程序设计实习

郭炜　微博 http://weibo.com/guoweiofpku
http://blog.sina.com.cn/u/3266490431
刘家瑛 微博 http://weibo.com/pkuliujiaying

使用多态的游戏程序实例(P250)

游戏《魔法门之英雄无敌》

游戏中有很多种怪物，每种怪物都有一个类与之对应，
每个怪物就是一个对象。

类：CSoldier

类：CDragon

类CPhonex

类：CAngel

游戏《魔法门之英雄无敌》

怪物能够互相攻击，攻击敌人和被攻击时都有相应的动作，动作是通过对象的成员函数实现的。

游戏《魔法门之英雄无敌》

游戏版本升级时，要增加新的怪物——雷鸟。
如何编程才能使升级时的代码改动和增加量较小？

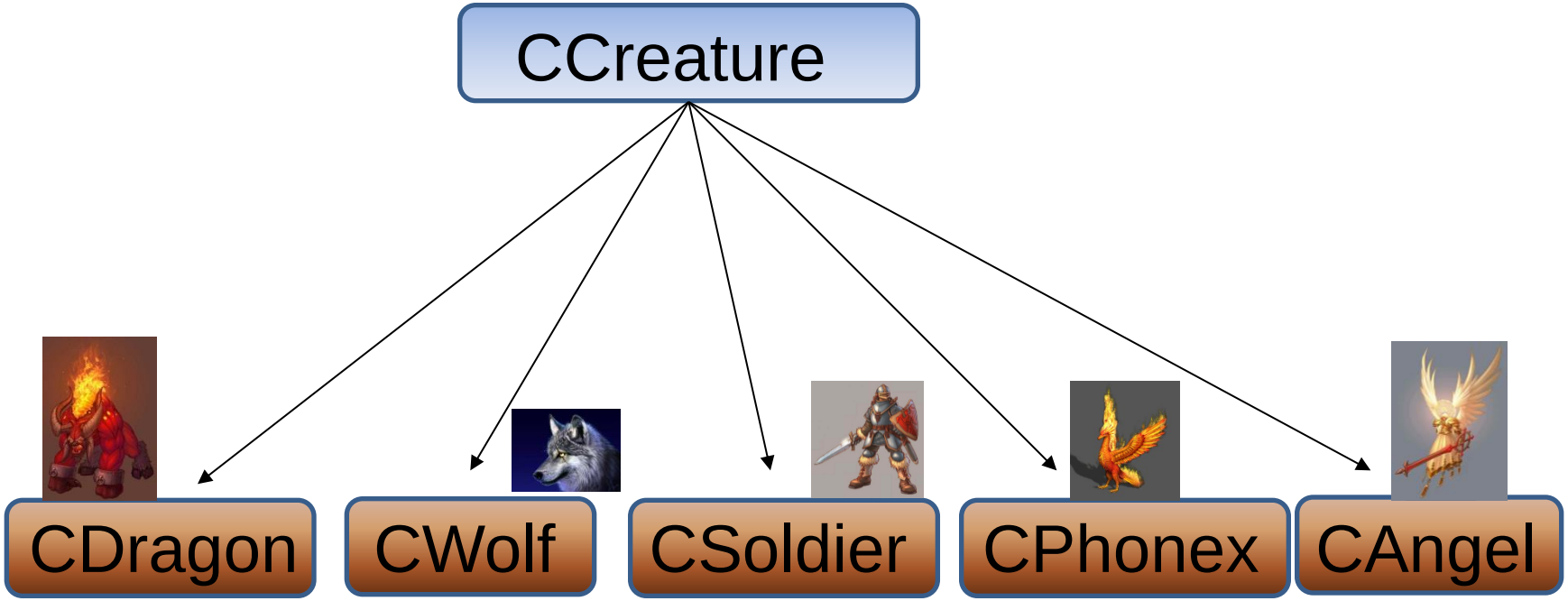
新增类：CThunderBird

基本思路：

- 为每个怪物类编写 Attack、FightBack和 Hurted成员函数。
- Attact函数表现攻击动作，攻击某个怪物，并调用被攻击怪物的Hurted函数，以减少被攻击怪物的生命值，同时也调用被攻击怪物的 FightBack成员函数，遭受被攻击怪物反击。
- Hurted函数减少自身生命值，并表现受伤动作。
- FightBack成员函数表现反击动作，并调用被反击对象的Hurted成员函数，使被反击对象受伤。

基本思路：

设置基类 CCreature，并且使CDragon，CWolf等其他类都从CCreature派生而来。



非多态的实现方法

```
class class CCreature {
  protected:          int nPower ; //代表攻击力
                      int nLifeValue ; //代表生命值
};
class CDragon:public CCreature  {
   public:
          void Attack(CWolf * pWolf) {
                      . . . 表现攻击动作的代码
                      pWolf->Hurted( nPower);
                      pWolf->FightBack( this);
          }
          void Attack( CGhost * pGhost) {
                      . . . 表现攻击动作的代码
                      pGhost->Hurted( nPower);
                      pGohst->FightBack( this);
          }
```

非多态的实现方法

```
        void Hurted ( int nPower) {
                . . . . 表现受伤动作的代码
                nLifeValue -= nPower;
        }
        void FightBack( CWolf * pWolf) {
                . . . . 表现反击动作的代码
                pWolf ->Hurted( nPower / 2);
        }
        void FightBack( CGhost * pGhost) {
                . . . . 表现反击动作的代码
                pGhost->Hurted( nPower / 2 );
        }
}
```

➢有n种怪物，CDragon 类中就会有n个 Attack 成员函数，以及 n个FightBack 成员函数。对于其他类也如此。

非多态的实现方法的缺点

- 如果游戏版本升级，增加了新的怪物雷鸟 CThunderBird，则程序改动较大。

- 所有的类都需要增加两个成员函数：

```
void Attack( CThunderBird * pThunderBird) ;
void FightBack( CThunderBird * pThunderBird)
```

- 在怪物种类多的时候，工作量较大有木有！！！

多态的实现方法

```
//基类 CCreature：
class CCreature  {
        protected :
                int m_nLifeValue, m_nPower;
        public:
          virtual void Attack( CCreature * pCreature) {}
          virtual void  Hurted( int nPower) { }
          virtual void FightBack( CCreature * pCreature) { }
};
```

- 基类只有一个 Attack 成员函数；也只有一个 FightBack成员函数；所有CCreature 的派生类也是这样。

多态的实现方法

```
//派生类    CDragon:

class CDragon : public CCreature {
        public:
         virtual void Attack( CCreature * pCreature);
         virtual void Hurted( int nPower);
         virtual void FightBack( CCreature * pCreature);
};
```

多态的实现方法

```
void CDragon::Attack(CCreature * p)
{       …表现攻击动作的代码
        p->Hurted(m_nPower);  //多态
        p->FightBack(this);  //多态
}
void CDragon::Hurted( int nPower)
{       …表现受伤动作的代码
        m_nLifeValue -= nPower;
}
void CDragon::FightBack(CCreature * p)
{       …表现反击动作的代码
        p->Hurted(m_nPower/2); //多态
}
```

多态实现方法的优势

- 如果游戏版本升级，增加了新的怪物雷鸟 CThunderBird……

只需要编写新类CThunderBird，不需要在已有的类里专门为新怪物增加：

```
void Attack( CThunderBird * pThunderBird) ;
void FightBack( CThunderBird * pThunderBird) ;
```

成员函数，已有的类可以原封不动，没压力啊！！！

原理

```
CDragon Dragon;  CWolf  Wolf;  CGhost Ghost;
CThunderBird Bird;
Dragon.Attack( & Wolf);        //(1)
Dragon.Attack( & Ghost);       //(2)
Dragon.Attack( & Bird);        //(3)
```

- 根据多态的规则，上面的(1)，(2)，(3)进入到CDragon::Attack函数后，能分别调用：

  CWolf::Hurted

  CGhost::Hurted

  CBird::Hurted

```
void CDragon::Attack(CCreature * p)
{
        p->Hurted(m_nPower);  //多态
        p->FightBack(this);  //多态
}
```