Tallinna Ülikool
Digitehnoloogiate Instituut

INTERAKTIIVSETE REKLAAMIDE TUTVUSTAMINE JA ÜHE NÄITEREKLAAMI LOOMINE NUTISEADMELE CURLIFY CMS ABIL

Seminaritöö

Autor: Kevin Rull
Juhendaja: Jaagup Kippar

Autor: ……..... „ ……. „ 2016
Juhendaja: ……... „ ……. „ 2016
Instituudi direktor: …….. „ ……. „ 2016

Tallinn 2016

Autorideklaratsioon

Deklareerin, et käesolev seminaritöö on minu töö tulemus ja seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

……..	…….
(kuupäev)	(autor)

Sisukord

Sissejuhatus

Tänaseks oleme jõudnud punkti, kus tavalised banner tüüpi reklaamid ei kogu inimeste tähelepanu niipalju, kui nad võiksid koguda. Banner reklaamid, mis on pea iga lehe päises ning mis tavaliselt käivad kasutajatele pigem närvidele, kui on rõõmuks või informatsiooniks. Selleks, et inimestele reaalset kasu pakkuda reklaamist, tuleb mõelda välja uusi lahendusi.

Antud töö eesmärk on tutvustada üht võimalikku funktsionaalsust interaktiivsetel reklaamidel ning luua kõige tavalisem interaktiivne reklaam Curlify CMS (edaspidi CCMS) keskkonnas ja kirjeldada loomise käiku. Kuna antud töö on loodud Apple Macbook OS X süsteemis, siis kohati võib erineda nuppude paigutus mõningates rakendustes ning ka käsurea(Command line) käsukoodid võrreldes teiste operatsioonisüsteemidega. Väheke tuleb juttu ka Lua programmeerimiskeelest ning sellest, mis keskkond on Content Management System (CMS).

1. Interaktiivne reklaam ja bänner tüüpi reklaam

Interaktiivne reklaam iseenesest ei ole väga uus mõiste seda on kasutatud juba käesoleva sajandi algusest peale. "Interactivity is at the center of where advertising is going." Robert Schmetterer ajakirjas *Advertising Age,* 4. jaanuar, 1999. Robert Schmetterer'i ütlust võib küll võtta suunaks, kuhu tol ajal reklaam püüdis liikuda. Banner reklaam ise sai alguse 1993. aastal, kui Global Network Navigator nimeline firma müüs esimese veebireklaami, kuhu peale sai ka klikata (vt joonis 1).

Joonis 1 Maailma esimene bänner tüüpi reklaam

1.1 Interaktiivne reklaam

Interaktiivne meedia, mille üks osa on interaktiivne reklaam, on kommunikatsiooni meetod, kus meedia lõppseis sõltub meedia kasutaja sisendist[1]. Ehk teisisõnu saame eeldada, et reklaami lõpptulemus sõltub kasutaja tegevusest või sellest, mida kasutaja saab etteantud reklaamis teha. Interaktiivse reklaami tellija maksab ka ühel viisil kolmest, mis on välja toodud käesolevas töös bänner tüüpi reklaami tutvustuses. Kuna reklaamiagentuur NEXD kasutab Curlify platvormi, siis nende kodulehelt on võimalik näha sama tüüpi interaktiivsete reklaamide raporteid. Antud võrdlusse on võetud kõige esimene olemasoleva reklaami raport[2]. Seal on välja toodud, et uutmoodi reklaami CTR on (2,94%) koguni 8,6 korda kõrgem, kui teistel reklaamidel. Kasutaja tegelemine reklaamiga oli koguni 20 korda kõrgem, kui tavalisel keskmisel reklaamil.

1.2 Bänner tüüpi reklaam

Bänner tüüpi reklaam on ristkülikukujuline pilt või animatsioon, mida on mitmes erinevas mõõdus ning mida kasutatakse mitmeid moodi, veebilehe küljel, veebilehe päises ning ka

[1] http://www.investopedia.com/terms/i/interactive-media.asp 29.02.2016
[2] http://www.nexd.ee/american-express-case-study/ 29.02.2016. American Express raport

kohati kogu veebilehe ulatuses. Bänner reklaami eesmärk nagu ka interaktiivse reklaami eesmärk on promoda tootemarki ja/või viia kasutaja reklaami tellija lehele.
Tellija tasub reklaami eest ühel viisil kolmest:

- Cost per impression (CPM) – iga tuhande kasutaja kohta, kes näeb reklaami.
- Cost per click (CPR) – iga kasutaja kohta, kes klikkab reklaamil ja külastab tellija lehte.
- Cost per action (CPA) – iga kasutaja kohta, kes klikkab reklaamil ja suundudes tellija lehele täidab seal mõne vormi või ostab midagi[3].

Esimese bännerreklaami CTR oli 44% nüüdseks on sama näitaja kukkunud umbes 0.1% peale.[4]

Arvestades, kui madale CTR praeguseks hetkeks langenud on, siis interaktiivne reklaam on tõstnud korralikult ikka kasutajate klikkamise arvu. Võrreldes interaktiivse reklaami ja bännerreklaami CTR näitajaid. Võime oleteda, et tulevikus võivad langeda kaasahaaraval reklaamil kasutajate klikkamise arv samale tasemele nagu seda on hetkel bännerreklaami puhul. Seda loomulikult juhul, kui interaktiivne reklaam ei muutu rohkem intrigeerivamaks.

[3] http://www.investopedia.com/terms/b/banneradvertising.asp 29.02.2016
[4] http://mashable.com/2013/08/09/first-banner-ad/#q2fCTid8XkqY 29.02.2016

2. Programmeerimiskeel Lua

Lua on võimalusterohke, kiire, kerge ja teiste keeltega seotav – just nii kirjeldatakse Luat enda kodulehel. Lual on lihtsalt järgitav süntaks ning võimas andmete töötlemine massiivides.[5]

Kiireks prototüüpimiseks, skriptimiseks ja ka konfiguratsiooniks on Lua ideaalne, sest võimaldab automaatset mäluhaldust koos kasvava mälukoristusega.

Lua sai alguse 1993. aastal Brasiiliast, Paavstliku Katoliku Ülikoolist Rio de Janeiros (PUC-Rio, Pontifical Catholic University of Rio de Janeiro). Täpsemalt sündis keel Tecgraf'is, mis oli PUC-Rio ülikooli arvutigraafika tehnoloogia grupp (Computer Graphics Technology Group of PUC-Rio).

Lua nimi tuleneb portugali keelest ja see tähendab Kuud, Maa kaaslane Kuu. Häädusena on mainitud, et tuleks hääldada *LOO-ah*. Mainitakse veel ka, et nagu iga nimigi, tuleks ka Lua kirjutada algustäht suurelt ja ülejäänud tähed väikselt, vältida tuleks *LUA* kirjutamist, mis antud kontekstis autoritele ei meeldiks, sest see on kole ja segadusse ajav.

Lua on ennast tõestanud kui töökindel programmeerimiskeel. Seda just seetõttu, et seda on kasutatud paljudes tööstuslikes rakendustes (nt Adobe Photoshop Lightroom). Kuid põhirõhk on keerulisematel süsteemidel, nagu näiteks Brasiilia digitaalse televisiooni vahevara[6] *Ginga* ja ka mängudes (World of Warcraft[7], Angry Birds[8]).

On levinud ütlus: "Olla nii kiire kui Lua". Seda siis sellepärast, et Lua on pälvinud lugupidamise jõudluse ja kiiruse osas omal alal, mida on näidanud mitmed *benchmark*[9] testid. Olulised osad mahukatest rakendustest on kirjutatud Lua programmeerimiskeeles.

Lua on lisaks kõigele ka paindlik. Teda on võimalik jooksutada kõikvõimalikes operatsioonisüsteemides. Lua't on võimalik ka siduda väga mitmete keeltega sealhulgas Java ja C#.

Kokkuvõtlikult on Lua võimas, lihtne, väike ja avatud lähtekoodiga tarkvara, mida kasutatakse keerulistes rakendustes ning ka paljudes mängudes.

---

[5] http://www.lua.org/about.html 27.10.2015

[6] Tarkvara, mis seob kokku kaht muidu iseseisvat rakendusprogrammi. http://www.vallaste.ee 29.02.2016

[7] *Online* seiklusmäng http://us.battle.net/wow/en/game/guide/ 29.02.2016

[8] Kaasahaarav hittmäng arendatud Rovio Entertainment poolt http://www.rovio.com/games/angry-birds 29.02.2016

[9] Ühe ja sama programmi käitamine erinevatel arvutitel ja tulemuste võrdlemine. http://www.vallaste.ee 29.02.2016

3. Sisuhaldussüsteem (CMS)

Sisuhaldussüsteem on süsteem, kus on võimalik hallata suures koguses andmeid ja informatsiooni ilma, et peaks igal HTML lehel eraldi koodi ümber kirjutama[10]. Sisu haldaja enamjaolt ei pea oskama otseselt mingisuguseid programmeerimiskeeli, sest üldjuhul on olemas sisuhaldussüsteemides tekstiredaktorid ja igasugused graafilised liidesed, mis aitavad asju õigesti paika panna ning hallata. Üldiselt CMS eesmärk on lasta hallata veebilehe erinevat sisu erinevatel inimestel.

3.1 Curlify CMS

Curlify CMS (edaspidi CCMS) ei erine oma otstarbe poolest üldiselt teistest sisuhaldussüsteemidest. Ligipääs on ainult valitud kasutajatel nagu üldjuhul ka teistel süsteemidel. CCMS's on nimetatud 4 kasutaja astet:

- Admin
- Developer
- Editor
- Read-only

Vastavalt kasutajatüüpi järjekorrale järjestuvad ka õigused igale kasutajale. Admin tüüpi kasutajal on õigus näha ja teha kõike vastava konto all. Developer kasutajal on ka võimalik üldiselt teha kõike, kuid tal ei ole õigust kasutajaid konto alt ära kustutada või uusi kasutajaid lisada. Editor kasutajal on veel vähem õigusi, kui Developer tüüpi kasutajal. Nimelt ei saa Editor kasutaja Reklaami koodi muuta. Lõpuks jääb alles Read-only tüüpi kasutaja, kellel on ainult nagu nimigi ütleb, lugemise õigused, Ehk Read-only tüüpi kasutaja saab ainult näha reklaami koodi ja pildifaile, kuid tal ei ole õigusi neid muuta.

[10] http://edoc.hu-berlin.de/oa/articles/reXWUcDNwYelM/PDF/22sWVsHpHgNRQ.pdf 29.02.2016

4. Interaktiivse reklaami loomine algusest lõpuni

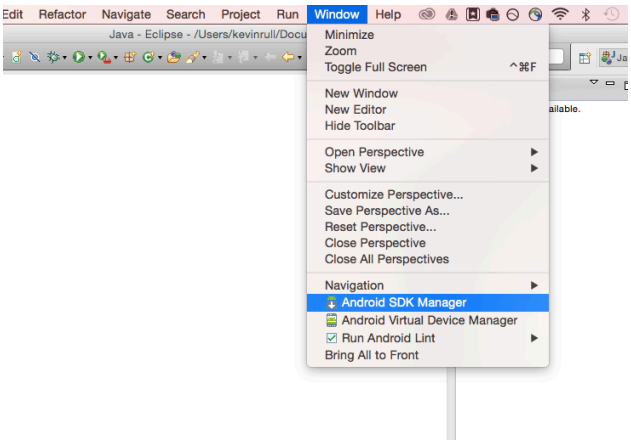
Reklaami loomine esimest korda näeb välja nii, et tuleb installida nutiseadmele CurlifyCMS rakendus, millega on võimalik oma loodavat reklaami näha ning katsetada, kuidas see tegelikult käes hoides tundub. CurlifyCMS rakenduse installimiseks ei ole vaja midagi muud, kui vähemalt 4.0 tarkvaraversiooniga Android nutiseadet ning MicroUSB juhet. Hakkama saab ka Apple iPhone-ga, kuid iOS-ile installimiseks on vaja Apple Developer kontot ning arvutit, millel on installitud OS X operatsioonisüsteem ja Xcode. Kindlasti on ka vaja Curlify raamistiku algfaile, mida saavad kasutada ainult teatud arendajad sealhulgas ka töö autor.

4.1 CurlifyCMS paigaldamine ja ülesseadmine nutiseadmele

Selleks, et CCMS keskkonnas koodi kirjutav arendaja oma töö vilju näeks, peab ta oma arvuti seadeid natuke seadistama ja mõningaid asju paigaldama, kui neid ennem paigaldatud ei ole. Kuid midagi üle jõu käivat ei tohiks olla ja kui ennem asi korra läbi tehtud, läheb järgmine kord juba kiiremini.

4.1.1 Eclipse ja Android SDK paigaldamine

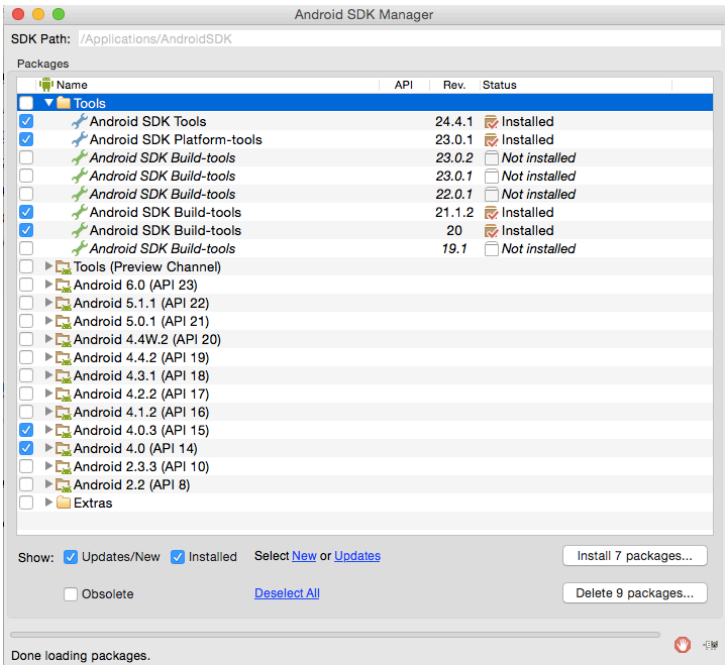
Antud töös käsitletakse rakenduse installimist Android nutiseadmele. Kõigepealt tuleks installida arvutisse Eclipse aadressilt https://eclipse.org/. Eclipse installitud, tuleks see avada. Programmi käivitamisel pakutakse vaikimisi töökausta, see võib jääda, sest otseselt Eclipsega tööd ei pea tegema. Kui programm on avatud, siis ülevalt menüüribalt tuleb valida Window ning edasi Android SDK Manager (vt joonis 2).



Joonis 2 Android SDK Manager'i valik Eclipse menüüribalt

Kui on avanenud Android SDK Manager, siis tuleks valida paketid, mida soovitakse alla

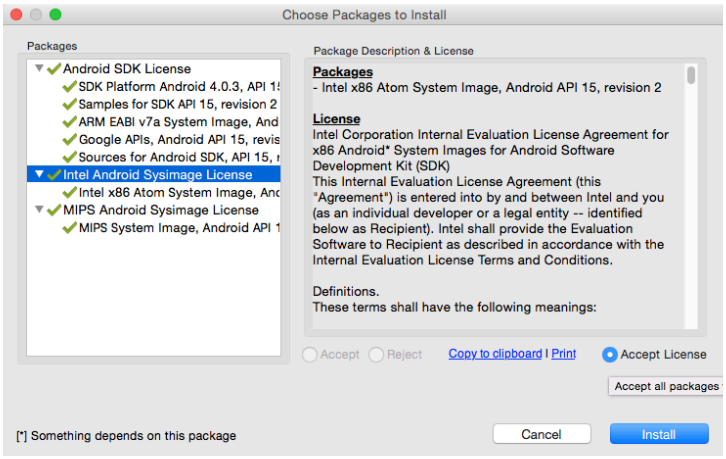
laadida ja paigaldada. Tools kaustas on osad paketid mis tuleb kindlasti paigaldada: Android SDK Tools ja Android SDK Platform-tools. Android SDK Build-tools valik on enda teha, millist versiooni soovitakse paigaldada. Alates 20. versioonist sobib kõik. Antud töö jaoks on kasutatud versiooni 21.1.2. Edasi on vaja valida Android API paketi valimine. See sõltub nüüd nutiseadmest, kuhu soovitakse rakendust installida, täpsemalt nutiseadme Android versioonist. Õige Android versioon tuleb valida ning edasi juba Install *nr*[11] packages (vt joonis 3).



Joonis 3 Valitud paketid ning võib edasi minna paigaldama

[11] Valitud pakettide arv, mida hakatakse paigaldama

Järgmisel vaatel tuleb nõustuda õigustega. Paremalt poolt valida litsents, see läbi lugeda ja nõustuda või vastasel korral paketti ei paigaldata (vt joonis 4). Edasi tuleb vajutada Install nupule ning vajalikud paketid laetakse alla ning paigaldatakse teie arvutisse.



Joonis 4 Litsentsidega nõustumine

4.1.2 Käsurea seadistamine ja rakenduse esimene paigaldus

Selleks et konstrueerida paigaldatav rakendus, tuleb muuta käsurea PATH asukoht kausta, kuhu sai installitud Android SDK. Selleks tuleb käivitada näitena toodud koodirida (vt koodinäide 1) käsureal.

```
export PATH=/Applications/AndroidSDK/platform-tools:/Applications/AndroidSDK/build-tools/android-4.4W:/Applications/AndroidSDK/platform-tools:$PATH
```

Koodinäide 1 Käsureal käivitatav koodirida

Nagu näitest näha võib, siis sellel arvutil paigaldati AndroidSDK Applications kausta ning see on kaust, mis võib erineda olenevalt kuhu AndroidSDK paigaldatakse. Lisaks tuleb suunduda käsureal kausta, kus on vajalikud Curlify CMS platvormi failid. Jõudes sinna kausta, tuleb käivitada koodirida (vt koodinäide 2) käsureal. Programm teeb vajalikud toimingud selleks, et kõigepealt ehitatakse õige apk[12] fail ning seejärel automaatselt paigaldatakse see Android seadmesse, mis on arvutiga ühendatud USB[13] kaudu.

```
sh install.sh
```

Koodinäide 2 Rakenduse paigaldamiseks vajalik käivitada

[12] Android platvormi rakenduste faililaiend
[13] Universal Serial Bus(USB) on ühendus, mille kaudu saab ühendada välisseadmeid arvutiga http://www.usb.org/home 03.04.16

Selleks, et hõlbustada töö kulgemist ning ka saaks logida välja infot, mis rakenduses toimub, tuleks avada uus käsurea aken ning ka seal muuta PATH asukoht koodinäites 1 näidatud kausta. Edasi tuleks käivitada ADB(Android Debug Bridge) logcat[14] (vt koodinäide 3) funktsioon, mis siis suudab kuvada ja filtreerida Android süsteemi logi. Filtreerimiseks lisatakse `-s` filtreering, mis ei logi välja süsteemi vaikimisi infot, `liblo` on lihtsalt nimetus raamistiku debug[15] süsteemist. Kindlasti tuleb telefonis ära seadistada arendaja valikud (Developer options). Android 4.2 ja uuema versiooni puhul tuleb kõigepealt liikuda *Settings – About phone* ning klikata seitse korda valikul nimega *Build number*. See teeb võimalikuks üldse arendaja valikute kuvamise. Järgmiseks tuleb liikuda *Settings – Developer options* ning aktiveerida sealt USB debugimine[16].

```
adb logcat -s liblo
```

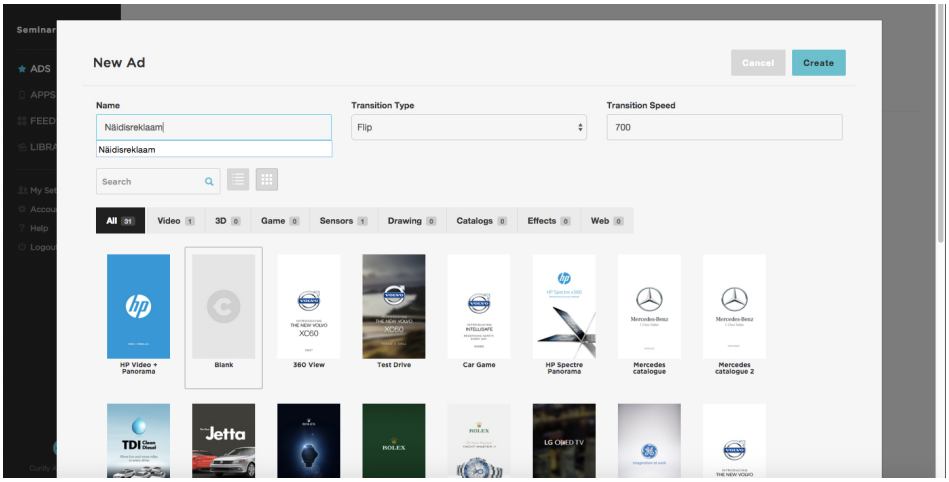
Koodinäide 3 ADB logcat funktsioon süsteemi logimiseks

4.2 Näidisreklaami loomine CurlifyCMS keskkonnas

Näidiseks on valitud kõige lihtsama raskusastmega, kuid samaaegselt ka kõige intrigeerivama sisuga reklaam. Näites kasutatakse ühte 3D[17] telefoni, mida kasutaja saab vaadata vabalt valitud nurga alt. Lisaks on kasutatud ühte hinnasilti ning ühte taustapilti.

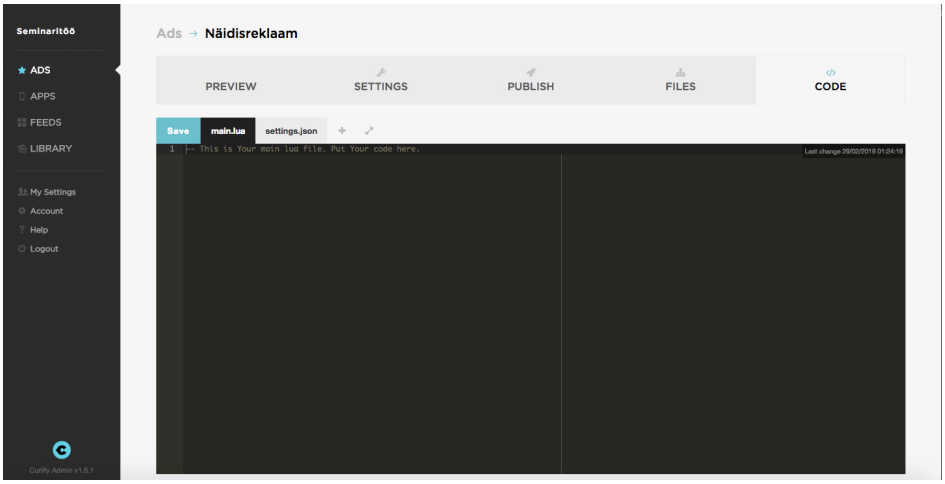
Kõigepealt tuleb luua uus reklaam, klikates *New Ad* nupule, kus edasi on võimalik valida ka mõni reklaami algvorm, kuid neid hetkel ei ole kasutatud. Tuleb muuta ära ainult nimi, mis on vabalt valitav (vt joonis 5) ning siis klikata nupul *Create* ja reklaam ongi tehtud.

[14] http://developer.android.com/tools/help/logcat.html 26.02.16
[15] Programmides vigu avastama, lokaliseerima ja kõrvaldama http://www.vallaste.ee 03.03.16
[16] http://developer.android.com/tools/device.html 03.03.16
[17] Kolmemõõtmeline, ruumiline objekt



Joonis 5 Reklaamile nime andmine

Nüüd kõige tähtsam osa reklaamist – koodi kirjutamine. Klikates äsja loodud reklaami peale, tuleb liikuda *Code* sektori peale ning seal olevas koodiredaktoris (vt joonis 6) tulebki raamistikule vastavat koodijuppi kirjutama hakata.



Joonis 6 Hetkel tühi olev koodiredaktor

Reklaami loomisel midagi muud raskemat ei olegi. Tuleb luua vajalikud pildifailid, mis hoolitsevad reklaami välimuse eest ning teiseks pooleks on koodiread, mis tagavad vajaliku kasutajakogemuse.

Antud näitereklaami funktsionaalsuses on silmas peetud kõige lihtsamat võimalust kasutaja intregeerimiseks. Reklaami alguses midagi erilist ei toimu. Telefon pööratakse õigetpidi ning ennem, kui kasutaja ekraani ei puuduta, telefoni hakatakse vaikselt näitama, ühest ja teisest küljest, seda vaikselt pöörates, et tähelepanu püüda (vt joonis 7). Niipea, kui kasutaja ekraani puudutab toimub mitu asja. Esiteks, animeeritakse hinnasilt nähtamatuks, et see ei segaks telefoni vaatamist. Teiseks, skaleeritakse telefon suuremaks, et tuua ese kasutajale lähemale.

Kolmandaks, kui kasutaja liigutab näppu ekraanilt tõstmata, pööratakse telefoni vastavalt kasutaja näpu asukohale ekraanil (vt joonis 8). Tõstes näpu ekraanilt, skaleeritakse telefon algsuurusesse tagasi, hinnasilt tuuakse taaskord nähtavale ning käivitakse uuesti funktsioon, mis käivitati reklaami alguses, selleks, et püüda kasutaja pilke.

Joonis 7 Telefoni olek hetkel, kui kasutaja ei puuduta ekraani

Joonis 8 Telefoni olek hetkel, kui kasutaja liigutab näppu ekraanil

5. Curlify raamistik

Raamistiku töö tagab OpenGL ES[18] ja Lua programmeerimiskeele kasutamine. Raamistikus on olemas mitmeid valmis mooduleid, tänu millele käib reklaami ehitamine tegelikult väga kiirelt ja kergelt. Käesoleva töö jaoks loodud näidisreklaamis on kasutatud ainult nelja erinevat moodulit:

- *Object*
- *Animator*
- *Image*
- *Mesh*

Object moodul, on peamiselt reklaami alus. Kõik teised moodulid sisaldavad objekt mooduli peamisi elemente (positsioon, rotatsioon, skaleering, igasugused ekraanipuudutused). Luues *Object* tüüpi muutuja, ei kuvata erkaanile mitte midagi, sest vaikimisi puudub objektil tekstuur.

Animator moodul on vajalik objektide animeerimiseks, ehk nende liigutamiseks ühest kohast teise.

Image moodul on teisena kõige rohkem kasutatav moodul Curlify reklaamides. See hoolitseb selle eest, et *jpg* või *png* pilt oleks korrekteselt kuvatud ekraanil, ehk kuvab etteantud pildifaili ekraanile.

Mesh on vähemkasutatud moodul, sest seda on ka kõige keerulisem üles sättida. Selleks on vaja 3D objekti faili, millel on olemas ka UV-mapitud[19] tekstuur. Kuid, kui objekt on olemas ja vastav tekstuurifail ka, siis ei tohiks probleeme tekkida.

Üldiselt raamistik *renderdab*[20] pilti 60 kaadrit sekundis, olenevalt seadme jõudlusele, tänu sellele on ka animatsioonid ning pildi liikumine sujuv.

[18] 2D ja 3D graafika funktsioneerimiseks mobiilsetel seadmetel vajalik API https://www.khronos.org/opengles/ 29.02.2016

[19] UV-mapping – 2D tekstuuri paigutamine 3D objektile nii, et 3D objekt lõigatakse lahti 2D-sse ja sinna peale paigutatakse tekstuur. http://wiki.blender.org/index.php/Doc:2.4/Manual/Textures/Mapping/UV 29.02.2016

[20] Andmete teisendamine kuvamiseks sobivasse vormingusse. http://www.vallaste.ee 29.02.2016

Kokkuvõte

Töö eesmärgiks oli avardada lugejat aru saama, millised on interaktiivse reklaami võimalused. Näidata kasutajatele, milliseid reklaame on tegelikult võimalik luua ning kui kiiresti ja lihtsalt käib see Curlify platvormi peal.

Lisades on toodud välja ka reklaami kood täielikus pikkuses ning ka pildimaterjalid, mida oli kasutatud antud töö näidisreklaami loomisel.

Käesoleva töö edasiarendus võib näiteks olla kasvõi tänapäeval üha rohkem kasutatavate HTML-5 tüüpi reklaamide sügavuti võrdlemine NEXD tüüpi reklaamidega Curlify platvormil.

Töö teostamine avardas autori silmaringi bännerreklaamide ja ka interaktiivsete reklaamide ajaloost ning üldisest toimimisest.

Kasutatud kirjandus

D'Angelo F. (2009, 26. oktoober). Happy Birthday, Digital Advertising. Loetud 27. oktoober 2015 aadressil http://adage.com/article/digitalnext/happy-birthday-digital-advertising/139964/

Wasserman T. (2013, 9. august). This Is the World's First Banner Ad. Loetud 29. veebruar 2016 aadressil http://mashable.com/2013/08/09/first-banner-ad/#q2fCTid8XkqY

Chaffey D. (kuupäev puudub). Display advertising clickthrough rates. Loetud 29. veebruar 2016 aadressil http://www.smartinsights.com/internet-advertising/internet-advertising-analytics/display-advertising-clickthrough-rates/

Rooma-Katoliku Kirik. (2015, 16. november). Joseph Ratzingeri 2015. auhinnad läksid Liibanoni ja Brasiiliasse. Loetud 29. veebruar 2016 aadressil http://www.katoliku.ee/index.php/et/esyndmused/v%C3%A4lismaa/1051-joseph-ratzingeri-2015-auhinnad-l%C3%A4ksid-liibanoni-ja-brasiiliasse

Lua. (kuupäev puudub). About Lua. Loetud 27. oktoober 2015 aadressil http://www.lua.org/about.html

Kohan B. (2010, 15. november). What is a Content Management System (CMS)? Loetud 29. veebruar 2016 aadressil http://www.comentum.com/what-is-cms-content-management-system.html

Seadle M. (2005, 5. detsember). Content management systems. Loetud 29. veebruar 2016 aadressil http://edoc.hu-berlin.de/oa/articles/reXWUcDNwYelM/PDF/22sWVsHpHgNRQ.pdf

Li H., & D. Leckenby J. (2005, oktoober). Internet Advertising Formats and Effectiveness. Loetud 29. veebruar 2016 aadressil https://brosephstalin.files.wordpress.com/2010/06/ad_format_print.pdf

Investopedia. (kuupäev puudub). Interactive Media. Loetud 29. veebruar 2016 aadressil http://www.investopedia.com/terms/i/interactive-media.asp

Investopedia. (kuupäev puudub). Banner Advertising. Loetud 29. veebruar 2016 aadressil http://www.investopedia.com/terms/b/banneradvertising.asp

Krustok I. (2015, 9. juuli). American Express Case Study. Loetud 29. veebruar 2016 aadressil http://www.nexd.ee/american-express-case-study/

LISAD

Näidisreklaami jaoks loodud koodiread lisakommentaaridega.

```
-- impordime vajaminevaid mooduleid raamistikust
local object = require("object") -- objekti moodul
local animator = require("animator") -- animatsiooni moodul
local image = require("image") -- pildi moodul
local mesh = require("mesh")-- 3D objektide moodul

-- move funktsioon, mis võtab sisendandmeteks objekti hetke asukoha,
objekti lõppasukoha ning liikumise kiiruse ja tagastab asukoha kerge sujuva
viitega
local function move(source,target,timedelta)
  if timedelta > 100 then timedelta = 100 end
  local dir = 1
  if source > target then dir = -1 end
  local moveamnt = (-dir*timedelta / 390) * math.abs(source-target)
  if math.abs(moveamnt) < 0.00001 then source = target else source =
source-moveamnt end
  if dir == 1 and source > target then source = target end
  if dir == -1 and source < target then source = target end
  return source
end

-- teeme uue lokaalse funktsiooni new ja tagastame selle programmile,
sellest saab siis funktsioon, mis käivitatakse korra, kui reklaam avatakse
local function new()
  local myapp = object.new("Interactive ad") -- lokaalne muutuja(objekt)
myapp, mis tagastatakse new funktsioonist
-- põhilise funktsionaalsuse koodi algus
  myapp.startx = 0 -- myapp külge pandud muutuja, mille abil saame telefoni
õigesti pöörata x-telge pidi
  myapp.starty = 0 -- myapp külge pandud muutuja, mille abil saame telefoni
õigesti pöörata y-telge pidi

  local bg = myapp:add(image.new("bg.jpg")) -- lokaalne muutuja bg, mis on
pilt tüüpi ning mille abil kuvame telefoni taha taustapildi
```

```
  local phone = myapp:add(mesh.new("phone_white.obj")) -- lokaalne muutuja phone, mis on siis 3D objekt ning mis on peamine objekt
  phone.roty = 0 -- phone külge pandud muutuja, mille abil määrame, kuhu poole peab telefon pöörama y-telge pidi
  phone.rotx = -math.pi/2 -- phone külge pandud muutuja, mille abil määrame, kuhu poole peab telefon pöörama x-telge pidi, esialgu -math.pi/2 sellepärast, et telefon oleks ekraaniga vaataja poole
  -- skaleerime telefoni suurust vastavalt vajadusele, kuna tegemist 3-mõõtmelise objektiga, siis tuleb kõiki telgi suurendada võrdselt, vaikimisi ühik on 1
  phone.scale.x, phone.scale.y, phone.scale.z = 2, 2, 2

  local price = myapp:add(image.new("price.png")) -- lokaalne muutuja price, tavaline pildi tüüpi objekt
  price.position.y = 300 -- määrame hinna asukohaks y-telge pidi 300 pikslit allapoole keskkohast

  -- kirjeldame ära telefoni step funktsiooni, mis käivitatakse kuni 60 korda minutis, olenevalt seadme jõudlusest
  -- vajalik selle jaoks, et oleks sujuv telefoni pööramine vastavalt kasutaja näpu liigutusele
  phone.step = function(self,timedelta)
    -- kasutame telefoni rotatsiooni muutmiseks move funktsiooni, mis teeb liikumise sujuvamaks
    phone.rotate.y = move(phone.rotate.y, phone.roty, 20) -- y-telje rotatsioon
    phone.rotate.x = move(phone.rotate.x, phone.rotx, 20) -- x-telje rotatsioon
  end

  -- kirjeldame ära myapp relativePress funktsiooni, mis käivitatakse iga kord, kui kasutaja vajutab ekraanile, x ja y on tagastatavad muutujad vastavalt kasutaja sõrme asukohale ning self viitab myapp'le, kuid seda antud kontekstis ei kasutata
  myapp.relativePress = function(self,x,y)
    phone.roty = phone.rotate.y
    phone.rotx = phone.rotate.x
    myapp.startx = x -- määrame näpu puudutuse alguseks x'i x-teljel
    myapp.starty = y -- määrame näpu puudutuse alguseks y'i y-teljel
    phone.anim:stop() -- ennem uue animatsiooni määramamist, igaks juhuks lõpetame kõik käimasolevad animatsioonid
```

```
    -- animeerime telefoni, kõik kolm telge, animatsiooni aeg ja
animatsiooni ease ehk liikumise sujuvus
    phone.anim:animate(phone.scale,{x=3.5, y=3.5, z=3.5, time=500,
ease=animator.inOutQuad}) -- skaleerime telefoni suuruse 3.5, ajaga 500
millisekundit
    price.anim:stop()
    price.anim:animate(price, {alpha=0, time=500, ease=animator.inOutQuad})
-- animeerime hinnasildi läbipaistvuse nulli, et see ei segaks telefoni
vaatamist
  end

  -- kirjeldame ära myapp relativeDrag funktsiooni, mis käivitatakse iga
kord, kui kasutaja liigutab oma näppu ekraanil, seda üles tõstmata. x ja y
on tagastatavad muutujad vastavalt sõrme asulohale
  myapp.relativeDrag = function(self,x,y)
    -- määrame telefoni telgedele uued rotatsiooni punktid
    phone.roty = phone.roty - (x - myapp.startx)/100 -- siin arvutame maha
kasutaja näpu alguspositsiooni, et telefoni pööramine oleks vastavuses
kasutaja näpu x-telje asukohaga ja jagame valitud arvuga 100, et pööramise
kiirust vähendada.
    phone.rotx = phone.rotx - (y - myapp.starty)/100 -- siin arvutame maha
kasutaja näpu alguspositsiooni, et telefoni pööramine oleks vastavuses
kasutaja näpu y-telje asukohaga ja jagame valitud arvuga 100, et pööramise
kiirust vähendada.
    myapp.startx = x -- määrame uuesti näpu alguskoha x-teljel, selleks, et
pööramine oleks tundlikum ja täpsem
    myapp.starty = y -- määrame uuesti näpu alguskoha y-teljel, selleks, et
pööramine oleks tundlikum ja täpsem
  end

  -- kirjeldame funktsiooni relativeRelease myapp objektile, mis
käivitatakse iga kord, kui kasutaja tõstab näpu ekraanilt, x ja y on
tagastatavad muutujad, kus näpu asukoht viimati oli.
  myapp.relativeRelease = function(self, x, y)
    phone.anim:stop() -- igaks juhuks lõpetame käimasolevad animatsioonid.
    phone.anim:animate(phone.scale,{x=2, y=2, z=2, time=500,
ease=animator.inOutQuad}) -- animeerime telefoni suuruse algsuuruseks
tagasi.
    price.anim:stop()
    price.anim:animate(price, {alpha=1, time=500, ease=animator.inOutQuad})
-- animeerime hinnasildi läbipaistvuse tagasi ühe peale
```

```
    phone.wiggle() -- käivitame telefoni pöörlemise funktsiooni
    -- pöörame telefoni igal juhul otseks x-telge pidi onComplete
funktsioon käivitatakse siis kohe, kui animatsioon on lõpule jõudnud
    phone.anim:animate(phone.rotate, {x=-math.pi/2, time=4000,
ease=animator.inOutQuad, onComplete=function()
      -- määrame telefoni lõpprotatsiooniks -math.pi/2 radiaani, ehk -90
kraadi
      phone.rotx = -math.pi/2
    end})
  end

  -- kirjeldame telefoni pöörlemise funktsiooni, mis siis lihtsalt pöörab
telefoni ühele poole ja teisele poole, kui kasutaja ei liiguta telefoni
  phone.wiggle = function()
    local finaly = math.pi/4 -- määrame pöörderaadiuseks 45 kraadi
    if phone.rotate.y > 0 then finaly = -math.pi/4 end -- kui telefoni
rotatsioon on üle nulli siis määrame pöörama teisele poole
    -- animeerime rotatsiooni ning animatsiooni lõpus kutsume välja uuesti
sama funktsiooni
    phone.anim:animate(phone.rotate,{y=finaly,time=4000,
ease=animator.inOutQuad, onComplete=phone.wiggle})
  end

  phone.wiggle()
-- põhilise funktsionaalsuse koodi lõpp
  return myapp -- tagastame kogu myapp objekti
end

return {
  new = new,
}
```

Näidisreklaamis kasutatud pildimaterjalid ja 3D objektid on kättesaadavad aadressilt http://www.tlu.ee/~rullkev/seminaritoo/. Lisaks on lisatud ka videofail, mis demonstreerib reklaami toimimist ning ka üks apk fail Android seadmete omajatele, et asja oma käega järgi proovida.