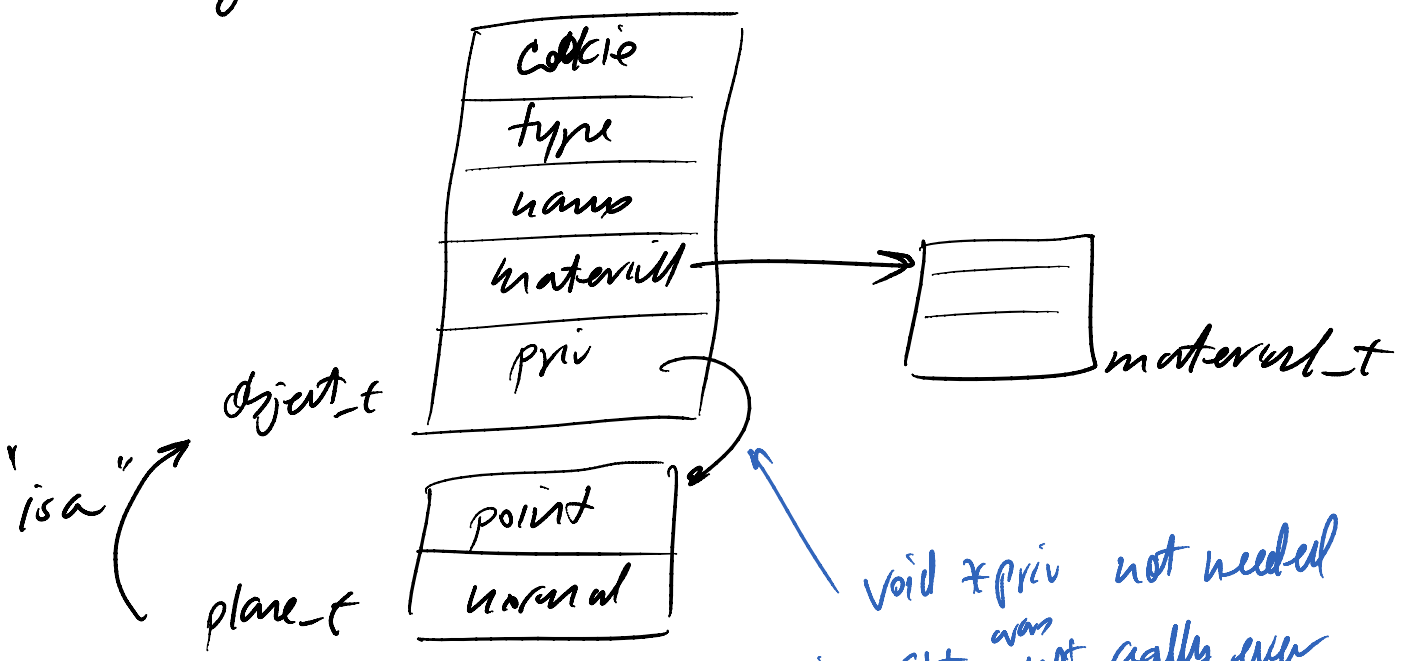
C++ inheritance and polymorphism: object_t (lab11)

Wednesday, October 20, 2010
8:59 AM

- previously, we had:

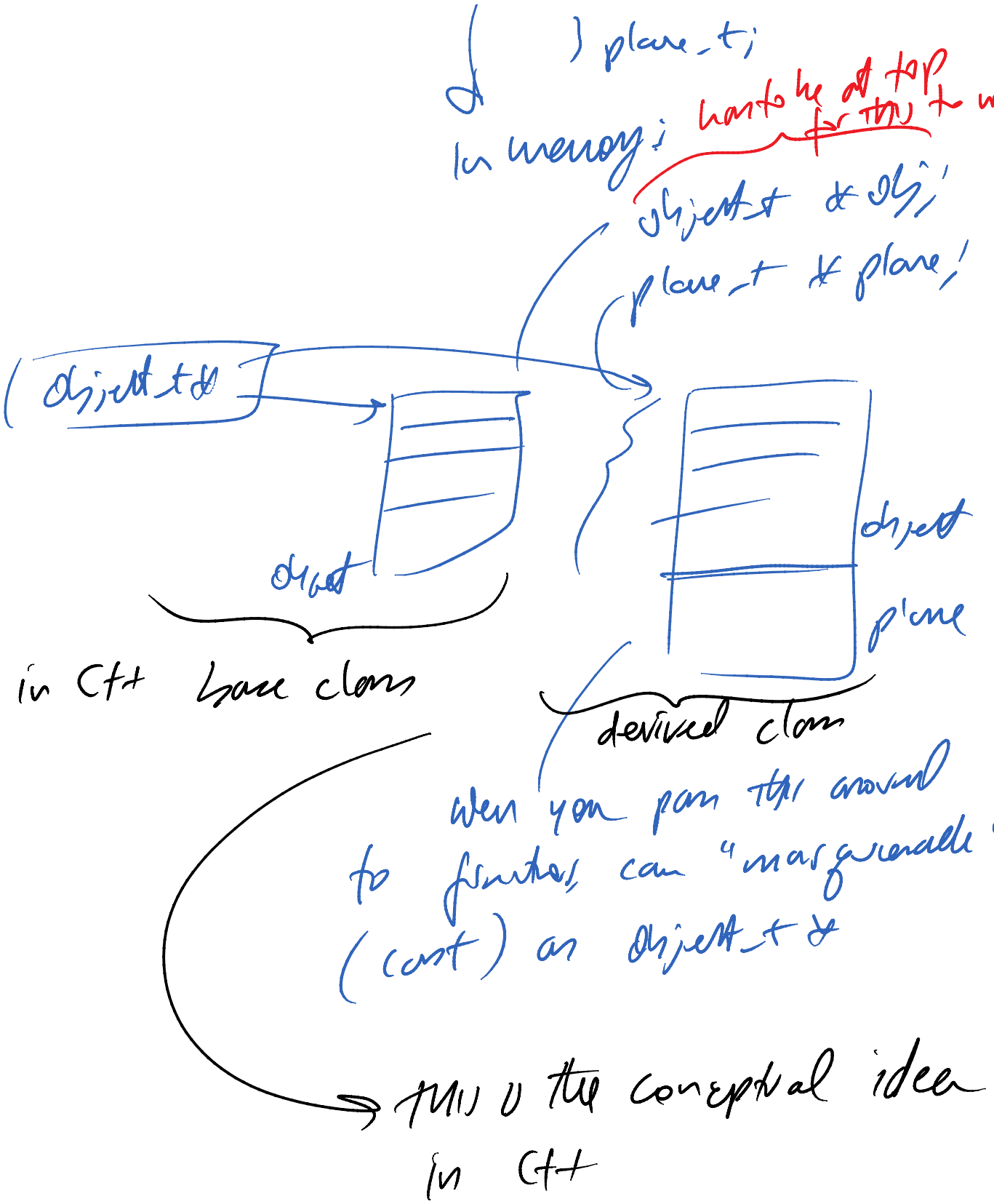


void *priv not needed in C++, was not really ever needed in C either

```
typedef struct {
    int collide;
    char *name;
    material_t* mat;
} object_t;

typedef struct {
    object_t *parent;
    int point;
    int normal;
} place_t;
```

child points to parent.


) plane_t;
in memory:
want to be at top for this to work
Object *obj;
plane_t *plane;
Object
Object
plane
in C++ base class
derived class
when you pass this around to functions, can "masquerade" (cast) as Object_t *
this is the conceptual idea in C++

Wednesday, October 20, 2010
9:12 AM

```
#ifndef OBJECT_H
#define OBJECT_H

#define OBJ_COOKIE 12345678

class object_t
{
    public:
    // constructor
    // copy constructor
    // destructor
    // assignment operator
  friend std::ostream& operator<<(std::ostream& s, const object_t& rhs)
    { return rhs.put(s); }
  // similarly for std::istream& operator>>(...)
  // define virtual function get( )

  virtual std::ostream& put(std::ostream& s) const;
  virtual double hits(const vec_t&, const vec_t&);
  // data members
  protected:
    int cookie;
    std::string type;
```

w/out this, compiler says something like "put function ignores const"

overloaded by plane_t

I want derived classes to overload this — no function pointer needed

says we can overload

means that these data members are inherited by derived classes (if private, wouldn't be inherited)

```
    std::string type;   (if private wouldn't get inherited)
    std::string name;
    std::string material;  // just a string, not a pointer
};                         (could use a pointer, but using string as an index)
```

Note:

- if you want to declare a function in this special way, e.g.,

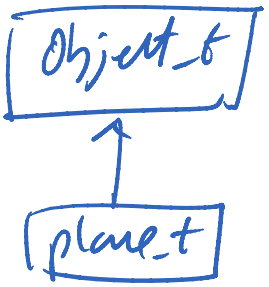
  virtual double hits(...) = 0;

  at least one

  this says a) this function MUST be overloaded by derived class

  b) object_T is now a pure virtual object meaning it cannot be instantiated (can't declare object_t obj;)

Wednesday, October 20, 2010
9:32 AM




```
#ifndef PLANET_H
#define PLANET_H

class planet_t : public object_t
{
    public:
    // constructor (only sets values for own data
                    members)
    planet_t(std::string token) : \
            object_t(token), \
            mass(0.0) \
            { };
    // copy constructor (don't forget to set
                    base class to copy itself)

    // destructor (default ok)

    // assignment op.
```

derived from this parent

// call object_t constructor

```
// friends
std::ostream& put(std::ostream& s) const;
std::istream& get(std::istream& s);

double hit(...)

private:
vec_t normal;
vec_t point;
double d0fg;
};
```

Keep a unit normal vec_t also "has a" type, name, color, material

- in object.cpp:

```
std::ostream& object_t::put(std::ostream& s) const
{
    // print out type, name, '{', material

}
```

- in plane.cpp:

```
std::ostream& plane_t::put(std::ostream& s) const
{
  object_t::put(s);  // call parent's put
  // print out normal, point, }

}

double plane_t::hit(const vec_t& p1, const vec_t& dir)
{
}
```

In main.cpp:

```
list_t mats, objs;
object_t *obj;
...
if (token == "material")
{
    ...
    //read material into list_t mats list
}
if (token == "plane")
{
    std::cin >> (obj = new plane_t(token))
    objs.add((void*)obj);
}
```