TAMPEREEN
AMMATTIKORKEAKOULU

VIDEOIDEN JÄRJESTELY .NET-SOVELLUS-KEHYKSELLÄ

Sampo Lähdesmäki

Opinnäytetyö
Joulukuu 2016
Tietotekniikka
Ohjelmistotekniikka

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka

LÄHDESMÄKI, SAMPO:
Videoiden järjestely .NET-sovelluskehyksellä

Opinnäytetyö 24 sivua, joista liitteitä 0 sivua
Joulukuu 2016

Opinnäytetyön tarkoituksena oli luoda tietokoneelle ohjelma, jolla pesäpallon otteluvideoiden tarkastelu olisi helpompaa. Työn tavoitteena on opetella ja tutustua C#-ohjelmointikieleen, jolla kyseinen ohjelma on toteutettu. Ohjelma tehdään omaan käyttöön, ja idea opinnäytetyön aiheelle lähti omasta mielenkiinnosta lajia kohtaan. Tavoitteena oli lisäksi opetella uusi ohjelmointikieli. Työssä käydään läpi ohjelman tekemiseen käytetyt vaiheet sekä esitellään hieman C#-kielen ominaisuuksia.

Pesäpallossa, kuten myös monissa muissa urheilulajeissa, ottelut videoidaan myöhempää analysointia varten. Videoilta voidaan käydä läpi mm. eri tilanteita sekä tutkia pelaajien tekemiä ratkaisuja. Työssä tehtiin kaksi eri ohjelmaa, joilla videoiden purkaminen ja katsominen suoritetaan. Toinen ohjelma tallentaa videosta tietoja tekstitiedostoihin, jotka käyttäjä itse syöttää ja toisella ohjelmalla voidaan katsoa videoita ja lukea tietoja tallennetuista tiedostoista. Tietojen tallennus tapahtuu paikallisesti tietokoneen kovalevylle. Ohjelman tekemiseen käytettiin C#-kieltä sen tehokkuuden ja helppokäyttöisyyden vuoksi.

Ohjelma oli tarkoitus saada valmiiksi alkukesästä, jolloin sen käyttämiseen, testaamiseen ja jatkokehittämiseen jäisi myös aikaa. Ohjelmasta saatiin toimiva versio, mutta sen ominaisuuksia pystyisi vielä kehittämään pidemmälle. Käyttöliittymän parantaminen sekä datan tallennus tietokantaan ovat ominaisuuksia, joita voisi kehittää käyttökokemuksen parantamiseksi. Pienillä muutoksilla ohjelmaa voisi mahdollisesti käyttää myös muissa urheilulajeissa kuin pesäpallossa.

Asiasanat: ohjelmistokehitys, c#-ohjelmointi, .net-sovelluskehys

ABSTRACT

Tampere University of Applied Sciences
Information Technology
Software Engineering

LÄHDESMÄKI, SAMPO
Video organizing with .NET framework

Bachelor's thesis 24 pages, appendices 0 pages
December 2016

Purpose of this thesis was create an application for computer, which you can analyze your videos recorded from baseball matches. Main goal was learn and explore the C# programming language in which the program is implemented. The Program is made for personal use only. Idea of this thesis topic is my interest towards baseball. This thesis presents the developing process of this program and introduce a little C# language features.

In baseball as in many other sports, the matches are recorded for the later analyzing. From those videos you can take a look the match from different perspective and analyze players. Two different program was made by this thesis. One for the organizing and saving data from the videos and the other for watching them. Application saves data locally to user computer hard disk. Program was made by using C# programming language because of its efficient and user friendly features.

Program was planned to be completed in early summer so there is time for testing and further development of the project. Improving user interface and saving data to database are features which can be developed more for better user experience. With few changes this program could possibly used in other sports as well.

Key words: software development, c# programming, .net framework

SISÄLLYS

LYHENTEET JA TERMIT

Java	oliopohjainen ohjelmointikieli
C++	ohjelmointikieli
C#	ohjelmointikieli
bugi	ohjelman koodissa oleva virhe
WMP	Windows Media Player
VLC	avoin, alustariippumaton mediasoitin
CLR	Common Language Runtime, ajonaikainen ympäristö
WinForm	Windows Form Sovellus

1 JOHDANTO

Opinnäytetyön aiheena on tehdä ohjelma, jolla pesäpallon otteluvideoita voisi leikata pienemmiksi leikkeiksi, sekä järjestellä videoita, jotta niiden analysointi jälkeenpäin olisi helpompaa. Idea ohjelman tekemiseen lähti tekijän omasta mielenkiinnosta lajia kohtaan. Opinnäytetyö tehtiin omaan käyttöön ja työssä käsitellään ohjelman eri vaiheita, sekä käydään läpi itse ohjelmaa.

Työn tavoitteena on myös tutustua tekijälle uuteen ohjelmointikieleen ja tässä tapauksessa kieleksi on valittu C#. Ohjelma on tehty Windows käyttöjärjestelmälle käyttäen kyseistä ohjelmointikieltä.

Ensimmäisessä kappaleessa käydään läpi hieman taustaa miksi ohjelmaa lähdettiin toteuttamaan, sekä käydään läpi mitä ohjelman pitäisi tehdä. Tässä kappaleessa on myös tehty hieman suunnitelmaa, miten ohjelmaa tultaisiin toteuttamaan.

Toisessa kappaleessa on esiteltynä työkalut, joita tämän ohjelman tekemiseen on käytetty. Tässä on kerrottu millä ohjelmointikielellä ja millä ohjelmointiympäristöllä työ on toteutettu. Ohjelma toteutettiin Windows -käyttöjärjestelmälle, joten suuri osa on Microsoftin Windowsille suositeltuja työkaluja.

Kolmannessa ja neljännessä kappaleessa käydään läpi projektin teknistä toteutusta, sekä ohjelman toimintaa. Aluksi kerrotaan ohjelman toteutus vaihe vaiheelta, jonka jälkeen ohjelmaa on testattu ja kehitetty toimivammaksi. Neljännessä kappaleessa on esitelty valmista ohjelmaa ja käydään läpi yksityiskohtaisemmin, kuinka ohjelma toimii.

Viimeisessä kappaleessa on pohdintaa työn tekemisestä ja esitetty, kuinka ohjelmaa pitäisi kehittää eteenpäin.

2 TAUSTAA

Pesäpallossa, kuten monissa muissakin urheilulajeissa, ottelut kuvataan videokameroilla myöhempää tarkastelua varten. Videoiden avulla voidaan tehostaa joukkueen valmistautumista seuraavaan otteluun, sekä joukkueen omaa peliä on helpompi analysoida. Nykyisin korkeimmilla sarjatasoilla ottelut ovat ladattuna videopalveluun kaikkien katseltavaksi. Koska kaikkien joukkueiden pelit ovat saatavilla, pystytään vastustajien yksittäisiä pelaajia, sekä yksittäisiä lyöntejä ja eri tilanteita tarkastelemaan paremmin. Myös omaa peliä on helpompi tutkia jälkeenpäin videolta, kun yksittäisiä tilanteita voi katsoa uudestaan.

Kokonaisesta otteluvideosta eri tilanteiden etsiminen vie oman aikansa, joten tätä taustaa vasten lähdettiin toteuttamaan ohjelmaa, jolla videoita voisi järjestellä paremmin. Ohjelman idea yksinkertaisesti olisi leikata kokonaisesta otteluvideosta pienempiä pätkiä, joiden avulla yksittäiset tilanteet olisi helpompi löytää jälkeenpäin ja järjestellä eri tilanteiden mukaan. Ohjelma on tarkoitus tehdä omaan käyttöön, eikä ole tavoitteena tehdä kaupallista sovellusta.

2.1 Suunnittelu ja vaatimukset

Aluksi mietittiin, mitä ohjelman olisi tarkoitus tehdä, josta saatiin luotua vaatimukset sovellukselle. Ohjelman pitäisi tallentaa kokonaisesta videosta pätkä, jonka käyttäjä on itse määritellyt ja josta kävisi ilmi mitä videossa tapahtuu. Ohjelma toimisi ikään kuin videoeditori, jolla pystyisi leikkaamaan pätkiä ja sisällyttämään tietoja videoon. Käyttäjä merkkaisi halutun alku- ja loppukohdan, sekä tietoja lyöjästä, lyönnistä, tilanteesta jne., jonka jälkeen ohjelma leikkaa kyseisen videonpätkän ja tallentaisi sen tiettyyn kansioon. Ohjelma järjestelisi videot kansioihin järkevästi, jotta niiden etsiminen ei veisi aikaa. Käyttäjän pitäisi pystyä myös helposti katsomaan videoita esimerkiksi tietyn lyöjän, tilanteen tai jonkin muun kriteerin mukaan.

Ohjelmaa lähdettiin suunnittelemaan siten, että ohjelmasta tehdään aluksi toimiva versio, jotta sitä voisi testata ja hyödyntää käytännössä mahdollisimman nopeasti. Koska ohjelma

tuli omaan käyttöön, graafiseen ulkoasuun, sekä käyttöliittymään ei alkuvaiheessa kiinnitetä huomiota, vaan niiden kehittämiseen palataan, kun videoiden järjestely ja leikkaus toimii. Koska tavoitteena oli nopeuttaa videoiden läpikäyntiä, täytyy käyttöliittymän olla helppokäyttöinen ja nopea. Sen vuoksi käyttöliittymään kiinnitetään huomiota sen jälkeen, kun videoiden leikkaus onnistuu. Graafista ulkoasua voidaan parantaa ohjelman toteutuksen ohella, sekä panostaa siihen, kun käyttöliittymä on saatu hyvin toimivaksi.

Työn tavoitteena oli myös tutustua uuteen ohjelmointikieleen, joten kieleksi valittiin tekijälle ennestään tuntematon kieli. Työ päätettiin tehdä C#-kielellä, sillä kyseinen kieli on samankaltainen C++ ja Java kielten kanssa. Edellä mainitut kielet olivat tekijälle ennestään tuttuja ja joista oli hieman ohjelmointi kokemusta. C# soveltuu myös hyvin Windows käyttöjärjestelmille tehtävien sovellusten tekemiseen, joten sen vuoksi oli luontaista valita kyseinen kieli. C#-kieleen on saatavilla myös paljon ohjelmakirjastoja ja lisäosia, joilla ohjelman toteuttaminen olisi helpompaa. Ohjelmointiympäristönä päätettiin käyttää Visual Studio ohjelmaa ja sovellus tulisi pelkästään Windows käyttöjärjestelmälle

3 KÄYTETYT TYÖKALUT

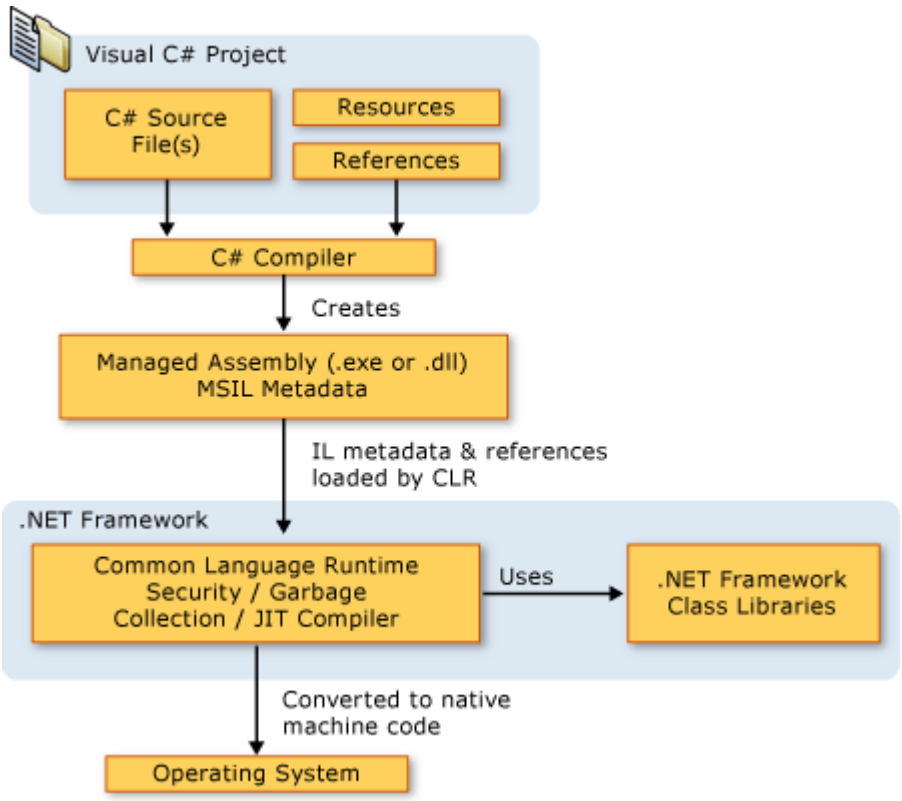
Tässä kappaleessa käydään lyhyesti läpi millä työkaluilla projekti on suoritettu. Kappaleessa tutustutaan C#-ohjelmointikieleen ja .NET-sovelluskehykseen, mutta esimerkiksi itse ohjelmointikielen perusrakenteita tai olio-ohjelmoinnin perusteisiin ei tässä paneuduta. Teorian lähteenä on käytetty pääosin Microsoftin kehittäjä sivuston tietoja (Microsoft developer network: Introduction to the C# Language and the .NET Framework), sekä C#-ohjelmoinnin perusteisiin kirjoitettua kirjaa (C#-ohjelmointi, Ghodrat Moghadampour, Docendo 2009).

3.1 C#-ohjelmointikieli

C#-ohjelmointikieli on 2000-luvun alussa Microsoftin kehittämän kieli .NET alustalle, joka yhdistelee C++ ja Java kielen ominaisuuksia. C#-kielen etuna on sen tehokkuus, kuten C++ kielessä, sekä yksinkertaisuus, kuten Javassa. Niin kuin edellä mainitut kielet, myös C# on olio-ohjelmointi kieli. C# on erittäin monikäyttöinen kieli ja sillä voidaankin tehdä web -sovelluksia, mobiiliohjelmia ja paljon muuta.

3.2 .NET-sovelluskehys

Sovelluskehys on ohjelmisto, joka muodostaa rungon, jonka päälle voidaan rakentaa tietokoneohjelmia. Se tarjoaa valmiiksi osia, joita ei tarvitse käyttää ohjelman kehityksen aikana. Windows ympäristössä käytetään .NET-sovelluskehystä, jonka ensimmäinen kaupallinen versio tuli vuonna 2000. Sen pääkomponentteja ovat ohjelmointikielet ja työkalut kuten Visual Basic, C#, C++, jotka mahdollistavat ohjelman kirjoittamisen. Toinen pääkomponentti on ajonaikainen ympäristö, jossa ohjelmia voidaan ajaa turvallisesti. Samalla tavalla kuin Java -ympäristössä on käytössä JRE (Java Runtime Environment) on .NET-sovelluskehyksessä ajoympäristö CLR (Common Language Runtime), joka suorittaa mm. muistin hallintaa ja säikeiden hallintaa. Kolmantena pääkomponenttina .NET-sovelluskehyksessä on luokkakirjasto, joka tarjoaa jo valmiiksi kirjoitettuja luokkia, joita kehittäjä voi vapaasti käyttää ohjelmissaan. Tässäkin opinnäytetyössä on hyödynnetty paljon näitä tarjottuja kirjastoja.



KUVA 1. Kaavio .NET sovelluskehyksen toiminnasta.

Kuvan 1 kaaviossa on havainnollistettu, kuinka C# sovellus käännetään konekieleksi, ja näin ollen toimivaksi ohjelmaksi halutulle käyttöjärjestelmälle. C#-projekti resursseineen käännetään aluksi kääntäjän avulla yleiskieleksi (IL, Intermeditate Language). Tämän jälkeen sovelluskehyksen tarjoama ajonaikainen ympäristö (CLR) kääntää natiiviksi konekieleksi, jota voidaan ajaa käyttöjärjestelmässä. .NET-sovelluskehys tarjoaa myös palveluja, kuten roskien keruuta, muistinhallintaa ja käyttää myös luokkakirjastoja.

3.3 Visual Studio Community

Ohjelmointiympäristönä työssä käytettiin Visual Studio Communityä. Se on Microsoftin Visual Studio tuoteperheeseen kuuluva ilmainen ohjelmistokehitysalusta. Ohjelma tukee useita eri kieliä kuten C#, C++, JavaScript, Python jne. ja sillä voidaan kehittää niin Web -sovelluksia kuin mobiilisovelluksia. Visual Studio valittiin käytettävän työssä, sen helppokäyttöisyyden vuoksi. Myös aikaisempaa kokemusta kyseisestä ohjelmasta on, joten ohjelman käyttäminen oli luontevaa.

4 PROJEKTIN TOTEUTUS

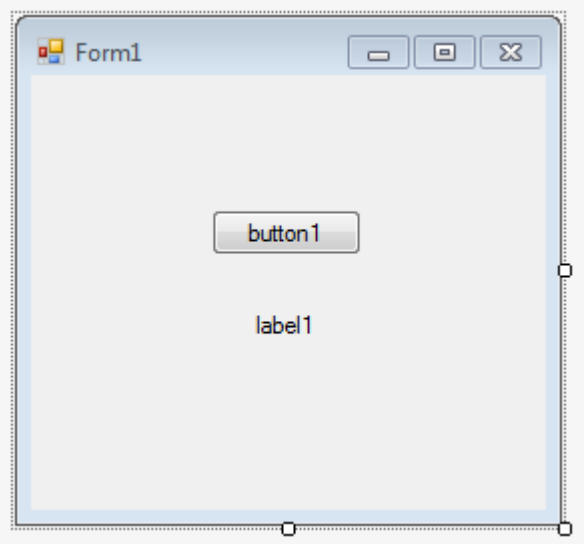
Seuraavassa kappaleessa esitellään, kuinka ohjelma on tehty ja paneudutaan toteutuksen tekniseen puoleen. Kappaleissa käydään läpi ohjelman toteutus ja käydään läpi eri ohjelmistosta valmistuneet versiot. Tässä käydään myös läpi lyhyesti, miten ohjelman testaaminen vaikutti siihen tehtäviin muutoksiin ja käyttöliittymän parantamiseen. Vaikka tämän kappaleen aiheena on esittää ohjelman tekninen toteutus, käydään myös kappaleissa läpi eri versioiden toimintaa läpi lyhyesti, jotta lukija tietää minkälaista ohjelmaa on pyritty toteuttamaan.

4.1 Ohjelman toteutus

4.1.1 Windows Form Application

Ohjelman käyttöliittymä toteutettiin käyttämällä valmista Windows Form sovellusta, jonka pohjalle ohjelma rakennettiin. Windows Form sovellus luo valmiit resurssit ja kirjastot, joilla voi helposti toteuttaa graafisen käyttöliittymän Windows -käyttöjärjestelmälle. Form on käytännössä visuaalinen näkymä, jonka avulla käyttäjä käyttää ohjelmaa. Tähän näkymään voidaan tuoda (drag & drop -tyyliin) valmiita nappuloita, tekstikenttiä, vetovalikoita ja muita komponentteja, sekä antaa niille erilaisia toimintoja. Uutta projektia luotaessa voi valita Windows Form sovelluksen, jolloin ohjelma luo graafisen Windows -ikkunan. Kuvassa 2 on luotu uusi projekti ja Form:iin on lisätty yksi nappula ja yksi tekstikenttä. Lisäämällä alla olevan aliohjelma esimerkin ohjelmakoodiin voi kuvan 2 tekstikenttään vaihtaa tekstin ”Hello World!”, kun nappulaa painetaan. Näitä komponentteja yhdistelemällä ja hyödyntämällä on toteutettu projektin käyttöliittymä.

```
private void nappulaPainettu(object sender, EventArgs e)
{
    label1.Text = "Hello World!";
}
```

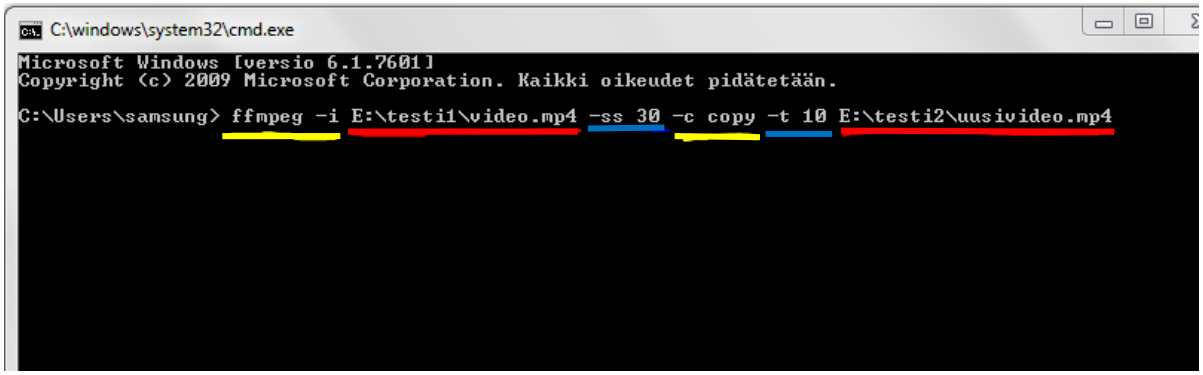


KUVA 2. Windows Form sovellus, johon lisätty nappula ja tekstikenttä.

4.1.2 Ensimmäinen versio käyttäen FFmpeg ohjelmaa

Aluksi oltiin siis päätetty, että ohjelmalla leikataan videosta käyttäjän määrittämä pätkä ja tallennetaan se tiettyyn kansioon. Käyttäjä määrittää videosta halutun alku- ja loppukohdan, sekä tietoja tilanteesta, kuten lyöjä, etenijä, tilanne jne. Tämän jälkeen ohjelma leikkaa videosta kyseisen pätkän ja tallentaa sen haluttuun kansioon. Samalla ohjelma kopioi kyseisen videonpätkän myös muihin kansioihin mihin tilanne halutaan tallentaa.

Ensimmäiseksi lähdettiin etsimään keinoa, jolla videosta voidaan leikata pätkiä. Tässä päädyttiin käyttämään ohjelmaa nimeltä FFmpeg. Se on komentoriviltä ajettava ohjelma, jolla pystyy käsittelemään mediatiedostoja. Sillä pystyy mm. kääntämään videotiedostoja muodosta toiseen, sekä leikkaamaan videoita ja paljon muita ominaisuuksia, joita ei kuitenkaan tässä työssä tarvita. Ohjelmassa ei ole minkäänlaista käyttöliittymää, joten sen käyttäminen on tehokasta ja nopeaa. Myös videoiden käsittely ohjelmalla on hyvin yksinkertaista, joten se soveltuu erinomaisesti käytettävän toisen ohjelman kautta. Sen vuoksi työssä päädyttiin siihen, että videot leikataan käyttämällä kyseistä ohjelmaa.



KUVA 3. Komentorivillä ajettu FFmpeg komento.

Kuvassa 3 on esitetty, kuinka FFmpeg ohjelma käytännössä toimii. Sitä ajetaan komentoriviltä, johon kirjoitetaan haluttuja toimintoja. Ensimmäinen punaisella alleviivattu komento kertoo tiedoston, jota halutaan muokata ja toinen punainen kansion ja tiedostonimen, minne leikattu pätkä halutaan tallentaa. Kuvassa sinisellä alleviivatut komennot kertovat ajankohdan, joka videosta halutaan leikata. Tässä esimerkissä aloituskohta on 30 sekuntia videon alusta eli ajassa 00.00.30 (tunnit, minuutit, sekunnit) ja lopetuskohta 10 sekuntia myöhemmin eli aikaan 00.00.40. Näin ollen alkuperäisestä videosta on leikattu kymmenen sekunnin pätkä ja tallennettu se uuteen erilliseen kansioon -c copy käskyllä. Komentoja voi putkittaa myös enemmän, joten saman tiedoston saa kopioitua moneen eri kansioon lisäämällä aina uuden -c copy komennon ja kohteen minne tiedosto tallennetaan. Koska FFmpeg ohjelmassa tallennuskansio täytyy olla olemassa, eikä ohjelma itse luo uusia kansioita, täytyi kansiot ensiksi luoda. Uuden kansion luontiin käytettiin C#-kielestä valmiiksi löytyvää kirjastoa, jolla kyseinen toimenpide onnistuu.

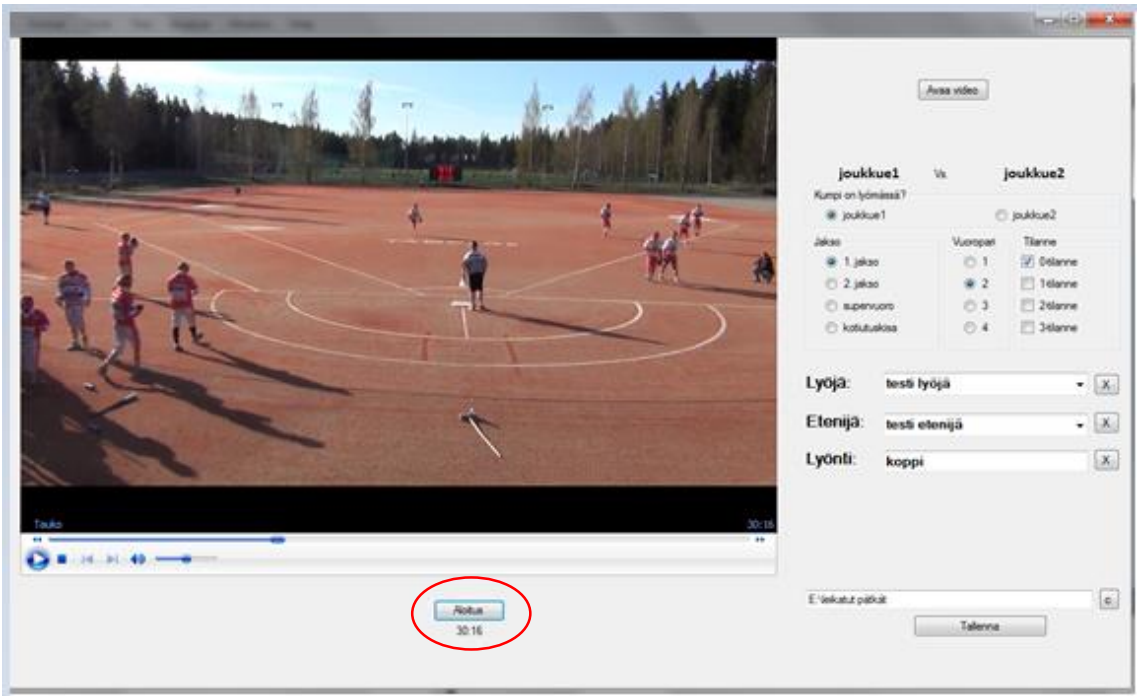
Ohjelmassa leikkaus ajankohdat, sekä kansiot mihin tiedostot tallennetaan ovat tallennettu muuttujiin, jotka muuttuvat käyttäjän syötteen mukaan. Ohjelma tallentaa leikatut pätkät lyöjän nimen mukaan omiin kansioihinsa ja tiedostonimi kertoo hieman tietoja tilanteesta. Näin ollen yhden otteluvideon saa pätkittyä lyhyemmiksi pätkiksi, joita on helpompi tarkastella jälkeenpäin. Käyttäjän tarvitsee katsoa koko ottelu kerran ja merkata haluamansa lyöjät jolloin ohjelma leikkaa ja järjestää ne kansioihin, joista niitä on helpompi tarkastella jälkeenpäin. Alla olevassa koodissa on esimerkki, jolla saadaan ohjelmallisesti käynnistettyä komentorivi ja ajettua siinä haluttu komento. Ensimmäinen komento luo ensiksi uuden kansion, johon leikattu videon pätkä tallentuu. Viimeinen rivi käynnistää komentorivin ja ajaa siinä edellä luodun komennon. Myös komentorivin käyttämiseen löytyy valmiiksi kirjasto mitä käyttää.

```
public void ajaKomentorivi()
        {
             System.IO.Directory.CreateDirectory("E:\testi2");

             string komento = @"/C ffmpeg -i E:\testi1\video.mp4 -ss 30 -c copy -
             t 10 E:\testi2\uusivideo.mp4";

            System.Diagnostics.Process.Start("CMD.exe", komento);
        }
```

Edellä mainittuja toimintoja hyödyntäen ohjelma saatiin toimimaan mainiosti ja haluttuun lopputulokseen päästiin. Ohjelma leikkasi videoita yhdestä ottelusta käyttäjän syötteiden mukaan, sekä järjesteli ne haluttuihin kansioihin. Työ saatiin melko nopeasti valmiiksi, mikä oli tavoitteena, mutta heti työn valmistumisen jälkeen huomattiin ongelma. Videot ovat kooltaan melko isoja ja niistä leikattuja pätkiä syntyi paljon, joten tiedostot veivät aivan liian paljon tilaa kovalevyltä, joten kyseinen menetelmä ei olisi kovinkaan järkevä. FFmpeg ohjelma täytyi myös asentaa tietokoneelle valmiiksi, joten ohjelmaa pystyi käyttämään vain niissä koneissa, joihin ohjelma oli asennettu. Ensimmäisen version tekemiseen ei kulunut kovin paljoa aikaa, joten päätettiin lähteä kehittämään toista versiota, joka toimisi hieman eri tavalla. Käyttöliittymään ei tässä vaiheessa tehty muutoksia

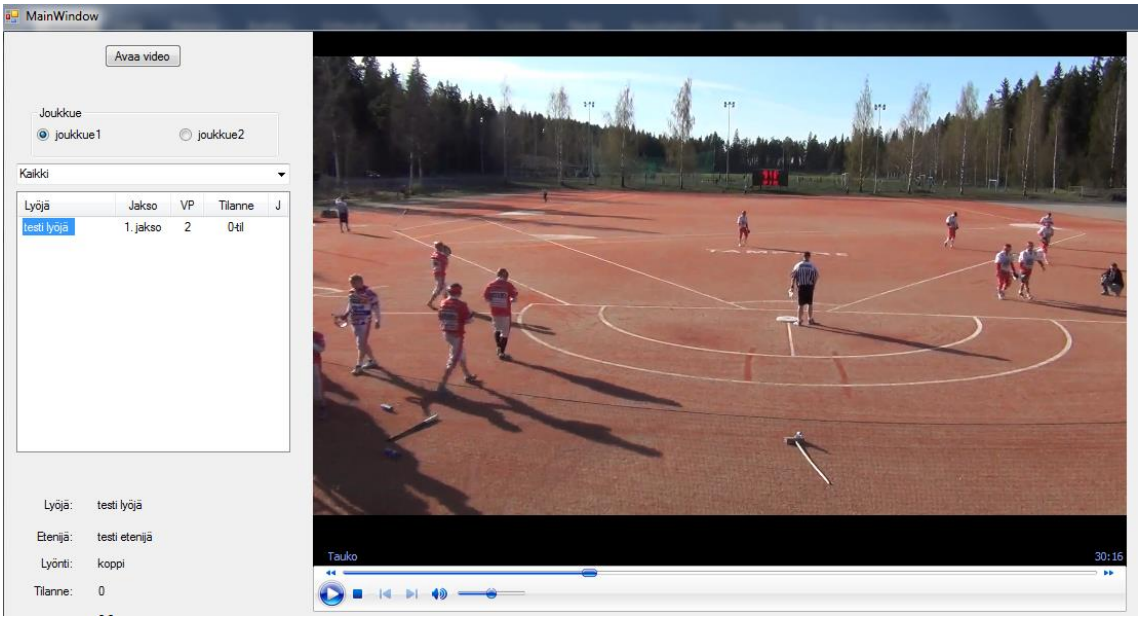
4.1.3 Toinen versio

Koska ensimmäinen versio ohjelmasta vei liian paljon tilaa kovalevyltä, täytyi ohjelmaa kehittää niin, ettei videoita enää leikata erillisiksi pätkiksi. Näin ollen päädyttiin ratkaisuun, jossa käyttäjän syöttämä aloitusajankohta tallennetaan muistiin. Kun videota katsotaan jälkeenpäin, ohjelma hakee muistista ajan, joka oli tallennettu ja käynnistää videon siitä ajankohdasta. Kyseinen menetelmä ei veisi tilaa kovalevyltä ja olisi vieläkin yksinkertaisempi tapa toteuttaa, koska ohjelmassa ei tarvitse ajaa toista ohjelmaa. Seuraavaksi on hieman esitelty, kuinka yhden tilanteet tallennus käytännössä tapahtuu.



KUVA 4. Käyttöliittymä, jolla videoita tallennetaan

Kuvassa 4 punaisella ympyröidystä napista painamalla ohjelma ottaa ajan muistiin. Käyttäjä painaa nappia haluamassaan kohdassa, sekä lisää oikealla olevassa paneelissa tietoja lyöjästä, tilanteesta jne. Sen jälkeen käyttäjä painaa tallenna nappia, jolloin ohjelma on tallentanut videosta kyseisen ajankohdan tietoineen. Tässä esimerkissä haluttu ajankohta on 30.16, joka tallentuu muistiin.



KUVA 5. Käyttöliittymä videoiden katseluun.

Kuvassa 5 vasemmalla olevassa paneelissa näkyy kaikki tallennetut tilanteet. Tässä esimerkissä näkyy yksi esimerkki tallenne, joka on tehty kuvassa 4. Lyöjän nimeä painamalla ohjelma lukee tiedostosta kyseisen tilanteet ja aloittaa videon toistamisen kyseisestä ajankohdasta. Tässä esimerkissä video käynnistyisi ajasta 30.16, joka oli tallennetun leikkeen ajankohta. Kuvassa 5 vasemmalla alareunassa näkyy tietoja, jotka käyttäjä on syöttänyt tallennus vaiheessa. Ohjelma listaa kaikki tallennetut tilanteet valitusta ottelusta ja näyttää ne listassa. Käyttäjä voi järjestää listaa painamalla otsikkoa, jolloin järjestys muuttuu, joko nousevaksi tai laskevaksi. Esimerkiksi painamalla ”Lyöjä” otsikkoa, vaihtuu listan järjestys lyöjän nimen mukaan. Listan ylläolevasta vetovalikosta voi valita myös pelkästään tietyn lyöjän tilanteet, jolloin eri pelaajien tilanteet on helpompi löytää.

Listaus on toteutettu ListView komponentilla ja järjestely toiminto käyttämällä ListViewItemComparer Luokkaa. Luokka vertailee kahta objektia ja järjestää ne aakkosjärjestykseen. Tämä tehdään kaikille objekteille, jolloin ohjelma järjestää listan aakkosjärjestykseen joko nousevasti tai laskevasti, sen mukaan mitä otsikkoa on painettu. Alla luokka, joka vertailee objekteja. Se saa parametriksi ListView komponentin otsikon jota on painettu.

```
class ListViewItemComparer : IComparer
    {
        private int col = 0;

        public ListViewItemComparer(int column)
        {
            col = column;
        }
        public int Compare(object x, object y)
        {
            return String.Compare(((ListViewItem)x).SubItems[col].Text,
((ListViewItem)y).SubItems[col].Text);
        }
    }
```

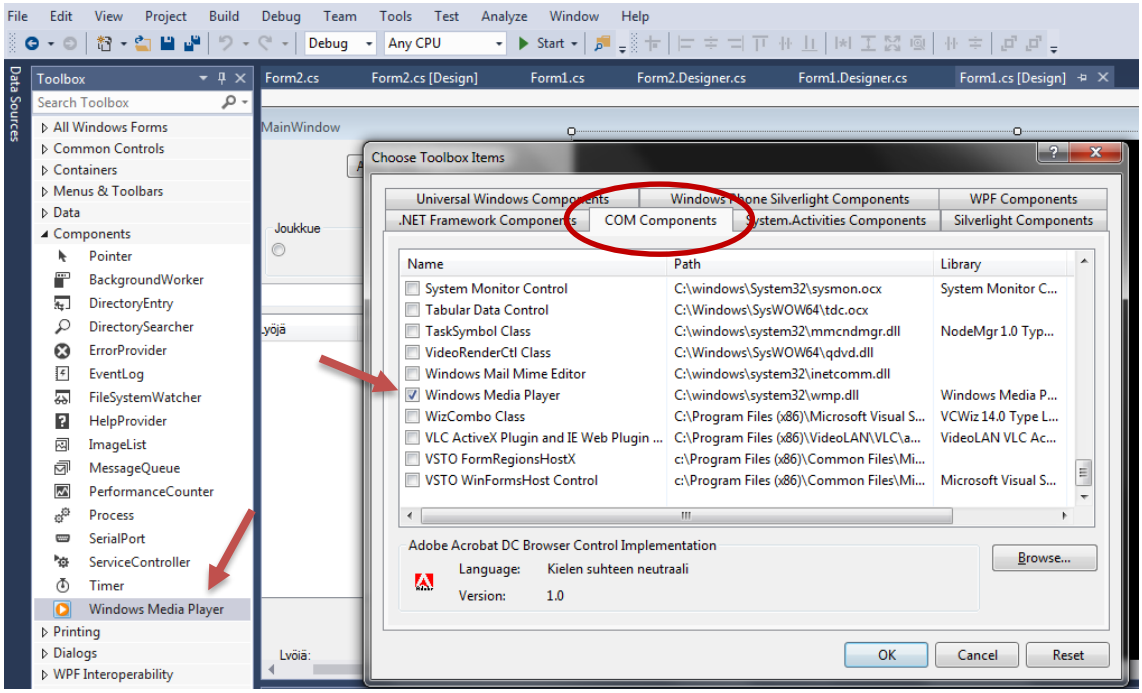
Tilanteiden tallennus tapahtuu paikallisesti tietokoneen kovalevylle. Tallennus tapahtuu tekstitiedostoon, jonne yhdestä tapahtumasta kertyy n. 20 riviä tekstiä. Tapahtumia yhdestä ottelusta kertyy paljon, mutta ottelusta syntyvän tekstitiedoston koko on kuitenkin huomattavasti pienempi kuin leikatun videonpätkän. Tekstitiedostoihin kirjoittaminen ja niistä lukeminen onnistuu C#-kielellä hyvin yksinkertaisesti käyttämällä StreamReader ja StreamWriter luokkia. StreamWriter luokkaa käyttämällä voidaan kirjoittaa mitä tahansa tekstiä tiedostoon. Tässä tapauksessa ohjelma tallentaa kaikki tiedot .txt -tiedosto-

muotoon. Tiedostosta lukemiseen käytetään puolestaan StreamReader luokkaa, joka lukee halutun tekstitiedoston rivi kerrallaan tiedoston loppuun asti. Kyseiset luokat löytyvät valmiiksi ohjelmakirjastosta, joten niiden käyttäminen ohjelmassa oli luontevaa, eikä omaa toteutusta tarvinnut keksiä kyseiseen toimenpiteeseen. Ohjelma tallentaa yhdestä ottelusta vain pari tekstitiedostoa, joten myös niiden jakaminen ja järjestely, on huomattavasti helpompaa, kuin kymmenien eri videonpätkien.

```
using (StreamWriter write = new StreamWriter(path1, true))
                {
                    write.WriteLine("testi teksti");
                }
```

Ylläolevalla koodi on esimerkki StreamWriter luokan toiminnasta. Kyseisellä koodiesimerkillä saadaan kirjoitettua tekstitiedostoon teksti ”testi teksti”. Path1 -muuttujaan laitetaan polku, josta tekstitiedosto löytyy ja true -parametri ei kirjoita tekstitiedoston päälle. mikäli tiedostossa on jotain tekstiä, jatkuu tiedostoon kirjoittaminen viimeisen tekstirivin jälkeiselle riville. Tässä tilanteessa ohjelma kirjoittaisi kyseisen tekstin uudelle riville.

Videoiden toistaminen tapahtuu käyttäen Windows Media Playerin valmista ohjelmakirjastoa. Visual Studiosta löytyy valmiiksi komponentti, jolla videoiden toistaminen on hyvin yksinkertaista. WMP lisäosalla pystyy hoitamaan yksinkertaiset videontoisto toiminnot, sekä sillä saa pääsee käsiksi joihinkin videon tietoihin, kuten esimerkiksi aikaan.



KUVA 6. Windows Media Player komponentin asennus.

Komponentti täytyy lisätä projektiin ja se onnistuu Visual Studiossa Tools -valikosta valitsemalla ”Choose Toolbox Items”. COM Components välilehden alta löytyy Windows Media Player lisäosa, jonka lisäämällä komponentti näkyy suoraan työkalu paneelissa. Visual studio lisää automaattisesti wmp.dll tiedoston projektiin, jolloin Windows Media Playerin voi vetää valikosta suoraan paneelille ja siinä toimii normaalit toiminnot. Kun komponentti on lisätty paneelille, voidaan video toistaa komennolla

```
wmp.URL = open.FileName
```

Jossa wmp on lisätyn Windows Media Player komponentin nimi. Metodi URL aloittaa videon toistamisen polusta, joka sille annetaan. Tässä tilanteessa kohteen polku avataan käyttäen valmista Windowsin tiedoston avaus dialogia.

4.2 Testaus ja käyttöliittymän kehittäminen

Ohjelma saatiin valmiiksi, joten sen jälkeen voitiin aloittaa sen testaaminen käytännössä. Testauksen ideana oli saada karsittua pois bugit, sekä sen avulla käyttöliittymää ja ohjelmaa voitiin parannella. Aluksi ohjelmaa testattiin yhdellä ja samalla videolla, jolla saatiin karsittua pois pieniä bugeja, sekä tehtiin pieniä parannuksia. Käytettyään vähän aikaa ohjelmaa, huomattiin että yhden ottelun läpikäymiseen kuluu vieläkin hieman liian paljon aikaa, sillä tietojen syöttäminen, sekä ohjelman käyttäminen tapahtuu hiirellä. Hiiren käyttö on helppoa, mutta se vie kuitenkin aikaa, joten lyhyen testauksen jälkeen tultiin siihen tulokseen, että käyttöliittymää pitää parantaa niin, että ohjelman käyttäminen on nopeampaa ja sujuvampaa.

Muutoksia ohjelmaan tehtiin niin, että tietoja ei tarvitse juurikaan syöttää, vaan ohjelma lukee tilanteet automaattisesti ja ohjelmaa käytetään pääosin näppäimistöllä. Näillä muutoksilla ohjelman käyttö nopeutui hieman ja käyttö helpottui. Yhden ottelun järjestelyyn meni huomattavasti vähemmän aikaa, kuin mitä siihen kului ennen parannuksia. Viidennessä kappaleessa esitellään ohjelman toimintaa, jossa kyseiset muutokset näkyvät ja jossa niihin paneudutaan hieman tarkemmin. Alla on koodiesimerkki, jolla ohjelma lukee mitä näppäintä painetaan. Tässä esimerkissä ohjelma toteuttaisi jotakin, kun käyttäjä painaa t -näppäintä.

```
protected override bool ProcessCmdKey(ref Message msg, Keys keyData)
        {

            if (keyData == Keys.T)
            {
                // Do something here...

                return true;
            }


            return base.ProcessCmdKey(ref msg, keyData);
        }
```

Toinen iso muutos ohjelmaan, oli sen muuttaminen kahdeksi eri ohjelmaksi. Aluksi samalla ohjelmalla hoidettiin sekä järjestely, että videoiden katselu, mutta ne päätettiin erottaa toisistaan selkeyden vuoksi. Näin ollen toinen ohjelma toimii editorina ja toinen videoiden katseluun. Ohjelman käyttäminen esimerkiksi joukkueen sisällä helpottuu, kun yksi ihminen voi hoitaa editoinnin ja muut voivat käyttää ohjelmaa vain videoiden katseluun. Sen vuoksi kyseinen muutos ohjelmaan päätettiin tehdä, vaikka ohjelmaa ei juurikaan ole käytetty muussa kuin tekijän omassa käytössä. Jatkoa ajatellen ohjelmaan tehtiin myös asennustiedosto, jolla ohjelman voi asentaa ajamalla asennustiedoston.

Ohjelmaa testattiin noin 1-2 kuukauden verran, minkä aikana edellä mainitut ongelmat korjattiin, sekä parannukset tehtiin. Vasta käytännön testauksessa huomasi mitä asioita ohjelmassa täytyy parantaa, vaikka ohjelmasta luuli tehneensä hyvän jo ensimmäisellä kerralla. Tässä vaiheessa tuli oppineeksi, kuinka tärkeä vaihe testaus on projektissa. Ohjelmaan jäi vielä paljon paranneltavaa ja kehitettävää, mutta mikäli ohjelmaa parantelisi jatkuvasti, niin työtä ei saisi koskaan valmiiksi. Kuitenkin ohjelmasta saatiin melko hyvin toimiva versio, mitä voisi hyödyntää nyt käytännössä.

5 OHJELMAN TOIMINTA

Tässä luvussa käydään hieman läpi, kuinka ohjelma toimii. Ohjelmaa kehitellään jatkuvasti, joten tässä on esitelty ohjelman viimeisintä versiota. Kuten aikaisemmin on tullut esille, ei graafiseen ulkoasuun olla panostettu, vaan sitä parannellaan myöhemmin. Tietojen tallennus tapahtuu vielä tekijän tietokoneen kovalevylle. Muuten ohjelma toimii halutulla tavalla ja ohjelmalla voidaan otteluvideoita käydä läpi, esimerkiksi lyöjä kerrallaan. Koska ohjelmasta tehtiin kaksi eri ohjelmaa, käydään molemmat ohjelmat erikseen läpi.

5.1 Videoiden purkaminen

Ensiksi käydään läpi ohjelmaa, jolla videot puretaan. Aikaisemmin on kerrottu enemmän ohjelman teknisestä toteutuksesta, joten tässä kappaleessa kerrotaan pääpiirteittäin, kuinka ohjelma toimii.

KUVA 7. Ohjelman päänäkymä.

Kuvassa 7 näkyy ohjelman päänäkymä. Käyttöliittymä on hieman muuttunut siitä, mitä aiemmin on esitetty (kuvassa 4), mutta ohjelman toiminta on edelleen sama. Oikealla ylhäällä olevasta nappulasta avataan video, mikä halutaan käydä läpi. Videotiedosto etsitään oikeasta kansiosta, jonka jälkeen ohjelma aukaisee ikkunan, johon syötetään ottelun

joukkueet, sekä kokoonpanot. Tämän jälkeen ohjelma aloittaa automaattisesti toistamaan videota ruudulla. Videon toistamiseen on tehty yksinkertaiset näppäinkomennot. Välilyönti -näppäimellä tauko ja nuolinäppäimillä kelaus eteen- ja taaksepäin. Mediasoittimessa on myös näppäimet, joita voi käyttää hiirellä, mutta niiden käyttämiseen ei ole yleensä tarvetta. Näppäin toiminnot ovat helpompia ja luontevampia käyttää, sillä ohjelmaa käytetään pääosin näppäimistöllä.

Oikealla alhaalla olevassa tekstikentässä näkyy kansio mihin ottelun tiedot tallennetaan. Kansion pystyy vaihtamaan, mutta oletuksena kansio on sama kansio, missä videotiedosto on. Näin ollen sitä ei yleensä ole tarpeen vaihtaa, sillä toinen ohjelma lukee tekstitiedoston samasta kansiosta missä videotiedosto on, joten ne on hyvä tallettaa samaan kansioon.

KUVA 8. Kuva käyttöliittymästä ajon aikana.

Kuvassa 8 on esitetty tilanne, joka halutaan tallentaa. Kuvassa oikealla on kuva pesäpallokentässä, mihin merkataan kyseinen tilanne. Tässä tilanteessa ykköspesällä on etenijä ja lyöjä lyö lyönnin nuolen suuntaan. Ohjelma lukee kuvasta automaattisesti tilanteen, lyöjän, lyönnin suunnan jne. ja tallentaa ne oikeaan paikkaan. Näin ollen käyttö nopeutuu, kun käyttäjä voi seurata peliä ja käyttää näppäimistöä tilanteiden merkkaamiseen. Aiemmassa versiossa hiirellä käytettäessä joutui videon pysäyttämään useaan kertaan, jotta kaikki tiedot sai kirjattua ylös, mutta tässä versiossa näppäimistön käyttö nopeuttaa huomattavasti ottelun läpi käymistä.

5.2 Videoiden katselu

Videoiden katseluun tehtiin toinen ohjelma, jolla voidaan vain katsoa otteluita. Ohjelman käyttöliittymä on hyvin pitkälti samanlainen, mitä on esitetty kuvassa 5, mutta muutamia pieniä muutoksia tehtiin. Ohjelman toiminta on kuitenkin samanlainen mitä aiemmin on kuvattu.

KUVA 9. Videoiden katseluun käytetyn ohjelman käyttöliittymä

Ylläolevassa kuvassa on videosoittimen päänäkymä, joka on myös hyvin pelkistetty, kuten editori ohjelma. Graafiseen ulkoasuun ei ole panostettu, mutta sen käyttäminen on yksinkertaista. Oikealla ylhäällä olevasta nappulasta avataan video, jota halutaan katsoa. Sen jälkeen ohjelma listaa kaikki kyseisestä pelistä tallennetut tilanteet listaan. Ohjelmassa voi valita kummanko joukkueen tilanteita katsoo ja vetovalikosta voi valita yhden lyöjän tilanteet. Näin ollen yhden lyöjän kaikki tilanteet yhdestä ottelusta on helpommin katsottavissa. Listaan on myös tehty järjestely toiminto jolloin otsikkoa painamalla voi muuttaa listan järjestystä aakkosjärjestyksen mukaan joko nousevaksi tai laskevaksi. Listasta nimeä klikkaamalla ohjelma toistaa videon kyseisestä tilanteesta.

6 POHDINTA JA JATKOKEHITYS

Projektin tarkoituksena oli tutustua C# ohjelmointikieleen, sekä tehdä ohjelma, jolla otteluvideoita voitaisiin leikata pienemmiksi osiksi myöhempää tarkastelua varten. Ohjelmasta saatiin loppujen lopuksi toimiva versio valmiiksi, jota päästiin hyödyntämään omassa käytössä. Ohjelmaan kuitenkin täytyy tehdä vielä parannuksia, jotta käyttäminen on sujuvaa ja nopeaa. Ensimmäisenä tietojen tallennus pitäisi saada tietokantaan, jolloin niiden järjestely olisi selkeää, sekä niitä ei olisi sidottu yhteen koneeseen. Näin ollen kaikki videot ja tiedot löytyisivät tietokannasta, joten tekstitiedostoja ja videoita ei tarvitsisi ladata omalle tietokoneelle vaan ne olisi helposti myös muidenkin saatavilla. Toinen iso kehitettävä kohde olisi käyttöliittymän parantaminen. Varsinkin graafiseen ulkoasuun täytyisi tehdä parannuksia, joihin tämän työn aikana ei ole ehditty panostaa. Paljon parannuksia käyttöliittymään saatiin tehtyä projektin aikana, mutta käytettävyys voisi olla parempi. Myös muita pienempiä parannuksia, virheenkorjauksia ja mahdollisesti lisäominaisuuksia tullaan ohjelmaan tekemään myöhemmin.

Pienillä muutoksilla ohjelmaa voisi käyttää mahdollisesti myös muissakin urheilulajeissa, sillä ohjelma yksinkertaisuudessaan tallentaa dataa joita videoista merkataan. Vaikka ohjelmaa voisi mahdollisesti hyödyntää muussakin käytössä, kyseistä ohjelmaa tullaan jatkossakin käyttämään vain omassa käytössä.

Koska C# on hyvin samankaltainen kieli kuin C++ ja Java, kynnys kyseisen kielen oppimiseen ei ole kovin suuri, sillä perusrakenteet ovat tuttuja. Kuitenkin kielen syntaksi, valmiiden ohjelmakirjastojen käyttö, sekä käytännössä hyödynnettävän tietokoneohjelman tekeminen on ollut uuden oppimista. Mielestäni tässä työssä itse uuden ohjelmointikielen oppiminen ei ole ollut tärkein asia, sillä ohjelmointikieliä ja eri työkaluja on niin monia. Siksi uskon, että tiedonhaku ja niiden hyödyntäminen, ongelmanratkaisu, sekä uusien asioiden sisäistäminen ovat tärkeimpiä taitoja, joita tämä työ on kehittänyt tulevaa työuraa ajatellen. Projektin aikana on myös huomannut, kuinka tärkeää hyvä suunnittelu ja testaus ovat ohjelmistoa kehittäessä.

LÄHTEET

Microsoft developer network. Introduction to the C# Language and the .NET Framework. Luettu 21.09.2016
https://msdn.microsoft.com/en-us/library/z1zx9t92.aspx

Moghadampour, G. 2009. C#-ohjelmointi. Docendo.