NHE2422 Assignment 2 – Ross Hamilton

U1358389

Presentation

Key design:

- Scope reduced greatly due to compatibility difficulties, ultimately resulting in a product which does not function as originally intended.
- Design remained basically the same, but the plans for the underlying code had to be changed entirely.
- The system will output the webcam view in Unity, but does not correctly pass this to OpenCV/retrieve any data about detection, meaning that the functional elements of my game cannot work.

Development:

- Programmed using C# scripts in Unity and C++ for OpenCV, interacting using interop services, which allow methods to be called across different languages using standard C declarations.
- My code is contained within "Source1.cpp" (OpenCV) and "WebcamTexture.cs" (Unity)
- I had originally planned to use a Unity plugin which integrated OpenCV in a very easy to use manner, however this plugin was only available at great cost.
- Upon discovering the cost of this plugin I decided to write my code into a 'wrapper', which translates code between languages at runtime. It soon became apparent that this was very complicated to write and extremely slow at run time, and so I moved on to attempting to use interop services.
- The C++ code written to be called from within C# is included into a .dll file, imported into the Unity project and called using direct references to method names.
- Unity receives the webcam image as a texture, applies it to a plane, and tries to pass it to OpenCV.
- Due to limitations in passing complex object types between C# and OpenCV, I had to resort to passing the image between the two languages pixel by pixel, which is an extremely slow procedure, but is functional.
- After my OpenCV code receives the pixels, it rebuilds a texture, applies various operations to it in order to isolate a certain colour, uses the findNonZero() method to retrieve all pixels where the colour in question was detected and returns this list pixel by pixel when requested by the Unity code. Somewhere in this OpenCV flow, my OpenCV code fails to work as intended – either during pixel setting or detection.

- The nature of using a pre-compiled .dll file within Unity has negative implications on the development and debugging procedure. Namely, Unity can only access variables from the C++ code which are made available to it using method return values.
- Both method parameters and return values must be used in a very simple way, so that both languages are able to interpret them correctly. This gives a great deal of difficulty when working with WebcamTexture objects and cv::Mat objects.
- The inability to pass vector or List objects greatly hinders the process of detection and image interpretation – a list of detected object locations and shape data would be much easier to work with on return to Unity than a list of pixels.
- OpenCV is a powerful toolkit for Computer Vision purposes, and encapsulating it within limited methods which are difficult to test stunts OpenCV's abilities, as well as the development process.

Review:

- My application displays very little in the way of function, as so much of my available programming time was devoted to attempting to develop a framework which would allow my game to function.
- It does not meet my original plan goals due to my choice of language and engine.
- If I had the chance to begin this project again, I would place much more emphasis on my game engine decision, as I presumed the theoretical compatibility between OpenCV and Unity would translate well into a practicable development process.
- I would most likely use native C++ and OpenCV to program my game if I were to begin again, to ensure compatibility. Although this makes gameplay programming much more difficult, the underlying technical framework would be much more likely to be successful.
- Although using inefficient methods to pass each frame to OpenCV, my application does run at a playable frame rate on the 3 mid-range PCs I used for testing.

Conclusion:

In all, my resulting system was inefficient and not functional, and would likely not achieve its goals without a rebuilding in a different language. Had I moved to a different language/engine earlier in my development process, I may have been able to produce a working product that met my development goals. However, I spent a large amount of time trialling different methods of passing frames between Unity and OpenCV, experimenting with differing object types and interop procedures, which ultimately did not result in a working system, but did result in me learning a great deal about data structures, and working between multiple languages.