Final thesis

A Software Framework for Facial Modelling and Tracking
by
Mattias Strand

LiTH-ISY-EX--10/4098--SE

2010-03-05
Final thesis

A Software Framework for Facial Modelling and Tracking

by

Mattias Strand

LiTH-ISY-EX--10/4098--SE

Supervisor : Jörgen Ahlberg
   Dept. of Electrical Engineering
   Linköping University

Examiner : Robert Forchheimer
   Dept. of Electrical Engineering
   Linköping University
Abstract

The WinCandide application, a platform for face tracking and model based coding, had become out of date and needed to be upgraded. This report is based on the work of investigating possible open source GUIs and computer vision tool kits that could replace the old ones that are unsupported. Multi platform GUIs are of special interest.
Acknowledgements

At last! Thank you my apple angel.
Contents

1 Introduction ............................................. 1
  1.1 Purpose .................................................. 1
  1.2 Background ............................................. 1
  1.3 Methodology ............................................ 2

2 WinCandide ............................................... 3
  2.1 Functionality ........................................... 3
  2.2 Core elements .......................................... 3
  2.3 Existing architecture .................................. 5
  2.4 File structures ......................................... 5

3 The new application ....................................... 9
  3.1 Requirements ........................................... 9
  3.2 New core elements ..................................... 9
    3.2.1 Graphical user interface packages ............... 10
    3.2.2 Image and math packages ....................... 11
    3.2.3 The new configuration of classes .............. 11
  3.3 File structures ......................................... 13
  3.4 Implementation ....................................... 13
    3.4.1 Implementing the new GUI ....................... 13
    3.4.2 The tracker API .................................. 13
    3.4.3 Functionality review ............................. 14
    3.4.4 Functionality implementation ................ 14
4 Results

4.1 The finished application ........................................... 18
4.1.1 Platform independency ......................................... 18
4.1.2 Updating WinCandide to available new standards .... 18

5 Conclusion

5.1 Discussion .............................................................. 19
5.2 Software packages ..................................................... 19
5.2.1 FLTK ................................................................. 19
5.2.2 OpenCV .............................................................. 20
5.2.3 GNU license software ...................................... 20
5.3 Future work ............................................................. 20

Bibliography

Appendix

A Appendix

A.1 Specification of the tracker interface ...................... 25
Chapter 1

Introduction

1.1 Purpose

WinCandide is a software application for modelling, tracking and synthesising of a human face. The purpose of this master thesis is to investigate and convert the WinCandide application to an application that is up to date and possible to compile on as many as possible of modern operating system platforms, emphasis will be on the current Windows and Linux platforms. Also to add a possibility to add face trackers to WinCandide using an API.

The application will also undergo a thorough investigation and remake of the code structure and code to meet new code standards and conventions.

1.2 Background

The WinCandide application was developed by Jörgen Ahlberg. He started to develop the application in 1999 as a part of his PhD studies. WinCandide was developed as a part of model based coding and the reason was to develop a platform for tracking the movements of a face and then transfer and apply them to a realistic and moving model on another platform. This way you don not have to transfer the complete video stream between platforms. He has received attention about the application from many dif-
ferent places. Realising that it now lacks compatibility with most modern platforms a master thesis project was initiated to make it regain its past glory and usability.

1.3 Methodology

The intention is to find a new graphical user interface, preferably open source, by searching the market. Also to update all the bindings to packages the application relies on, and to replace packages that are out of date. The new application will be constructed with the new platform and all the new requirements found during the process of evaluating the old application.

A free, available face tracker will be used to build and test the tracker interface that will be available for testing of face trackers.

As a final step compilations on at least three different platforms will be conducted in order to test if a truly platform independent application has been achieved.
Chapter 2

WinCandide

2.1 Functionality

An important purpose of the WinCandide application is to be a base for testing of face trackers. The application loads a background image and on top of that it loads a Candide [1] face model which can grab the underlying texture. The application can load a sequence of images, step forward, backward and animate the sequence.

Video sequences from camera and from file can be grabbed and displayed as an underlying texture and on top of this a tracker can be applied.

2.2 Core elements

The existing application is written in C++, it is developed and compiled in version 6 of Microsoft Visual Studio and uses MFC (part of the code is slightly modernised to use STL) as a result it is only possible to compile and run on the Windows 98 operating system.

The application uses DLL files frequently and incorporates functionality from the Intel Image Processing Library, the Intel Signal Processing Library and the Intel JPEG Library. Those libraries has become out of date and appropriate replacements must be found. The application uses Video for
Figure 2.1: The Candide face model version 3

Figure 2.2: Captions from the old WinCandide application
Windows to capture the frames from video that are used for tracking, and that too has become out of date and is highly platform dependent.

2.3 Existing architecture

The existing application consists of six DLL files:

- AppModel - tying it together
- eruFace - polygon face model
- eruMath - math computations
- FapStream - stream for facial animation parameters
- GLRender - OpenGL functions
- eruImg - image processing

The main application WinCandide has about 14 classes which are all MFC specific classes which implement the existing GUI.

2.4 File structures

Jörgen Ahlberg has himself constructed some of the file structures that are used. The base file is the Candide model and this is stored in a wire frame model file (.wfm) which was developed by Märten Strömberg for the first Candide software ”xproject” in the 80’s. This file includes:

- A vertex list
- A face list
- A list of animation units (AU)
- Facial animation parameters (FAP)
- A list of shape units (SU)
2.4. File structures

- Animation parameters
- Shape parameters
- A global motion

The file structure Jörgen has introduced separates the animation parameters, the shape parameters, and the global motion from the wire frame file. The first file is the animated model file (.amd) which includes:

- Animation parameters
- A global motion
- A link to a static model file (.smd)

The static model file (.smd) includes:

- Shape parameters
- A link to a Candide3 wire frame model file (.wfm)

The Candide3 wire frame model file (.wfm) contains the remaining parts of the original file:

- A vertex list
- A face list
- A list of animation units (AU)
- Facial animation parameters (FAP)
- A list of shape units (SU)

It also includes headlines for the parameters in the animated model file and static model file but only for backward compatibility reasons.
Figure 2.3: Layout of the file structure
2.4. File structures
Chapter 3

The new application

3.1 Requirements

At the start of the project an informal Software Requirements Specification was made and a list of requirements was introduced:

- Open source
- Possible to compile on as many platforms as possible
- Possible to connect your own tracker
- Chosen packages still being supported
- Contain a face tracker API

3.2 New core elements

With the requirement that the new application should be possible to compile on as many operating systems as possible the starting task was to find a new graphical user interface (GUI). Another than the one provided by Visual Studio because that limits the application to Windows.
3.2. New core elements

A first idea was to change the implementation language to Java and thereby getting a GUI and a multiple platform application. This idea however was discarded due to the fact that the application uses OpenGL and the Java OpenGL is not as good as the original OpenGL package and keeping the implementation language also simplifies the new implementation. As a result C++ was kept as the implementation language.

3.2.1 Graphical user interface packages

A search on the Internet resulted in a number of candidates to a new graphical interface package. First some OpenGL frameworks and Graphic User Interface frameworks:

- freeglut - The Free OpenGL Utility Toolkit [3]
- GLFW - OpenGL Framework [5]
- GLUI - OpenGL User Interface Library [7]
- NUI/NGL - C++ API, multiplatform, 3D hardware accelerated GUI[8]
- FLTK - Fast Light Toolkit [9]

And two more or less commercial Graphic User Interface frameworks.

- Presenter - A graphical user interface development environment [10]
- Qt - A cross-platform application and UI framework[11]

An informal requirement was that the package to use was still being supported. The OpenGL frameworks probably would have worked just as fine as the more advanced GUI frameworks but due to simplicity and time the choice was to use one of the GUI frameworks. The result then came down to two packages, FLTK and Qt’s GNU-license version. Finding the FLTK package satisfactory and even containing a graphical editor for windows and menus, FLUID, the choice was made to use this package.
3.2.2 Image and math packages

As mention the Intel Image Processing Library, the Intel Signal Processing Library and the Intel JPEG Library have become out of date but they have all been superseded by the OpenCV [12] package which fits all the requirements and even replacing the Video for Windows functionality and thereby ensuring platform independency for that part.

3.2.3 The new configuration of classes

Because the new application will consist of less functionality than the previous version, e.g. a reduced face model and no built in math functions, the complete structure of the model classes will be remade into just a model class and a Glmodel class that will deal with the drawing of the model.

The GUI will contain the following classes:

- Wincandide - main application
- Wincandidesettings - dealing with all the settings
- GlView - the OpenGL view on which the model is rendered
- activeModelSettingsDlg - a dialog to set model parameters
- editAnimateSequenceDlg - a dialog to load and set animation details
- CaptureSettingsDlg - a dialog to load movies
- FL_Native_FileChooser - platform independent file chooser
- recieveDlg - a dialog to connect to another session
- globalDlg - a dialog for global settings
- trackerDlg - a dialog to set tracker settings

The rest of the application will consist of a stripped number of classes from the original application.
3.2. New core elements

![Diagram of GUI classes]

- WinCandide
  - WinCandidesettings
  - GLView
  - activeModelSettingsDlg
  - editAnimateSequenceDlg
  - CaptureSettingsDlg
  - FI_Native_File_Chooser
  - receiveDlg
  - globalDlg
  - trackerDlg

Figure 3.1: Layout of the GUI classes
3.3 File structures

All the file structures used for the old application will be kept. As a result of using OpenCV as the image handler a lot of previously unsupported image formats are now supported. Available formats are:

- Windows bitmaps - BMP, DIB
- JPEG files - JPEG, JPG, JPE
- Portable Network Graphics - PNG
- Portable image format - PBM, PGM, PPM
- Sun rasters - SR, RAS
- TIFF files - TIFF, TIF

Exporting from the new application can be done to all the above formats by defining the equivalent file ending while saving. Exporting can also be done to a VRML format.

3.4 Implementation

3.4.1 Implementing the new GUI

First a complete investigation of the existing application was made and all the menus and dialogs were documented as part of the informal requirements. A check of what code standards that are applicable were made and resulted in a document called "C++ Standard Library Coding Style Guidelines" [13]. This is the common standard used in libstdc++ files and this probably helps in getting a platform independent application. Then the work to transfer the application to FLTK begun.

3.4.2 The tracker API

The tracker interface will be a simple interface however due to the fact that a lot of time was spent trying to get the face tracker to work in WinCandide the API part was dropped and resulted in a class interface which is stated in the appendix.
3.4.3 Functionality review

When the old GUI structure had been investigated a discussion of what functionality should be kept, what functionality should be altered and what functionality should be excluded was held. The existing application was in some ways tailor made to some specific users and the conclusion was to drop all that functionality. Also, all the code concerning tracker specific image tasks could be dropped due to the fact that the trackers themselves should be responsible for implementing their own image processing. The new GUI was decided to be able to load and save images, step images in a series of images, run movies for tracking and incorporate a tracker for tracking and also to connect to another session over network.

3.4.4 Functionality implementation

The process of getting the functionality working in the new environment was started and the first step, to load an image and present it on the background was completed. Because OpenCV uses the old IplImage structure from the Intel Image Processing Library (IPL) and has an internal image loader function the task was fairly easy. The existing application does all the image loading at a basic level so a lot of now unused image loading code was cut out.

When the basics was working the next step was to set up loading of the Candide model and getting the controls that changes the static parameters to work. The controls changes the scale, rotation on three axis and translation on the x and y-axis and to get the indicators on the bottom of the frame to display correctly.

Now there was an effort done on getting the application to be able to communicate with a client application via the network and thereby creating a possibility to render on one machine and sending that to another machine. A feature used for model base coding. This task however is very platform dependent due to standard OS hardware handling differences so at the moment this is only implemented on the Windows platform.

Focus was now on getting the video capturing to work. OpenCV nicely implements this part, actually for Windows it uses Video for Windows like the old application. Depending on which codecs that are installed on the
computer a range of formats can be used. At this time the functionality to grab frames from a web camera was established. It works fine although a bit slow, mostly because the resolution is so high, $640 \times 480$. Notably, if camera -1 is chosen, option is given to OpenCV to choose the first available camera.

The last and big part of the application was to get the facial tracking to work. A master thesis was done on face tracking in 2007, "A feature based face tracker using extended Kalman filtering" [14] was perfect to try to incorporate into the application because the source code was freely available. Because the facial tracker is a working application in itself and because FLTK has a limit in where you can access OpenGL functions the application graphic functions had to be extended to deal with tasks from the face tracker. A function for drawing a mask was developed. A tracker adapter class was built to be the natural interface to the trackers.
3.4. Implementation
Chapter 4

Results

A functioning application has been achieved. Due to too lack of time, testing on other platforms than Windows has not been conducted and a lot of work to achieve an application with higher finish remains. Functionality like the toolbars from the original application has not been implemented which would have been nice. The application can load a face model and a background. It can produce a texture from the background and apply it on the model. It can save background images and export in VRML format. It can take a sequence of images and animate them and also take a video feed from camera or a file and display.

Figure 4.1: Captions from the new WinCandide application
4.1 The finished application

4.1.1 Platform independency
One of the goals was to achieve as much platform independency as possible however lack of time made it impossible to test this, my belief is that in order to run this on another platform a bit of tweaking is necessary.

4.1.2 Updating WinCandide to available new standards
Another goal was to update the application to available new standards, as the investigation showed some packages had to be updated, in Windows updating is satisfactory however a compilation on another platform or with another compiler would possibly have revealed more information regarding updating. Because the incorporated face tracker never run in a satisfying way there is no measured performance.
Chapter 5

Conclusion

5.1 Discussion

The work started out with finding a new package for the user interface and that proved to be easy. In fact there was only two good candidates, FLTK and Qt. After choosing FLTK the implementation was straight forward, but when the work continued with adapting the code to OpenGL FLTK started to show some disturbing flaws. The OpenGL handling made the final step, incorporation of the face tracker, harder then expected. Getting to grips with the face adapter was also harder than expected.

5.2 Software packages

5.2.1 FLTK

FLTK is Open source and freeware and therefore lacks the usual finesse that a commercial software has, but it works satisfactory. The version used is 1.1.7 which was the release that was stable at the start of this project but there was a 1.1.x and a 2.0.x but unstable releases available. A stable 1.1.8 release emerged during the project but no effort was done to switch to this package. While incorporating the tracker it was discovered
that FLTK only allows drawing access to the OpenGL window in a draw function which altered the chosen way of dealing with the tracker. If this limitation had been known at the beginning of this thesis Qt would have been chosen instead.

5.2.2 OpenCV

OpenCV is, as the name says, open source. It is a good package and a perfect replacement to many of the packages in the system because it actually is the successor. OpenCV makes all the image loading and saving easy because it wraps a lot of the most used images formats in the internal image loader which functions in a nice way and separates that part from the application.

On the other hand a flaw or feature in the function that creates a new image was discovered. The problem is that, for some reason, in small images memory is allocated but not reset to zero which happens in images smaller than $417 \times 417$ pixels.

5.2.3 GNU license software

The experience of using GNU licence software is both good and bad, the good parts are of course that it’s free, that you can change it and that you have complete access to the source code if you like to dig deeper into what really is going on.

The bad parts is that it might contain a lot of bugs and undocumented features. The bugs and features are often not likely to be fixed in a near future and you can not rely on that it happens in a systematic way.

5.3 Future work

Because no matter what project you work on there will always be more work than you can do in the assigned time. Thus there are a number of issues that are not completely solved. The biggest is concerning the FAP streams that need a rebuild and incorporation into the system.
A reevaluation of the eruImage::Image class, the class dealing with the polygons of the face, should be done because of the new OpenCV package and due to the fact that a lot of functionality has been dropped and also the potential in OpenCL.

The OpenCV package displayed some flaws but still is very good and is the given choice for the face tracker. On the other hand, there exist other potent C++ image libraries but without the video capability, a different image solution might be worth looking into.

The FLTK package required some tweaking to get all the functionality to work exactly as planned. The decision to use it over Qt might be worth to reconsider or even to use the FLTK 1.3 or 2 package.

The face tracker never started to work as it should, i believe that the main problem was the access to the OpenGL context in FLTK, I also suspect that FLTK has some bugs concerning its OpenGL handling and the redrawing of the context i.e. the double buffering and the calling of the redraw function. The error in the tracker that I tested manifested it self after a few picture updates so the error might have been introduced by me during the incorporation of the tracker. Although I was careful I may have introduced some errors not knowing it and not realising what was wrong.

Due to lack of time no toolbars have been implemented and that would be nice. The network functionality is also a bit untested and need some serious attention to get to work on other platforms than Windows.

Recently OpenCL was developed which runs on CUDA GPUs. OpenCL makes it possible to use more power of the GPU for calculations and image calculations are perfect for the GPU.

A reimplementation with OpenCL and Qt would be my choice today.
Bibliography


[10] Presenter - A graphical user interface development environment.
    http://www.presenter5.com/.


[12] Open Source Computer Vision library, OpenCV.

    http://gcc.gnu.org/onlinedocs/libstdc++/17intro/C++STYLE.

Appendix A

Appendix

A.1 Specification of the tracker interface

class Tracker
{
    public:

        //Constructor
        Tracker();

        //Destructor
        ~Tracker();

        //Initiate tracker with a new frame
        virtual void init(GlView* view,
                          double gl_width,  //width of opengl buffer
                          double gl_height,  //height of opengl buffer
                          const IplImage* frame){};

        //Update tracker with a new frame
        virtual void update(const IplImage* frame){};
}
A.1. Specification of the tracker interface

```c
GlView* theView;
GLModel* theModel;

void UpdateModel(double rotX, double rotY, double rotZ, double transX,
                  double transY, double transZ);

void DrawMask(CvMat* maskVertices, CvMat* maskTriangles,
               CvMat* maskTriIndex);

void GetImage(IplImage* frame);

void SaveStates();
void RestoreStates();

bool m_bBackground;
bool m_bModel;
bool m_bVertices;
bool m_bTexture;
bool m_bMask;
bool m_bStatesSaved;
```

Overload the initiation and update function with the ones from your personal tracker class, the program will use these to feed the initiate and feed the tracker with images to track.
The WinCandide application, a platform for face tracking and model based coding, had become out of date and needed to be upgraded. This report is based on the work of investigating possible open source GUIs and computer vision tool kits that could replace the old ones that are unsupported. Multi platform GUIs are of special interest.
Copyright

Svenska
Detta dokument hålls tillgängligt på Internet - eller dess framtida ersättare - under 25 år från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en attläsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/

English
The publishers will keep this document online on the Internet - or its possible replacement - for a period of 25 years from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for your own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its WWW home page: http://www.ep.liu.se/

© Mattias Strand
Linköping, March 17, 2010