Short Program 4: Projection manager

Due: Friday, Sep 23 at 11:59 pm

Overview

Write a module named named *projection.c.* It must export three functions:

Function 1:

```
/**/
/* Initialize projection data */

proj_t *projection_init(
int  argc,          /* argc from main */
char **argv,        /* argv from main */
FILE *in);
```

It must operate as follows:

1. *malloc()* a structure of type *proj_t*
2. Fill in the *x* and *y* pixel dimensions of the window from command line parameters
3. Read in the *x* and *y* world coordinate dimension of the window from *in.*
4. Read in the *x, y,* and *z coordinates* of the viewpoint from *in.*
5. Return a pointer to the *proc_t* structure to the caller.

Sample input

```
8   6.25         world x and y dims
0 0 4.5          viewpoint (x, y, z)
```

Notes: Values are double precision floats so you *must* use the *%lf* format code to read them with *fscanf(in,....);* After reading the expected number of input values on each line, you must use *fgets(in, ...);* to consume the descriptive text which will be present in the input file.

Function 2:

```
void projection_dump(
FILE   *out,
proj_t *proj)
```

This function should write a "civilized" listing of the contents of the *proj_t* structure into the *out* file. Here is an acceptable example.

```
Window size in pixels
   100     75

Window size in world coordinates
  8.00    6.25

Location of Viewpoint
  0.00    0.00    4.50
```

Function 3:

```
void projection_test(
FILE *out,
proj_t *proj);
```

This function must fire a ray through every pixel in the window and for each pixel fired print exactly one line in exactly the order and format shown. The line must contain the *x* pixel coord, *y* pixel coord, *x* world coord, *y* world coord of the point through which the ray passed along with the *x, y,* and *z* components of a UNIT length vector in the direction that the ray is traveling. This function will rely upon *map_pix_to_world(), vl_diff3(), and vl_unitvec3().*

Sample output from

a.out 4 3 < proj.txt 1> proj.map 2> proj.err

```
   0    0  -4.00  -3.12  -0.59  -0.46  -0.66
   1    0  -2.00  -3.12  -0.34  -0.54  -0.77
   2    0   0.00  -3.12   0.00  -0.57  -0.82
   3    0   2.00  -3.12   0.34  -0.54  -0.77
   0    1  -4.00  -1.04  -0.65  -0.17  -0.74
   1    1  -2.00  -1.04  -0.40  -0.21  -0.89
   2    1   0.00  -1.04   0.00  -0.23  -0.97
   3    1   2.00  -1.04   0.40  -0.21  -0.89
   0    2  -4.00   1.04  -0.65   0.17  -0.74
   1    2  -2.00   1.04  -0.40   0.21  -0.89
   2    2   0.00   1.04   0.00   0.23  -0.97
   3    2   2.00   1.04   0.40   0.21  -0.89
```

Other resources

A sample main file, header file, input, and output files are available in the *sp4data* directory. If you build a.out and run (using the *bash* shell)

a.out 4 3 < proj.txt 1> proj.1 2> proj.2

Your output should match what is in the *sp4data* directory.

How to submit your program:

NOTE: This procedure has NOTHING in common with "handin" nor "sendlab"
Do NOT even TRY to think about how they fit into this procedure because
THEY DON'T!!

<<<Do NOT turn in any image files, core files, makefiles etc.>>>

You must turn in 2 files: *projection.c* must contain the three functions described previously along with your *map_pix_to_world()* function. You must also turn in *veclib.c* which contains the functions you wrote in lab.

Do NOT turn in a .tar file.

But if you need assistance, DO e-mail me a tar file containing precisely the .h and .c files required to build your program and also containing the input you are attempting to use.

After you think you have turned your programs in, its a good idea to

cd /local/jmw2/215/sp4/wjsmith

and make sure your files are their and they still compile and work correctly.