Speeden Sepelit

ttv18kmm

Iiro Laukkanen ja joonas varis

2018

Contents

[1 Johdanto 2](#_Toc522283722)

[2 Projektin suunnittelu 2](#_Toc522283723)

[2.1 Käyttöliittymä 2](#_Toc522283724)

[2.2 Nopeuspeli 6](#_Toc522283725)

[2.3 Korttipeli 7](#_Toc522283726)

[2.4 Tietokanta 7](#_Toc522283727)

[3 Työn toteutus ( tietoa ohjelmasta) 8](#_Toc522283728)

[3.1 Korttipeli 8](#_Toc522283729)

[3.2 Nopeuspeli 11](#_Toc522283730)

[3.3 Käyttöliittymä 14](#_Toc522283731)

[3.4 MySQL-Database 19](#_Toc522283732)

[4 Pohdintaa 22](#_Toc522283733)

[4.1 Jatko kehitys 22](#_Toc522283734)

[4.2 Itsearviointi 23](#_Toc522283735)

# Johdanto

Osana tietokannat ja käyttöliittymäohjelmointi opintojaksoa Iiro Laukkanen ja Joonas Varis toteuttivat projektin, missä he yhdistivät kurssien projektitöiden vaatimusmäärittelyt ja toteuttivat yksinkertaisen pelikokoelman Speden Spelejen innoittamana.

# Projektin suunnittelu

Ennen projektin toteutuksen aloitusta tehtiin suunnitelma siitä mitä ryhdyttäisiin tekemään, ja millä työnjaolla projekti toteutettaisiin. Projektissa päätettiin käyttää GitLab-versiohallinta palvelua, minne dokumentit ja itse ohjelma koottiin työstöjärjestyksessä. Suunnitteluvaiheessa projektiryhmä päätti yhdessä alustavat toteutustavat, tekniset vaatimukset ja luonnokset käyttöliittymän ulkonäöstä. Kukin suunnitelma on esiteltynä omassa osiossaan tarkemmin.

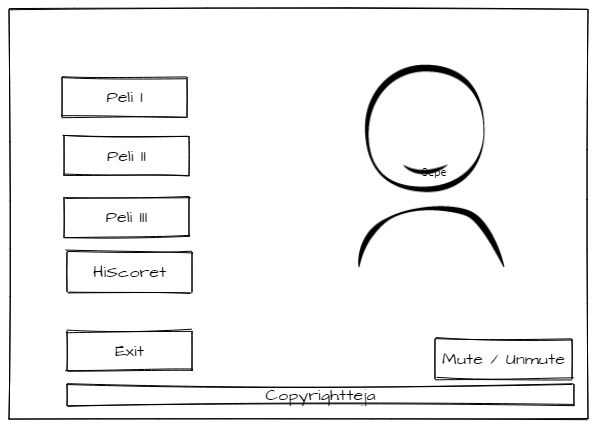
Suunnittelussa ideoitiin myös kolmatta pelimahdollisuutta, Ajanarviointi-peli. Pelissä käyttäjän olisi pidettävä painiketta x-aikaa pohjassa, ja hänen tulos pisteytettäisiin tavoiteajan ja oman tuloksen erotuksen itseisarvosta.

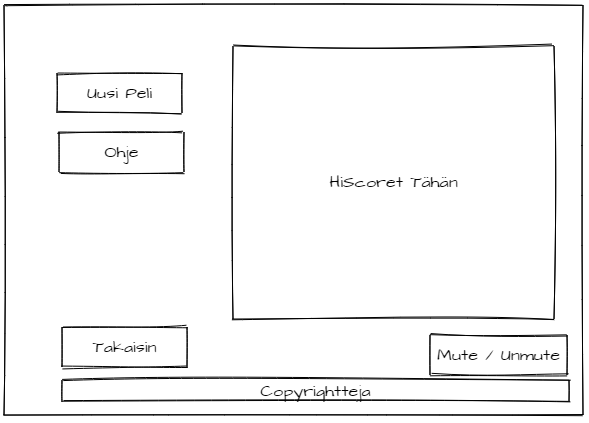
## Käyttöliittymä

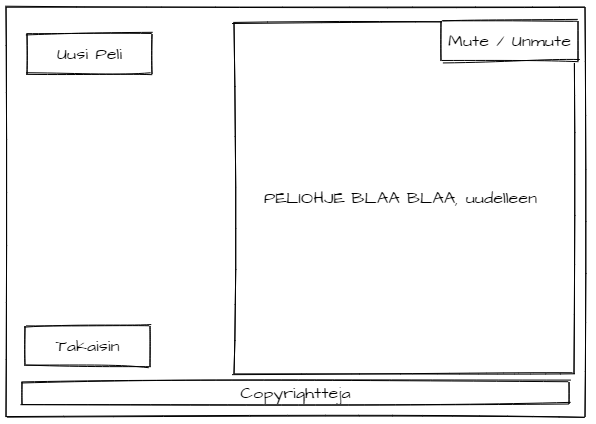
Käyttöliittymän suunnittelussa käytettiin hyväksi Jakob Nielsenin määrittelemiä käytettävyyden periaatteita. Nielsen jakaa käytettävyyden viiteen laadulliseen komponenttiin: opittavuuteen, tehokkuuteen, muistettavuuteen, virheiden käsittelyyn ja tuotteen tyydyttävyyteen.

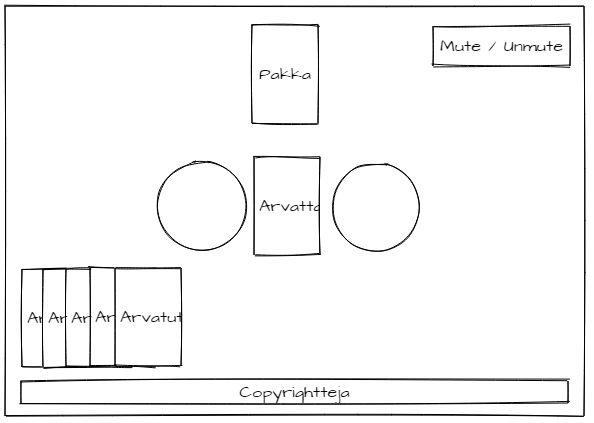
Käyttöliittymä haluttiin olla saman teemainen jokaisella menun näkymällä. Napit sijaitsevat samoilla kohdilla, ja samaa toimintoa ohjaavat napit on sijoiteltu samalla tavalla. Käyttäjän tekemien virheiden mahdollisuus haluttiin minimoida niin hyvin kuin tuotetestaus vaiheessa oli mahdollista. Sovelluksen visuaalinen ilme, musiikki ja äänitehosteet päätettiin asettaa vastaamaan arcade-tyyppisiä pelejä.

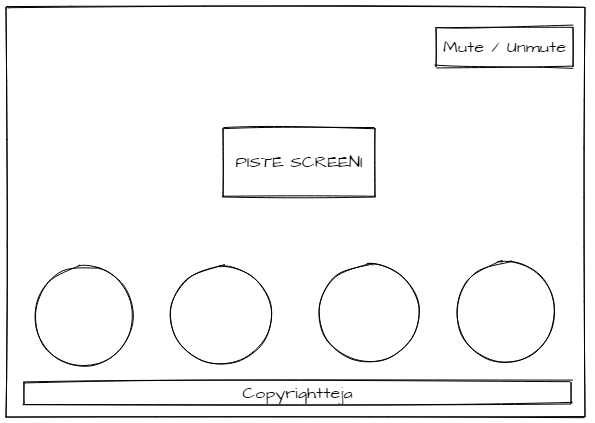
Käyttöliittymäsuunnittelu vaiheessa jokaisesta sovelluksen ikkunasta tehtiin mockup versiot, mitkä ovat esitelty alla. Mockup suunnitelmat tehtiin mockflow-verkkosovelluksessa.

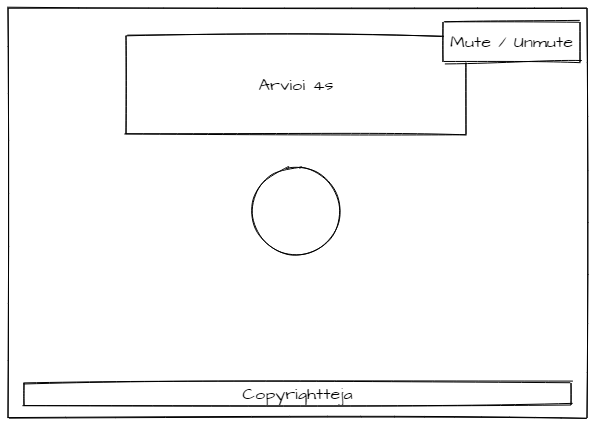












## Nopeuspeli

Nopeuspelin tarkoituksena oli tehdä speden speleistä ja huoltoasemilta tuttu nelipainikkeinen peli, mikä sytyttäisi sattumanvaraisen painikkeen kiihtyvällä tahdilla siihen asti, että käyttäjä tekisi virheen. Peräkkäiset painikkeiden syttymiset eivät voi osua samaan painikkeeseen. Sovelluksen piti tallentaa arvottuja tietoja listaan siltä varalta, että käyttäjä jää pelistä jälkeen. Jokaisella käyttäjän syötteellä nykyistä syötettä verrattaisiin seuraavaan listan elementtiin. Toteutuksessa päätettiin käyttää myös tilakone ratkaisua. Haluttiin myös, että peliä voi pelata sekä hiirellä, että näppäimistöllä.

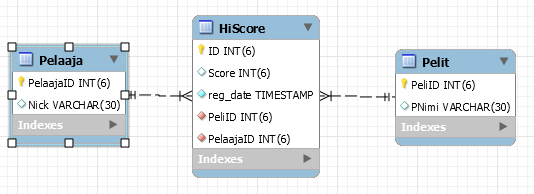
## Korttipeli

Korttipelin perusidea on yksinkertainen: pelaajalla on yksi pakka, joka sekoitetaan. Sekoituksen jälkeen pelaaja saa selata pakkaa läpi. Tarkoitus on, että pelaaja muistelee korttien järjestyksen ja itse pelissä antaa seuraavan kortin värin. Jos väri on oikein, pelaaja saa pisteen ja siirrytään seuraavaan arvattavaan korttiin.

Korttien käsittelyä varten luodaan oma class-tiedosto, jossa on tarvittavat konstruktorit, oliot ja metodit ovat. Kortit toteutetaan oliona, joilla on maa ja numero, kuten oikeilla korteilla. Pakka toteutetaan Stack-rakenteena, johon korttioliot syötetään, ja olioita luodaan aidon pakan mukaisesti 52 kappaletta. Tarvittavia metodeja ovat ainakin pakan sekoitus.

## Tietokanta

Ohjelman tietokantaan on tarkoitus saada tallennettua peleissä pelaajien saamat pisteet ja heidän valitsemansa pelaajatunnukset eli nickit. Tietokanta toteutetaan MySQL-kantana, joka toimii JAMKin LabraNetissä. Tietokantaan tulee kolme taulukkoa: yksi pelaajien nimiä varten, yksi pisteitä varten ja yksi pelejä varten. Taulukot linkitetään toisiinsa seuraavan kuvan mukaan:

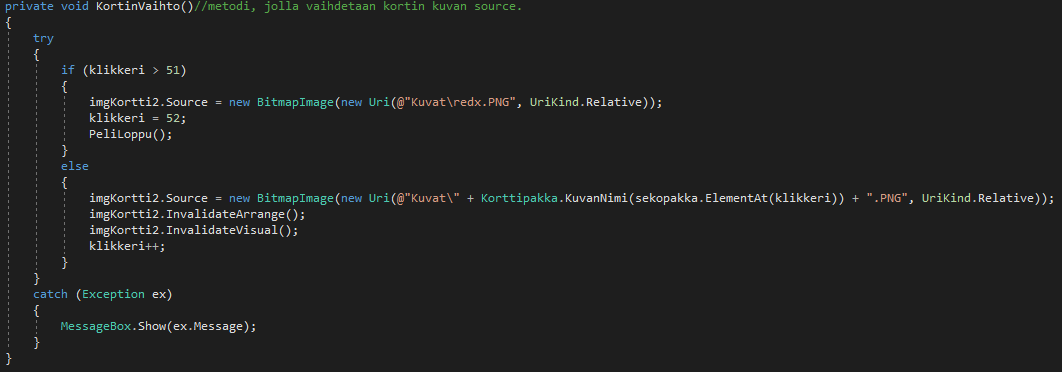


Ohjelmaan tulee myös tietokantahakuja, joilla voi etsiä nickin tai pelin mukaan ennätyksiä. Tietokanta suojataan SQL-injectioneilta ja salasana tulee erilliseen config-tiedostoon, eli sitä ei ”kovakoodata” suoraan ohjelmaan.

# Työn toteutus ( tietoa ohjelmasta)

## Korttipeli

Pelin käynnistyttyä luodaan lista, johon myöhemmin tallennetaan kortti-olioita. Sitten kutsutaan korttipakka-luokan LuoPakka-metodia, johon lähetetään kyseinen lista. Metodi luo ensin 52 kortti-oliota (13 numeroa neljään maahan) ja tallentaa ne saatuun listaan. Kyseistä listaa saa tässä vaiheessa kelata edestakaisin. Käyttöliittymään ohjelma etsii korttiolion numeron ja maan perusteella oikean kuvan näytettäväksi käyttäjälle; kaikki tapahtuu KortinVaihto-metodin avulla, joka kutsuu Korttipakka-luokan KuvaNimi-metodia, joka palauttaa esimerkiksi ristikuutosen ollessa kyseessä merkkijonon ”C6”, joka samalla tulee osaksi kuvan lähdettä. Itse PNG-kuvat on nimetty samalla tavalla jotta source on järkevä.

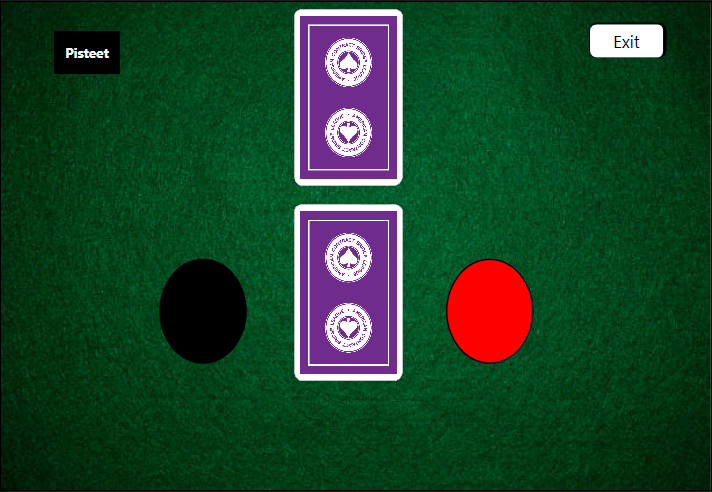


Seuraava vaihe on pakan sekoitus, joka on pakko tehdä ennen varsinaisen pelin aloittamista (pelin aloitusnappi on disabloitu kunnes sekoitus on tehty).

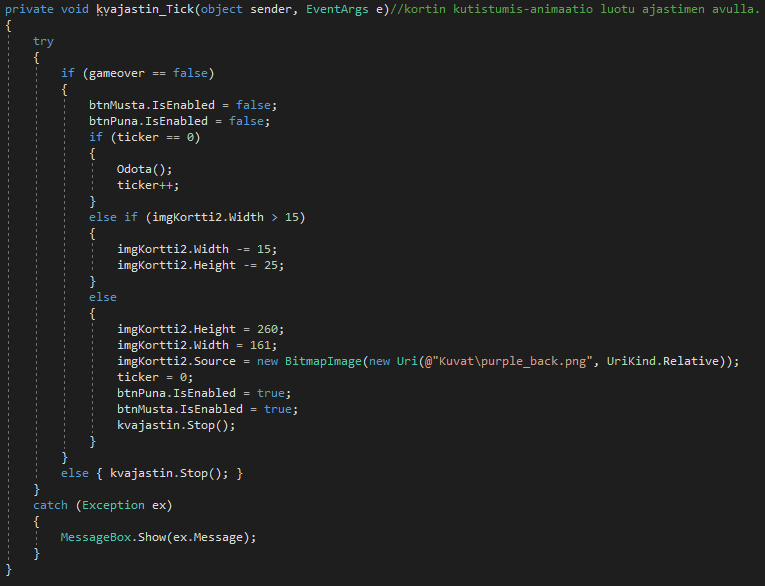


Sekoitus tapahtuu niin, että sekoittamaton lista kortteja lähetetään Korttipakka-luokan SekoitaPakka-metodiin. Se käyttää C#-random-metodia jonka avulla sekoittaa korttien indeksit ja siirtää ne sitten stackkiin for-loopilla. Viimeisenä kutsutaan Korttipakka-luokan TallennaPakka-get-setteriä, johon stack tallennetaan. Nyt peli voidaan käynnistää ja se luo uuden ikkunan, jossa varsinainen peli tapahtuu, ja sulkee nykyisen.

Peli-ikkunan auettua luodaan uusi globaali stack, johon ikkunan pääohjelma hakee sekoitetun pakan Korttipakka-luokan get-setteristä. Lisäksi luodaan DispatcherTimer-ajastin, jota käytetään korttien animoinnissa. Seuraavaksi käyttäjä painaa joko punaista tai mustaa nappia (kts kuva).

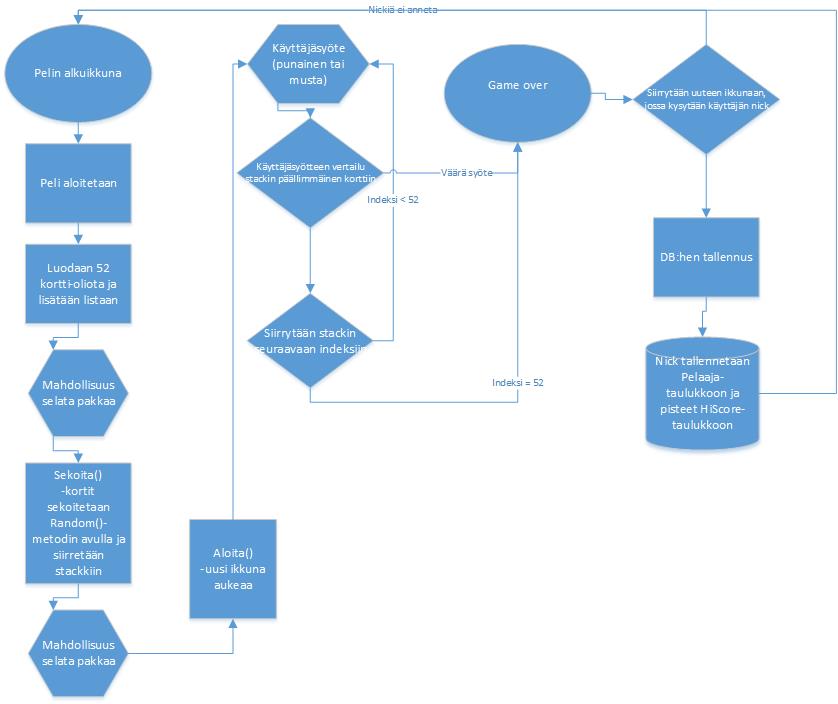


Jompaakumpaa nappia painamalla ohjelma vertaa stackin seuraavan kortin maata painettuun väriin: jos painetaan mustaa nappia ja kortti on Pata tai Risti, kortti näytetään. Sama punaisella. Oikean arvauksen tapahduttua ohjelma antaa käyttäjälle pisteen ja aloittaa kortin kutistumis-animaation, joka on toteutettu ajastimella joka kutistaa kortin kuvaa tietyin intervallein ja lopuksi palauttaa kuva-elementin alkuperäiseen kokoonsa.



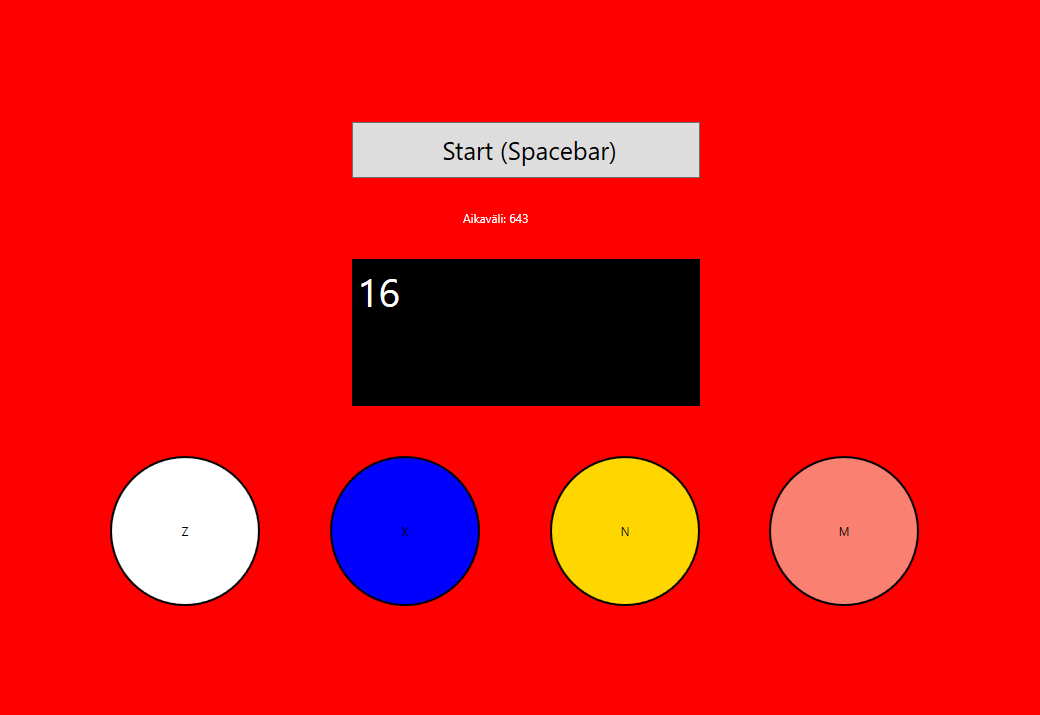
Pisteet tallennetaan PojoTallennus-get-setteriin, josta ne haetaan GameOver-ikkunan aikana tallennettavaksi tietokantaan. Väärän arvauksen kohdalla peli loppuu, eli ohjelma kutsuu PeliLoppu-metodia joka julistaa gameover-tilan päälle ja avaa GameOver-ikkunan ja sulkee nykyisen.

Kaikkiaan Korttipelin UML-kaavio on seuraavanlainen:

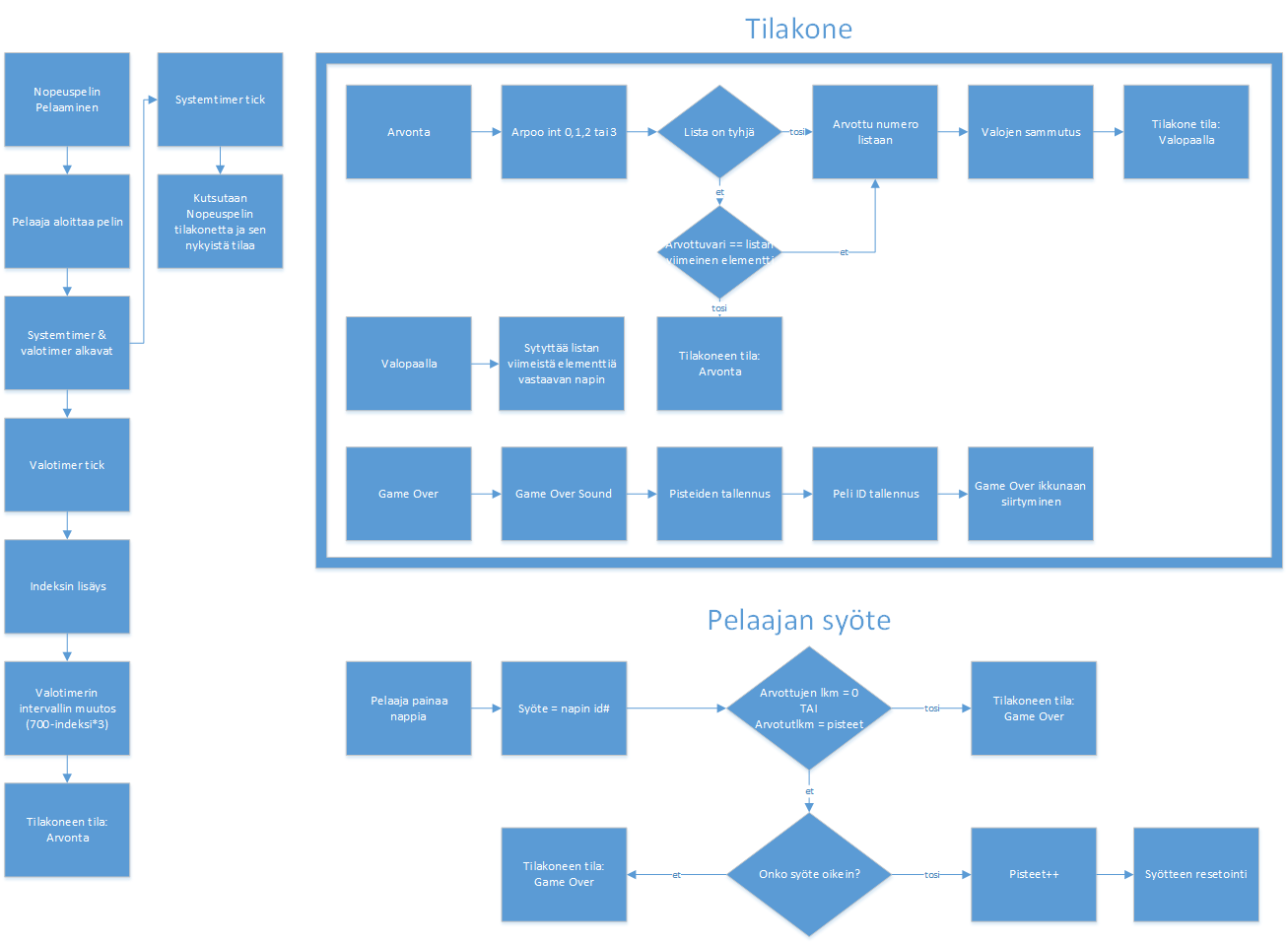


## Nopeuspeli

Nopeuspelin ulkoasu vastaa suunniteltua mallia. Suunnitellut teknologiat on sisällytetty ratkaisuun, mutta niiden lisäksi jouduttiin käyttämään ajastimia, mitkä ohjasivat pelin päivitysnopeutta, sekä tilakonetta. Myös suunniteltuun vuokaavioon jouduttiin tekemään muutoksia.



Pelinäkymässä näkyy aloituspainike, jota voidaan painaa vain kerran, 4 eri väristä painiketta, piste-screen ja aikaväli, missä näytetään nykyinen nappuloiden syttymistahti. Ohjelmaa voi käyttää näppäimistöllä tai hiirellä. Näppäimistöä käytettäessä painikkeet aktivoidaan niissä ilmoitetuilla näppäimillä. Valon syttymistä painikkeessa mallinnetaan sen värin muuttumisella valkoiseksi, kun seuraava valo syttyy palaavat muut painikkeet alkuperäisen värisiksi.



Nopeuspelin UML-kaaviosta selviää ohjelman kulku. Kun peli alkaa, aloitetaan ajastimet “systemtimer” ja “valotimer”. Tästä eteenpäin näiden ajastimien tick-tapahtuma ohjaa pelin toimintaa. Systemtimer on pelin kellotaajuus (100Hz), joka kutsuu tilakonetta jokaisella tickillään. Valotimer ohjaa painikkeiden syttymisnopeutta. Jokaisella valotimer tick-tapahtumalla sen arvoa pienennetään 3ms ja asetetaan tilakoneen tila arvontaan.

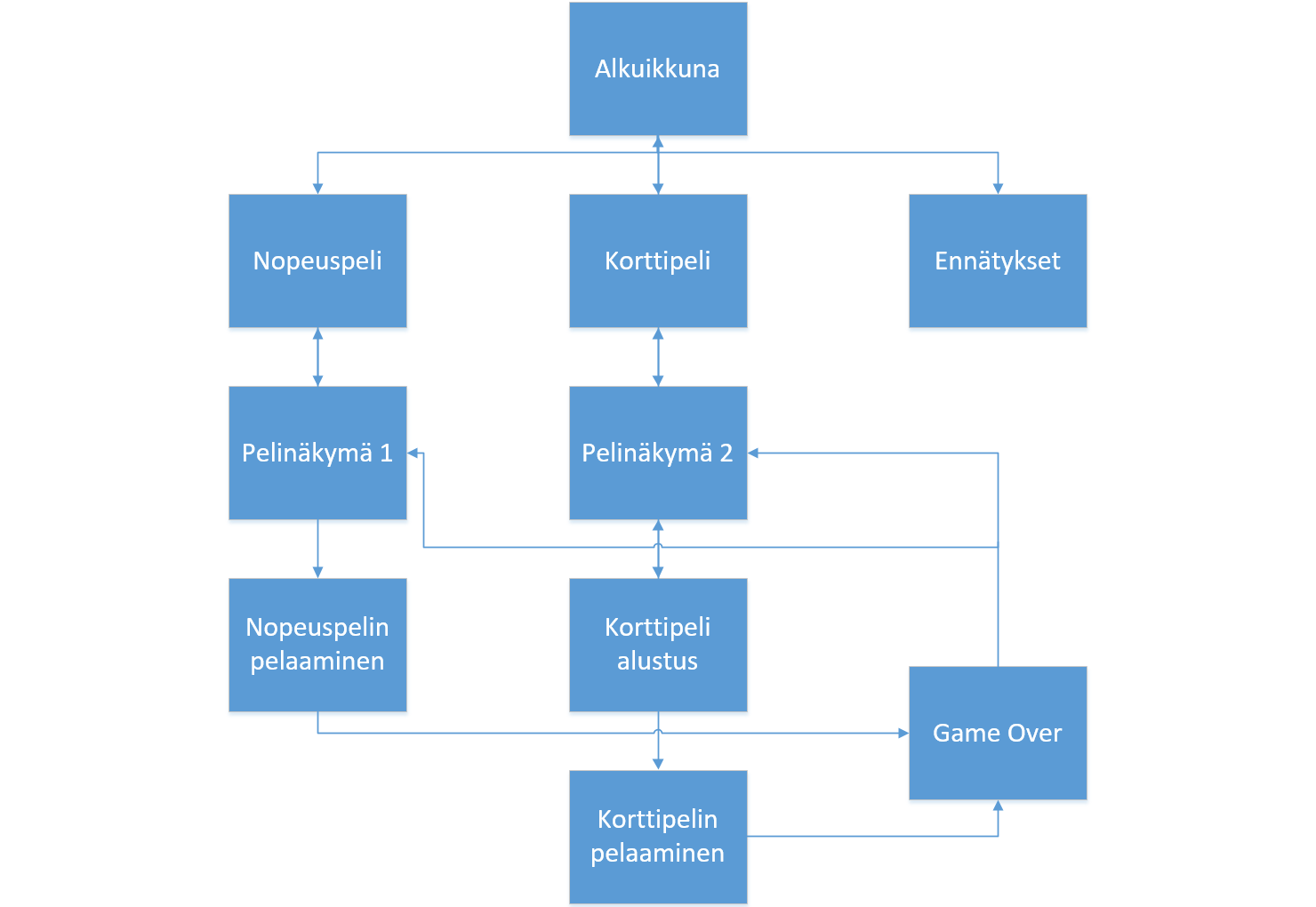
Tilakoneessa on 3 tilaa: Arvonta, Valopaalla ja Game Over. Kun valotimer asettaa tilakoneen tilan arvontaan ja systemtimer kutsuu tilakonetta, arpoo ohjelma int-arvon välillä 0-3, tarkistaa onko lista tyhjä tai että arvottu numero on eri kuin listan viimeinen elementti, jos ei arvotaan se uudelleen, kunnes saadaan eri arvo. Tämän jälkeen arvottu arvo lisätään listaan, käyttöliittymän painikkeiden värit asetetaan alkuperäisiksi ja tilakoneen tila asetetaan valopaalla-tilaan. Valopaalla-tilassa tilakone asettaa viimeksi arvottua numeroa vastaavan napin värin valkoiseksi. Game Over-tilassa ohjelma pysäyttää ajastimet, kutsuu Music-luokan PlayPause-metodia ja avaa game over ikkunan.

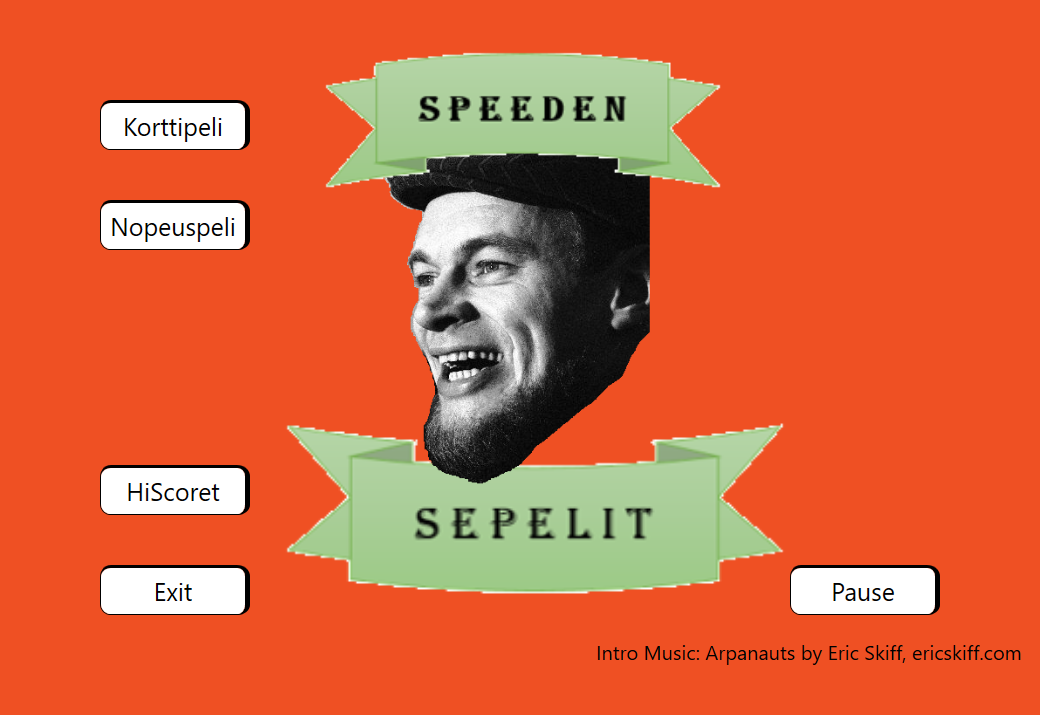
Käyttäjän syötteellä soitetaan kustakin painikkeesta eri ääni, mikä on tehty audacity ohjelmistolla. Tämän jälkeen syötettä verrataan listaan. Mikäli käyttäjä on listaa edellä, katsoo ohjelma käyttäjän tehneen virheen ja lopettaa pelin. Samoin tapahtuu silloin, kun käyttäjän syöte ei vastaa listan seuraavaa elementtiä. Tämän tarkistukseen käytetään pelaajan pisteitä, joka on myös listan indeksi missä arvottu numero on. Pelaaja voi siis jäädä peliä jälkeen siten, että painaa painikkeita myöhässä oikeassa järjestyksessä, eikä tästä rangaista pelaajaa.

## Käyttöliittymä

Toteutettu käyttöliittymä oli erittäin lähellä alkuperäistä suunnitelmaa. Käyttäjän navigointia helpottamaan sovelluksen napit ovat nimetty niiden toimintoa kuvastavalla tavalla. Käyttöliittymään lisättiin vähäisesti animointia ja äänitehosteita sovelluksen elävöittämiseksi.

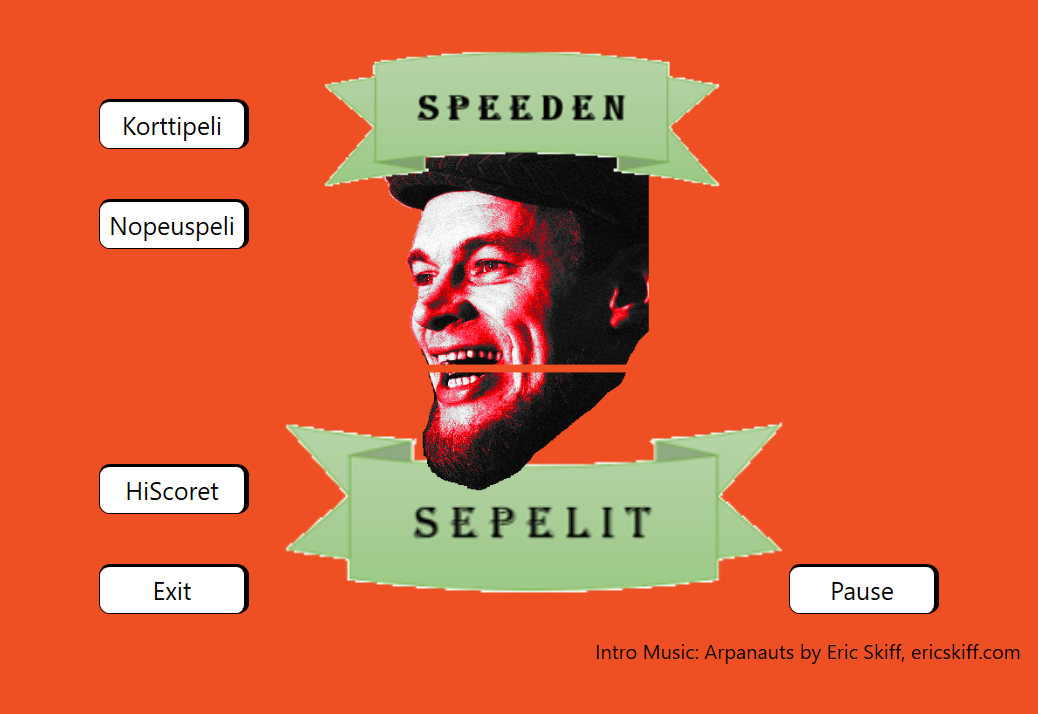
Käyttöliittymän flow on esitetty UML-kaaviossa.



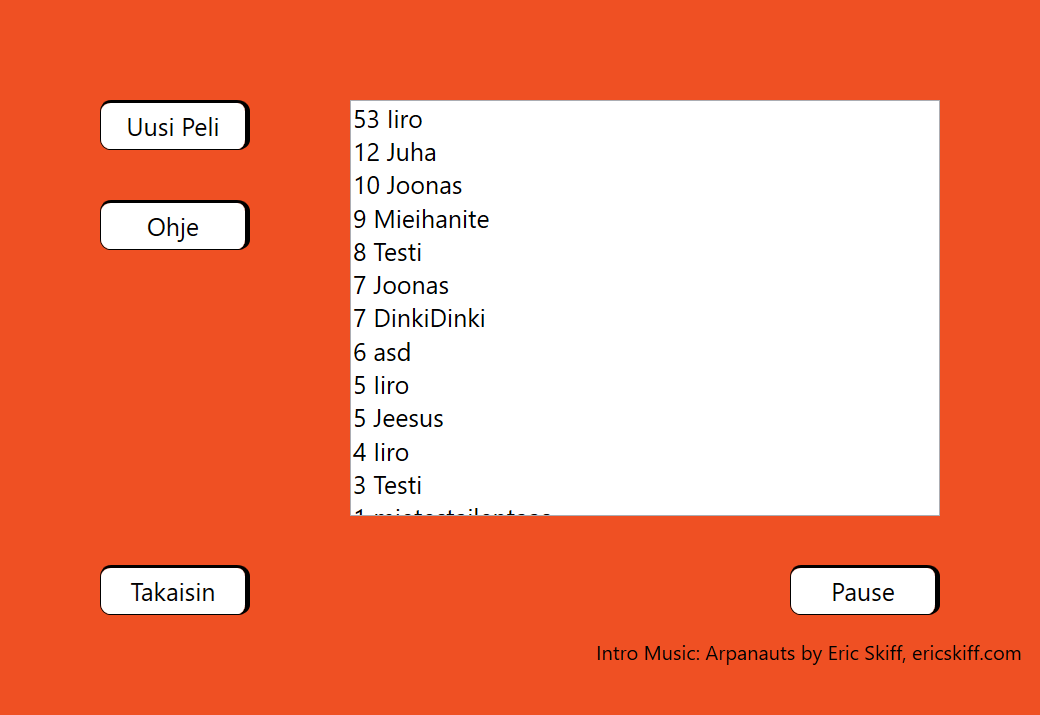


Käyttöliittymän käytettävyys onnistui kiitettävästi. Sovelluksen käyttö on itse-selitteistä, eikä vaadi ulkopuolista ohjeistusta. Napit ovat aseteltu selkeille paikoille, sovelluksen yleisilme on yhtenäinen ja selkeä. Sovelluksen estetiikka ja äänimaailma ovat sovelluksen luonteeseen sopivat. Kun pelaaja painaa nappia sovelluksessa, saa hän äänipalautteen omasta toiminnastaan.

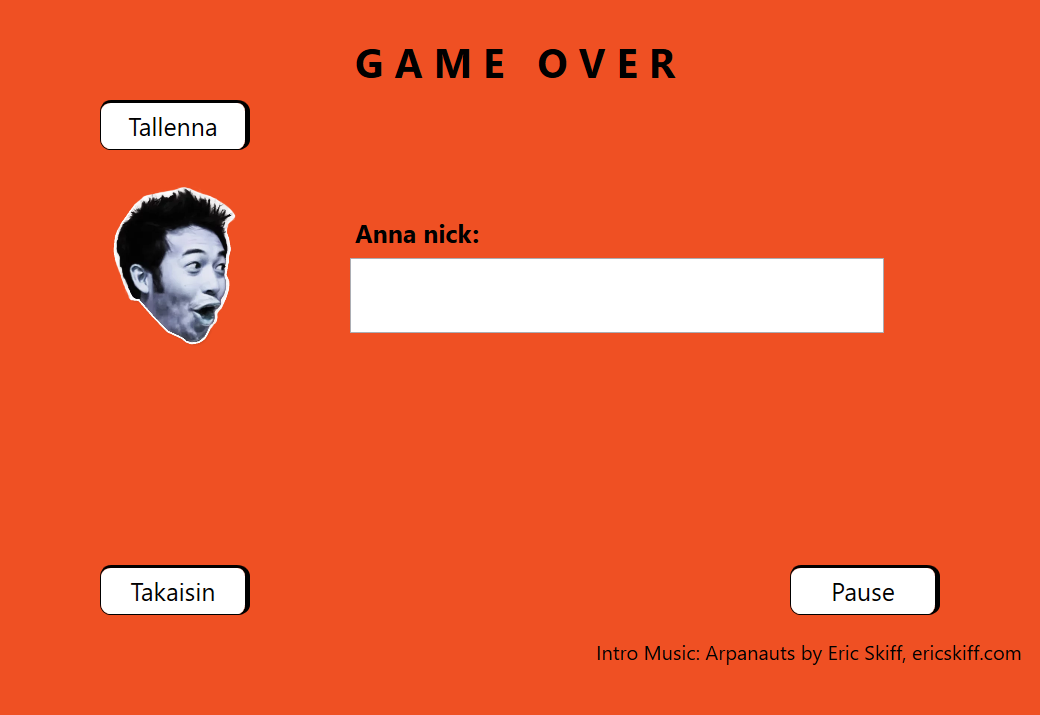
Käyttöliittymän oikeassa alakulmassa on musiikille kontrolli, mikä säilyttää tilansa eri ikkunoiden välillä. Kun pelaaja klikkaa Korttipeli tai Nopeuspeli painiketta nauraa Spede animoidusti käyttäjälle ja soittaa ääniraidan.

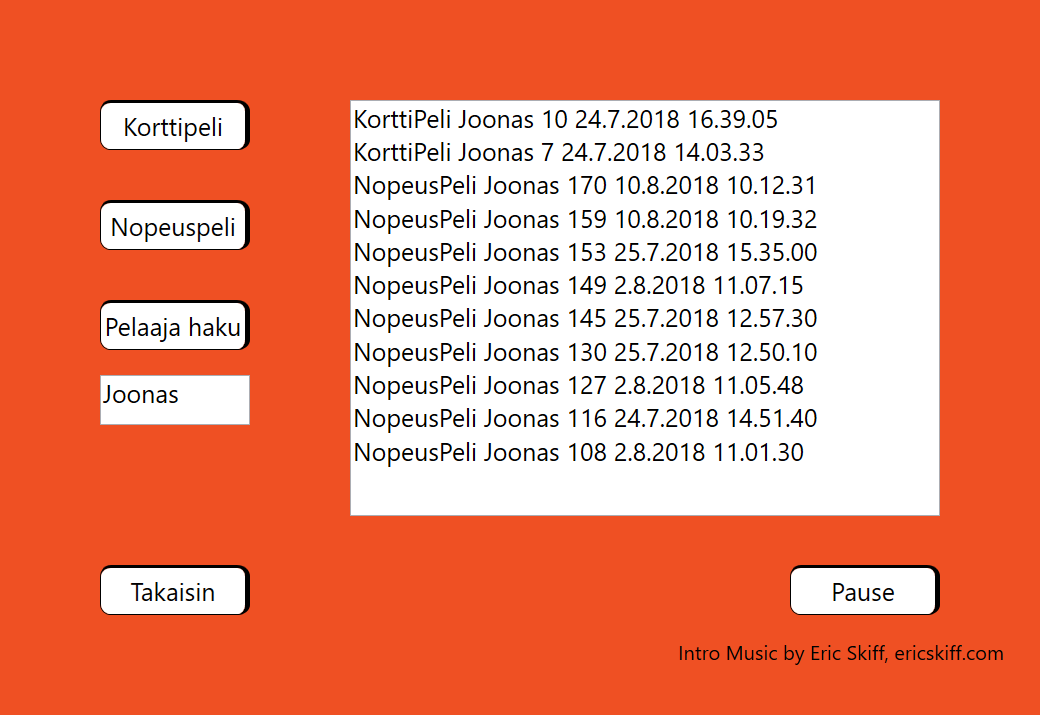


Korttipelin ja nopeuspelin ikkunat ovat samanlaiset, niissä on Uusi Peli, takaisin Pause/Play ja Ohje/Hiscores painikkeet. Kun Pause/Play painaa vaihtuu napin sisältö ja se pysäyttää tai aloittaa musiikin. Kun Ohje/Hiscore nappia painaa, vaihtuu textbox elementin sisälle pelikohtaiset ohjeet, jos napin sisältö oli ohje, tai takaisin pelikohtaisiin hiscore näkymään.



Kun peli päättyy saapuu pelaaja GAME OVER-ikkunaan, missä hän voi tallentaa halutessaan tuloksensa antamallaan nickillä tai palata takaisin saman kuin pelattu peli peli-ikkunaan mistä voi aloittaa uuden pelin tai palata alkuikkunaan.





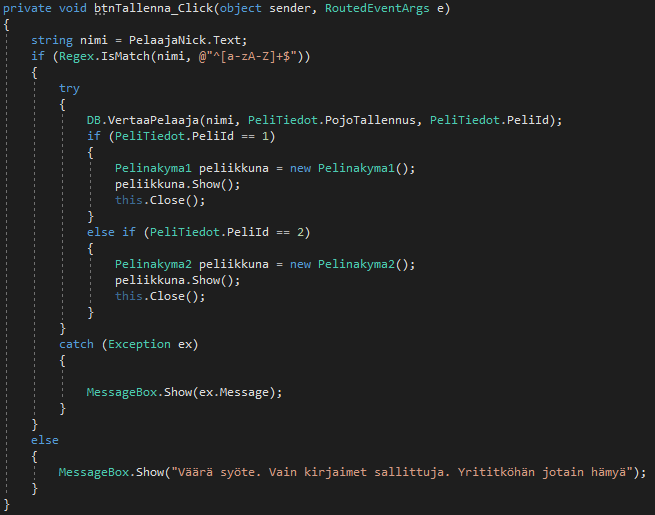
Päävalikosta löytyvästä Hiscore ikkunassa voidaan tarkastella 1) Korttipelin ennätyksiä, 2) Nopeuspelin ennätyksiä tai 3) hakea tietokannasta pelaajan kaikki ennätykset kaikissa peleissä.

## MySQL-Database

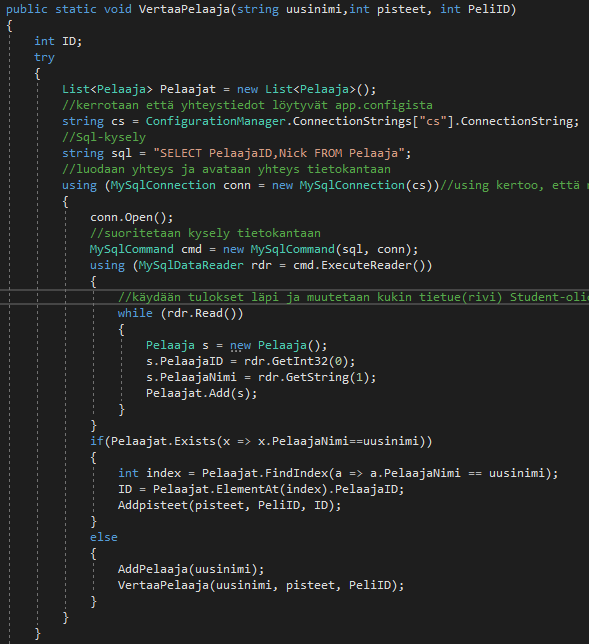
Tietokantaan tallennetaan käyttäjän peleistä saamat pisteet sekä hänen valitsemansa nickin. Tietokanta on JAMKin LabraNetissä, MySQL-tietokantana. Tallennus tapahtuu aina GameOver-ikkunassa, johon käyttäjä syöttää nickinsä. Tietokantayhteyksissä tarvittava tieto löytyy erillisestä app.config-tiedostosta, jota kutsutaan jokaisen metodin yhteydessä.



Tallenna-nappia painamalla PojoTallennus-get-setteriin tallennetut pisteet, käyttäjän syöttämä nick ja pelin ID annetaan DB-luokan VertaaPelaaja-metodille. Lisäksi GameOveri-ikkuna suljetaan ja uusi ikkuna aukeaa. Aukeava ikkuna riippuu pelistä, eli palataan jommankumman pelin alkuikkunaan. Lisäksi SQL-injectionit ovat estetty tässä vaiheessa: nickiin kelpaa vain merkit a-z, ja rajaus tapahtuu RegEX-funktiolla. Jos jotain muuta yritetään syöttää, ohjelma palauttaa virheilmoituksen.

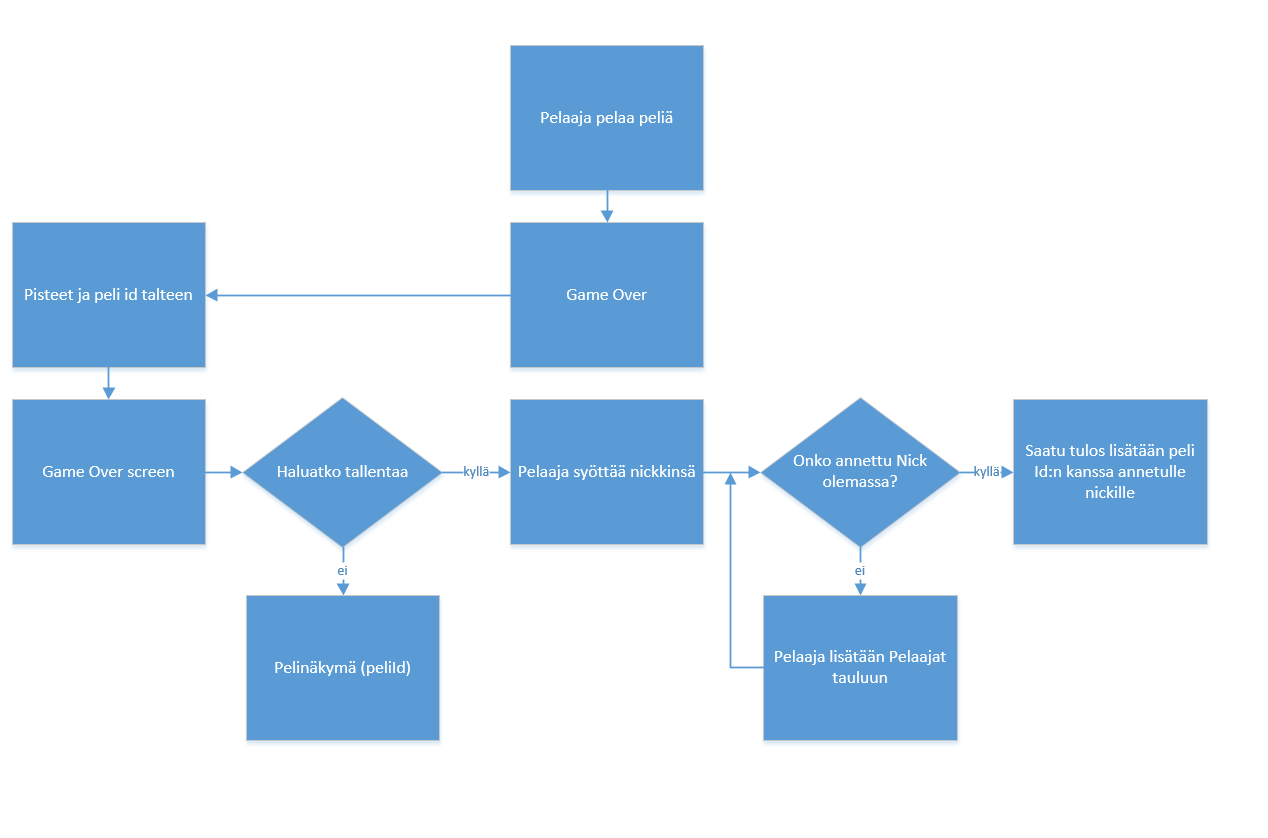


VertaaPelaaja-metodi avaa yhteyden tietokantaan, hakee kaiken datan Pelaaja-taulusta ja syöttää ne listaan Pelaaja-olioina. Sitten metodi vertaa syötettyä nickiä listassa oleviin pelaajien nimiin. Jos pelaaja löytyy, kutsutaan AddPisteet-metodia, johon lähetetään saadut pisteet, pelin ID ja nick. Jos pelaajaa ei löydy, kutsutaan AddPelaaja-metodia, johon lähetetään vain uusi nick, jonka jälkeen rekursiivisesti VertaaPelaaja-metodia jolloin kutsutaan väkisinkin AddPisteet-metodia koska uusi pelaaja on lisätty kantaan.



AddPisteet-metodi avaa yhteyden tietokantaan johon se yksinkertaisesti syöttää INSERT-komennolla HiScore-tauluun pelaajan ID:n, pelin ID:n ja pisteet. AddPelaaja- metodi avaa myös yhteyden tietokantaan ja lisää INSERT-komennolla pelaajan nickin Pelaaja-tauluun.

Ohjelmassa suoritetaan myös tietokantahakuja, jotka tapahtuvat, kun käyttäjälle näytetään eri pelien ja pelaajien pisteitä. Tätä varten on kaksi metodia, HaePisteet- ja PelaajanPisteet-metodit. Nämä eroavat toisistaan lähinnä sillä, että toisessa saatu data rajataan olemassa olevan nickin mukaan ja toinen pelin mukaan. Kaikkiaan tietokantatoiminnan logiikka selviää seuraavasta kaaviosta:



# Pohdintaa

Projekti oli ensimmäinen laatuaan molemmille projektiorganisaation jäsenille. Täysivaltaista sovelluksen kehitys tietotaitoa ei aiemmin siis ollut. Projekti opetti hahmottamaan suurpiirteisesti ohjelmointiprojektin läpiviennin vaiheita ja antoi kuvan kunkin vaiheen haasteellisuudesta, ajankäytöstä ja tärkeydestä muihin projektivaiheisiin nähden.

## Jatko kehitys

Projektissa käytetyt tekniset ratkaisut olivat osittain “rujoja”. Kun osaamista kertyy enemmän, ohjelman olisi voinut toteuttaa varmasti siistimmin ja tehokkaammin. Pelikokoelmaa voisi kasvattaa lisäämällä siihen pelejä.

Kustomoitavuus lisäisi ohjelman käytettävyyttä. Yksilöllinen kontrollien luonti pelien hallintaan olisi hyvä ratkaisu tähän. Nickiin voitaisiin liittää salasana, ja lisätä sovellukseen sisäänkirjautumistoiminto, jolloin omaa tiliään voisi hallita sille tehdyllä ikkunalla.

Tietokannan kasvaessa on myös tärkeää rajoittaa hakuja siten, ettei käyttöliittymässä näytetä tarpeetonta informaatiota. Kunkin pelinäkymän hiscore-ruutu voisi näyttää esimerkiksi 25 parasta tulosta. Tiedot voisi kuitenkin säilyttää siten, että kaikkia tuloksia pääsisi selaamaan hiscore ikkunasta. Databasen kokoa voisi rajoittaa siten että pelaaja taululle ja hiscore taululle antaisi maksimi rivimäärä arvot. Tälläisenaan sovellus olisi kallis ylläpitää, jos se saisi suosiota.

Ohjelmasta voisi luoda myös erillisen dev-sovelluksen, missä järjestelmän ylläpitäjä pääsisi poistamaan ja muokkaamaan tietokantaa vapaasti. Tämän voi kuitenkin toteuttaa MySQL-Workbench-sovelluksella, joten emme näe sitä kovinkaan tarpeellisena.

Ohjelman interaktiivisuutta ja käytettyjen medioiden hallintaa voidaan myös parantaa. Animointeja käytettiin projektissa hyvin niukasti. Myös musiikki- ja äänitehosteiden tasoja olisi voinut toteuttaa paremmin.

## Itsearviointi

Mielestämme onnistuimme projektissa erinomaisesti. Tekniset rajaukset, mitä teimme alussa, pitivät. Lukuun ottamatta ajanarviointipelin toteuttamista. Tietokanta jonka loimme ohjelmaa varten toimi sille annettuine hakuineen ja lisäyksineen moitteetta. Käyttöliittymä on siisti ja teemaltaan yhtenäinen. Ohjelmaan ei jäänyt yhtään tiedossa olevaa bugia, ja viimeinen versio on hyvin toimintavarma.

Suurimmat kehityskohteet projektiorganisaatiolla ovat projektin suunnittelussa ja sen aikataulutuksessa. Tämän projektin jälkeen osaamme priorisoida eri projektivaiheita paremmin ja kiinnittää huomioita erityisesti alkutöihin.

Ehdotamme työlle arvosanaa 5 tässä raportissa esitettyjen perusteluiden pohjalta.