Face Tracking in Unity Document

# About

This purpose of this document is to record important information regarding the project that has been undertaken during the semester.

The project I have undertaken is face-tracking in the Unity Game Engine. During initial discussion about this project, there was a heavy emphasis on accurate jaw movement. However due to several issues that has arose while progressing through this project I was unable to find a suitable way if tracking the jaw specifically. I hope that the contents in this document can help the next person in continuing this project.

The key parts of this document are the Unity, EmguCv and OpenCvSharp sections.

# Unity

The [Unity](https://unity3d.com/) game engine is a powerful game engine which provides a wide verity of tools. Some key benefits of this engine are the augmented reality (AR), the virtual reality (VR) and the wide support of build capabilities.

However, one feature that this engine lacks is a face-tracking library. There are plugins available from the Unity store that can easily be imported to your Unity project. Unfortunately, this is a costly solution as it costs $95. More detail about this plugin can be found [here](https://www.assetstore.unity3d.com/en/#!/content/21088).

When creating scripts in Unity there is an option of two different programming options; C# and JavaScript. I have decided to program in C# as I have more familiar with the language and for other reason which will be talked about in the next section.

# OpenCv

[OpenCv](http://opencv.org/) is a native C++ library which deals with object detection. Due to its C++ nature, it is not easily implemented into Unity. A solution to this problem is to use one of the several wrappers around this library to allow it to be used in conjunction with another programming language.

When researching possible wrappers that could be used there was two criteria’s that I followed:

1. Needed to be free. (Could not afford a wrapper that needed paid licence)
2. Wrapper is in C# to be easily compatible with Unity.

Regardless of what wrapper is selected it should still be like the OpenCv functions; just with different method calls and slightly different structure names.

From the research undertaken the two potential wrappers that can be used with Unity is EmguCv and OpenCvSharp.

General information

Regardless of what wrapper is selected the main issue that will occur is the conversion between the Unity Texture2D format and the Mat structure that OpenCv would use. In the projects provided along with this document, there are a couple of different solutions to converting them, however, both have an impact on performance. Another solution is to use the OpenCv camera instead.

Another possibility is converting the image on the GPU instead of the CPU.

Training your own cascade

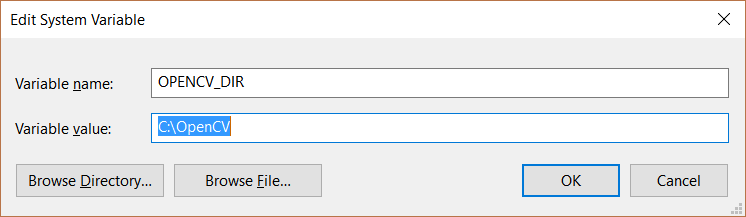
To use object detection, a xml file which contains the cascade classifier needs to be used. This file is what contains the data to help the computer know what object it is looking for. As part of the OpenCv library, there is a way to create the xml file.

To start off with you will need a series of positive and negative images, these are images with and without the object or feature you are looking for.

1. Include steps here

Installation

To use face-tracking I would highly advice installing OpenCv to your computer and adding it to your system path. The steps to do this are as follows:

1. Download and install [OpenCv](http://opencv.org/)
2. Go to Environment Variables, which can be found in system properties.
3. Create a new system variable, create a suitable name for it and add the path to your OpenCv library. For example:  
   

This should allow Visual Studio find the DLL for OpenCv allowing creation of OpenCv applications.

An easy way to find out if this is working is creating an empty C++ project. If you can access the OpenCv namespace, then it has been configured correctly.

For more information on setup of OpenCv and how to use its features please have a look at the tutorial section of their [website](http://docs.opencv.org/master/d9/df8/tutorial_root.html).

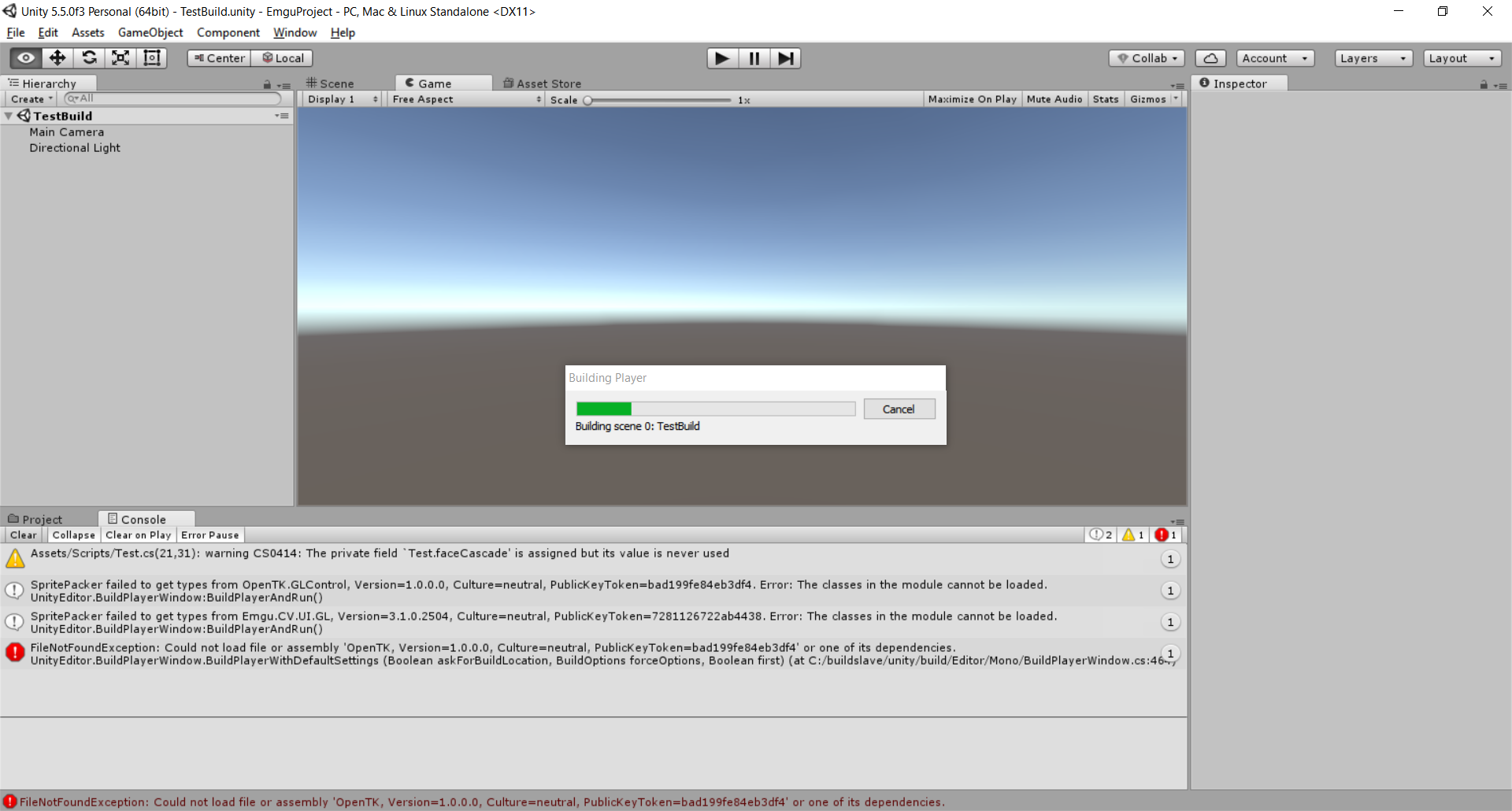
# EmguCv/Unity

[EmguCv](http://www.emgu.com/wiki/index.php/Main_Page) is the first of the two wrappers that I found while researching ways to easily implement face-tracking into Unity engine.

Installation

1. Download and install [EmguCv](http://www.emgu.com/wiki/index.php/Download_And_Installation) to a suitable location on your computer.
2. Once installed, add to the system path of your computer. For Unity, only the 32bit path needs to be included.
3. Add the required DLL’s to the plugin folder in your Unity project\*.

\* Create a folder in the assets folder of the Unity project and call it ‘plugins’. However, listing the dll’s required is in practical as it is quite a large list. What I would suggest doing is copying the plugin folder from the EmguCV project that I provide.

Remember to test if you can build the project afterwards to ensure that the setup has been configured correctly. If an error such as this occurs:

Then you are missing the required dll. To solve this, find the missing dll and copy it into the plugin folder.

The repository for the official EmguCv wrapper can be found [here](https://sourceforge.net/projects/emgucv/).

How to use

Examples

Combined face/edge detection

The example project utilising this project makes use of haar cascades and edge detection algorithms. When the project is running, it connects to the first available webcam and then while the webcam is active it will detect regions of interest (ROI). In this situation, it is faces it is looking for. After it has searched it will then find all the contours in the ROI and draw them and then display the image back to us.

GrabCut

When researching how to draw contours around the entire face I came across a method called GrabCut. It is a means of background removal. I have created a script which find potential faces then tries to remove the background from it leaving just the face. There has not been much success so far but I think it has potential to work if it is not too performance costly.

The original webpage which lead to this idea can be found [here](http://www.bogotobogo.com/python/OpenCV_Python/python_opencv3_Image_Segmentation_by_Foreground_Extraction_Grabcut_Algorithm_based_on_Graph_cuts.php).

Improved Conversion (GPU approach)

As the GPU is powerful for parallel computation it would make sense to do the conversion between an OpenCV mat object and a Unity2D. I managed to set a texture up with the proper dimension and have all the pixels set the red. To finish this solution this is what needs to be done.

1. Convert Mat into array of pixel colours (byte array?).
2. Convert bye array into Color/ Vector4 array?
3. Send Color/Vector4 array to the GPU.
4. Set the texture pixel colour to corresponding array values.

Final evaluation

Overall I have preferred using EmguCV the most, although I have had the least time working with it I have felt that performance has overall been better over OpenCvSharp.

# OpenCvSharp/Unity

Installation

1. Create empty Unity project.
2. Open project solution folder in Visual Studio.
3. Go to Tools-> NuGet packet Manager.
4. Open manage packages for solution.
5. Search for ‘OpenCVSharp3- AnyCPU’ by Shimat.
6. Install package.
7. Project can now use OpenCvSharp but will not be able to build an exe.

Once this has been done you can use the OpenCvSharp features. I chose this approach as it was the simplest approach, you are also noticed if there is an update to the package. To build this project to run as an executable you need to follow the same steps EmguCv when configuring the build. You need to find the DLL that says are missing and manually add them into the Assets folder of your Unity project. Once all of them have been correctly added then you can build the exe. I would recommend the 32-bit DLL to ensure there are no compatibility errors.

How to use

Examples

As a proof of concept that this project works I created a small application that …

Final evaluation

This approach is better if you are looking to learn about computer vision and image manipulation. It has good supporting documentation and a small, but active community. I found however, that this wrapper has worse performance over EmguCv due to how the wrapper has been created.

**Update:** As I have been improving performance off applications usuning EmguCv I need to check if the same techniques could be applied to OpenCvSharp to achieve higher frames per second.

# Other alternatives

OpenCvForUnity

This is a paid plugin from the Unity store which handles all the installation for you. As there is a cost attached to it I cannot vouch for its performance. The User reviews were not that promising so I would avoid using this option personally.

Vuforia

This is augmented reality software which is in partnership with Unity. Therefore, it is easy to set up an AR project with this software. There is a lack of facial recognition capabilities thus there is still an additional library required. In terms of licensing, it is free to use as long as the application is in

Dlib

This is a C++ library which extends from OpenCv. Nativly it is supposed to be more optimised and produce better results than OpenCv. It also benefits from HOG (Histogram of oriented gradients) as well as the haarCascades which OpenCv provides. Another benefit is landmark detection which can create results such as this.

**Update:** as I was finishing this documents I can across [this](https://github.com/EnoxSoftware/DlibFaceLandmarkDetector) project on GitHub. It is supposed to be Dlib integrated with Unity therefore it is possible. However, you need to install it from the Unity store. This costs $40.

# Issues

* A lot of the methods are computationally expensive – need to have optimal code and select methods/ parameters effectively.
* Relying on an external wrapper could mean it can become out of date.
* Depending on the wrapper selected, documentation will vary.
* Training own cascade is difficult and time-consuming.
* Face-detection seems to be rotation variant.
* OpenCvSharp refuses to build even if correct dll are in place.

# Links.

<http://www.cs.ccu.edu.tw/~damon/photo/,OpenCV/,Mastering_OpenCV.pdf>

<http://alereimondo.no-ip.org/OpenCV/34>

<https://github.com/qian256/OpenCVSharp3Unity>

<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2745713/>

<http://www.sbdmj.com/053/053-01.pdf>

<http://docs.opencv.org/2.4.13.2/doc/user_guide/ug_mat.html>

<http://stackoverflow.com/questions/7899108/opencv-get-pixel-channel-value-from-mat-image>

<http://docs.opencv.org/3.2.0/dc/d88/tutorial_traincascade.html>

<http://stackoverflow.com/questions/36259712/save-output-image-in-jpg-format-opencv>

<http://www.sciencedirect.com/science/article/pii/S188276160900009X>

<http://docs.opencv.org/2.4/modules/core/doc/drawing_functions.html>

<http://docs.opencv.org/2.4.13.2/doc/user_guide/ug_traincascade.html>

<https://en.wikipedia.org/wiki/Cascade_algorithm>

<http://opencv.org/>

<http://coding-robin.de/2013/07/22/train-your-own-opencv-haar-classifier.html>

<http://emovu.com/docs/html/getting_started.htm>

<http://www.justapixel.co.uk/how-to-make-an-ar-app-in-5-minutes-with-unity-and-vuforia/>

<https://library.vuforia.com/articles/Solution/Compiling-a-Simple-Unity-Project>