Väinö Kojo

Demosovellus Koru-sovelluskehyksestä älykellolle

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

1.3.2021

Tiivistelmä

Tekijä:	Väinö Kojo
Otsikko:	Demosovellus Koru-sovelluskehyksestä älykellolle
Sivumäärä:	34 sivua
Aika:	10.4.2022
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikka
Ammatillinen pääaine:	Ohjelmistotuotanto
Ohjaajat:	Lehtori Vesa Ollikainen

Teknologian kehittyessä ihmisen ja laitteen välinen tiedonvaihto kasvaa. Uusimpien älykellojen ominaisuudet ovat jo vertailtavissa älypuhelimiin. Soittaminen, viestien lähettäminen, musiikin kuuntelu ja säätilan tarkastaminen onnistuvat älykelloilla. Ranteeseen kiinnitettävät älykellot pystyvät myös mittaamaan käyttäjästä terveystietoja, kuten aktiivisuutta ja unenlaatua pitkien aikavälien ajalta. Älykellojen kehittyessä on tullut tarve nopeammalle ja tehokkaammalle ohjelmistopohjalle, joka tyydyttää niin käyttäjän kuin kehittäjän tarpeita.

Kannettavan laitteen käyttökokemuksen kannalta tärkeimpiä asioita ovat helppokäyttöisyys, selkeys, monipuolisuus ja käyttömukavuus. Koska älykelloissa käyttäjäkokemus perustuu enimmäkseen näytöllä tapahtuviin asioihin, on käyttöliittymän grafiikalla suuri merkitys hyvän käyttäjäkokemuksen saavuttamiseksi.

Tämän opinnäytetyön tarkoituksena on selvittää älykellojen sovelluskehityksen lähtökohtia sekä rajoituksia. Opinnäytetyössä laaditaan demosovellus, jossa yhdistetään web-sovelluskehityksen ja älykellojen sovelluskehityksen käytäntöjä. Ensin esitellään älykellojen teknisiä ominaisuuksia ja rajoitteita älykellojen sovelluskehityksessä. Seuraavaksi luodaan uusi älykellon demosovellus Blender-3D-grafiikkasovellusta sekä KoruLabin Koru-sovelluskehystä käyttäen. Lopuksi vertaillaan Koru-sovelluskehystä muihin vastaaviin sovelluskehyksiin.

Avainsanat: Älykello, Koru

Abstract

Author:	Väinö Kojo
Title:	Smartwatch Demo Application with Koru Software Framework
Number of Pages:	34 pages
Date:	10 April 2022
Degree:	Bachelor of Engineering
Degree Programme:	Information and Communication Technology
Professional Major:	Software Engineering
Supervisors:	Vesa Ollikainen, Senior Lecturer

As the technology advances information exchange between human and machine increases. The features in modern smartwatches are already comparable to those of smartphones. Making a call, writing a text message, listening to music or checking the weather can all be done with a smartwatch. Smartwatches as they are attached to the user's wrist, can also measure health information from the user e.g., activity and quality of sleep for long periods of time. As the smartwatches evolve, the need emerges for faster and more efficient software for users and developers.

The most important features for good user experience on all portable devices are ease of use, clarity, versatility, and comfort. Because most of the user experience on a smartwatch is based on what is happening on the screen, user interfaces graphics is one of the most important features on a device with a screen.

This thesis aims to examine software development based on smartwatch software and the restrictions the software development has. The study contains a demo application wherein the practices of web and smartwatch software development are used. First, the technical features and limitations of smart watches in the application development of smart watches are presented. Next, a new smart watch application created using the Blender-3D graphics application and KoruLab´s Koru software framework is introduced. Finally, the Koru-framework is compared to two equivalent software frameworks.

Keywords: Smartwatch, Koru

Sisällys

Lyhenteet

3D: *Three-dimensional*. Kolmiulotteinen.

AMOLED: *Active-Matrix Organic Light-Emitting Diode*. Näytön tyyppi, joka eroaa LCD-näytöstä siten, että yksittäisiä pikseleitä saadaan päälle ja pois päältä.

AOD: *Always On Display*. Toiminto, jossa vain osa näytöstä pidetään päällä rajoitetun datan näyttämiseksi.

API: *Application Programming Interface*. Ohjelmointirajapinta. Määritelmä ohjelmien väliseen tiedonvaihtoon.

CPU: *Central Prosessing Unit.* Laitteen prosessori ohjelmien ajamiseen.

GPS: *Global Positioning System*. Maailmanlaajuinen paikallistamisjärjestelmä, jossa paikannus toteutetaan satelliittien avulla.

GUI: *Graphical User Interface*. Graafinen käyttöliittymä, joka esittää grafiikaa näytöllä.

KoruCSS: *Koru Cascading Style Sheet*. KoruLab Oy:n kehittämä CSS-tyylisivuja jäljittelevä kieli sovellusten tyylien määrittelyyn.

KoruXML: *Koru Extensible Markup Language*. KoruLab Oy:n kehittämä XML-kieltä jäljittelevä merkintäkieli sovelluksen elementtien määrittelyyn.

LCD: *Liquid Crystal Display*. Yleinen näytön tyyppi, jossa taustavalo valaisee koko näytön.

MCU: *Micro Control Unit.* Kooltaan pieni tietokone, joka sisältää yhden tai useamman prosessorin (CPU).

1 Johdanto

Mikrokontrollerit (MCU) ovat pieniä tietokoneita, jotka sisältävät yhden tai useamman prosessorin (CPU), ohjelmoitavat tulo- sekä menoportit dataliikenteelle ja muistia. Mikrokontrollereita käyttävien laitteiden yleistyessä on noussut tarve tehokkaille, mutta energiapiheille sovellusratkaisuille. Mikrokontrollereita käyttävät laitteet ovat yleensä pieniä kuten älykellot. Laitteiden fyysinen koko rajoittaa sisälle mahtuvien komponenttien, kuten prosessorin, akun ja muistin kokoja. KoruLab Oy:n Koru on graafisen käyttöliittymän sovelluskehys (GUI-framework), joka on kehitetty vastamaan kysyntään [1]. KoruLab Oy on 2013 perustettu yritys, jonka päätoimiala on puettavat teknologiat (engl. Wearables). KoruLab Oy oli vuonna 2019 Deloitten tekemän tutkimuksen mukaan Euroopan 129. nopeimmin kasvava teknologiayritys [2]. Koru GUI:ta käyttäviä laitteita oli vuoden 2020 helmikuussa yli 10 miljoonaa maailmassa [3]. Graafisen käyttöliittymän luonti Korulla on helppoa, sillä rajapinnan funktiot on nimetty samoiksi kuin Web-kehityksessä käytetyt. Korussa graafisen käyttöliittymän GUI-komponentit esitellään KoruXML:llä ja tyylit KoruCSS:llä. Käyttöliittymän toiminta voidaan toteuttaa C- tai JavaScript-ohjelmointikielillä.

Insinöörityön tarkoituksena on tarkastella älykelloihin suunnatun sovelluskehityksen rajoitteita ja käytäntöjä sekä kehittää demopelisovellus KoruLabin Koru-alustalle, joka esittelee Korun mahdollisuuksia GUI:n kehityksessä. Pelisovelluksen pohjana käytetään 1980-luvulla kehitettyä Coinlinen suunnittelemaa Nopeustesti-peliä. Nopeustesti sai ensi esiintymisen Pertti ”Spede” Pasasen tv-viihdeohjelmassa Speden Spelit [4]. Pelistä on tarkoitus tehdä uudistettu avaruus-teemainen versio, jota pystyy pelaamaan pienellä älykellon kosketusnäytöllä. Ulkoasussa käytetään Blender-sovelluksella luotavia kuvia ja animaatioita sekä Nasan tekijänoikeuden suojaamatonta materiaalia.

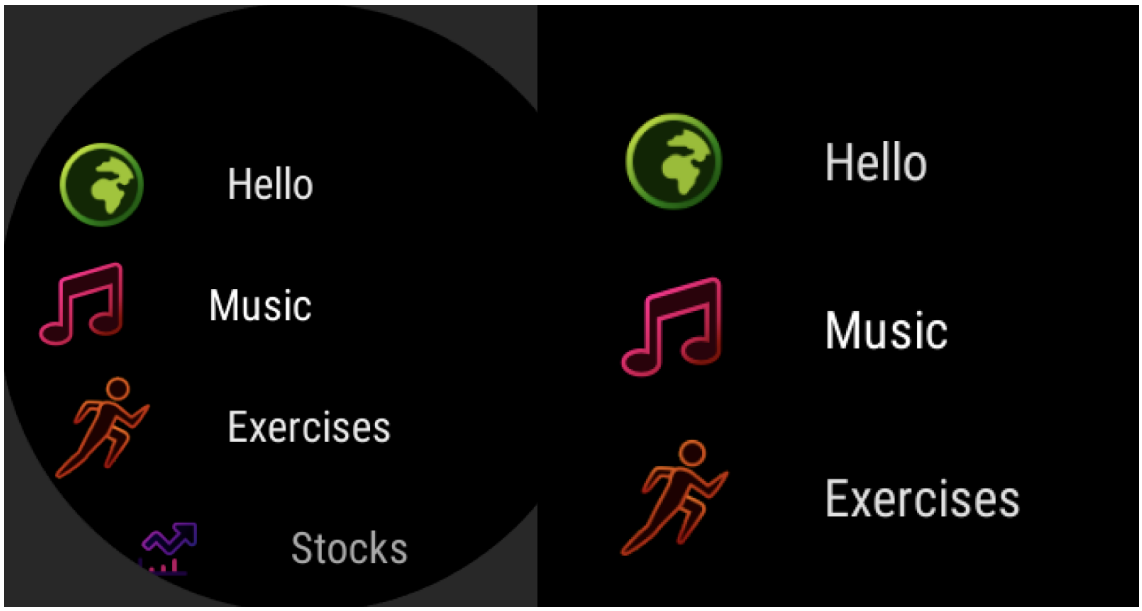
Luvussa 2 kerrotaan älykellojen teknisistä ominaisuuksista ja niiden tuomista rajoituksista sovelluskehityksessä. Luvut 3 ja 4 käsittelevät demosovelluksen

kehittämisen. Kolmannessa luvussa esitellään tässä opinnäytetyössä kehitetyn demosovelluksen rakenne ja neljännessä luvussa demo-sovellukselle lisätään toiminnallisuus. Luvussa 5 vertaillaan KoruLab Oy:n Koru-sovelluskehystä Googlen ja Applen vastaaviin sovelluskehyksiin.

2 Sovelluskehitys älykelloihin

Suurimmat komponentit älykelloissa ovat näyttö, akku ja prosessori. Komponenttien koot älykelloissa on rajattu älykellon kokoon. Älykelloissa ja muissa fyysisesti pienissä laitteissa käytettäviä prosessoreita (CPU) kutsutaan mikrokontrollereiksi (MCU). Mikrokontrollereita valmistavia yrityksiä ovat muun muassa Ambiq, ARM ja AMD. Akkujen kapasiteetit vaihtelevat paljon, kuten Xiaomi Watch S1 440 milliampeerituntia [5] ja Fitbit Sense 266 milliampeerituntia [6]. Harvat älykellojen valmistajat julkaisevat kellojensa tarkkoja teknisiä ominaisuuksia, joten kellojen tekninen vertailu on vaikeaa. Esimerkiksi Apple kertoo epämääräisesti, että seitsemännen sarjan Apple Watchin akku kestää 18 tuntia [7]. Monesti tekniset tiedot tulevat käyttäjien tietoisuuteen vasta, kun käyttäjä itse avaa laitteen ja julkaisee löydöksensä internetiin. Useat käyttöjärjestelmät, kuten Googlen Wear OS, Applen WatchOS ja Fitbitin Fitbit OS, sisältävät julkisen rajapinta käyttäjille kellotaulujen ja sovellusten kehittämiseen sekä julkaisuun.

Suurempi akun koko ja kapasiteetti ei aina tarkoita sitä, että akkua tarvitsisi ladata harvemmin. Näytön ja prosessorin tyypeillä sekä ohjelmistolla on suuri vaikutus akun kulutukseen. Optimaalisen akunkeston saavuttamiseksi kaikkien komponenttien ja ohjelmiston yhteistyö on välttämätöntä. Älykellojen käyttöjärjestelmiä ja sovelluksia kehitettäessä kehittäjän tulee ottaa huomioon älykellon tekniset ominaisuudet, kuten näytön koko ja muoto. Älykellon näyttö on yleensä 300–500 pikseliä leveä ja korkea. Näyttö voi olla ympyrän, ellipsin, neliön, nelikulmion tai ympyrän ja neliön yhdistelmän (engl. Squircle) muotoinen. Sovelluksen tulee tarvittaessa mukautua kaiken kokoisilla ja muotoisilla näytöille, kuten kuvassa 1.



Kuva 1. Valikko, ympyrän ja nelikulmion muotoisilla näytöillä esitettynä.

Älykellojen yleisimmät näytön tyypit ovat LCD ja AMOLED. LCD-näytössä taustavalo valaisee kaikki näkyvät pikselit ja mustan näyttämiseen pikseli peitetään optisilla filttereillä. LCD-näytön vahvuuksia ovat kirkas näkyvyys auringonvalossa ja halvemmat tuotantokustannukset. AMOLED-näytössä ei ole taustavaloa, vaan kaikilla pikseleillä on omat valonlähteensä ja pikseleitä saadaan tarvittaessa pois päältä. Sovelluksia suunnitellessa valaistujen pikseleiden määrä ja niiden värit merkitsevät, sillä valkoinen pikseli kuluttaa AMOLED-näytöllä enemmän virtaa kuin LCD-näytöllä, kun taas LCD-näytöllä musta pikseli kuluttaa enemmän virtaa verrattuna AMOLED-näyttöön. AMOLED-näytön vahvuuksia on vaaleiden ja tummien värisävyjen kontrasti sammuttamalla käyttämättömät pikselit. Mekaanisissa ja useimmissa digitaalisissa rannekelloissa kellotaulu näkyy koko ajan. AMOLED- ja LCD-näytöllisessä kellossa näytön pitää olla päällä, jotta kellotaulu näkyisi. Always On Display (AOD) on Nokian vuonna 2009 kehittämä ominaisuus, jonka avulla AMOLED-näytöllä pystytään näyttämään tietoja, kuten aika, päivämäärä ja erilaisia ilmoituksia laitteen ollessa lepotilassa [8]. Suurin osa näytöstä on AOD-tilassa pois päältä, mikä säästää akun kulutusta. Kellon virkistystaajuudella on myös merkitystä. Kellotaulu, joka näyttää päivämäärän, tunnit ja minuutit, ei tarvitse 60 kuvaa sekunnissa (FPS)

virkistystaajuutta, sillä aika päivittyy vain kerran minuutissa. Kellotaulun, joka näyttää esimerkiksi reaaliaikaisen sykkeen, tulee päivittyä kerran sekunnissa.

Älykellon ominaisuuksista riippuen älykellolla voidaan mitata monia erilaisia tietoja käyttäjästä. Älykellon kiihtyvyyssensorilla saadaan laskettua esimerkiksi päiväkohtaisia askelmääriä ja GPS-vastaanottimen avulla saadaan laskettua liikuttua matkaa korkeus- ja pituussuunnissa. Kellon kiinniolo ranteessa pitkiä aikoja antaa mahdollisuuden optisille sensoreille, joilla voidaan mitata käyttäjän ranteesta esimerkiksi sykettä, happisaturaatiota ja unenlaatua.

Koru-sovelluskehys poikkeaa sen kilpailijoista siten, että samaa sovelluskehystä voidaan käyttää kaikilla laitteilla ilman tarvetta muuttaa sovelluskehyksen rakennetta. Koru on alusta alkaen suunniteltu niin, että sama ohjelmisto toimisi mahdollisimman monella eri mikrokontrollerilla. Ominaisuudet on tehty modulaarisiksi, jotta asiakkaat pystyvät valitsemaan itse, mitä ominaisuuksia he haluavat sisällyttää ohjelmistoonsa. Modulaarisuuden hyötyjä on tehokkaampi ja nopeampi järjestelmä, pienempi muistin käyttö ja pienempi virran kulutus. Funktioiden toiminta ja nimeäminen on toteutettu vastaamaan JavaScriptin ja HTML:n käytänteitä. Koru-sovelluskehys sisältää paljon valmiita käyttöliittymäkomponentteja, kuten esimerkiksi pyyhkäisypainike, dynaaminen tekstikenttä ja vierityspalkki, jotka nopeuttavat kehittäjän kehitystyötä. Omien käyttöliittymäkomponenttien toteuttaminen on myös helppoa. Tiiviin ja hyvin rakennetun sovellusrakenteen ansiosta Koru tukee 60 kuvaa sekunnissa virkistystaajuutta Cortex M4 -pohjaisilla mikrokontrollereilla ja power-up screen-sleep -sykli, eli sovelluksen herääminen lepotilasta ja palautuminen takaisin lepotilaan kestää vain 20 millisekuntia [1].

Tässä työssä luodaan uusi pelisovellus KoruAPI:a [9] käyttäen Korun simulaattorille, mikä tulee toimimaan kaikilla ennalta määritetyillä näytöillä. Ensimmäisenä suunnitellaan ja toteutetaan KoruXML-komponentit ja asetellaan ne kohdilleen. Seuraavaksi pelin toiminnallisuus tehdään JavaScriptillä. Sovelluksen käynnistyessä näkyviin tulee aloitusnäyttö, joka sisältää tekstikentät pelin nimelle ja ohjeelle pelin aloittamiseen sekä painikkeet helpolle sekä vaikealle

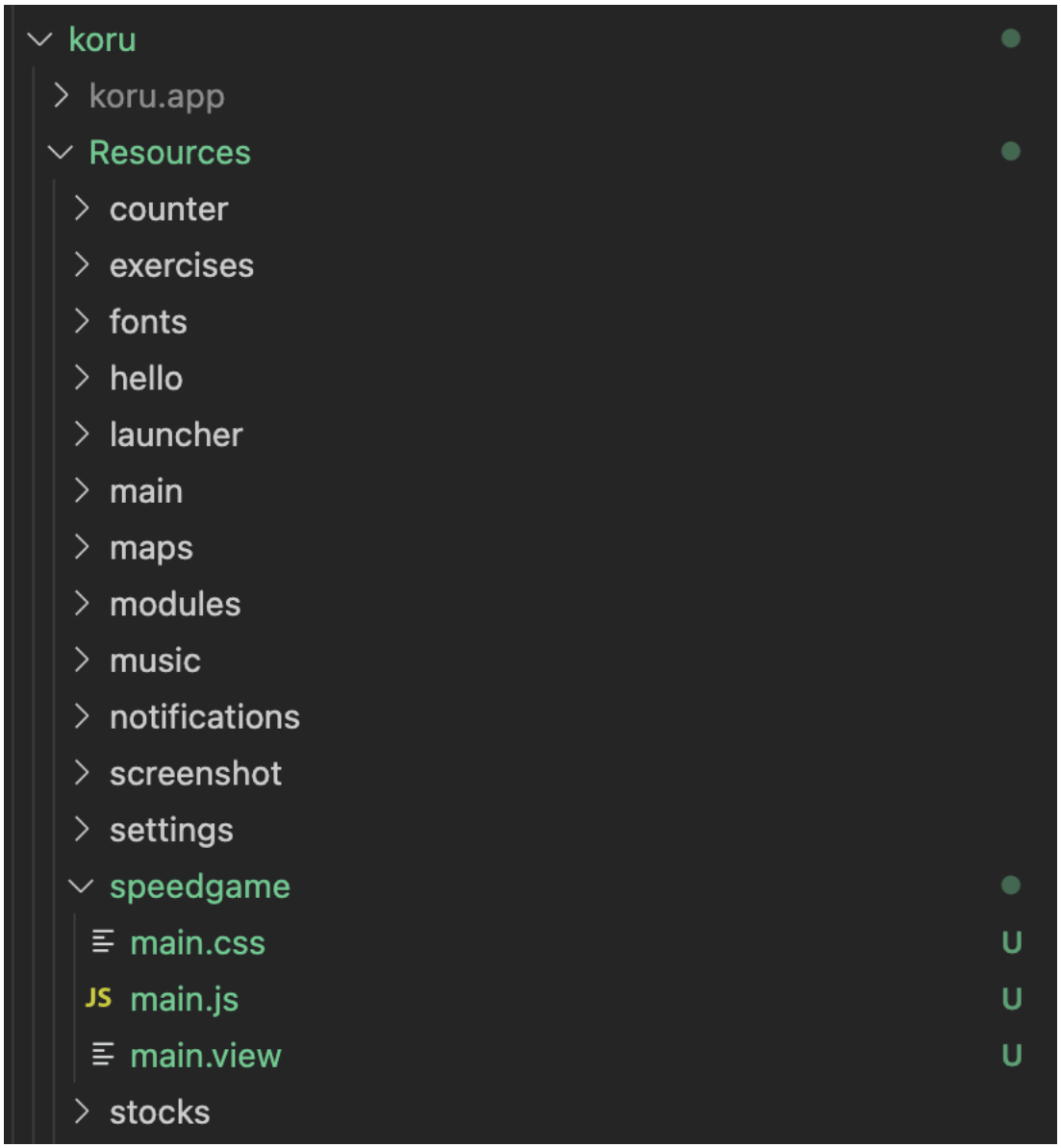
pelille. Edellisen pelin pisteet ja kaikkien aikojen parhaat pisteet tulevat näkyviin vasta pelin loputtua. Pelin tulee aktivoida painikkeita, joita käyttäjän täytyy painaa tietyn ajan kuluessa, ja jos painiketta ei paineta ajoissa, pelin tulee loppua. Peli nopeutuu jokaisen kolmen onnistuneen painalluksen jälkeen.

3 Koru-sovelluksen rakenne

Tässä luvussa kuvataan uuden Koru-sovelluksen rakenteen kehittäminen. Sovellukselle luodaan KoruXML-komponentit ja tyylit asetetaan KoruCSS-kielellä. Tuloksena saadaan sovelluksen pohja, jolla ei vielä ole toiminnallisuutta. Sovelluksen toiminnallisuus lisätään luvussa 4.

3.1 Koru-sovelluksen tiedostorakenne

Koru sisältää simulaattorin, jolla sovelluksia voidaan ajaa erikokoisilla ja -muotoisilla näytöillä. Simulaattorilla voidaan testata sovelluksia ennen, kun ne asennetaan älykelloon. Koru-sovelluksen kehitys aloitetaan luomalla uusi kansio simulaattorin Resources-kansioon (kuva 2). Uuden kansion nimeksi asetetaan uuden sovelluksen nimi: speedgame. Tässä työssä kehitettävä sovellus käyttää JavaScript-, KoruXML- ja KoruCSS-kieliä, joten luotuun kansioon lisätään tiedostot main.js, main.view ja main.css.



Kuva 2. Koru-simulaattorin tiedostorakenne.

Korun demosovellusta ohjataan C-ohjelmointikielen tiedostossa launcher_app.c. Kaikki oletussovellukset lisätään staticApplications-nimiseen listaan, jotta ne ovat laitteen käynnistyessä valmiiksi asennettuina. Kuvassa 3 speedgame-sovellus on lisätty muiden sovellusten joukkoon.

```
staticApplications[] =
{
    { "speedgame",  APP_COLOR_DEFAULT },
    { "hello",      APP_COLOR_HELLO },
    { "music",      APP_COLOR_MUSIC },
    { "exercises",  APP_COLOR_EXERCISES },
    { "stocks",     APP_COLOR_STOCKS },
    { "maps",       APP_COLOR_MAPS },
}
```

Kuva 3. Speedgame-sovellus lisättynä staattisten sovellusten listaan.

Koru-simulaattorissa myös sovellusten jälkiasennus eli ajon aikainen asennus on mahdollista. Tällöin sovellusta ei tarvitse lisätä staattisten sovellusten listaan, vaan se asennetaan, kun Koru-simulaattori on käynnistynyt. Jälkiasennuksen ongelmana on pienen prosessorin muistin rajoitukset. Sovelluksen sisältäessä paljon kuvia sovelluksen koko saattaa nousta suuremmaksi, kuin prosessori pystyy käsittelemään yhdellä kerralla. Tällöin sovellus joudutaan asentamaan osissa, mikä ei ole optimaalista ja vie enemmän aikaa. Sovellusta kehitettäessä onkin tärkeää, että kuvat pyritään tiivistämään tiedostokooltaan mahdollisimman pieniksi siten, että ne säilyttävät silti mahdollisimman hyvän kuvanlaadun.

3.2 KoruXML

KoruXML-kieltä käytetään näkymien (engl. View), käyttöliittymäkomponenttien määrityksien (engl. Definitions) ja tuontien (engl. Import) määrittelyihin. KoruXML-tiedostotyypit ovat view, def ja import. KoruXML-tiedostossa käyttöliittymäkomponentteja luodaan symbol-elementillä. Lisäämällä käyttöliittymäkomponenttien sisälle muita komponentteja saadaan rakennettua monipuolisia sovelluksia. Speed Game -pelisovellus koostuu kahdesta pääkomponentista: aloitusnäytöstä StartView ja pelinäytöstä GameView.

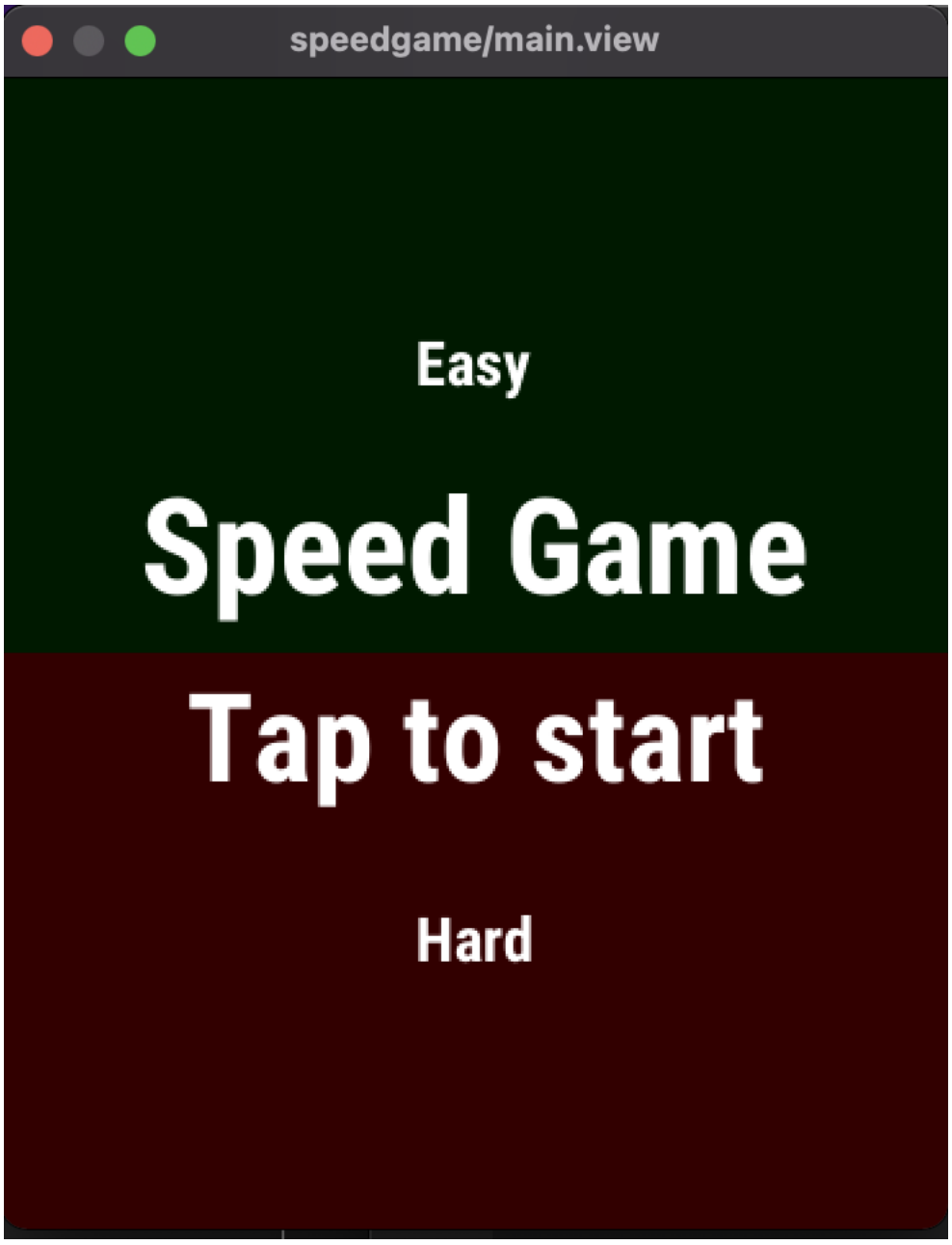
3.2.1 StartView

Aloitusnäyttö sisältää sovelluksen nimen, ohjeistuksen pelin aloittamiseen, tekstielementit pisteiden näyttämiseen ja painikkeet kahdelle vaikeustasolle: helppo ja vaikea. Esimerkkikoodi 1 esittelee StartView-komponentin määrityksen. StartView-symboli sisältää StartViewTexts-symbolin, joka sisältää neljä tekstielementtiä: sovelluksen nimen, pelin käynnistämisohjeen, viimeisen pelin pisteet ja kaikkien aikojen parhaat pisteet. Viimeisen pelin pisteet ja kaikkien aikojen parhaat pisteet on asetettu StaticText-komponentiksi, jotta niiden arvoja, kuten tekstin pituutta, voidaan muuttaa ajon aikana. StaticText-elementit jätetään aluksi tyhjiksi. StaticText-komponentti laskee tekstin tarvitseman muistin määrän automaattisesti ja varaa sen kasamuistista (engl. Heap memory).

```
<symbol id='StartViewTexts'>
     <text id='game-title-text' text-buffer='_Speed Game'/>
     <text id='start-view-text' text-buffer='_Tap to start'/>
     <StaticText id='final-score-text'>
          <set href='text' to=''/>
     </StaticText>
     <StaticText id='high-score-text>
          <set href='text' to=''/>
     </StaticText>
</symbol>

<symbol id='StartView'>
     <Button id='easy-button/>
     <Button id='hard-button/>
     <StartViewTexts id='application-texts'/>
</symbol>
```

Esimerkkikoodi 1. StartView-komponentin alustava määrittely.

Aloitusnäyttö StartView on valmis, mutta tekstien ja painikkeiden nollapisteet ovat sijoittuneet oletuspaikalle x ja y-arvoilla 0, kuvaruudun vasempaan yläkulmaan. Teksteille asetetaan luokat (engl. Class) h2 ja middle, mikä siirtää tekstit kuvaruudun keskelle ja asettaa toisen tason otsikon HTML-tyylin. Painikkeille asetetaan väliaikaiset sijainnit ja taustavärit vihreä ja punainen, jotta ne näkyisivät ruudulla (kuva 4).



Kuva 4. StartView-näkymän ulkoasu.

Aloitusnäytön ulkoasu on pääpiirteittäin valmis, mutta se tulee vielä muuttumaan, kun taustakuva ja animoidut painikkeet lisätään. Seuraavassa aliluvussa luodaan sovelluksen pelinäkymä.

3.2.2 GameView

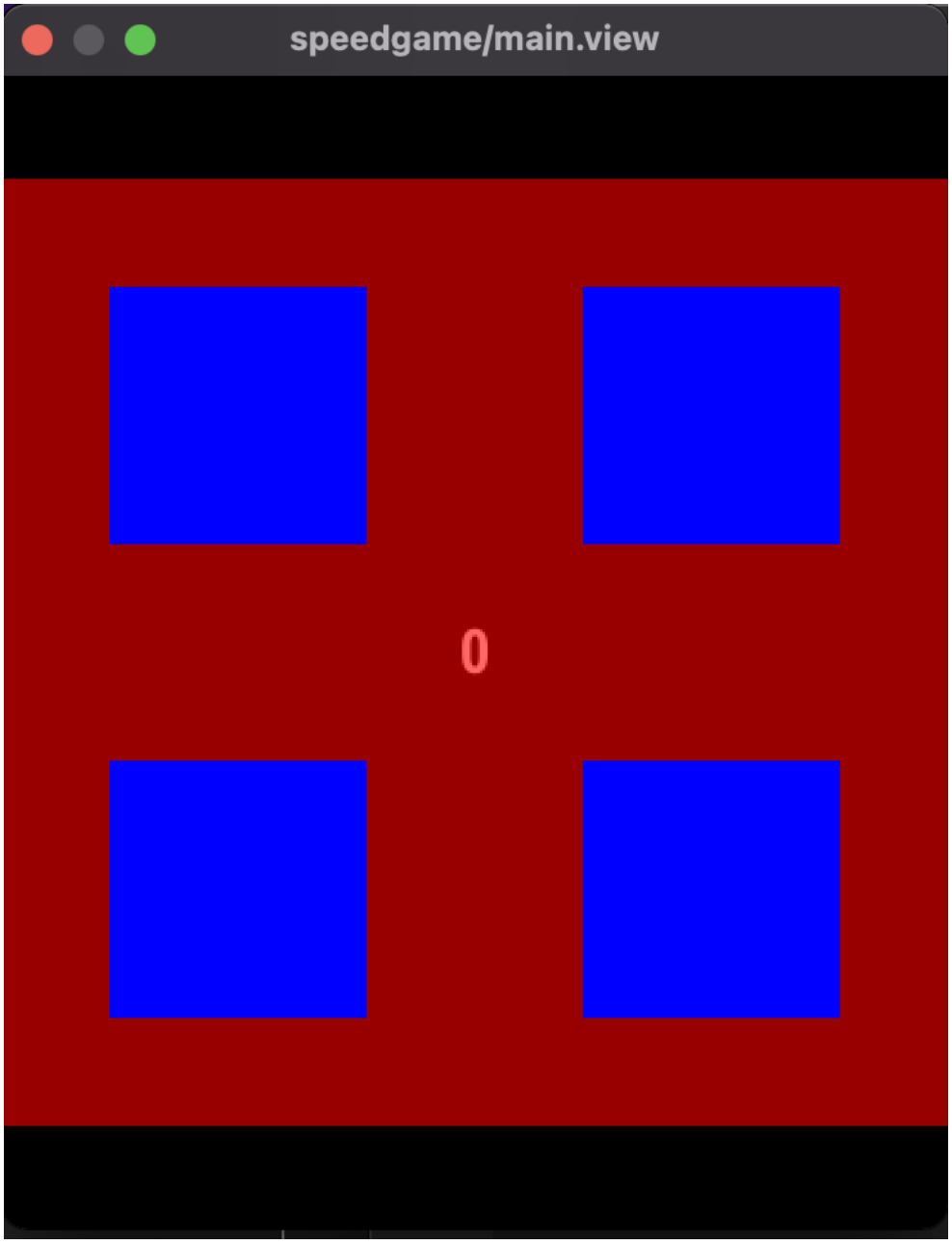
Sovelluksen pelinäkymä koostuu tekstielementistä pisteiden näyttämiseksi sekä säiliöstä, jonka sisälle lisätään pelin neljä painiketta (esimerkkikoodi 2). Painike-elementtien sisälle lisätään image-elementit.

```
<symbol id=’GameButton’ type=’push-button’ pointer-events=’all’>
     <animate id=’animate-to’ attributeName=’opacity’/>
     <animate id=’animate-from’ attributeName=’opacity’/>

<symbol id=’GameControls’>
     <g id=’buttons-container’>
          <animateTransform id=’animate-buttons-rotate’
                            attributeName=’rotate’/>
          <GameButton id’red-button’>
               <image id=’button-image’/>
          </GameButton>
          <GameButton id’blue-button’>
               <image id=’button-image’/>
          </GameButton>
          <GameButton id’green-button’>
               <image id=’button-image’/>
          </GameButton>
          <GameButton id’yellow-button’>
               <image id=’button-image’/>
          </GameButton>
     </g>
</symbol>

<symbol id=’GameView’>
     <StaticText id=’score-count-text’>
          <set href=’text’ attributeName=’text-buffer’ to=’0’/>
     </StaticText>
     <GameControls id=’game-controls’/>
</symbol>
```

Esimerkkikoodi 2. GameView-komponentin määritykset.

Painikkeet lisätään säiliön sisälle, jotta ne pysyvät tasaisin välimatkoin toisiinsa nähden kuvaruudun kasvaessa korkeus- tai pituussuunnassa. Säiliön nollapiste asetetaan säiliön keskipisteeseen lisäämällä elementin anchor-attribuutille arvo middle. Kun säiliön keskipiste siirretään näytön keskipisteeseen, painikkeet pysyvät oikeilla kohdillaan kuvaruutuun nähden. Säiliön korkeus ja leveys asetetaan näytön korkeuden tai leveyden mukaiseksi sen mukaan, kumpi niistä on pienempi. Pienempää arvoa käyttämällä säiliöstä tulee neliö, joka mahtuu näytölle jokaisella mahdollisella näytön koolla. Arvo asetetaan JavaScript-

ohjelmointikielellä luvussa 4. Kuvassa 5 säiliölle on asetettu punainen taustaväri havainnollistamaan säiliön asettelua.



Kuva 5. Säiliö mukautuu kaiken kokoisille ja muotoisille näytöille.

Helpolla vaikeustasolla painikkeet pysyvät paikallaan. Vaikeaa vaikeustasoa varten säiliöön lisätään animateTransform-elementti pyörittämään säiliötä, kun vaikea vaikeustaso aktivoidaan.

3.3 KoruCSS

Sovelluksen käyttöliittymäkomponenttien tyylit, kuten sijainnit ja koot, asetetaan KoruCSS-kielellä main.css-tiedostoon. KoruCSS on KoruXML:lle tehty CSS-kieleen pohjautuva tyylikieli, jossa tyylejä voidaan antaa myös omille symbol-elementeille. Esimerkkikoodissa 3 tyylit on asetettu StartViewTexts-symbolin lapsielementeille main.css-tiedostossa.

```
text { fill: white; font-weight: bold; text-anchor: middle; }

StartViewTexts #game-title-text  { y: 45%; font-size: 52; }
StartViewTexts #high-score-text  { y: 5%; font-size: 12; }
StartViewTexts #final-score-text { y: -10%; font-size: 42; }
```

Esimerkkikoodi 3. StartViewText-komponentin tyylit.

CSS-tyyleissä pyritään käyttämään paljon prosenttiarvoja, jotta sovellus saadaan toimimaan mahdollisimman monilla erikokoisilla ja -muotoisilla näytöillä. Esimerkiksi, jos 200 pikseliä leveän näytön 50 pikseliä leveälle nelikulmioelementille asetettaan x-attribuutin arvo 75, elementin sijainti tulee näytön x-akselin mukaisesti keskelle. Mutta jos näyttö olisikin 400 pikseliä leveä, elementin sijainniksi tulisi näytön vasen laita. Elementti saadaan näytön keskelle asettamalla x-attribuutille arvo 50 % ja vähentämällä siitä elementin leveys jaettuna kahdella, koska elementin nollapiste sijaitsee elementin vasemmassa yläkulmassa. Esimerkkikoodissa 4 on esitelty painikkeiden sijainnit säiliöelementin sisällä.

```
GameButton { width: 200; height: 200; }

GameControls #button-container #red-button    { x: -25%-100; y: -25%-
100; }
GameControls #button-container #blue-button   { x: 25%-100; y: -25%-
100; }
GameControls #button-container #green-button  { x: -25%-100; y: 25%-
100; }
GameControls #button-container #yellow-button { x: 25%-100; y: 25%-
100; }
```

Esimerkkikoodi 4. Painike-elementtien sijainnit säiliöelementin sisällä. 200 pikseliä leveän ja korkean kuvan keskipiste saadaan liikuttamalla kuvaa x- ja y-akselin suunnassa -100 pikseliä.

KoruCSS tukee myös CSS-tyylisivujen media-attribuuttia, jolla tyylit voidaan antaa esimerkiksi lokalisaation ja näytön koon perusteella. Hyötynä on esimerkiksi se, että käytettävien kuvien href-attribuutteilla voidaan määritellä oikean kokoiset kuvat erikokoisille näytöille eikä kuvia tarvitse skaalata, mikä huonontaisi kuvan laatua. Lokalisoinnin media-attribuuttia voidaan käyttää esimerkiksi arabian kielelle muuttamaan tekstien kirjoitustyylejä, koska arabian kieltä luetaan oikealta vasemmalle.

3.4 JavaScript

Koru-sovelluksessa sovellus vastaanottaa tapahtumia (engl. Event), joita voidaan käsitellä JavaScript-tiedostossa. Sovelluksen käynnistyessä pääohjelma lähettää sovellukselle load-tapahtuman. Speed Game -sovelluksen vastaanottaessa load-tapahtuman kutsutaan seuraavassa luvussa implementoitua handleLoad-metodia, joka alustaa kaikki käytettävät elementit. Rajatun muistinmäärän takia vain elementit, joita käytetään useassa metodissa, alustetaan julkisiksi muuttujiksi. Pienimmän Korua tukevan mikrokontrollerin (MCU) pinossa (engl. Stack) riittää tilaa noin kolmellekymmenelle elementti-kahvalle. Seuraavassa luvussa Speed Game -sovelluksen toiminta määritellään main.js-tiedostoon.

4 Sovelluksen toiminnallisuus ja viimeistely

Koru-sovelluksen toiminnallisuus voidaan toteuttaa C- tai JavaScript-ohjelmointikielillä. Tässä luvussa aiemmin määritellylle Speed Game -sovellukselle lisätään toiminnallisuus JavaScript-ohjelmointikielellä sekä luodaan pelin kuusi painiketta Blender-3D-grafiikkaohjelmalla ja viimeistellään sovelluksen ulkoasu.

4.1 Perusominaisuudet

Sovelluksen toiminnallisuuden luomisessa hyödynnetään KoruAPI:a [8]. Sovelluksen toiminnallisuus lisätään simulaattorin Resources/speedgame-kansion main.js-tiedostoon.

HandleLoad-metodin (esimerkkikoodi 5) vastaanottaessa load-tapahtuman kaikki myöhemmin käytettävät elementit alustetaan julkisiin muuttujiin ja sovelluksen painikkeille asetetaan tapahtumakuuntelijat. Aiemmin esimerkkikoodi 2:ssa esitellylle painikkeet sisältävälle säiliölle asetetaan leveys ja korkeus. Näytön leveydestä ja korkeudesta valitaan pienempi arvo ja se asetetaan painikkeet sisältävän säiliön leveydeksi ja korkeudeksi, mikä tekee säiliöstä ruudulle sopivan kokoisen.

```
function handleLoad(view, widget, time, event, x, y, z) {

     if ( (event == load || event == reload) && z == 2 ) {
          mainView   = view;
          startView = view.getElementById("start-view");
          gameView  = view.getElementById("game-view");

          // Set button-container width and height.

          const container = gameView.getElementById("game-controls");
          const size = view.width > view.height ? view.height :
     view.width;
          container.setAttribute("width", size);
          container.setAttribute("height", size);

          // Get text elements.

          scoreCountText = gameView.getElementById("score-count-text");
          finalScoreText = startView.getElementById("final-score-
     text");
          highScoreText  = startView.getElementById("high-score-text");
          startViewText  = startView.getElementById("stat-view-text");
          gameTitleText  = startView.getElementById("game-title-text");

          // Get button elements and set event listeners.

          colors[0].button = gameView.getElementById("red-button");
          colors[1].button = gameView.getElementById("blue-button");
          colors[2].button = gameView.getElementById("green-button");
          colors[3].button = gameView.getElementById("yellow-button");
          colors[0].button.setEventListener("click", handleScore);
          colors[1].button.setEventListener("click", handleScore);
          colors[2].button.setEventListener("click", handleScore);
          colors[3].button.setEventListener("click", handleScore);

          // Get background animate rotate element.

          animateButtonsRotate = gameView.getElementById("animate-but-
     tons-rotate");

     }
}
```

Esimerkkikoodi 5. handleLoad-metodi, jossa alustetaan näkymät, tekstielementit ja painikkeet.

Seuraavaksi luodaan metodi handleStart, joka ohjaa pelin kahta näkymää StartView ja GameView. HandleStart-metodi hyödyntää julkista gameState-muuttujaa, jonka arvot voivat olla true tai false. Kun gameState-muuttujan arvo on false, handleStart-metodi asettaa GameView-elementin display-atribuutin arvoksi none, startView-elementin display-arvoksi inline, näyttää edellisen pelin pisteet ja ennätyspisteet. Kun handleStart-metodia kutsutaan uudestaan, gameState-muuttujan arvo vaihtuu todeksi ja interval-muuttujan arvoksi asetetaan STARTINGINTERVAL, eli 3000 millisekuntia, startView-elementti piilotetaan, gameView-elementti asetetaan näkyväksi, pistelaskuri nollataan ja sovellukselle asetetaan intervalli setInterval-metodilla. Intervalli kutsuu handleGame-metodia interval-muuttujassa määritellyn ajan välein. Esimerkkikoodissa 6 on esitelty handleStart-metodi. HandleStart-metodia kutsutaan painamalla helpon tai vaikean vaikeustason painiketta ja automaattisesti pelin loputtua.

```
function handleStart(id) {
     gameState = !gameState;

     // Show gameView and start the game.

     if ( gameState ){
          interval = STARTINGINTERVAL;
          scoreCount = 0;
          scoreCountText.setText(scoreCount);
          startView.setAttribute(“display”, “discard”);
          gameView.setAttribute(“display”, “inline”);
          startViewText.setAttribute(“display”, “discard”);
          gameTitleText.setAttribute(“display, “discard”);

          if ( id == “hard-button” )
               animateButtonsRotate.animate(“custom”, true);

          timer = mainView.setInterval(handleGame, interval);
     }

     // Show startView and score.

     if ( !gameState ) {
          gameView.setAttribute(“display”, “discard”);
          startView.setAttribute(“display”, “inline”);
          finalScoreText.setText(“Score: ” + scoreCount);

          if ( scoreCount > highScore )
               highScore = scoreCount;
          highScoreText.setText(“High score: ” + highScore);
     }
}
```

Esimerkkikoodi 6. handleStart metodi, joka näyttää peli- ja aloitusnäkymän gameState-muuttujan arvon mukaan.

HandleGame-metodi ohjaa pelin kulkua (esimerkkikoodi 7). Ensimmäisenä funktiossa tarkastetaan, että onko peli hävitty kutsumalla checkLose-metodia. Jos checkLose-metodi palauttaa tosi, peli on hävitty.

```
function handleGame(id) {


     // Check if no active button is pressed in time.

     if ( checkLose() ) {
          reset();
          handleStart();
          return;
     }

     // Speed up the game on every third point.

     if ( scoreCount % 3 == 0 && scoreCount != 0 ) {
          mainView.clearInterval(timer);
          progressDifficulty();
     }

     // Set random button (0-3) state and start animation.

     var index = Math.floor(Math.random() * 4);
     activateButton(index);
}
```

Esimerkkikoodi 7. handleGame-metodi ohjaa pelin kulkua.

Metodi checkLose (esimerkkikoodi 8) tarkastaa kaikkien neljän painikkeen tilat, ja jos yksikin on aktiivinen, metodi palauttaa tosi. Jos aktivoitua painiketta on painettu interval-muuttujaan määritellyn ajan sisällä, checkLose-metodi palauttaa epätosi ja peli jatkuu.

```
function checkLose() {

     // If buttons state is true = flashing, lose the game.

     for ( var s in colors ) {
          if ( colors[s].state == true ) {
               return true;
          }
     }

     return false;
}
```

Esimerkkikoodi 8. checkLose-metodi tarkistaa, onko jokin painikkeista aktiivinen.

Seuraavaksi handleGame-metodissa tarkistetaan pistemäärä, ja sen ollessa kolmella jaollinen, mutta ei nolla, nopeutta nostetaan kutsumalla

progressDifficulty-metodia. Esimerkkikoodissa 9 esitelty progressDifficulty-metodi laskee interval-muuttujan arvoa jakamalla sen 1,1:llä ja aloittaa uuden ajastimen, jolloin handleGame-metodia kutsutaan nopeammissa sykleissä ja painikkeet aktivoitumat nopeammin.

```
function progressDifficulty() {
     interval = Math.floor(interval / 1.1);
     timer = mainView.setInterval(handleGame, interval);
}
```

Esimerkkikoodi 9. progressDifficulty-metodi laskee interval-muuttujan arvoa ja asettaa uuden ajastimen.

Viimeiseksi handleGame-metodissa (esimerkkikoodi 10) arvotaan luku nollan ja kolmen väliltä. Painike, jonka indeksi on arvottu luku, asetetaan välkkymään kutsumalla activateButton-metodia arvottu luku parametrina. ActivateButton-metodi asettaa painikkeen tilaksi tosi ja käynnistää animaatiot (ks. Esimerkkikoodi 2). Seuraavan interval-muuttujaan määritellyn arvon jälkeen handleGame-metodia kutsutaan uudestaan ja nyt yhden painikkeen tila on tosi.

```
function activateButton(index) {
     colors[index].state = true;
     var nodes = colors[index].button.childNodes;
     nodes[0].animate(“enable”, true);
     nodes[1].animate(“enable”, true);
}
```

Esimerkkikoodi 10. activateButton-metodi asettaa painikkeen tilan ja käynnistää animaatiot.

Esimerkkikoodissa 2 määriteltyjä GameButton-painike-elementtejä painettaessa kutsutaan handleScore-metodia ja parametrina lähetetään event-tapahtuma. HandleScore-metodissa (esimerkkikoodi 11) parametrina tulleesta event-oliosta saadaan napsautetun painike-elementin id-tunnus. Id-tunnuksen perusteella tarkistetaan, onko napsautettu painike-elementti aktiivinen. Jos painike-elementti on aktiivinen, pistelaskurin pistemäärää nostetaan yhdellä, painikkeen tila asetetaan epätodeksi ja kutsutaan resetButton-metodia, joka lopettaa aiemmin käynnistetyt välkkymisanimaatiot.

```
function handleScore(event) {

     // Check if the pressed button is activated and count score.

     var target = event.target.id;

     if ( target == "red-button && colors[0].state == true ) {
          scoreCount.setText(++scoreCount);
          resetButton(1);
     }
     else if ( target == "blue-button && colors[1].state == true ) {
          scoreCount.setText(++scoreCount);
          resetButton(2);
     }
     else if ( target == "green-button && colors[2].state == true ) {
          scoreCount.setText(++scoreCount);
          resetButton(3);
     }
     else if ( target == "yellow-button && colors[3].state == true ) {
          scoreCount.setText(++scoreCount);
          resetButton(4);
     }
}
```

Esimerkkikoodi 11. handleScore-metodi tarkistaa, onko painettu painike aktivoitu, ja laskee pisteet.

Aktivoidun painikkeen välkyntäanimaation tulee päättyä, kun painiketta painetaan. ResetButton-metodi (esimerkkikoodi 12) alustaa valitun painike-elementin tilan ja animaatiot. Jos parametria ei anneta, metodi alustaa kaikkien painikkeiden tilat ja animaatiot.

```
function resetButton(index) {

     // Reset button state and stop the flashing animation of the but-
ton by index.

     if ( index ) {
          colors[index-1].state = false;
          var nodes = colors[index-1].button.childNodes;
          nodes[0].animate(“disable”, false);
          nodes[1].animate(“disable”, false);
          return;
     }

     // Reset states and animations for all buttons if no index is
given.

     for ( var id in colors ) {
          colors[id].state = false;
          var nodes = colors[id].button.childNodes;
          nodes[0].animate(“disable”, false);
          nodes[1].animate(“disable”, false);
     }
}
```

Esimerkkikoodi 12. resetButton-metodi.

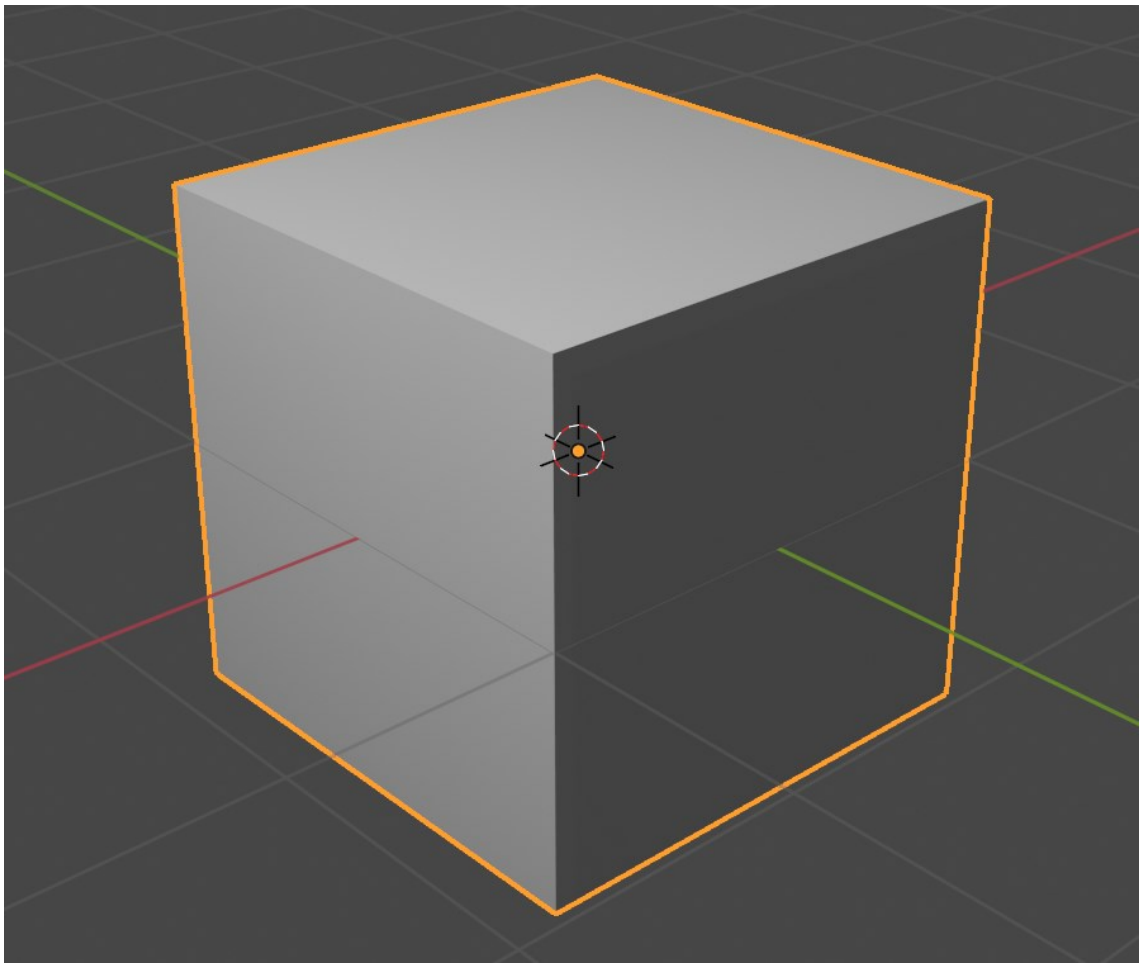
Pelisovelluksen runko ja toiminta on valmis. Sovelluksen käynnistyessä käyttäjä voi aloittaa uuden pelin koskettamalla helpon tai vaikean pelin painiketta. Peli aktivoi painikkeita nopeutuvaan tahtiin ja tarkastaa, että painiketta on painettu ajoissa.

4.2 Blender

Sovelluksen rakenne ja toiminnallisuus ovat valmiit ja paikkamerkkeinä käytetyt nelikulmiot vaihdetaan Blenderillä luotuihin 3D-kuviin. Blender on ilmainen, avoimenlähdekoodin 3D-grafiikkasovellus 3D-kuvien, -mallien ja -animaatioiden tekoon [10]. Tässä työssä Blenderillä luodaan 3D-kuvat ja -animaatiot sovelluksessa käytettäville painikkeille. Painikkeista tehdään avaruus-teeman mukaisesti planeettoja.

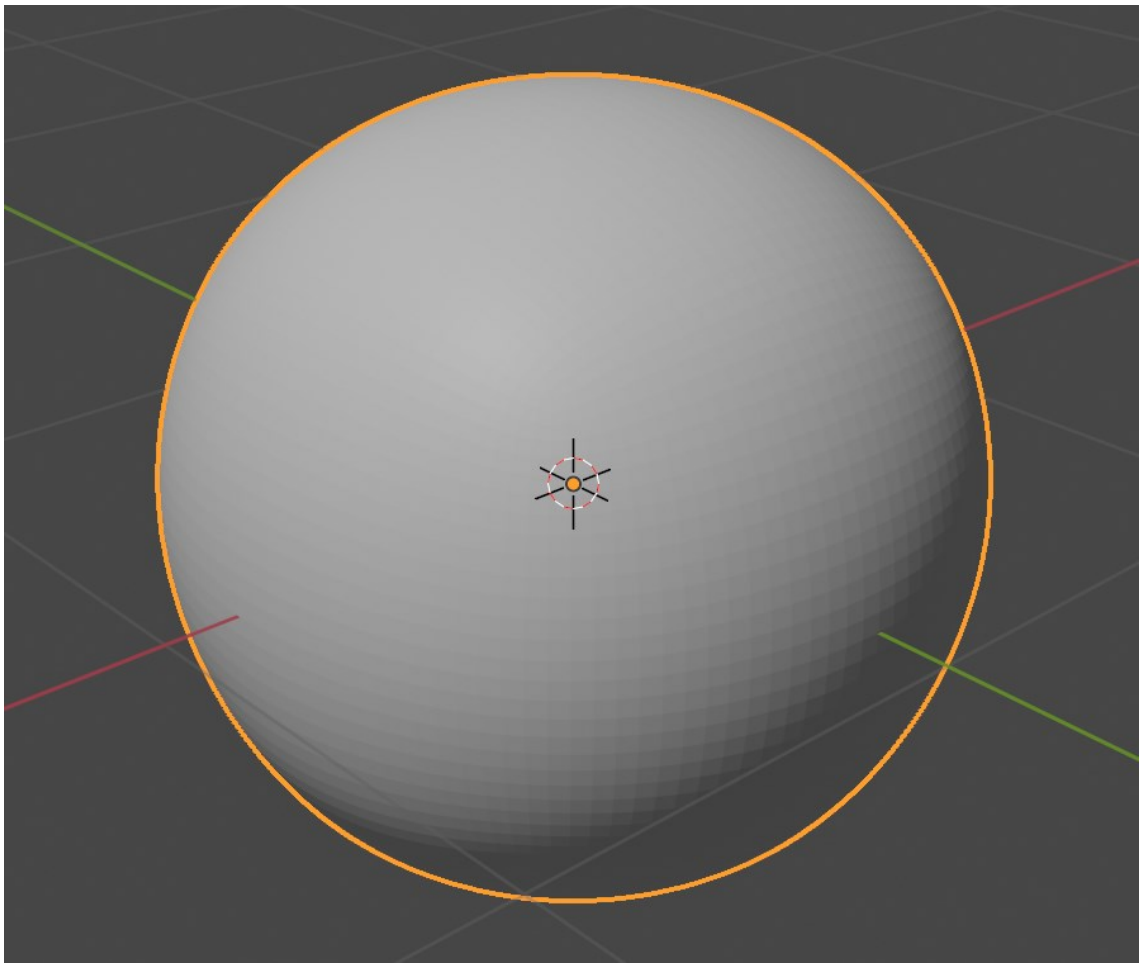
3D-mallinnuksessa paras käytäntö on käyttää nelikulmioita. Kolmiot ovat myös sallittuja, mutta niiden muokkaaminen ei ole yhtä helppoa kuin nelikulmioiden [11]. Alkuun luodaan uusi mesh cube, eli kuutio, mikä koostuu kuudesta

nelikulmaisesta tasosta. Kuvassa 5 on 3D-avaruuteen lisätty kuutio. Kuution keskipiste sijoittuu 3D-avaruuden keskipisteeseen XYZ-akselistossa nollakohtaan.



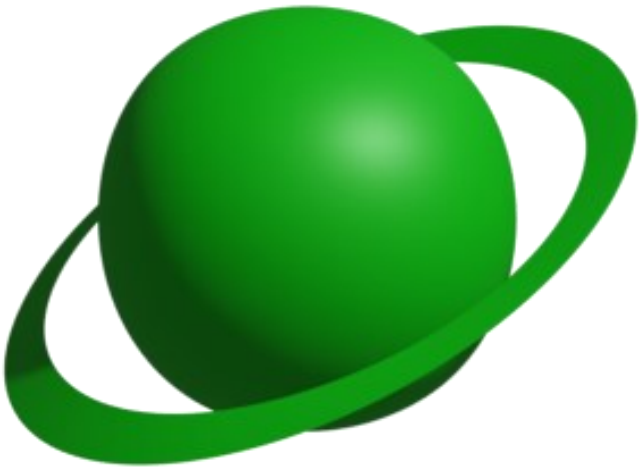
Kuva 6. Mesh-cube, eli kuutio Blenderin 3D-avaruudessa.

Luodulle kuutiolle lisätään muunnin (engl. Modifier) Subdivision Surface, mikä jakaa valitut tasot pienempiin osiin. Kokeilemalla eri arvoja selvisi, että arvo 5 oli optimaalinen. Pienemmällä arvolla pallon tasojen kulmat näkyivät selkeästi. Suurempi arvo ei antanut visuaalisesti parempaa tulosta ja kappaleen vertekseien määrän kasvu kasvatti myös tiedostokokoa. Kuvassa 5 kuutiolle on lisätty Subdivision Surface -muunnin arvolla 5, mikä saa kuution näyttämään pallolta.



Kuva 7. Kuutio, jolle on lisätty Subdivision Surface -muunnin arvolla 5.

Pelissä käytettäville painikkeille pallosta otetaan kuvat punaisena, vihreänä, keltaisena ja sinisenä. Värin voi lisätä Blenderissä Material Properties -valikon alla. Pallolle lisätään uusi materiaali ja materiaalille lisätään haluttu väri. Resoluutioksi asetetaan 500x500. Valikon vaikeustaso-painikkeita varten pallon ympärille lisätään avaruusteeman mukaisesti rengas. Vihreä renkaallinen planeetta on mallinnettu kuvassa 7.



Kuva 8. Vihreän planeettapainikkeen renderöity 3D-malli.

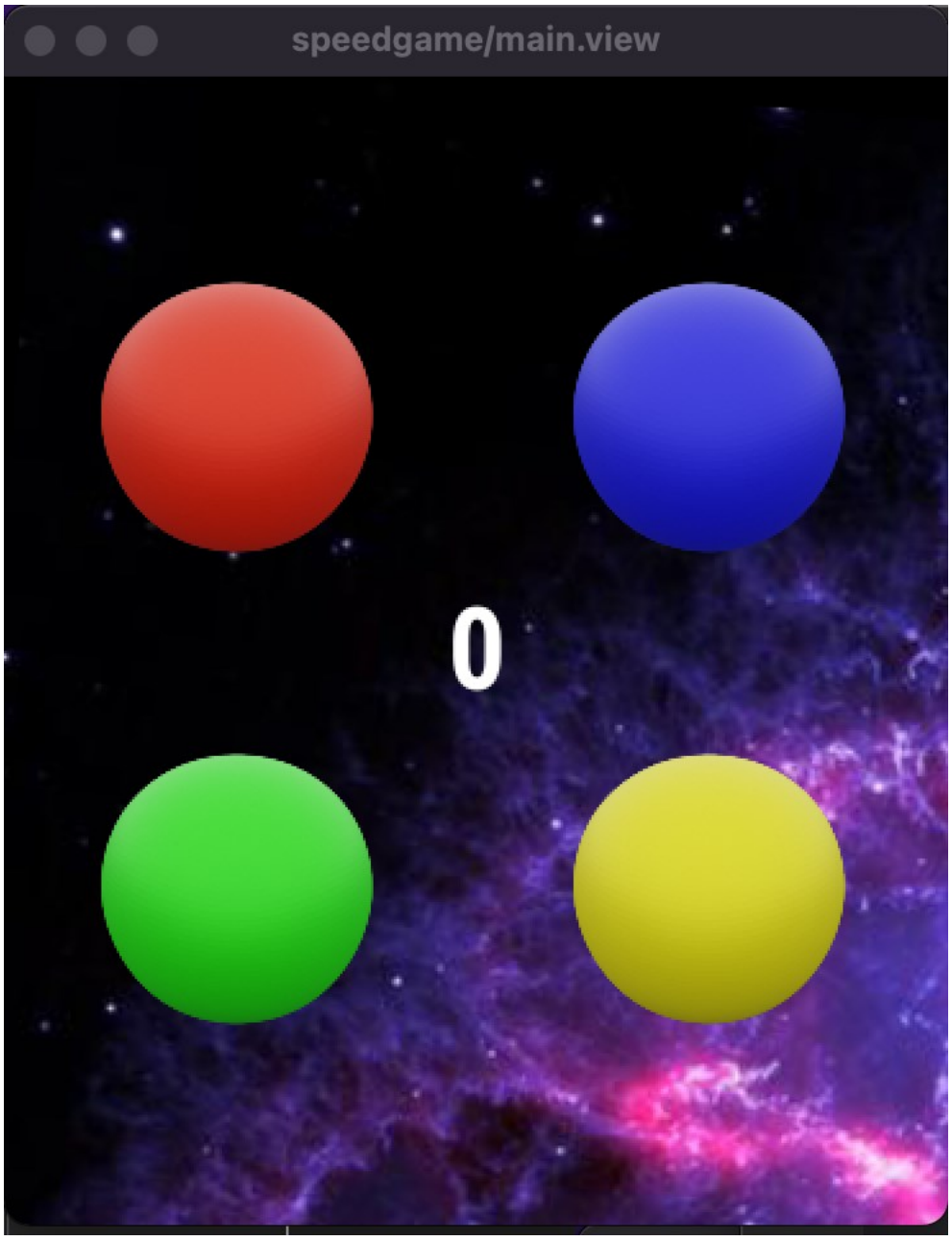
3D-malli on nyt valmis, ja sille tehdään animaatio. Animaatiot tehdään Blenderin Animation-näkymässä. Planeetta halutaan pyörimään tasaisella nopeudella 360 astetta z-akselin ympäri 100 kuvan (engl. Frame) aikana. Animaatio asetetaan alkamaan ensimmäisestä kuvasta ja päättymään sadanteen kuvaan. Oletuksena lisätty bezier-käyrän mukainen interpolaatiomoodi muutetaan lineaariseksi, jotta planeetan pyörimisliike pysyy tasaisena. Ruutunopeus (FPS) pidetään oletusarvossa 24 kuvaa sekunnissa, eli animaatiosta tulee hieman yli neljän sekunnin mittainen. Sama animointi toistetaan uudelleen punaiselle planeetalle.

4.3 Sovelluksen tyyli

Sovelluksen runko, toiminnallisuus ja 3D-kuvat sekä animaatiot ovat valmiit. Nyt elementit asetellaan lopullisille paikoilleen ja animaatio sekä kuvat asetetaan elementteihin. Pelisovelluksen taustakuvaksi asetetaan Nasan Herschel ja Hubble teleskoopeilla kuvattu Crab Nebula [12]. Kuva on Nasan tekijänoikeudetonta materiaalia [13]. Taustakuvasta tehdään 800 pikseliä leveä ja korkea, ja sen keskipiste asetetaan näytön vasempaan laitaan. Kuvaa varten luodaan animationTranslate-elementti, jolla voidaan animoida g-elementin arvoja. G-elementin transform-attribuutille voidaan asettaa siirtymä- (engl. Translate), skaala- (engl. Scale) ja kierto- (engl. Rotate) arvot. Animaatio asetetaan kierto-attribuutille. Nyt animaatio pyörittää kuvaa 360 astetta 180 sekunnin aikana. Lopuksi kuva ja animaatioelementti lisätään g-elementin sisälle. Vaikeustason valitsinpainikkeet lisätään aloitusnäytölle ja tekstien sijainnit ja fonttikoot asetetaan paikoilleen. Pelisovelluksen aloitusnäyttö on esitelty kuvassa 8.

Kuva 9. Vaikeustason valitsemispainikkeet sekä taustakuva lisättynä aloitusnäytölle.

Pelinäkymän paikkamerkkeinä toimineet siniset nelikulmiot vaihdetaan aiemmin Blenderissä luotuihin kuviin (kuva 7), pistelaskureiden tekstit siirretään kohdilleen ja fonttikoot asetetaan sopiviksi. Kuva 10 esittää valmiin pelinäkymän GameView.



Kuva 10. Pelinäkymä, johon on lisätty taustakuva, pistelaskuri ja painikkeiden kuvat.

Pelisovelluksen rakenne, toiminta ja ulkoasu ovat nyt valmiit. Sovellus mukautuu erimuotoisille ja -kokoisille näytöille, ja pelin pelaaminen onnistuu. Tulevaisuudessa peliin voisi lisätä lisää vaikeustasoja ja painikkeisiin voisi lisätä yksityiskohtia, kuten planeetan pinnan muotoja.

5 Wear OS ja WatchOS

Luvuissa 3 ja 4 käsiteltiin älykellosovelluksen kehittämistä Koru-sovelluskehyksen avulla. Tässä luvussa Koru-sovelluskehystä verrataan Googlen ja Applen vastaaviin sovelluskehyksiin. Google ja Apple valittiin vertailuun, koska niiden markkinaosuudet älykellojen käyttöjärjestelminä ovat suurimmat [14].

Monikansallinen teknologiayritys Google julkaisi Android Wear -käyttöjärjestelmänsä vuonna 2014, jonka nimi muutettiin Wear OS:ksi maaliskuussa 2018. Niin ikään monikansallinen teknologiayritys Apple julkaisi WatchOS-käyttöjärjestelmän ensimmäisen version vuonna 2015. Nykypäivänä vuonna 2022, uusin julkaisu on versio 8.5.1 [15]. Applen WatchOS-käyttöjärjestelmä on käytössä vain Applen omissa älykelloissa.

5.1 Wear OS -ohjelmointirajapinta

Wear OS on Androidiin pohjautuva älykelloille optimoitu käyttöjärjestelmä. Android antaa kehittäjälle paljon mahdollisuuksia niin toiminnan kuin ulkoasun kehityksessä. Koru ja Wear OS tukevat näkymien komponenttien määrittelyä XML-kielellä. Android- ja Wear OS -sovellusten rakenne määritellään XML-kielellä AndoridManifest.xml-tiedostoon. Wear OS -sovelluksen komponentin ajonaikainen rakenne, tyylit ja toiminta määritellään yhdessä tiedostossa (kuva 11). Wear OS -käyttöjärjestelmää käyttäviä laitteita valmistaa muun muassa Samsung, LG, Suunto ja Motorola [16].

```
@Composable
fun WatchDetailScreen(
    id: Int,
    scrollState: ScrollState,
    watchRepository: WatchRepository,
    viewModel: WatchDetailViewModel = viewModel(
        factory = WatchDetailViewModelFactory(
            watchId = id,
            watchRepository = watchRepository
        )
    ),
) {
    val watch: WatchModel? by viewModel.watch

    Column(
        modifier = Modifier
            .fillMaxSize()
            .verticalScroll(scrollState)
            .padding(
                top = 26.dp,
                start = 8.dp,
                end = 8.dp,
                bottom = 26.dp
            ),
        verticalArrangement = Arrangement.Top
    ) {
        if (watch == null) {
            Text(
                modifier = Modifier
                    .fillMaxSize()
                    .align(Alignment.CenterHorizontally),
                color = Color.White,
                textAlign = TextAlign.Center,
                text = stringResource(R.string.invalid_watch_label)
            )
```

Kuva 11. Esimerkki Wear OS -sovelluksen näkymän määrittelystä Kotlin-ohjelmointikielellä [17].

Wear OS -sovelluksen toiminnan määrittelyyn voidaan käyttää Kotlin-, Java- tai C++-ohjelmointikieltä. Wear OS -laitteita pystytään käyttämään myös Applen valmistaman iPhonen kanssa [18].

5.2 WatchOS-ohjelmointirajapinta

WatchOS sovelluksen kehityksessä käytetään Applen Swift-ohjelmointikieleen pohjautuvaa SwiftUI:a [19]. SwiftUI:lla luodaan käyttöliittymän rakenne, toiminta ja ulkoasu. SwiftUI:ssa näkymien komponentit ja tyylit lisätään samaan swift-tiedostoon (kuva 12).

```
import SwiftUI

struct Content : View {

    @State var model = Themes.listModel

    var body: some View {
        List(model.items, action: model.selectItem) { item in
            Image(item.image)
            VStack(alignment: .leading) {
                Text(item.title)
                Text(item.subtitle)
                    .color(.gray)
            }
        }
    }
}
```

Kuva 12. SwiftUI-esimerkki [19]. SwiftUI-sovelluksessa elementit, tyylit ja toiminta lisätään samaan tiedostoon.

SwiftUI sisältää paljon erilaisia käyttöliittymäkomponentteja ja kokonaan itse tehtyjä komponentteja pystytään myös toteuttamaan Switf-ohjelmointikielellä. Sovelluksen kehitys SwiftUI:lla eroaa Korusta siten, että SwitftUI:ssa toiminta, rakenne ja tyylit määritellään yhteen tiedostoon. Koru tarjoaa kehittäjälle

helpommat lähtökohdat kehityksen aloittamiseen, koska toiminta, rakenne ja ulkoasu on eroteltu web-kehitystä mukailevaksi.

Lisäksi WatchOS toimii vain Applen valmistamilla laitteilla ja myöskään Bluetooth-yhteyden muodostaminen ei onnistu muiden valmistajien valmistamien laitteiden välille [20]. Koru-sovellukset toimivat muun muassa Windows-, MacOS-, Android- ja web-selainympäristöissä.

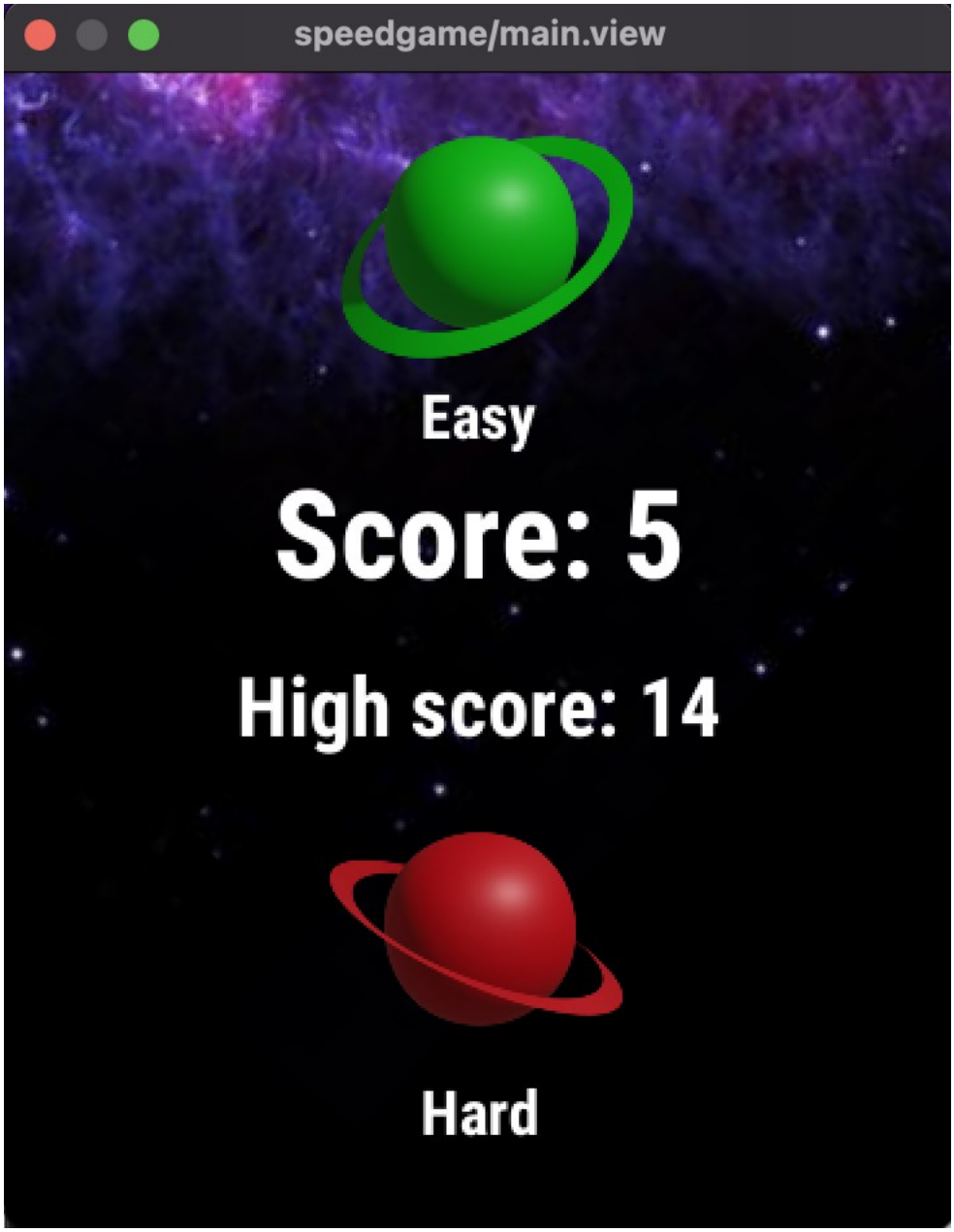
6 Yhteenveto

Tämän työn tarkoituksena oli tutkia sovelluskehityksen rajoitteita ja käytäntöjä älykellojen sovelluskehityksessä ja rakentaa oma demosovellus KoruLab Oy:n Koru-sovelluskehystä käyttäen. Sovellukseen tehtiin kuusi painiketta Blender-3D-mallinnusohjelmalla, joista kaksi animoitiin pyörimään paikallaan 360 astetta z-akselinsa ympäri neljän sekunnin aikana.

Sovelluskehitys älykelloihin Koru-sovelluskehyksellä onnistuu hyvin, kun kehittäjä tuntee web-kehityksessä käytettäviä teknologioita, kuten XML, CSS ja JavaScript. Web-kehityksestä tutut käytännöt ja JavaScriptin standardien mukaiset funktiot helpottavat kehitystyötä. Koru ei tue vielä kaikkia toimintoja, kuten JavaScriptin uutta standardia ECMAscript 6 tai CSS-tyylikielen ominaisuuksia kuten flex-box tai grid. Uusia ominaisuuksia julkaistaan kuitenkin koko ajan lisää. Muistinkäyttö Koru-sovelluksessa eroaa web-sovelluksesta siten, että Korua ajavassa laitteessa on yleensä vähemmän muistia käytettävissä, mikä tulee ottaa huomioon monimutkaisempia sovelluksia kehitettäessä. Koru-sovelluskehityksessä opittuja käytäntöjä ei voida suoraan soveltaa web-kehitykseen, sillä toimintaperiaatteet ovat erilaiset. Web-sovellus toimii yleensä tietokoneen selaimella ja muistia on käytettävissä lähes rajattomasti verrattuna Koru-sovellukseen.

Demo-sovellusta kehittäessä etenkin rajattu näytönkoko osoittautui haasteeksi. Pelin pisteitä ei saatu mahtumaan järkevästi aloitusnäytölle. Ongelma ratkaistiin

niin, että pisteet tulevat näkyviin vasta ensimmäisen pelin jälkeen ja pelin aloittamisen ohjeet sekä pelin nimi puolestaan piilotetaan (kuva 13).



Kuva 13. Aloitusnäyttö pelin jälkeen.

Sovelluskehitystä tutkittiin tarkastelemalla laitteiston teknisiä ominaisuuksia ja vertailemalla KoruLab Oy:n Korua Googlen Wear OS- ja Applen WatchOS-ohjelmointirajapintoihin. Wear OS- ja WatchOS-ohjelmointirajapintojen vertailu vaatii lisää tuntemusta kyseisistä rajapinnoista ja niiden käytöstä sovelluksissa,

jotta heikkoudet ja vahvuudet sovelluskehityksessä tulisivat paremmin esille. Demosovellus kehitettiin onnistuneesti vastaamaan työlle asetettuja vaatimuksia. Se on responsiivinen sekä toimii erikokoisilla ja -muotoisilla näytöillä. Sovellus mukautuu näytön kokoon sekä muotoon, ja näkymät ovat selkeitä ja helposti ymmärrettäviä.

Lähteet

1 Koru Graphical UI Framework. Verkkoaineisto. KoruLab Oy. https://www.korulab.com/product.html. Luettu 19.3.2022.

2 Technology Fast 500 Europe, Middle East & Africa (EMEA). Verkkoaineisto. Deloitte. https://www2.deloitte.com/global/en/pages/technology-media-and-telecommunications/articles/technology-fast-500-emea.html. Luettu 10.3.2022.

3 KoruLab and NXP Semiconductors Collaborate on Next-Generation Wearable User Experiences. Verkkoaineisto. KoruLab Oy. https://korulab.com/Koru_NXP_Press_Release_25022020.pdf. Luettu 10.3.2022.

4 100 suomalaista peliä. Verkkoaineisto. Suomen pelimuseo. https://www.vapriikki.fi/pelimuseo/pelit/. Luettu 16.3.2022.

5 Xiaomi Watch S1. Verkkoaineisto. Xiaomi. https://www.mi.com/global/product/xiaomi-watch-s1/. Luettu 14.3.2022.

6 Fitbit Sense Teardown. Verkkoaineisto. https://www.ifixit.com/Teardown/Fitbit+Sense+Teardown/137130. Luettu 14.3.2022.

7 Apple Watch Battery. Verkkoaineisto. Apple Inc. https://www.apple.com/watch/battery/. Luettu 14.3.2022.

8 Symbian's 'little feature that could' still to be equalled, even on Windows Phone. Verkkoaineisto. All About Symbian. http://www.allaboutsymbian.com/features/item/17601_Symbians_little_feature_that_c.php. 27.5.2013. Luettu 17.3.2022.

9 Documentation. Verkkoaineisto. KoruLab Oy. https://developer.korulab.com/docs/. Luettu 18.3.2022.

10 Blender. Verkkoaineisto. Blender Foundation. https://www.blender.org/. Luettu 18.3.2022.

11 Triangles vs Quads in Blender. Verkkoaineisto. Artistic Render. https://artisticrender.com/triangles-vs-quads-in-blender/. Luettu 20.3.2022.

12 Crab Nebula, as Seen by Herschel and Hubble. Verkkoaineisto. NASA. https://www.nasa.gov/jpl/herschel/crab-nebula-pia17563. Luettu 19.3.2022.

13 NASA Image Use Policy. Verkkoaineisto. NASA. https://gpm.nasa.gov/image-use-policy. Luettu 19.3.2022.

14 Smartwatch Market Share. 2021. Verkkoaineisto. T4 Labs Inc. https://www.t4.ai/industry/smartwatch-market-share. 23.1.2021. Luettu 9.4.2022.

15 WatchOS 8. Verkkoaineisto. Apple Inc. https://www.apple.com/watchos/watchos-8/. Luettu 6.4.2022.

16 Find the perfect watch for you. Verkkoaineisto. Google. https://wearos.google.com/#find-your-watch. Luettu 6.4.2022.

17 WatchFace Sample (Kotlin). Verkkoaineisto. Android. https://github.com/android/wear-os-samples/tree/main/WatchFaceKotlin. Luettu 9.4.2022.

18 Painter, Lewis. 2020. How to use a Wear OS smartwatch with an iPhone. Verkkoaineisto. Tech Advisor. https://www.techadvisor.com/how-to/apple/wear-os-smartwatch-iphone-3791094/. 29.6.2020. Luettu 6.4.2022.

19 SwiftUI. Verkkoaineisto. Apple Inc. https://developer.apple.com/xcode/swiftui/. Luettu 7.4.2022.

20 Persaud, Christine. 2021. Can you use an Apple Watch on Android? Verkkoaineisto. Android Central. https://www.androidcentral.com/can-you-use-apple-watch-android. 8.3.2021. Luettu 7.4.2022.