Hide Implementation Classes

eryar@163.com

摘要：很多程序员都用过 private 来隐藏函数和成员变量，实际上有些类也是可以被隐藏起来的。本文是对《API Design for C++》中 2.2.5 的翻译，若不不当之处，欢迎指出。

关键字：API Design for C++, Hide Classes

除了可以隐藏类的变量与方法之外，还可以隐藏任意单纯实现细节的类。很多程序员都用过隐藏方法和变量，但是好多也好像忘记了并不是所有的类都是公有的。实际上，有些类只在你的实现中需要，而不应该作为公开的接口暴露出来。

例如，考虑一个简单的烟花（Fireworks）类：一个接口可以用来指定烟花在屏幕上位置，控制烟花的颜色、速度和烟花颗粒（particle）的数量。明显地，这个 API 就需要对烟花的每个颗粒进行追踪，以便于在每帧更新颗粒的位置。这暗示着需要引入一个用来保存每个颗粒状态的类 FireParticle，但是这个 API 的用户并不需要访问这个类，它只在实现 API 时才需要。这样的类就可以设置为私有（private），即作为类 Fireworks 私有的部分。代码如下所示：

```
/// -*- tab-width: 4; c-basic-offset: 4; indent-tabs-mode: t -*-
///
/// \file   fireworks.h
/// \author Martin Reddy
/// \brief  An illustration of using private classes.
///
/// Copyright (c) 2010, Martin Reddy. All rights reserved.
/// Distributed under the X11/MIT License. See LICENSE.txt.
/// See http://APIBook.com/ for the latest version.
///

#ifndef FIREWORKS_H
#define FIREWORKS_H

#include <vector>

namespace apibook {

///
/// A simple fireworks particle system, used to demonstrate
/// the use of private classes to hide implementation state.
///
class Fireworks
{
public:
    Fireworks();

    /// Set the (x, y) origin of the fireworks effect
    void SetOrigin(double x, double y);
```

```
    /// Set the RGB color (0..1) for each particle
    void SetColor(float r, float g, float b);
    /// Set the gravity acting on each particle (meters/sec)
    void SetGravity(float g);
    /// Set the speed of the particle simulation
    void SetSpeed(float s);
    /// Set the number of particles in the simulation
    void SetNumberOfParticles(int num);

    /// Start (or continue) the simulation
    void Start();
    /// Stop the simulation
    void Stop();
    /// Advance the simulation by dt seconds
    void NextFrame(float dt);

private:
    // FireParticle represents internal hidden state
    // (You could also forward declare this class and
    // only provide the definition in the .cpp file.)
    class FireParticle
    {
    public:
        double mX, mY;
        double mVelocityX, mVelocityY;
        double mAccelerationX, mAccelerationY;
        double mLifeTime;
    };

    double mOriginX, mOriginY;
    float mRed, mGreen, mBlue;
    float mGravity;
    float mSpeed;
    bool mIsActive;
    std::vector<FireParticle *> mParticles;
};

}

#endif
```

注意到在类 FireParticle 中我并没有使用 getter/setter。只要你愿意，当然也可以这样做。但是也不是非常必要这么做，因为公有的接口是不能访问这个类的。有些工程师也比较喜欢使用 struct 来代替 class，to reflect that the structure is a Plain Old Data(POD) type。

当然，你可能也想过隐藏类 FireParticle，即在头文件中也不出现。我将在下一节中讲到怎样来实现。