# VTK-m Quick Start

## 3.1初始化(Initialize)

初始化VTK-m是可选的，建议允许VTK-m配置设备和logging

调用函数vtkm::cont::Initialize

Initializing VTK-m.



## 3.2读取文件

VTK-m有自己的简单的IO库，可读写VTK legacy格式(后缀.vtk)

读取文件使用函数vtkm::io::VTKDataSetReader



ReadDataSet方法返回数据在vtkm::cont::DataSet对象中

对象DataSet在第7章介绍。

VTK-m的IO库在第8章介绍。

## 3.3运行一个过滤器(filter)

过滤器：封装VTK-m算法的单元。VTK-m的过滤算法见第9章介绍。

介绍使用过滤器vtkm::filter::MeshQuality（定义在vtkm/filter/MeshQuality.h）。MeshQuality过滤器计算输入数据的每个单元，计算单元的网格质量，有不少指标。执行输入的DataSet，产生输出的DataSet



## 3.4渲染一个图像

第10章将介绍如何定义渲染图像。下面只执行简单的渲染：

渲染数据





第1步：设置一个渲染，创建一个scene。scene包括一些actors，代表一些待渲染的数据。上例中仅有一个DataSet需要渲染，因此仅创建一个actor，然后添加到scene(第1~5行代码)。

第2步：设置一个渲染，创建一个view。view包含之前提到的scene，一个mapper（描述如何渲染数据），一个canvas（包含图像缓存和其他的渲染上下文）。调用Paint产生图像。但是，VTK-m的渲染类执行的渲染是offscreen的，意思是结果不出现在显示器上。观察图像的最容易的方法是使用SaveAs方法保存一个图像文件。

## 3.5完整的示例代码

## 3.6编译配置

使用CMake编译VTK-m的代码是最容易的方式。

CMakeLists.txt to build a program using VTK-m

