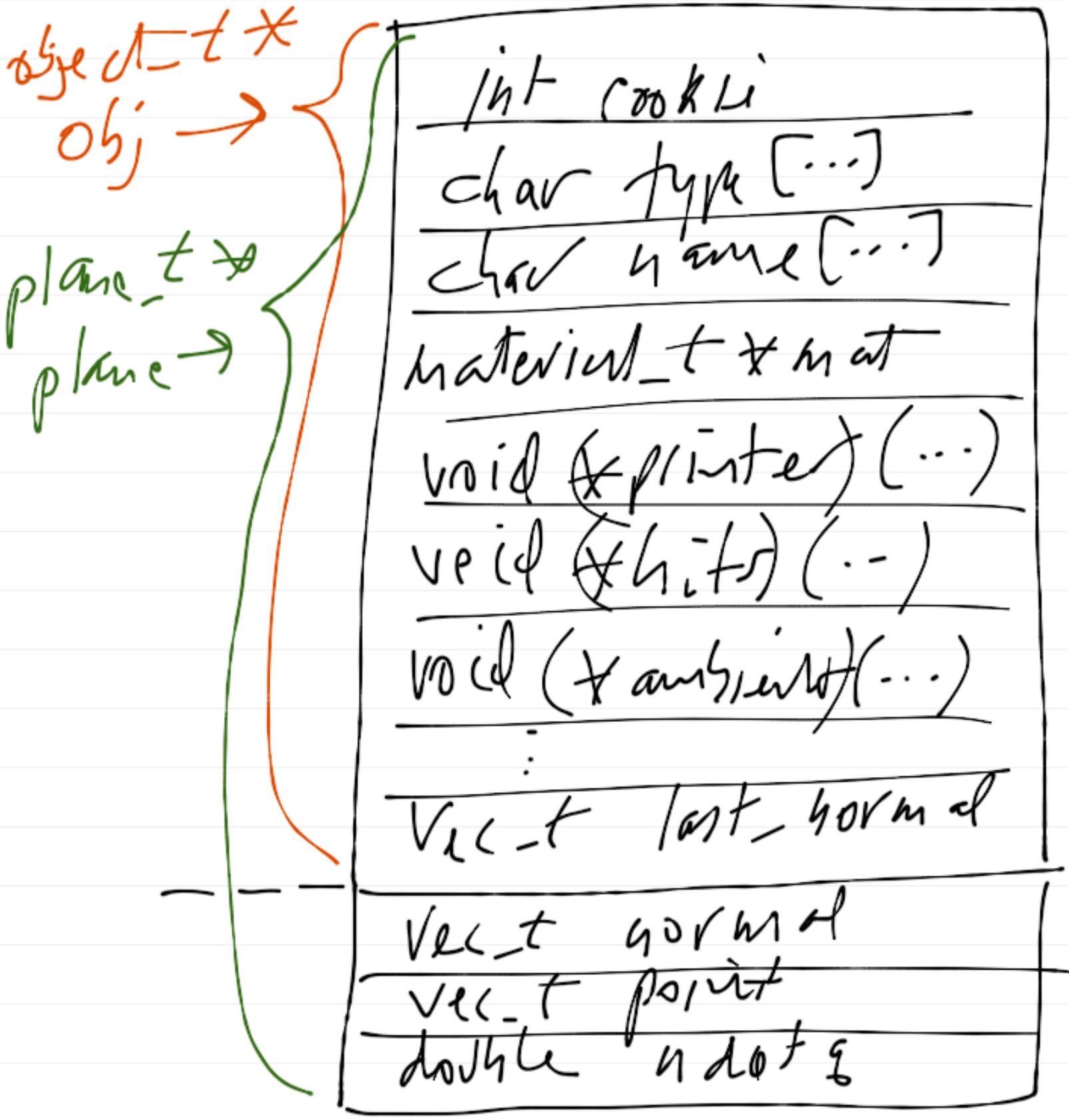
from last time:



hits function:

- like object_print()
  & plane_print()

⇒ virtual function

⇒ plane_print()
overrides object_print()

[plane_print calls object_print
but plane→hits()
does not call obj→hit()]

Recall:

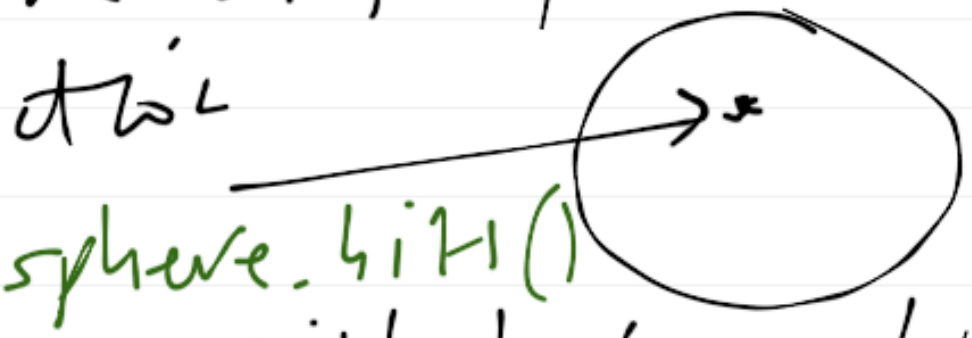
```
plane_print(object_t *obj,
                FILE *out)
{
    //print_first(common)
    // part of plane object
    //   plane world
        {
            material flag
    object_print(obj);
    // now gain access to
    // rest of data struct
    plane_t *pla =
        (plane_t *) obj;
```

type cast →

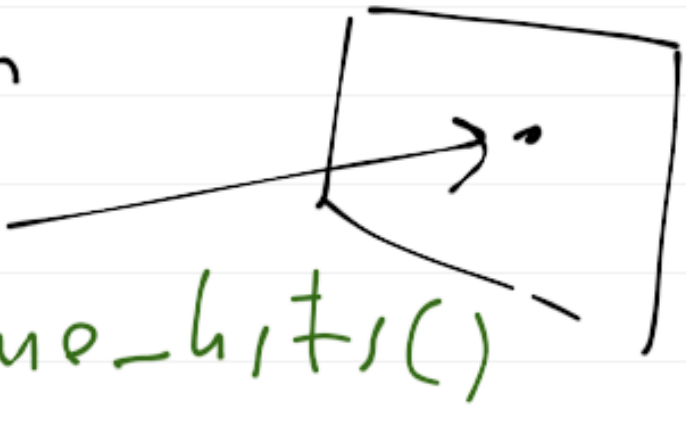
```
// and print out rest of plane
//      normal  0 1 0
//      point   3 0 0
// }
  vec_print(out, "  ",
        pln->normal);
  vec_print(out, "  ",
        pln->point);
  fprintf(out, "}\n\n");
}
```

hits() works the same way:

- sphere will have a hits() function



- plane will have a hits() function



- during plane_init():

```
obj->hits = plane_hits;
```

- during sphere_init():

```
obj->hits = sphere_hits;
```

revisit:

```
plane_init(FILE *in,
           list_t *objs,
           list_t *mats,
           int attrmax)
{
  // allocate memory:
  plane_t *pln =
    (plane_t *) malloc(sizeof(
                  plane_t));
  if (pln == NULL) // error
  // zero out
  memset(pln, 0, sizeof(plane_t))
```

```
// init parent structure
object_t *obj =
    (object_t *)pln;
```

typecast, no malloc

```
object_init(obj, in, objs, mats);

// set type string
strcpy(obj->type, "plane");
// overload virtual functions
obj->printer = plane_print;
obj->hits = plane_hits;
```

```
//continue parsing file
while((c=fgetc(in))!=EOF
    && c!='}') {
    fputc(c,in);
    fscanf(in,"%s",attrname)
    if(!strcmp(attrname,"normal"))
        vec_read(in,plt->normal)
    if(!strcmp(attrname,"point"))
        vec_read(in,plt->point)
    while((c=fgetc(in))!=EOF
    && c!='\n');
```

```
// finished parsing, set parent
// values
vec_unit(pln->normal,
         obj->last_normal);
// normalize pln->normal
vec_unit(pln->normal,
         pln->normal);
// set ndotc
pln->ndotc = vec_dot(
    pln->point, pln->normal);
}
```

remarks:

```
object_init(object_t *obj,
            FILE *in,
            list_t *objs,
            list_t *mats)
{
  obj->cookie = OBJ_COOKIE;
  fscanf(in, "%s", obj->name);
  // parse first part of object
  while((c=fgetc(in)) != EOF
        && c != '{');
  fscanf(in, "%s", attrname);
```

"material"

```
fscanf(in, "%s", matname);
```

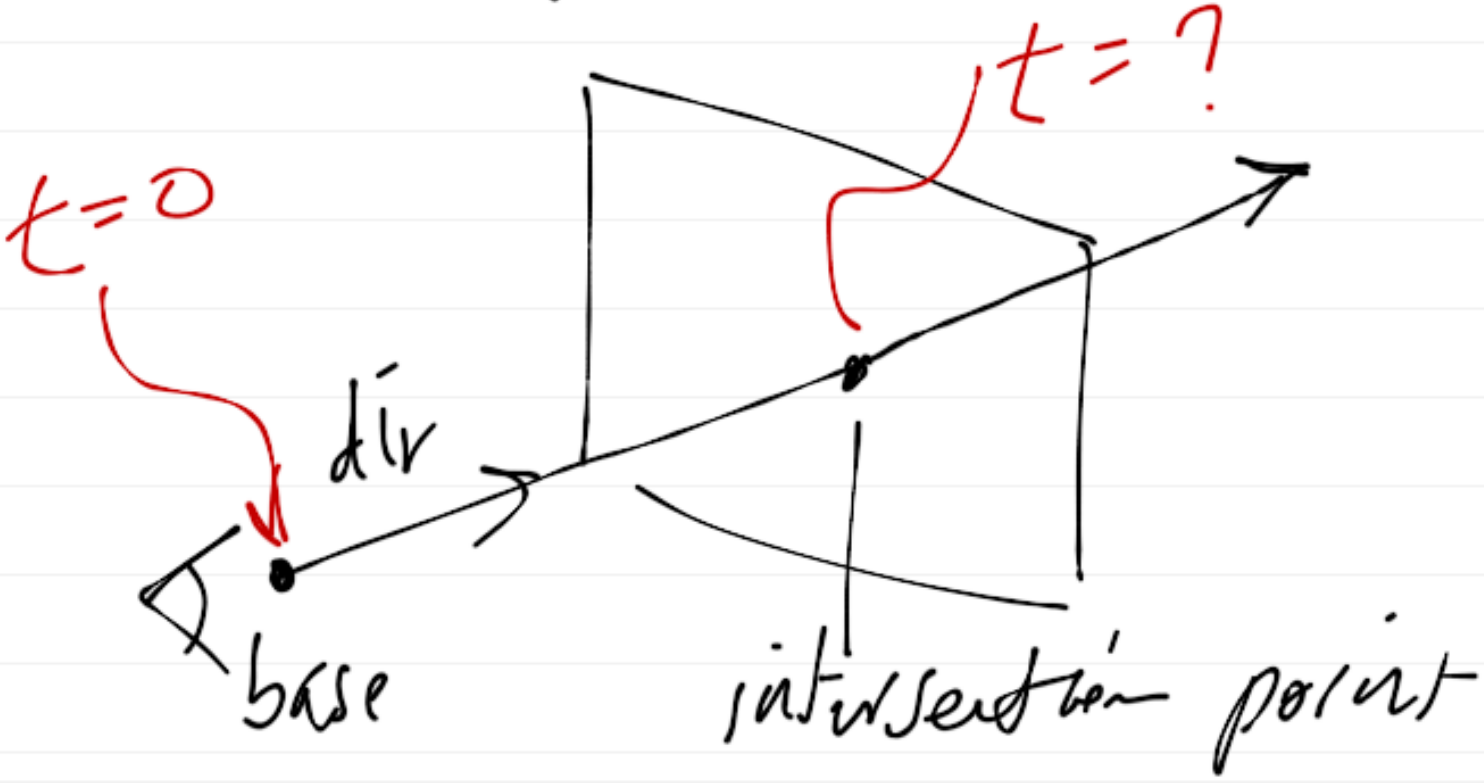
[material name e.g., blue

```
obj->mat =
    material_getbyname(
        mats, matname);
// consume whitespace until EOL
while ((c = fgetc(in)) != EOF
    && c != '\n');
```

```
// init default virtual
// functions
obj->pointer = object_point;
obj->hits = object_no_hit;
obj->ambient =
        material_getamb;
obj->diffuse =
        material_getdiff;
obj->specular =
        material_getspec;
list_add(objs, (void *) obj);
}
```

```
double
object_no_hit(object_t *obj,
              vec_t base,
              vec_t dir)
{
    fprintf(stderr,
      "Object %s failed to
      provide hits function\n",
      obj->name);
    return(-1.0);
}
```

- What is plane_hits supposed to do?
- return double t value along ray at ray-plane intersection point (see p. 72 in PDF notes)



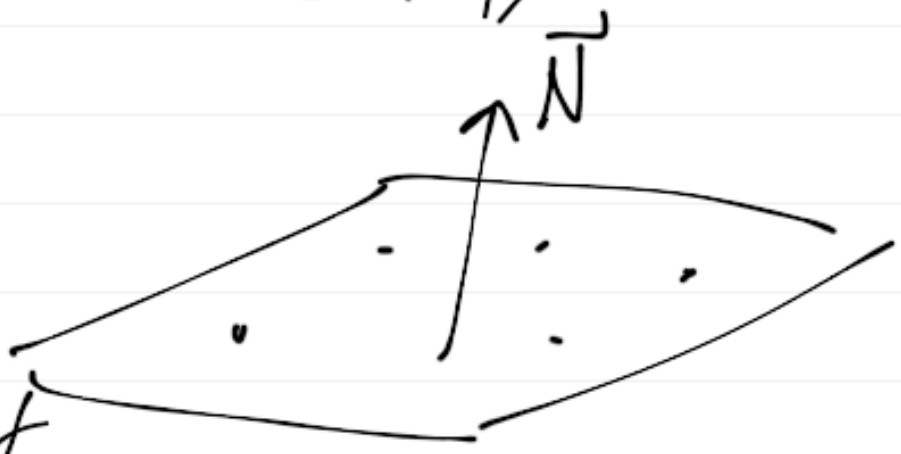
- remember

plane → ndotc = vec_dot(
plane → normal, plane → point)

- plane is defined by plane equation:

$$Ax + By + Cz + D = 0$$

for any point (x, y, z) on the plane



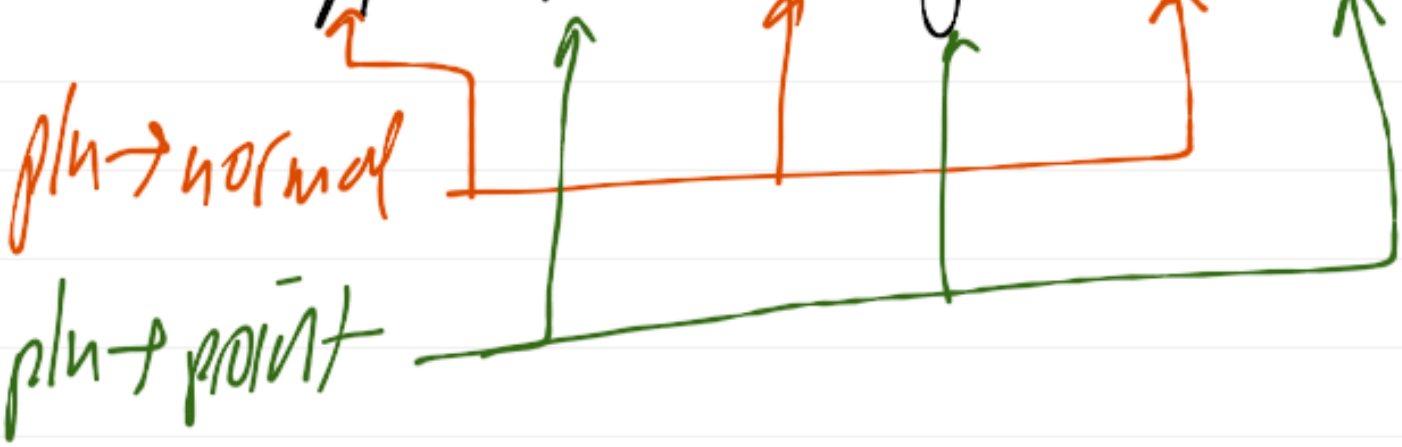
A, B, C, D just scalars but $\vec{N} = (A, B, C)$

- given plane normal = (A, B, C) (from file)

solve for D

- need known point on the plane, plane → point = (x, y, z) also from file

$$A*x + B*y + C*z$$



Rewrite in vector notation

$$\begin{pmatrix} A \\ B \\ C \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$ dot product.

$$Ax + By + Cz + D = 0$$

solve for D:

$$D = -(Ax + By + Cz)$$

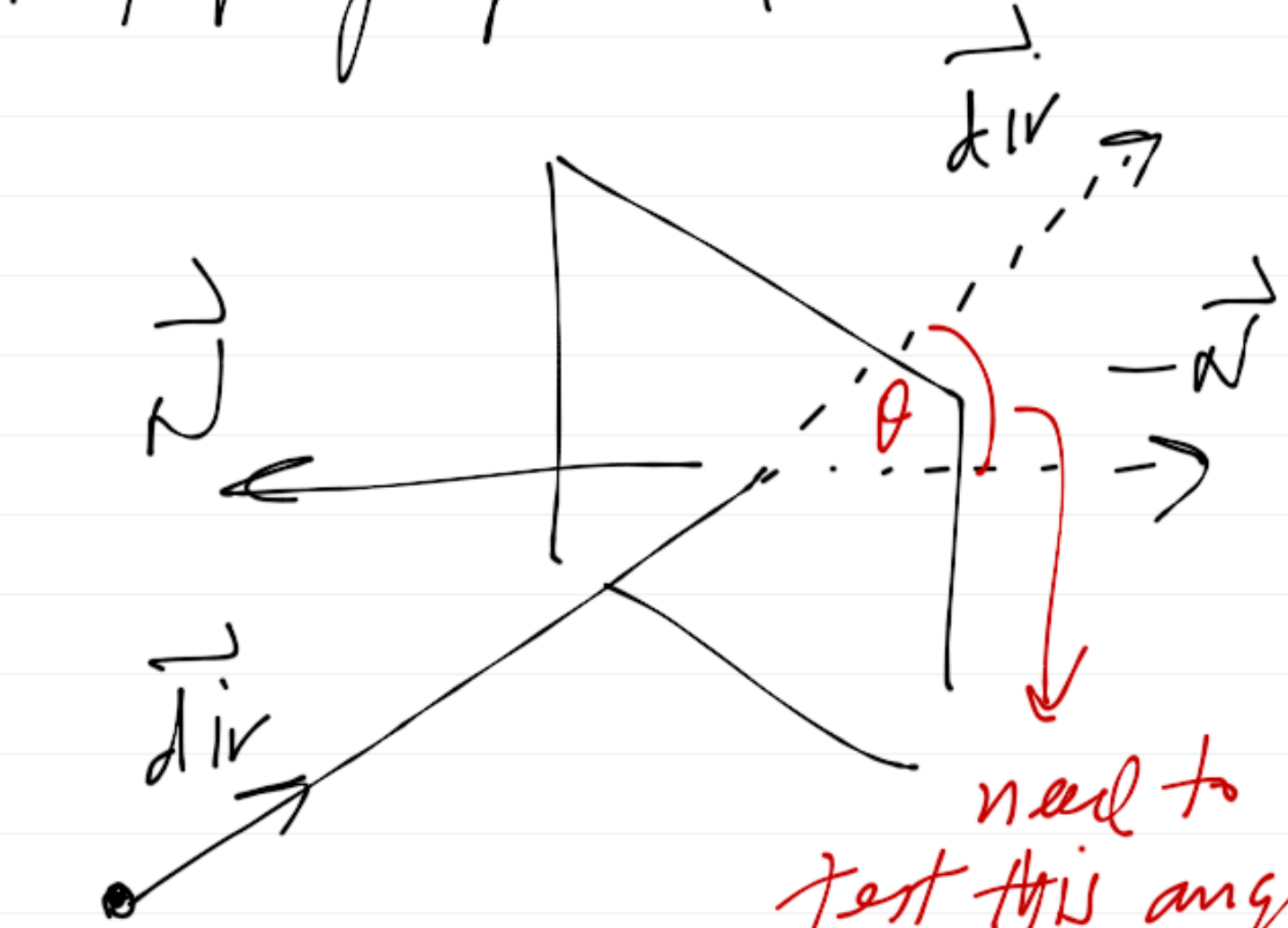
$$= -\left[\begin{pmatrix} A \\ B \\ C \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} \right]$$

$$= -\vec{N} \cdot P$$

plane plane

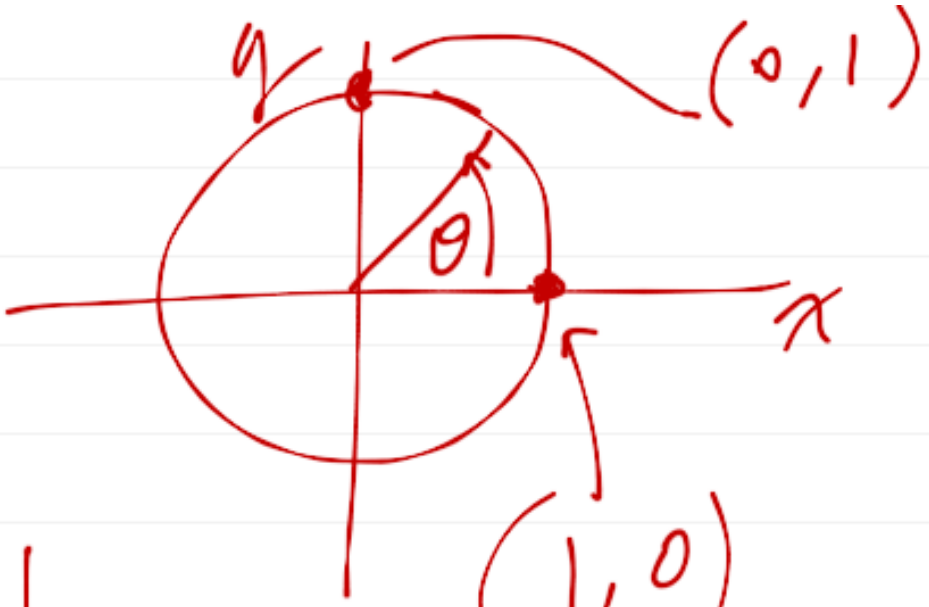
plane->ndotp = vec_dot(normal, point);

Now, ray-plane intersection:



θ: if θ = 90°, no intersection

$$\cos\theta = \frac{-\vec{N} \cdot \vec{dir}}{\|\vec{N}\| \, \|\vec{dir}\|}$$



$\cos(0) = 1$

$\cos(90) = 0$

$(1, 0)$

$\|$

$(\cos\theta, \sin\theta)$

if $-\vec{N} \cdot \vec{dir} == 0.0$

$\Rightarrow \theta = 90°$, no intersection

return -1

- plane_hits alg:

1. ndotd = N · dir
   (use vec_dot)

```
if (N.dir == 0.0)
    return(-1.0)
```

testing floating pt. equality, watch it!
use fabs()

② (N · dir is not 0 so we have intersection)

calculate distance from ray's base to plane:

- calculate $\vec{N} \cdot \overrightarrow{base}$,

ndotb = $\vec{N} \cdot \overrightarrow{base}$;

- subtract from D and scale by $1/(\vec{N} \cdot \vec{dir})$

$t = (D - (\vec{N} \cdot \overrightarrow{base})) / (\vec{N} \cdot \vec{dir})$

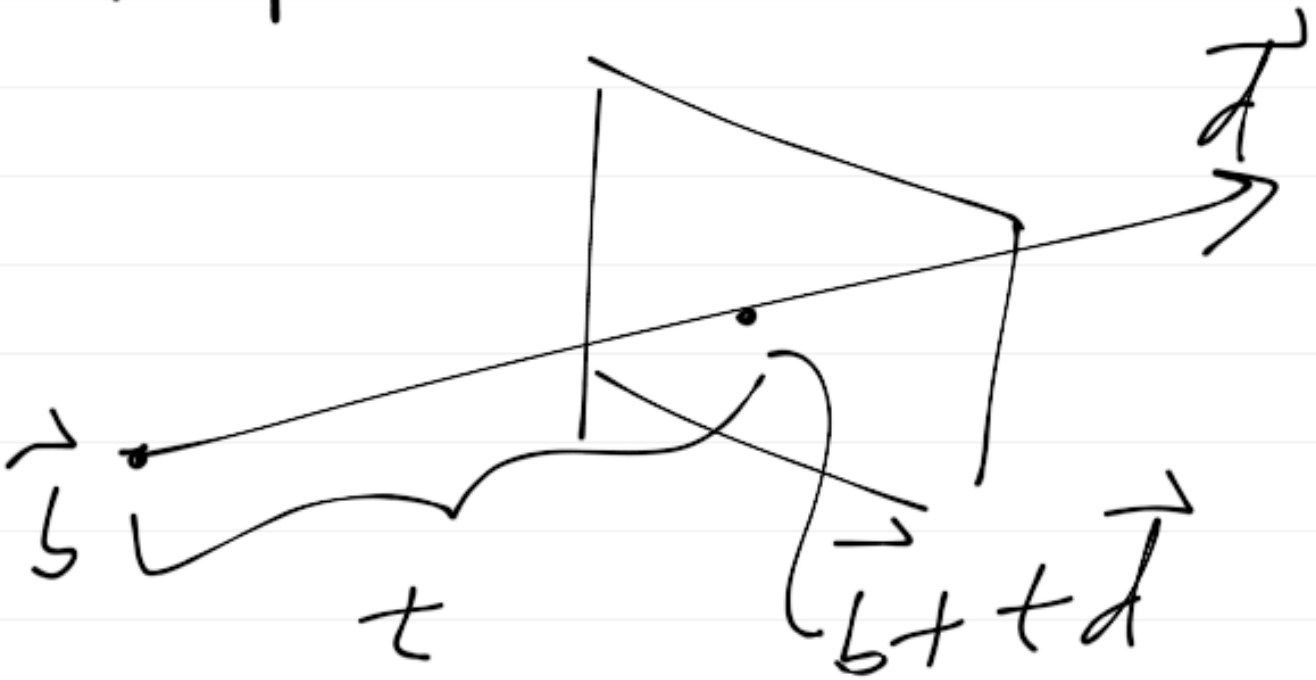
t = (pl->ndotg - ndotb) / ndotd;

③ if $t \leq 0$, intersection is behind base point

```
if (t <= 0) return(-1)
```

④ calculate hit point

hitpoint = base + t * dir



```
vec_scale(t, dir, obj->last_hit)
vec_sum(obj->last_hit,
        base,
        obj->last_hit)

// if z > 0, hit point
// in front of image plane,
// invalid
  if (obj->last_hit[2] > 0.0)
      return (-1.0);
  return (t);
}
```

Numerical example:

plane: $\vec{N} = 0\ 0\ 1$

$p = 0\ 0\ -7$

ray: $pos = 4\ 3\ 5$

$dir = 0\ 0\ -1$

① normalize plane dir

Vec_Unit(dir, dir)

$$\sqrt{0^2+0^2+1^2} = \sqrt{1^2} = 1$$

$$dir = \frac{1}{1}\begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix}$$

② $n \cdot d = \begin{pmatrix}0\\0\\1\end{pmatrix} \cdot \begin{pmatrix}0\\2\\-1\end{pmatrix} = \sum \begin{pmatrix}0\\0\\-1\end{pmatrix} = -1$

③ $n \cdot b = \begin{pmatrix}0\\2\\1\end{pmatrix} \cdot \begin{pmatrix}4\\3\\5\end{pmatrix} = \sum \begin{pmatrix}0\\0\\5\end{pmatrix} = 5$

④ $\dfrac{n \cdot g - n \cdot b}{n \cdot d} = \dfrac{-7-5}{-1} = 12$

(because $n \cdot g = \sum \begin{pmatrix}0\\0\\1\end{pmatrix}\begin{pmatrix}4\\3\\-7\end{pmatrix} = -7$

⑤ hit point : base + $t\vec{d}$

$\begin{pmatrix}4\\3\\5\end{pmatrix} + 12\begin{pmatrix}0\\0\\-1\end{pmatrix} = \begin{pmatrix}4\\3\\5-12\end{pmatrix} = \begin{pmatrix}4\\3\\-7\end{pmatrix}$