Short Program 5: Object loaders and dumpers

Due: Friday, Sep 30 at 11:59 pm

Overview

In this assignment you will create three *four* new modules: object.c, sphere.c, plane.c, and material.c. These modules will contain the following functions described below. You must add the proper function prototypes to rayhdrs.h.

object.c:

```
/**/

/* Constructor for generic object type */
obj_t *object_init(
FILE   *in,
int objtype);
```

Thes function should *malloc()* a new *obj_t* structure, set the *objtype* and the *objid* fields and invoke *material_init()* to read in theambient, diffuse, and specular reflectivity of the object.

material.c

```
/**/
int material_init(
FILE   *in,
material_t  *mat);
```

This function should use the standard fscanf()/fgets() combination to read reflectivity and consume descriptive text. Because this action is repeated in many contexts. I built a collection of *vl_getn()* functions that will read 1, 2, or 3 values into a vector. An alternative would be to build a single *vl_get()* function that could be passed the number of values it is supposed to get. Its also OK to just do it the hard way (call fscanf()/fgets() three times). However, it is REQUIRED that you include some mechanism for detecting defective input and aborting.

```
   int sum = 0;

   sum += vl_get3(in, mat->ambient);
   sum += vl_get3(in, mat->diffuse);
   sum += vl_get3(in, mat->specular);

   if (sum != 9)
   {
      fprintf(stderr, "Matload failed with %d values \n", sum);
      material_dump(stderr, mat);
      return(-1);
   }
```

```
/**/
void material_dump(
FILE *out,
material_t *mat);
```

This module should print a reasonably formatted listing of the material structure. See the example output.

sphere.c:

```
/**/
obj_t    *sphere_init(
FILE     *in,
int      objtype);
```

This function should invoke *object_init()* to create and initialize the common components of a new *obj_t*. On succesful return to *sphere_init(),* a new *sphere_t* structure should be allocated and the *priv* pointer of the *obj_t* structure be set to point to it. Then the location of the center and the radius should be read in. The address of the *obj_t* should be returned to model init.

```
int sphere_dump(
FILE *out,
obj_t *obj);
```

This function should print a reasonably formatted listing of the attributes of a sphere type object. It should use the *material_dump()* function to print the reflectivity.

plane.c:

```
/**/
obj_t    *plane_init(
FILE     *in,
int      objtype);
```

This should work in a manner analogous to *sphere_init()* but should read in the plane's normal vector and the coordinates of a single point on the plane.

```
int plane_dump(
FILE *out,
obj_t *obj);
```

This should work in a manner analogous to *sphere_dump()*.

Development approach

If you try to write all of this stuff before you test any of it you almost certainly will create an unmitigated disaster. Here is the proper order to write and test. NOTE you will need to create several auxillary "main.c" test drivers to do this:

(1) material_load()/material_dump()

(2) object_init()

(3) sphere_init() sphere_dump()

(4) plane_init() plane_dump()

Here is a sample main() for step (1)

```
#include "ray.h"
main()
{
   material_t mat;
   material_init(stdin, &mat);
   material_dump(stdout, &mat);
}
```

The input for *this* test should be

```
1 2 3 ambient
4 5 6 diffuse
7 8 9 specular
```

Other resources

A sample main.c, model.c , header files, input (sp5.txt), and output (sp5.2) files are available in the *sp5data* directory. If you build a.out and run (using the *bash* shell)

a.out 400 300 < sp5.txt 1> sp5.1 2> sp5.2

Your output should match what is in the *sp5.2 file* in the *sp5data* directory. The sp5.1 file *must* contain NO data because that is where your .ppm image will eventually go.

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 the following .c files:

main.c
model.c
plane.c
object.c
veclib.c
list.c
projection.c

and any .h files that are needed to build your program.

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/sp5/wjsmith

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