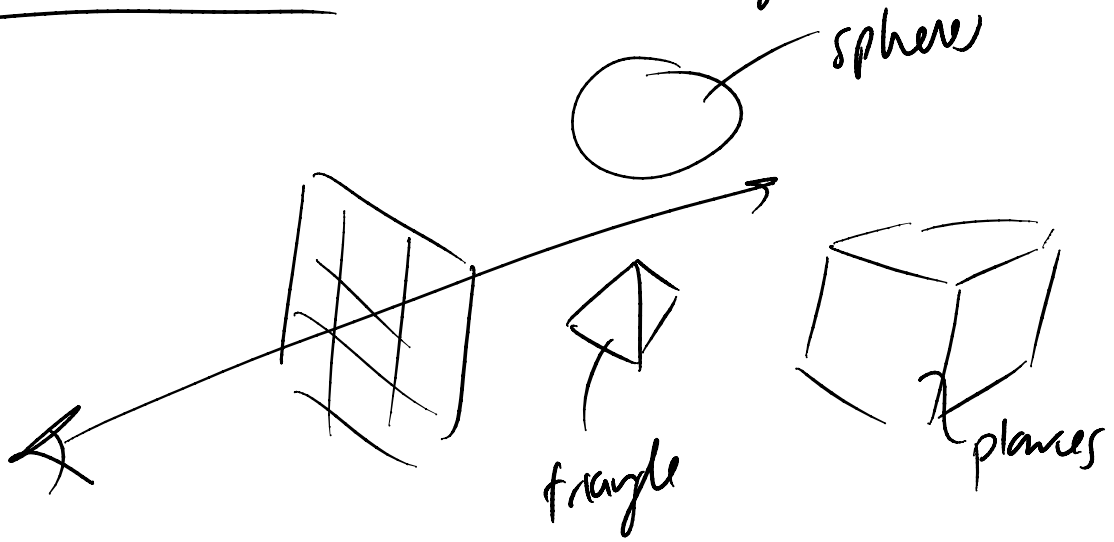
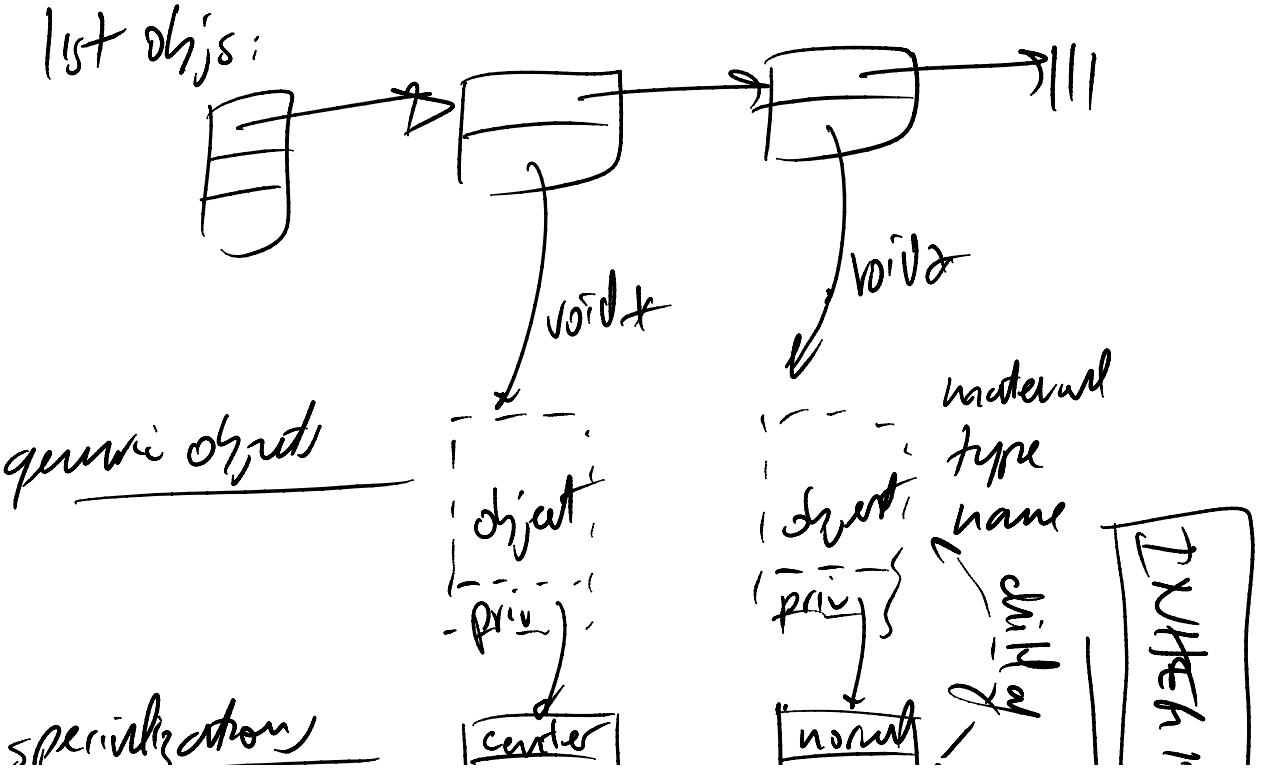
OOP (lab4)

Wednesday, September 08, 2010
8:59 AM

Object-oriental programming



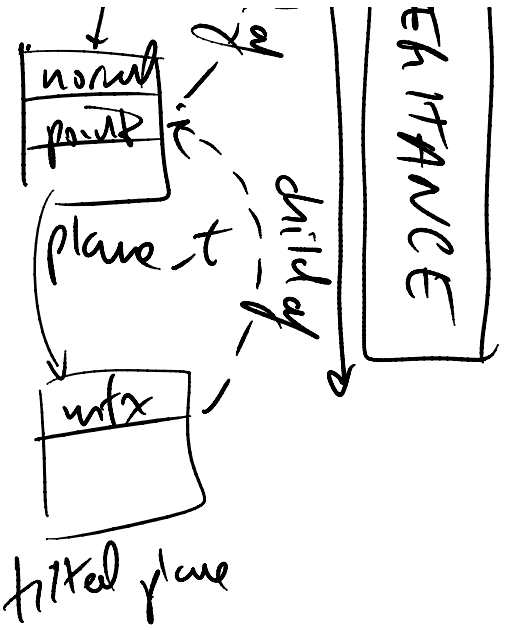
put these in a list



specialization

radius

sphere



how we want to use this (i.e., in lab 4)

for each object obj in objs list {

obj → print()

}

POLYMORPHISM

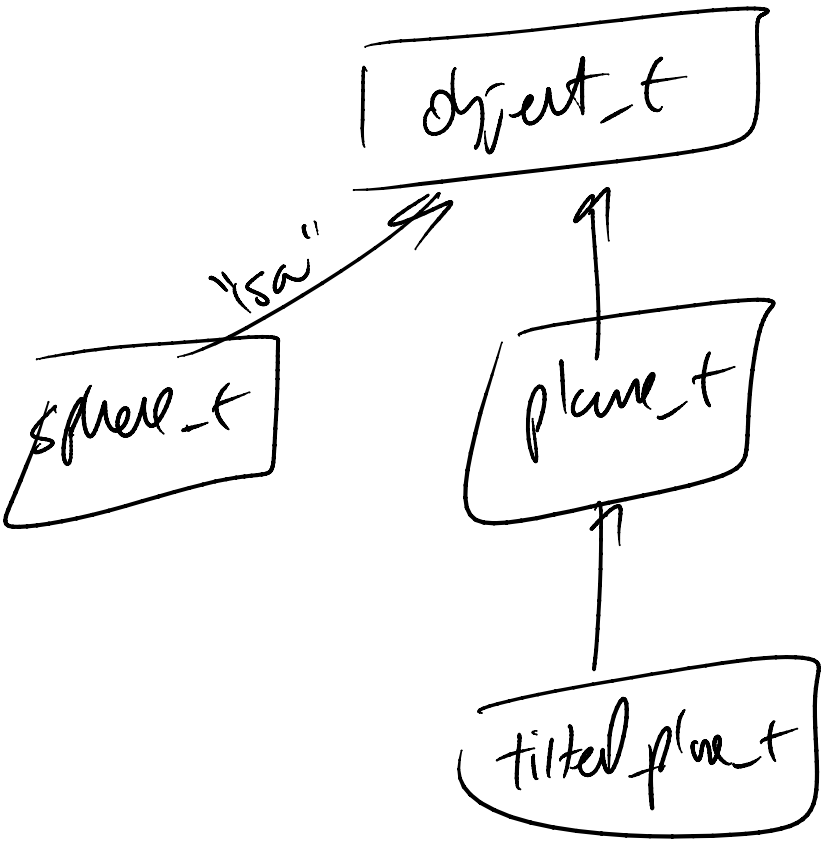
generic obj

only "knows" about type & name

function pointer to specialized object's print() function

being able to morph into many (poly) things

in notes:


object_t
"isa"
sphere_t
plane_t
tilted_plane_t

Data file

```
plane floor
{
    material gray        <-- stored in object_t
    normal   0 1 0   } stored in plane_t
    point    0 -.1 0 }
}
```

```
#define OBJ_COOKIE 12345678
typedef struct object_type
{
    int   cookie;
    char  type[NAME_LEN]; //e.g., plane, sphere,...
    char  name[NAME_LEN]; //e.g., wall, floor

    material_t *mat;
```

fptr { void (*printer)(struct object_type *, FILE *);
⋮
void *priv; // ptr to specific "class" (obj.) (e.g. sphere_t, plane_t, etc.)
⋮
} object_t;

function pointer (p. 58)

```
int adder(int a, int b)
{
    return(a+b);
}

int main()
{
    int sum;
    int (*fptr)(int, int);

    fptr = adder;
    sum = adder(3,4);
    sum = (*fptr)(3,4);
}
```

} save this way — usg ptr. to fct.

```
    printf("sum = %d\n", sum);
    return (0);
}
```

```
typedef struct plane_type {
    vec_t  normal;
    vec_t  point;
    double ndotq;
    void  *priv; // for subclasses
}
```

// implicitly plane_t "gets" with, that object_t has (type, name, material)

in C++ it would be

```
class object_t
{
    type -
    we -
    material
}

class plane_t : public object_t
{
    normal
```

inheritance from

point

object_t

}

- to print complete object contents:

for each obj:

obj->print() void (*printer)(object_t *, FILE*)

d

function pointer to specific object's print(),

so really we call plane->print()

plane_print(object_t *obj; FILE *out)

{

pointer to plane's parent, get parent to print itself first

object_print(obj, out);

plane_t *pln = (plane_t *)obj->privi

(now we have access to plane data struct

//print out stuff in plane_t

(ex. normal, point)