Atitit 面向对象弊端与问题 **[坏处](http://blog.csdn.net/fishmai/article/details/52079766) 缺点**

[1.1. 状态fsm，反模块化，又是反并行的，这是它的非常固有的特征， 1](#_Toc866)

[1.2. 现时的OOP编程有可能不缓存友好（cache friendly）， 1](#_Toc32226)

[1.3. 面向对象是一个很重量级的方法。你得设计类的继承关系。 3](#_Toc14815)

[1.4.](#_Toc31583) **[继承的代价](#_Toc31583)**[庞大的继承体系来获得代码的可重用性 3](#_Toc31583)

[1.5. ，毕竟很难一开始就做好规划，所谓的is-a，has-a 3](#_Toc11943)

## 状态fsm，反模块化，又是反并行的，这是它的非常固有的特征，

## 现时的OOP编程有可能不缓存友好（cache friendly），

**过度封装**  
  
使用OOP时，会把一些复杂的问题分拆抽象成较简单的独立对象，通过对象的互相调用去实现方案。但是，由于对象包含自己封装的数据，一个问题的数据集会被分散在不同的内存区域。互相调用时很可能会出现数据的cache miss的情况。  
  
**2. 多态**  
  
在C++的一般的多态实现中，会使用到虚函数表。虚函数表是通过加入一次间接层来实现动态派送。但在调用的时候需要读取虚函数表，增加cache miss的可能性。基本上要支持动态派送，无论用虚函数表、函数指针都会形成这个问题，但如果类的数目极多，把函数指针如果和数据放在一起有时候可放缓问题。

1. **数据布局**  
     
   虽然OOP本身并无限制数据的布局方式，但基本上绝大部分OOP语言都是把成员变量连续包裹在一段内存中。甚至使用C去编程的时候，也通常会使用到OOP或Object-based的思考方式，把一些相关的数据放置于一个struct之内：

即使不使用多态，我们几乎不加思索地会使用这种数据布局方式。我们通常会以为，由于各个成员变量都紧凑地放置在一起，这种数据布局通常对缓存友好。然而，实际上，我们需要考虑数据的存取模式（access pattern）。  
  
在OOP中，通过封装，一个类的各种功能会被实现为多个成员函数，而每个成员函数实际上可能只会存取少量的成员变量。这可能形式非常严重的问题，例如：

for (Particle\* p = begin; p != end; ++p)

p->position += p->velocity \* dt; // 或 p->SimulateMotion(dt);

在这种模式下，实阶上只存取了两个成员变量，但其他成员变量也会载入缓存造成浪费。当然，如果在迭代的时候能存取尽量多的成员变量，这个问题可能并不存在，但实际上是很困难的。

如果采用传统的OOP编程范式及实现方式，数据布局的问题几乎没有解决方案。所以在[1]里，作者提出，在某些情况下，应该放弃OOP方式，以数据的存取及布局为编程的考虑重中，称作面向数据编程（data-oriented programming, DOP）。

## 面向对象是一个很重量级的方法。你得设计类的继承关系。

得写类的声明，得实现不同的虚方法；实现时要注意是否要调用父类的方法。调试的时候经常搞不清楚到底执行哪个类的虚方法了。而用一大堆switch case，语句执行顺序一目了然。

可能还是觉得代码层级过多？有很多办法可以整理得更干净。用了面向对象，省了switch case，却要写更多的virtual function的声明和实现

## ****继承的代价****庞大的继承体系来获得代码的可重用性

。这并不是一个新颖的话题，自面向对象编程诞生之日起就饱受争议，我们经常要忍受着愈加繁杂和庞大的继承体系来获得代码的可重用性，而且随着继承层次的增加，代码的复杂性会加速增长，随之而来的bug也会越来越难以发现

面向对象之所以被很多人诟病的原因之一，是在框架和库的层次（如果这两个一起开发的话），他会形成巨型的继承树，一旦这颗树的设计出现问题，做细节的微调还可以，比如功能函数的上推下拉，局部重构等等，但是如果做大的调整很容易造成重大兼容性问题。

## ，毕竟很难一开始就做好规划，所谓的is-a，has-a

特别是java，c++这种静态强类型系统，如果编码使用自顶向下的继承，代码的灵活性非常差，毕竟很难一开始就做好规划，所谓的is-a，has-a，在现实世界里碰到特例会死的很惨。

面向对象之所以被很多人诟病的原因之二，是在写功能的层次，很多程序员都试图做出更大更深层次的抽象，不停的做抽象类和接口，在代码中生成大量的粘合层，而这些内容应该放到框架和库的层次去做，而不是在做功能开发时为了未来可能有的功能埋下大量的代码接口而忽略了开发功能的本质工

参考资料

浅谈面向对象的坏处 - fishmai的专栏 - 博客频道 - CSDN.NET.html

面向对象编程已死？\_知识库\_博客园.html

作者:: 绰号:老哇的爪子 （ 全名：：Attilax Akbar Al Rapanui 阿提拉克斯 阿克巴 阿尔 拉帕努伊 ）

汉字名：艾提拉（艾龙），   EMAIL:1466519819@qq.com

转载请注明来源： http://blog.csdn.net/attilax

Atiend