OULUN AMMATTIKORKEAKOULU

Palosaari Mikko

PELISOVELLUS MOBIILILAITTEELLE REACT NATIVE -SOVELLUSKEHYK-SELLÄ

PELISOVELLUS MOBIILILAITTEELLE REACT NATIVE -SOVELLUSKEHYKSELLÄ

Palosaari Mikko
Opinnäytetyö
Syksy 2019
Tietojenkäsittelyn tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietojenkäsittely, Web-ohjelmointi

Tekijä(t): Palosaari Mikko
Opinnäytetyön nimi: Pelisovellus Mobiililaitteelle
Työn ohjaaja: Viitala Matti
Työn valmistumislukukausi ja -vuosi: Syksy 2019 Sivumäärä: 28

Toimeksiantajana toimi AXO-Service Oy (AXO), peliteknologiaa myyvä yritys. AXO:n perustaja, toimitusjohtaja ja pääomistaja on Atte Kananen Kuusamosta. Tässä opinnäytetyössä kuvataan ainutlaatuista Pikselipeli-teknologiaa, joka mahdollistaa informaation pelillistämisen. Pikselipeliteknologiaa kuvataan tässä opinnäytetyössä paljastamatta tarkempia ohjelmistollisia ja muita yksityiskohtia, siten että se kunnioittaa toimeksiantajan ja opinnäytetyöntekijän välistä sopimusta.

Opinnäytetyön tavoitteena on toteuttaa prototyyppinen mobiilisovellus perustuen AXO:n pikselipeliteknologiaan. Jossa pelaaja maalaa pikseleitä, joita sitten ohjelma laskee ja toteuttaa sen mukaista todennäköisyyslaskentaa. Mobiilisovelluksen tarkoitus oli laajentaa saatavuutta yrityksen jo aikaisemmin julkaistuille internettipeleille, toteuttamalla samantyylisen pikselipeliteknologiaan perustuvan pelin myös Android-laitteistolle.

Työssä hyödynnettiin tuotettavan mobiilisovelluksen tuotannon tukena Atelta / AXO:lta saatua taustatietoa, mm. jo olemassa olevilta AXO-games nettisivuilla toimivista pelisovelluksista, joissa tekstiä ja kuvaa voi pelata, sekä peliteknologiaan liittyvästä aineistosta ja analyysistä.

Työn toteutus tapahtui React Native -sovelluskehyksellä ja siihen rakennetulla React Native Clientillä sekä Canvas-ominaisuuksilla, jotta pikseleitä pystyttiin käsittelemään samanlailla kuten HTML5 web-selaimessa. Piirto-ominaisuus eli pikseleiden maalaaminen toteutettiin alusta lähtien itse, koska monien lisäosien käyttö, joita React Native Cli käyttää, oli joko liian sekavaa käyttää halutulla tavalla, tai dokumentaatio oli liian puutteellista. Päätuloksena syntyi prototyyppinen mobiilisovellus/komponentti, jossa piirros on pelillistetty.

Koko sovelluksen teko yksin alusta loppuun oli haastavaa, siksi se myös oli todella hyvää harjoitusta tulevaisuutta varten. Ongelmia ilmeni paljonkin, mutta suurin haaste oli aikaraja, joka vääjäämättä rajoitti sitä, miten sovelluksen tulisi toimia.

Asiasanat: Mobiili, kehitys, JavaScript, pelit, mobiilipelit

ABSTRACT

Oulu University of Applied Sciences
Bachelor degree programme in Business Information Systems, Web-development

---

Author(s): Palosaari Mikko
Title of thesis: Pixel game for Mobile device
Supervisor(s): Viitala Matti
Term and year when the thesis was submitted: Autumn 2019 Number of pages: 28

The objective of the thesis was to implement a prototype Android mobile game application for AXO-Games company. AXO-Games is a little game- and gamification company from Kuusamo Finland. This company did not have previous mobile applications, so this was their first one. So that they can get in touch with higher audience rates and higher player base.
This Android mobile application uses same pixel-game-technology than clients previous games published in the internet as a web-based-games.

Main component that was developed is the prototype mobile application where user input drawings are made playable, to show a little bit about clients unique pixel-technology. Application will calculate all painted pixels and make propability calculations based of that.

During the writing of the thesis, Android game application was developed with React Native framework that allows the use of different built in npm-packaces, such as Canvas, so that we could handle the pixels properly. Drawing tool had to be build from the scratch, because there was not good and simple enough npm-packace for the requirements or the documentation was lacking. Good references for thesis work was got from the company's own website; axogames.com.

The whole development of the application from start to finish was difficult task, there was many problems, but all in all, everything went mostly good, and thats why it also was good practice for the future.

Keywords: Mobile, development, JavaScript, games, mobile games

SISÄLLYS

SANASTO

Android

Googlen kehittämä Linux-pohjainen avoimen lähdekoodin käyttöjärjestelmä mobiililaitteille.

Android Studio

Virallinen ohjelmointiympäristö Android-käyttöjärjestelmälle.

API

Application Programming Interface. Ohjelmointirajapinta, joka mahdollistaa eri ohjelmien keskustelun keskenään.

APK

Android Package Kit. Android-sovellusten tiedostotyyppi.

AVD

Android Virtual Device. Emulaattori, joka simuloi tietokoneella mobiililaitetta.

DOM

Document Object Model. Hierarkiapuu HTML-dokumentista.

Funktio

Ohjelmointikielen osa, joka kuvaa jonkin toiminnon ja jolla on oma nimi eli tunnus.

Java

Laitteistoriippumaton oliopohjainen, tyypitykseltään vahva yleiskäyttöinen ohjelmointikieli.

JavaScript

Pääasiassa web-ympäristössä käytettävä dynaamista toiminnallisuutta lisäävä komentosarjakieli. React Nativen ohjelmointikieli.

JDK

Java Development Kit. Javan Ohjelmistokehityspaketti.

NetBeans

Apachen kehittämä ohjelmointiympäristö.

Node.js

Alustariippumaton ajoympäristö JavaScript-koodin suorittamiseen palvelimella nettiselaimen sijaan.

NPM

Node Packace Manager. Node.js:n paketinhallintajärjestelmä.

Pikselipeli-teknologia

Toimeksiantajan kehittämä teknologia.

Python

Oliokeskeinen, tulkattu, korkean tason ohjelmointikieli dynaamisella semantiikalla.

React

Facebookin kehittämä JavaScript-kirjasto web-käyttöliittymien tekoon.

React Native CLI

Nodessa tai Shellissä asennettava React Native npm komentoriviliittymä.

Responsiivisuus

Verkkosivuston mukautuminen kaikille päätelaitteille sopivaksi.

UX

User Experience. Käyttäjäkokemus, eli kuinka käyttäjä näkee ja kokee tuotteen.

Visual Studio Code

Avoimen lähdekoodin monialustainen sekä kevyt, ohjelmoijille tarkoitettu tekstieditori.

1 TYÖN LÄHTÖKOHDAT

1.1 Toimeksiantaja

Toimeksiantaja, AXO-Service Oy (AXO) on peliteknologiaa myyvä yritys. AXO:n perustaja, toimitusjohtaja ja pääomistaja on Atte Kananen (Atte). Atte / AXO omistavat yksin tai yhdessä myytävät peliteknologiat. Atte / AXO omistaa tässä opinnäytetyössä kuvattujen peliteknologioiden ja niiden osien kaikki oikeudet. AXO on patentoinut pelimenetelmä-teknologian (FI, US, RU ja AU), jota voidaan hyödyntää osana Pikselipeli-teknologiaa. (AXO-Service Oy. Keskustelut 2019.)

Toimeksiantajalla on myös omat nettisivut, jossa esitellään heidän aikaisemmin suunnittelemiaan sekä tekemiään nettipelejä perustuen pikselipeli-teknologiaan. www.axogames.com. Pikselipeliteknologia tarkoittaa esimerkiksi sitä, että kuva, piirros tai tekstin osa ovat interaktiivisia, eli käyttäjä voi manipuloida näitä objekteja vaikka piirtämällä. Nämä objektit koostuvat pikseleistä, ja objektin pikseleiden seasta voidaan lukea ja sattumanvaraisesti arpoa voittopikseleitä.

1.2 Tavoitteet

Opinnäytetyön tavoitteena oli toteuttaa mobiili-pelisovellus toimeksiantajalle, perustuen toimeksiantajan Pikselipeli-teknologiaan käyttäen rakennettua piirtotyökalua pääkomponenttinaan. Työn tavoitteena oli sovellus, jonka jatkokehitys tulevaisuudessa on ongelmitta mahdollista, ja koska aikaa oli rajoitetusti, tämän sovellus voidaan luokitella prototyypiksi. (AXO-Service Oy. Keskustelut 2019.)

Pikselipeli-teknologia ratkaisee ongelman ”Miten informaatio voi olla peli?” Toimeksiantajan tarkoitus on hyödyntää GooglePlay-sovelluskauppaa peliteknologioidensa myyntikanavana. Tämä helpottaisi yritystä saatavuudessa, näkyvyydessä sekä tunnettavuudessa. Toimeksiantaja on keksinyt ko. Pikselipeli-teknologian ja hän on ostanut ulkopuolista koodaus- tai muuta palvelua tähän ja muihin peliteknologioihinsa liittyen.

Pelin tarkoitus on hyödyntää Pikselipeli-teknologiaa visuaalisesti, eli tuoda käyttäjälle näytille sen toimintaa. Tässä tapauksessa kyseessä on hyvin yksinkertainen pikselipeli, joka toimii samaan tapaan kuin lotto- tai muu vastaava peli, jossa käyttäjä ei voi tietää etukäteen voittaako vai ei. Pelialueena toimii piirtoalue, joka on rakennettu pikseleistä ja johon käyttäjä voi maalata pikseleitä sormillansa. Logiikka perustuu siihen, kuinka ison osan, tai kuinka monta pikseliä pelialueesta käyttäjä on maalannut suhteessa tyhjään pikseleistä koostuvaan canvakseen, eli taustaan.

Pikselipeli-teknologiaa voidaan hyödyntää osana mitä tahansa sovellusta, jolloin kyseessä olevaan sovellukseen saadaan lisäarvona peliominaisuus. Johtopäätöksenä informaation pelillistäminen Pikselipeli-teknologian avulla mahdollistaa uuden pelaamiskulttuurin ja uuden pelimarkkinan syntymisen. Perusteknologialtaan Pikselipeli-teknologia ei välttämättä vaadi kehitystä, mutta teknologian käyttöönotto osana eri sovelluksia vaatii luonnollisia integrointi toimia, eli sitä kuinka jo valmis ohjelma saadaan toimimaan osana toista ohjelmaa ja vaatiiko se ehkäpä koodin kirjoittamista uudelleen eri kielellä.

Opinnäytetyön kirjallisen osuuden tavoitteena oli tarkastella mobiilisovelluksen kehitystä projektin käynnistyksestä ohjelman julkaisuun saakka, sekä pohtia erilaisia ratkaisuja kehitystyön aikana ilmenneisiin ongelmiin. Raportoida esiin ilmestyviä mahdollisia jatkotoimenpiteitä, jos projektia haluaa viedä eteenpäin tulevaisuudessa.

2 VAATIMUKSET JA RAJAUKSET

2.1 Vaatimukset

Mobiilisovelluksesta tuli olla Android-versio ja se tuli toteuttaa React Native -sovelluskehyksellä. React Native valittiin kehitysympäristöksi siksi, koska sen tiedettiin olevan hyvä ja suhteellisen nopea tapa rakentaa mobiilisovelluksia, sekä se oli yhteensopiva Android-järjestelmille.

Sovelluksen kuuluu toimia siten, että riippumatta miten ja kuinka paljon pelialueelle piirretään/ maalataan pikseleitä, niin sovellus tunnistaa piirtyneet pikselit pelivalintoina. Opinnäytetyön myötä syntyneet tulokset ja palvelut kuuluvat toimeksiantajalle.

Mobiilisovellukseen tuli toteuttaa seuraavat toiminnot.

Käyttöliittymä

Käyttöliittymässä tulee olla toiminnot, joita pelin pelaamiseen vaaditaan, eli pelialue jota käyttäjä voi maalata tai vastaavasti pyyhkiä maalaamistaan vähemmälle, sekä napit arvonnalle ja uudelle pelille.

Piirtäminen

Sovelluksella tulee pystyä piirtämään piirtoalueelle sormia käyttämällä.

Pikselikartan lukeminen

Sovellus tulee pystyä skannaamaan piirtoalue aina käyttäjän niin halutessaan ja tallentamaan kaikki piirtoalueen pikselit talteen pelaamista varten.

Pelaaminen

Sovellus valitsee sattumanvaraisesti voittavat pikselit, kymmenen kappaletta, piirtoalueelta ja antaa sen mukaan palautteen, kuinka moneen pikseliin pelaaja osui.

2.2 Rajaukset

Ihan puhtaasti natiivia sovellusta ei lähdetty tekemään Androidille, koska ensinnäkin tästä ei ollut yhtään aikaisempaa kokemusta. Toiseksi työn määrä olisi ollut niin iso, ettei se olisi mahtunut parin kuukauden sisään. Lisäksi React-Native-Cli:n käyttämien NPM-pakettien eli lisäosien käyttö oli yllättävän vaivalloista: Eivät toimineet toivotulla tavalla, jotkut olivat liian puutteellisesti dokumentoitu,

jotta niitä olisi voitu hyödyntää, sekä omakohtaisen kokemuksen puute NPM-pakettien rakentamisesta vaikutti siihen, että emme alkaneet käyttämään kuin paketteja, jotka osasivat käsitellä Canvas-elementtejä.

Sovelluksen tuli alun perin olla laajempi, mutta ottaen huomioon sen, että rakennettiin ohjelmaa yksin ilman apujoukkoja, sekä työn aikana uusien asioiden opetteluun kuluva aikamäärä, rajoittivat kehitysvaihetta. Tästä johtuen päädyttiin rajaamaan työnkuvaa yllämainittuihin raameihin.

3 TEKNIIKAT JA TYÖKALUT

3.1 JavaScript

JavaScript-ohjelmointikieli sai alkunsa vuonna 1995 Netscape Navigator -selaimessa. Sen kehitti alun perin Netscapen Brendan Eich nimellä Mocha. Nimi ehti olemaan vähän aikaa LiveScript, ennen kuin se muutettiin markkinointisyistä JavaScriptiksi Netscapen liittouduttua Java-kielen kehittäneen Sun Microsystemsin kanssa. (MikaBug 2007.)

JavaScript on oliopohjainen dynaaminen ohjelmointikieli, jolla saadaan lisättyä dynaamisia toimintoja sekä asynkronoituja kutsuja www-sivuille. Soveltuu hyvin vuorovaikutteisten toimintojen luomiseen. Esimerkiksi linkin tai napin ulkoasu voi muuttua, kun hiiren osoitin viedään päälle, sekä erilaiset ”pop-up” -ikkunat ja animaatiot voidaan toteuttaa JavaScriptin avulla. JavaScript ei nimestään huolimatta ole Javaa, vaikka kielten syntaksissa onkin paljon yhteistä. (MikaBug 2007.)

JavaScriptiä käytetään paljon palvelinten verkko-ohjelmoinnissa, kuten Node.js-ajoympäristössä, pelien kehityksessä sekä mobiilisovellusten luomisessa.

3.2 React

React sai alkunsa Facebookin mainonnasta vastaavassa organisaatiossa, jossa hyödynnettiin perinteistä MVC-arkkitehtuuria. React on nykyään suosittu JavaScript-kirjasto web-käyttöliittymien tekoon. Facebook kehitti Reactin ratkaisemaan haasteita, joita ilmenee monimutkaisten käyttöliittymien kehityksessä ja joissa esitettävä data on jatkuvassa muutoksessa, eli ratkaisuja datan esittämiseen käyttöliittymässä. (Facebook Inc. React. 2019.)

React rakennettiin käyttämään deklaratiivia komponentteja. Deklaratiivisuus ohjelmoinnissa tarkoittaa sitä, että kuvaillaan, mitä ohjelmassa tulee tapahtua sen sijaan, että kuinka se tapahtuu. Deklaratiivisuus tekee koodista myös helpommin ennustettavamman ja ymmärrettävän, sekä virheenkorjausystävällisemmän. (Facebook Inc. React. 2019.)

3.3 React Native

React Native on Facebookin kehittämä avoimen lähdekoodin mobiilisovelluskehitykseen perustuva kehys, jonka avulla web-kehittäjät voivat luoda vankkoja mobiilisovelluksia nykyisen JavaScript-tietonsa avulla. Se tarjoaa nopeamman mobiilikehityksen ja tehokkaamman koodinjaon iOS:in, Androidin, ja Webin välityksellä uhraamatta liiaksi loppukäyttäjän kokemuksia tai sovelluksen laatua. (Facebook Inc. React Native. 2019.)

Toimintaperiaatteet ovat virtuaalisesti identtisiä Reactin kanssa, paitsi React Native ei manipuloi DOM:ia samaan tapaan. Eikä myöskään käytä HTML:ää, vaan JavaScriptiä. React Native tarjoaa myös omat JavaScriptillä tehdyt rajapinnat alustojen API-rajapinnoille, joten React Native -sovellukset voivat käyttää alustojen ominaisuuksia, kuten sijaintia tai kameraa. (Facebook Inc. React Native. 2019.)

Reactissa sekä React Nativessa ominaista on Render-funktio, joka tuo eri komponentit näkyviin käyttäjälle. Esimerkki kuviossa 1 näytetään Render-toiminnassa. Render palauttaa näytille objekteja ja komponentteja, joita on käsitelty muualla.

```
render() {
  console.log("test");
  return (
    <View              onPress={() => {
      console.log('You tapped the button!');
    }} style={styles.container}>

      <View style={{ flex: 1, flexDirection: 'row' }}>

        <RNSketchCanvas
          containerStyle={{ backgroundColor: 'transparent', flex: 1 }}
          canvasStyle={{ backgroundColor: 'transparent', flex: 1 }}
          defaultStrokeIndex={0}
          defaultStrokeWidth={5}
          closeComponent={<View style={styles.functionButton}><Text style={{color: 'white'}}>Close</Text></View>}
          undoComponent={<View style={styles.functionButton}><Text style={{color: 'white'}}>Undo</Text></View>}
          clearComponent={<View style={styles.functionButton}><Text style={{color: 'white'}}>Clear</Text></View>}
          eraseComponent={<View style={styles.functionButton}><Text style={{color: 'white'}}>Eraser</Text></View>}
          strokeComponent={color => (
            <View style={[{ backgroundColor: color }, styles.strokeColorButton]} />
          )}
          strokeSelectedComponent={(color, index, changed) => {
            return (
              <View style={[{ backgroundColor: color, borderWidth: 2 }, styles.strokeColorButton]} />
            )
          }}
```

Kuvio 1. Render-funktio toiminnassa

3.4 Visual Studio Code

Visual Studio Code (VS Code) on kevyt, mutta tehokas lähdekoodieditori, jonka Microsoft on kehittänyt Windowsille, Linuxille ja macOS:lle Se sisältää tuen vianetsinnälle, sulautetulle Git- ohjaukselle ja GitHubille, syntaksin korostamiseen, älykkään koodin loppuun saattamiseen, katkelmiin ja koodin uudelleenkehittämiseen. Sisään rakennettu tuki mm. JavaScriptille, TypeScriptille, Node.js:lle, sekä laaja valikoima lisäosia muille kielille kuten C++, C#, Java, Python, PHP. (Visual Studio Code. Docs.)

VS Codea ei pidä sekoittaa Visual Studioon. VS Code on editori, kun Visual Studio on ohjelmointiympäristö. VS Code toimii useimmilla alustoilla sekä on kevyt rakenteinen ja nopea, kun taas Visual Studio toimii vain Windows- ja Mac-laitteilla, eikä ole lainkaan yhtä nopea. Loppujen lopuksi se on käyttäjästä kiinni, mitä ohjelmia haluaa käyttää. (Visual Studio, Support.)

3.5 Android Virtual Device

Android Virtual Device (AVD) on laitekokoonpano, jota käytetään Android Studion Android-emulaattorin kanssa, tarjotakseen virtuaalisen laitekohtaisen ympäristön, jossa Android-sovellukset voidaan asentaa sekä käyttää. Toisin sanoen, virtuaalinen versio mobiililaitteesta tietokoneella. (Android docs. Avd-manager.)

Testikäyttöön mainio työkalu, jos ei halua tai kykene käyttämään konkreettista mobiililaitetta kehitettävän sovelluksen kehitysvaiheessa. Kannattaa pitää mielessä, että AVD vaatii tietokoneelta myös prosessointi tehoja ja muistia.

Kehityksen alkuvaiheessa käytettiin fyysistä laitetta, kun testattiin kuinka kaikki ohjelmat toimivat, mutta lopulta siirryttiin sitten suoraan virtuaalisen laitteen pariin. Se oli jotenkin helpompaa, kun kaikki toiminnot olivat samalla tietokoneella, eikä tarvittu huolehtia ulkopuolisen laitteiston kunnosta ja virrasta. Varsinkin jos ei ole varalla toista puhelinta testaukseen, jonka voisi jättää koko päiväksi tietokoneeseen kiinni.

4 SOVELLUKSEN TOTEUTUS

4.1 Kehitysympäristö

Aluksi lähdettiin tutustumismielessä liikkeelle React Nativen Create React Native App -nimisellä komentorivityökalulla. Tämän työkalun avulla kehityksen voi aloittaa nopeasti ilman kehitysympäristön pystytystä tai konfigurointia. Työkalun avulla oli myös helppo testata oikealla laitteella sovellusta. Laitteeseen tuli vain asentaa Expo-niminen sovellus, tarkistaa, että verkkoyhteys oli sama, ja sitten skannata laitteellaan komentoriville ilmestyvä QR-koodi. Tämän jälkeen koodimuutokset päivittyvät automaattisesti käytettävällä laitteella ja pystyy näkemään heti mitä tapahtuu. Kätevää, mutta Expon käyttäminen tukee pelkästään JavaScript-sovelluksia. Tämä oli tietysti ongelmallista, jos haluaisi käyttää tiettyjä NPM-paketteja, jotka eivät toimineetkaan pelkästään Create React Native App:in sekä Expon avulla. Expo voi antaa esimerkiksi kuvion 2 kaltaisen virheen. (Expo. Examples and tutorials.)

Kuvio 2. Virheraportti Expolla

Seuraavaksi muutettiin hieman projektia, jotta sovelluskehitys pystyi jatkumaan. Hylättiin aikaisemmat työkalut ja siirryttiin käyttämään komentorivityökalua React Native Cli. React Native Cli:n käyttöönotto olikin huomattavasti vaivalloisempaa kuin Create React Native App. Seuraavaksi käydään läpi, mitä kaikkea pitää tehdä aloittaakseen sovelluskehityksen React Native Cli:llä.

4.2 Asennusohjeet

Android-kehitystä React-Native sovelluskehyksellä tehtäessä, seuraavat komentorivityökalut sekä ohjelmat tulee olla asennettuina tietokoneelle:

- React Native Cli
- Node.js
- Python
- JDK
- Android Studio

Yllä mainitut ohjelmat ovat vapaasti saatavilla sekä ladattavissa internetistä ilman korvausta.

4.2.1 Aluksi

Ensimmäisenä suositellaan asentamaan Node.js, komentoriviohjelma, joka toimii samaan tapaan kuin esimerkiksi Power Shell, joka tulee Windows käyttöjärjestelmän mukana. Node.js:ää käytetään aina projektien teossa ja avaamisessa. Ohjelma toimii taustalla ja päivittyy sitä mukaa, kun käyttäjä tekee muutoksia koodiin ja tallentaa tuotoksen.

Toisena suositellaan asentamaan Python, joka vaaditaan siksi, koska Node on osittain kirjoitettu ja rakennettu Pythonilla ja se käyttää Pythonia tiettyjen komentojen ajamiseen. Kolmantena suositellaan asentamaan JDK, jota vaaditaan aina kun halutaan kehittää jotain Java-kielillä. Neljäntenä suositellaan asentamaan Android Studio, vaikka ohjelmoija voi käyttää jotain muuta työkalua koodin kirjoittamiseen, kuten Visual Studiota tai Net-Beansia. Android Studio täytyy silti asentaa, jos haluaa tehdä React Nativella ohjelmia, koska Android Studion mukana tulee muutamia lisäohjelmia, joita vaaditaan työympäristön käyttöönotossa.

4.2.2 Android Studio

Valitaan ”Custom” -vaihtoehto asennusvaiheessa, jotta päästään muokkaamaan asennettavia paketteja. Varmistetaan, että seuraavat asiat ovat ruksattuina, kun jatketaan asentamista;

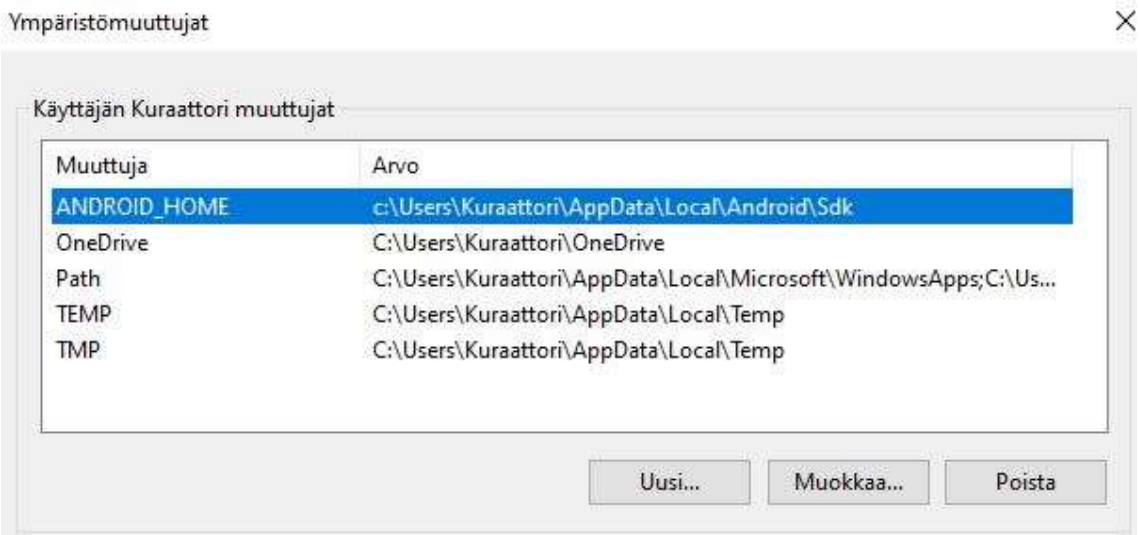
- Android SDK
- Android SDK Platform

- Performance (Intel HAXM)
- Android Virtual Device

Kun asennus on valmis, avataan Android Studio ja etsitään yläpalkista ”Tools”-valikko, sieltä SDK-Manager. Varmistetaan että ”SDK Platforms” -välilehdestä on asennettuina ”Android 9.0 (Pie)”. Laitetaan ruksi ”näytä pakettien sisältö” kohdalle ja varmistetaan, että Android 9.0 sisältä on ruksattuna ”Android SDK Platform 28” sekä ”Intel x86 Atom_64 System Image”. Sitten varmistetaan, että ”SDK Tools” -välilehdestä on asennettuina ”Android SDK Build-Tools”, ”Android Emulator”, ”Android SDK Platform-Tools”, ”Android SDK Tools” ja ”Intel x86 Emulator Accelerator (HAXM installer)”, jos näin ei ole, ne voidaan asentaa. Katsotaan samalla, että ”Android SDK Build-Tools” sisällä on vähintään asennettuna versio numero 28.0.3.

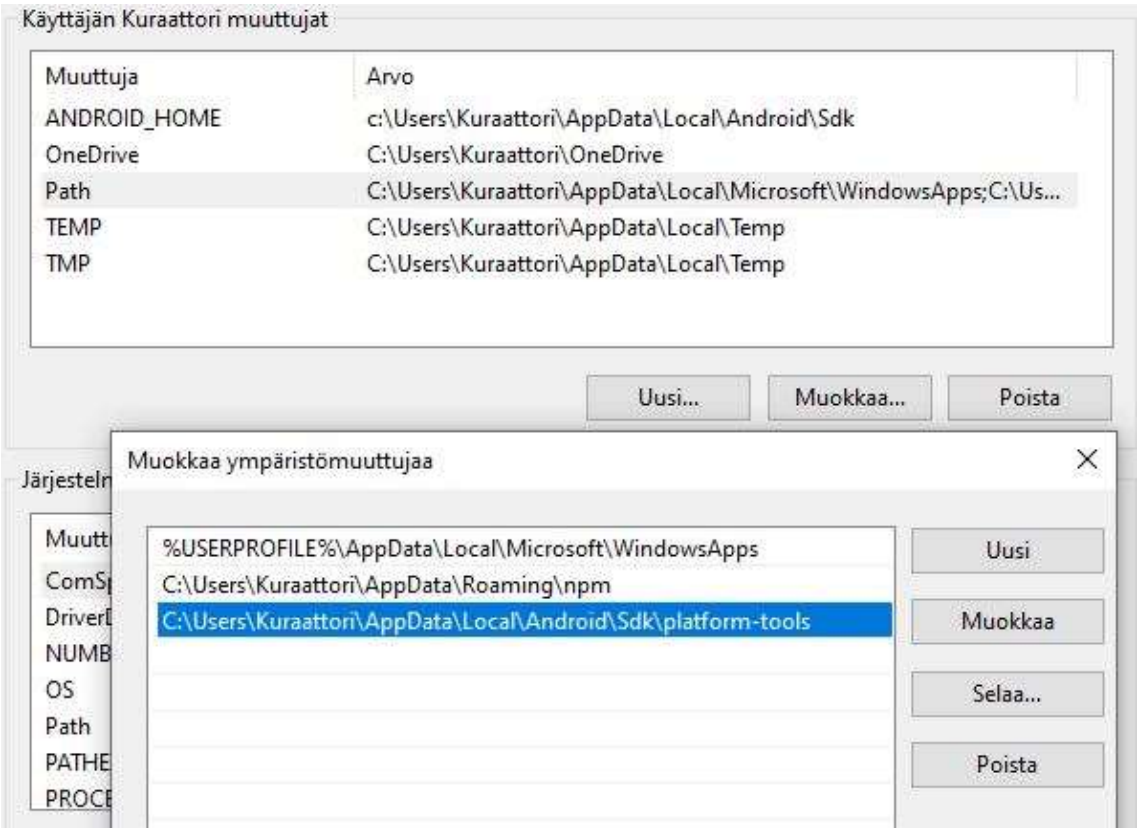
4.2.3 Ympäristömuuttujat

React Nativen työkalut vaativat myös tietokoneen järjestelmän ympäristömuuttujien muokkausta. Kirjoitetaan Windows hakuun ”Ympäristömuuttujia”, valitaan ensimmäinen vaihtoehto, joka aukaisee ”järjestelmän ominaisuudet”. Painetaan ”Ympäristömuuttujat…”-nappia. Sitten painetaan ”Uusi”-nappia ylemmällä Ympäristömuuttuja tasolla ja tehdään uusi ympäristömuuttuja kuvion 3 mukaan, sisältäen muuttujanimen ANDROID_HOME sekä arvon c:\Users\OMA_NIMI\AppData\Local\Android\Sdk.



Kuvio 3. Ympäristömuuttujat

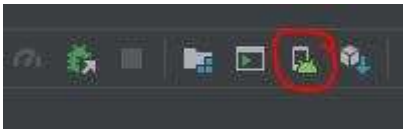
Painetaan ”OK” ja varmistetaan että ympäristömuuttuja on tallennettu, sillä seuraavaksi täytyy muokata toista ympäristömuuttujaa. Valitaan ylemmältä ympäristömuuttujatasolta ”Path” ja painetaan ”Muokkaa…” -nappia. Lisätään seuraava rivi ”c:\Users\OMA_NIMI\AppData\Local\Android\Sdk\platform-tools” kuvion 4 mukaan.



Kuvio 4. Ympäristömuuttujat

4.2.4 Android laite

Työskennellessä voidaan käyttää joko fyysistä tai virtuaalista Android laitetta. Fyysistä laitetta käytettäessä ohjeet vaihtelevat paljon mobiililaitteen merkistä riippuen, joten keskityimme kertomaan tässä, kuinka tehdään virtuaalinen laite. Virtuaalisen laitteen voi tehdä Android Studiolla ”AVD Manager” -napista. Esimerkki Kuvio 5. AVD Manager.



Kuvio 5. AVD Manager

Painetaan ”Create Virtual Device” -nappia luodaksemme uuden virtuaalisen laitteen. Valitaan mikä tahansa puhelin listalta ja painetaan ”Next”. Seuraavaksi valitaan Pie API Level 28 ja painetaan ”Finish” -nappia. Tämän jälkeen listalla pitäisi näkyä uusi virtuaalinen laite, jonka oikealla puolella olevaa vihreää kolmiota painamalla avataan kyseinen laite.

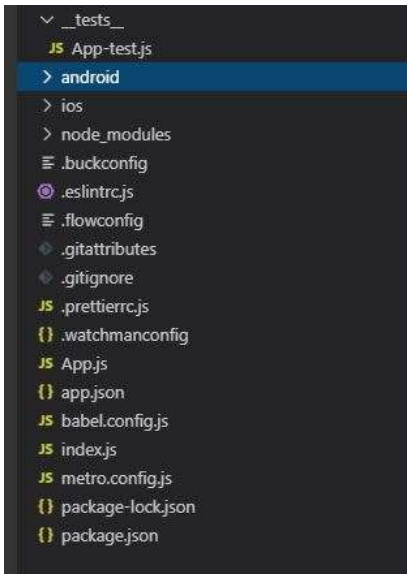
4.3 Uuden projektin aloittaminen

Avataan joko Node.js tai muu vastaava komentokehote ohjelma kuten Power Shell ja kirjoitetaan siellä seuraavat komennot seuraavassa järjestyksessä;

- npm install -g react-native-cli
  Tätä komentoa ei tarvitse kuin kerran käyttää, koska tämä asentaa itse ohjelman.
- react-native init PROJEKTIN_NIMI
  Tällä komennolla luodaan aina uusi projekti, jonka nimen voi itse päättää.
- cd PROJEKTIN_NIMI
  Tällä komennolla siirrytään luodun projektin kansiorakenteeseen.
- react-native run-android
  Tämän komennon avulla avataan oma projekti.

4.4 Projektin kansiorakenne

Yksi projekti sisältää kymmeniä kansioita ja satoja tiedostoja ja laajenee sitä mukaan mitä enemmän toimintoja halutaan käyttää. Projektissa käytetty kansiorakenne on kuvion 6 mukainen.



Kuvio 6. Projektin kansiorakenne

”App.js” -tiedosto sisältää pääsisällön ohjelmasta. Tänne kirjoitetaan koko ohjelman logiikka muuttujineen ja funktioineen. ”Package.json” -tiedosto sisältää kaikki nimet sekä vaadittavat komponentit, jotka ohjelma lataa automaattisesti, kunhan ne ovat tänne määritelty. Yleensä ohjelma tekee nämä automaattisesti, mutta lisätä voi myös manuaalisesti. ”Index.js” -tiedostossa määritellään sovelluksen juurikomponentit (engl. root component). ”Node_modules” -kansiosta löytyvät kaikki React-Nativen tarvitsemat moduulit ja lisäosat. NPM-paketit ladataan myös tähän kansioon.

4.5 Komponentit

React- sekä React-Native sovellukset koostuvat komponenteista, jotka kuvaavat renderöitävää käyttöliittymää. Komponentit mahdollistavat koodin uudelleen käytettävyyden, mikä tekee niiden käytöstä hyvinkin kätevää ja skaalautuvaa. Reactissa renderöidään normaaleja HTML-elementtejä, kun taas React-Nativessa näiden sijaan käytetään alustalle ominaisia komponentteja. Tällainen komponentti on esimerkiksi View, joka vastaa div-elementtiä Reactin puolella. Taulukossa 1 on esitetty React ja React-Nativen elementtejä. (Opettaja. Oulun Ammattikorkeakoulu.)

Taulukko 1. React ja React-Native elementtejä

React	**React-Native**
	<View>

	<Text>
<img>	<Image>
	<FlatList>

Reactissa käytetään normaalisti kahdentyyppisiä komponentteja: säiliökomponentteja ja esityksellisiä komponentteja. Säiliökomponentit sisältävät logiikkaa, koostavat muita komponentteja toimiviksi kokonaisuuksiksi, mutta ne eivät ota kantaa ulkoasuun. Esitykselliset komponentit taas eivät sisällä logiikkaa ja keskittyvät vain esittämään käyttöliittymän osia. On fiksua eritellä nämä toisistaan koodin uudelleen käyttämisen helpottamiseksi. Esimerkki säiliökomponentista on esitetty kuviossa 7.

```
67      context.getImageData(0, 0, 100, 100).then(imageData => {
68        var data = Object.values(imageData.data);
69        var dataLength = Object.keys(data).length;
70
71        for(let i = 0; i < dataLength; i += 4){
72          var pixel = [data[i], data[i + 1], data[i + 2], data[i + 3]];
73          pixels.push(pixel);
74
75          data[i] = 1; data[i + 1] = 1; data[i + 2] = 1;
76        }
```

Kuvio 7. Säiliökomponentti

Esimerkki esityksellisestä komponentista löytyy kuviosta 8. Tyylimäärittelyihin viitataan myös esityksellisissä komponenteissa.

```
<View style={styles.footer}>
  <TouchableOpacity onPress={() => this.draw()} style = {styles.drawButton}>
    <Text style = {styles.drawButtonText}>Piirrä kuvio</Text>
  </TouchableOpacity>

  <TouchableOpacity onPress={() => this.play()} style = {styles.playButton}>
    <Text style = {styles.playButtonText}>Pelaa</Text>
  </TouchableOpacity>
</View>
```

Kuvio 8. Esityksellinen komponentti

4.6 Tyylit

Reactissa komponenttien tyylit määritellään CSS:llä, mutta React-Nativessa nämä määritellään hieman eri tavalla. React-Nativessa tyylit kirjoitetaan JavaScript-objekteina. Tyylien näkyvyys on rajattu komponentteihin. React-Nativessa ei tarvitse huolehtia selaimesta, jonka takia on järkevää yhdistää tyylit, merkintätapa sekä käyttäytyminen yhteen tiedostoon jokaiselle komponentille. Esimerkki tyylien määrittelystä on esitetty kuviossa 9.

```
const styles = StyleSheet.create({
  container: {
    backgroundColor: 'teal',
    flex: 1,
    padding: 16,
  },

  headerText: {
    textAlign: 'center',
    fontSize: 30,
    fontWeight: 'bold',
    padding: 10,
  },

  body: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    backgroundColor: 'powderblue',
    marginBottom: 120,
  },
```

Kuvio 9. Tyylien määrittely

4.7 Pelin toiminta

Ohjelma hakee pelialueen datan, eli pikselit ja käy jokaisen data-objektin yksi kerrallaan läpi. Jokainen pikseli on jaettu neljään osaan, jotka koostuvat väreistä; punainen, vihreä ja sininen sekä alpha. Eli sata kertaa sata pelialueen datan koko on tällöin 40 000 yksittäistä data-objektia. Esimerkki pikseleiden luonnista on esitetty kuviossa 10.

```
67      context.getImageData(0, 0, 100, 100).then(imageData => {
68        var data = Object.values(imageData.data);
69        var dataLength = Object.keys(data).length;
70
71        for(let i = 0; i < dataLength; i += 4){
72          var pixel = [data[i], data[i + 1], data[i + 2], data[i + 3]];
73          pixels.push(pixel);
74        }
75
76        shuffleArray(pixels);
77
78        for(let i = 0; i < 10; i++) {
79          winPixels.push(pixels[i]);
80        }
81      });
```

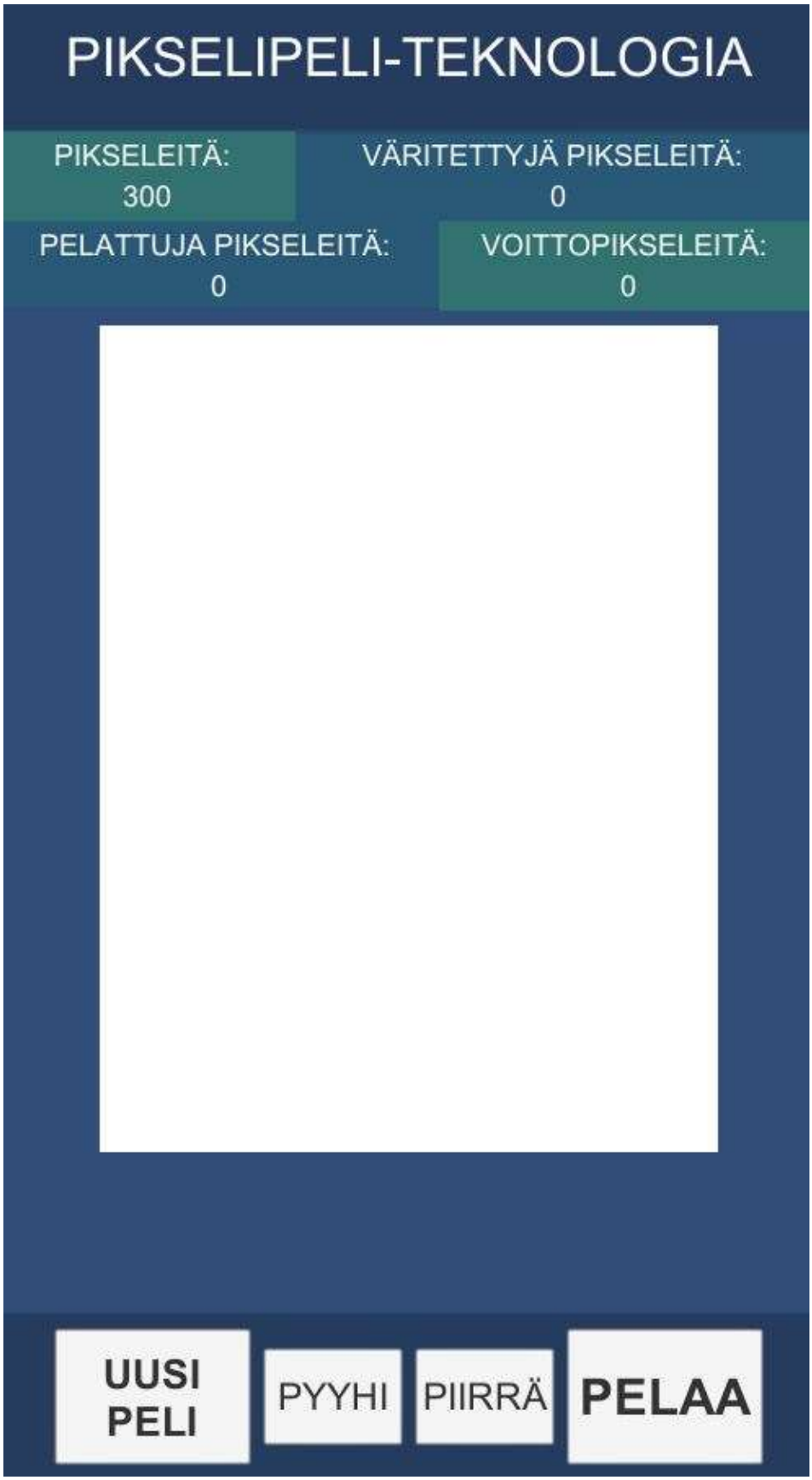
Kuvio 10. Pikselien luonti

Dataa käydään läpi neljä objektia kerralla (kolme väriä + alpha). Luodaan pikseli niminen muuttuja, johon talletetaan jokaisen pikselin väriarvot sekä alpha kanava, jonka jälkeen tämä pikseli talletetaan pikseleistä koostuvaan yhteiseen pikseliryhmään, ennen kuin hypätään seuraavaan neljään arvoon. Kun data on käyty läpi ja pikselit on tallennettu, kutsutaan ”shuffleArray” -nimistä funktiota, joka sekoittaa juuri tallettamamme pikselit uuteen järjestykseen. Lopuksi haetaan 10 ensimmäistä pikseliä tästä sekoitetusta pikseliryhmästä ja pusketaan nämä voittopikseli ryhmään. Kuviossa 11 on esitetty sekoittaja funktion toimintaa.

```
 8    function shuffleArray(array) {
 9      for (var i = array.length - 1; i > 0; i--) {
10          var j = Math.floor(Math.random() * (i + 1));
11          var temp = array[i];
12          array[i] = array[j];
13          array[j] = temp;
14      }
15    }
```

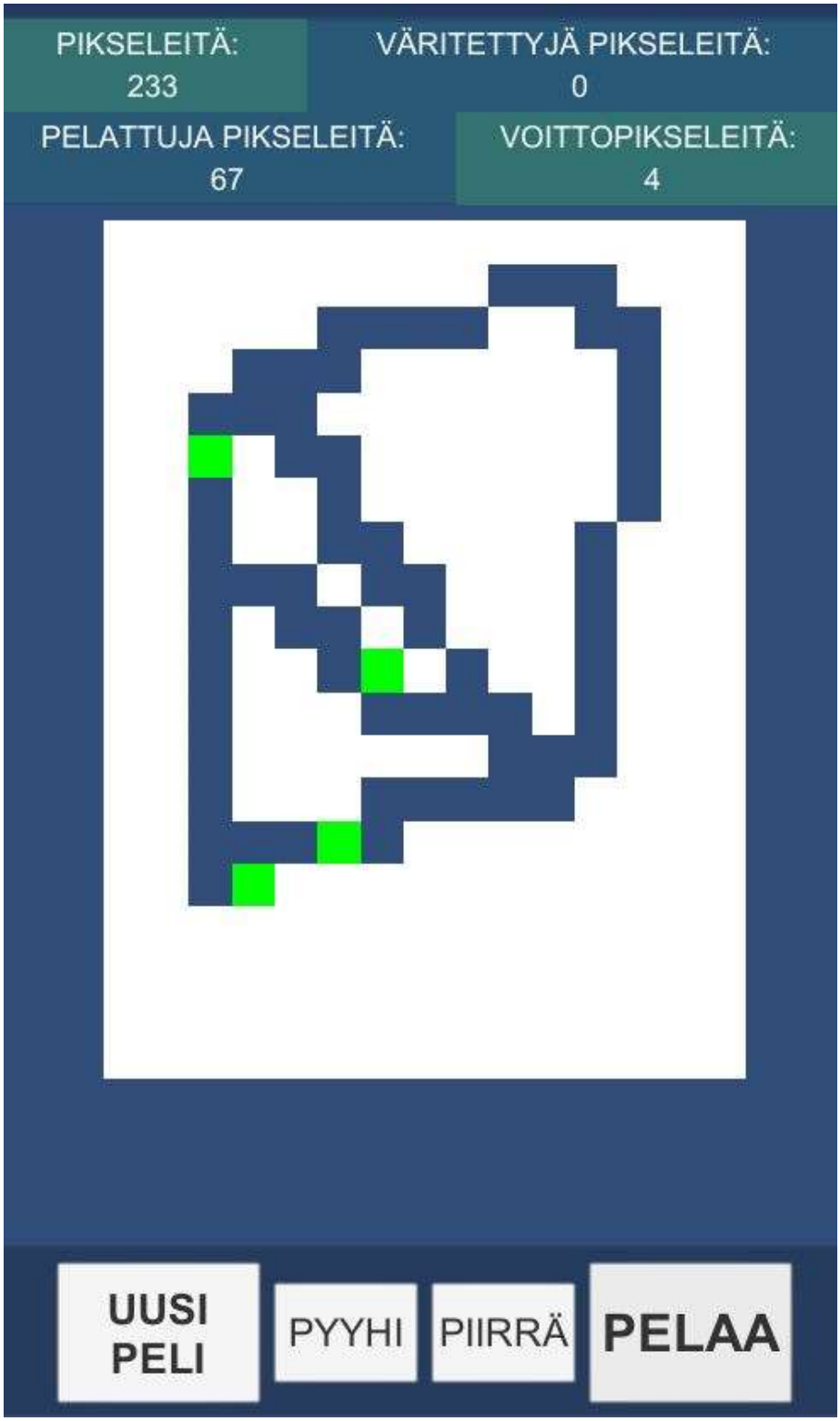
Kuvio 11. ”shuffleArray” -niminen funktio

Seuraavaksi kuviossa 12 on esitetty pelin ulkoasua. Pelialueelle on luotu 300 pikseliä, joista 10 kappaletta ovat niin sanottuja voittopikseleitä. Pelin ulkoasu voi näyttää hieman karulta, mutta opinnäytetyön tavoitteena ei ollutkaan graafinen puoli, vaan pelin toiminta.



Kuvio 12. Pelin aloitusnäyttö

Seuraavaksi kuviossa 13 esitetään voittopikseleiden löytämistä. Pikseleiden kokonaismäärä vähenee sitä mukaan, kun peli etenee, samalla lisäten pelattujen pikseleiden määrää.



Kuvio 13. Voittopikseleiden löytäminen

5 POHDINTA

5.1 Haasteet

Opinnäytetyön haastavimpana osuutena oli oman piirtotyökalun rakentaminen. Erilaisia NPM-piirtotyökalupaketteja löytyi kyllä internetistä ja niitä testattiin, mutta niiden muokkaaminen omien tarpeiden mukaan huomattiin liian monimutkaiseksi prosessiksi, joten lähdettiin luomaan uutta. Lisä haastetta työhön toi tietenkin aika, joka ei millään meinannut riittää, koska opinnäytetyö alkoi sen verran myöhään syksystä ja useiden uusien asioiden opetteluun sekä opinnäytetyön kirjoittamiseen meni oma aikansa. Eli ohjelmointia ei pystynyt harrastamaan niin paljoa kuin olisi haluttu, mutta loppujen lopuksi saatiin työstä valmista.

Yksi vaikea seikka oli rakentaa käyttöliittymä ilman aikaisempaa React-Native kokemusta, koska React-Native ei käytä tavallisia HTML- tai CSS- määrityksiä. Eli periaatteessa koko opinnäytetyö oli jatkuvaa uuden oppimista ja vanhan tiedon soveltamista.

5.2 Yhteenveto ja jatkokehitys

Yhteenvetona olen tyytyväinen työn lopputulokseen, vaikka työtä täytyikin karsia rankasti, sillä Pikselipeli-teknologiaa voidaan hyödyntää niin laajalti, mutta tässä opinnäytetyössä keskityin työstämään yhden toimivan piirtokomponentin. Tämä oli todella opettavainen matka sovelluskehitykseen, pääsin tutustumaan ensimmäistä kertaa kunnolla mobiilikehityksen maailmaan ja vahvistamaan osaamistani työn eri osa-alueilla. Projekti opetti myös itsenäiseen työskentelyyn sekä tiedonhakuun. Tästä on hyvä jatkaa eteenpäin.

Tämän opinnäytetyön aikana kehitetty prototyyppi Pikselipeli-teknologian toiminnasta kuvastaa vain yhtä osa-aluetta koko Pikselipeli-teknologian kirjosta, eli tässä tapauksessa piirto-ominaisuutta. Kun tässä opinnäytetyössä tehdyssä ohjelmassa hyödynnetään vain piirto-ominaisuutta, niin seuraaja tai kuka tahansa, joka haluaisi tai osaisi tehdä jatkotutkimusta tai kehitystä aiheesta, pystyy jatkamaan siitä mihin tämän työn puolesta ollaan jääty. Esimerkiksi tekstin, äänen ja kuvan pelaaminen hyödyntäen samaa Pikselipeli-teknologiaa. Ääniviestit voidaan esimerkiksi näyttää ruudulla ääniaaltoina ja näitä piirrettyjä aaltoja sitten hyödyntää samaa logiikkaa käyttäen, minne

on piirtynyt ja millä äänen korkeudella. Kuvan pelaaminen voi esimerkiksi toimia niin, että käyttäjä lataa ruudulle haluamansa kuvan ja sitä analysoidaan värien perusteella. Tai esimerkiksi kuva voisi latautua ensin näkymättömänä ruudulle, jonka jälkeen se alkaa hiljalleen paljastua pikseli kerrallaan. Logiikka ja laskeminen toimii periaatteessa kaikissa näissä samalla tavalla, joten ainoastaan käyttöliittymään tarvitsisi tehdä enimmät muutokset ja korjailut, kuten data inputit, eli mitä dataa kerätään ja mistä, kuinka se muutetaan muotoon, jota voidaan helposti käyttää olemassa olevissa logiikka funktioissa, jonka jälkeen tulokset voidaan tulostaa käyttäjälle näytettävään muotoon.

Jatkotoimenpiteinä Pikselipeli-teknologia komponenttia voidaan myös kehittää laajemmaksi kokonaisuudeksi mitä se nyt on. Sisältäen useampia erilaisia vaihtoehtoja muunnella informaatiota ja sitä kuinka paljon dataa voidaan hyödyntää. Sekä komponenttien, funktioiden ja metodien toimintaa tehostaa toimivuudeltaan eritavoin, mm. prosessi-intensiivisyyttä, eli kuinka monta laskentaa sekunnissa voidaan tehdä ilman suorituskyvyn laskua, sekä koodin optimointia edelleen lisäämällä. (AXO-Service Oy. Keskustelut. 2019.)

Kaikki tähän asti kerätty informaatio tämän opinnäytetyön puolesta, funktiot, metodit on kerätty talteen tuotteen omistajalle, eli toimeksiantajalle, jotta tulevaisuudessa kehitys voisi jatkua helposti siitä mihin se jäi.

LÄHTEET

Android docs. Avd-manager. Viitattu 1.12.2019.
http://www.androiddocs.com/tools/help/avd-manager.html.

AXO-Service Oy, Palaverit sekä viestit. Viitattu 29.11.2019.
www.axogames.com.

Developer. Math. Random.
developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math/random.

Developer. Mozilla. ImageData.
developer.mozilla.org/en-US/docs/Web/API/ImageData.

Expo. Examples and tutorials. Viitattu 28.11.2019.
https://blog.expo.io/examples-tutorials-e471ba902b1f.

Facebook Inc. 2019. React – A JavaScript library for building user interfaces. Viitattu 18.10.2019.
https://reactjs.org/.

Facebook Inc. 2019. React Native – Getting started. Viitattu 26.11.2019.
https://facebook.github.io/react-native/docs/getting-started.

GitHub. Terrylinla. React Native. Sketch-canvas.
github.com/terrylinla/react-native-sketch-canvas.

HTML5 canvas. Pixel Manipulation.
beej.us/blog/data/html5s-canvas-2-pixel/.

Material-UI. React components for faster and easier web development.
https://material-ui.com/.

MikaBug 2007. Ohjelmointiputka, opasarkisto: JavaScript. Viitattu 20.11.2019.

www.ohjelmointiputka.net/oppaat.

Opettaja. Oulun Ammattikorkeakoulu. Keskustelut. Viitattu 1.12.2019.

Pusher. Live paint. React.
pusher.com/tutorials/live-paint-react.

React-native-canvas. Npm. Kanvas-työkalu.
www.npmjs.com/package/react-native-canvas.

React Sketch. A sketch tool for React based applications.
www.npmjs.com/package/react-sketch.

YouTube 2017. React Native Tutorial for Beginners.
https://www.youtube.com/watch?v=6ZnfsJ6mM5c.

YouTube 2016. The Mathematics, Probability, Logarithm.
www.youtube.com/watch?v=bwKQPIyyVQw.

Visual Studio, Support. Viitattu 1.12.2019.
visualstudio.microsoft.com.

Visual Studio Code. Docs. Viitattu 25.11.2019.
https://code.visualstudio.com/docs.

W3Schools. HTML DOM Canvas Object.
www.w3schools.com/jsref/dom_obj_canvas.asp.