# NFC-MOBIILIPELIEN KEHITYS

Jussi Yli-Rantala

Opinnäytetyö
Joulukuu 2014
Tietotekniikan
koulutusohjelma
Ohjelmistotekniikka

TAMPEREEN AMMATTIKORKEAKOULU
Tampere University of Applied Sciences

**TIIVISTELMÄ**

Tampereen ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka

Jussi Yli-Rantala:
NFC-mobiilipelien kehitys

Opinnäytetyö 37 sivua, joista liitteitä 6 sivua
Joulukuu 2014

---

Työ perustuu projektiin, jossa tehtävänä oli toteuttaa demo mobiilipelistä, joka hyödyntää NFC-teknologiaa. Työssä käydään läpi teknistä taustaa, työkaluja ja ajatuksia suunnittelusta, joita tarvitaan luodessa NFC-mobiilipeliä.

Ensin tarkastellaan NFC-teknologiaa ja ohjelmistoja, joita hyödynnettiin projektin aikana. Lisäksi käymme läpi NFC-mobiililaitteita käyttöjärjestelmittäin.

Laitteiden jälkeen tarkastellaan ensin pelimarkkinoilla menestyneitä tuotteita, joissa on hyödynnetty NFC-teknologiaa.

Sen jälkeen perehdytään mobiilipelien suunnitteluun yleisesti, jonka jälkeen käydään läpi projektin aikana syntyneitä ajatuksia suunnittelusta NFC:n hyödyntämisestä peleissä.

Ennen lopullista pohdintaa, käydään läpi projektia, johon työ perustui.

---

Asiasanat: NFC, mobiili, pelaaminen, suunnittelu

**ABSTRACT**

Tampere University of Applied Sciences
Degree programme in ICT Engineering
Software Engineering

Jussi Yli-Rantala:
Developing of NFC mobile games

Bachelor's thesis 37 pages, appendices 6 pages
December 2014

---

This work is based on a project, which had a goal to design a demo of a mobile game that utilizes NFC technology. In this work we will go through the technical side, tools and thoughts about design, that are necessary for developing a NFC mobile game.

First we will have a look at the NFC technology and software, which were used during project. In addition we will go through NFC mobile devices by operating system.

Then we will have a look on successful products on the gaming market that utilizes NFC technology. After that we will familiarize ourselves on the mobile game design in general and then we will go through the thoughts about utilizing NFC in mobile game design that arose during the project.

Before the final thoughts, we will go through the project this thesis is based upon.

---

Key words: NFC, mobile, gaming, development

**SISÄLLYS**

**LYHENTEET JA TERMIT**

| | |
|---|---|
| NFC | Near field connection, RFID:n modifikaatio |
| RFID | Radio Frequency IDentification, radiotaajuinen tunnistus |
| Tagi | RFID:n tallennusmedia |

# 1 JOHDANTO

Tässä työssä käydään läpi työvaiheita projektissa, jossa tehtävänä oli suunnitella mobiilipeli, jossa hyödynnetään NFC-teknologiaa. Luvussa 2 perehdytään NFC-teknologiaan ja eri työkaluihin, joita projektissa käytettiin. Luvussa 3 tarkastellaan käyttöjärjestelmittäin laitevalikoimaa, jossa NFC-teknologia on käytössä. Luvussa 4 tarkastellaan eri konsolipeliratkaisuja, joissa NFC-teknologiaa on hyödynnetty ja käydään läpi, ensin mobiilipelin suunnittelua yleisesti ja lopuksi NFC:tä hyödyntäen. 5. luvussa punotaan yhteen ajatukset, joita projekti herätti ja minkälaista potentiaalia NFC:llä on pelimarkkinoilla. Lopuksi käydään läpi projektissa tekemäni työ.

NFC on yleisesti vielä melko tuntematon tekniikka, vaikka sen käyttämät tiedonsiirtostandardit kehitettiin jo vuonna 2003 (Penttilä 2014, 73). Sen läpilyöntiä on teknologiapiireissä odoteltukin kymmenisen vuotta, mutta sovelluksia alustalle mobiilimaailmassa ei ole juuri tullut erilaisten maksupalveluiden lisäksi.

Pelaajien määrä maailmassa on kasvavassa määrin jatkuvasti, ja tällä hetkellä heitä on noin 1,2 miljardia (Takahashi 2014). Näistä 53 % pelaa mobiilipelejä (Grubb 2014). Mobiilipelaajien iät vaihtelevat vauvoista vaareihin ja keski-ikä on tällä hetkellä noin 27,7 vuotta (Campbell 2014).

Suomalaiset ovat olleet edelläkävijöitä suosittujen mobiilipelien luomisessa, Rovion Angry Birdsin ja Supercellin Clash of Clansin hallitessa latauslistoja ympäri maailman. Käsittämättömän suosion saaneilla peleillä on pelaajia useita kymmeniä miljoonia ja pelien tuottoja laskiessa liikutaan samoilla tai moninkertaisilla luvuilla.

Konsolipeleissä NFC:tä on hyödynnetty useissa toimivissa ratkaisuissa. Skylanders ja Disney's Infinity ovat varmasti tuttuja nimikkeitä lapsiperheellisille henkilöille, mutta harvat ovat jääneet miettimään teknologiaa näiden ihmeellisten lelujen takana.

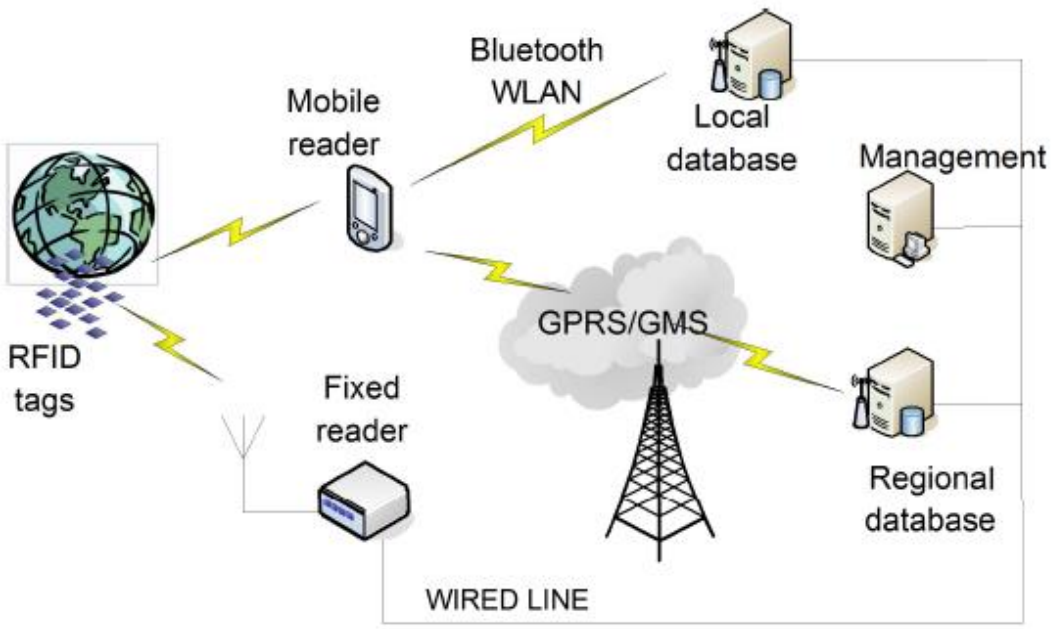
Mobiilipeleihin NFC-teknologiaa ei ole vielä valjastettu menestyksekkäästi, vaikka NFC-ominaisuudet löytyvätkin useista puhelimista ja täten sen käyttäminen mobiilipeleissä olisi melko looginen vaihtoehto.

Suunniteltaessa pelejä mobiilikäyttöön, voidaan ottaa huomioon puhelinten mahdollistama intuitio NFC-teknologian hyödyntämisessä – erillistä lukijaa ei tarvita ja puhelimen ollen yleensä kädessä pelitilanteessa voidaan NFC-sirun sisältämä esine lukea halutulla hetkellä. Tällöin mahdollisuudet NFC-esineiden käyttömahdollisuudet kasvavat huomattavasti ja voidaan väittää että vain kehittäjien mielikuvitus on rajana.

# 2 TEKNOLOGIOITA JA KEHITYSTYÖKALUJA

## 2.1 RFID ja NFC

NFC perustuu teknologiaan nimeltä RFID, Radio Frequency Identification, radiotaajuinen tunnistus. Teknologia perustuu siihen että erityisellä lukijalaitteella luetaan tai kirjoitetaan tietoa eräänlaisiin antennilla varustettuihin mikrosiruihin, tageihin. Tagi ei lähetä itsestään mitään signaalia vaan vastaa vain lukijan kyselyyn, eli reader talks first. Lisäksi tagi ”voi yksinkertaisimmillaan toimia tunnistekoodin sisältävänä saattomuistina ja monimutkaisemmillaan radiona, jolla on oma energianlähde, korkea salaustaso, ja johon on liitetty useita antureita.” (Penttilä 2014, 6.) RFID-järjestelmään kuuluu lisäksi tietojärjestelmä, johon lukija vertaa lukemaansa tietoa.




KUVA 1. RFID-järjestelmä (Penttilä 2014, 5)

Tagin mikrosiru sisältää rajapinnan antennille, tehon käsittelyn/teholähteen, analogia- ja digitaalilohkot, sekä muistin. Tehonsa tagi voi saada kolmella eri tavalla. Passiivinen tagi saa kaiken tehon lukijan sähkömagneettisesta aallosta tai magneettikentästä, aktiivisella on sisäinen teholähde ja semipassiivisella on sisäinen teholähde energian saantiin, mutta se saa viestintäenergiansa lukijan kentästä.

Tageja on usean muotoisia ja niiden koko vaihtelee kynnen kokoisista (Kuva 2) noin 20cm*30cm:iin. Tageja luettaessa lukijalaite lähettää moduloimatonta sähkömagneettista aaltoa tai magneettikenttää, mikä herättää tagin, jolloin se vastaanottaa ja prosessoi lukijan käskyn. Vastatessaan lukijalle tagi moduloi, perustuen tagin etupään transistorikytkentään, lukijan lähettämää sähkömagneettista aaltoa tai magneettikenttää ja se palautuu lukijalle takaisinheijastuksella. Aktiivinen tagi sisältää radiolähettimen.


KUVA 2. NFC-tageja (Kuva: Emma Valkki 2014)

Tagien muisti voi olla pelkästään luettavaa tai kerran ohjelmoitavaa, jolloin muutettava tieto sijaitsee palvelimella, tai ohjelmoitavaa jolloin tagin tietoa voidaan muokata n kertaa riippuen käytettävästä muistityypistä. Tageja pystyy ohjelmoimaan NFC-laitteilla, tai suoraan tuotannossa. NFC-laitteilla ohjelmointi toimii käytännössä samalla tavalla kuin lukeminenkin. Ensin tarkoitukseen sopivalla ohjelmistolla luodaan haluttu koodaus tagille, joka voi olla mm. tekstiä, yhteystieto, http-linkki tai sijainti. Tämän jälkeen tagi tuodaan laitteen lähelle ja laite kirjoittaa informaation tagille. Tageja on myös mahdollista tilata suoraan valmistajalta halutulla koodauksella (RapidNFC).

NFC-laite voi toimia RFID-lukijana sekä tagina. Tarkemmin sanottuna:

> NFC on lyhyen kantaman radiolaitejärjestelmä, joka mahdollistaa:
> RFID tunnistuksen perustuen ISO 18092 ja ISO 14443 standardeihin
> Kaksisuuntaisen linkin kahden ”lukijalaitteen” välillä.

Tällä hetkellä NFC:n pääsovellutukset matkapuhelimissa ovat sähköinen maksaminen, tiedoston jakaminen, eri laitteiden parittaminen Bluetooth-yhteyttä varten. Kun ennen

Bluetooth-yhteys muodostettiin etsimällä laitteet toistensa Bluetooth-verkoista ja sen jälkeen vaihtamalla laitteiden tunnusavaimia, voidaan se nyt muodostaa vain koskettamalla laitteita toisillaan. Tiedonsiirtonopeus NFC:llä on 106/212/424 kbit/s. NFC-tagi pystyy tallentamaan tietoa pääasiassa 64–924 tavua riippuen sen mallista (Taulukko 1), mutta markkinoilta löytyy myös tageja, jotka yltävät 4096 tavun muistin määrään. Muistin määrä 1 viittaa ennalta kirjoitettuun ja 2 viittaa uudelleenkirjoitettavaan tagiin.

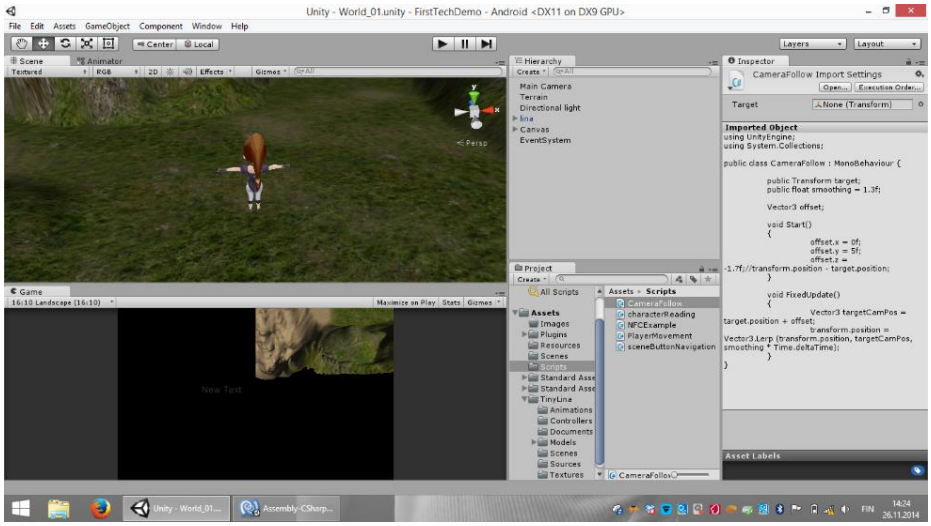
TAULUKKO 1. NFC tagien muistin määrä mallista riippuen (RapidNFC 2014)

| Malli | **Ultralight** | **NTAG203** | **NTAG210** | **NTAG213** | **NTAG215** | **NTAG216** | |
|---|---|---|---|---|---|---|---|
| Muistin määrä 1 | 64 | 168 | 80 | 180 | 540 | 924 | tavua |
| Muistin määrä 2 | 48 | 144 | 48 | 144 | 504 | 888 | tavua |

## 2.2 Unity

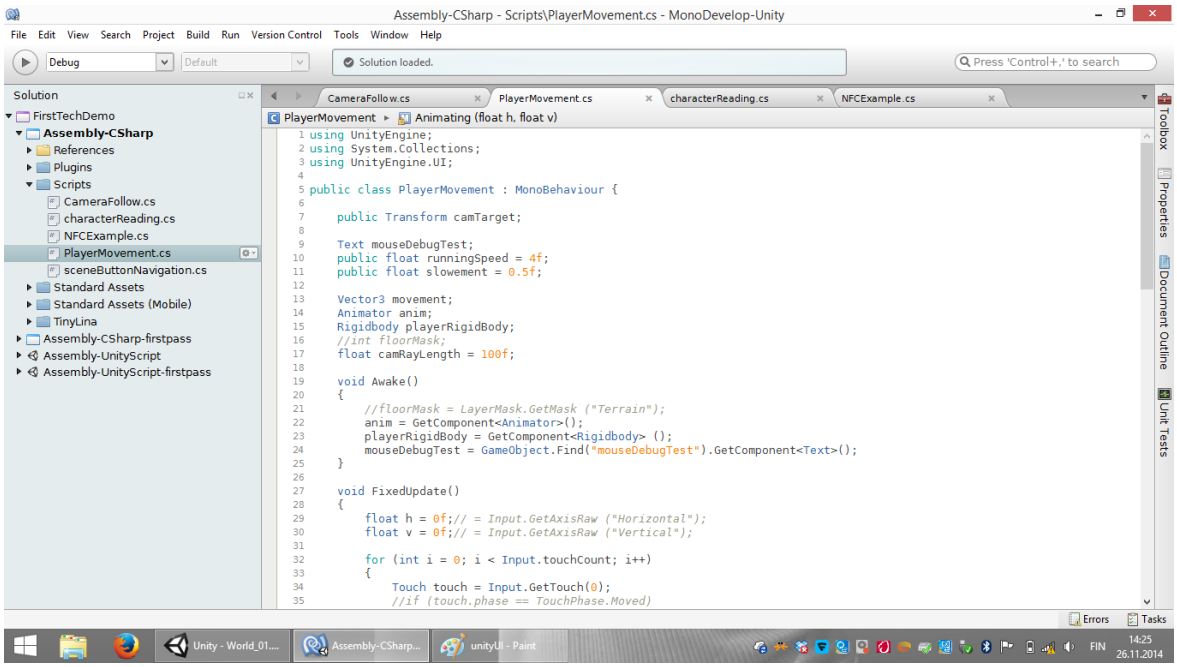
Unity on Unity Technologiesin kehittämä pelimoottori ja ohjelmointiympäristö, jonka avulla on mahdollista kehittää pelejän Web-selaimelle, PC, Mac, Linux -alustoille, iOS, Android, Blackberry, Windows Phone 8-puhelinkäyttöliittymille, Xbox 360, Xbox One, Playstation 3, Playstation Vita ja Playstation 4 -laitteille.

Käytännössä Unityllä pelien ohjelmointi toimii käyttämällä monipuolisia drag&drop-työkaluja, joilla pystytään hallitsemaan pelin ominaisuuksien riippuvuuksia toisistaan. Huomattavin osa kehitysikkunassa on vasemmasta yläosasta (Kuva 3) löytyvä Scene-eli kohtausnäkymä, jossa tiettyyn osioon pelissä, yleensä taso tai käyttöliittymäikkuna, voidaan lisätä haluttuja osia. Scene-ikkunaa on mahdollista hallita 2D- ja 3D-ympäristöissä, jolloin pystytään helposti tarkkailemaan pelien osien sijaintia toisiinsa nähden ja muuttamaan niiden ominaisuuksia.



KUVA 3. Unityn käyttöliittymä

Peleihin voidaan luoda myös Javascript tai C# -kielillä erilaisia scriptejä (Kuva 4), joilla pystytään vaikuttamaan pelin osien riippuvuuksiin ja ominaisuuksiin monipuolisemmin. Tällä pystytään vaikuttamaan vaikka pelin valaistuksen muutokseen riippuen kellon ajasta tai kameran sijaintiin riippuen hahmosta.



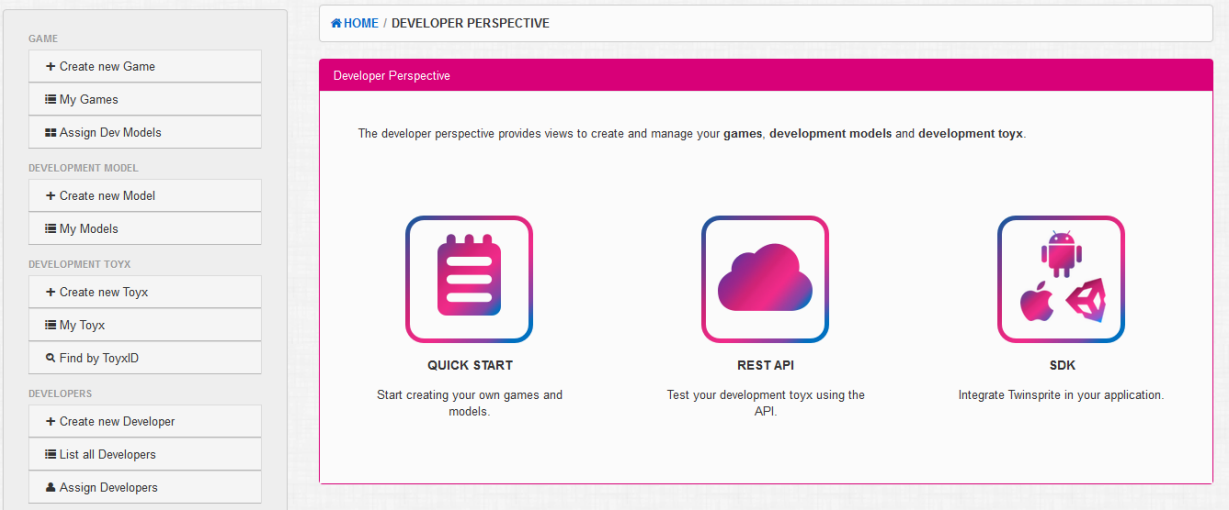
KUVA 4. Unityn script-työkalu

## 2.3 Twinspriten NFC plugin Unityyn

Twinsprite tarjoaa työkaluja NFC:n hyödyntämiseen Unityn päällä. Heidän sivuiltaan löytyvät ohjeet Android-laitteen NFC-lukijan hallitsemiseen. Lisäksi he tarjoavat

mahdollisuuden tallentaa peliin halutun ominaisuuden tietoja tietokantaansa ja hakea ne 40 merkkiä pitkän ToyxID -tunnisteen perusteella.

Käytännössä tämä tapahtuu luomalla erilaisilla parametreilla olioita heidän sivuillaan, joita voi monistaa, jolloin yksittäinen olio sidotaan tiettyyn ToyxID:hen (Kuva 5). Tämä monipuolistaa NFC-tagien potentiaalin, kun esineisiin sidottu tieto haetaan tietokannasta ja tageihin voidaan tallentaa vain haluttu tunniste, jolloin tagiin tallennetun datan määrä, yleensä 144 tavua, ei toimi esteenä ominaisuuksien monipuolisuudelle.




KUVA 5. Twinsprite kehittäjäportaali

## 2.4 NFC MOBIILILAITTEET

Tällä hetkellä yhdeksän kymmenestä suurimmasta mobiililaitevalmistajasta tukee NFC:tä laitteissaan, pois lukien Apple, joka ei tällä hetkellä tue NFC-teknologiaa, mutta IPhone 6:n eri versioissa on mahdollisuus ominaisuuden käyttöönottoon tulevaisuudessa. Täten iOS-käyttöliittymän NFC-ominaisuuksiin ei perehdytä tässä kirjoituksessa. Lisäksi Blackberry tukee NFC:tä melko voimakkaasti ja Unitylla on mahdollista kehittää pelejä myös tälle alustalle, mutta johtuen käyttöliittymän pienestä markkinaosuudesta, alle 1,5 % Euroopassa (bgr.com, 2014), sitä ei käsitellä tässä kirjoituksessa.

### 2.4.1 Windows Phone 8.1

Nokia on ollut edelläkävijä NFC-laitteissa, kun se oli mukana perustamassa 2004 NFC Forumia – järjestöä, jonka tavoitteena on kehittää ratkaisuja hyödyntäen NFC:tä. Ominaisuus löytyykin suurimmasta osasta heidän laitteistaan (Liite 2).

Twinsprite ei vielä tarjoa työkaluja Windows Phone -laitteille NFC-lukijan hallintaan Unity-projektissa.

Windows Phone –laitteille löytyy useita sovelluksia tagien lukemiseen ja kirjoittamiseen.

### 2.4.2 Android

Android-ympäristöstä valmistajia on useampia. Tämä vaikuttaa laitteiden ominaisuuksiin, kuten näyttöjen resoluutioon ja prosessorin suorituskykyyn, jotka voivat vaihdella huomattavasti, mikä on hyvä pitää mielessä pelin kehityksessä.

Twinsprite tarjoaa työkalun Android -laitteen NFC-lukijan hallintaan Unityssä. Käytännössä se toimii luomalla Android-luokasta olio Unityn script-työkalulla. Android-luokka ottaa puhelimen lukijan käyttöön ja estää muita ohjelmia aktivoitumasta, kun NFC-lukijan lähelle tuodaan tagi. Luettaessa tagi se tallentuu laitteen sisäiseen muistiin ja haluttaessa käyttää NFC-tagin sisältämää tietoa, tuodaan se Android-luokan oliosta *nfcvalue = pluginTutorialActivityJavaClass. CallStatic<string> ("getValue");* -käskyllä (Kuva 6).

Android-laitteille löytyy useita sovelluksia tagien kirjoittamiseen ja lukemiseen, sekä tiedonsiirtoon NFC-laitteiden välillä.

```
using UnityEngine;
using System.Collections;
public class NFCExample : MonoBehaviour
{
    public GUIText tag_output_text;
    AndroidJavaClass pluginTutorialActivityJavaClass;

    void Start ()
    {
        AndroidJNI.AttachCurrentThread ();
        pluginTutorialActivityJavaClass = new AndroidJavaClass ("com.twinsprite.nfcplugin.NFCPluginTest");
    }

    void Update ()
    {
        string value = pluginTutorialActivityJavaClass.CallStatic<string> ("getValue");
        tag_output_text.text = "Value:\n" + value;
    }
}
```

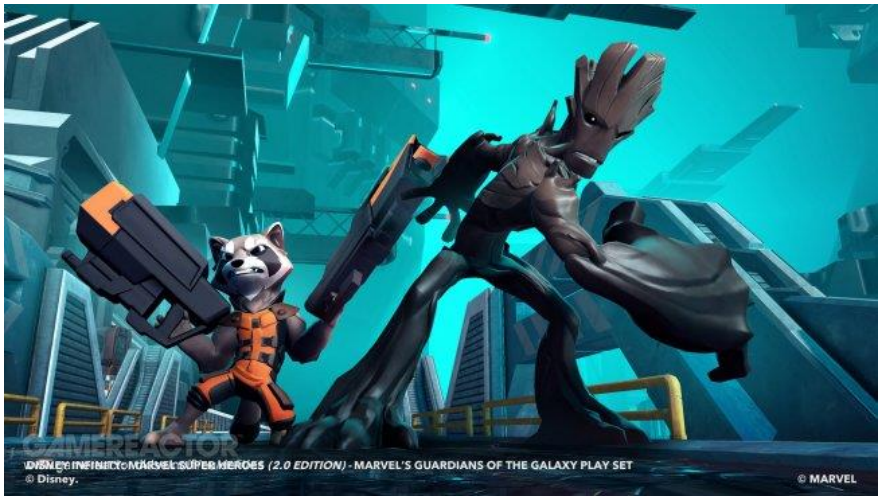
KUVA 6. Esimerkki Unityn Script-työkalusta haettaessa NFC-arvo

# 3 PELIT JA NFC

## 3.1 Konsolipelit ja NFC

Konsolipeleissä menestyneimmät tuotteet, jotka hyödyntävät NFC:tä ovat Disneyn Infinity ja Activisionin Skylanders.

Disneyn Infinity luottaa pääasiassa Disneyn omien tuotemerkkien voimaan ja figuurit toimivat siten, että ne avaavat peliin näköisiänsä hahmoja.




KUVA 7. Pelikuvaa Disney Infinity 2.0: Marvel Super Heroes –pelistä (GameReactor)

Skylandersissa NFC-toiminnallisuudet ovat pidemmälle kehitettyjä. Alkuperäinen peli toimii, kuten Disneyn Infinity, eli haluttu hahmo ladataan peliin sen näköisfiguurista. Myöhemmin lisäosat ovat tuoneet lisäominaisuuksia NFC:n hyödyntämisessä. Swapforce-lisäosa mahdollisti hahmojen rakentamisen kahdesta eri palasta, jolloin ne saavat peliin eri ominaisuuksia. Trap Team:ssä taas pelaaja pystyy vangitsemaan vihollisia pelin sisältä erilaisiin kristalleihin ja pelaamaan niillä jälkeenpäin.

KUVA 8. Pelikuvaa Skylanders: Swap Force –pelistä (Skylanders)

Nintendo julkaisi 21.11.2014 oman NFC-tuotteensa nimeltä Amiibo. Amiibot ovat Nintendon pelisarjojen näköishahmoja (Kuva 9). Amiibo mahdollistaa datan siirtämisen peliin ja pelistä pois, jolloin pelaajat voivat siirtää esimerkiksi pelihahmojaan pelien välillä. Tällä hetkellä ainoastaan Nintendon Super Smash Bros. –peli mahdollistaa tietojen tallentamisen Amiiboon.

KUVA 9. Nintendon Amiibo-figuureita (Kotaku)

Lisäksi Koreassa on hyödynnetty NFC:tä opetuspelissä nimeltä Badanamu.

### 3.2 Pelin suunnittelu

Supercellin Clash of Clansin pääsuunnittelija Lasse Louhento antaa 5 neuvoa mobiilipelien kehitykseen.

Pelin kuuluisi vedota niin kutsuttuihin tosipelaajiin ja satunnaisiin pelaajiin. Tämä vaikuttaa pelin graafiseen ulkonäköön, kun tosipelaajat haluavat yleensä tummia sävyjä, satunnaiset pelaajat pitävät rennosta, hieman lapsellisesta tyylistä. Pelimekaniikoiltaan tosipelaajat haluavat haastetta pelin sisällä, jolloin omien taitojen täytyy kehittyä päästäkseen vaikka seuraavalle tasolle tai vaihtoehtoisesti kilpailua muita pelaajia vastaan. Satunnaiset pelaajat haluavat kevyttä pelattavaa, jonka avulla voi kuluttaa aikaa rennosti. Pelin sosiaalisen näkökulman täytyisi luoda edellytykset molemmille pelaajatyypeille.

Peliä suunniteltaessa ja valmistaessa täytyy kehittäjän mielessä olla tuotteen tavoite alusta asti. Keskittyykö peli monin- vai yksinpeliin. Pelin käytettävyyden täytyy tuntua hyvältä sillä alustalla, mille se julkaistaan ja pelin testaukseen täytyy käyttää tarpeeksi aikaa, jotta löydetään asioita, jotka eivät tunnu hyvältä.

Pelin tutoriaalia, eli yleensä alussa tapahtuva opetusosio, jossa käydään läpi pelimekaanikoita, ei saa liioitella. Jos peli on hyvin suunniteltu toimii oletetulla tavalla, pelaajat oppivat pelatessaan.

Terve kilpailu vahvistaa yritystä, kun valmistaessa peliä, on jotakin mitä vasten vertailla omaa tuotostaan ja edistymistään. Kilpailu voi tapahtua vaikkapa yrityksen sisällä, mikäli sen sisällä eri ryhmät työskentelevät eri tuotteiden parissa.

Pelaajia ei saa häkellyttää suurella sisällön määrällä. Jos peliä kehitetään jatkuvasti ja siihen lisätään ominaisuuksia, voidaan osa päivityksistä avata pelaajille vasta tietyn peliajan jälkeen. Uusien ominaisuuksien kuuluu viihdyttää sekä tosi-, että satunnaisia pelaajia, jolloin ne eivät saa olla liian monimutkaisia, mutta ne voivat lisätä piilossa olevaa syvyyttä peliin (Rose 2014).


KUVA 10. Pelikuvaa Clash of Clans –pelistä (Supercell)

NFC-pelin suunnittelu kannattaa aloittaa pelityypin valinnasta. Olipa kyseessä seikkailu-, ongelmanratkonta-, strategiapeli, täytyy miettiä, mitä lisäarvoa NFC:n lisääminen tuo peliin. Monissa tapauksissa tähän voi olla vaikea vastata. Voidaan kuitenkin olettaa, että NFC mahdollistaa pelin objektien helpon vaihtokaupan pelaajien kesken, mikä lisää sosiaalista arvoa peliin. Sosiaalista arvoa lisää myös NFC:n mahdollistama laitteiden nopea paritus vaikkapa moninpeliä varten.

Varmin lähestymistapa NFC:n hyödyntämiseen on hahmon tallentaminen tagiin, eli mitä tähän mennessä menestyneet konsolipelit ovat käyttäneet. Sen sijaan että tallennettaisiin vain geneerisiä hahmoja ilman muita kuin graafisia ominaisuuksia, voitaisiin tageihin sitoa ominaisuuspisteitä kuten hyökkäys- tai kokemusarvoja, jolloin hahmot saavat lisäarvoa, kun ne kehittyvät pelaajan pelatessa.

Hahmon tallentamisen jälkeen potentiaalisia sovellutuksia ovat esineet pelin sisällä. Tässä vaiheessa täytyy ottaa huomioon kuitenkin NFC-tagien hinnat ja esineiden potentiaalinen käyttömäärä. Olisi pelaajan kannalta epämiellyttävää, mikäli joitain tiettyjä esineitä tarvitsee satoja, minkä vuoksi olisi kannattavaa sitoa tageihin esineitä, joilla on arvoa yksittäin ja pidemmällä matkalla, eli esimerkiksi varusteita. Varusteiden lisääminen tagien avulla mahdollistaa kuitenkin ongelman, jonka vuoksi esineet olisi hyvin tärkeä tasapainottaa tarkasti, jotta vältytään tällä hetkellä yleisesti karsastetulta pelin rahoitusmuodolta, jossa peliin on mahdollista ostaa etulyöntiasema rahalla, niin kutsuttu pay-to-win.

Peli, jossa kerätään kortteja, on selvästi loogisimpia lähestymistapoja NFC:n hyödyntämiseen peleissä. Puhelin voi toimia pelin alustana, jolloin pelaajan itse ei tarvitse vaikkapa laskea pisteitä. Tageja myydään myös suoraan korttimuodossa. Korttipeliä suunniteltaessa tärkeimmät ratkaisut tulevat tagin lukemisen kohdalla. Tagin lukemiseen menee n. 1-3 sekuntia, ja siinä saattaa ajoittain tapahtua virheitä, esimerkiksi silloin jos tagi siirretään liian aikaisin pois laitteen lukuetäisyydeltä, joten useamman kortin lukeminen peräkkäin voi olla vaivaannuttavaa pelaajan näkökulmasta.

NFC:tä voi myös hyödyntää niiden pienen koon ja uudelleenkäytettävyyden vuoksi erilaisissa mainoshankkeissa, kun tagien avulla voidaan jakaa vaikka jokin harvinainen esine, tai hahmo pelissä. Tagin pienen koon vuoksi niitä voitaisiin upottaa vaikkapa pelin mainosjulisteisiin, joista pelaajat voisivat avata itselleen jotain erityistä peliin.

## 4 CASE: Positioning NFC technology into mobile gaming experience

Olin mukana projektissa, jossa tehtävänä on luoda demo mobiilipelistä, jossa hyödynnetään NFC-teknologiaa. Projektiin kuului lisäkseni 3 muuta henkilöä.

Aluksi kävimme läpi miten teknologiaa on aiemmin hyödynnetty peliteollisuudessa. Tämän pohjalta lähdimme suunnittelemaan peliä asiakkaan toiveiden ja omien kokemuksiemme mukaan.

Jaoimme tehtävät henkilöittäin kunkin osaamisalueen perusteella, jolloin vastasin teknologisesta puolesta pelissä, käytännössä ohjelmoinnista. Pelin suunnitteluun osallistuivat aluksi kaikki ryhmän jäsenet, mutta päästyämme toteutusvaiheeseen, suunnittelusta vastasi yksi ryhmän jäsenistä, muiden mielipiteitä kuunnellen.

Pelin suunnittelun aloitimme asiakkaan meille antamien ideoiden mukaisesti. Valittuamme pelin tyypin, aloimme suunnittelemaan sen kohdeasiakaskuntaa, joksi valitsimme 5-12-vuotiaat, koska uskoimme, että tälle asiakaskunnalle olisi helpoin esitellä uutta teknologiaa pelimarkkinoilla ja olivathan jo menestyneet konsolipelitkin kohdistettu sen ikäisille.

Tämän jälkeen suunnittelimme useita eri käyttötarkoituksia NFC-tageille, mutta päädyimme aluksi vain tallentamaan ja lukemaan hahmoja ja niiden tilastoja tagilta. Pelin jatkokehityksessä tultaisiin peliin lisäämään muitakin NFC-ominaisuuksia.

Tarkoitus olisi, että hahmoja kerättäisiin pelin sisältä ja niillä taisteltaisiin muita pelaajia tai pelin sisältämiä olentoja vastaan. Hahmon suunnittelimme siten, että se pohjautui osittain Tamagotcheihin ja Pokemoneihin, jolloin siitä täytyisi huolehtia hieman, jolloin pelaaja tuntisi siihen osittain kiintymystä. Tavoitteena oli myös luoda erilaisia hahmoja, joiden keräämisestä pelaaja hyötyisi hieman, mutta ei peliä rikkovalla tavalla.

Saimme asiakkaalta testausta varten Google Nexus 5 –laitteen, joka toimii Android-käyttöliittymällä.

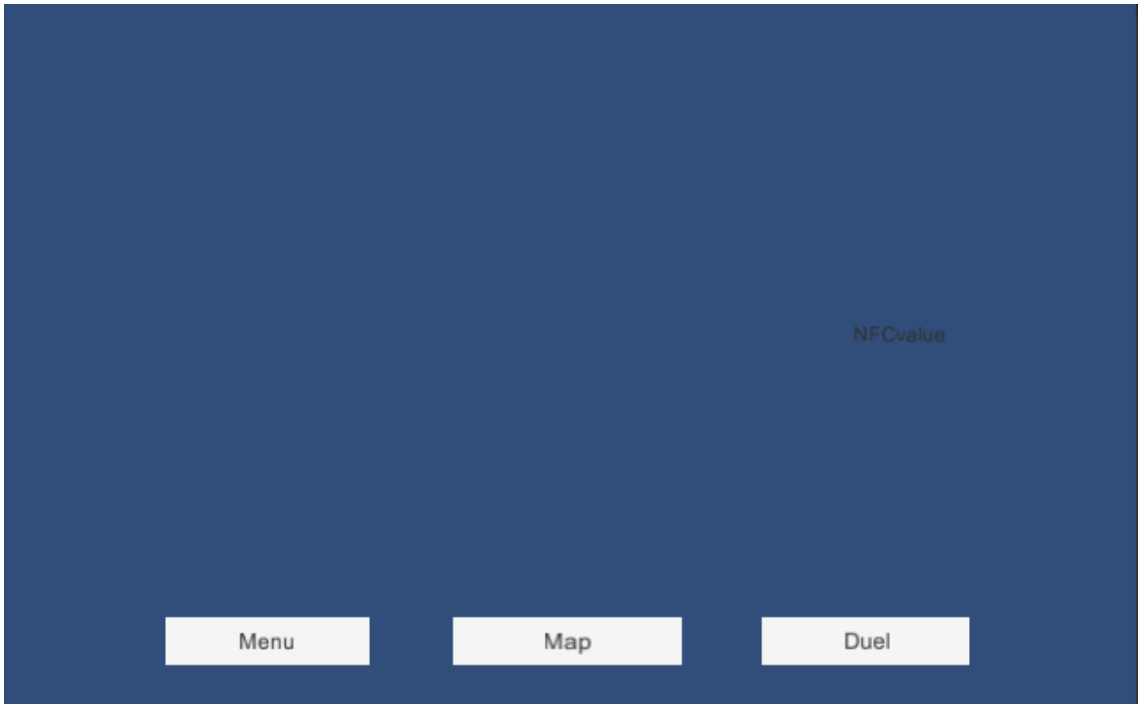
Aluksi testasimme Twinspriten pilvipalvelua, luomalla sinne geneerisiä olentoja, joilla oli ominaisuuksina nimi ja ikä. Haimme näitä tietoja käyttäen Unityn script-työkalua (Kuva 11).

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;
using TwinSpriteSDK;

public class TwinSpriteScript : MonoBehaviour {

    string API_KEY = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX";
    string SECRET_KEY = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX";

    private List<string> attributes;

    private int age;

    private Toyx toyx;
    private string toyxId = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX";

    // Use this for initialization
    void Start () {
        TwinSprite.initialize(API_KEY, SECRET_KEY);
        toyx = new Toyx(toyxId);

        toyx.CreateSessionInBackground(
        delegate(TwinSpriteError error)
        {
            if (error != null)
            {
                Debug.Log("Error: " + error.message);
            }
            else
            {
                Debug.Log("Created session!!!");
            }
        });

        toyx.FetchInBackground(
            delegate(TwinSpriteError error)
            {
            if (error != null)
            {
                Debug.Log("Error: " + error.message);
            }
            else
            {
                Debug.Log("Fetched: " + toyx);
                age = toyx.GetInt ("Age");
                attributes = toyx.GetAttrNames ();
            }
        });

        Debug.Log (age);


    }
```

KUVA 11. Twinsprite-palvelun testausta

Twinspriten palvelun käyttäminen oli melko yksinkertaista ja halutuilla tunnusavaimilla pystyimme hakemaan tietoja heidän palvelimiltaan.

Tämän jälkeen loimme yksinkertaisen käyttöliittymän (Kuva 12), jossa pystyimme testaamaan NFC:n lukemista laitteella ja Unityn uusimpaan versioon muokattuja käyttöliittymätyökaluja.




KUVA 12. Ensimmäinen käyttöliittymä

Käyttöliittymän luominen Unityssa on melko yksinkertaista. Se voidaan sijoittaa kameraan nähden haluttuun paikkaan, joko suoraan pelimaailman eteen, tai vaihtoehtoisesti sen sisälle.

Luotaessa käyttöliittymää, muodostetaan peliin eräänlainen kangas, johon voidaan lisätä haluttuja ominaisuuksia, tässä tapauksessa 3 nappia, kuvake, joka vaihtui NFC:n arvon perusteella ja tekstikenttä, johon luettu NFC:n arvo tuotiin script-työkalulla (Kuva 13).

NFC-lukeminen toimi erinomaisesti, mutta jostain syystä Google Nexus 5 –laitteen NFC-lukija kaatui silloin tällöin lukemisen aikana, jonka jälkeen se täytyi resetoida, jotta sitä voitiin käyttää uudelleen. Ilman muita testilaitteita emme osaa sanoa, mistä vika johtuu.

```
using UnityEngine;
using UnityEngine.UI;
using System.Collections;

public class characterReading : MonoBehaviour {
    public Sprite babySprite;
    public Sprite adultSprite;
    public Sprite treshold;
    public string nfcvalue = "";
    public string character = "";
    Text nfcUItext;
    AndroidJavaClass pluginTutorialActivityJavaClass;

    //NFCExample reader = new NFCExample();
    // Use this for initialization


    void Start () {
        //babySprite = Resources.Load("baby1", typeof(Sprite)) as Sprite;
        //adultSprite = Resources.Load("baby2", typeof(Sprite)) as Sprite;
        //treshold = Resources.Load("baby3", typeof(Sprite)) as Sprite;
        AndroidJNI.AttachCurrentThread ();
        pluginTutorialActivityJavaClass = new AndroidJavaClass ("com.twinsprite.nfcplugin.NFCPluginTest");
        //reader.StartReader ();
        nfcUItext = GameObject.Find("NFCText").GetComponent<Text>();
    }

    public void changeCharacterSprite()
    {
        nfcUItext.text = nfcvalue.ToString();
        if (nfcvalue == "baby")
        {
            GameObject.Find("character_image").GetComponent<SpriteRenderer> ().sprite = babySprite;
        }
        else if (nfcvalue == "baby2")
        {
            GameObject.Find("character_image").GetComponent<SpriteRenderer> ().sprite = adultSprite;
        }
        else if (nfcvalue == "baby3")
        {
            GameObject.Find("character_image").GetComponent<SpriteRenderer> ().sprite = treshold;
        }
        else
        {
            GameObject.Find("character_image").GetComponent<SpriteRenderer> ().sprite = null;
        }
    }

    // Update is called once per frame
    void Update ()
    {
        //character = reader.readTag();
        //characterText.text = character;
        nfcvalue = pluginTutorialActivityJavaClass.CallStatic<string> ("getValue");
        changeCharacterSprite ();
        //changeCharacterSprite ();
    }
}
```

KUVA 13. NFC:n lukemisen testausta

NFC:n testauksen jälkeen aloin luomaan ensimmäistä testimaailmaa (Kuva 14). Tässä vaiheessa suunnitelmissa oli vielä 3D-ympäristön käyttö. Unity tarjoaa monipuoliset työkalut oman ympäristön luomiseen. Käytännössä se toimii siten, että pelimaailmaan luodaan halutun kokoinen taso, jonka pintaa voidaan muokata erilaisilla työkaluilla, nostamalla ja laskemalla haluttuja kohtia ja piirtämällä pintaan erilaisia tekstuureja.


KUVA 14. Ensimmäinen 3D-ympäristö

Kun olin luonut ensimmäisen testiympäristön, lisäsin peliin ensimmäisen pelattavan hahmon, jonka avulla testasin liikkumista 3D-ympäristössä kosketusnäytön avulla. Tarkoituksena oli, että hahmo liikkuisi pelaajan ruudulla osoittamaan suuntaan. Käytännössä se tapahtui siten, että pelaajan kosketus mitattiin ruudulta jolloin siitä saatiin X- ja Y-koordinaatit ja niiden suuruuden perusteella hahmon nopeus muuttui riippuen etäisyydestä ruudun keskikohtaan.

Tässä vaiheessa päätimme kuitenkin siirtyä 2D-ympäristöön, johtuen rajoitetusta ajasta ja siitä että projektin jäseniin kuului vain yksi henkilö, jolla oli kokemusta grafiikan luomisesta.

Tämän jälkeen ryhdyin suunnittelemaan ensimmäistä pelikenttää ja animaatioita ja niiden logiikkaa hahmolle, joissa hyödynnettiin grafiikoita, jotka oli luotu tähän mennessä.


KUVA 15. 2D-ympäristön luonti

2D-ympäristön luonnissa Unityssä tärkeimmät työkalut ovat grafiikkana käytettävien spritejen suhtautuminen toisiinsa erilaisten tasojen avulla. Eri spritejä voidaan asettaa kameraan nähden toistensa eteen ja taakse ja ne voidaan asettaa vaikuttamaan hahmoon, jolloin tässä tapauksessa voidaan luoda vaikkapa käveltävä taso.

Tässä projektissa päädyin käyttämään 5 erilaista tasoa, hahmo, taustalavasteet, keskilavasteet, etulavasteet ja taso, jossa voidaan liikkua. Tällöin ympäristöstä saadaan luotua monipuolisen ja täyteläisen näköisen, kun osa grafiikoista jää pelihahmon eteen tai taakse.

Luotuani ensimmäisen pelikentän aloin hiomaan ohjausta sopivaksi kosketusnäytölle ja testaamaan että hahmon animaatiot toimivat oikein.

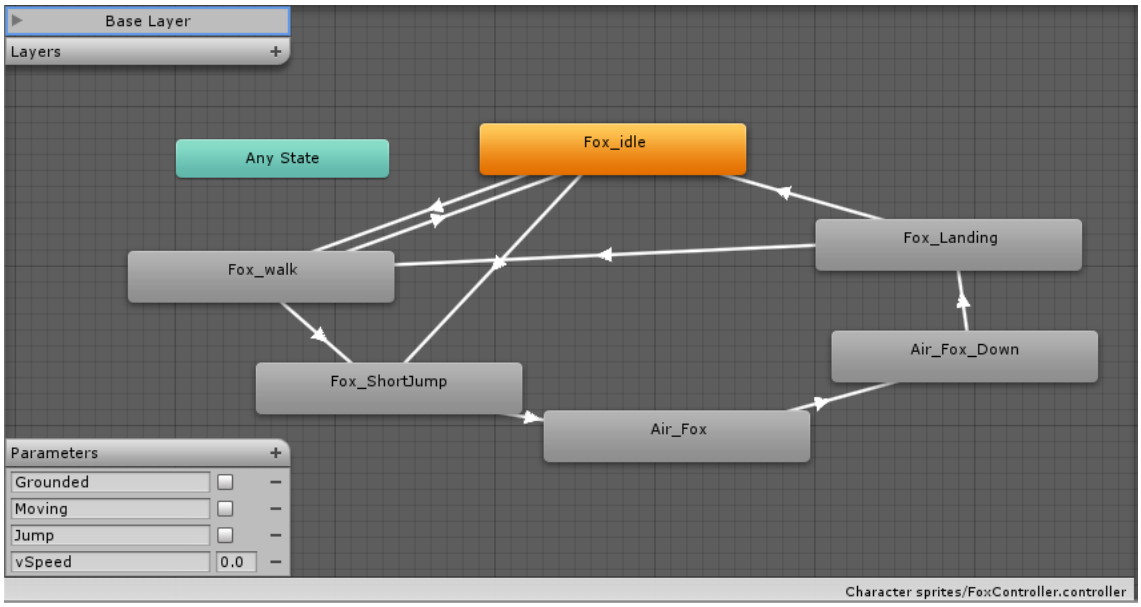
Päädyin käyttämään tässä vaiheessa käyttöliittymälle piirrettäviä nappeja hahmon ohjauksessa, mutta tulevaisuuden suunnitelmissa olisi mahdollista myös ohjata hahmoa suoraan kosketusnäytön avulla. Ohjaus toimi siten, että kun nappia painetaan ruudulla, kutsutaan scriptissä (Kuva 16) haluttua metodia, eli MyMoveLeft, MyMoveRight tai Jump.

```
using UnityEngine;
using System.Collections;
using UnityEngine.UI;
using UnityEngine.EventSystems;

public class CharacterMovement : MonoBehaviour {
    public float maxSpeed = 6f;
    bool facingRight = true;

    Animator anim;

    bool grounded = false;
    public Transform groundCheck;
    float groundRadius = 0.2f;
    public LayerMask whatIsGround;
    public float jumpForce = 300f;

    bool movingLeft = false;
    bool movingRight = false;

    // Use this for initialization
    void Start () {
        anim = GetComponent<Animator> ();
    }

    // Update is called once per frame
    void FixedUpdate () {
        grounded = Physics2D.OverlapCircle (groundCheck.position, groundRadius, whatIsGround);

        anim.SetBool ("Grounded", grounded);
        anim.SetFloat ("vSpeed", rigidbody2D.velocity.y);

        float move;

        if (movingLeft) {
            move = -1f;

        }
        else if (movingRight) {
            move = 1f;
        }
        else {
            move = Input.GetAxis ("Horizontal");
        }


        rigidbody2D.velocity = new Vector2 (move * maxSpeed, rigidbody2D.velocity.y);

        if (move != 0) {
            anim.SetBool ("Moving", true);
        }
        else {
            anim.SetBool ("Moving", false);
        }

        if (move > 0 && !facingRight) {
            Flip();
        }
        else if (move < 0 && facingRight) {
            Flip();
        }
    }
```

```
63
64      void Update()
65      {
66          if (grounded && Input.GetKeyDown (KeyCode.Space)) {
67              Jump ();
68          }
69      }
70
71      void Flip()
72      {
73          facingRight = !facingRight;
74          Vector3 theScale = transform.localScale;
75          theScale.x *= -1;
76          transform.localScale = theScale;
77      }
78
79      public void MyMoveLeft() {
80          movingLeft = true;
81      }
82
83      public void MyMoveRight () {
84          movingRight = true;
85      }
86
87      public void StopMoving()
88      {
89          movingLeft = false;
90          movingRight = false;
91      }
92
93      public void Jump()
94      {
95          if (grounded) {
96          anim.SetBool ("Grounded", false);
97          anim.SetTrigger ("Jump");
98          rigidbody2D.AddForce (new Vector2 (0, jumpForce));
99          }
00      }
01
02  }
```

KUVA 16. Hahmon liikkuminen 2D-ympäristössä

Unityssä hahmojen animaatioiden luonti tapahtuu animator-työkalulla luoduilla animaatiologiikoilla, jolloin eri animaatiopalikoista luodaan haluttuja yhteyksiä toisiinsa ja niiden välillä liikutaan riippuen halutuista parametreista.




KUVA 17. Hahmon animaatiologiikka

Oranssilla merkitty animaatiopalikka on oletuksena pelin alkaessa. Tässä tapauksessa Fox_idle:stä siirrytään Fox_walkiin, kun moving-parametri saa true arvon

liikkumisscriptissä (Kuva 16). Fox_walkista voidaan palata takaisin Fox_idleen, kun hahmo lakkaa liikkumasta, tai sitten voidaan siirtyä Fox_ShortJumpiin hahmon hypätessä. Tällöin käydään läpi hahmon hyppäämisanimaatio riippuen liikesuunnasta ylös ja alas ja sen jälkeen laskeutumisanimaation jälkeen palataan Fox_idleen tai Fox-walkiin, mikäli hahmo on vielä liikkeessä.

Tässä vaiheessa NFC-tageista oli mahdollista lukea tunnisteen perusteella eri hahmoja peliin, mutta ne olivat vielä staattisia kuvia. Lisäksi hahmoilla oli tiettyjä tilastoja, jotka tallentuvat laitteen sisälle, kuten nälkäisyys ja hyvinvointi, jotka muuttuvat ajan kuluessa.


KUVA 18. Pelikuvaa tässä vaiheessa kehitetystä versiosta

Asiakas oli erittäin tyytyväinen tähän mennessä aikaansaamaamme työhön.

Jatkossa kehitettäväksi jäi tähän mennessä tehdyn pelikentän laajentaminen ja NFC:n tarkempaan hyödyntämiseen tarvittavan logiikan luonti. Tällä hetkellä hahmojen kerääminen toimi napauttamalla kosketusnäytöllä näkyvää hahmoa, mutta tarkoituksena olisi, että tämä ominaisuus tapahtuisi lukemalla NFC-tagi laitteella.

Demon lopullisessa versiossa olisi tarkoitus olla useampi hahmo kerättäväksi ja pelattavien hahmojen tietojen tallentaminen Twinspriten pilvipalveluun tai suoraan NFC-tagille.

# 5 POHDINTA

Menestyvän pelin luominen vaatii toimivan idean ja ymmärrystä nykyajan trendeistä. Viime vuosina eniten myydyissä uusissa pelisarjoissa on ollut 2 huomattavaa ominaisuutta. Pelit ovat periaatteessa ilmaisia pelattavia, mutta niihin on mahdollista ostaa haluamansa määrä objekteja, niin kutsutuilla mikromaksuilla, jotka vaihtelevat huomattavasti pelien välillä. Ne voivat olla peliä helpottavia esineitä tai ominaisuuksia, esineitä jotka vaikuttavat vain hahmojen ulkonäköön, pelin sisäistä rahaa, tai poistaa aikarajoituksia pelin sisältä (Spence 2014).

Lisäksi peleihin kuuluu sosiaalinen näkökulma ja niissä esiintyy yleensä tietynlaista kilpailullisuutta. Tämä kuuluu suunnitella satunnaisten pelaajien ja tosipelaajien näkökulmasta. Satunnaiset pelaajat haluavat tuntea yhteisöllisyyttä. Tosipelaajat haluavat kilpailla. Vaikka suoranaisesti ei pelattaisi toisia pelaajia vastaan, voidaan tuloksia vertailla pelaajien kesken erilaisilla ranking-listoilla.

NFC tuo jotain uutta ja mielenkiintoista pelimaailmaan. Tässä työssä kuvailtujen sovellutuksien lisäksi on vielä vaikea arvioida NFC:n kokonaispotentiaalia mutta kunhan teknologia saa enemmän huomiota, tullaan tulevaisuudessa varmasti näkemään jotain, joka hämmästyttää ympäri maailman.

NFC-teknologialla on vielä pieniä ongelmia johtuen suorituskyvystä. Joskus tagien lukeminen ei onnistu, tai kuten esimerkkiprojektissa, laitteen lukija kaatui selittämättömästä syystä, mutta uskoisin että näihin ongelmiin löytyy ratkaisuja, kun teknologia yleistyy. Jos näin käy, niin teknologialla on oletettavasti loistava tulevaisuus pelimarkkinoilla.

Nykyaikana voidaan kuvitella NFC:n paikkaavan hyvin mikromaksurahoitusta, kun pelaajat nykyään maksavat vain datasta, voidaan nyt rahan arvolle tarjota jotain fyysistä vastinetta. Toisaalta edelleen täytyy miettiä tarkkaan mitä ominaisuuksia tageihin tallennetaan, sillä huonosti suunnitellut mikromaksut ovat yrityksen maineen kannalta tuhoisia (Spence 2014).

Sosiaalista lisäarvoa NFC tuo monin tavoin riippuen NFC-ominaisuuksien toteutuksesta. On mahdollista, että NFC-tagien vaihdosta voi olla hyötyä pelaajille, vaikkapa kerätessä hahmoja tai esineitä. Pelaajat voisivat pelata vastakkain käyttäen omia keräilyhahmojaan, jolloin yleensä hävinnyt osapuoli joutuisi luopumaan jostain NFC-tagistaan.

**LÄHTEET**


BGR.com. BlackBerry handsets are nearly extinct. Luettu 26.11.2014. http://bgr.com/2014/01/27/blackberry-market-share-collapse/

Forbes. Summing Up Mobile Gaming In 2013 With A Single Word... Freemium. Ewan Spence. Luettu 28.11.2014. http://www.forbes.com/sites/ewanspence/2013/12/28/summig-up-mobile-gaming-in-2013-with-a-single-word-freemium/

Gamasutra. Clash of Clans' 5 keys to success. Mike Rose. Luettu 3.12.2014. http://www.gamasutra.com/view/news/185406/Clash_of_Clans_5_keys_to_success.php#.USgPSFrwIZQ

Penttilä, K. 2014. RFID luennot kevat2014.pdf

Polygon, Average age of mobile gamer drops seven years as kids and teens grab smartphones. Colin Campbell. Luettu 3.12.2014. http://www.polygon.com/2014/11/7/7176029/average-age-of-mobile-gamer-drops-seven-years-as-kids-and-teens-grab

RapidNFC. NFC Enabled Phones And Tablets. Luettu 24.11.2014. http://rapidnfc.com/nfc_enabled_phones

RapidNFC. Which NFC Chip?. Luettu 20.11.2014. http://rapidnfc.com/which_nfc_chip

RapidNFC. White NFC Tags. NTAG213 Round 38mm. Luettu 23.11.2014. http://rapidnfc.com/item/321/white_nfc_tags_ntag213_round_38mm

Venturebeat, More than 1.2 billion people are playing games. Dean Takahashi. Luettu 1.12.2014. http://venturebeat.com/2013/11/25/more-than-1-2-billion-people-are-playing-games/

Venturebeat, Gaming advocacy group: The average gamer is 31, and most play on a console Jess Grubb. Luettu 3.12.2014. http://venturebeat.com/2014/04/29/gaming-advocacy-group-the-average-gamer-is-31-and-most-play-on-a-console/

GameReactor, Kuvia pelistä Disney Infinity 2.0: Marvel Super Heroes. Luettu 10.12.2014 http://www.gamereactor.fi/images/?productid=42884&id=1232194

Skylanders, SKYLANDERS – KOTI. Luettu 11.12.2014 http://www.skylanders.com/fi/

Kotaku, New Amiibo Figures Include Wind Waker Link, Mega Man & Sonic, Luettu 11.12.2014 http://kotaku.com/new-amiibo-figures-include-wind-waker-link-mega-man-1657159727

Supercell, Supercell, Luettu 16.12.2014
http://www.supercell.net/games/view/clash-of-clans

# LIITTEET

## Liite 1. Twinspriten Android-luokan Java koodi NFC-lukijan hallintaan Unityssä

```
package com.twinsprite.nfcplugin;

import java.nio.charset.Charset;
import java.util.Arrays;

import android.app.PendingIntent;
import android.content.Intent;
import android.content.IntentFilter;
import android.nfc.NdefMessage;
import android.nfc.NdefRecord;
import android.nfc.NfcAdapter;
import android.nfc.Tag;
import android.nfc.tech.NfcF;
import android.os.Bundle;
import android.os.Parcelable;
import android.util.Log;

import com.google.common.collect.BiMap;
import com.google.common.collect.ImmutableBiMap;
import com.google.common.primitives.Bytes;
import com.unity3d.player.UnityPlayerActivity;

public class NFCPluginTest extends UnityPlayerActivity {

    public static final String MIME_TEXT_PLAIN = "text/plain";

    private static final BiMap URI_PREFIX_MAP = ImmutableBiMap.builder().put((byte) 0x00, "")
            .put((byte) 0x01, "http://www.").put((byte) 0x02, "https://www.").put((byte) 0x03, "http://")
            .put((byte) 0x04, "https://").put((byte) 0x05, "tel:").put((byte) 0x06, "mailto:")
            .put((byte) 0x07, "ftp://anonymous:anonymous@").put((byte) 0x08, "ftp://ftp.").put((byte) 0x09, "ftps://")
            .put((byte) 0x0A, "sftp://").put((byte) 0x0B, "smb://").put((byte) 0x0C, "nfs://")
            .put((byte) 0x0D, "ftp://").put((byte) 0x0E, "dav://").put((byte) 0x0F, "news:")
            .put((byte) 0x10, "telnet://").put((byte) 0x11, "imap:").put((byte) 0x12, "rtsp://")
            .put((byte) 0x13, "urn:").put((byte) 0x14, "pop:").put((byte) 0x15, "sip:").put((byte) 0x16, "sips:")
            .put((byte) 0x17, "tftp:").put((byte) 0x18, "btspp://").put((byte) 0x19, "btl2cap://")
            .put((byte) 0x1A, "btgoep://").put((byte) 0x1B, "tcpobex://").put((byte) 0x1C, "irdaobex://")
            .put((byte) 0x1D, "file://").put((byte) 0x1E, "urn:epc:id:").put((byte) 0x1F, "urn:epc:tag:")
            .put((byte) 0x20, "urn:epc:pat:").put((byte) 0x21, "urn:epc:raw:").put((byte) 0x22, "urn:epc:")
            .put((byte) 0x23, "urn:nfc:").build();

    private NfcAdapter mNfcAdapter;

    private PendingIntent pendingIntent;

    private IntentFilter[] mIntentFilter;

    private static String value = "";

    private String[][] techListsArray;

    @Override

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    // Foreground Dispatch: 1. Creates a PendingIntent object so the Android
    // system can populate it with the details of the tag when it is
    // scanned.
    pendingIntent = PendingIntent.getActivity(NFCPluginTest.this, 0,
            new Intent(NFCPluginTest.this, getClass()).addFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP), 0);

    // Foreground Dispatch: 2. Declare intent filters to handle the intents
    // that you want to intercept
    mIntentFilter = new IntentFilter[] { new IntentFilter(NfcAdapter.ACTION_TAG_DISCOVERED) };

    // Foreground Dispatch: 3. Set up an array of tag technologies that your
    // application wants to handle.
    techListsArray = new String[][] { new String[] { NfcF.class.getName() } };

    mNfcAdapter = NfcAdapter.getDefaultAdapter(this);
    if (mNfcAdapter == null) {
        Log.e(NFCPluginTest.class.toString(), "This device doesn't support NFC.");
        finish();
        return;
    }

    if (!mNfcAdapter.isEnabled()) {
        Log.e(NFCPluginTest.class.toString(), "NFC is disabled.");
    } else {
        Log.i(NFCPluginTest.class.toString(), "NFC reader initialized.");
    }
}

@Override
public void onResume() {
    super.onResume();
    mNfcAdapter.enableForegroundDispatch(this, pendingIntent, mIntentFilter, techListsArray);
}

@Override
public void onPause() {
    super.onPause();
    mNfcAdapter.disableForegroundDispatch(this);
}

@Override
protected void onNewIntent(Intent intent) {
    super.onNewIntent(intent);
    handleIntent(intent);
}

private void handleIntent(Intent intent) {
```

```
        Tag tag = intent.getParcelableExtra(NfcAdapter.EXTRA_TAG);
        if (tag != null) {

            // parse through all NDEF messages and their records and pick text
            // type only
            Parcelable[] data = intent.getParcelableArrayExtra(NfcAdapter.EXTRA_NDEF_MESSAGES);

            String s = "";

            if (data != null) {
                try {
                    for (int i = 0; i < data.length; i++) {
                        NdefRecord[] recs = ((NdefMessage) data[i]).getRecords();
                        for (int j = 0; j < recs.length; j++) {
                            if (recs[j].getTnf() == NdefRecord.TNF_WELL_KNOWN
                                    && Arrays.equals(recs[j].getType(), NdefRecord.RTD_TEXT)) {
                                /*
                                 * See NFC forum specification for
                                 * "Text Record Type Definition" at 3.2.1
                                 *
                                 * http://www.nfc-forum.org/specs/
                                 *
                                 * bit_7 defines encoding bit_6 reserved for
                                 * future use, must be 0 bit_5..0 length of IANA
                                 * language code
                                 */
                                byte[] payload = recs[j].getPayload();
                                String textEncoding = ((payload[0] & 0200) == 0) ? "UTF-8" : "UTF-16";
                                int langCodeLen = payload[0] & 0077;
                                s += new String(payload, langCodeLen + 1, payload.length - langCodeLen - 1,
                                        textEncoding);
                            } else if (recs[j].getTnf() == NdefRecord.TNF_WELL_KNOWN
                                    && Arrays.equals(recs[j].getType(), NdefRecord.RTD_URI)) {
                                /*
                                 * See NFC forum specification for
                                 * "URI Record Type Definition" at 3.2.2
                                 *
                                 * http://www.nfc-forum.org/specs/
                                 *
                                 * payload[0] contains the URI Identifier Code
                                 * payload[1]...payload[payload.length - 1]
                                 * contains the rest of the URI.
                                 */
                                byte[] payload = recs[j].getPayload();
                                String prefix = (String) URI_PREFIX_MAP.get(payload[0]);
                                byte[] fullUri = Bytes.concat(prefix.getBytes(Charset.forName("UTF-8")),
                                        Arrays.copyOfRange(payload, 1, payload.length));
                                s += new String(fullUri, Charset.forName("UTF-8"));
                            }
                        }

                    } catch (Exception e) {
                        value = e.getMessage();
                        Log.e(NFCPluginTest.class.toString(), e.getMessage());
                    }
                }
                Log.i(NFCPluginTest.class.toString(), s);
                value = s;
            }
        }

        public static String getValue() {
            return value;
        }
    }
```

Liite 2. Lista Windows Phone –laitteista (RapidNFC 2014)

| Valmistaja | Malli |
|---|---|
| Nokia | Lumia 1020 |
| Nokia | Lumia 1520 |
| Nokia | Lumia 2520 |
| Nokia | Lumia 620 NFC Version |
| Nokia | Lumia 720 |
| Nokia | Lumia 735 |
| Nokia | Lumia 820 |
| Nokia | Lumia 830 |
| Nokia | Lumia 920 |
| Nokia | Lumia 925 |
| Nokia | Lumia 930 |
| HTC | 8X |

Liite 3. Lista Android –laitteista (RapidNFC 2014)

| | |
|---|---|
| Acer | Liquid E |
| Acer | Liquid E1 |
| Acer | Liquid S2 |
| Asus | Padfone 2 |
| Google | Nexus 10 |
| Google | Nexus 5 |
| Google | Nexus 7 (Current Version) |
| HTC | Desire 610 |
| HTC | Desire 816 |
| HTC | HTC One (M8) |
| HTC | One |
| HTC | One Max |
| HTC | One Mini 2 |
| Huawei | Ascend G6 |
| Huawei | Ascend P2 |
| LG | G2 D802 |
| LG | G2 Mini D620 |
| LG | G3 D855 |
| LG | Optimus 4X HD |
| LG | Optimus L5 |
| LG | Optimus L7 II |
| LG | Prada 3.0 |
| Motorola | Moto x |
| Samsung | Ativ S |
| Samsung | Galaxy Ace 3 |
| Samsung | Galaxy Alpha |

| | |
|---|---|
| Samsung | Galaxy Core Advance |
| Samsung | Galaxy Express 2 |
| Samsung | Galaxy Fame |
| Samsung | Galaxy K Zoom |
| Samsung | Galaxy Mega |
| Samsung | Galaxy Note 2 |
| Samsung | Galaxy Note 3 |
| Samsung | Galaxy Note 3 Neo |
| Samsung | Galaxy S2 |
| Samsung | Galaxy S3 |
| Samsung | Galaxy S4 |
| Samsung | Galaxy S4 Active |
| Samsung | Galaxy S4 Mini |
| Samsung | Galaxy S4 zoom |
| Samsung | Galaxy S5 |
| Samsung | Galaxy S5 Mini |
| Samsung | Galaxy Young 2 |
| Sony | Xperia L |
| Sony | Xperia M |
| Sony | Xperia M2 |
| Sony | Xperia M2 Aqua |
| Sony | Xperia SP |
| Sony | Xperia T3 |
| Sony | Xperia Z |

| | |
|---|---|
| Sony | Xperia Z Ultra |
| Sony | Xperia Z1 |
| Sony | Xperia Z1 Compact |
| Sony | Xperia Z2 |
| Sony | Xperia Z3 |
| Turkcell | Turkcell T40 |