**程式碼放置於github請使用Git獲取**

[**https://github.com/fajoy/RTSPExample.git**](https://github.com/fajoy/RTSPExample.git)

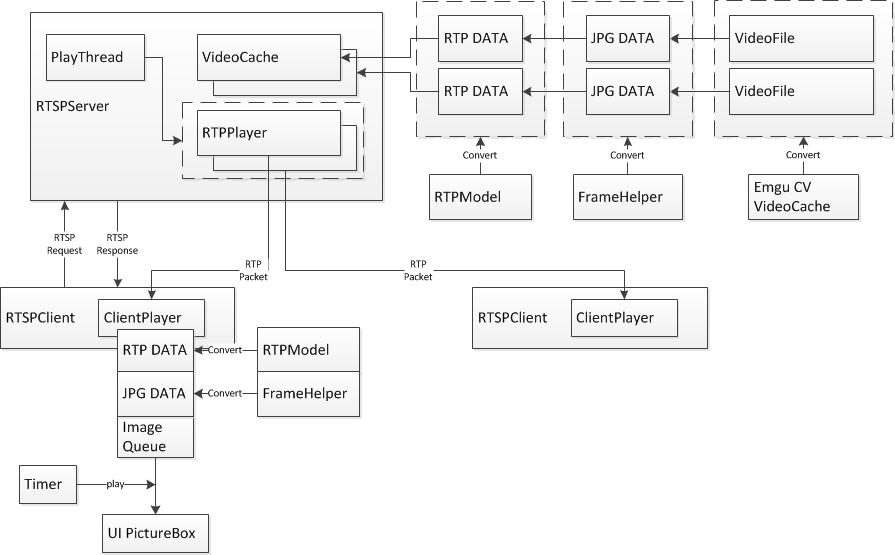
**程式需求**

* Microsoft Windows7 x64
* Microsoft Visual Studio 2008
* Emgu CV x64

**專案說明**

* Emgu.CV (Emgu CV原始碼)
* Emgu.Util (Emgu CV原始碼)
* HelloWorld (Emgu CV原始碼 可測試Emgu CV是否正常運作)
* VideoTest (可測試Emgu CV是否可正常擷取影格)
* VideoServer (RTSP伺服器)
* VideoClient (RTSP客戶端)

**程式架構**



**物件說明**

**共用物件**

* RTPModel
  + RTPModel class <-> rtp packet byte array 轉換
* FrameHelper
  + Bitmap class <-> jpge byte array 轉換
* SockHelper
  + Async TCP 與 Async UDP 物件

**伺服器端**

* VideoCache
  + 將VideoFile 轉成 RTP封包 byte存放
* RTPPlayer
  + 負責發送RTP封包與儲存播放狀態
* RTSPServer
  + 負責處理Client TCP的RTSP Rquest 並控制RTPPlayer

**客戶端**

* ClientPlayer
  + 負責處理接收到的UDP封包並轉成Image存放進Queue
* RTSPClient
  + 負責儲存RTSP狀態、RTSP Rquest發送與RTSP Response處理

**開發流程**

FrameHelper.cs撰寫

1. 先完成影片影格擷取動作，需仍夠將影片的影格提取出來成C#物件使用
2. 再將C#物件轉成jpge格式 byte array 格式使用
3. 再將byte array看使否能轉回Image物件，提供UI使用
4. 後來發現UDP封包最大只能到65535大小，所以又加入了壓縮jpge15%品質程式碼進去。

RTPModel.cs撰寫

1. 測試將jpge byte轉成RTP封包狀況
2. 須將byte array轉成parse 成RTPModel取得playload
3. 發現有些地方會有溢位的狀況發生需調整timestemp與seq長度大小

SockHelper.cs撰寫

1. 使用asyc方式撰寫，之前都使用multithread撰寫socket，改成這樣寫程式碼精簡多了

VideoServer與VideoClient 撰寫

1. 先完成RTP單向傳輸，能夠正常的由server發送RTP封包至client播放，完成物件有VideoCache、RTPPlayer
2. 完成RSTP Request與Response動作，Server須將存放多個VideoCache與RTPPlayer，並持續播放已存在的RTPPlayer封包，Client需建立UDP接收端後，Request SETUP取得Sessions，方可控制影片動作，Server收到Request後需依照SessionID對RTPPlayer做控制，並回覆Response，完成物件有RTSPServer、RTSPClient