UNIVERSITE DE LILLE

IUT A DE LILLE
DÉPARTEMENT INFORMATIQUE

INTERNSHIP REPORT

ERASMUS+
TRONDHEIM - NORVÈGE

CAMERA MANAGER - DEVELOPMENT OF A NEW SOFTWARE MODULE FOR INTEGRATION OF COLOR CAMERA 3D Motion Technologies AS

Presented by Hugo Fournier

Knowledge for a better world

3D motion analyses

Supervised by :

Mr Jan H. Nilsen – NTNU tutor
Mr Tomas Holt – NTNU tutor
Mr Patrick Lebegue – IUT A tutor

1

1 Hugo Fournier – DUT Report 2019

TABLE OF CONTENTS

Acknowledgments

I would like to express my very great appreciation to the following persons and institutions who made my internship abroad possible :

First, my tutors Pr Jan H. NILSEN, Tomas HOLT and Alexander HOTL for giving me all the resources I need to lead this project to his end and helping me to achieve what I wanted to do.

Second, Mr Patrick LEBEGUE to make this trainee the most supervised as possible.

Last, my colleagues from the University de Lille, Gauthier and Adrien, who were the best working partners during these three months.

I also would like to thank :

The international resources office of NTNU and the IUT A de Lille for their help through this internship and their kind welcoming.

All of my teachers from the Université de Lille who gave me all the IT skills to be able to do such a project.

Finally, a huge thanks to Sara, Andrea, Olav, Benedikte, Camilla, Edvard, Hannah, Jorgen, Kristine, Markus, Sanja, Torstein and Vemund, the norwegian students who make this Erasmus experience as best as possible.

Introduction

The beginning of my trainee was the 2nd of April and it ends the 20th of June. The main subject was clearly set the first time I went to visit my tutor. He clearly said that I, with my two other colleagues from the University of Lille, will have to set up new cameras with the software application they currently use.

The main application, called Camera Manager, has two principal features. The first is to be able to record a camera stream with the cameras connected to the PC and the second to make the cameras detecting motion sensors when someone moves.

Camera Manager is useful for medical purposes and for my tutor's company, 3D Motion Technologies AS, wich is a company focuses on the Motion capturing systems and the way we can do it. I will talk about the compagny details later in the report.

The trainee organization was very simple but efficient, we had a meeting once per week with our tutor or one of his colleague, that was at this time we have to show, Gauthier, Adrien and me, what we have done during this week. It was primarily a quick demo of some features we implemented and there was also a time to speak about our difficulties and how we can figure them out.

I was working at my student room most of the time because there was no dedicated workspace but some time i was working with Gauthier and Adrien and we did some meeting just me and them to be sure that we were on the right way and to divide the tasks properly.

Firstly, I will talk in greater detail about 3D Motion Capture AS and of my life as an Erasmus student in order to state differences between Computer Science in France and in Norway.

Secondly, I want to focus on what I have done as a programmer, the different steps to achieve the main application, my failures and my successes.

To conclude, a summary of what I have learned professionaly and socially speaking.

3D Motion Technologies and her line of business

1 The sector

1.1 Presentation

3D Motion Technologies is a company working on the development of motion capturing system, but what is motion capture ?

Motion capture is the process of recording the movements of object or people. We know motion capture from long time espacialy for the entertainment in video games. The first console to use motion capture was the Wii from Nintendo. From that, a lot of video games constructor used this technology like Sony with the PS move and Microsoft with the Kinect.

Nowadays, the motion capture technology adds another dimension with the virtual reality headset such as the Oculus Rift or the HTC Vive.

Moreover, this technology takes now a huge part of medical purposes and open a new way of doing things.

1.2 Economical Sector

3D Motion Technologies focuses on recording humans, animals and industrial objects movements in complex environments using 8 cameras.

Their technology is described as a high precision, low-cost portable system. I think that is the main thing to remind because motion capture does not appeare as an accessible thing.

When it is used for entertainment, it does not need to be high performance so it is accessible.

Otherwise, it is very expensive and used only for films. 3D Motion Technologies presents here an accessible way to use an expensive technology in order to form part of an approach of democratization of this main technology.

In my opinion, that's a good thing for technology but also for medical purposes to be able to access as such a technology, this is no more restricted to a tiny part of the industry and it's open to many sectors.

5 Hugo Fournier – DUT Report 2019

2 *The Enterprise in comparison with her sector*

2.1 3D Motion Capture AS Background

3D Motion Capture is first and foremost a research project that is conducted at the Norwegian University of Science and Technology. It began when they participated on an entrepreneurship schedule named « Take off » at NTNU in 2012.

The same year, they have been awarded with the HIST/NTNU Entreneur price and one month later, the company was founded. One year later, 3D Motion Capture had commercial rights to all software developed. One year later, in 2014, the company were granted from Innovasjon Norge.

Through the next years, they were a new board and co-worker who joined the project.

2.2 3D Motion Capture AS Now

The company currently numbers some 6 persons who are involved into this project. Jan Nilsen who is my tutor is the current CEO and Thomas Holt who also supervised my trainee is the vice CEO. They are both founder of the company with Else Lervik, Mildrid Ljosland and Grethe Sandstrak. The last member is Svein Tryggestad who is the Chairman of the Board.

From what I have seen when we had a quick review the first day of my trainee, they are currently working with motion capture for medical purposes but that is not the main subject they are into. On the company's youtube channel, we can see they previously worked on Football tracking and Game Simulator with Oculus Rift.

Now, the main goal is to allow people in rehabilitation to use Motion Capture as an alternative and playful solution. It can be the beginning of revolution for these people in rehabilitation and for the medicine in general.

Through this background, we can see a huge evolution just has it is the motion capture.

6 Hugo Fournier – DUT Report 2019

The Trainee Conditions

1 My life as an Erasmus student

Before I went to Norway and especially Trondheim, I had the chance to read some last years reports made by the other students from Lille. I was aware that I would have to work at my flat most of the time. It was difficult to think about it until you are in the situation because you really have to organize your time to make it worth. With one meeting per week, there is not a lot of time with your tutor and it is easier to go to the wrong way.

I had these feeling at the beginning that I was lost because the DUT is a school training which is very well oversee and if I could give an advise to the next students, make sure that you have a good working method and work continually through the trainee because three months is not that long.

Otherwise, Trondheim is a great student city, you can go through the town easily because it is well-deserved by buses. The campus is huge, there is everything you need to work, you can also book a room to make your own meetings. I think this is a dream place for student life.

The main reason I went to Norway beyond the fact that I was aware of the main project was to improve my English speaking. I followed the advise from my English teacher and it was the best thing to do. It was really satisfying to speak English with the Norwegian people because I felt that they make effort to make you feel comfortable with the language. With my roommates, it was really easy to communicate and to make me understandable by the others. I can tell that I have made important improvements in terms of vocabulary but above all I am no more hesitant to speak to the people and that part will be very useful for my future jobs or travels.

To conclude, I think Trondheim is the ideal place for french students to improve his English speaking and to discover the student life of Norwegian students. If I could mention a defect it would certainly be the prices which has everyone now are very high. Nevertheless, after two weeks here, you forgot about it because there are many benefits to be here that it is worth it.

I talked about my life as an Erasmus student, it is also important for me to speak about the learning of Computer Sciences here.

7 Hugo Fournier – DUT Report 2019

2 Studying Computer Science in Norway

In France, we are studying Computer Science like another subject area. You have conduct, practical and tutorial classes. Of course, the main classes are the practical one because you need to be on the computer as much as possible. My point is that this is a way of teaching, but I was interested to go to Norway also to see if the way of teaching Computer Science is different.

I know that this is a bit difficult to compare because I was here for a trainee but I had the chance to be with students who are following Computer Science studies in my building.

I talk with one of them, and she had a project on machine learning, this is the kind of project that we will never had in France because we are focused on the basis of programming but I thought that is fascinating to be able to do machine learning at the university.

Moreover, and this is the main difference for me, studying Computer Science in Norway means home working most of the time. There are also a lot of working and meeting room inside the campus, there are in free access for every student and that's very efficient for co-working during project like the trainee I had.

The only time you spend with your tutor is during the weekly meeting that's at this moment you can share what you have done and tell also the difficulties encountered. This is quite difficult to be familiar with this at the beginning because in our studies, we saw our teachers everyday and we are supervised.

I like both of this way of studying because on the one hand, that is convenient to work whenever you want with your own pace of work and on the other hand, to be supervised make sure that you have a strict schedule.

For me, despite I like the way of teaching in France, I think studying Computer Science will be more like in Norway in the future. There are already a lot of training you can make online to have diplomas and there are a lot of countries which have already adopted this way of studying.

8 Hugo Fournier – DUT Report 2019

My trainee

1 The Langage and the IDE

1.1 The Langage

Illustration 1: C++ logo

The project which is named « Camera Manager » has been made with C++. C++ is a general-purpose object-oriented programming language developed by Bjarne Stroustrup of Bell Labs in 1979. C++ was originally called 'C with classes,' and was built as an extension of the C language. It is considered a mid-level programming language, combining some elements of low-level programming languages, such as the need to learn memory management, with high-level features. Because of this, C++ is considered quite a complex language—in comparison to languages such as Python you need to know quite a bit more before you can create your first truly useful programs.

1.2 The IDE

For this project, the previous students used Qt Creator IDE to make the GUI so I used it too because it is made for c++ application. Qt Creator is an integrated development environment that provides you with tools to design and develop applications with the Qt application framework. Qt is designed for developing applications and user interfaces once and deploying them to several desktop, embedded, and mobile operating systems. Qt Creator provides you with tools for accomplishing your tasks throughout the whole application development life-cycle, from creating a project to deploying the application to the target platforms.

Illustration 2: Qt logo

9 Hugo Fournier - DUT Report 2019

2 The Camera Manager Software

As I said in the introduction part, Camera Manager Software has two features. For my trainee, I only worked on the first one which is to be able to start a camera stream.

Illustration 3: Camera Manger GUI

There are three main overviews for the user which are Projects, Cameras and Trackpoint.

Projects allows the user to import directly his project with some images to inspect, it works like a tree overview as it is used in most of the IDE now.

Cameras allows the user to see which cameras are plugged into your computer and ready to use.

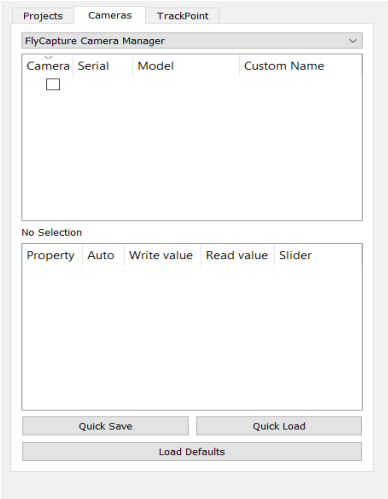


Illustration 4: Cameras Overview

10 Hugo Fournier – DUT Report 2019

You can also use a properties file to modify some cameras as you can see at the bottom.

When a camera is plugged in, the GUI is up to date.

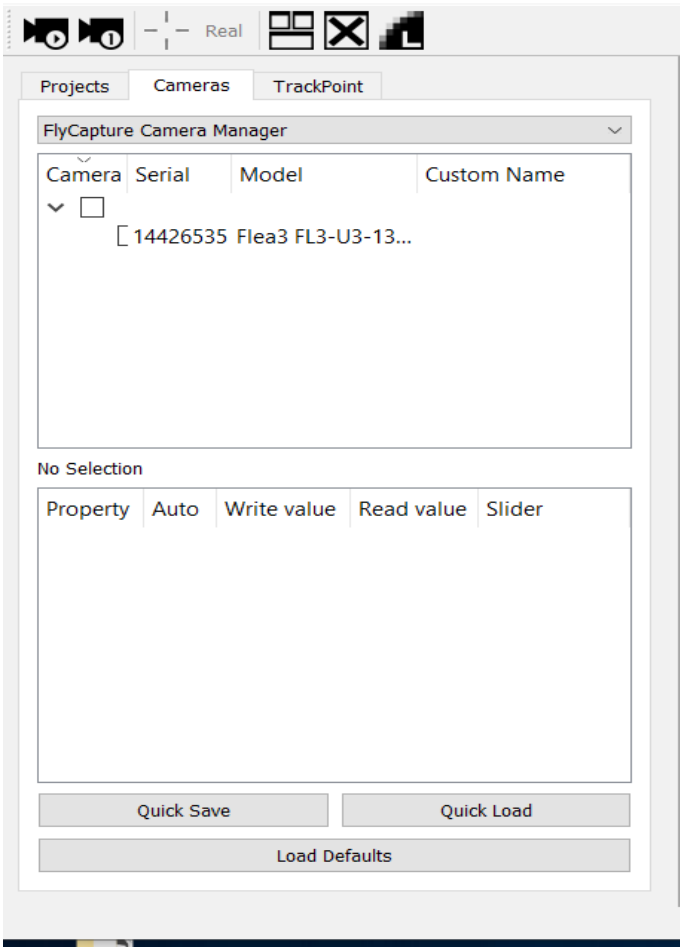


Illustration 5: Up to date GUI

The last one is the TrackPoint, this is the interface used for the motion capture recording but as I don't work on this part, I will just show the overview of it.

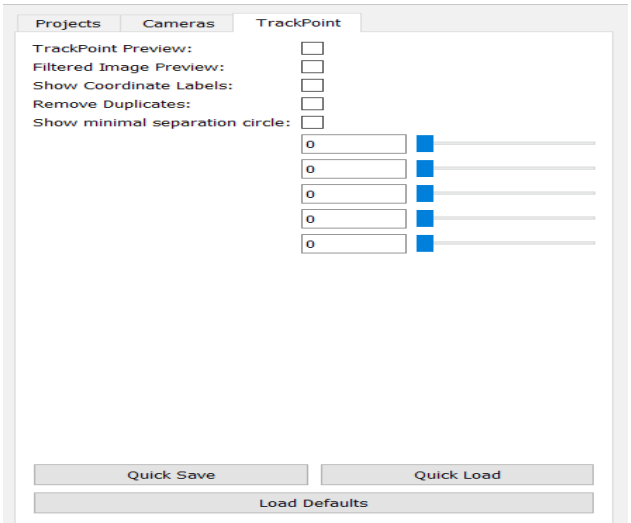


Illustration 6: Trackpoint Overview

11 Hugo Fournier – DUT Report 2019

To start a camera stream, because this is the main feature we are interested in, you have to select which camera you want.

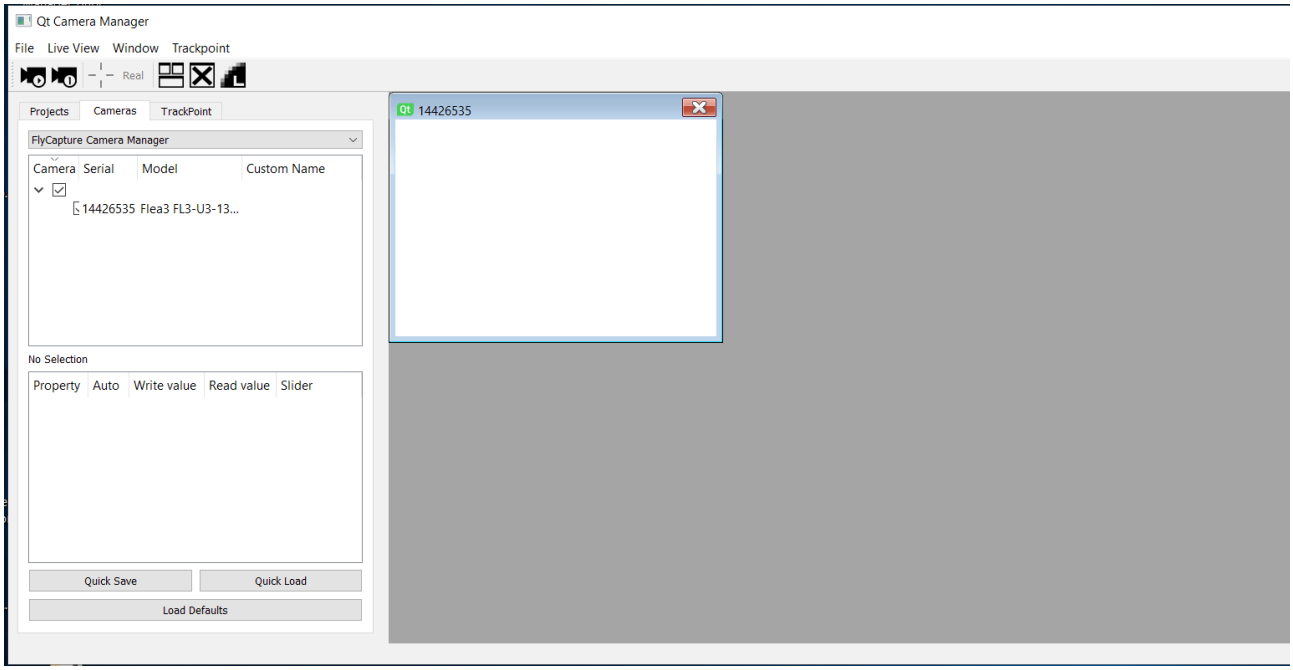


Illustration 7: Camera Selected in the GUI

On top of those overviews, there is the menu bar.

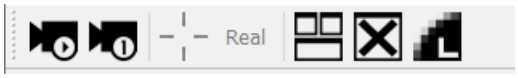
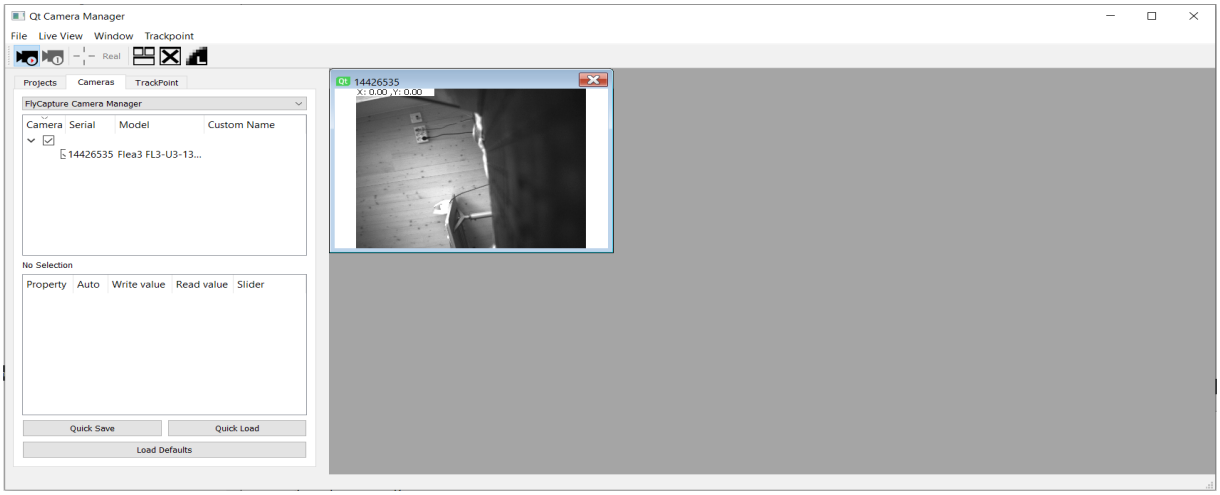


Illustration 8: Camera Manager MenuBar

The main button for us here is the first one. Once you click on it, it will start the stream.



Now, I will explain how we get through the re implementation of Camera Manager with the new API.

3 My tasks

3.1 Understanding of the code and the old API

The first two weeks were dedicated to the reading of the Camera Manager code and how the FlyCapture API works.

The project was written in C++ and this language was not familiar to me because I had never practicing it before. Thanks to the OpenClassRoom WebSite, I did a lot of tutorials, example codes and also learn how the Qt IDE works. Once I had a quick overview of the language and the IDE, I began to import the project into the IDE on my personal computer. I was working on macOS and the project was only compatible with Linux or Windows so I had to create a virtual machine in order to achieve this.

At first, I had a lot of issues trying to install the project on Ubuntu but Gauthier, who was trying to install it on Windows, hadn't this issues so I look at how he did it on my issues was from the Qt installation so after one week, it was completely done and I could move on to the next step.

I also had my first meetings with Gauthier, Adrien and our tutors. I said that the virtual machine was not really practical to work on so they gave me a pc with a dual boot so I could work on both Linux and Windows Operating Systems now.

The project was really consequent and it was the first time I had to face such a project. Thanks to the documentation from the previous students, I quickly understand that the project was divided in four parts.

The first one was dedicated to the main class and how the GUI was made. The second one explains how the camera implementation works, the different classes and functions. The third one shows how they managed to open contents like files, images with the different classes and functions. The last one described more the usage of OpenGL to display the image inside the label on the software.

Without all this documentation, I think it will have taken me two weeks more to understand all these classes. After that it was time for me to understand all the classes which were from the FlyCapture API because it was with this API that the camera stream could be displayed on the screen.

So now, let's take a look at **FlyCapture SDK**.

13 Hugo Fournier – DUT 2019 Report

The FlyCapture® Software Development Kit (SDK) provides a common software interface to control and acquire images for FLIR area scan USB 3.1, GigE, FireWire, and USB 2.0 cameras using the same API under 32- or 64-bit Windows or Linux.

So this API allows the user to fully control the cameras and that's useful for us because without this SDK, we couldn't start a camera stream.

When you download the SDK, it comes with a lot of example codes such as how to save an image, how to use the bus manager, how to handle camera acquisition and to handle multiple cameras too. It makes It was not really matter that I fully understand how the API works because I will only have to use this code and make it works with the one that I will write next.

After these two weeks, it was the time for me to move on the next part and start writing code lines with the new API.

3.2 Stand alone projects with the Spinnaker API

The new API is called **Spinnaker**. It is made by the same company, PointGrey, and this is in fact an improvement of FlyCapture for PointGrey new cameras.

The Spinnaker SDK is FLIR's next generation GenICam3 API library built for machine vision developers. It features an intuitive GUI called SpinView, rich example code, and comprehensive documentation designed to help you build your application faster. The Spinnaker SDK supports FLIR USB3, 10GigE, and most GigE area scan cameras.

This new SDK also contains example codes that I firstly got a look at. At the same time, I read a lot of the documentation in order to be able to try an example code myself. During my school formation, I worked with one or two API but it was with Java on a different IDE.

On Qt, it is not the same but I have an idea of what I should to thanks to the Camera Manager project. Once I successfully tested the example code, I began to write my first program with the Spinnaker API. My goal with it was to be able to see the camera stream with the API on a basic GUI.

14 Hugo Fournier – DUT 2019 Report

3.2.1 The first project

In order to be able to fully understand the new SDK, Gauthier, Adrien and me decided that it should be interesting that each one of us made his own stand-alone project to understand the functioning by themselves.

For my first project, the goal was to be able to start a camera stream. The GUI of this first project was really simple, just a label and a push button.

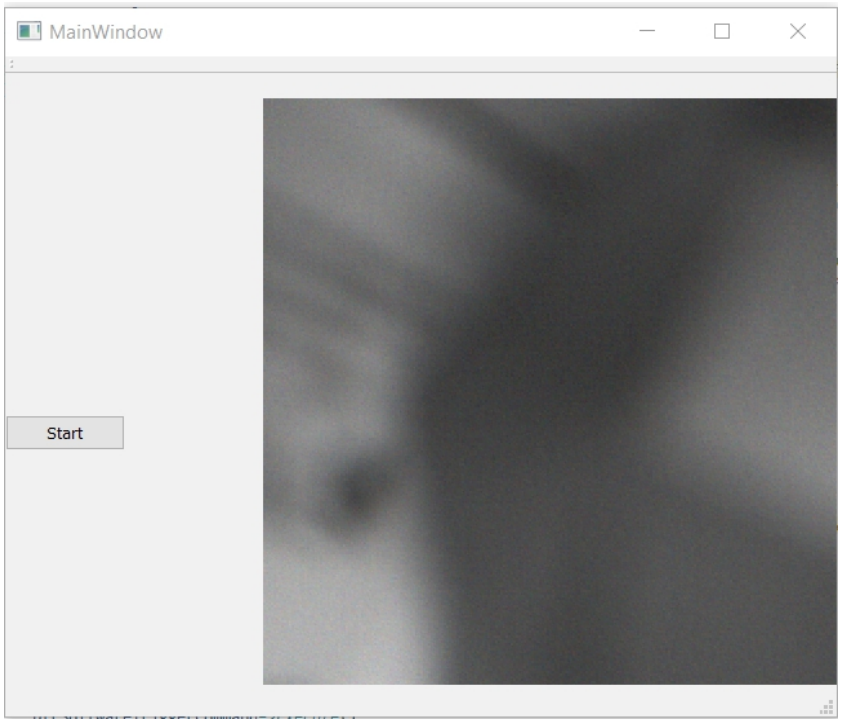


Illustration 9: GUI

To do that, my idea was to do a loop with the acquisition example code given. I knew that with this code, I could retrieve one image so with a loop, it will retrieve an infinite amount of images which was gonna be the camera stream.

I couldn't use a while loop because it was too quick for the images to be display so I used one of the specificities of Qt which are the signal and slot with a timer.

```
void MainWindow::on_pushButton_clicked()
{
    connect(timer,SIGNAL(timeout()),this,SLOT(launchEveryThing()));
    timer->start(20);

}
```

Illustration 10: Signal & Slot connection with a timer

[15]

15 Hugo Fournier – DUT Report 2019

Every 20 milliseconds, the slot « launchEverything » is called, which is a method that initialize the camera, retrieve an image and display it in the GUI.

```
void MainWindow::launchEveryThing(){

    int result = 0;
    // Retrieve singleton reference to system object
    SystemPtr system = System::GetInstance();     ⚠ instantiation of function

    // Retrieve list of cameras from the system
    CameraList camList = system->GetCameras();       ⚠ instantiation of funct

    unsigned int numCameras = camList.GetSize();

    // Finish if there are no cameras
    if (numCameras == 0)
    {
        // Clear camera list before releasing system
        camList.Clear();

        // Release system
        system->ReleaseInstance();

        cout << "Not enough cameras!" << endl;
        cout << "Done! Press Enter to exit..." << endl;
        getchar();

        exit(-1);
    }

    // Run example on each camera
    for (unsigned int i = 0; i < numCameras; i++)
    {
        cout << endl << "Running example for camera " << i << "..." << endl;

        result = result | RunSingleCamera(camList.GetByIndex(i));

        cout << "Camera " << i << " example complete..." << endl << endl;
    }

    // Clear camera list before releasing system
    camList.Clear();

    // Release system
    system->ReleaseInstance();

    cout << endl << "Done! Press Enter to exit..." << endl;

}
```

Illustration 11: launchEveything method

The method « RunSingleCamera » is called once a camera is detected.

```
int MainWindow::RunSingleCamera(CameraPtr pCam)
{
    int result = 0;
    int err = 0;
    try
    {
        // Initialize camera
        pCam->Init();

        // Retrieve GenICam nodemap
        INodeMap & nodeMap = pCam->GetNodeMap();
        //set trigger to software trigger
        result = ConfigureTrigger(nodeMap);

        ImagePtr pImage;
        //take an image
        pImage = AcquireImage(pCam, nodeMap);

        //convert to CVmat format
        result = ConvertToCVmat(pImage);
        pImage->Release();
        pCam->EndAcquisition();
        // Deinitialize camera
        pCam->DeInit();
    }

    catch (Spinnaker::Exception &e)
    {
        cout << "Error: " << e.what() << endl;
        result = -1;
    }
    return result;
}
```

Illustration 12: RunSingleCamera method

16 Hugo Fournier – DUT Report 2019

In this method, there are two methods which are called. The « AcquireImage » method which is here to retrieve the image from the camera and the « ConvertToCVmat » method which is here to convert the image into a matrix with the **OpenCV library** in order to be able to display it in the label.

```
ImagePtr MainWindow::AcquireImage(CameraPtr pCam, INodeMap & nodeMap)    ⚠ instantiation of function 'Spinnaker::BasePtr<Spin
{
    // Set acquisition mode to continuous
    CEnumerationPtr ptrAcquisitionMode = nodeMap.GetNode("AcquisitionMode");
    if (!IsAvailable(ptrAcquisitionMode) || !IsWritable(ptrAcquisitionMode))
    {
        cout << "Unable to set acquisition mode to continuous (node retrieval). Aborting..." << endl << endl;
        return -1;    ⚠ instantiation of function 'Spinnaker::BasePtr<Spinnaker::IImage, Spinnaker::IImage>::operator=' requi
    }

    CEnumEntryPtr ptrAcquisitionModeContinuous = ptrAcquisitionMode->GetEntryByName("Continuous");
    if (!IsAvailable(ptrAcquisitionModeContinuous) || !IsReadable(ptrAcquisitionModeContinuous))
    {
        cout << "Unable to set acquisition mode to continuous (entry 'continuous' retrieval). Aborting..." << endl << endl;
        return -1;
    }

    int64_t acquisitionModeContinuous = ptrAcquisitionModeContinuous->GetValue();

    ptrAcquisitionMode->SetIntValue(acquisitionModeContinuous);

    cout << "Acquisition mode set to continuous..." << endl;

    // Begin acquiring images
    pCam->BeginAcquisition();                    ⚠ instantiation of function 'Spinnaker::BasePtr<Spinnaker::Camera, Spinnake

    // Execute software trigger
    CCommandPtr ptrSoftwareTriggerCommand = nodeMap.GetNode("TriggerSoftware");
    if (!IsAvailable(ptrSoftwareTriggerCommand) || !IsWritable(ptrSoftwareTriggerCommand))
    {
        cout << "Unable to execute trigger. Aborting..." << endl;
        return -1;
    }

    ptrSoftwareTriggerCommand->Execute();

    cout << "Juste avant de recupérer la prochaine image" <<endl;

    // Retrieve the next received image
    ImagePtr pResultImage = pCam->GetNextImage();

    if (pResultImage->IsIncomplete())                ⚠ instantiation of function 'Spinnaker::BasePtr<Spinnaker::IImage, Spi
    {
        cout << "Image incomplete with image status " << pResultImage->GetImageStatus() << "..." << endl << endl;
    }
    else
    {
        // Print image information
        cout << "Grabbed image: width = " << pResultImage->GetWidth() << ", height = " << pResultImage->GetHeight() << endl;
    }

    return pResultImage;
}
```

Illustration 13: AcquireImage method

```
/*
 * This function shows how to convert between Spinnaker ImagePtr container to CVmat container used in OpenCV.
*/
int MainWindow::ConvertToCVmat(ImagePtr pImage)
{
    int result = 0;
    ImagePtr convertedImage = pImage->Convert(PixelFormat_BGR8, NEAREST_NEIGHBOR);

    unsigned int XPadding = convertedImage->GetXPadding();                    ⚠ implicit conversion loses intege
    unsigned int YPadding = convertedImage->GetYPadding();                    ⚠ implicit conversion loses intege
    unsigned int rowsize = convertedImage->GetWidth();                        ⚠ implicit conversion loses intege
    unsigned int colsize = convertedImage->GetHeight();                       ⚠ implicit conversion loses intege

    //image data contains padding. When allocating Mat container size, you need to account for the X,Y image data padding.
    this->frame = cv::Mat(colsize + YPadding, rowsize + XPadding, CV_8UC3, convertedImage->GetData(), convertedImage->GetStride());

    this->qt_image = QImage((const unsigned char*) (frame.data), frame.cols, frame.rows, QImage::Format_RGB888);

    ui->label->setPixmap(QPixmap::fromImage(this->qt_image));
    ui->label->resize(ui->label->pixmap()->size());

    return result;

}
```

Illustration 14: ConvertToCVmat method

17 Hugo Fournier – DUT Report 2019

My goal was reached for this first project but the camera stream displayed was very jerky and I knew that I could make something better so it was the purpose of my second project.

3.2.2 The second project

I understood why the camera stream was jerky, it was because I was closing the steam and opened it every 20 seconds so it could not be smooth when it is displayed.

So for this new project, I took the same functions but I changed it a bit and I also add a stop button in the GUI.

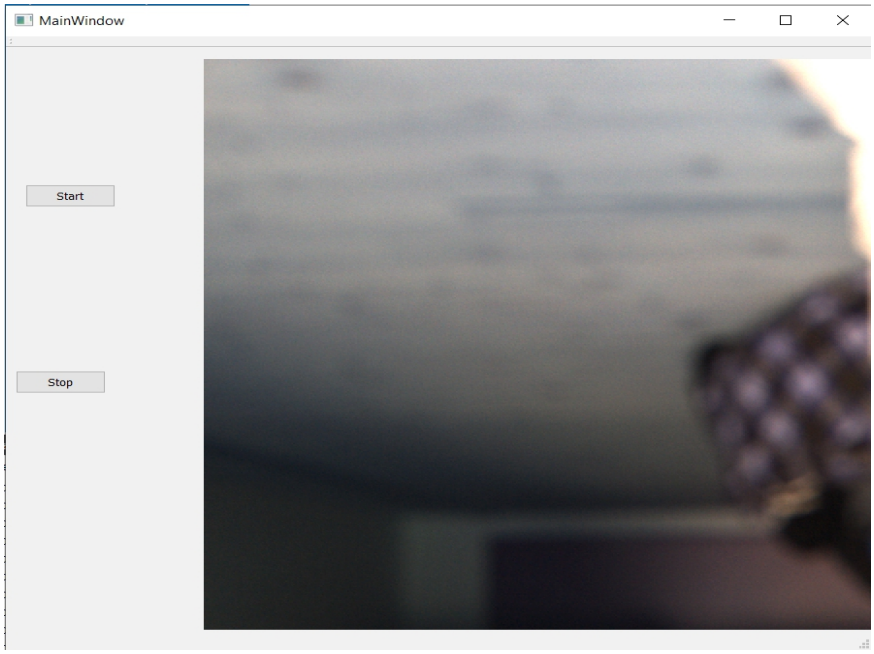


Illustration 15: New GUI for the second project

I decided to add this new button because I knew it would be helpful for the future and into the main software.[18]

```
/*
 *Method to stop the camera stream displayed in the label
 * /*/
void MainWindow::stopEveryThing() {

    //stop the loop with the timer
    this->timer->stop();
    disconnect(timer,SIGNAL(timeout()),this,SLOT(doSomething()));

    this->pCam->EndAcquisition();
    this->pCam->DeInit();

    this->pCam = nullptr;
    // Clear camera list before releasing system
    this->camList.Clear();

    // Release system
    this->system->ReleaseInstance();

    // clear the label display
    this->ui->label->clear();


// reinitialize the camera when the user clicks on the start button again
    new MainWindow();


    cout << endl << "Done!" << endl;

}
```

Illustration 16: Stop capturing function

18 Hugo Fournier – DUT Report 2019

The main change was to move the connection method into the main slot instead of the on click method of the start button.

```
/*
 *Method to initialize the camera and start the camera stream
 * /*/
void MainWindow::launchEveryThing(){

    int result = 0;

    // Retrieve list of cameras from the system
    this->camList = this->system->GetCameras();    ⚠ instantiation of funct

    unsigned int numCameras = camList.GetSize();

    // Finish if there are no cameras
    if (numCameras == 0)
    {
        // Clear camera list before releasing system
        this->camList.Clear();

        // Release system
        this->system->ReleaseInstance();

        cout << "Not enough cameras!" << endl;
        cout << "Done!" << endl;
        getchar();

        exit(-1);
    }



    cout << endl << "Running example for camera " << 0 << "..." << endl;
    this->pCam = this->camList.GetByIndex(0);    ⚠ instantiation of functio

    result = result | RunSingleCamera();


    cout << "Camera " << 0 << " example complete..." << endl << endl;

    // call the slot every 20 ms
    connect(timer,SIGNAL(timeout()),this,SLOT(acquireAndConvertImage()));
    timer->start(20);

}
```

Illustration 17: New launchEveryThing method

Now, the acquisitions began in the Acquire image method.[19]

```
int MainWindow::RunSingleCamera()
{
    int result = 0;
    int err = 0;
    try
    {
        // Initialize camera
        this->pCam->Init();

        // Retrieve GenICam nodemap
        INodeMap & nodeMap = pCam->GetNodeMap();
        //set trigger to software trigger
        result = ConfigureTrigger(nodeMap);

        // Set acquisition mode to continuous
        CEnumerationPtr ptrAcquisitionMode = nodeMap.GetNode("AcquisitionMode");
        if (!IsAvailable(ptrAcquisitionMode) || !IsWritable(ptrAcquisitionMode))
        {
            cout << "Unable to set acquisition mode to continuous (node retrieval). Aborting..." << endl << endl;
            return -1;
        }

        CEnumEntryPtr ptrAcquisitionModeContinuous = ptrAcquisitionMode->GetEntryByName("Continuous");
        if (!IsAvailable(ptrAcquisitionModeContinuous) || !IsReadable(ptrAcquisitionModeContinuous))
        {
            cout << "Unable to set acquisition mode to continuous (entry 'continuous' retrieval). Aborting..." << endl << endl
            return -1;
        }

        int64_t acquisitionModeContinuous = ptrAcquisitionModeContinuous->GetValue();

        ptrAcquisitionMode->SetIntValue(acquisitionModeContinuous);

        cout << "Acquisition mode set to continuous..." << endl;

        //begin the image acquisition
        this->pCam->BeginAcquisition();

    }catch (Spinnaker::Exception &e)
    {
        cout << "Error: " << e.what() << endl;
        cout << "erreur dans run single camera" << endl;
        //result = -1;
    }


    return result;
}
```

Illustration 18: New RunSingleCamera method

19 Hugo Fournier – DUT Report 2019

3.2.3 The third project

The last stand-alone project I wanted to do to be sure that I perfectly understand the API was a multiple display camera stream project because on the main software, there is a multiple display interface so I knew that I will help me for the next part.

I based this project on the « multiple camera acquisition » example from the Spinnaker SDK. It helps me to choose between an array or a vector to store my camera's pointer.

```
void MainWindow::launchEveryThing() {

    int result = 0;

    // Retrieve list of cameras from the system
    this->camList = this->system->GetCameras();      ⚠ instantiation of fu

    unsigned int numCameras = camList.GetSize();

    // Finish if there are no cameras
    if (numCameras == 0)
    {
        // Clear camera list before releasing system
        this->camList.Clear();

        // Release system
        this->system->ReleaseInstance();

        cout << "Not enough cameras!" << endl;
        cout << "Done!" << endl;
        getchar();

        exit(-1);
    }

    // Run example on all cameras
    cout << endl << "Running example for all cameras..." << endl;

    this->allLabel = new QLabel[numCameras];
    for ( int i = 0; i < numCameras; i++) {
        allLabel[i].setText("Waiting for images ...");
        ui->verticalLayout_3->addWidget(&allLabel[i]);

    }
    this->pCamList = new CameraPtr[numCameras];
    this->allImages = new ImagePtr[numCameras];

    result = RunMultipleCameras();

    // call the slot every 20 ms
    connect(timer,SIGNAL(timeout()),this,SLOT(acquireAndConvertImage()));
    timer->start(20);
}
```

Illustration 19: Main Method

20 Hugo Fournier – DUT Report 2019

```
int MainWindow::RunMultipleCameras()
{
    int result = 0;
    unsigned int camListSize = 0;

    try
    {
        // Retrieve camera list size
        camListSize = camList.GetSize();

        for (unsigned int i = 0; i < camListSize; i++)
        {

            // Select camera
            this->pCamList[i] = camList.GetByIndex(i);
            this->pCamList[i]->Init();

            // Retrieve GenICam nodemap
            INodeMap & nodeMap = this->pCamList[i]->GetNodeMap();
            //set trigger to software trigger
            result = ConfigureTrigger(nodeMap);

            // Set acquisition mode to continuous
            CEnumerationPtr ptrAcquisitionMode = nodeMap.GetNode("AcquisitionMode");
            if (!IsAvailable(ptrAcquisitionMode) || !IsWritable(ptrAcquisitionMode))
            {
                cout << "Unable to set acquisition mode to continuous (node retrieval). Aborting..." << endl << endl;
                return -1;
            }

            CEnumEntryPtr ptrAcquisitionModeContinuous = ptrAcquisitionMode->GetEntryByName("Continuous");
            if (!IsAvailable(ptrAcquisitionModeContinuous) || !IsReadable(ptrAcquisitionModeContinuous))
            {
                cout << "Unable to set acquisition mode to continuous (entry 'continuous' retrieval). Aborting..." << endl << endl;
                return -1;
            }

            int64_t acquisitionModeContinuous = ptrAcquisitionModeContinuous->GetValue();

            ptrAcquisitionMode->SetIntValue(acquisitionModeContinuous);

            cout << "Acquisition mode set to continuous..." << endl;

            this->pCamList[i]->BeginAcquisition();
        }

    }
    catch (Spinnaker::Exception &e)
    {
        cout << "Error: " << e.what() << endl;
        result = -1;
    }
```

Illustration 20: Multiple Camera Acquisition method

```
void MainWindow::acquireAndConvertImage() {
    try {
        int result = 0;
        for (int i = 0; i < this->camList.GetSize(); i++) {

            this->allImages[i] = AcquireImages(this->pCamList[i]->GetNodeMap(), i);
            //convert to CVmat format
            result = ConvertToCVmat(i);
            this->allImages[i]->Release();

        }


    } catch(Spinnaker::Exception &e) {
        cout << "Error: " << e.what() << endl;

    }

}
```

Illustration 21: Retrieve Image and Convert method

It is the same methods as previously shown in the report, I just had to do a loop to retrieve image for each camera which are connected.

21 Hugo Fournier – DUT Report 2019

The GUI for this project looked like this :

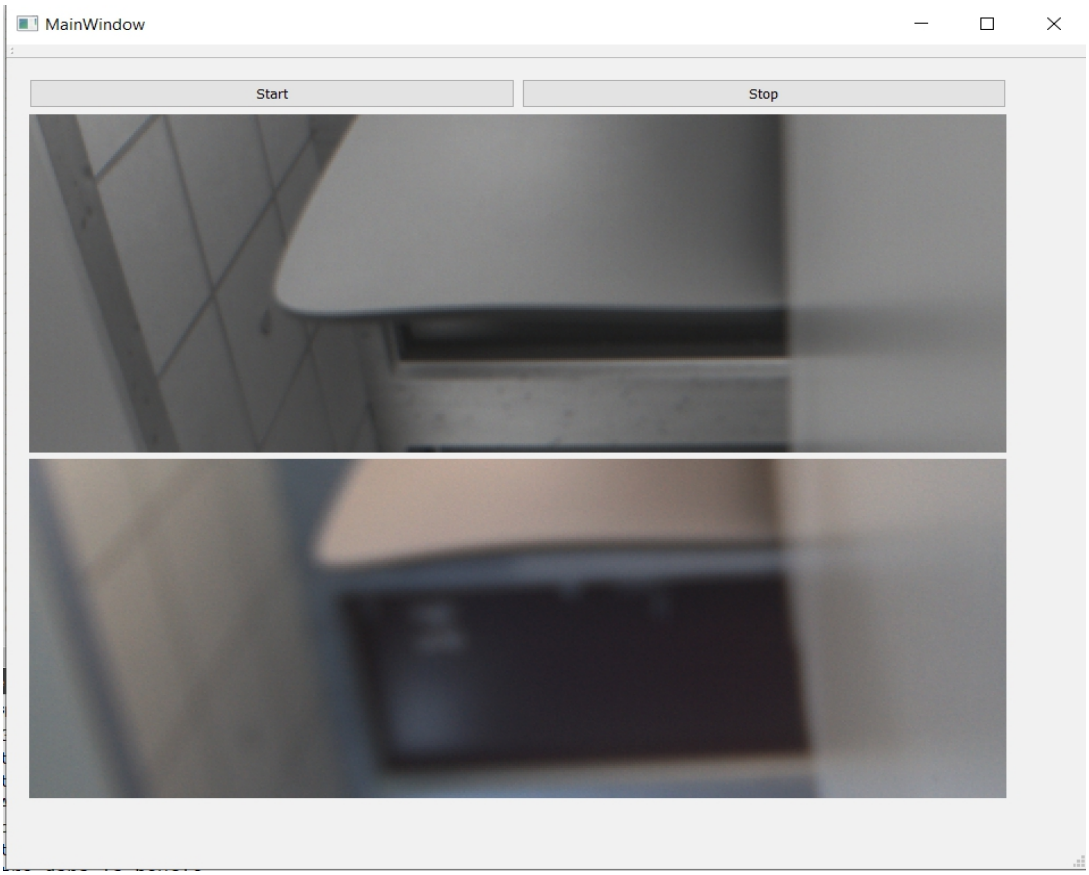


Illustration 22: MultiDisplay GUI

It was now the time for me to join Gauthier on his task in order to implement what I did with this three little projects.

3.3 Updating the main project

Gauthier's work was to change everything from FlyCapture SDK to Spinnaker SDK. We wanted to do that first because we thought it would be easier for us when we will gather the two SDK into the project.

My job now was to make work the camera stream. Thanks to the classes which were already created , I did not that much to make it works.

At first, I wanted to use the same methods that I created in my own project but it was not very working because I used a timer and it involved a new thread to start. So I realized that I just had to use the *VideOpenGLWidget* classes.

22 Hugo Fournier – DUT Report 2019

The next thing to do was to re implement the thread which has been disabled by Gauthier to progress on his task.

For this part, the biggest mistake I made was to not have a look at the initial project because during the time we try to implement the functionalities with Spinnaker. So it took me two days to realize that there were missing code lines and that was the cause of all my issues. Finally, the thread were implemented and it was fully working with the Spinnaker SDK.

At this time, there were three tasks left which were to be able to choose between Spinnaker and FlyCapture SDK inside the software, implement the properties for the cameras and make the technical and user guide.

Gauthier and Adrien decided to work on the properties so I worked on the possibility to choose the API the users wants.

I knew what I had to do and how I could do it but I had to create some classes in addition of the existing ones. I had to separate the FlyCapture SDK from the Spinnaker one. There already is a FlyCapture classe with a FlyCamera Manager so I created the Spinnaker classe with his Spinnaker Camera Manager, based on the same methods and properties.

I used the comboBox inside the GUI to add the Spinnaker possibility.

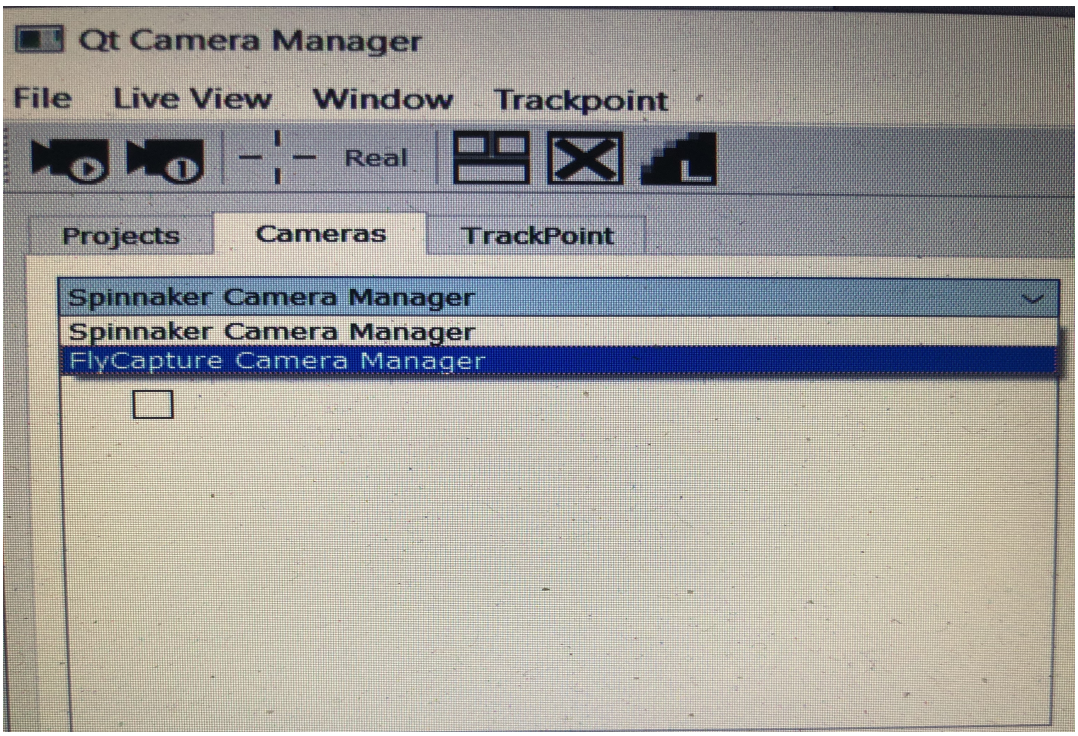


Illustration 23: ComboBox for the user

23 Hugo Fournier – DUT Report 2019

I was wondering how I could make this choice applicable for all the classes. The simple way to do that was to use a static variable so that is what I did inside the MainWindow class.

```
/**
 * MainWindow
 * \brief handle ui events.
 */
class MainWindow : public QMainWindow {
    Q_OBJECT
public:
    /** CONSTRUCTOR - DESTRUCTOR **/
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();

    //Hugo Fournier - 03.06.2019 - IMPLEMENT FLY/SPIN
    static int selection;
```

To make the change effective each time the user chose a different manager, I created a slot connected to a signal.

```
/* Slot called when selection in combobox changed */
void MainWindow::combobox_changeSDK() {
    if(ui->selectCameraManager->currentText() == "FlyCapture Camera Manager") {
        selection = 1;
        cout << "------- ON SELECTIONNE FLYCAPTURE ----------------------" << endl;
        cameraManagers[selectedCameraManager]->loadPropertiesDefaultsInit();


    } else {
        selection = 2;
        cout << "-------------ON SELECTIONNE SPINNAKER ----------" << endl;
        cameraManagers[selectedCameraManager]->loadPropertiesDefaultsInit();
    }

}
```

Illustration 24: Method called each time the manager selected is different

```
//Hugo Fournier - 03.06.2019 - IMPLEMENT FLY/SPIN
connect(ui->selectCameraManager, SIGNAL(currentIndexChanged(QString)), this, SLOT(combobox_changeSDK()));
```

Illustration 25: Connection between signal and my method

Finally, for all the methods related to the manager, I had to make a condition depends on if it was FlyCapture or Spinnaker which are selected.

24 Hugo Fournier – DUT Report 2019

The last thing to resolve was the display issue we had with the color cameras. Inside the subwindow, there were some squares displayed on the camera stream. It was a real problem because for the motion capture and especially the « TrackPoint » feature, my tutors needed a clear camera stream.

Thanks to one of my tutor during a meeting, he said that we should convert the image captured by the camera into an 8 bit mono image instead of trying to get a colored image.

That is what I did with a simple method which is currently existing with ImagePtr object.

```
ImagePtr image = nullptr;
ImagePtr convertedImage = nullptr;
try {
    if ( cam->IsStreaming()) {

        image = cam->GetNextImage();
        convertedImage = image->Convert(PixelFormat_Mono8, NEAREST_NEIGHBOR);

    } else {
        cout <<" not streaming" << endl;
    }

    return convertedImage;
```

Illustration 26: Image conversion

The last part of our trainee was dedicated to the making of the technical and user guides.

3.4 Guides realisation

When our tutors gave us the project, there were two guides written : the technical guide and the user guide.

The user guide is made to handle properly the GUI and all the items in the menu bar or the possible functionalities with the GUI.

The technical guide is made especially for people who will continue to work on the app. There are descriptions of the different classes and how to install the project on your personal computer.

The last guides were made five years ago so it was the time for us to make a new one to add what we did during our internship.
Here are some pages of the guides I made with Gauthier and Adrien, based on the previous one.

25 Hugo Fournier – DUT Report 2019

Table of contents

Illustration 27: Technical Guide : Table of contents

I. Set up the environment

1. Installation on Windows

The project has been written to use the defaults installation paths of all the software you are going to install. If you change the installation paths, you may have to modify the .pro file in Qt. You will get more details on that later.

1.1. Executables pack

In order to let the software works, you will need to install the FlyCapture and Spinnaker SDK, Qt and Microsoft Visual Studio 2015. Please note that the project won't work if you try to use another version of Microsoft Visual Studio, it has to be the 2015 version (but you can use community or professional version, it doesn't matter).

In order to make the downloads easier for you, we made an archive with all the executables files so that you don't have to download everything. Please note that if you use that archive, Qt will update and retrieve its files as you install it, Visual Studio is up to date, the FlyCapture SDK is up to date too but Spinnaker may have received some update since the release of the Camera Manager software (June 2019). Camera Manager is fully operational with the version included in the archive but you may need to update the SDK to implement other features later. Here is the link of the archive: https://mega.nz/#!rZclUYLL!HJE8Mpwwa6zSB6-FQH6xYe8vH3EN_1Cnnz9Zqxg786Y

1.2. FlyCapture and Spinnaker SDK

If you choose not use the archive solution described above or if you need to download a more recent version of the SDK, just follow these steps:

1.2.1 FlyCapture installation

- Go to this link: https://www.flir.eu/products/flycapture-sdk/
- Click on the Download button and then Download from Box
- Click on the Windows folder then FlyCapture Latest Full SDK
- I recommend you to download both the 64 and the 32 bits version of the SDK

Now that you have the executables files, just launch the 32-bits installer, follow the steps, be sure to check the box "I will use USB cameras" and then "The installer will register the DirectShow DLLs".
When the first installation is over, repeat the process with the 64-bits installer.

Illustration 28: How to set up environment

2. Installation on Ubuntu

2.1. GCC/GPP Compiler

First, you need to install the GCC/GPP compiler. The installation is quite simple: open a terminal and run this command:

```
sudo apt-get install build-essential gcc
```

2.2. FlyCapture and Spinnaker SDK

- Go to this link: https://www.flir.eu/products/flycapture-sdk/
- Click on the Download button and then Download from Box
- Click on the Linux folder then download the SDK according to your system

The download of the Spinnaker is roughly the same but you have to go on this link instead: https://www.flir.eu/products/spinnaker-sdk/

Concerning FlyCapture, the "Xenial Xerus" version is for Ubuntu 16.04 and the "Bionic Beaver" one is for Ubuntu 18.04. Choosing amd64 file allows you to install it on Intel/AMD CPUs. If you are running on an ARM CPU, choose the "arm" file.

When you have downloaded and extracted the two files and, you can use the readme to continue the installation of the two SDK. You just have to copy-paste the commands that are in these files.

In case you are experiencing some troubles, with the installation of Spinnaker, you can try to raise the memory limit:
Open the /etc/default/grub file in any text editor.
Find and replace (You may open this file with the super admin rights):

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"
```

With this:

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash usbcore.usbfs_memory_mb=1000"
```

Then update the grub with `sudo update-grub` and reboot the pc.

To check if it worked, execute this command:

```
cat /sys/module/usbcore/parameters/usbfs_memory_mb
```

If it returns 1000 it worked, if not, you can try to execute this command:

```
sudo sh -c 'echo 1000 > /sys/module/usbcore/parameters/usbfs_memory_mb'
```

Then, try again to install the Spinnaker SDK.

Illustration 29: How to install environment 2

Table of contents

Illustration 30: User Guide : Table of contents

III. Camera interface

1. The camera tab

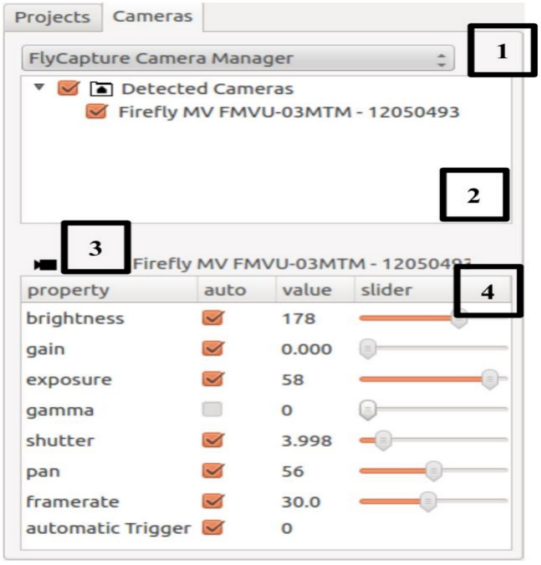


Figure 1 : Camera tab

1: This is the menu where you can choose the API you want to run. Currently, there are two choices: *Fly Camera Manager* which runs the FlyCapture API and *Spinnaker Camera Manager* which runs the Spinnaker API.

2: This is a tree where all the found cameras will be. For both of the managers, each time you plug or unplug a camera, the software will automatically detect it and update the tree.
To select a camera, just click on its name. If you want to watch the images coming from this camera, tick its corresponding checkbox and a sub window will appear on the right (more details further).
Performing a right click on it pop up a menu, which is shown on *Figure 2*. With it, you can *add* and *remove* a camera group. This can be very useful if you have several cameras and want to sort them by combination. You can also *edit* and *reset* the name of a camera.

3: This is the name of the camera which is currently selected. if there is no camera selected, *No Selection* will be written.

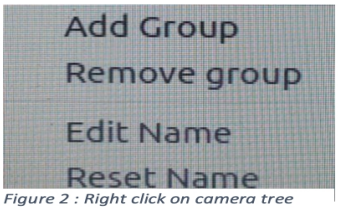


Figure 2 : Right click on camera tree

4: This is the table with the selected camera properties. The camera **must** be recording to have properties updated. There are fourth columns per property:
- the *property* name
- the *auto* box: checking the checkbox allows you to set the value. Leave it unchecked if you want the camera to auto-adapt the value itself.
- the *value:* the integer value of the parameter corresponding to the slider position.

Illustration 31: Camera Interface in GUI

2. The toolbar

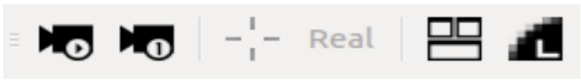


Figure 3 : Toolbar

This is the toolbar "menu", at the top of the frame, just below the menu bar. It provides six actions. From left to right:

- *Live View:* start, or stop the live view. If you changed properties in the properties box, the stream from the camera will have their properties updated.

- *Update Image:* get one image from the camera to display it in a sub window.

- *Cross hair:* if *Live View* is enabled, draw X and Y axis according to you mouse position to help you get the coordinates of your mouse pointer. These coordinates will be displayed at the top left corner of the sub window. You can see more details on that function further.

- *Integer coordinates:* show coordinates with integers and not decimal numbers.

- *Mosaic view:* once clicked, all sub windows will be stretched to fill all the space.

- *Quality:* Change the quality of the pictures. Be careful: this function causes high CPU usage.

Note: the toolbar can be hidden (to get more space) by using the menu bar (*Window →Hide ToolBar*) or by performing the keyshort *Ctrl+H*, *Ctrl+T*.

Illustration 32: MenuBar in the GUI

Conclusion

At the end of our project, I was really proud of what we did with Gauthier and Adrien. The goal was reached and it was a real satisfaction to look at it.

Through all these months, I learned a lot in terms of programming, way of working but also generally speaking, as a student in a foreign country.

I had never practiced C++ language before so now, I can say that I have very good notions and I am sure I will be able to make another projects more efficiently now with all I learned during this three months.

Moreover, this language is very common in the world of work so it is also a plus for my future career in IT.

Working on such a project was also a first time for us and it makes me realize that I learned a lot during these 2 years of study in France and to know that I can handle a big project, understand the classes and the architecture makes me satisfied.

As a student, I went to Norway to improve my English skills and I had to say that it was a wonderful experience for me, the people around here, other students or local merchants, make a lot of efforts to make feel us conformable in the fact that we didn't speak Norwegian. I made some strong improvement and it is very challenging to be here to improve because the Norwegians are good at both English and mother tongue.

29 Hugo Fournier – DUT Report 2019

Bibliographie

Courses to learn C++ : https://openclassrooms.com/fr/courses/1894236-programmez-avec-le-langage-c

My tutor's compagny website : http://www.3dmotech.com/index.php

Qt Documentations : https://doc.qt.io/

Cameras and API documentations : https://www.flir.com/

30 Hugo Fournier – DUT report 2019

31 Hugo Fournier – DUT Report 2019