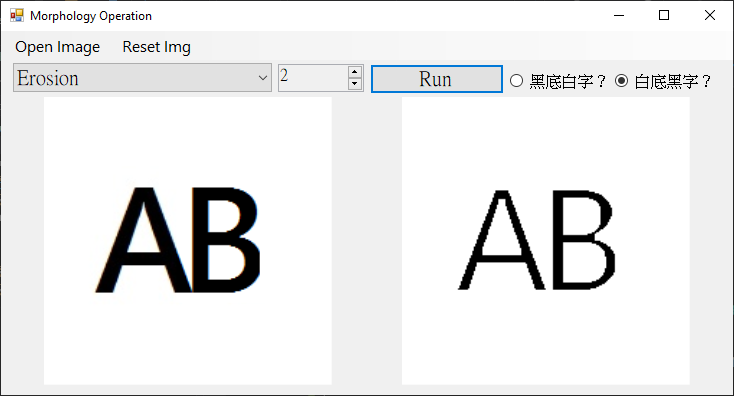
程式執行畫面：

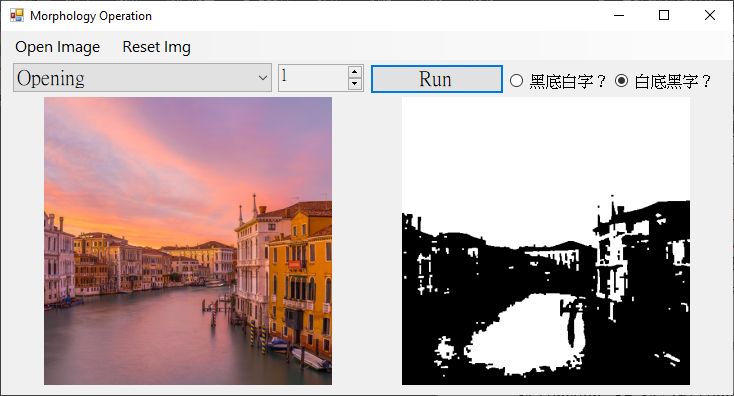


下圖為全國高中職學生技藝競賽歷屆試題102年第4題，白底黑字的範例：



當然也可以做全彩影像，下圖是上學期某個作業的範例圖片：

轉二值化影像後，經過一次膨脹一次侵蝕與一次開運算。



說明：

全域變數、Function我全部都宣告在.h檔中，而以下說明一下Function：

1. Bitmap^ Binarize(Bitmap^ bmp);

將圖片轉為二值化影像。

1. Bitmap^ DilationOrErosion(int mode, Bitmap^ bmp);

我將膨脹與侵蝕寫在同個Function內，因為只是相反運算，code大致相同。

引數中的mode可以設定要做膨脹還是侵蝕。（0: 膨脹 (Dilation)，1: 侵蝕 (Erosion)）

而裡面的code有點多，因此我拆成三個部分，並個別寫Function。如下：

1. System::Void GetOriginImgData(int mode, BitmapData^ bmpBD, array<bool, 2>^ bIsTextPixel, array<unsigned char, 2>^ bmpPixel);

該Function會先取得原圖的圖片資料（像素值）。

Mode也是藉由上層丟下來決定，膨脹跟侵蝕紀錄的值不同。

1. System::Void HandlePixels(int mode, BitmapData^ bmpBD, array<bool, 2>^ bIsTextPixel, array<unsigned char, 2>^ bmpPixel);

該Function是最核心的Function，膨脹與侵蝕運算皆在此做。

依照上層丟下來的Mode，以及紀錄的值，去做膨脹與侵蝕。

1. Bitmap^ CopyPixelArrToOutput(Bitmap^ bmp, array<unsigned char, 2>^ pixelArray);

該Function是將結果回存，因為我皆是在CLI的Array中處理，最後會將圖片Return回去。

而Opening / Closing就是運用Dilation跟Erosion搭配的順序來處理。

Opening: 先侵蝕在膨脹；Closing: 先膨脹在侵蝕。