The model_t object (lab5) and mtx_t "class" (lab6)

Wednesday, September 15, 2010
8:54 AM

- up to now we've used two lists: objs, mats for objects & materials, resp.
- Now we want to construct a model_t *model object that holds both lists (and other stuff)
- Here's a rough idea of what we want (p. 90 of the notes)

```
int main(int argc, char *argv[])
```

e.g., % ./main plane.txt hittest.txt

Linux prompt

argv[0] argv[1] argv[2]

argc = 3

```
{
    model_t   *model = NULL; // declare & init model

    model = model_init(stdin);
```

```
model = model_init(stdin);
model_print(model, stdout);
}
```

prints materials & objects lists

---

updates to Makefile:

```
:
OBJS = \
pixel.o \
material.o \
object.o \
plane.o \
model.o

all: libvec libist main

libvec: libvec.a
libvec.a: vector.o matrix.o
	ar rcs $@ $?
	ranlib $@
```

;

Wednesday, September 15, 2010
9:15 AM

```
typedef double vec_t[3];
vec_t  v1; // v1[0], v1[1], v1[2]
```

matrix.h :

```
typedef double mtx_t[3][3]; // ( m[0][0] m[0][1] m[0][2]
                            //   m[1][0] m[1][1] m[1][2]
                            //   m[2][0] m[2][1] m[2][2] )
```

like a vec_t ⟶ m[0] : the 0th row (1D array)
m[1] : the 1st row (1D array)
m[2] : the 2nd row (1D array)

```
void vec_xform(mtx_t, vec_t, vec_t);
                ↑      ↑      ↑
              input  input  output

void mat_xpose(mtx_t, mtx_t);
                ↑      ↑
               in     out
```

---

model.h :

```
#ifndef NAME_LEN
#define NAME_LEN 16
```

```
#endif

typedef struct model_type
{
    list_t      *mats;
    list_t      *objs;
} model_t;

model_t* model_init(FILE*);
```

model.c :

```
#include <stdlib.h>
        <stdlib.h>
        <string.h>
        <assert.h>
        <math.h>

#include "vector.h"
        "pixel.h"
        "list.h"
        "material.h"
        "object.h"
        "model.h"
```

depends on definitions in list.h & material.h & object.h so need to be at bottom

```
model_t * model_init(FILE *in)
```

could be stdin

```
{
    model_t *model;
```

```
material_t   *mat;
object_t     *obj;
char         token[16];

// create new model & zero it
assert((model = (model_t *)malloc(sizeof(model_t))) != NULL);
memset(model, 0, sizeof(model_t));

// create lists
model->mats = list_init();
model->objs = list_init();

// input model file (e.g., plane.txt)
while (fscanf(in, "%s", token) == 1) {
        : // as in lab 4
        material_init(in, model->mats, 0);
        :
       mat = (material_t *) list_get_data(model->mats);
        :
        plane_init(in, model->objs, model->mats, 0);
        :
        obj = (object_t *) list_get_data(model->objs),
```

```
    obj = (object_t *)MST_get_down(model->obj),
}
return(model);
}
```

Wednesday, September 15, 2010
9:35 AM

matrix.c:

```
#include <stdio.h>
"       <stdlib.h>
"       <string.h>
"       <assert.h>
"       <math.h>

#include "vector.h"
#include "matrix.h"

void mat_xpose (mtx_t m1, mtx_t m2)
{
```

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \rightsquigarrow \begin{pmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{pmatrix}$$

m1 m2

```
    int i,j;
    for(i=0; i<3; i++)
```

```
    for ( j=0; j<3; j++)
        m2[i][j] = m1[j][i];
}

void vec_xform (mtx m, vec_t v1, vec_t v2)
```

$$\begin{pmatrix} v2[0] \\ v2[1] \\ v2[2] \end{pmatrix} = \begin{pmatrix} m[0][0] & m[0][1] & m[0][2] \\ m[1][0] & m[1][1] & m[1][2] \\ m[2][0] & m[2][1] & m[2][2] \end{pmatrix} \begin{pmatrix} v1[0] \\ v1[1] \\ v1[2] \end{pmatrix}$$

```
    int i;

    for(i=0; i<3; i++) v2[i] = vec_dot(m[i], v1);
}
```