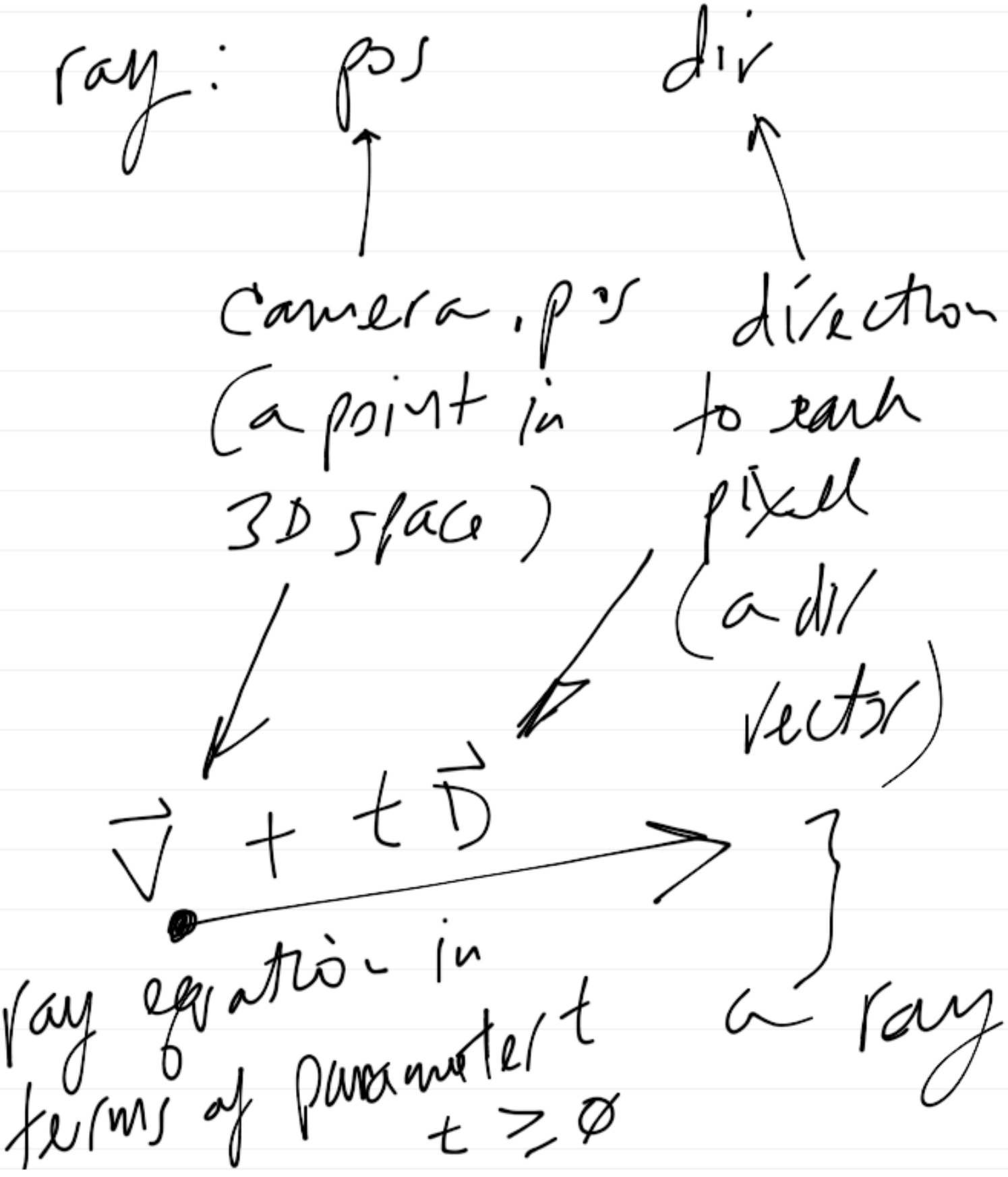
We now have:

ray: pos dir



- where does camera info come from?
  - our scene description or model file
- ray tracing alg:
  1. load (read/parse) the world model:
     - a list of objects (we don't know how many)

spheres, cones, ...

- a list of materials
(surface properties)
(green, yellow, ...)
↑
diffuse
ambient
specular

- a camera
image size (w, h)
world size (ww, wh)
position (x, y, z)

② for each pixel in the image,

- init color of pixel to (0,0,0) (R,G,B)
- spawn ray thru pixel we know $\vec{pos}$

$$\vec{dir} = \frac{\vec{pixel} - \vec{pos}}{\|\vec{pixel} - \vec{pos}\|}$$

- trace ray: returns color
- add color to pixel

ray trace:

- find closest obj to ray (known as the hit point, point on surface such that distance is minimum (smallest t))
- get surface ambient color, scale by 1/t

③ - convert colors

([0,1], [0,1], [0,1])

R G B

to unsigned characters

[0,255], [0,255], [0,255]

8 bit bytes

- output ppm image

excerpt from input file:

token

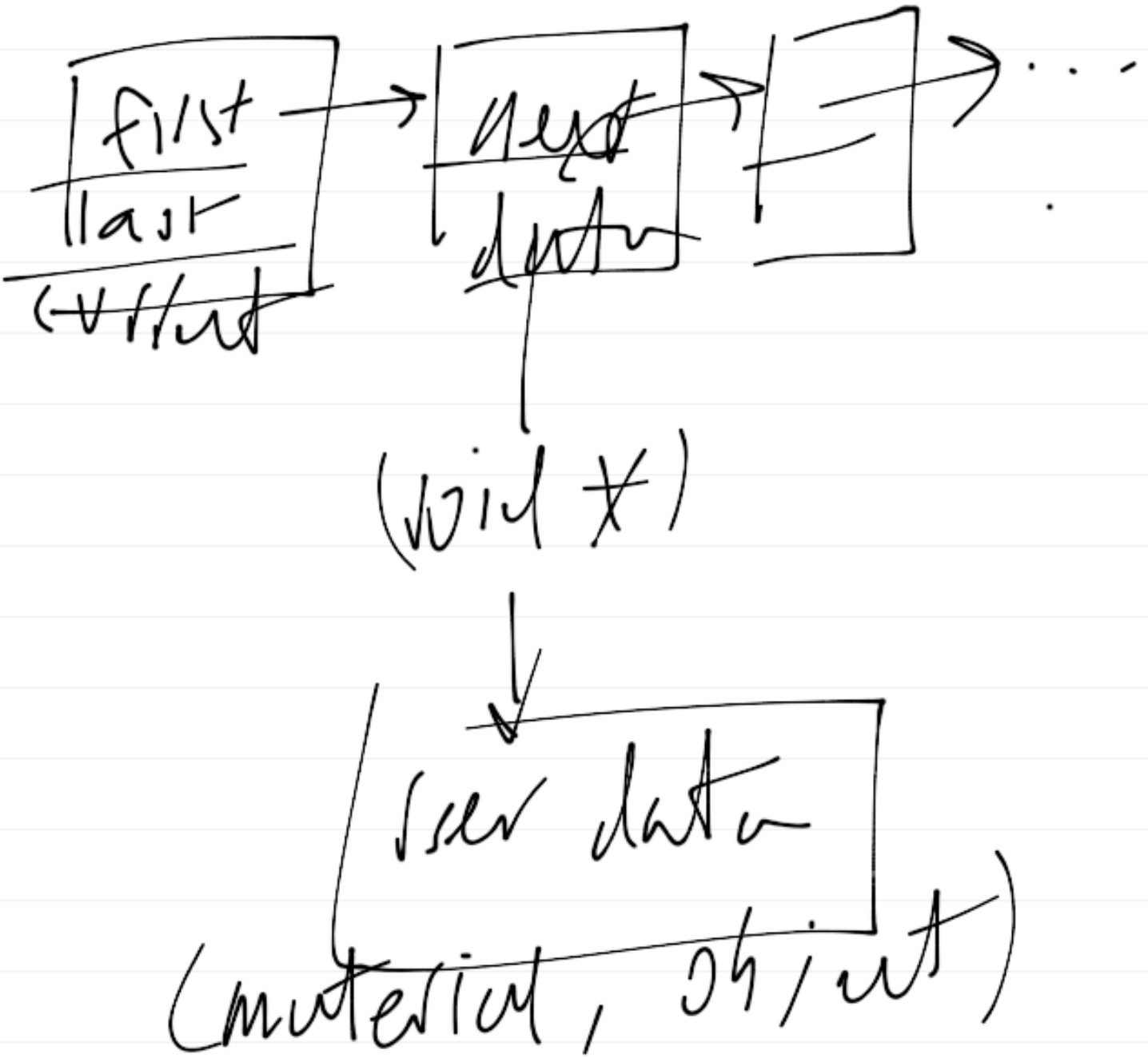
```
camera cam1
{
    pixel dim 640 480
    world dim 8 6
    viewpoint 4 3 6
}
```

whitespace

delimiters

tokens (any order)

- need a container (list)
  data structure



- first, use list to store materials:

e.g.,

material yellow

{

diffuse 4 4 0

ambient 5 4 0

specular 1 1 1

}

material green
{
  ambient 0 5 0
}

material data type:

typedef struct material_type
{
  int cookie; // magic no. like a
  char name[NAME_LEN];
  rgb_t ambient;
  rgb_t diffuse, specular;
} material_t

drgb_t : from pixel.h

a double[3]

just like vec_t

first method (member function)
for material_t:

"constructor" to allocate
mem & initialize
(& in this case, also
read from file, AND
add to list)

materials_init (FILE *in,
list_t *list, int attrmax)

notes

ignore
(p. 48)

have
model_t *model

idea is to:

1. malloc a material_t struct

material_t *mat =

(material_t *) malloc

(sizeof (material_t));

2. init values to

(0,0,0)

3. load material attributes
(read FILE *in)
call material_load_
attributes(in, mat)
4. add mat to list
list_add(list,
(void *) mat);

```
material_load_attributes
    (FILE *in, material_t *mat)

// assume we've been
called just after token
"material" has been
read in
@ read in name/label
fscanf(in, "%s", mat->name);
```

while ((c = fgetc(in)) != EOF &&
c != '\n');

while ((c = fgetc(in)) != EOF
&& c != '{');

→ Consume '\n' again

fscanf(in, "%s", attr(name));

```
if (!strcmp(attr_name, "diffuse")
    rec_read(in, mat->diffuse);
if (!strcmp(attr_name, "ambient"))
    rec_read(in, mat->
    ========          ambient);
    (get drgb_t to
        read itself)
if (!strcmp(attr_name,
        "specular"))
    rec_read(in, mat->specular)
```

put previous three if lines into loop that "peeks" at next char, testing for '}'

```
while((c=fgetc(in)) != EOF
    && c != '}')
{
    fputc(in, c)
    if ...
    if ...
    if ...
}
```