```
/*******************************************************************************
* Copyright 1990-2014 MBARI
* MBARI Proprietary Information. All rights reserved.
*******************************************************************************
* Summary  : Example: implements interface to ACM 2-D Current Meter
* Filename :
* Author   :
* Project  : Example: Benthic Rover
* Version  :
* Created  : Example: Nov 08
* Modified :
******************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <math.h>

#include "ImageFile.h"

// Function shows a concise help
//
void usage()
{
   const char* help =
   "Usage:\n"
   "sesia  [-d] [-r reference-image] [-m masking-image] image-file-a [image-file-b...]\n"
   "  -d   Debug mode produces a lot of info\n"
   "  -r   Use the reference-image file as a baseline for comparison\n"
   "  -m   Use the masking image to refine the area for comparison\n"
   "\nExamples:\n"
   " sesia  -r reference-image.b16  image-file-1.b16  image-2.b16  ../test-images/*.b16\n"
   " sesia  -r reference-image.b16  -m mask-image.b16  *.b16\n";

   printf(help);
}


int main(int argc, char** argv)
{
   // We will eventually set the debug flag using an option
   // sesia -d filename
   //
   bool debug = false;

   unsigned int threshold = 0;

   //
   char* ref_filename  = NULL;
   char* mask_filename = NULL;
   char* mod_filename  = NULL;
   int error = 0;
   int c;
   int fnames = 1;
```

```
    while ((c = getopt (argc, argv, "?w:dt:r:m:")) != -1)
    {
       switch (c)
       {
          case 'd':
             debug = true;
             fnames += 1;
             break;

          // Set the threshold value, entered as a percentage (0-100),
          // which is then converted into a pixel intensity (0-4096 = 2**12)
          case 't':
             if ((atoi(optarg) < 0) || atoi(optarg) > 100)
             {
                 printf("Invalid old value. Threshold value ranges from 0 to 100.");
                 error = 1;
             }
             threshold = atoi(optarg) * 40.96;
             fnames += 2;
             break;

         // Identify the reference file
          case 'r':
             ref_filename = strdup(optarg);
             if (strlen(ref_filename) <= 0)
             {
                printf ("Reference file: bad filename length!\n");
                error = 1;
             }
             fnames += 2;
             break;

         // Identify the masking file
          case 'm':
             mask_filename = strdup(optarg);
             if (strlen(mask_filename) <= 0)
             {
                printf ("Masking file: bad filename length!\n");
                error = 1;
             }
             fnames += 2;
             break;

          case 'w':
                mod_filename = strdup(optarg);
                 if (strlen(mod_filename) <= 0)
                 {
                printf ("Output file: bad filename length!\n");
                    error = 1;
                 }
             fnames += 2;
                 break;

             case '?':
                 error = 1;
                 break;

```

```
            default:
               printf ("bug\n");
            return (1);

       }
   }

   // Check for other errors (threshold limit, reference filename, etc)
   //
   if (error) {
      printf("Usage: sesia -r ref_filename imfile...]\n");
      return (2);
   }

   // Let's get it started
   //
   if (debug) printf("sesia processing...\n");

   // Check input. Show some help if invalid
   //
   if (argc < 2)
   {
      usage();
      return 1;
   }

   // Load the reference file
   ImageFile *ref = new ImageFile(ref_filename, debug);
   ref->load();
   if (mod_filename != NULL)
   {
     printf("Writing modified %s to %s\n", ref_filename, mod_filename);
      ref->dump(mod_filename);
   }

   // Load the mask file, if any
   //
   ImageFile *mask = NULL;
   if (mask_filename)
   {
      mask = new ImageFile(mask_filename, debug);
      if (0 != mask->load())
      {
         delete mask;
         mask = NULL;
      }
   }
   // Process the arguments
   //
   for (int i = fnames; i < argc; i++)
   {
      if (debug) printf("processing... %s\n", argv[i]);

      // Create an ImageFile object for this file
      ImageFile *ifile = new ImageFile(argv[i], debug);

      // Tell object to load file
      ifile->load();
```

```
        //for (int j = 0; j < 2448*2050; j++) printf("%d \n", ifile->_data[j]);

        // Tell object to process the file
        //float cov = ifile->calc_coverage(ref);
        //if (debug) printf("Coverage of %s is %f\n", argv[i], cov);
        float cov_t = ifile->calc_coverage(ref, mask, threshold);
        float intense_t = ifile->calc_intensity(ref, mask);

        printf("%s, %04d/%02d/%02d %02d:%02d:%02d, %f, %f\n", argv[i],
           ifile->_year, ifile->_month, ifile->_day, ifile->_hour, ifile->_min,
           ifile->_sec, fabs(cov_t), intense_t);

        delete ifile;
    }
    // do_stuff();

    // All done
    //
    return 0;
}

```