debugging

a) print statements

b) debugger (gdb) gdb ./main

break <file>:<line>

r <args>

p <var>

n <next>

s <step into fun>

where

linked list member functions

list_add(list_t *list,
void *data)

1. get a link

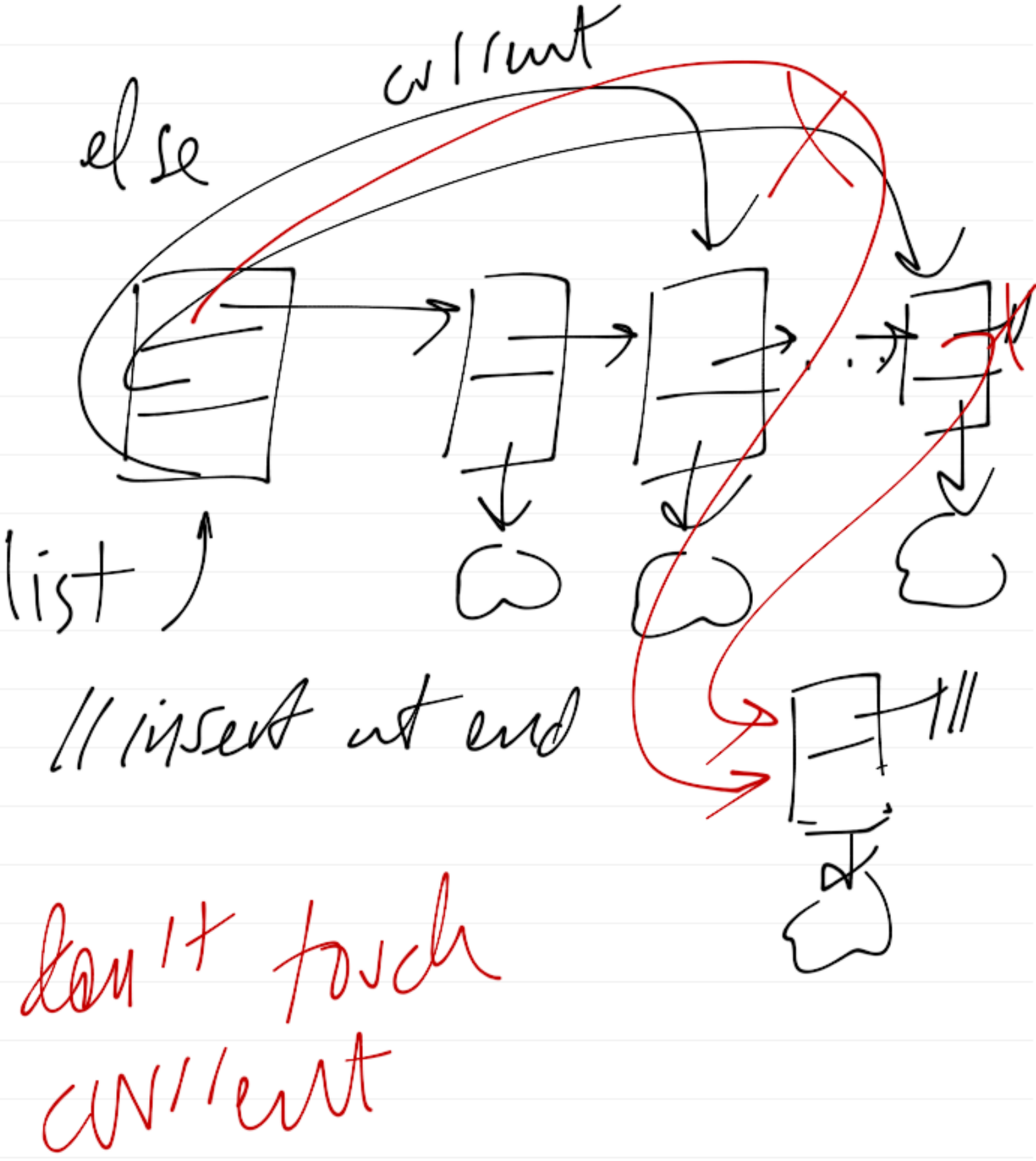
link_t *link

2. if list is empty

convention:

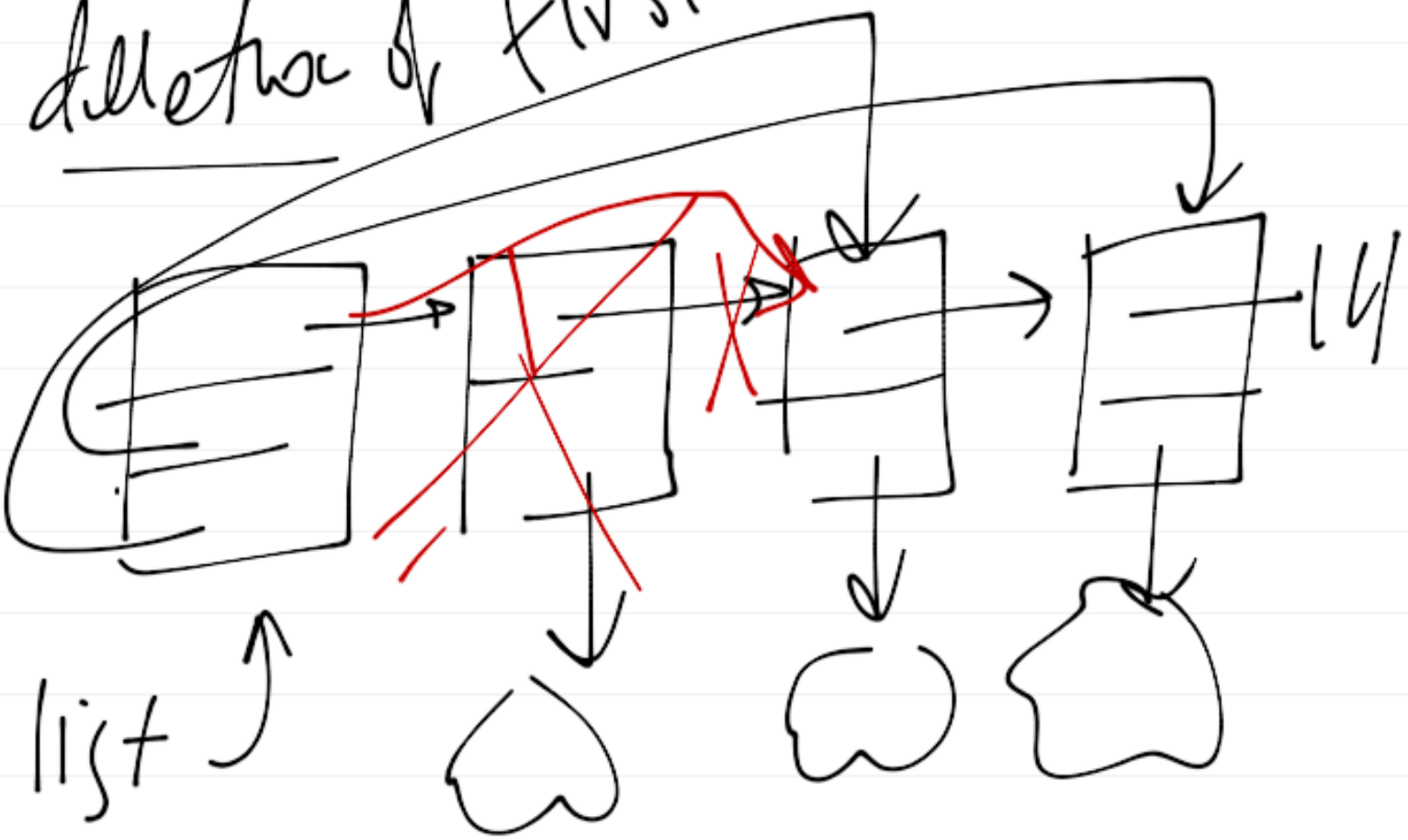
set current to first

only when list empty


current
else
list
// insert at end
don't touch current

```
link_t *link = link_init
                    (data)
if (list_empty(list)) {
    list->first = list->last =
        list->current = link;
} else {
    list->last->next =
                    link;
    list->last = link;
} // don't touch current
```

deletion of first



list → first =

list → first → next;

what to do with current?

one way to handle current pointer is just to reset if to point to first

```
void list_fdel(list_t *list)
{
    link_t *link;

    assert(list->first != NULL);

    -OR-

    if(list_empty(list))
        // do nothing
        // issue warnings

    -OR-
```

```
link = list->first;

list->first =
    list->first->next
(same as link->next)

list_reset(list);
    // sets current to
       front
```

free(link->data)

// free user data

free(link)

}

really nasty — list should not mess with user data

Note: in future versions of list, list_fdel should return ptr to user data

---

```
void *list_fdel(...)
    void *vdata;
vdata = link->data;
free(link); return vdata
```

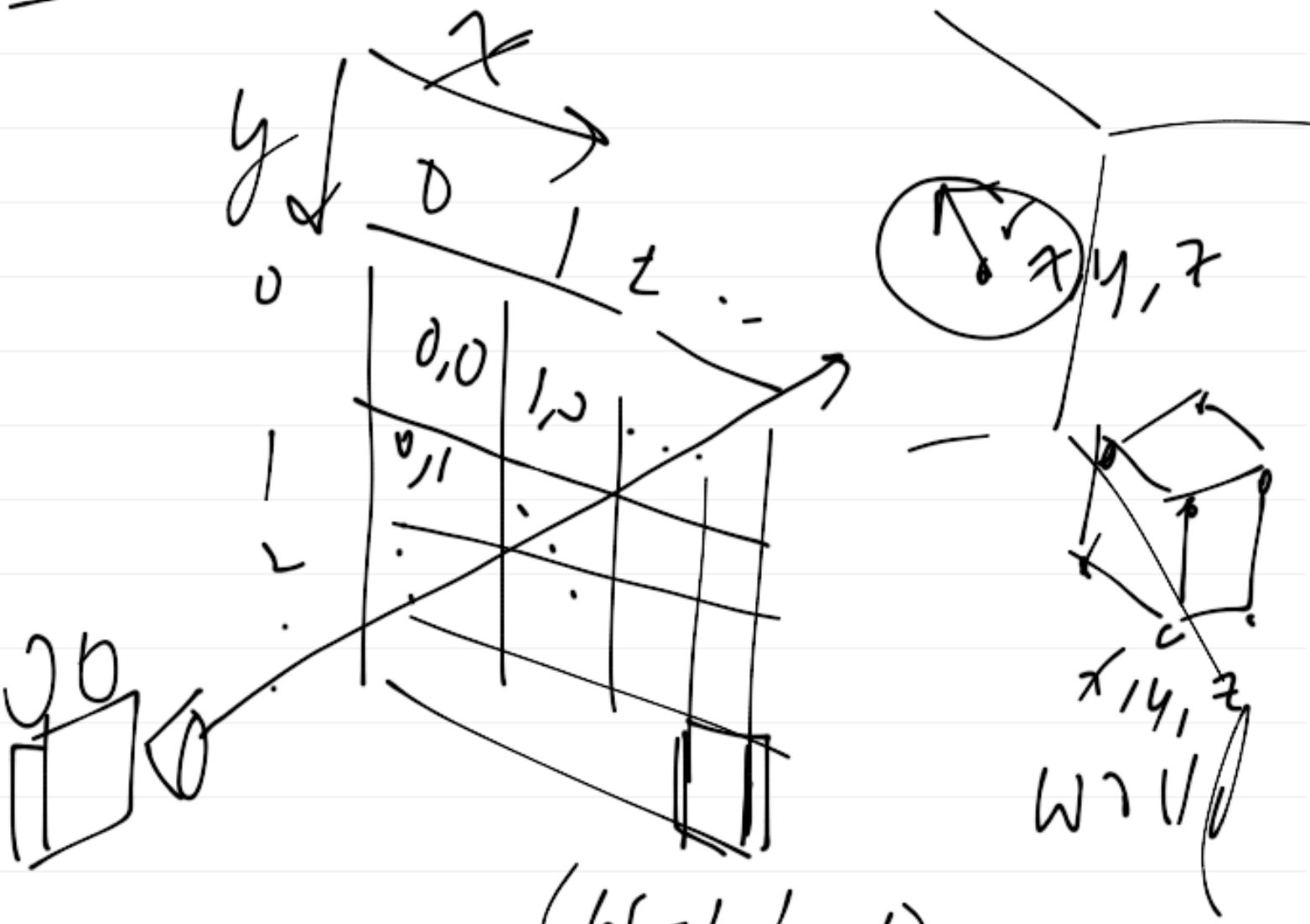
How to delete entire list ?

```
list_del(list_t *list)
{
    list_reset(list)
    //list->current = list->first
    while(list_not_end(list))
        list_fdel(list)
}
```

```
list_not_end(list):

return list->current != NULL?
    1 : 0;
```

Ray Tracing Intro



Camera pos = (x, y, z) in world coords

image plane in pixels e.g. 640 × 480 screen coords

- image pixels in screen coordinates $x \in [0, 640]$

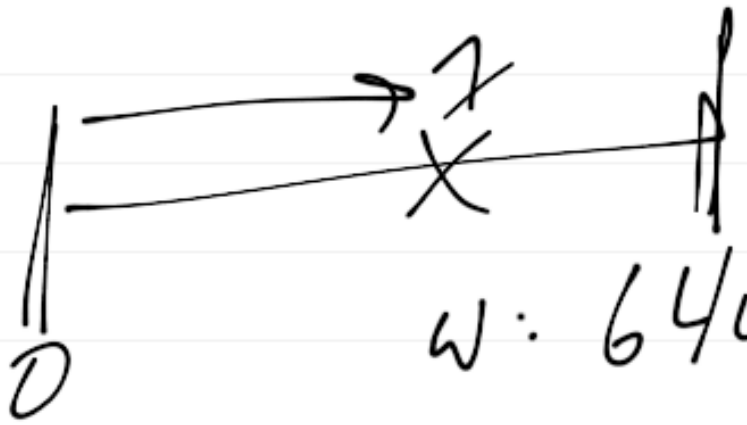
$y \in [0, 480]$, $z = 0$

world coords

- iterate thru screen coords, convert to world coords given world dimensions of width, height

e.g. 8' x 6'

- use linear mapping
(linear interpolation)
to convert screen to
world coords



$\frac{x}{w}$ normalizes $x \to [0,1]$

range

$x' = \frac{x}{w} * w'$

$x' \quad 8' = w'$

e.g. pixel at (100, 40)

integers

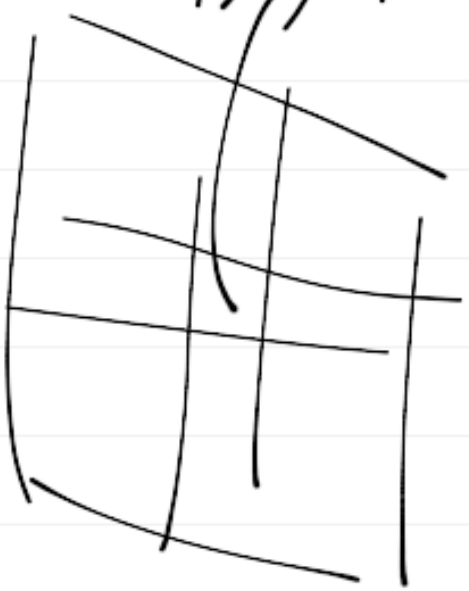
$$\frac{100}{640} \times 8.0 = 1.25$$

$$\frac{40}{480} \times 6.0 = 0.5$$

1.25, .5, 0

100, 40

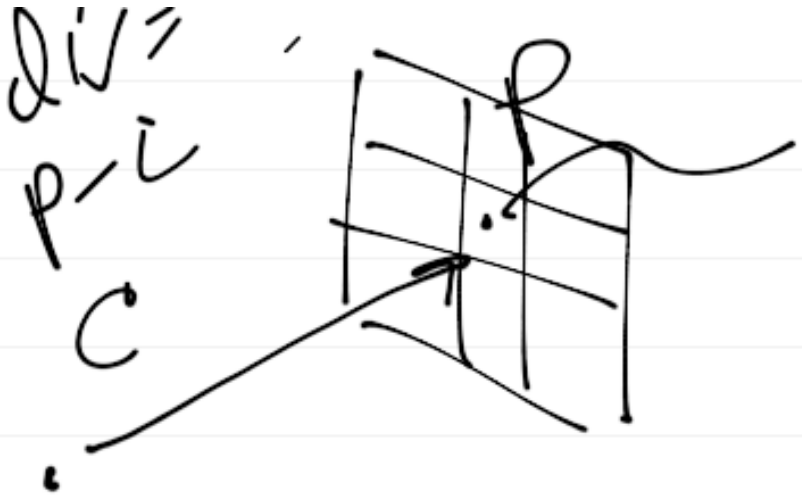
$$z = 0.0$$



goal: Compute ray direction from camera pos to each pixel

```
int x, y;
int w = 640, h = 480;
for(y=0; y<h; y++) {
  for(x=0; x<w; x++)
```

```
for(y=0; ... ) {
  for(x=0.- ) {
    float worldw = 8.0;
    float worldh = 6.0;
    float worldx =
    (float)x / (float)w *
    worldw;
    //same for worldy
    float worldz = 0.0;
```



cx, cy, cz

```
Vec_t pixel(worldx,
            worldy
            worldz);

Vec_t div;

Vec_Diff(C, pixel, div)
```

↑ Comes from input file.

```
Vec unit(dir, dir)
    //normalize dir
    :
    //more to come later
}
}
```