

BIM引领着建设领域的第二次革命

BIM系统二次开发技术



清华大学土木工程系 胡振中 副教授

huzhenzhong@tsinghua.edu.cn

<http://www.huzhenzhong.net>

提纲

- 概述
- 面向对象编程
- BIM-Star二次开发及实例
- 结语

提纲

- 概述
- 面向对象编程
- BIM-Star二次开发及实例
- 结语

概述

- 越来越多的BIM软件提供二次开发的SDK (Software Develop Kit) 和 API (Application Programming Interface)
 - ✓ Revit/Navisworks/Civil 3D (Autodesk) — C#
 - ✓ MicroStation (Bentley), Catia (Dassault) — C++
 - ✓ BIM-Star (筑星) — C#
- **二次开发**：在现有的软件上进行功能的定制修改和扩展，是软件系统**开放性的**体现，可实现用户**个性化的**功能需求，而**不改变原有系统的内核**
 - ✓ 软件的开放性是软件性能的重要指标
 - ✓ 专业知识是二次开发的原动力和支撑力

概述

□ 例如，Revit二次开发可以实现的操作

访问Revit数据

- 访问BIM的几何数据和属性数据
- 创建、编辑和删除建筑模型构件，如楼板，墙等
- 导入外部数据，生成新构件和模型

创建Revit应用

- 集成其他应用程序到BIM平台
- 创建用户交互界面，创建自定义命令按钮
- 实现基于BIM的计算分析程序
- 各参与方的各种定制需求，等等

Revit二次开发的工具栏

石材形状选择区
(使用了Revit实体)

项目浏览器
此处调用了Revit
系统自带窗口

墙体拼装方案
(图形平台开发)

石材切割图
开发了基于信息模型的
几何计算模块

项目浏览器
此处调用了Revit
系统自带窗口

墙体拼装方案 (图形平台开发)

石材切割图

开发了基于信息模型的
几何计算模块

概述

□ 二次开发所需的知识

了解BIM软件二次开发的环境

- 大多数在Windows或ios平台
- 特定编程语言，如 C++、C#、Python等
- Windows平台下，一般是基于Microsoft .NET编程框架

需要掌握

- 面向对象程序设计
- 特定编程语言的用法
- 集成开发环境（如Microsoft Visual Studio）的用法
- BIM软件发布的API提供的系统框架功能

提纲

- 概述
- 面向对象编程
- BIM-Star二次开发及实例
- 结语

面向对象编程

□ 程序设计方法

程序和程序设计方法的本质

- 程序 = 数据结构 + 算法
- 程序设计 = 数据结构 + 算法 + 方法 + 工具

程序设计方法反映对数据结构和算法进行组织的特征

- 面向数据：数据结构
- 面向功能：算法
- 面向对象：数据结构 + 算法

- ✓ 是80年代兴起的先进的程序设计方法
- ✓ 基本概念由日常生活导入

依据程序组织的侧重点进行分类

日常生活

人们周围有很多对象

人/张三、小汽车、建筑物、对话等

人们根据对象来思考

映入眼帘看作对象，不看作颜色点阵

对象有一定的属性和行为

小汽车：大小、形状；会加速

对象有一定的类属

人/张三，李四

对象类属之间有一定继承性

小汽车:汽车 加速

对象相互作用实现功能

张三让李四立正，张三可将小汽车开走

对象相互作用不必涉及内部细节

人让小汽车加速

程序设计

对象

类

继承

消息

封装

面向对象编程

□ 面向对象程序设计的特征

- 面向对象程序设计的特征是使用对象、类、继承、消息、封装等基本概念来进行程序设计
- 以下三者缺一均不属于面向对象程序设计
 - ✓ 使用类和对象
 - ✓ 以类和对象作为程序的基本构件
 - ✓ 使用类之间的继承关系

面向对象编程

□ C#中类的定义和组成

```
访问修饰符 class 类名{  
    访问修饰符 字段类型 字段名;  
    访问修饰符 属性类型 属性名{  
        get{ ...}  
        set{ ...}  
    }  
    访问修饰符 返回值 方法名(参数列表){  
        ...// 方法的执行  
    }  
}
```

- C#中类的组成：**类 = 字段 + 属性 + 方法**

面向对象编程

□ 常用访问修饰符

- `public` : 公有访问。一个类或类成员标志为public时，其他类的任何实例都允许访问
- `private` : 私有访问。只能从该类的成员（即类的方法、字段和属性）来访问。类成员默认为private，如果希望它是公共的，需要明确标志为public
- `protected` : 受保护访问。可以由该类中所有其他成员，以及该类子类中的所有成员可以访问

面向对象编程

□ 对象的用法

初始化对象

如果没有构造函数，采用对象初始化方法

`new 类名(){设置属性的初始值};`

```
var mRec = new MonitorRecord() {  
    RecordId = 23,  
    MonitorValue = 0.618  
};
```

若有构造函数，采用构造函数初始化对象

```
var mRec =  
new MonitorRecord(23,0.618)
```

```
public class MonitorRecord {  
    public MonitorRecord(int id, double val)  
    {  
        RecordId = id;  
        MonitorTime = DateTime.Now;  
        MonitorValue = val;  
    }  
    public int RecordId { get; set; }  
    public DateTime MonitorTime { get; set; }  
    public int ElemId { get; set; }  
    public double MonitorValue { get; set; }  
  
    public string SomeMethod()  
    {  
        // 方法的执行  
    }  
}
```

面向对象编程

□ 对象的用法

对象引用

在创建对象时，必须用一个对象引用指向该对象

```
var mRec1 = new MonitorRecord();  
var mRec2 = new MonitorRecord();  
mRec1 = mRec2;
```

只能通过引用使用对象中的字段、属性、方法

```
var time = mRec1.MonitorTime;  
var str = mRec1.SomeMethod();
```

```
public class MonitorRecord {  
    public MonitorRecord(int id, double val)  
    {  
        RecordId = id;  
        MonitorTime = DateTime.Now;  
        MonitorValue = val;  
    }  
    public int RecordId { get; set; }  
    public DateTime MonitorTime { get; set; }  
    public int ElemId { get; set; }  
    public double MonitorValue { get; set; }  
  
    public string SomeMethod()  
    {  
        // 方法的执行  
    }  
}
```

面向对象编程

□ 数据封装

数据封装的概念

- 公有将对象的实现与对象的界面分离开来，使得外部只能通过对象的界面来利用对象提供的服务
- 对象的界面：对外公开成员
- 对象的实现：包括数据，内部操作，操作的实现

数据封装带来的好处

- 将改变程序带来的影响局部化，避免程序中出现Bug

面向对象编程

□ 数据封装

C#对封装的实现

- 在类中对成员的访问权限进行定义
- 将类中的成员用private或protected修饰来控制对其的访问
- 外部对private修饰的成员的访问：只能使用返回该成员的公共方法

```
private int _member;  
public int GetMember() {  
    return _member;  
}
```

面向对象编程

□ 数据封装

```
public class Line {  
    private Point2D point1;  
    private Point2D point2;  
  
    public Line(Point2D point1, Point2D point2){  
        this.point1 = point1;  
        this.point2 = point2;  
    }  
  
    public double GetDistance(){  
        var dx = point1.X - point2.X;  
        var dy = point1.Y - point2.Y;  
        return Math.Sqrt(dx * dx + dy * dy);  
    }  
}
```

private成员仅供内部的函数使用

外部无法修改它们，故不会在Line对象中引入错误

面向对象编程

□ 接口：实现多态的途径

接口定义

- 接口（interface）是一种约定，用来定义实现该接口的类应该具有的方法和属性
- 但本身不包含具体实现

接口的特点

- 实现接口的类必须包括接口的所有方法，并给出具体实现
- 接口可以像类一样有继承关系

面向对象编程

□ 接口：实现多态的途径

```
public class WaterRecord : IShowable {  
    public double ShowValue {get; set;}  
    public WaterRecord(double val) {  
        ShowValue = val; }  
    public void Show() {  
        MessageBox.Show("water :"  
            + ShowValue); }  
}  
  
public class TemperatureRecord : IShowable {  
    public double ShowValue {get; set;}  
    public TemperatureRecord(double val) {  
        ShowValue = val; }  
    public void Show() {  
        MessageBox.Show("temperature :"  
            + ShowValue); }  
}
```

```
public interface IShowable {  
    double ShowValue {get; set;}  
    void Show();  
}
```

```
public void Execute() {  
    var rec1 = new WaterRecord(0.6);  
    var rec2 =  
        new TemperatureRecord(27);  
  
    IShowable shower;  
  
    shower = rec1;  
    shower.Show();  
  
    shower = rec2;  
    shower.Show();  
}
```

输出？

面向对象编程

□ 继承

- 利用已有的类来生成新类，使新类不仅具有新定义的成员，而且还拥有已有的类的成员
- 已有的类被称为**基类**；新生成的类被称为**子类**
- 从一个基类派生称为单继承；从多个基类派生称为多继承。C#不支持类的多继承，但可以继承多个接口
- 继承带来的好处：可以利用现有程序，代码表达经济

思考：为什么C#设计成只允许继承一个基类，却允许继承多个接口？这样解决了C++中的什么弊端？

面向对象编程

□ 继承

```
public class MonitorRecord {  
    public int RecordId  
    { get; set; }  
    public int ElemId { get; set; }  
    public double MonitorValue  
    { get; set; }  
  
    public string SomeMethod()  
    {  
        // 方法的执行  
    }  
}
```

```
public class WaterRec : MonitorRecord {  
    public double WaterValue { get; set; }  
  
    public void ShowValue() {  
        MessageBox.Show(RecordId  
            + “ : WaterValue is” +  
            WaterValue);  
    }  
}
```

子类可以使用基类的属性

面向对象编程

□ 面向对象的好处

程序的结构自然、直观，易于理解

- 可以找到现实中的对应物及其作用机制

较易形成可复用的部件

- 每创建的一个新类，将有潜力在未来的软件开发过程中使用，从而可以提高开发速度和质量
- 未来的大部分程序将用标准化的、可交换的部件组合起来进行构筑

提纲

- 概述
- 面向对象编程
- BIM-Star二次开发及实例
- 结语

BIM-Star二次开发及实例

□ BIM-Star简介

- BIM-Star是基于清华大学4D-BIM课题组多年研究成果，由深圳筑星科技有限公司研发的具有自主知识产权的BIM软件产品，可进行包括设计、施工、运维在内的全生命期BIM软件开发
- 主要功能软件均以插件的形式挂接到BIM-STAR平台上
- BIM-STAR教育版目前提供了一些应用于施工阶段和运维阶段的基础BIM插件，同学们也可以根据自己的需求自主开发一些插件（第三次作业）

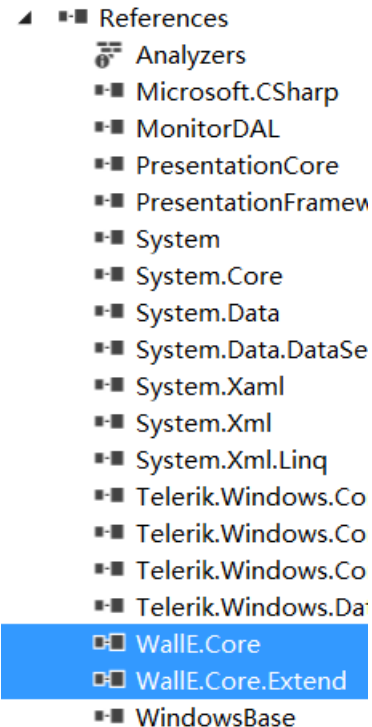
BIM-Star二次开发及实例

□ 让BIM-Star说Hello world

- is not easy

新建插件工程

- 创建一个文件夹存储所有的插件工程
- 用Visual Studio创建新项目，类型为
class library (.dll)
- 引用BIM-Star提供的**API**：
 Walle.Core
 Walle.Core.Extend

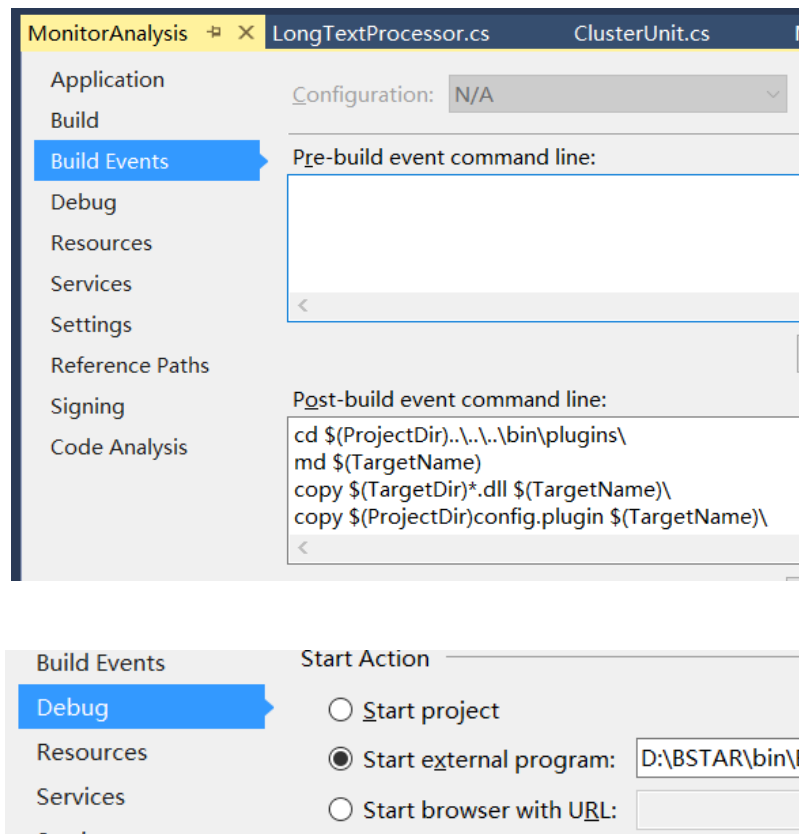


BIM-Star二次开发及实例

□ 让BIM-Star说Hello world

配置调试环境

- 进入项目属性页
- **build events**中设置编译后的文件拷贝至BIM-Star安装目录下，用于安装此插件
- **Debug**选项中设置调试时启动外部程序（即BIM-Star主程序的路径）
- 因为dll必须在主程序运行时进行调试



BIM-Star二次开发及实例

□ 让BIM-Star说Hello world

新建配置代码

- 创建两个配置文件
 config.plugin
 _xxxConfig.xml
- 以规定的格式描述了插件的信息
- 加载插件时读取

```
AnalysisConfig.xml  config.plugin  X
1  <?xml version="1.0" encoding="utf-8" ?>
2  <Plugin Key="489707D1-E6CD-4280-9C0C-A1B40E"
3      Description="监测数据的分析" LoadTime="0"
4      IsEnable="true">
5      <SqlFiles>
6          <SqlFile Name="sqls/MonitorRecordListBu
7          <SqlFile Name="sqls/MonitorRecordListBu
8
9      </SqlFiles>
10 </Plugin>
```

```
_MonitorAnalysisConfig.xml  X  config.plugin
1  <?xml version="1.0" encoding="utf-8" ?>
2  <Root>
3      <RibbonTab>Monitor</RibbonTab>
4      <TabIndex>0</TabIndex>
5  </Root>
```

BIM-Star二次开发及实例

□ 让BIM-Star说Hello world

建立插件类

- 创建插件类，并继承IPlugin接口
- `public class SayHelloPlugin : IPlugin`
- 实现IPlugin接口里的两个方法：加载和卸载
- `public void Install(IPluginInfo pluginInfo)`
- `public void Uninstall()`
- 所有的功能写在Install里面

BIM-Star二次开发及实例

□ 让BIM-Star说Hello world

建立命令对象、注册命令

- 1、声明一个命令对象，为插件类的成员属性
- `public RelayCommand SayHello;`
- 2、给命令变量赋值
- `SayHello = new RelayCommand(OnSayHello);`
- 其中OnSayHello是一个返回void且无参数的函数
- 3、向主程序注册此命令，并给它一个名字
- `M.CommandManager.Register(new
CommandInfo("Hello_SayHello", SayHello));`

BIM-Star二次开发及实例

□ 让BIM-Star说Hello world

定义菜单按钮

- 创建菜单描述代码SayHelloGroup.xml

```
SayHelloGroup.xml  + X
1 <?xml version="1.0" encoding="utf-8" ?>
2 <Group Header="打招呼">
3   <Button Text="说Hello" ButtonSize="Large" Click="Hello_SayHello"/>
4 </Group>
```

- 定义一个菜单组，名字是“打招呼”
- 定义一个Button
- 指定点击时执行命令的名称
- Click="Hello_SayHello"

BIM-Star二次开发及实例

□ 让BIM-Star说Hello world

写命令响应函数

- 创建OnSayHello函数作为命令响应
- 必须是一个返回void且无参数的函数

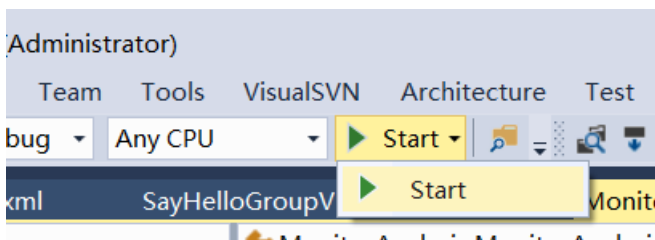
```
private void OnSayHello()
{
    MessageBox.Show("Hello, world!");
}
```


BIM-Star二次开发及实例

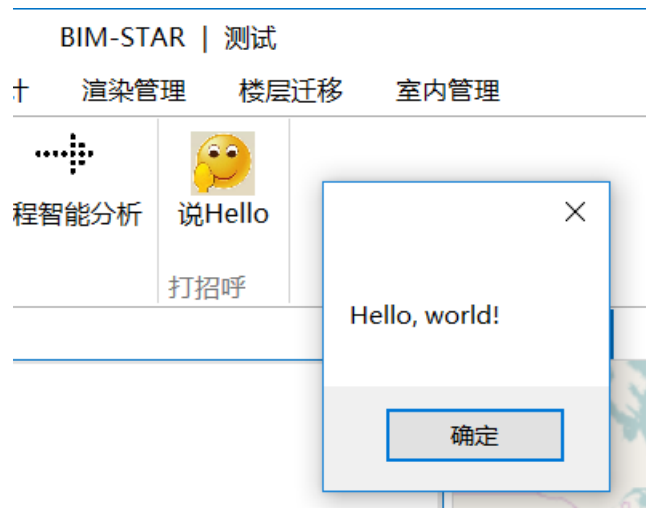
□ 让BIM-Star说Hello world

编译、运行

- 使用Start按钮开始编译调试



- 开始运行主程序
- 加载插件后即可运行 →



BIM-Star二次开发及实例

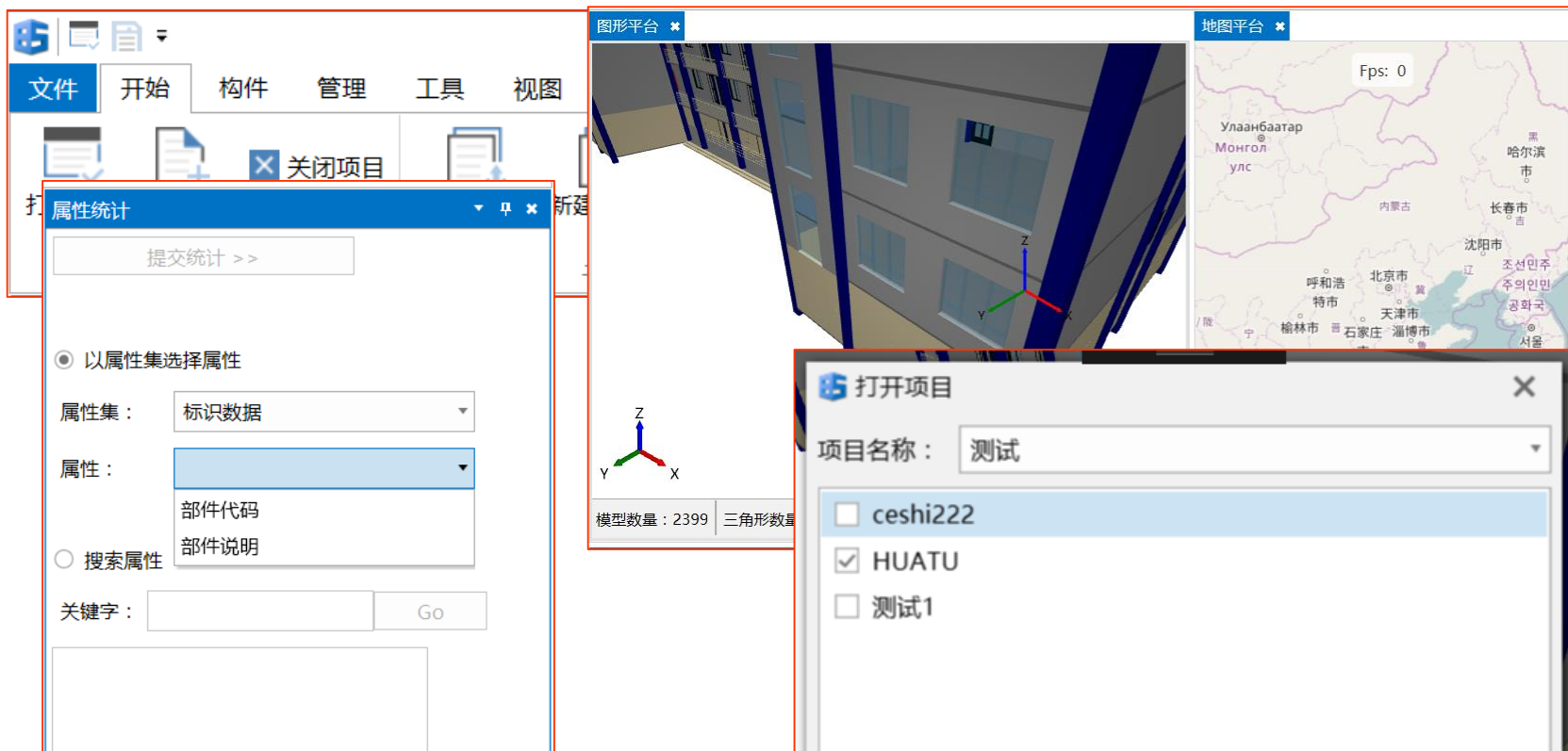
□ 图形用户界面开发

- 图形用户界面 (Graphical User Interface , GUI) 是指采用图形方式显示的计算机操作用户界面
- 窗口(Windows)、图标(Icons)、菜单(Menus)、指示器(Pointing Device)四位一体，形成桌面(Desktop)
- 用户界面技术的特点体现在三方面
 - ✓ 多视窗技术
 - ✓ 菜单技术
 - ✓ 联机帮助

BIM-Star二次开发及实例

□ 用户界面举例

- Ribbons菜单 弹出窗口 面板 图形平台



BIM-Star二次开发及实例

□ 编写用户界面的工具——WPF

- WPF (Windows Presentation Foundation) 是微软推出的用户界面框架
- 提供了统一的编程模型、语言和框架，真正做到了分离界面设计人员与开发人员的工作
- 同时它提供了全新的多媒体交互用户图形界面
- 是Windows系统的全新一代的界面编程框架
- 完全面向对象的对象模型。使用对象描述语言XAML
- 可以使用开发工具进行可视化编辑

BIM-Star二次开发及实例

□ WPF常用界面元素

```
<Label Margin="5" Content="选择日期：" Width="90"/>
```

```
<DatePicker ...../>
```

```
<Label Margin="5" Content="监测组：" Width="90"/>
```

```
<ComboBox Margin="5" Width="200" ...../>
```

```
.....  
<Button Content="查看构件详细..." ...../>
```

```
<DataGrid .....>
```

```
<DataGrid.Columns>
```

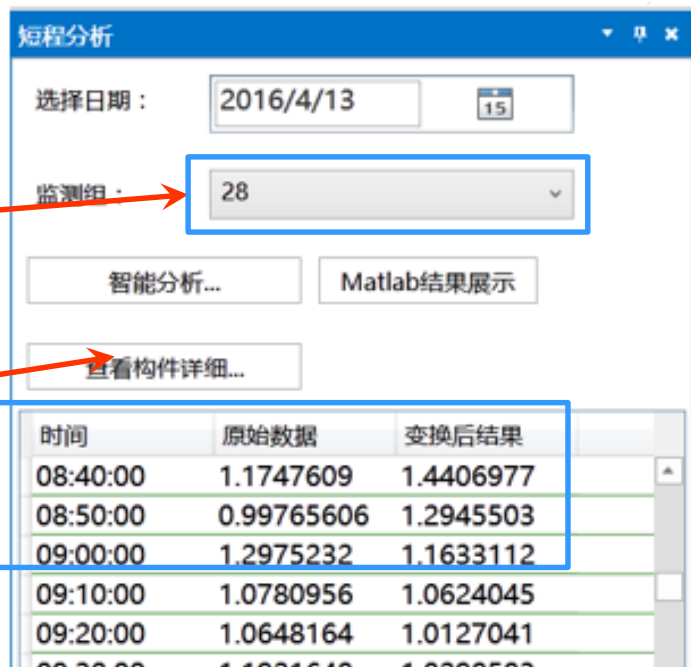
```
<DataGridTextColumn Header="时间" ...../>
```

```
<DataGridTextColumn Header="原始数据" ...../>
```

```
<DataGridTextColumn Header="变换后结果" ...../>
```

```
</DataGrid.Columns>
```

```
</DataGrid>
```



Xaml界面描述语言

BIM-Star二次开发及实例

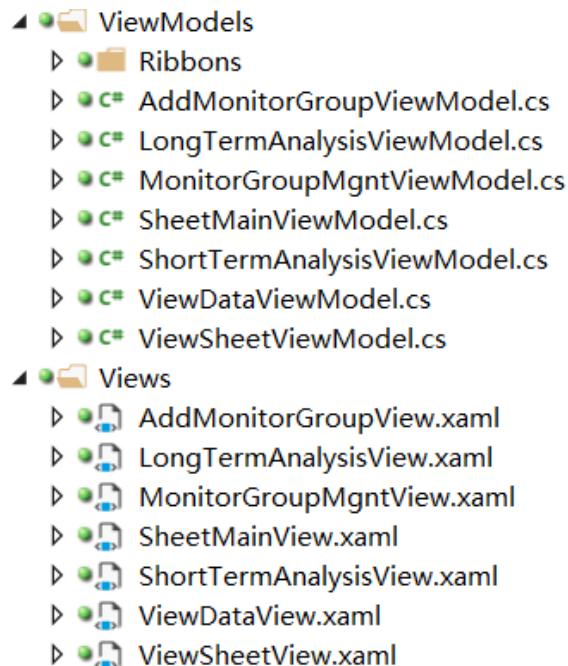
□ MVVM编程模型

Model-View-ViewModel

- 主要思路：采用WPF的数据绑定将界面和后台逻辑分离出来

特点和优势

- 1. 低耦合：视图和模型互不影响
- 2. 可重用性
- 3. 独立开发：[开发人员](#)可以专注于业务逻辑和数据的开发，[界面设计人员](#)可以专注于设计和展示数据
- 4. 方便测试



BIM-Star二次开发及实例

□ 按钮和触发命令

- 根据MVVM的编程原则，调用的命令和界面要分离
- 定义MyCmd命令

```
MyCmd = new RelayCommand(OnMyCmd);
```

- Ribbons菜单按钮（注册然后在XML中赋予命令名称）

```
M.CommandManager.Register(new CommandInfo("Cmd_MyCmd", MyCmd));
```

```
<Button Text="我的命令" ..... Click="Cmd_MyCmd"/>
```

- 用户界面的按钮（xaml中绑定命令变量）

```
<Button Content="我的命令" Command="{Binding Path=MyCmd}"/>
```

BIM-Star二次开发及实例

□ 其它控件

- BIMStar的API提供了快速创建面板和窗口的方法
- 面板：M.DockingManager.InsertPane(...)
- 对话框：M.DialogManager.ShowDialog(...)
- BIM图形平台为内置控件
- 图形平台支持Python脚本的控制
- 这些控件在MVVM模式下的用法将在上机课进行演示

BIM-Star二次开发及实例

□ 开发实例-背景

- 监测数据存在长期的周期规律
- 短程以天为单位、长程例如周和月
- 在小范围内无规律：楼中用户的非规律性行为导致一小段时间内的数据有较大的波动
- 需要一种方法分析长程规律和短程无规律的数据
- 以找到起主导因素的规律
- 若能在智能分析的基础上，进一步提供BIM监测数据的交互展示功能，则具有工程应用价值

BIM-Star二次开发及实例

□ 开发实例-技术和方法

《应用BIM和离散傅里叶变换的建筑监测数据智能分析和展示》

$$\tilde{X}(k) = DFS[\tilde{x}(n)] = \sum_{n=0}^{N-1} \tilde{x}(n) e^{-j\frac{2\pi}{N}nk} = \sum_{n=0}^{N-1} \tilde{x}(n) W_N^{nk}$$

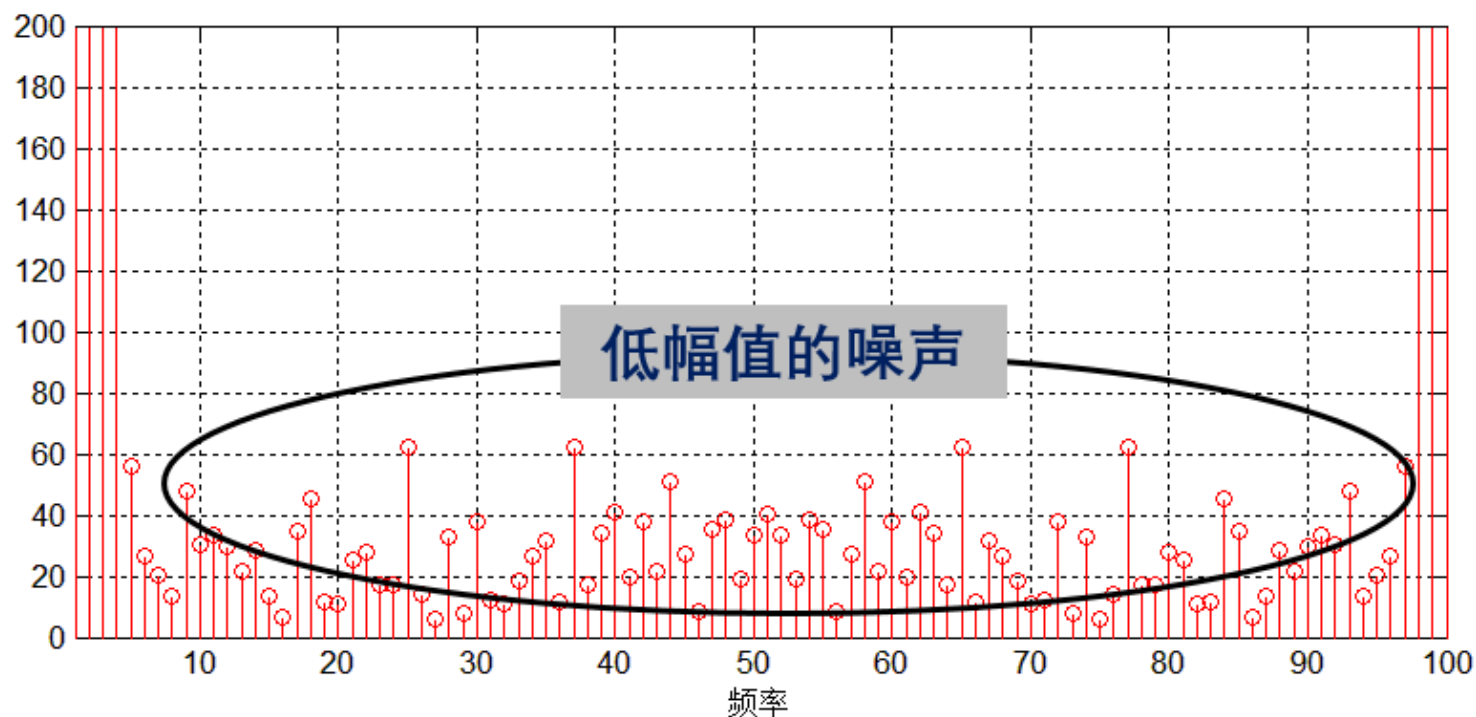
$$\tilde{x}(n) = IDFS[\tilde{X}(k)] = \frac{1}{N} \sum_{k=0}^{N-1} \tilde{X}(k) e^{j\frac{2\pi}{N}nk} = \frac{1}{N} \sum_{k=0}^{N-1} \tilde{X}(k) W_N^{-nk}$$

- 数学细节不展开讲，主要步骤概括为：
- 1、来自BIM集成的监测数据序列经过DFS（傅里叶正变换），得到频率
- 2、过滤低幅值的噪音
- 3、经过IDFS（逆变换），得到处理后的序列

BIM-Star二次开发及实例

□ 开发实例-技术和方法

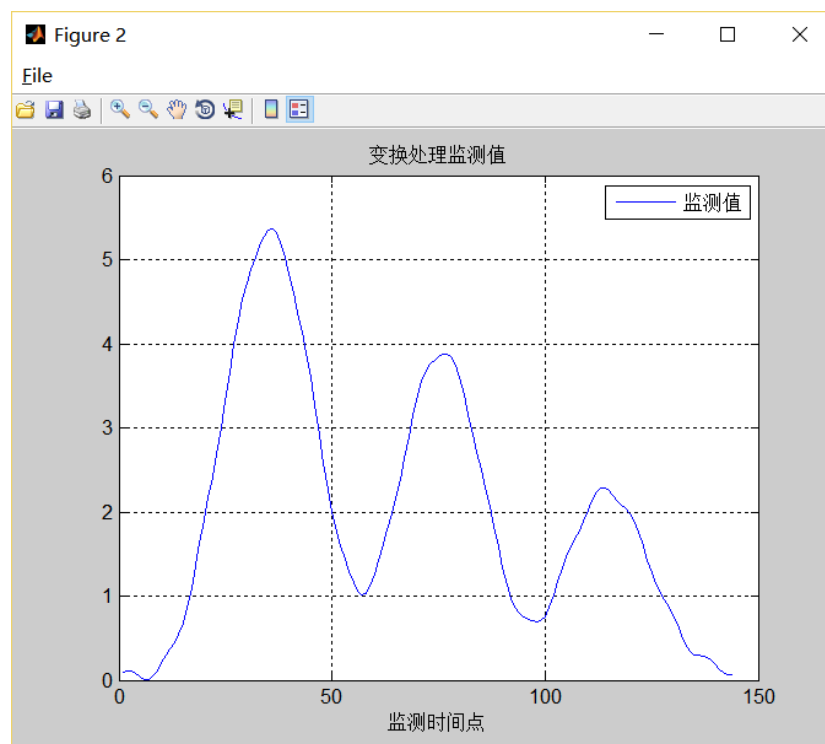
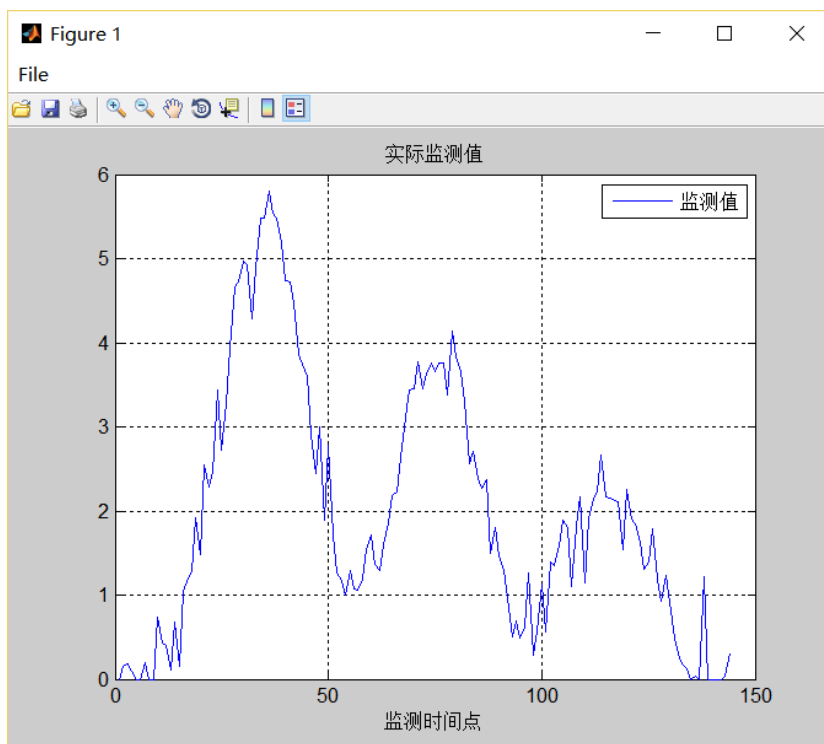
- BIM监测数据变换至频率表示：



BIM-Star二次开发及实例

□ 开发实例-技术和方法

- 原始监测数据 → 变换处理后



BIM-Star二次开发及实例

□ 开发实例-界面

The screenshot displays the BIM-Star software interface with several key components highlighted by red callouts:

- 监测分析功能区** (Monitoring Analysis Function Area): Located at the top, it includes buttons for '文件' (File), '开始' (Start), '构件' (Component), '管理' (Management), '工具' (Tools), '视图' (View), '系统' (System), '监测分析' (Monitoring Analysis), '属性统计' (Attribute Statistics), and 'Information Mining'.
- 关联构件的 BIM属性查询** (BIM Attribute Query for Associated Components): A panel on the right titled '属性管理器' (Attribute Manager) showing details for a specific component (ID: 383, Name: 止回阀 - H44 型 - 单...).
- 图形平台展示** (Graphic Platform Display): A 3D model of a mechanical system (pump and piping) in the center, with some components highlighted in yellow and green.
- 交互分析界面** (Interactive Analysis Interface): A panel on the left titled '短程分析' (Short-range Analysis) showing a date selector (2016/4/14), a monitoring group (28), and buttons for '智能分析...' (Smart Analysis...) and 'Matlab结果展示' (Matlab Results Display).
- MATLAB绘图** (MATLAB Plotting): Two windows at the bottom showing time-series plots of monitoring data. 'Figure 1' shows '实际监测值' (Actual Monitoring Value) and 'Figure 2' shows '变换处理监测值' (Transformed Monitoring Value).

Additional data visible in the interface includes a table of monitoring data and a summary of the short-range analysis results.

时间	原始数据	变换后数据
07:50:00	2.1033141	2.5059041
08:00:00	2.1500847	2.2026912
08:10:00	2.0646601	1.9209791
08:20:00	1.200874	1.6709743
08:30:00	1.1156426	1.4579909
08:40:00	1.9672324	1.2863186
08:50:00	1.4707965	1.159121
09:00:00	1.3128077	1.078366

【短程数据摘要】
变换后日平均: 2.011638862777794
变换后日最大值: 5.522645

BIM-Star二次开发及实例

□ 开发实例-功能1：导入数据，关联模型构件

- 导入数据文件
- 以监测数据的聚合汇总，并且关联至不同层次的构件组
- 关联则按数据所属监测组去匹配构件

导入数据至BIM数据库：

```
await mdata.MonitorRecordHub.NewRecord  
Async(recordElec.ToDictionary(), true);
```

```
await mdata.MonitorRecordHub.NewRecord  
Async(recordWater.ToDictionary(), true);
```



BIM-Star二次开发及实例

□ 开发实例-功能2：原始数据查询

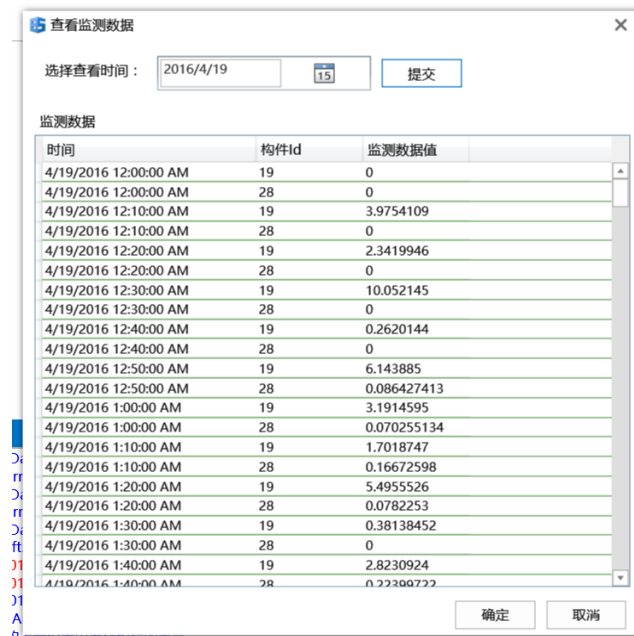
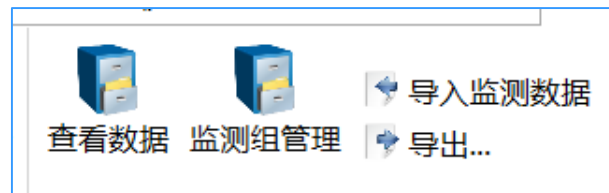
- 随时以交互界面查看BIM数据库

从BIM数据库获得数据：

```
var findResult = await mdata.MonitorRecordHub.GetAllAsync();
```

转换为监测记录类型并添加进AllRecords：

```
foreach (var r in findResult)
{
    var rcd = r.As<MonitorRecord>();
    AllRecords.Add(rcd);
}
```



BIM-Star二次开发及实例

□ 开发实例-功能3：监测组管理

- 在这里可以创建监测组，提供信息录入、更改和删除功能。然后将相关的机电构件添加至对应的组，在图形平台可查看各监测组包含的构件

得到当前项目所有构件的列表：

```
var nodelist = M.SceneManager.ProjectScene.DescendantsAndSelf<ISpatialNode>();
```

如果构件是选中状态，则添加一个关联关系：

```
foreach (var n in nodelist.Where(c => c.IsSelected && c.SpatialType ==  
SpatialType.Element)) {  
    var newItem = new RelSpatialToMonitorGroup(){  
        SpatialId = (int)n.Key,  
        MonitorGroupId = SelectedGroup.Id };  
    await _relHub.NewRecordAsync(newItem.ToDictionary(), true); }
```


BIM-Star二次开发及实例

□ 开发实例-功能3：监测组管理

监测组信息管理

图形平台展示

本监测组的关联构件列表

交互添加构件至监测组

形数量：514882 | FPS：60

【调试】- Core_Material中的数据已是最新版本！
【通知】- 正在构造子项目“监测”的项目树，请
【通知】- 子项目“监测”的项目树构造完毕！
【通知】- 子项目 监测 加载完毕！
【调试】- Mgnt_User同步完成，其数据已是最新版本！
【调试】- Monitor_Group同步完成，其数据已是最新版本！
【调试】- Monitor_Rel_SpatialToMonitorGroup中的数据已是最新版本！

监测组管理

监测组

将列标题拖放到此处，然后以该列分组

名称	创建时间	标识号
三层排水泵	5/3/2016 8:36:29 AM	12
三层水管总组	5/5/2016 3:45:03 PM	13
底层排水	5/5/2016 4:52:29 PM	14
二层管道	5/6/2016 11:58:58 AM	15

添加监测组... 删除组

关联构件

名称	标识号
管道类型:标准:842714	1192
管道类型:标准:842723	1193
管道类型:标准:842779	1194
管道类型:标准:861385	1197
管道类型:标准:863117	1200
弯头 - 常规:标准:802837	1205
弯头 - 常规:标准:802841	1206
弯头 - 常规:标准:802847	1207

添加关联构件 删除关联

属性管理器 监测组管理

BIM-Star二次开发及实例

□ 开发实例-功能4：短程、长程智能分析

- **短程智能分析** 分析的元素：天 / 细度：时、分、秒
- **长程智能分析** 分析的元素：长时间跨度 / 细度：天

运行本插件附带的外部数学分析程序：

```
ProcessStartInfo info = new ProcessStartInfo();  
info.FileName = typeof(MonitorAnalysisGroupViewModel).  
GetPluginResPath("Assets/coreDFS.exe"); // 获取插件路径，API提供功能  
info.Arguments = "";  
info.WindowStyle = ProcessWindowStyle.Minimized;  
Process pro = Process.Start(info);  
pro.WaitForExit();  
  
MessageBox.Show("分析完成！");
```

BIM-Star二次开发及实例

□ 开发实例-功能4：短程、长程智能分析

短程分析

选择日期：2016/4/13 15

监测组：28

智能分析... Matlab结果展示

查看构件详细...

时间	原始数据	变换后结果
08:40:00	1.1747609	1.4406977
08:50:00	0.99765606	1.2945503
09:00:00	1.2975232	1.1633112
09:10:00	1.0780956	1.0624045
09:20:00	1.0648164	1.0127041
09:30:00	1.1921648	1.0298583
09:40:00	1.5451201	1.1158217
09:50:00	1.7192693	1.2570518

显示 >>

【短程数据摘要】

变换后日平均：2.0033162745104169

变换后日最大值：5.371111

长程分析

监测组：19

智能分析... Matlab结果展示

选择日期：2016/4/22 15

查看构件详细...

单日平均监测值 22.444134579268116

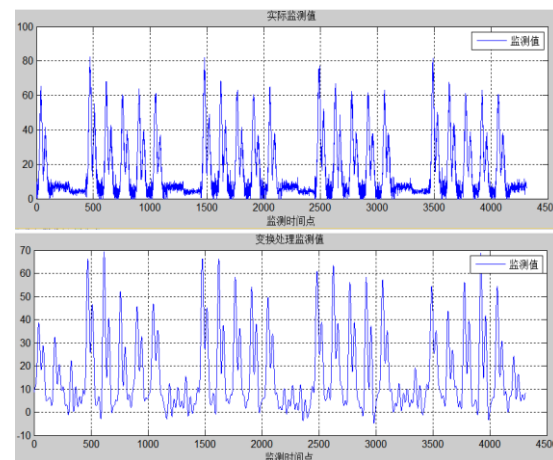
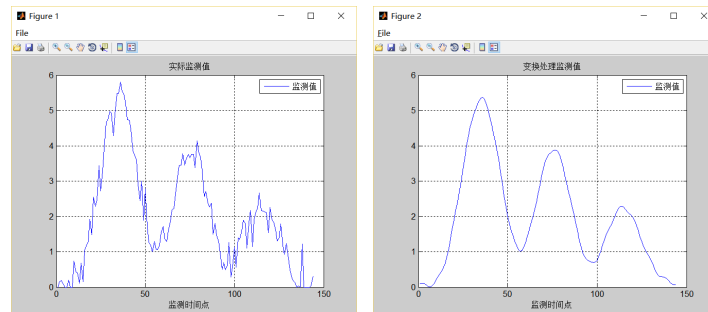
单日最大监测值 57.091406

显示 >>

【长程数据摘要】

变换后长程均值：16.247637742053893

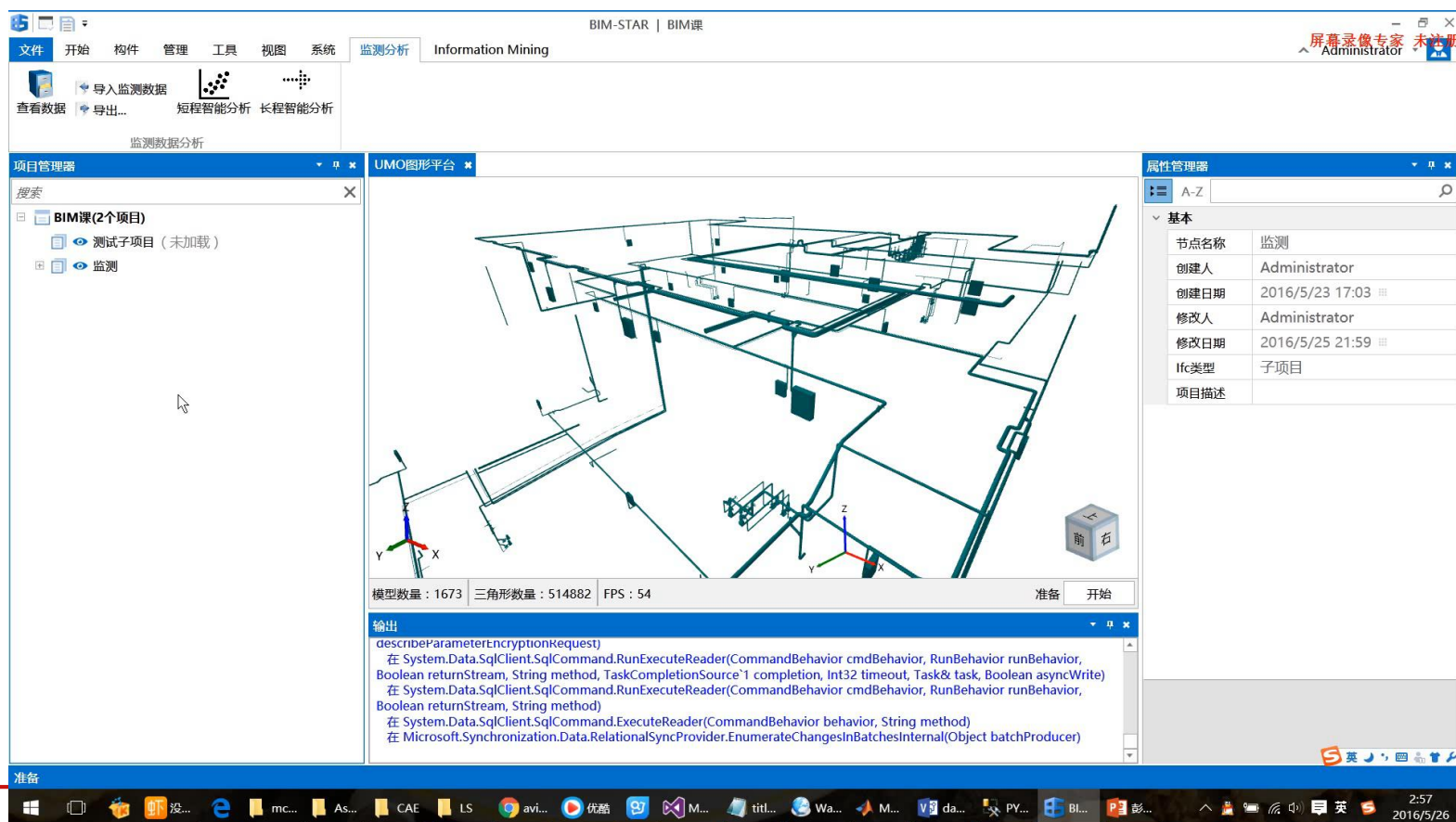
变换后长程最大值：69.391099



BIM-Star二次开发及实例

□ 开发实例-演示视频

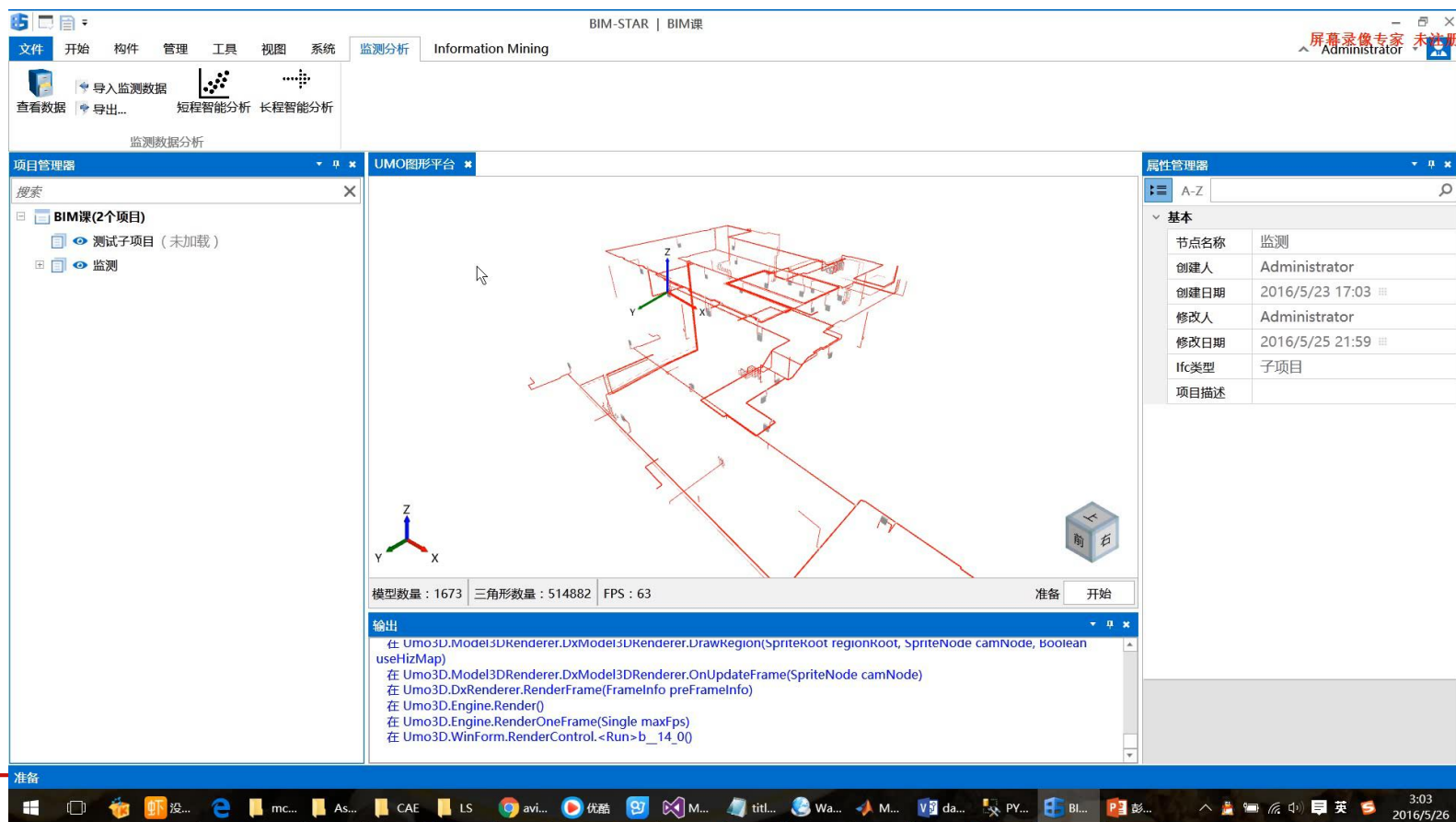
- 短程分析



BIM-Star二次开发及实例

□ 开发实例-演示视频

- 长程分析



BIM-Star二次开发及实例

□ 开发实例-图形平台的控制：API控制

- BIMStar提供了操作图形显示状态的API

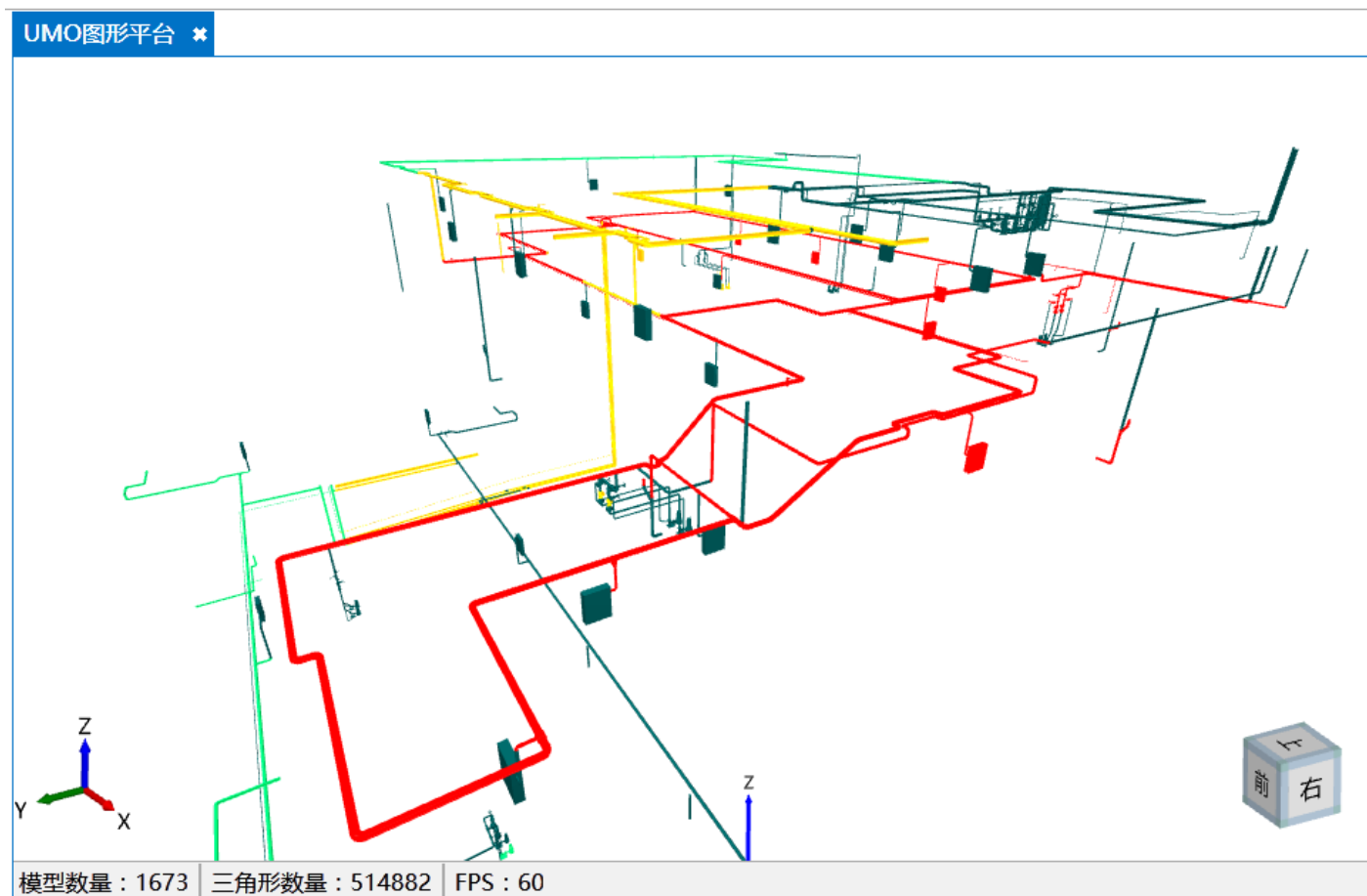
为当前分析组的构件指定颜色：

```
if (SelectedMonitorGroup == currentGroup) {  
    foreach (var node in nodelist)  
    {  
        node.RenderStyle = new RenderStyle(); // API提供的样式类型  
        node.RenderStyle.R = colorR;  
        node.RenderStyle.G = colorG;  
        node.RenderStyle.B = colorB;  
    }  
}
```

注：修改RenderStyle.A的值即是更改透明度（0.00-1.00）

BIM-Star二次开发及实例

□ 开发实例-图形平台的控制：API控制

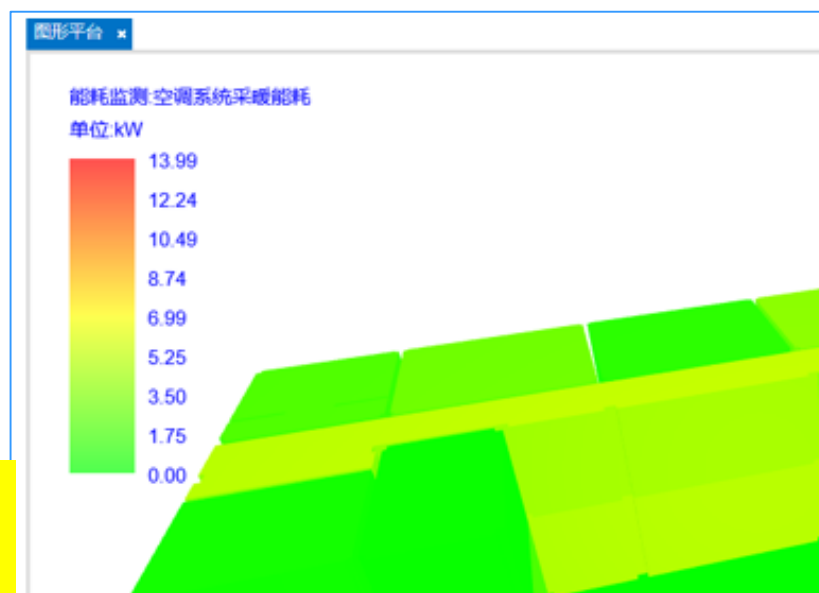


BIM-Star二次开发及实例

□ 开发实例-图形平台的控制：脚本控制

脚本控制方法的使用场合

- 1、需要更精细地控制BIM元素的显示
- 2、直接与图形几何计算有关的功能
- 3、动画效果
- 4、画二维图形



图形平台中插入固定的
文字和图例标尺

BIM-Star二次开发及实例

□ 开发实例-演示视频

- 使用脚本控制摄像机动画



提纲

- 概述
- 面向对象编程
- BIM-Star二次开发及实例
- 结语

结语

- 知识层面

- 了解面向对象程序开发方法
- 了解BIM系统二次开发
- 掌握BIM-Star二次开发技术

- 科学方法论层面

- 授人以鱼不如授人以渔
- 互联网思维在BIM系统开发的渗透

谢谢！



清华大学土木工程系

胡振中 副教授

Email: huzhenzhong@tsinghua.edu.cn

个人网站: <http://www.huzhenzhong.net>
