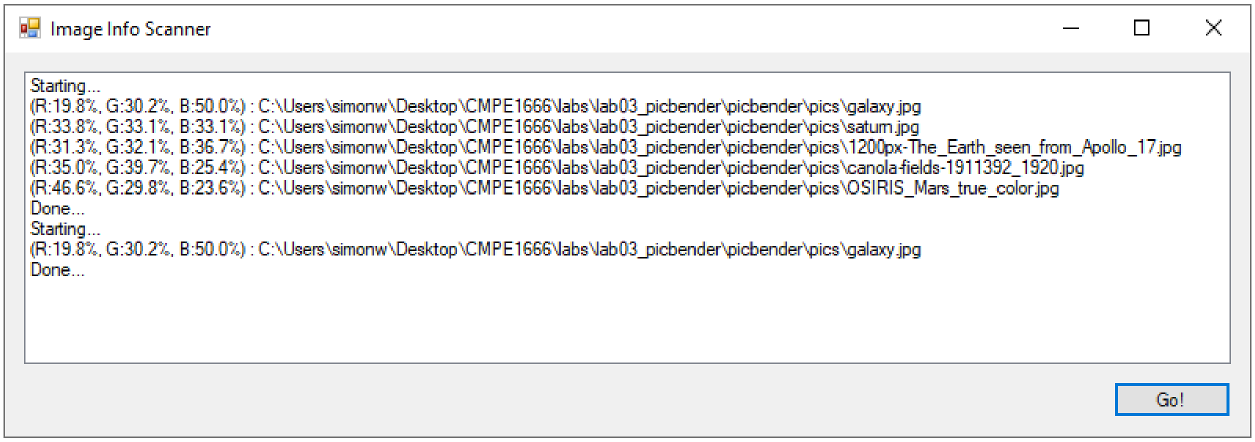
School of Media And Information Technology
Department of IST
Program: CNT

CMPE1666 –ICA 17- Thready Colour Analyzer

You will create an application with the following appearance:



```
Starting...
(R:19.8%, G:30.2%, B:50.0%) : C:\Users\simonw\Desktop\CMPE1666\labs\lab03_picbender\picbender\pics\galaxy.jpg
(R:33.8%, G:33.1%, B:33.1%) : C:\Users\simonw\Desktop\CMPE1666\labs\lab03_picbender\picbender\pics\saturn.jpg
(R:31.3%, G:32.1%, B:36.7%) : C:\Users\simonw\Desktop\CMPE1666\labs\lab03_picbender\picbender\pics\1200px-The_Earth_seen_from_Apollo_17.jpg
(R:35.0%, G:39.7%, B:25.4%) : C:\Users\simonw\Desktop\CMPE1666\labs\lab03_picbender\picbender\pics\canola-fields-1911392_1920.jpg
(R:46.6%, G:29.8%, B:23.6%) : C:\Users\simonw\Desktop\CMPE1666\labs\lab03_picbender\picbender\pics\OSIRIS_Mars_true_color.jpg
Done...
Starting...
(R:19.8%, G:30.2%, B:50.0%) : C:\Users\simonw\Desktop\CMPE1666\labs\lab03_picbender\picbender\pics\galaxy.jpg
Done...
```

When the 'Go' button is pressed, the application will allow the user to select one or more files from an open file dialog. The application will use separate threads to analyze each file for the quantity of red, green, and blue, expressed as a percentage of the image as a whole. The percentage of red, for example, would be the sum of all red values divided by the sum of all colour components in the image x 100.

Once the threads have completed the colour analysis for the image, they will invoke the findings back to the main thread for display in the list box as shown. You may use the stock images and the results shown above to validate your values.

The application will manage the threads in a list, and you must add a timer to check the threads for completion (the thread state will be 'Stopped'). The timer should only be running when image processing is happening. The timer body will check all threads for the stopped state, and if all threads are stopped, the 'Done' message will be shown (as above) to let the user know that all images have been processed. The 'Start' message will also be required. These messages are required so that the user will have start/stop feedback.

When you increase the size of the form, the size of the list box should increase accordingly, both in width and height while the button should retain its relative position near the lower-right corner.

Additional Information

In the main form, at class level, add a **`List`** of **`Thread`** (don’t forget to include **`System.Threading`**).

Add a timer and open file dialog components to your form.

In the button event handler, or at design time, configure the OFD to filter on ‘All File (Images) | *.*’ and allow multi-select. Show the dialog. If OK, enable your thread checking timer then loop the **`FileNames`** collection and start a parameterized thread start for each name in the collection, passing the file name to the thread. Don’t forget to mark the thread as background and add it to your list of thread before you start it.

In the processing thread (there will be one started for each file selected), attempt to open the passed file name as a bitmap:

```
Bitmap bm = (Bitmap)Bitmap.FromFile(fname);
```

If it fails, just return out of the threads (you won’t let the user know that the file was invalid… unless you want to…). If it opened OK, do the color count calculations. Create a method that will accept the information on the file. This will likely be **`string`** (file name), **`double`** x 3 (R,G,B %). Create a delegate type that can target your method. **`Invoke`** the method from the processing thread when you have the numbers calculated, not forgetting to **`try`**/**`catch`**.

In the reporting method (the one you are invoking from the thread), add the stats to the list box as shown.

In your timer tick handler, check all threads for the stopped state. If they are all stopped, display the done message and disable the timer.

Rubric

This application will require visual inspection of functionality and code.

Mark loss is at your instructor’s discretion but will be applied consistently across all students.

Item	Marks	Penalties
UI Design (40%)		
• UI is as directed.	2	
• Anchoring is correct.	2	
• Formats for color ratios are correct.	2	
• Start/Stop Messages are correct.	2	
• Open File Dialog Filters Set as required	2	
• Open File Dialog allows choice of multiple files	2	
Code Design and Implementation (60%)		
• . Go button functions as described.	3	
• Individual threads created correctly for each file.	4	
• Threads stored/operated correctly in a List<Thread>	5	
• Threads are checked correctly for the stopped state.	3	
• Timer is used correctly to poll for all stop state.	3	
Documentation :		Penalties: -1 to -6 based on instructor’s judgement
• Programmer Block		
• Well commented code		
• Appropriate Variable Names		
• Proper spacing between blocks of code		
• Control names are consistent and appropriate.		

•