完了点编译，几乎不耗费时间，而 C++ 有时候编译需要十几秒，甚至在项目代码变多之后会达到一两分钟（视机器性能而定）。

第三，蓝图的节点函数名大多数和 C++ 版本的能够一一对应，学会蓝图编程以后，再改用 C++ 编写游戏逻辑，转换成本不会太高。

当然，蓝图也有它的缺点——在某种情况下代码易读性低。蓝图的节点占用视觉面积过大，所以一旦某个逻辑比较复杂，使用到的节点就可能会很多，会导致读代码的人非常痛苦。同时，蓝图节点间的连线如果过多，也会让人产生眼花缭乱的感觉，不利于维护。

这些缺点在小规模使用下可以通过一些代码规范来规避。为了规避这个问题，我们最好把代码分成一段一段的，或者用一些方法将可重复利用的代码封装起来。

由于大量的蓝图代码不利于维护，所以在大型项目的开发中，我们可能会完全使用 C++（或其他脚本语言）进行开发，或者使用小部分蓝图配合大部分的 C++（或其他脚本语言）进行开发。

但是在现阶段，或者说在刚接触 UE5 的阶段，蓝图还是我们入门学习的不二选择。

3.2 蓝图类

这一节，我们来正式了解蓝图。首先，我们要从“蓝图类”这个概念开始。

3.2.1 面向对象编程中的类和实例

在了解什么是蓝图类之前，我们先要了解一下“类”这个概念。“类”这个概念可不是 UE5 蓝图的专属，它是面向对象编程（OOP）的核心概念。

在面向对象编程中，有两个重要的概念，一个是“类”，一个是“对象”。类表示的是“种类”，比如有一个种类叫作动物，一个种类叫作哺乳动物，一个种类叫作人。类有几个特性：

首先，类最重要的功能在于封装。什么叫封装呢？简单来说，有一个类叫作动物，那么你可以给动物这个类添加各种功能，比如移动，比如呼吸，还可以给这个类添加各种属性，比如体重，颜色等。

其次，类之间可以有继承关系，一个类可以继承另一个类的所有公开特性。我们假定有关系：“人”属于“哺乳动物”，而“哺乳动物”又属于“动物”，如图 3.2 所示。那么：

- 如果规定了“动物”是能够移动的物体，给了它移动的功能。那么，你就可以不用再编写“哺乳动物”和“人”的移动功能。
- 只需要编写哺乳动物特有的其他功能或特性——比如它是恒温的且胎生。接下来，可以再让人继承于哺乳动物，继承它的所有公开特性和功能，人也会拥有恒温和胎生属性了。