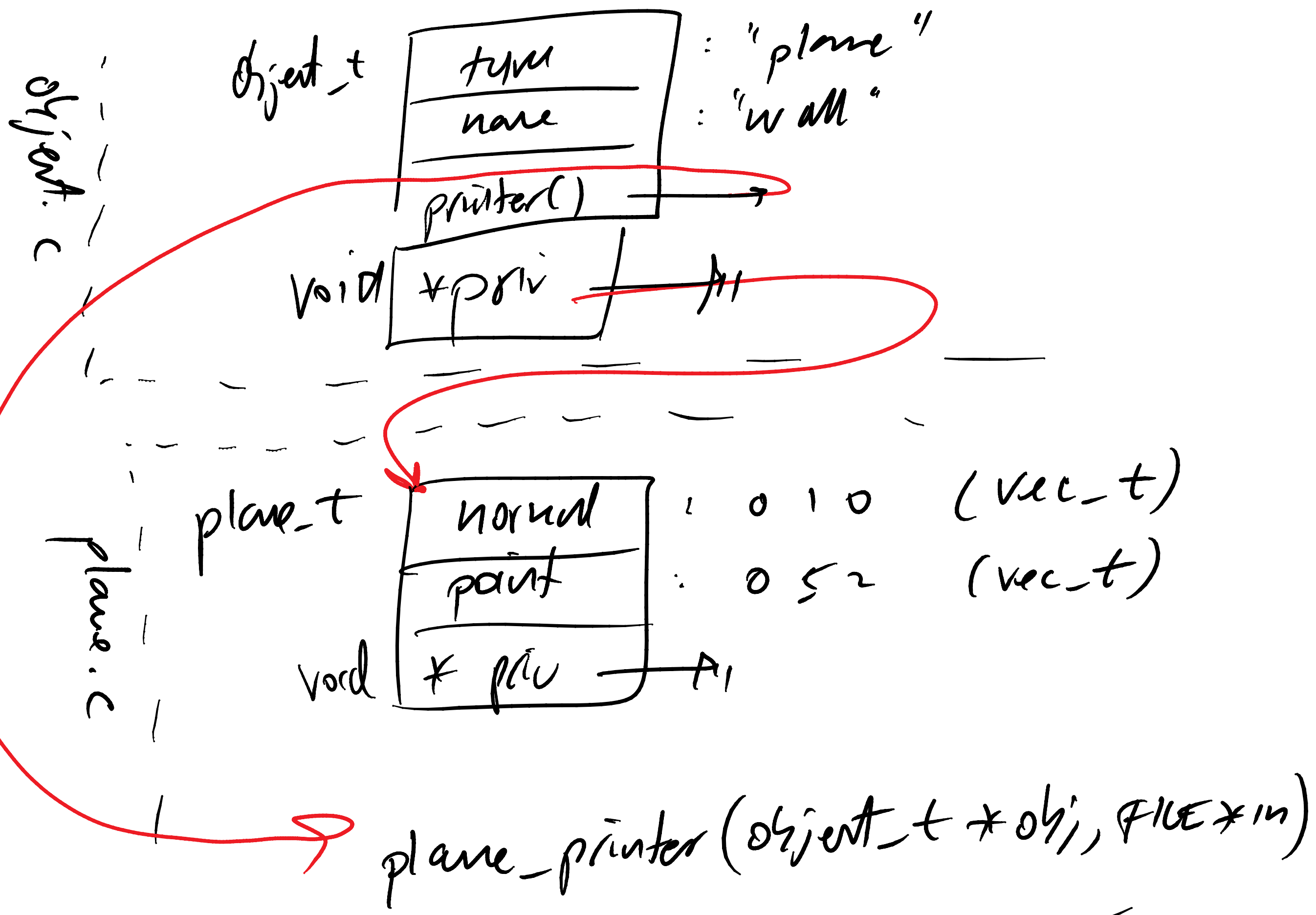
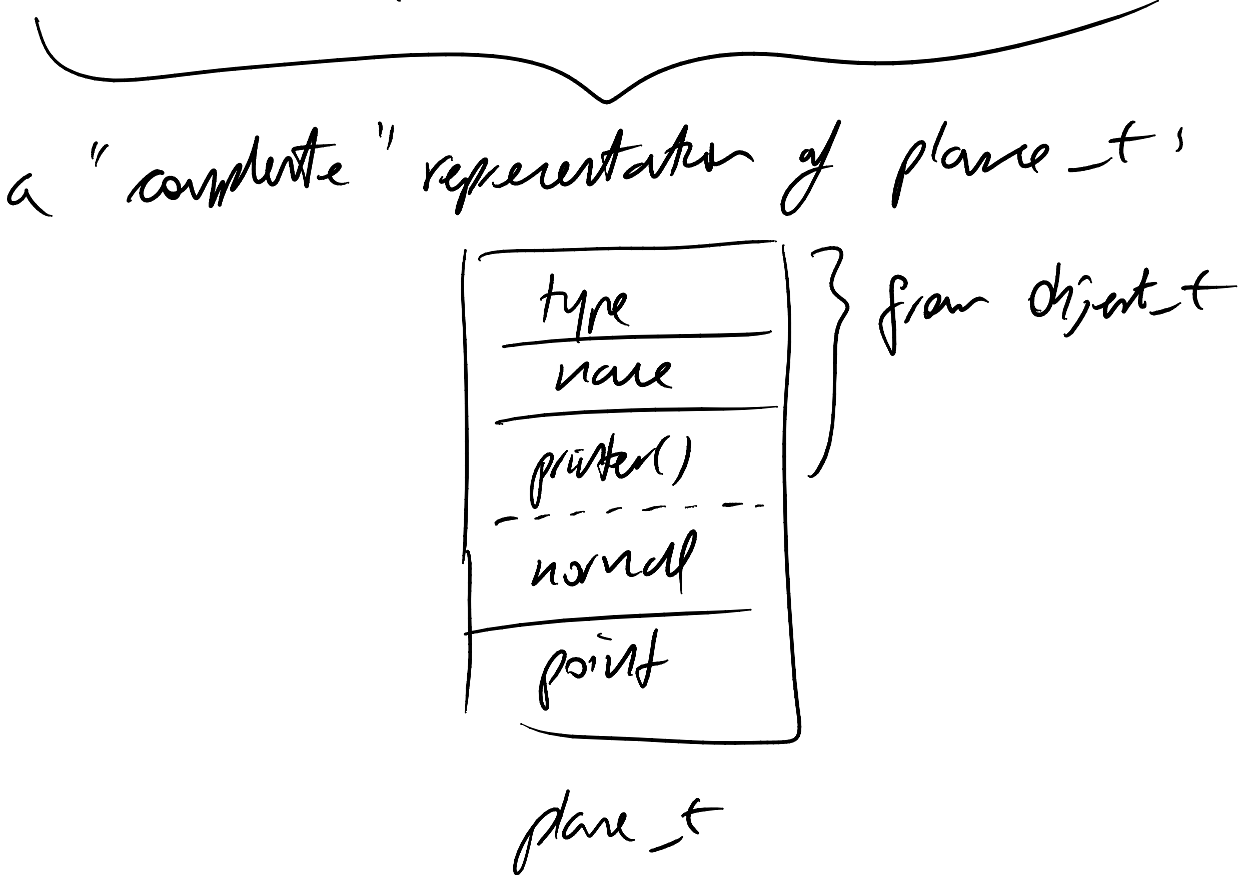
Initializing object and plane "objects"

Friday, September 10, 2010
9:01 AM





Friday, September 10, 2010
9:22 AM

$$\text{void object\_init}(\underbrace{\text{FILE *in}}_{\text{input file}},\ \underbrace{\text{list\_t *objs}}_{\text{list of objects (generic)}},\ \underbrace{\text{list\_t *mats}}_{\text{list of materials}})$$

{

```
char matname(NAME_LEN);

// objnames :
    1. malloc generic object
    2. zero it out
    3. read in just generic part
            (material)
    4. add new obj onto objs list
```

$$\text{assert}(\underbrace{(\text{obj} = (\text{object\_t *})\text{malloc}(\text{sizeof}(\text{object\_t})))}_{\text{obj}} \,!= \text{NULL});$$

```
memset(obj, 0, sizeof(object_t));

        - or -

obj->type = '\0';
obj->name = '\0';
obj->printer = NULL;
obj->prv = NULL;
```

type	0
name	0
printer	NULL
prv	NULL

// now parse file
// <u>assume</u> file ptr
// is at object name
// e.g., "wall"

```
plane wall
{
      material blue
      normal   0 1 0
      point    5 7 2
}
```

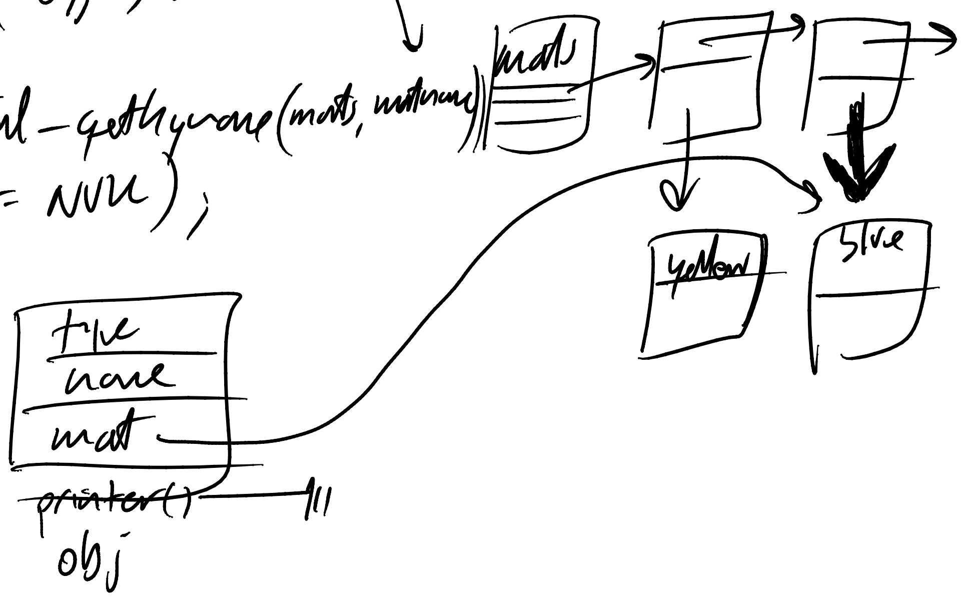
```
assert ((count = fscanf (in, "%s", obj->name))==1);
// consume all chars until we get to '{'
while ((c = fgetc (in)) != EOF && c != '{');

// set cookie
obj->cookie = OBJ_COOKIE;
// read in material (first attribute MUST be material)
assert ((count = fscanf (in, "%s", attrname)) ==1);
// read material name (id) (e.g., "blue")
if ( ! strcmp (attrname, "material"))
    assert ((count = fscanf (in, "%s", matname)) = 1).
```

// find material in mats list

material "blue" must have been read in from file already ; it be in mats list

mats list   like already ;
                                                              will be in mats list

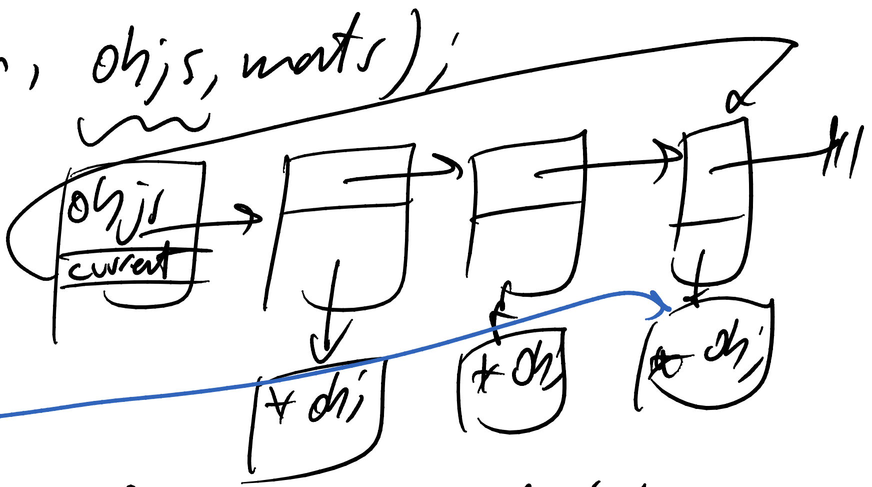
```
assert((obj->mat =
    material_getbyname(mats, matname))
        != NULL);
```



```
    // consume whitespace

    // add to objs list
    list_add(objs, (void *)obj);
}
```

Friday, September 10, 2010
9:43 AM

```
void plane_init(FILE *in, list_t *objs, list_t *mats)
{
                    object_t *obj;  plane_t *ply;
    object_init(in, objs, mats);
```



```
    obj = (object_t *) list_get_data(objs);

    // Allocate plane object
    // Zero out plane object
    // set object's type to "plane"
    strcpy(obj->type, "plane");

    // link generic obj. structure to plane
    // & set obj's ftn ptr
```

```
obj→priv = (void *) pln;
obj→printer = plane_print;
```

$\underbrace{\text{plane\_print}}$ function name found in plane.c

```
// continue parsing file
// read in normal vec_t
// read in point vec_t
```

} similar to material

```
// normalize vector
vec_unit(pln→normal, obj→last_normal);
vec_unit(pln→normal, pln→normal);

pln→ndotq = vec_dot(pln→point, pln→normal).
}
```