. Smartlabis oli ligipääs 14 erinevale seadmele ja autor kasutas füüsiliselt 3 erinevat omaenda seadet. Kõik testimine toimus Androidi seadmetel kuna iOSi jaoks polnud olulised nõuded nagu reklaamide kuvamine ja ostude sooritamine kirjutamise hetkeks veel teostatud.

5.3 Juhuslik ekraani puudutamine

Otsides võimalusi rakendust koormuse alla panna leiti ühest allikast [49] viide *adb shell monkey* programmile. *Monkey* on programm, mida käitatakse emulaatoris või seadmes ja mis genereerib pseudo-juhuslikke jadasid kasutaja sisenditest nagu klikid, puudutused või liigutused (samuti süsteemi-tasemel sündmustest). Seda saab kasutada rakendustele koormusttestide tegemiseks juhuslikel aga korratavatel viisidel [50].

Selleks, et *monkey* kasutus oleks võimalikult efektiivne ja testimine püsiks *leveli* sees ja mitte menüüdes tehti koodi mõned muudatused. Esiteks kõrvaldati võimalus mängus olles sealt menüü kaudu väljuda – väljumine oli võimalik ainult pärast *levelis* kõikide vajalike sammude sooritamist. Teiseks tehti mõistatuste lahendamine võimalikuks ka siis kui eeltingimused on täidetud ja *monkey* sisestab suvalise lahenduse. Eemaldati ka reklaamid. Viimaks lisati ühte *singletoni* võimalus testversiooni aktiveerimiseks ja inaktiveerimiseks, et vajadusel ei peaks paljudes kohtades koodi muutma.

Peale selle tuli *monkey* käivitamisel seada mitmeid parameetreid, et saada soovitud tulemust. Puudutused pidid toimuma ainult mängu sees. Seadistus, milleni jõuti pärast erinevaid variante, on näha Koodinäites 3. Käsu käivitamisel teeb *monkey* 50 000 puudutust ainult valikuga *-p* etteantud rakenduse sees.

```
adb shell monkey -p air.com.tedven.youmustescape2 --pct-touch 100 -v 50000
```

Koodinäide 3 – Käsk 50 000 pseudo-juhusliku puudutuse sooritamiseks

Paralleelselt *monkeyga* käitati ka *logcat* tööriista, et kinni püüda võimalikud veateated [51]. Koodinäites 4 oleva käsuga kuvatakse kõik mängust tulevad teated ja teised selle rakendusega seotud teated.

```
adb logcat -b all -v color | grep -e 'godot\|youmustescape2'
```

Koodinäide 4 – Käsk mängust tulevate ja sellega seotud olevate logide kuvamiseks

Vigade leidmine osutus sellist meetodit rakendades väga tulemuslikuks. Heaks indikaatoriks vea olemasolust oli näiteks see, kui *monkey* ei jõudnud *levelit* lõpetada 50 000 puudutusega. *Levelit* manuaalselt „kinnijäämise“ kohast edasi mängides selgus kiirelt ka viga, mis seda põhjustas.

Näiteks leiti viga laaduri kasutamisel. Jälgides *monkey* tegutsemist ja veateateid siis näis, et laaduri kasutamise ajal (või seda käivitades) võis väga kiire ekraani puudutamine käivitada laadurit mitu korda järjest. Lahendus oli laadimise ajaks viia mäng pausrežiimile, misjärel probleem kadus.

Paar viga tekkis ka inventaris liikumisel. Vead küll parandati aga kuna inventariga oli probleeme olnud ka varasemalt siis tuleb edasises arenduses *InventoryDirector* kriitiliselt üle vaadata – eriti see meetod, mis tegeleb pesa peal tehtavate klikkide töötlusega.

6 Kokkuvõte

Diplomitöö käigus valmisid migreeritava rakenduse baassüsteemid ja kolm *levelit*, mis on algusest lõpuni läbitavad. Samuti valmis Android plugin reklaamide kuvamiseks.

Alguses püstitatud eesmärk tuua üle terve mäng ei õnnestunud kuna enamus aega pühendati rakenduse kõige kasutatavamatele osadele, et veenduda nende töökindluses. Paljude nõuete täitmist polnud võimalik kontrollida, kuna need eeldavad, et kogu mäng on valmis.

Positiivne on see, et *levelite* loomine on tänu uuele süsteemile ja üle toomist abistavatele tööriistadele lihtne ja mugav ning edaspidi on võimalik mäng hõlpsasti lõplikult üle tuua. Uue lahendusega on võimalik hoida *levelid* iseseisvatena, mis aitab suurendada süsteemi osade eraldatust. See avab võimalused uute *levelite* lisamiseks olemasolevasse mängu nii, et algselt allalaaditav rakendus ei kasvaks liialt suureks. Reklaamide kuvamiseks arendatud *plugin* aitab vähendada riski kolmandate osapoolte laienduste toe kadumise suhtes. Samuti on nüüdsest võimalik sisse viia vajalikud muudatused kiiremini juhul kui reklaamivahendajate teegid peaksid näiteks uuenema.

Järgnevates etappides on plaanis arendada lõpuni iOS moodul reklaamide kuvamiseks ja teostada rakendusesisesed ostud. Plaanis on ka täiustada olemasolevat koodi.

Kokkuvõttes võib rahule jääda saavutatud tulemusega. Lisaks diplomitöös käsitletud rakendusele võimaldab valminud projekt ka teiste sarnaste rakenduste migratsioonid teha lihtsamaks, kiiremaks ja odavamaks.

Kasutatud kirjandus

[1] V. Radu ja D. Wong, „Get your apps ready for the 64-bit requirement,“ 15 Jan 2019. [Võrgumaterjal]. Saadaval: https://android-developers.googleblog.com/2019/01/get-your-apps-ready-for-64-bit.html. [Kasutatud 18 Apr 2020].

[2] arm, „64-bit Computing,“ Oct 2018. [Võrgumaterjal]. Saadaval: https://pages.arm.com/rs/312-SAX-488/images/64_bit_Computing_for_Mobile_White_Paper_v2.pdf. [Kasutatud 18 Apr 2020].

[3] Cisco Press, „CCNA Data Center DCICT 640-916 Official Cert Guide,“ 2015, p. 934.

[4] Adobe, „Download Adobe AIR SDK,“ [Võrgumaterjal]. Saadaval: https://www.adobe.com/devnet/air/air-sdk-download.html. [Kasutatud 18 Apr 2020].

[5] Google, „Sell digital purchases with Play In-app Billing,“ [Võrgumaterjal]. Saadaval: https://developer.android.com/distribute/best-practices/earn/in-app-purchases. [Kasutatud 18 Apr 2020].

[6] D. Galin, „Software Quality Assurance: From theory to implementation,“ Essex, Pearson Education Limited, 2004, pp. 37-47.

[7] R. Fitzpatrick, „Software Quality:Definitions and Strategic Issues,“ pp. 9-19, 1996.

[8] F. Wilson, „My Favorite Business Model,“ 23 Mar 2006. [Võrgumaterjal]. Saadaval: https://avc.com/2006/03/my_favorite_bus/. [Kasutatud 18 Apr 2020].

[9] bitwes, „Contributors to bitwes/Gut,“ [Võrgumaterjal]. Saadaval: https://github.com/bitwes/Gut/graphs/contributors. [Kasutatud 18 Apr 2020].

[10] R. Miracle, „Corona Labs annual update,“ 12 Feb 2020. [Võrgumaterjal]. Saadaval: https://coronalabs.com/blog/2020/02/12/corona-labs-annual-update/. [Kasutatud 18 Apr 2020].

[11] T. S. BV, „TIOBE Index,“ [Võrgumaterjal]. Saadaval: https://www.tiobe.com/tiobe-index/. [Kasutatud 18 Apr 2020].

[12] G2A, „Best AAA games – Everything you should know about Triple A,“ 12 Feb 2019. [Võrgumaterjal]. Saadaval: https://www.g2a.com/news/features/best-aaa-games/. [Kasutatud 18 Apr 2020].

[13] M. Giliazov, „Defold native extension for IronSource SDK,“ [Võrgumaterjal]. Saadaval: https://github.com/MaratGilyazov/def_ironsource. [Kasutatud 18 Apr 2020].

[14] YoYoGames, „GameMaker Studio 2 Features,“ [Võrgumaterjal]. Saadaval: https://www.yoyogames.com/gamemaker/features. [Kasutatud 18 Apr 2020].

[15] R. Manthorp, „GameMaker’s 20th anniversary,“ YoYo Games, 13 Nov 2019. [Võrgumaterjal]. Saadaval: https://www.yoyogames.com/blog/541/gamemaker-s-20th-anniversary. [Kasutatud 18 Apr 2020].

[16] Gitstar Ranking, „godotengine/godot - Gitstar Ranking,“ 10 Mar 2020. [Võrgumaterjal]. Saadaval: https://gitstar-ranking.com/godotengine/godot. [Kasutatud 18 Apr 2020].

[17] A. Gaviar, „GitHub’s Top 100 Most Valuable Repositories Out of 96 Million,“ 20 May 2019. [Võrgumaterjal]. Saadaval: https://hackernoon.com/githubs-top-100-most-valuable-repositories-out-of-96-million-bb48caa9eb0b. [Kasutatud 18 Apr 2020].

[18] R. Verschelde, „Here Comes Godot 3.2, with quality as priority,“ 29 Jan 2020. [Võrgumaterjal]. Saadaval: https://godotengine.org/article/here-comes-godot-3-2. [Kasutatud 18 Apr 2020].

[19] Patreon, „Juan Linietsky & Godot Core Contributors are creating Godot Engine,“ Aug 2017. [Võrgumaterjal]. Saadaval: https://www.patreon.com/godotengine. [Kasutatud 18 Apr 2020].

[20] „Godot - Android SDK statistics,“ [Võrgumaterjal]. Saadaval: https://www.appbrain.com/stats/libraries/details/godot/godot. [Kasutatud 18 Apr 2020].

[21] E. Peckham, „How Unity built the world’s most popular game engine,“ TechCrunch, 17 Oct 2019. [Võrgumaterjal]. Saadaval: https://techcrunch.com/2019/10/17/how-unity-built-the-worlds-most-popular-game-engine/. [Kasutatud 18 Apr 2020].

[22] D. Helgason, „Unity 1.0 is shipping,“ Unity, 29 Mar 2005. [Võrgumaterjal]. Saadaval: https://forum.unity.com/threads/unity-1-0-is-shipping.56/. [Kasutatud 18 Apr 2020].

[23] J. Brodkin, „How Unity3D Became a Game-Development Beast,“ Dice, 3 Jun 2013. [Võrgumaterjal]. Saadaval: https://insights.dice.com/2013/06/03/how-unity3d-become-a-game-development-beast/. [Kasutatud 18 Apr 2020].

[24] Unity, „DOTS - Unity's new multithreaded Data-Oriented Technology Stack,“ [Võrgumaterjal]. Saadaval: https://unity.com/dots. [Kasutatud 18 Apr 2020].

[25] K. Hougaard, „Creating a third-person zombie shooter with DOTS,“ Unity, 27 Nov 2019. [Võrgumaterjal]. Saadaval: https://blogs.unity3d.com/2019/11/27/creating-a-third-person-zombie-shooter-with-dots/. [Kasutatud 18 Apr 2020].

[26] R. Dillet, „Unity CEO says half of all games are built on Unity,“ TechCrunch, 5 Sep 2018. [Võrgumaterjal]. Saadaval: https://techcrunch.com/2018/09/05/unity-ceo-says-half-of-all-games-are-built-on-unity/. [Kasutatud 18 Apr 2020].

[27] M. Gentry, „10 Best Games Made With Unity [2020],“ Top Shelf Media LLC, 16 Jan 2020. [Võrgumaterjal]. Saadaval: https://www.highgroundgaming.com/best-games-made-with-unity/. [Kasutatud 18 Apr 2020].

[28] Godot Engine, „Custom modules in C++,“ [Võrgumaterjal]. Saadaval: https://docs.godotengine.org/en/stable/development/cpp/custom_modules_in_cpp.html. [Kasutatud 18 Apr 2020].

[29] Godot Engine, „Creating Android plugins,“ [Võrgumaterjal]. Saadaval: https://docs.godotengine.org/en/stable/tutorials/plugins/android/android_plugin.html. [Kasutatud 18 Apr 2020].

[30] ironSource, „Mediation Networks for Android,“ [Võrgumaterjal]. Saadaval: https://developers.ironsrc.com/ironsource-mobile/android/mediation-networks-android/. [Kasutatud 18 Apr 2020].

[31] ironSource, „Android SDK Integration,“ [Võrgumaterjal]. Saadaval: https://developers.ironsrc.com/ironsource-mobile/android/android-sdk/. [Kasutatud 18 Apr 2020].

[32] Shin-NiL, „Android Admob plugin for Godot Game Engine 3.2 or higher,“ 23 Mar 2020. [Võrgumaterjal]. Saadaval: https://github.com/Shin-NiL/Godot-Android-Admob-Plugin/blob/master/admob-plugin/gradle.conf. [Kasutatud 18 Apr 2020].

[33] Google, „Migrating to AndroidX,“ [Võrgumaterjal]. Saadaval: https://developer.android.com/jetpack/androidx/migrate. [Kasutatud 18 Apr 2020].

[34] Godot Engine, „Scenes and nodes,“ [Võrgumaterjal]. Saadaval: https://docs.godotengine.org/en/stable/getting_started/step_by_step/scenes_and_nodes.html. [Kasutatud 18 Apr 2020].

[35] Godot Engine, „SceneTree,“ [Võrgumaterjal]. Saadaval: https://docs.godotengine.org/en/stable/classes/class_scenetree.html. [Kasutatud 18 Apr 2020].

[36] Wikipedia, „Callback,“ [Võrgumaterjal]. Saadaval: https://en.wikipedia.org/wiki/Callback_(computer_programming). [Kasutatud 18 Apr 2020].

[37] Godot Engine, „GDScript basics,“ [Võrgumaterjal]. Saadaval: https://docs.godotengine.org/en/stable/getting_started/scripting/gdscript/gdscript_basics.html. [Kasutatud 18 Apr 2020].

[38] Godot Engine, „EditorPlugin,“ [Võrgumaterjal]. Saadaval: https://docs.godotengine.org/en/stable/classes/class_editorplugin.html. [Kasutatud 18 Apr 2020].

[39] Adobe, „Export animations for mobile apps and game engines,“ 22 Sep 2019. [Võrgumaterjal]. Saadaval: https://helpx.adobe.com/animate/using/create-sprite-sheet.html. [Kasutatud 18 Apr 2020].

[40] CreateJS, „The Easel Javascript library,“ [Võrgumaterjal]. Saadaval: https://github.com/CreateJS/EaselJS. [Kasutatud 18 Apr 2020].

[41] Adobe, „Adobe Animate,“ [Võrgumaterjal]. Saadaval: https://www.adobe.com/ee/products/animate.html. [Kasutatud 18 Apr 2020].

[42] A. Loew, „How to create a sprite sheet,“ [Võrgumaterjal]. Saadaval: https://www.codeandweb.com/texturepacker/tutorials/how-to-create-a-sprite-sheet. [Kasutatud 18 Apr 2020].

[43] E. Gamma, „Design Patterns: Elements of Reusable Object-Oriented Software,“ Addison-Wesley, 1994, p. 273.

[44] alexzheng, „Admob module for Godot, both iOS and Android.,“ [Võrgumaterjal]. Saadaval: https://github.com/alexzheng/admob_for_godot. [Kasutatud 18 Apr 2020].

[45] I. Luts, „Godot IronSource Android plugin,“ [Võrgumaterjal]. Saadaval: https://github.com/ksvslk/godot-ironsource-android. [Kasutatud 18 Apr 2020].

[46] Godot Engine, „GDScript exports,“ [Võrgumaterjal]. Saadaval: https://docs.godotengine.org/en/stable/getting_started/scripting/gdscript/gdscript_exports.html. [Kasutatud 19 Apr 2020].

[47] Agile Alliance, „Exploratory Testing,“ [Võrgumaterjal]. Saadaval: https://www.agilealliance.org/glossary/exploratory-testing/. [Kasutatud 24 Apr 2020].

[48] J. Bach, „Session-Based Test Managament,“ [Võrgumaterjal]. Saadaval: https://www.satisfice.com/download/session-based-test-management. [Kasutatud 23 Apr 2020].

[49] M. Gazar, „Stress-testing Android apps,“ 21 Dets 2016. [Võrgumaterjal]. Saadaval: https://proandroiddev.com/stress-testing-android-apps-601311ebf590. [Kasutatud 24 Apr 2020].

[50] Android Developers, „UI/Application Exerciser Monkey,“ [Võrgumaterjal]. Saadaval: https://developer.android.com/studio/test/monkey. [Kasutatud 24 Apr 2020].

[51] Android Developers, „Logcat command-line tool,“ [Võrgumaterjal]. Saadaval: https://developer.android.com/studio/command-line/logcat. [Kasutatud 24 Apr 2020].

[52] Godot Engine, „Performance,“ [Võrgumaterjal]. Saadaval: https://docs.godotengine.org/en/stable/classes/class_performance.html?highlight=orphan#enumerations. [Kasutatud 25 Apr 2020].

[53] Godot Engine, „Scripting / Creating Nodes,“ [Võrgumaterjal]. Saadaval: https://docs.godotengine.org/en/3.2/getting_started/step_by_step/scripting_continued.html. [Kasutatud 25 Apr 2020].

[54] A. Franke, „Background Load,“ [Võrgumaterjal]. Saadaval: https://github.com/godotengine/godot-demo-projects/tree/master/loading/background_load. [Kasutatud 25 Apr 2020].

[55] Godot Engine, „Signals,“ [Võrgumaterjal]. Saadaval: https://docs.godotengine.org/en/stable/getting_started/step_by_step/signals.html. [Kasutatud 25 Apr 2020].

[56] E. Gamma, „Design Patterns: Elements of Reusable Object-Oriented Software,“ Addison-Wesley, 1994, p. 293.

[57] R. E. Al-Qutaish, „Quality Models in Software Engineering Literature: An Analytical and Comparative Study,“ *Journal of American Science,* p. 167, 2010.

[58] A. Hudaib, R. Masadeh, M. H. Qasem ja A. Alzaqebah, „Requirements Prioritization Techniques Comparison,“ *Modern Applied Science,* kd. 12, nr 2, p. 66, 2018.

[59] N. Nagappan, E. M. Maximilien, T. Bhat ja L. Williams, „Realizing quality improvement through test driven development: results and experiences of four industrial teams,“ *Empirical Software Engineering,* kd. 13, nr 3, pp. 289-302, 2008.

[60] Z. Gao, C. Bird ja E. T. Barr, „To Type or Not to Type: Quantifying Detectable Bugs in JavaScript,“ [Võrgumaterjal]. Saadaval: http://ttendency.cs.ucl.ac.uk/projects/type_study/documents/type_study.pdf. [Kasutatud 29 Apr 2020].

[61] Godot Engine, „Singletons,“ [Võrgumaterjal]. Saadaval: https://docs.godotengine.org/en/stable/getting_started/step_by_step/singletons_autoload.html?highlight=singleton. [Kasutatud 29 Apr 2020].

[62] T. Robitaille, „psrecord,“ 16 Juuni 2018. [Võrgumaterjal]. Saadaval: https://pypi.org/project/psrecord/. [Kasutatud 11 Mai 2020].

[63] V. Hanson ja A. Tavast, „Arvutikasutaja sõnastik,“ [Võrgumaterjal]. Saadaval: http://www.keeleveeb.ee/dict/speciality/aks/.

[64] Google, „godot engine - Google Trends,“ [Võrgumaterjal]. Saadaval: https://trends.google.com/trends/explore?date=2013-02-05%202020-03-05&q=godot%20engine.

Lisa 1 – Testimise sessiooni raport

SESSIOON #1

EESMÄRK

Kontrollida võimalikult erineva resolutsiooniga seadmete peal, et menüüd sobitaksid end väiksemaks/suuremaks ja poleks muid visuaalseid defekte, mis puudutavad sobitumist.

OS	ALA	STRATEEGIA	KESTVUSE KATEGOORIA
ANDROID	MENÜÜD/RAAMID	VAATLUS	KESKMINE

SEADE	TOOTJA	VERSIOON	SDK	ABI	RESOLUTSIOON	PIKSLITIHEDUS
C6903	SONY	5.1.1	22	armeabi-v7a	1080x1920	XXHDPI
D5803	SONY	6.0.1	23	armeabi-v7a	720x1280	XHDPI
Nexus 7	ASUS	6.0.1	23	armeabi-v7a	1200x1920	XHDPI
SM-A300FU	SAMSUNG	6.0.1	23	armeabi-v7a	540x960	HDPI
SM-G920F	SAMSUNG	5.0.2	21	arm64-v8a	1440x2560	XXXHDPI
SM-T815	SAMSUNG	7.0	24	armeabi-v7a	1536x2048	XHDPI
SM-G950F	SAMSUNG	9.0	28	arm64-v8a	1440x2960	XXXHDPI
SM-T805	SAMSUNG	6.0.1	23	armeabi-v7a	1600x2560	XHDPI

START

04/23/2020 14:20 pm

TESTIJA

Indrek Luts

ÜLESANNETE KESTVUSE JAOTUS

ÜLESANNE	%
Probleemide otsimine	30
Vea uurimine ja raport	10
Sessiooni planeerimine	60

PLANEERITUD TESTIMINE %	MITTEPLANEERITUD TESTIMINE %
85	15

MÄRKMED

Ei leidnud menüüde ega muude detailide valepaigutusi.
Küll aga leiti muid võimalikke probleeme ja üks viga.

VÕIMALIKUD RISKID: -

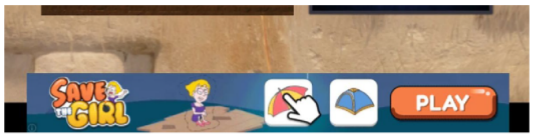
VEAD

#	KIRJELDUS
VIGA 1	Mõistatused (numbrite sisestamine, tähtede vahetamine) esimeses levelis tunduvad natuke uimased seadmes SM-T805. Ilmselt fondi detailsus on liiga üles keeratud.

VÕIMALIKUD PROBLEEMSED KOHAD

#	KIRJELDUS
PROBLEEM 1	Reklaambänneri paigutus SM-T815 seadmes võiks olla ilusam (vaata Joonis 1)

LISA



Joonis 1