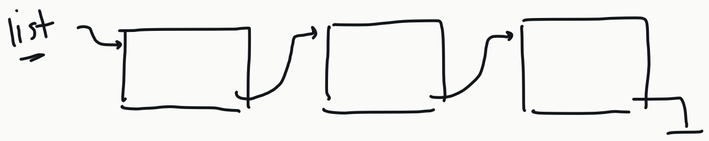
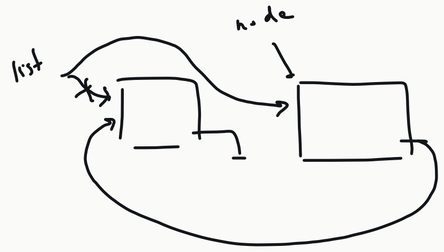
linked list




```
OBJ_T *list;
OBJ_T *node;
list = NULL;

while (new object) {

    node = (OBJ_T *) malloc (sizeof (OBJ_T));
    node -> sphere.ctr.x = 0;
     :
     :
    node -> next = list;
    list = node
```




```
typedef struct OBJ {
    ...
    ...
    struct OBJ *next;
    ...
} OBJ_T;
```

rt.h

```
SCENE_T
    OBJ_T *objs;
    LIGHT_T light;
     start_x
     start_y
     pixel_size
```

File I/O

```
    stdlib.h   filename
    FILE *fp;  / w
    fp = fopen (fn, " ");
fprintf (fp, "%c%c%c", ...);
```

img file output

```
fscanf (fp, "%c", &c)
```

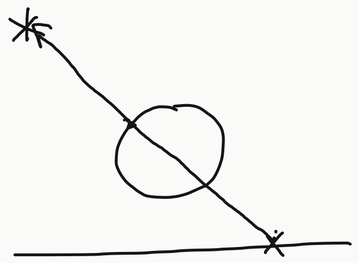
files
- rt.h — all objects here
- rt.c trace main init
- sphere.h fn proto
- sphere.c intersect_sphere
- plane.h
- plane.c
- light.h
- light.c illuminate shadow_test
- vp.h
- vp.c

trace

```
OBJ_T *curr;
for (curr = scene.objs; curr != NULL;
                        curr = curr -> next)
    if ((*curr -> intersect)( ... )
```

light.c

```
static int shadow_test
                int_pt
shadow_ray = light.loc - int_pt
shadow_ray = vp_subtract (light.loc, int_pt)
//normalize shadow_ray
for (objs)
    if intersect
        return 1
```



illuminate
- after ambient
- before diffuse specular