TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Tarkvarateaduse instituut

Pavel Šukis 142270IABB

MEELELAHUTUSFUNKTSIONAALSUS PANGA MOBIILRAKENDUSSE POCOPAY NÄITEL

Bakalaureusetöö

Juhendaja: Inna Švartsman
MSc

Tallinn 2018

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: Pavel Šukis

28.12.2017

Annotatsioon

Käesoleva bakalaureusetöö eesmärkideks on uurida meelelahutusfunktsionaalsuse vajadust panga mobiilrakenduses, anda ülevaade Kotlini keelest ja mängude kirjutamiseks populaarsematest mootoritest ja raamistikest ning arendada mängu ja hiljem integreerida seda Pocopay rakendusse.

Autor andis ülevaade Pocopay ettevõtest, uuris ka Kotlini eelisi ja puudusi võrreldes Java keelega. Autor käsitles ka erinevaid mootoreid ja raamistikke mängude kirjutamiseks nagu Unity, jPCT-AE, libGDX ja AndEngine, seletas mille poolest mootor erineb raamistikust.

Töö tulemusena valmis raamistik, mis põhineb libGDX raamistikul ning Kotlini keeles kirjutatud mobiilimäng, mille hiljem autor integreeris teegina Pocopay rakenduse alpha versiooni. Autor kirjeldab mängu mehhaanikat, integreerimis- ja loomiseprotsessi.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 38 leheküljel, 7 peatükki, 18 joonist, 1 tabel

Abstract

Entertainment functionality in the mobile banking application on the example of Pocopay

The purpose of this bachelor's thesis is to research and analyze the need of entertainment functionality in the mobile banking application, provide an overview of Kotlin language and the most popular engines and frameworks for gaming, develop the game and then integrate it into the Pocopay application.

The author gave an overview of the Pocopay company, also examined the advantages and disadvantages of Kotlin compared to the Java language. The author explained how engine differs from the framework. Most populat engines and frameworks for Android game developing Unity, jPCT-AE, libGDX and AndEngine were analyzed.

As a result of this thesis framework that was based on the libGDX framework and a mobile game were created. Game is written in the Kotlin language, and then integrated to the alpha version of the Pocopay application as a library. The author describes the game's creation process, game mechanics, and the integration process.

The thesis is written in Estonian and contains 38 pages of text, 7 chapters, 18 figures, 1 table

Lühendite ja mõistete sõnastik

AAR	**Android Archive** Android teegi fail [2]
Activity	Androidi rakendustes kasutatav komponent, mis loob akna, kust kasutajaliidest näidatakse [3]
Android Manifest	Document, mis kirjeldab rakendust [1]
APK	**Android Package Kit** Android rakenduse arhiiv, mis sisaldab lähtekoodi ja programmi tööks vajalikke lisafaile [4]
BUG	Tarkvara defekt
CPU	**Central Processing Unit** Keskprotsessor. Täidab arvutiprogrammide juhiseid ning on peamine vahend arvuti ülesannete täitmisel [5]
GIF	**Graphics Interchange Format** Rasterpildi vorming, mis toetab igas pildis kuni 8 bitti piksli kohta [6]
GUI	**Graphical User Interface** Graafiline kasutajaliides [7]
Intent	Android operatsiooni abstraktne kirjeldus [8]

JAR	**Java Archive** Failivorming, mida tavaliselt kasutatakse mitme Java faili ja nendega seotud metaandmete ja ressursside koondamiseks ühte levitamiseks mõeldud faili [9]
JVM	**Java Virtual Machine** Tarkvara süsteem, mis käivitab java baitkoodi [10]
KB	**Kilobait** Arvutites kasutatav infoühik, mis sisaldab 1024 baiti [11]
LLVM	**Low Level Virtual Machine** Programmide analüüsimise, ümberkirjutamise ja optimiseerimise universaalne süsteem [12]
MB	**Megabait** Arvutites kasutatav infoühik, mis sisaldab 1024 kilobaiti ehk 1 048 576 baiti [11]
NPE	**Null Pointer Exception** Erind [13]
PPI	**Pixels per inch** Pikslitihedus. Näitab, kui palju piksleid on ekraanil ühe tolli kohta [14]
Teek	**Library** Funktsionaalsus, mis on mõeldud korduvkasutamiseks erinevates programmides [15]

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Antud bakalaureusetöö eesmärgiks on uurida Android platvormile mängude kirjutamise põhimõtted ja analüüsida, kas kasutajad on huvitatud meelelahutusfunktsionaalsusest mobiilpanga rakenduses Pocopay näitel. Samuti uurida Kotlini keele võimalusi ja sobivust Android platvormile arendamiseks.

Töö teises peatükis tutvustab autor Pocopay rakendust, räägib millega Pocopay tegeleb ja kuidas saab ta hakkama selliste suurte konkurentidega nagu teised Eestis tegutsevad pangad. Uurib, miks Pocopay on huvitatud sellises meelelahutusfunktsionaalsuses nagu mängud.

Töö kolmandas peatükis teeb autor enda analüüsi ning püüab vastata sellistele küsimustele, nagu kas on selline meelelahutusfunktsionaalsus vajalik sellistele rakendustele nagu pangad ning kuidas see võib pangarakenduse heaks töötada.

Töö neljandas peatükis räägib autor Kotlini keele kasutusest, uurib selle eripärasusi, toob välja selle eelised ja puudused ning võrdleb programeerimiskeelega Java.

Töö viies peatükk on pühendatud mängude arendamisele Androidile. Autor uurib milliseid mänge arendatakse, milliseid raamistikke ja mootoreid kasutatakse Android platvormile arendamisel.

Töö kuuendas peatükis räägib autor enda poolt arendatud mängust, mille nimetuseks on “Pocosnake”. Teeb põhilist mängu analüüsi, räägib kuidas on mäng kirjutatud ning kirjeldab protsessi kuidas võib mängu integreerida olemasolevasse rakendusse.

2 Pocopay

Pocopay ettevõte sai alguse sügisel 2014 ning tuli turule oma rakendusega veebruaris 2016. Esialgne idee oli teha mobiilne pank noortele inimestele, kes on alati liikumises. Taheti alustada maksete tegemisest ning ehitada maksemootori (maksed, arved, pangakaart). Seejärel leida partnereid Euroopas, kes spetsialiseeruksid krediitidel ja investeeringutel ning teha nendega nii nimetutud ökosüsteemi. See sai kõige raskemaks kohaks, kuna tuli välja, et selliseid partnereid, kes pakuksid krediite või investeeringuid üle Euroopat pole. On olemas sarnased kontseptsioonid, kuigi ei ole nad veel hästi arenenud. Teiseks probleemiks oli see, et Euroopa maksesüsteem on üsna piiratud. Ei ole sellist mudelit, mis oleks ühine kõikides Euroopa riikides. Seepärast peab igas riigis enda toodet üsna kõvasti lokaliseerima.

Pocopay esialgselt pühendas rohkem aega ja vaeva enda mootori arendamisele. Siis ehitas selle mootori peal rakenduse. Praegu võib öelda, et Pocopay töötab kahes suunas: hästi töötav mootor, mille litsentseerimises on mõned Eestis töötavad pankad huvitatud ning rakendus.

Rakenduse osas otsib praegu Pocopay strateegilisi partnereid. Tänapäeval paljud ettevõtted (autode müüjad, kaubanduskeskused) erinevatest turusektoritest soovivad minna pangasektorisse. Pocopay aga otsib ettevõtteid, kellel oleks suur kliendibaas. Tulevikus tahetakse arendada selline süsteem, mis ei oleks ainult maksetele ja teistele pangaasjadele suunatud, vaid milles oleksid erinevad teenused. Esimene samm on juba tehtud: läbi Pocopay rakenduse on võimalik tellida kindlustust eritingimustel läbi IIZI kidlustusmaakleri. Tulevikus tahetakse teenuste arvu suurendada, teha inimestele eripakkumusi sfäärides, milles kliendid just sellel hetkel huvitatud.

Lapse konto (child account) mängis suurt rolli Pocopay süsteemi väljatöötamisel. Praegu ükski pank ei ole nii hästi läbimõelnud laste kontod. Paremal juhul on lapsel taskus pangakaart. Pocopay pakub aga sellist teenust, et lapsevanema ja lapse pangakontod on ühendatud. Lapsevanem läbi enda konto saab vaadata lapse kontot, laps saab küsida vanema käest raha jne.

Rakendus oli loodud sellise ideega, et inimesed tahaksid seda avada mitte ainult selleks, et vaadata enda kontojääki. Pocopay loojad tahtsid, et rakendus oleks selline rohkem mängu või meelelahutusrakenduse moodi. Sellepärast on rakenduse sees palju animatsioone, igasugused gif ja piltide võimalused maksete tegemisel, profiili avatari seadistamine. Siis oli otsustatud, et Pocopay võiks pakkuda kasutajatele meelelahutuseks erinevate mängude mängimist. Selleks, et äratada kasutajatel huvi ja motivatsiooni, võib mängude mängimise eest jagada igasuguseid boonused.

- Võistlemine enda poco kontaktidega (poco community) – igal kasutajal on nii nimetatud poco sõbrad. Need on inimesed kasutaja telefoniraamatust, kellel on Pocopay konto olemas ja inimesed, kellega kasutaja on tehinguid teinud.

- Võistlemine teiste inimestega ning erinevate boonuste ja kingituste saamine.

Samahästi võib seadistada süsteemi niimoodi, et kasutajal oleks piiratud mängu katsete arv. Katsete arv võib kasvada, kui inimene teeb erinevad tehingud Pocopay rakenduses või kutsub sõpru ühineda Pocopay-ga. See võib muutuda Pocopay populaarsemaks ja atraktiivsemaks.

3 Analüüs

Antud peatükis analüüsib autor, kas mobiilne pangarakendus peaks piirduma raha käsitleva funktsionaalsusega, või sinna võiks integreerida erineva lisafunktsionaalsust.

Analüüs põhineb küsimustikul, mida autor koostas ning mille täitsid inimesed erinevatest vanuserühmadest ning sotsiaalsetest staatustest. Kõik edaspidised järeldused on tehtud antud küsimustiku vastuste põhjal.

Kõige populaarsem pangarakendus, mida inimesed kasutavad on Swedbank (40.8%), tema järel tuleb SEB (24.5%), Pocopay'd kasutavad 8 inimest (16.3%) ning 26.5% küsinutest ei kasuta panga mobiilrakendust.

Kõige populaarsem funktsioon, mida inimesed kasutavad on saldo vaatamine (67.3%), siis maksete tegemine (46.9%) ning raha küsimine (12.2%).

71.4% inimestest, kes vastasid küsimustikule tahaksid, et rakendus saaks teha just neile vajalikud pakkumused. Näiteks, juhul kui inimene käib Stockmann toidupoes ning maksab kaardiga, siis mobiilrakendus võib läbi partnereid teha sellele inimesele pakkumusi või allahindlusi Stockmann toidupoes. Mida rohkem inimene kasutab pangakaardi, seda rohkem informatsiooni on mobiilrakendusel, seda kergem ja täpsem võib pakkumusi teha.

28.6% inimestest vastasid, et nad tahaksid, et panga mobiilrakenduses oleksid mängud, 26.5% vastasid, et ei tahaks. 10.1% vastasid, et pigem ei kui jah ning 8.3% pigem jah kui ei.

Juhul, kui rakenduses oleksid mängud, siis 67.3% küsinutest oleksid motiveeritud neid mängima, kui mingi tulemuse eest oleks võimalik võita erinevad kinkekaardid ja boonused, 51% vastas, et neid motiveeriks rahaline auhind ja 28.6% tahaksid sõpradega võistelda. 10% vastasid, et midagi neid ei motiveeriks. Siit tuleb selline järeldus, et peaaegu pool küsimustikule vastanutest ei soovinud, et rakenduses oleksid mängud sees, kuigi ainult 10% ei oleks motiveeritud neid mängida.

4 Kotlin

Kotlin on programmeerimiskeel, mis töötab Java Virtuaal Masinal aga samuti võib olla kompileeritud JavaScript lähtekoodi või kasutada LLVM kompilaatori. Ta on arendatud JetBrains meeskonna poolt. Kotlinis kirjutatud kood saab töötada koos Javas kirjutatud koodiga, kuna Kotlini koodist genereeritakse Java baitkood. See tähendab, et ei pea Javas kirjutatud projekti ümber teisendama, vaid lihtsalt alustada uue klassi kirjutamist Kotlinis, mis saab probleemideta töötada Java klassidega, meetoditega, ja teekidega. [16]

4.1 Kotlini eelised

4.1.1 Kokkuvõtlik

On üldteada, et arendajad kulutavad rohkem aega olemasoleva koodi lugemiseks kui uue koodi kirjutamiseks. Mida lihtsam ja kitsam on kood, seda kiiremini võib aru saada, mis koodis toimub. Loomulikult mängivad siin olulist rolli hea disain ja väljade deklareerimisnimed. Kuid keele valik ja selle lühendus on samuti olulised. Paljud Java standardvarustuses olevad plokid, nagu näiteks getterid, setterid ja klasside konstruktorid, on Kotlinis kaudsed ja ei sega lähtekoodi. Lühike kood võtab vähem aega kirjutamiseks ja veelgi olulisem, vähem aega lugemiseks. See suurendab arendaja tootlikkust. [16]

4.1.2 Turvaline

Programmeerimiskeel on turvaline, kui selle disain takistab teatud tüüpi vigu programmis. Muidugi keel ei takistata kõiki võimalikke vigu. Kotliniga oli püütud saavutada kõrgemat turvalisusetaset kui see praegu Javas on. JVM-is käivitamine pakub juba palju turvalisust: näiteks mälu erindid, puhvri ülevoolu vältimine ja muud mälu probleemid. Kotlinis ei pea kõik väljade tüübid täpsustama deklareerimisel, sest paljudel juhtudel määrab kompilaator tüübid automaatselt.

Üheks suurematest ja tihedamini esinevatest probleemidest Javas programmeerimisel on *null* viited. Nende kasutamine võib programmi tööd tõsiselt häirida, võivad tekkida *bug*id ja programmi kokku jooksmised. Kotlin on suunatud NullPointerExceptioni eemaldamiseks koodist. Kotlinis väljade deklareerimisel antakse programmile teada, kas

antud viide saab hoida *null* väärtust või mitte (Joonis 1). Juhul, kui viide saab hoida *null*i, edaspidi selle välja töötamisega peab kasutama kas '?.' või '!!.' operaatoreid. '?.' operaatori kasutades annab arendaja programmile teada, et juhul, kui viide on *null* siis seda tuleb ignoreerida. '!!.' operaatorit kasutatakse ainult juhul, kui ollakse kindel, et viide ei saa hoida *null* väärtust. Nulli leidmisel viskab kompilaator programmi töös erindi. On olemas veel '?:' või nii nimetatud elvis operaator. Selle kasutamine on väga sarnane Javas oleva 'if' kasutamisega (Joonis 3). [17]

```
var pocopay = "Pocopay"
val length = pocopay.length
```

Joonis 1. Viide, mis ei saa hoida nulli

```
var pocopay = "Pocopay"
pocopay = null
val length = pocopay.length
```

Joonis 2. Kompileerimisviga

```
var pocopay: String? = "Pocopay"
pocopay = null
var length = pocopay?.length ?: -1

String pocopay = "Pocopay";
pocopay = null;
int length = pocopay != null ? pocopay.length() : -1;
```

Joonis 3. Elvis operaator

4.2 Kotlini puudused

Kotlin on väga perspektiivne programmeerimiskeel, kuigi isegi temal on olemas mõned puudused. Üheks nendest on konkurentidega võrreldes aeglasem kompileerimine. Hoolimata sellest, et Kotlin tungis üsna kiirelt arendajate ellu, praegu esineb puudusi dokumentatsioonis. See raskendab võimalike probleemide lahendamist. Näiteks Kotliniga seotud probleeme *StackOverflow'*is on alla 6500, samas on seal üle 1 300 000

postitusi, mis on seotud Javaga. Kuigi paljudel juhtudel võib probleemi lahendust otsida Java keeles ja hiljem teisendada selle Kotlini. [18]

4.3 Kotlin ja Android

27. mail 2017 oli välja kuulutatud, et Kotlin saab Androidi ametlikuks keeleks. See tähendab seda, et Android Studio 3.0, mille stabiilne versioon ilmus 20. oktoobril 2017, toetab Kotlini. Nüüd Androidi arendajatel ei ole enam vaja paigaldada lisateeke ega ühilduvuse pärast muretseda. [19]

5 Mängude arendamine Androidile

Mängude arendamine ei ole lihtne protsess. Mitte eriti keerukate arvutuste tõttu, vaid sellepärast, et enne mängu arendamise alustamist, peab autor mõtlema läbi erinevad mänguloomise aspektid. Programmeerijal on vaja muretseda selliste tavaliste asjadega nagu failide sisend / väljund, heli ja graafika töö ning võrgutugi. Kui kõik need küsimused on lahendatud, tuleb luua mängu mehhaanika. See nõuab ka kindlat arusaamist: kas on võimalik luua lihtsat simulatsiooni ilma füüsilise mootorita, kes ja kuidas hakkab elama mängu maailmas ja kuidas kõike neid elemente ekraanil kuvada.
Väga tähtis on läbi mõelda mängu disain enne programmeerimise alustamist. Autor peab küsima endalt palju küsimusi. Kas mängus on *welcome* ekraan? Mida mängija näeb mängu peaekraanil? Millised elemendid on nähtavad mänguekraanil? Mis juhtub, kui mängija vajutab “paus” nuppu või keegi helistab mängu ajal? Milliseid seadistusi on võimalik kasutajal muuta ning kuidas kuvatakse kasutajaliides erinevate suurustega ekraanidel?
Huvitaval kombel pole ühemõttelist vastust kõigile nendele küsimustele. Rangeid reegleid pole – kui autor, tellija ja kasutajad on tulemusega rahul, siis on kõik korras. Peaasi, et lahendus oleks nii lihtne kui võimalik. [20]

5.1 Mängude žanrid

Projekti alustamisel otsustatakse tavaliselt, milline on mängu žanr. Kui autor ei kavatse midagi täiesti uut ja varem enneolematut teha, siis on väga suur tõenäosus, et mängu idee sobib hetkel üheks populaarsemaks žanriks. Enamik žanre kasutab juba olemasolevaid mänguautomaatika põhimõtteid (juhtimisskeemid, konkreetsed eesmärgid jne). Nendest standarditest kõrvalekaldumine võib muuta mängu eriliseks - mängijad otsivad alati midagi uut, kuid see on ka suur risk. [20]

5.1.1 Tavapärased mängud

Tõenäoliselt on Androidi turu suurim mängude osa on nii nimetatud tavapärased mängud. Mis need on? Küsimusele pole konkreetset vastust, kuid nendel mängudel on tavaliselt palju ühiseid funktsioone. Enamasti on neid väga lihtne õppida (nii et mitte ainult kogenud mängijad mõistavad neid kiiresti), mis suurendab potentsiaalse vaatajaskonna ringi. Mängu seanss kestab tavaliselt mitu minutit. Nende mängude mehaanika võib olla väga lihtne (ühe nupuvajutusega platvormid, nagu näiteks palli korvi viskamine). [20]

5.1.2 Puzzle mängud

Üsna suure osa Android-mängude turust moodustavad puzzle mängud. Mängud, mis arendavad mängija mõtlemisviisi, mälu ja tähelepanu. Sellised on näiteks Tetris või sudoku. [20]

5.1.3 Arcade mängud

Arcade ja aktiivsed mängud realiseerivad tavaliselt Android platvormi kogu potentsiaali. Paljud neist pakuvad hämmastavaid 3D-kujutisi, mis näitavad praeguse põlvkonna riistvara võimekust. See žanr sisaldab palju alamkategooriaid (võistlused, laskurid, strateegiad ja teised). [20]

5.1.4 Kaasaegsed mängud

Mõningaid mänge on raske määrata konkreetse kategooriasse - need kasutavad Android-seadmete kaamerat ja GPS, et luua ebatavalisi uuenduslikke lahendusi. Nende mängude keskmes on sotsiaalsed võrgustikud ja geograafiline asukoht ning mõned täiustatud reaalsuse elemendid. [20]

5.2 Mängumootorid ja raamistikud

Android platvormi jaoks on olemas mitu kaubanduslikke ja avatud lähtekoodiga raamistikke ja mängumootoreid. Raamistik võimaldab kontrollida mänguarenduskeskkonna kõiki aspekte. Arendaja peab ise aru saama, kuidas konkreetseid probleeme lahendada (mängu maailma korraldada, ekraane töödelda jne). Mängumootor on aga orienteeritud spetsiifilistele ülesannetele. Ta pakub mooduleid, mida on lihtne kasutada igasuguste probleemide lahendamiseks. See muudab mängude arendamist lihtsamaks. Teisest küljest see võib mängu arendamist kõvasti piirata. Sellisel juhul peab arendaja ise mootorit muutma, mis ei pruugi olla võimalik, kui selle lähtekood pole kättesaadav. Mootorid võivad märkimisväärselt vähendada arendusaega, kuid võivad seda suurendada juhul, kui tekib probleem. [20]

5.2.1 Unity mootor

Mootor, millel on suurepärased tööriistad ja funktsionaalsus. Sellega saab luua mänge erinevatele platvormidele: iOS, Android, Windows, PlayStation ja Xbox. Sellega saab luua brauserimänge, mis kasutavad Unity veebimängija tarkvaramoodulit. See võimaldab mängurežiimide programmeerimiseks kasutada paljusid keele, kuid Java neid ei toeta. [21]

5.2.2 UDK mootor

Unreal Development Kit mängmootor töötab mitmel platvormil. Mootor on üsna kõrge kvaliteediga ning kasutab oma skriptikeelt. [20]

5.2.3 jPCT-AE mootor

jPCT-AE mootor on ehitatud jPCT mootori baasil Androidi jaoks. Ta on kirjutatud Javas. Sellel on mitu erinevat funktsiooni, mis aitavad 3D-programmeerimisel kaasa. See töötab nii lauaarvutitel kui ka Androidi seadmetel. Selle lähtekood ei ole kättesaadav. [22]

5.2.4 Ardor3D mootor

Ardor3D on väga võimas 3D programmeerimismootor. See töötab nii Androidil kui ka lauaarvutitel. Sellel on avatud lähtekood ja suurepärane dokumentatsioon. [20]

5.2.5 libGDX raamistik

Javas kirjutatud raamistik, mis sobib nii 2D kui ka 3D mängude kirjutamiseks. Võimaldab kirjutada koodi üheaegselt nii Android, iOS kui ka Windows platvormidele. [20]

5.2.6 AndEngine raamistik

Javas kirjutatud 2D raamistik, mis osaliselt põhineb libGDX koodil. Sellega saab luua mänge ainult Android platvormile. Tema kontseptsioon on väga sarnane kuulsa cocos2d mootoriga, mis on ette nähtud iOS platvormile arendamiseks. [20]

5.3 Mängumootorite ja raamistike võrdlemine

Mängumootorite ja raamistike peamine ülesanne on mängude arendamise protsessi kergendamine. Kui mängu žanr on paika pandud, peab arendaja otsustama, kas hakkab kasutama ta mootori või raamistiku. Enne otsuse tegemist, peab ta pöörama tähelepanu järgmistele aspektidele:

- 2D või 3D – nii raamistikud, kui ka mootorid toetavad 2D ja 3D mängude kirjutamist, kuigi raamistikud on rohkem 2D jaoks, mootorid aga 3D jaoks.
- Programmeerimiskeel – paljud raamistikud ja mootorid toetavad Java keelt, kuid mitte kõik.
- GUI või kood – mootorid toetavad graafilist kasutajaliidest.
- Tasuta või tasuline – mõnedel motooritel on piiratud tasuta funktsionaalsus, raamistikud on peamiselt kõik avatud lähtekoodiga.
- Rakenduse suurus – minimaalne Unity Android projekti suurus on umbes 8MB, libGDX projekti suurus on umbes 200KB.

Kõige populaarsem raamistik Androidi jaoks on libGDX. Mootoritest on kõige populaarsemad Unity ja jPCT-AE. [23]

Autor oli otsustanud kasutama raamistiku, mitte mootori. Esiteks autor arendab 2D mängu, teiseks abivahend peab olema avatud lähtekoodiga, kolmandaks arendatud

mängu suurus peab olema võimalikult väike. Järgmiseks, peab autor valima, kas libGDX või AndEngine? Peamine erinevus nende vahel on see, et libGDX võimaldab arendada samaaegselt mitmetele platvormidele. Samuti libGDX-il on suur dokumentatsioon ning ta on palju populaarsem. [24]

6 Pocosnake

Pocosnake on ussimäng, mis on kirjutatud programmeerimiskeeles Kotlin. Mäng on arendatud Android platvormile. Mängu arendamiseks autor ei kasutanud olemasolevaid mootoreid ega raamistikke. Oli arendatud raamistik, mis põhineb libGDX raamistikul.

Kogu töö käigus valminud Pocosnake mängu lähtekood on saadaval järgneval aadressil: https://github.com/shukis/poco-snake

Mäng on arendatud teegina ning integreeritud Pocopay rakendusse. Mängul on kuus vaadet ja mitu juhtimisnuppu, mis on kirjeldatud Tabelis 1. Joonis 10 näitab mängu alustamise jadadiagrammi, Joonis 11 näitab mängu lõpetamise jadadiagrammi ja Joonis 12 näitab mängu *highscore* akna avamise jadadiagrammi.

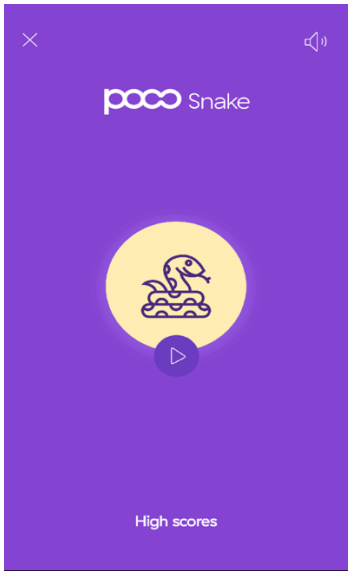
Tabel 1. Mängu ekraanid

Vaade nimetus	**Seadme tagasi-nupp**	**Juhtimisnuppud**	**Vaade kirjeldus**
Peavaade	Pocopay peavaade	Mängu alustamisnupp *Highscore* nupp Heli sisse- ja väljalülitamisnupp Mängu sulgemisnupp	Esimene vaade, millele suunab kasutajat Pocopay rakendus. Peale mängu lõppemist saab kasutaja samuti

			sellele ekraanile
Ready vaade	Peavaade	Tagasi nupp Heli sisse- ja väljalülitamisnupp Mängu alustamisnupp	Vaade, millele saab kasutaja, kui peavaades vajutab mängu alustamisnuppu.
Mänguvaade	Pause vaade	Neli mao juhtimisnuppu Heli sisse- ja väljalülitamisnupp Pause nupp	Vaade, mis vastutab mänguprotsessi eest.
Pause vaade	Mänguvaade	Mängu sulgemisnupp Heli sisse- ja väljalülitamisnupp Mängu jätkamise nupp	Vaade, millele saab kasutaja, kui mängu katkestatakse kas *pause* nupuga või mängust väljutades.
Game over vaade	Peavaade	Mängu sulgemisnupp Heli sisse- ja väljalülitamisnupp Mängu alustamisnupp	Vaade, millele saab kasutaja juhul, kui ussi pea puutub kokku ussi kehaga või kui terve mänguekraan on ussiga täidetud

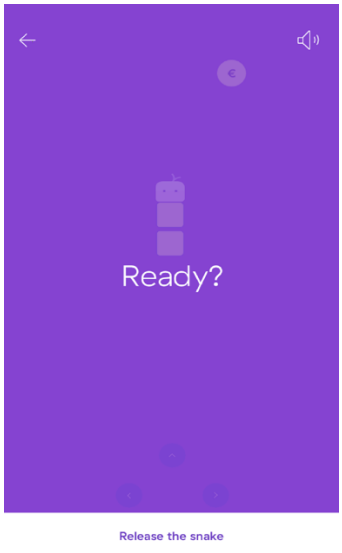
Highscore vaade	Peavaade	Tagasi nupp	Vaade, kus saab kasutaja vaadata enda skoori

Joonis 4 näitab mängu peaekraani.



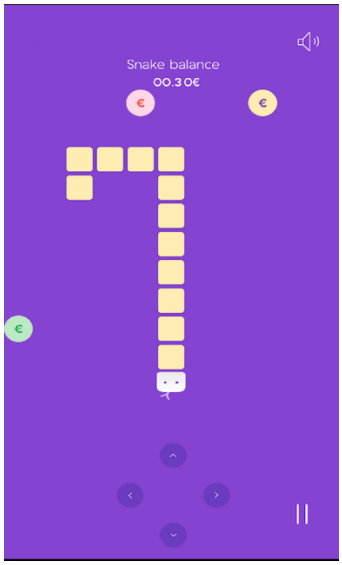
Joonis 4. Peaekraan

Joonisel 5 on näidatud *Ready* ekraan.



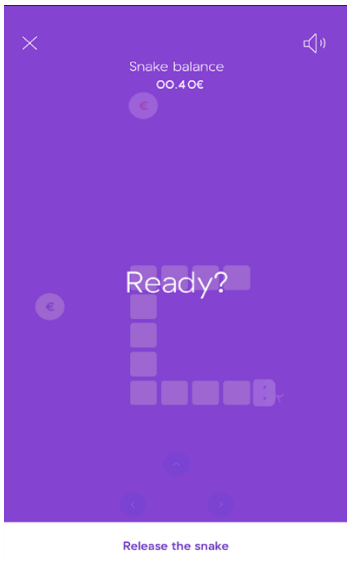
Joonis 5. *Ready* ekraan

Joonisel 6 on näidatud mängu ekraan.



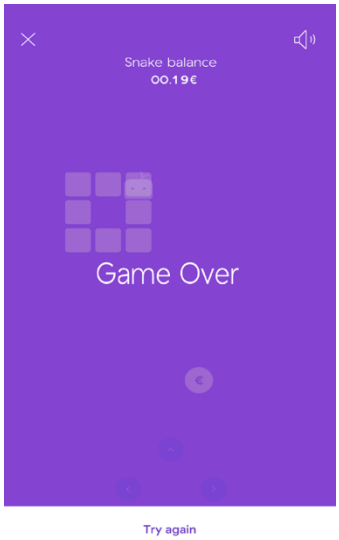
Joonis 6. Mängu ekraan

Joonisel 7 on näidatud mängu pausi ekraan.



Joonis 7. Pausi ekraan

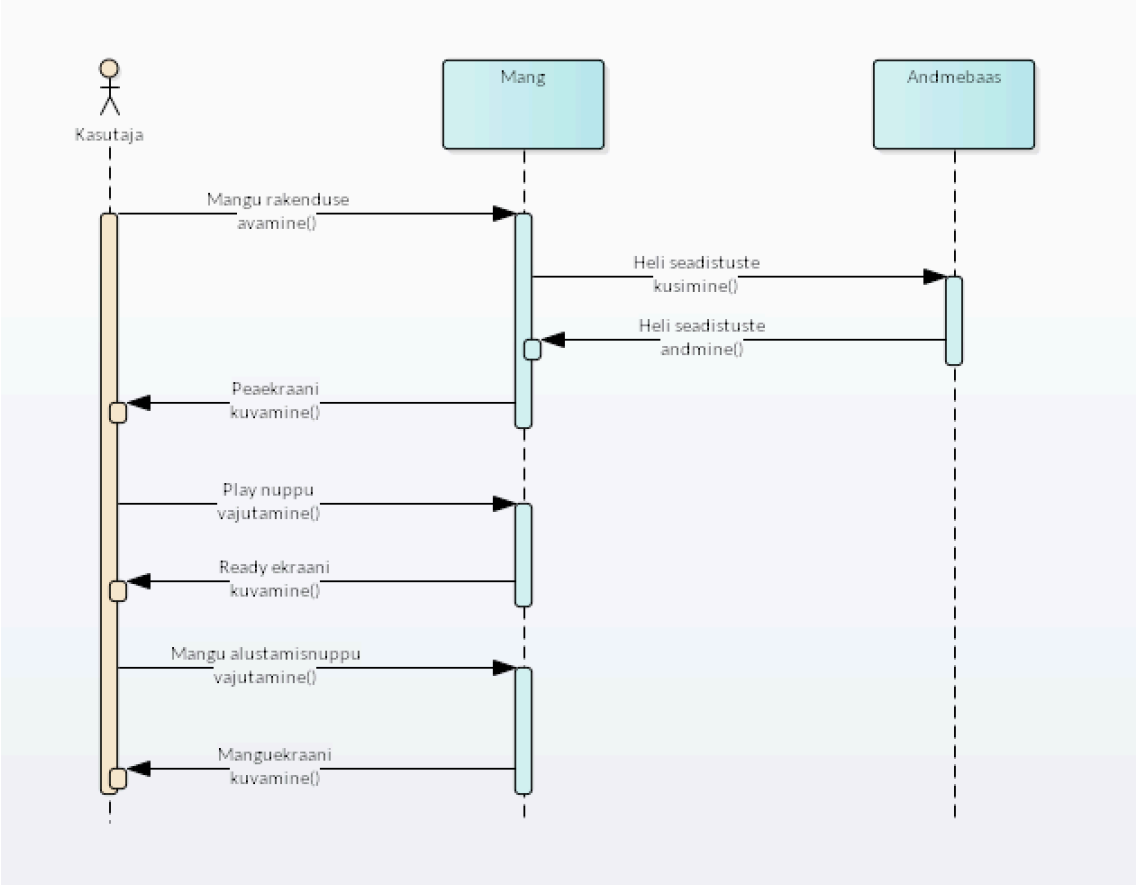
Joonisel 8 on näidatud mängu lõpetamise ekraan.



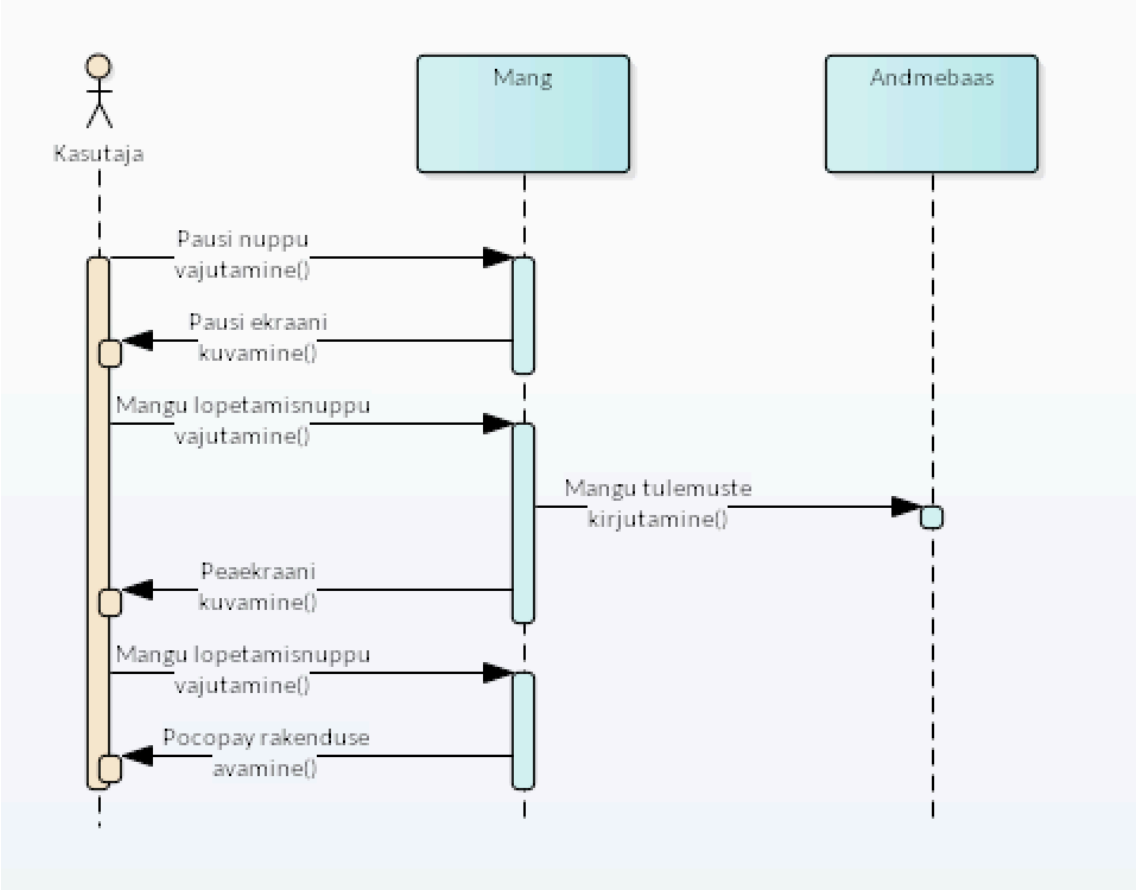
Joonis 8. *Game over* ekraan

Mängu *highscore* ekraan on näidatud Joonisel 9.

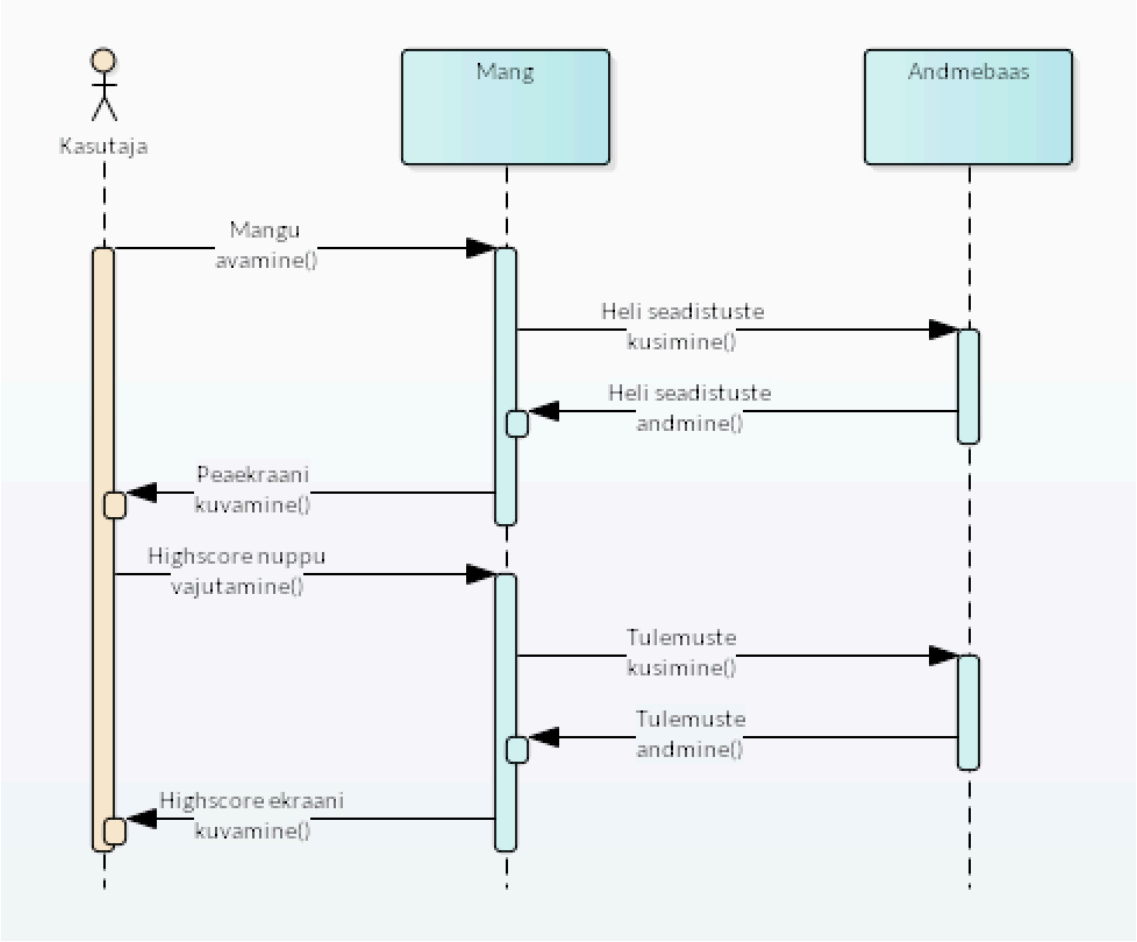
Joonis 9. *Highscore* ekraan



Joonis 10. Mängu alustamise jadadiagramm



Joonis 11. Mängu lõpetamise jadadiagramm



Joonis 12. *Highscore* ekraani avamise jadadiagramm

6.1 Rakenduse ülevaade

Pocosnake on kirjutatud programmeerimiskeeles Kotlin.

Rakenduse töötamiseks on vaja sõltuvuste alla lisada Kotlini teek (Joonis 13).

```
implementation "org.jetbrains.kotlin:kotlin-stdlib-jre8:$project.ext.KOTLIN_VERSION"
```

Joonis 13. Kotlini teegi lisamine projekti

Antud juhul kasutab pocosnake sama Kotlin teegi versiooni nagu ka Pocopay rakendus: 1.1.51. Samuti on vaja lisada AndroidManifest.xml faili järgmised load:

- android.permission.WAKE_LOCK – võimaldab rakendusel juhtida seadme aku ja CPU koormust [25]

- android.permission.WRITE_EXTERNAL_STORAGE – võimaldab rakendusel kirjutada ja lugeda faile seadme mälust [26]

6.1.1 SurfaceView

SurfaceView kasutatakse mängu erinevate elementide kuvamiseks mänguekraanile. *SurfaceView* pakub võimalust kasutada eraldi lõime elementide joonistamiseks ja mitte puudutada kasutajaliidese lõime.

Selleks, et joonistada *SurfaceView* eraldi lõimel, on vaja kasutada klassi *SurfaceHolder* eksemplari. Seda on vaja selle klassi kahe meetodi pärast:

- *Canvas SurfaceHolder.lockCanvas()* - meetod, mis blokeerib *Surface* joonistamiseks ja tagastab *Canvas*-i, mida saab edaspidi kasutada.
- *SurfaceHolder.unlockAndPost(Canvas canvas)* – vabastab *Surface* ja kontrollib, et programmi poolt joonistatud elemendid oleksid kuvatud ekraanile. Parameetrina antud *canvas* peab olema sama, mida tagastas meetod *SurfaceHolder.lockCanvas()* (Joonis 14).

```
val canvas = holder.lockCanvas()
canvas.getClipBounds(dstRect)
canvas.drawBitmap(framebuffer, src: null, dstRect, paint: null)
holder.unlockCanvasAndPost(canvas)
```

Joonis 14. *SurfaceHolder* näide

Surface pole loodud kohe pärast *SurfaceView* initsialiseerimist - seda tehakse asünkroonselt. Pind hävitatakse iga kord, kui tegevus on peatatud ning luuakse uuesti, kui see jätkub. Selleks, et saada *Canvas* objekt *SurfaceHolder*-ist, on vaja kontrollida, et *Surface* oleks korrektne. [20]

6.1.2 Graafika

Tänapäeva turul on väga palju erinevaid seadmeid, mis töötavad Android platvormil. Paljudel nendel seadmetel on erinevad ekraanid. Ekraanid erinevad mitmete parameetrite osas:

- Ekraani resolutsioon – näitab mis on ekraanide pixlite arv vertikaalis ja horisontaalis. Näiteks 1440 x 2560 pikslit.
- Ekraani suurus – näitab ekraani diagonaali suurust. Näiteks 5.5 tolli.
- Ekraani tihedus – näitab kui palju piksleid suudab ekraan kuvada ühe tolli kohta. Näiteks 534 ppi.
- Ekraani suhe – näitab ekraani pikslite arvu suhet vertikaalis ja horisontaalis. Näiteks 16:9.

Kõiki neid parameetreid tuleb silmas pidada mängu arendamisel. Mängu ekraan peab võimalikult ühesuguselt nägema erinevatel seadmetel.

Selleks, et kõikidel ekraanidel, vaatamata ekraani suurusele ja tihedusele, näeks mäng ühesuguselt võib teha Bitmap klassi objekt *frame buffer*. Panna talle vajalikud laius ja pikkus. Sellisel juhul ei pea arendaja muretsema, mis on tegelik ekraani suurus, sest kõik objektid hakkavad paiknema lähtudes selle *frame buffer* suurusest. Antud rakenduse *frame buffer* suuruseks on 1500 x 2668 pikslit, mis vastab ekraanisuhele 16:9. Juhul, kui ekraani pikslite arv on väiksem selle *frame buffer* pikslite arvust, võrdsustab seade selle ekraani suurusega. Samuti on vaja arvutada *frame buffer* pikslite ja ekraani pikslite arvu suhe, selleks, et töödelda ekraani vajutamisi. [20]

6.2 Nõuded

6.2.1 Funktsionaalsed nõuded

- Mängu käivitamine Pocopay rakenduse peavaatest
- Heli sisse ja välja lülitamine
- Boonuste teenimine

- Ussi juhtimine
- Mängu sees navigeerimine telefoni tagasi-nuppu vajutamisega
- Mängu peatamine
- Mängu jätkamine peale telefoni kõne
- Enda skoori vaatamine
- Mängust väljumine Pocopay rakendusse

6.2.2 Mittefunktsionaalsed nõuded

- Mängu toetavad 4.2.x ja uuemad Android versioonid
- Mängu toetavad enamus Android telefonide tootjad
- Mäng on stabiilne ja ei jookse kokku

6.3 Pocosnake spetsiifilisus

Pocosnake on arendatud Android teegi moodulina, mitte rakenduse moodulina. See tähendab seda, et mängu ei ole võimalik alla laadida ja käivitada seadmel. Selleks, et mäng töötaks on vaja Android rakendust, kust saab mängu *Activity* kutsuda. Sellisel viisil arendatud mängu on võimalik integreerida ükskõik millisesse Android projekti sisse. Seda on sama lihtne ka kustutada projektist.

6.3.1 Androidi teek

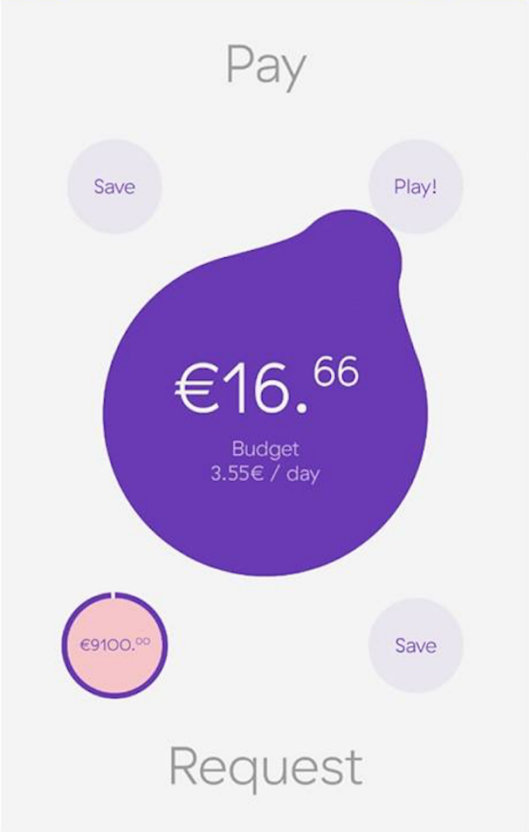
Androidi teek on strukturaalselt sama mis Androidi rakenduse moodul. See võib hõlmata kõike, mis on vajalik rakenduse loomiseks, sealhulgas lähtekoodi, ressursifaile ja Androidi manifesti. Kuid selle asemel, et kompileerida seadmesse käivitatavat APK faili, koondatakse kood Android arhiivi (AAR), mida saab kasutada Android rakenduse mooduli sõltuvusena. Erinevalt JAR-failidest võivad AAR-failid sisaldada Android ressursse ja manifesti. Juhul, kui ehitada AAR fail ja lisada see rakendusse, siis lähtekoodi, mida sisaldab see AAR fail, ei ole võimalik muuta. See tähendab, et AAR fail on lõplik mooduli versioon, mille käivitamiseks on vaja APK faili.

Teiseks võimaluseks on kasutada Android teeki ilma AAR arhiivi ehitamist. Sellisel juhul on võimalik teegi koodi muuta rakendusest, kuhu on see teek sõltuvusena lisatud. Android rakenduse moodulist on võimalik väga lihtsalt ehitada teegi moodul ja teistpidi. Projekti võib arendada rakendusena, paigaldada selle seadmele ja testida. Hiljem, kui on selline vajadus, teisendada see rakendus teegi mooduliks. Selleks on vaja viia mõned muudatused moodul-taseme *build.gradle* failis. Tuleb muuta *apply plugin: 'com.android.application' apply plugin: 'com.android.library'* vastu. Samuti tuleb kustutada *applicationId*, kuna ainult Androidi rakenduse moodul saab seda määratleda.

Siis on vaja teek lisada sõltuvusena rakendusse, kus hakkatakse seda kasutama. See protsess ei erine teiste teekide lisamisest. Tuleb rakenduse-taseme build.gradle failis sõltuvuste alla kirjutada "*implementation project(':teegi nimetus')*" juhul kui gradle versioon on 3.0 ja uuem ning "*compile project(':teegi nimetus')*" juhul kui gradle versioon madalam, kui 3.0. Samuti tuleb settings.gradle faili kirjutada "*include':teegi nimetus'*". Väga tähtis on lisada teegis olev *Activity* rakenduse AndroidManifest.xml faili. Siis muutub see rakendusest kättesaadavaks. [27]

6.3.2 Pocosnake integreerimine Pocopay rakendusse

Mängu jaoks oli Pocopay rakenduse peavaatele lisatud uus nupp "Play!". Animatsioon on tehtud niimoodi, et juhul, kui peamisest mullist libiseda sõrmega "Play!" mullile, siis avaneb mäng (Joonis 15).



Joonis 15. Mängu sisemine Pocopay rakenduse peavaatest

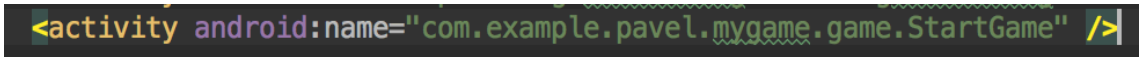
Mäng on lisatud sõltuvuste alla Pocopay *build.gradle* faili (Joonis 16) ning lisatud ka *settings.gradle* faili (Joonis 17). Selleks, et oleks võimalik mängu alustada, Pocopay rakenduse AndroidManifest.xml faili oli lisatud Pocosnake *Activity*, mis käivitab mängimisprotsessi (Joonis 18).

```
dependencies {
    implementation project(':poco-snake')
```

Joonis 16. Mängu lisamine sõltuvuste alla

```
include ':poco-snake'
```

Joonis 17. Mängu lisamine *settings.gradle* faili



Joonis 18. Mängu lisamine AndroidManifest.xml faili

Praegu mäng ei suhtle serveriga. See salvestab skoori lokaalselt. Põhimõtteliselt Pocopay ja Pocosnake võivad andmeid vahetada läbi intendi. See võiks tähendada seda, et mängule ei ole vaja teha serveriga ühendus, vaid see hakkab uuendama andmeid hetkel kui kasutaja navigeerib rakendusest mängu ja mängust rakendusse.

6.3.3 Mängu mehhaanika

Madu liigub selles suunas, kuhu vaatab tema pea. Madu saba liigub tema pea järgi. Pea ja saba on loodud ühesuurustest osadest, vaid välimuselt veidi erinevad. Kui madu jõuab ekraani piirile, ilmub see ekraani teisest küljest. Mao juhtimine käib nuppudega, iga suuna jaoks on eraldi nupp. Juhul, kui madu liigub ühes suunas ei ole võimalik pöörata teda ühe nuppu vajutamisega 180^{o} võrra. Ehk kui madu liigub vasakule, selleks et pöörata paremale peab kasutaja vajutama kas ülesse või alla ja ainult siis paremale. Kui mao pea läheb vastu mao saba, mäng lõpeb. Mängus on 3 erinevat elementi (münti), mida madu sööb: tavaline kollane münt, roheline münt ja punane münt.

- Kollane münt – mao pikkus kasvab ühe ruudu võrra ja skoor suureneb 0.03€ võrra.
- Roheline münt – mao pikkus väheneb 2 ruudu võrra ja skoor suureneb 0.06€ võrra. Roheline münt ilmub ekraanile iga üheksanda kollase mündiga.
- Punane münt – mao pikkus kasvab 2 ruudu võrra ja skoor väheneb 0.03€ võrra. Punane münt ilmub ekraanile iga neljanda kollase mündiga.

Esialgne mao kiirus on 1 ruut 0.5 sekundi jooksul. Iga kümnenda kollase mündi söömisega suureneb mängu kiirus 1 ruudu kohta 0.05 sekundi võrra. Kui kiirus jõuab 0.05 sekundit 1 ruudu kohta, siis rohkem see ei kasva.

Mänguväljaku suurus on 11 ruutu horisontaalis ja 15 ruutu vertikaalis. Mängu käigus võib heli lülitada välja või sisse ning mänguprotsessi panna pausi. Juhul, kui kasutaja väljub mängust või temal heliseb telefon mängu ajal, siis mäng läheb automaatselt pausi. Mäng lõpeb siis, kui kogu ekraani ruum on maoga täidetud. Mängu sees on seadistatud seadme tagasi-nuppu ülekatmine *(override)*.

7 Kokkuvõte

Käesoleva töö eesmärkideks olid:

- Uurida Android platvormile mängude kirjutamise põhimõtted.
- Analüüsida meelelahutusfunktsionaalsuse vajadust sellistes mobiilpanga rakendustes nagu Pocopay.
- Uurida Kotlini keele võimalusi ja sobivust Android platvormile arendamiseks.

Autor andis ülevaatuse Pocopay ettevõttest, millega see tegeleb ja kuidas end turul positsioneerib. Töö käigus olid uuritud programmeerimiskeele Kotlin võimalusied Androidile arendamiseks, selle positiivsed ja negatiivsed küljed. Samuti oli tehtud analüüs Android platvori mängude kirjutamise põhimõttetest, milliseid mänge arendatakse ja mis teeke selleks kasutatakse.

Töö tulemusena valmis “Pocosnake” mäng, mis on kirjutatud Kotlin keeles ja integreeritud Pocopay rakendusse teegina. Autor annab ülevaade mängust ja Pocopay rakendusse integreerimisest, kirjeldab selle nõuded, tööpõhimõtted, ja mehhaanika. Autor teeb enda analüüsi, kas inimesed sooviksid näha sellistes rakendustes nagu Pocopay meelelahutusfunktsionaalsust ja mis võiks neid motiveerida seda funktsionaalsust kasutama.

Tulevikus on plaan täiendada mängu lisafunktsionaalsusega. Seotada mängu töö Pocopay rakenduse tööga, et kasutajad oleksid huvitatud seda mängima ning integreerida seda *live* versiooni.

Kasutatud kirjandus

[1] Android "Manifest". [WWW] https://developer.android.com/guide/topics/manifest/manifest-intro.html (Kasutatud 22.12.2017)

[2] Android "Library". [WWW] https://developer.android.com/studio/projects/android-library.html (Kasutatud 22.12.2017)

[3] Android "Activity". [WWW] https://developer.android.com/reference/android/app/Activity.html (Kasutatud 22.12.2017)

[4] Wikipedia "Android APK" [WWW] https://en.wikipedia.org/wiki/Android_application_package (Kasutatud 22.12.2017)

[5] Wikipedia "CPU" [WWW] https://en.wikipedia.org/wiki/Central_processing_unit (Kasutatud 22.12.2017)

[6] Wikipedia "GIF" [WWW] https://et.wikipedia.org/wiki/GIF (Kasutatud 22.12.2017)

[7] Wikipedia "GIF" [WWW] https://et.wikipedia.org/wiki/Graafiline_kasutajaliides (Kasutatud 22.12.2017)

[8] Android "Intent" [WWW] https://developer.android.com/reference/android/content/Intent.html (Kasutatud 22.12.2017)

[9] Wikipedia “JAR” [WWW] https://en.wikipedia.org/wiki/JAR_(file_format) (Kasutatud 22.12.2017)

[10] Wikipedia “JVM” [WWW] https://en.wikipedia.org/wiki/Java_virtual_machine (Kasutatud 22.12.2017)

[11] Wikipedia “Bait” [WWW] https://et.wikipedia.org/wiki/Bait (Kasutatud 22.12.2017)

[12] Wikipedia “LLVM” [WWW] https://en.wikipedia.org/wiki/LLVM (Kasutatud 22.12.2017)

[13] TTÜ “Erindid” [WWW] https://ained.ttu.ee/javadoc/Exceptions.html (Kasutatud 22.12.2017)

[14] Wikipedia “PPI” [WWW] https://et.wikipedia.org/wiki/Pikslitihedus (Kasutatud 22.12.2017)

[15] Wikipedia “Teek” [WWW] https://et.wikipedia.org/wiki/Teek (Kasutatud 22.12.2017)

[16] D. Jemerov ja S. Isakova, Kotlin In Action, 2016

[17] JetBrains “Null safety” [WWW] https://kotlinlang.org/docs/reference/null-safety.html (Kasutatud 13.10.2017)

[18] ImperialJournals “Kotlin – The unrivalled android programming language lineage” [E-ajakiri] http://imperialjournals.com/index.php/IJIR/article/view/5491/5283 (Kasutatud 13.10.2017)

[19] JetBrains “Kotlin on android. Now official” [E-ajakiri]

https://blog.jetbrains.com/kotlin/2017/05/kotlin-on-android-now-official/
(Kasutatud 14.10.2017)

[20] R. Green ja M. Zechner, Beginning Android Games, 2012

[21] Wikipedia “Unity” [WWW] https://en.wikipedia.org/wiki/Unity_(game_engine)
(Kasutatud 18.10.2017)

[22] JPCT “jPCT-AE” [WWW] http://www.jpct.net/jpct-ae/ (Kasutatud 18.10.2017)

[23] DarioPenic “How to choose android game engine” [WWW]
http://dariopenic.com/how-to-choose-an-android-game-engine-libgdx-vs-unity/
(Kasutatud 20.10.2017)

[24] IyousufSyed “libGDX vs AndEngine” [WWW]
https://iyousufsyed.wordpress.com/2014/06/30/libgdx-vs-andengine/
(Kasutatud 20.10.2017)

[25] Android “Wake Lock” [WWW]
https://developer.android.com/reference/android/os/PowerManager.WakeLock.html (Kasutatud 29.10.2017)

[26] Android “Write External Storage” [WWW]
https://developer.android.com/reference/android/Manifest.permission.html#WRITE_EXTERNAL_STORAGE
(Kasutatud 29.10.2017)

[27] AndriyDruk “Mis on .AAR teek” [WWW]
https://andriydruk.com/post/what_is_aars/
(Kasutatud 8.11.2017)