一、 文档规范

1. 字体及样式使用规范

不使用非商用免费字体

中文使用思源黑体、黑体、隶书等免费字体

英文使用 consolas

2. 文档内编程语句规范

非必要情况下，不出现代码语句，只展示定义。

类型使用浅色标注(rgb(128,128,128))。

定义的名称使用加粗字体。

私有内容使用斜体。

方法定义，若参数列表为空，括号与函数名保持在同一行；若参数列表不为空，参数列表单独一行。。

代码使用“代码”格式。

如：

```
public static string Fun1()

private string Fun2
(string str, int start)
```

单独的代码块使用外侧框线框起来，配合表格使用的代码段（如模块具体功能中的方法的声明和解释）可忽略外框线

3. 注释规范

可使用中括号括起来的内容对内容做注释，并对内容以浅色颜色(rgb(128,128,128))

如

```
[已完成][优先]文档自动更新模块
```

- 统一注释：

[WPF]：使用了 WPF 技术的模块

[static]：静态的模块

[i]：interface，接口

[a]：abstract，抽象类

[FE]：前端工厂（FrontEndFactory）相关

[BE]：前端工厂（BackEndFactory）相关

二、 软件简介

1.开发目的

整合三维地形生成及环境模拟运算，为非精确的大规模地形建模工程提供便利，为快速的地形生成算法做实验性研究平台。 并且基于此软件提供的平台实验自行设计的随机趋势化地形生成算法，为以后的实验提供便利。

2.软件功能要求

提供一个统一的、易扩展的平台环境，为相关算法的实验、应用提供一个可行的环境。 Studio 要有对工程、项目和相关资源管理能力。

算法分为前端工厂部分和后端工厂部分。前端工厂为地形生成器，该部分要通过接收一组设定值而产生一个合理的二维的高度数据，作为地形的高度信息。后端工厂要根据用户的设定和其他后端工厂的产出数据，对前端工厂产出的地形进行合理的环境模拟。

软件将实现前端工厂的随机趋势化算法和后端工厂中的空气流动模拟、降水模拟及生态系统模拟。Studio 要求能对工厂产出的数据进行整合，并且可以导出为三维模型或是二维贴图，以方便后续对产出数据的使用。

3.开发技术及软件运行目标环境

a) 开发技术

- C# （C#7.0）
- WPF （WPF4.5）

- .NET Framework （.NET Framework 4.7.2）

b) 开发环境

- Visual Studio 2017 Community

c) 软件运行目标环境

- .NET Framework 4.7.2 及兼容版本
- Windows 10 1607 及后支持.NET Framework 4.7.2 的操作系统
- 显示器 800 * 600 及更大分辨率
- 内存：剩余 1G 以上可用的内存空间
- 磁盘：剩余 2G 以上可用的磁盘空间，磁盘可读写

4. 参考文献

1. 《WPF 编程宝典——使用 c#2012 和.NET 4.5》（第 4 版） （清华大学出版社 [美]Matthew MacDonald 著，王德才 译）

5. 相关链接

- **C#7.0**

https://docs.microsoft.com/zh-cn/dotnet/csharp/whats-new/csharp-7

- **WPF4.5**

https://docs.microsoft.com/zh-cn/dotnet/framework/wpf/getting-started/whats-new

- **.NET Framework 4.7.2 Download Page**

https://dotnet.microsoft.com/download/dotnet-framework-runtime/net472

- **Visual Studio**

https://visualstudio.microsoft.com/zh-hans/

- **MIT 开源协议**

https://opensource.org/licenses/mit-license.php

三、 软件逻辑设计

1.工程管理逻辑

软件工程逻辑为树形分层逻辑，根节点为工程节点（Project）。工程节点下分为多个工作节点（Work），每一个工作节点对应一个地形从生成到最终的环境模拟演变结束的整个过程。在同一工程内的各个工作是相互关联而又相互独立的，他们既可以是同一个项目的不同部分的地形，也可以通过更强大的合成器将多个地形合成到一起，实现更加复杂的效果控制。

工作分为图片资源管理器、前端工厂和后端工厂。图片资源管理器管理整个工作过程中产生的图片资源，进行统一管理。其他节点只能通过引用这些资源来进行图片资源的分类。

2.功能使用操作方式

软件使用流水线工作方式，只有在依赖的前一工作点完成工作后才可使用当前工作点。

流程示意如下：

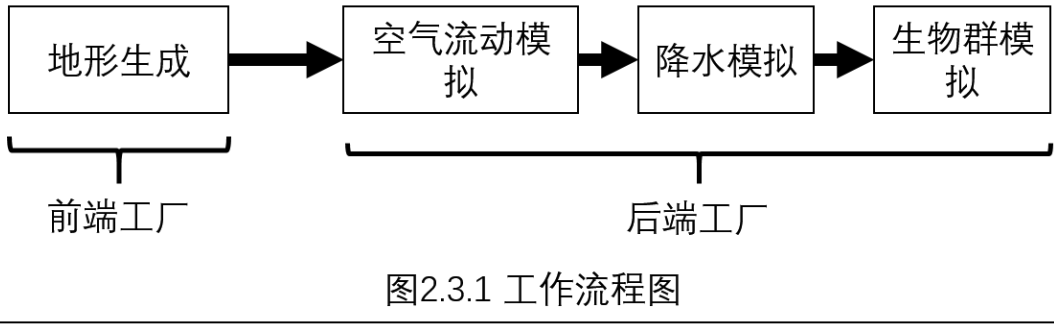


图2.3.1 工作流程图

同一节点只依赖前一节点产生的数据，不依赖前一节点的算法类型，故可根据需求自行选择合适的算法或自行扩展。

四、 工程文件格式及存储结构

工程是相关工作的集合，包含一个或多个工作，这些工作彼此相互独立但又相互联系。

工作是一个设定参数的具体实施过程和相关资源的管理和集合。每一个工作都可以产生一块地形，并且根据流程进行后端的环境模拟及结果的导出。

1. 工程文件格式

工程涉及到两种文件存储，为工程文件和工作文件。

a) 工程文件

工程文件后缀名为 **mriwcpro**（MiRal World Creater Project），以标准 xml 文档格式存储。

工程文件记录了工程的基本信息（GUID、工程名）和工程拥有的工作信息（GUID、工作所在的文件夹、工作文件名，工作名）。

工程文件内容具体定义如下：

```
<project guid="[工程 GUID]" name="[工程名称]">
  <work guid="[工作 1GUID]" dictionary="[工作 1 存储文件夹]" file="[工作 1 文件]"
name="[工作 1 名]" />
  <work guid="[工作 2GUID]" dictionary="[工作 2 存储文件夹]" file="[工作 2 文件]"
name="[工作 2 名]" />
</project>
```

如

```
<project guid="c99a3e2b-58eb-47e4-9f1a-85cf72f19deb" name="DemoPro">
  <work guid="46a1d881-f42d-4a7e-a1cf-f779325cc017" dictionary="demowork"
file="demowork.mriwcw" name="测试工作集" />
</project>
```

b) 工作文件

工作文件后缀名为 **mriwcw**（MiRal World Creater Work），以标准 xml 文档格式存储。

工作文件记录了工作的基本信息（GUID）和工作拥有的图片资源(images 节点)、工作的前端工厂信息、工作的后端工厂信息。

工作文件内容具体定义如下：

```
<work guid="[工作 GUID]">
      <images>
            <image key="[资源 1 唯一键]" name="[资源 1 展示名称]" file="[资源 1 文件，根
目录为工作文件夹下 images 文件夹]" description="[对资源 1 的描述信息]"/>
            <image key="[资源 2 唯一键]" name="[资源 2 展示名称]" file="[资源 2 文件，根
目录为工作文件夹下 images 文件夹]" description="[对资源 2 的描述信息]"/>
      </images>
```

```
      <FrontEndFactory creater="[记录 Creater 的 ProgramSet 信息]">
            <setting>
                  [该节点为相应类产生其具体内容]
            </setting>
            <images>
                  <image key="[引用的资源 key]">
            </images>
      </FrontEndFactory>

      <BackEndFactory>
            <Factory name="[后端工厂展示名称]" progress="[该后端工厂所处理的内容]"
factory="[后端工厂的 ProgramSet]">
                  [该节点为相应类产生其具体内容]
                  <images>
                        <image key="[引用的资源]">
                  </images>
            </Factory>
      </BackEndFactory>
</work>
```

如

```
<work guid="">
      <images>
            <image key="res_1" name="this is a name" file="path base on project
flodar/images" description="description"/>
            <image key="res_2" name="Random Map" file="f-random.png"
description="the radom value of whole map"/>
      </images>

      <FrontEndFactory creater="MiRaI.RandomTend.RandomTend|0.1">
            <setting>
                  <add key="" value="" />
            </setting>
            <images>
                  <image key="res_1">
            </images>
      </FrontEndFactory>

      <BackEndFactory>
            <Factory name="空气运动" progress="AirMotion"
factory="BE.MiRaI.AirMotion|0.1">
                  <data file="airmotion.data" />
                  <images>
                        <image key="res_2">
                  </images>
            </Factory>
```

```
    </BackEndFactory>
</work>
```

2.工程文件的存储逻辑

```
Project
|-proj.mrimcproj [工程文件]
|-Geo1
| |-geo1.mrigeoproj [工作文件]
| |-Images [工作中产生的图像资源]
| | |-img1.png
| | |-……
| |-RandomMap.dataraw [随机数图]
| |-HeightMap.dataraw [高度图]
| |-TerrainMap.dataraw [地势图]
| |-AirMotion.dataraw [空气流动图]
| |-Precipitation.dataraw [降水]
| |- [其他产生的过程数据]
|-Geo2
|-……
```

3.工程的展示逻辑

```
Project【不可点击】
|-Geo1【工程设置】
| |-RandomMap【随机数图】
| |-HeightMap【高度图】
| | |-SubMap 子图 1
| | |-SubMap 子图 2
| |-TerrainMap【地势图】
| |-LaterFactory【后期工厂】
|   |-AirMotion【空气流动图】
|   |-Precipitation【降水】
|   |-【水域】
|   |-【生物群】
|-Geo2
| |-RandomMap【随机数图】
| |-HeightMap【高度图】
| |-TerrainMap【地势图】
| |-LaterFactory【后期工厂】
|   |-AirMotion【空气流动图】
|   |-Precipitation【降水】
|   |-【水域】
|   |-【生物群】
```

五、 模块简介及功能概览

1. WorldCreaterStudio

a) 模块功能

软件 UI 提供模块，将各模块功能整合提供完整的操作流程。

2. WorldCreaterStudio_Core

a) 模块功能

WorldCreaterStudio 模块的核心算法、核心类，并为算法功能模块定义基本的开发接口。

3. WorldCreaterStudio_Resouses

a) 模块功能

存储 WorldCreaterStudio 所使用到的资源。

4. [FE]FrontEndFactorys.RandomTend

a) 模块功能

随机趋势化算法核心代码。且实现 WorldCreaterStudio_Core 模块定义的接口。

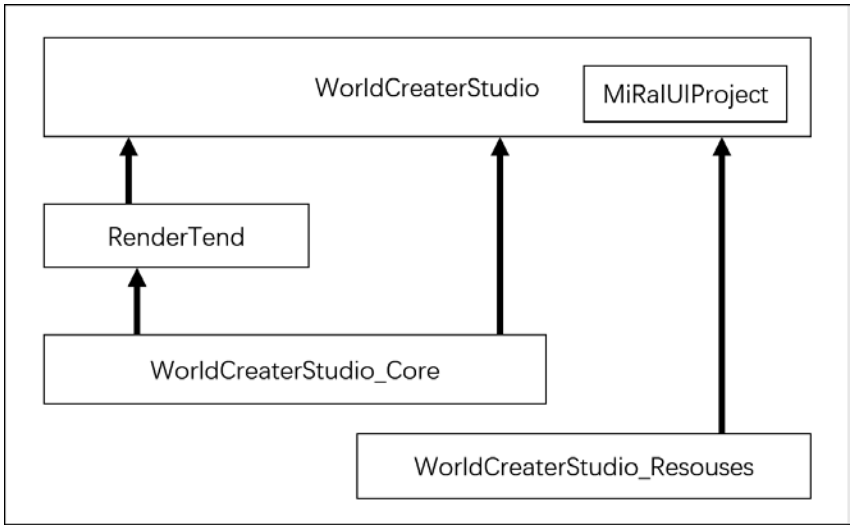
5. MiRaIUIProject

该模块现已独立为单独的工程，并已通过 MIT 协议开源。开原地址：https://github.com/ijrys/MiRaIUIProject

a) 模块功能

提供 WPF 程序的的 UI 样式。

6.模块依赖关系图



六、 ValueToImage 模块功能细则

1.[static]ValueToGrayImage

将高度数据转换为可视的灰度图

a) 方法

- **GetBitmap**

```
public static WriteableBitmap GetBitmap(int minvalue, int maxvalue, byte mingray, byte maxgray, int[,] map)
```

获取一个灰度图。

minvalue	高度的最小值，小于或等于该值的将显示为最小亮度
maxvalue	高度的最大值，大于或等于该值的将显示为最大亮度
mingray	设定最小亮度
maxgray	设定最大亮度
map	高度数据

- **GetBitmapWithError**

```
public static WriteableBitmap GetBitmapWithError (int minvalue, int maxvalue, int[,] map)
```

获取一个带有高度超出检查的灰度图，用于检查生成的地形高度数据是否在合法范围内。

minvalue	高度数据的下限，小于这个值的区域将设置为红色(255,0,0)
maxvalue	高度数据的上限，大于这个值的区域将设置为青色(0,255,255)
map	高度数据

- **GetPixel**

```
public static byte GetPixel (int minvalue, int maxvalue, byte mingray, byte maxgray, int value)
```

获取一个值在高度范围内的转换结果

minvalue	高度的最小值，小于或等于该值的将返回 mingray
maxvalue	高度的最大值，大于或等于该值的将返回 maxgray
mingray	设定最小亮度
maxgray	设定最大亮度
value	高度数据

2. [static]ValueToColorImage

将高度数据转换为可视的彩色图

a) 方法

- **GetBitmap**

```
public static WriteableBitmap GetBitmap (int minvalue, int maxvalue, int[,] map)
```

获取一张彩色图。海平面值为 0。

minvalue	高度数据的下限，小于这个值的区域将视为 minvalue

maxvalue	高度数据的上限，大于这个值的区域将视为 maxvalue
map	高度数据

3. [static]AtmosphericMotionToImage

配合 AtmosphericMotion 的值转换器。

a) 方法

- **GetBitmap**

```
public static WriteableBitmap GetBitmap (PointData[,] datas)
```

获取区域风向图

datas	风向数据

4. [static]RainfallMotionToImage

配合 RainfallMotion 的值转换器。

a) 方法

- **GetRainfallIntensityBitmap**

```
public static WriteableBitmap GetRainfallIntensityBitmap (PointData[,] datas)
```

获取降雨强度可视化图。

datas	原始数据

- **GetAreaTpyeBitmap**

```
public static WriteableBitmap GetAreaTpyeBitmap (PointData[,] datas)
```

获取区域类型可视化图

datas	原始数据

- **GetWaterDeepBitmap**

```
public static WriteableBitmap GetWaterDeepBitmap (PointData[,] datas)
```

获取水深可视化图

datas	原始数据

5. SolarIlluminance

配合 SolarIlluminance 的值转换器。

a) 方法

- GetBitmap

```
public static WriteableBitmap GetBitmap (byte[,] value)
```

获取光照强度图，返回值为 Gray8

value	原始数据

6. BiomesToImage

配合 BiomesToImage 的值转换器。

a) 方法

- **GetBitmap**

```
public static WriteableBitmap GetBitmap (BiomesType[,] value)
```

获取生物分布图

value	原始数据

七、 BackEndFactorys/AtmosphericMotion/MiRaI.BE.AM.SingleValue 模块功能细则

1. SingleValue

使用单值快速填充的空气运动模拟器。

继承：IAtmosphericMotionCalculaterAble

a) 属性

名称	类型	可赋值	可获取	默认值	描述
CreaterName	string	F	T	"单一值风力设定"	获取模拟器名称
CreaterProgramSet	string	F	T	"MiRaI.BE.AM.SV|0.1"	获取模拟器程序集
CreaterGuid	Guid	F	T	typeof (SingleValue).GUID	获取模拟器 GUID

b) 委托

- **OnProcessingChanged**

```
public event DataCalculatingProcessingEventType OnProcessingChanged;
```

模拟过程产生新进度消息时事件

c) 方法

- **GetAtmosphericMotionDatasBySpecialConfig**

```
public AtmosphericMotionResault GetAtmosphericMotionDatasBySpecialConfig
(SingleValueConfig config, int[,] heightMap, Work work)
```

使用 SingleValueConfig 类型的配置项进行模拟

config	配置设置

heightMap	高度数据
work	执行方法的工作集

- **GetAtmosphericMotionDatas**

```
public AtmosphericMotionResault GetAtmosphericMotionDatas
(IAtmosphericMotionConfigAble config, int[,] heightMap, Work work)
```

使用一个 IAtmosphericMotionConfigAble 类型的配置项进行模拟。

若不为 SingleValueConfig 类型，抛出 IncongruentConfigurationException

config	配置设置
heightMap	高度数据
work	执行方法的工作集

2. SingleValueConfig

记录 SingleValue 进行模拟所需要的配置数据

继承：IAtmosphericMotionConfigAble

a) 属性

b) 委托

- **ValueChanged**

```
public event NodeValueChangedEventType ValueChanged
```

内容发生更改时事件

- **PropertyChanged**

```
public event PropertyChangedEventHandler PropertyChanged
```

属性发生更改时通知绑定目标

c) 方法

- **LoadFromXMLNode**

```
public void LoadFromXMLNode (XmlElement xmlnode)
```

从 XML 节点中加载配置数据

xmlnode	数据来源的 XML 节点

- **XmlNode**

```
public XmlElement XmlNode (XmlDocument xmlDocument, bool save = false)
```

获取配置的 XML 描述节点

xmlDocument	XML 节点所在文档的根节点
save	是否为保存动作

3. SingleValueFactory

a) 属性

名称	类型	可赋值	可获取	默认值	描述
DisplayName	string	F	T	单一值风力设定	展示用的名称，与 Calculater 的 CreaterName 相同
DisplayType	string	F	T	BE.AM	展示所属类型
CalculaterProgramSet	string	F	T	MiRaI.BE.AM.SV|0.1	获取模拟器的程序集，与 Calculater 的 CreaterProgramSet 相同
CalculaterGuid	Guid	F	T	typeof (SingleValue).GUID	获取模拟器的 GUID，与 Calculater 的 CreaterGuid 相同

b) 方法

- **GetACalculater**

```
public IAtmosphericMotionCalculaterAble GetACalculater ()
```

获取一个模拟器

- **GetAConfiguration**

```
public IAtmosphericMotionConfigAble GetAConfiguration ()
```

获取一个相应的配置对象

八、 BackEndFactorys/RainfallMotion/MiRaI.BE.RM.SingleValu

e 模块功能细则

1. SingleValue

使用单值快速填充的降水运动模拟器。

继承：IRainfallMotionCalculaterAble

a) 属性

名称	类型	可赋值	可获取	默认值	描述
CreaterName	string	F	T	"单一值快速设定"	获取模拟器名称
CreaterProgramSet	string	F	T	"MiRaI.BE.RM.SV|0.1"	获取模拟器程序集
CreaterGuid	Guid	F	T	typeof (SingleValue).GUID	获取模拟器 GUID

b) 委托

- **OnProcessingChanged**

```
public event DataCalculatingProcessingEventType OnProcessingChanged
```

模拟过程产生新进度消息时事件

c) 方法

- **GetAtmosphericMotionDatasBySpecialConfig**

```
public AtmosphericMotionResault GetAtmosphericMotionDatasBySpecialConfig
(SingleValueConfig config, int[,] heightMap, Work work)
```

使用 SingleValueConfig 类型的配置项进行模拟

config	配置设置
heightMap	高度数据
work	执行方法的工作集

- **GetAtmosphericMotionDatas**

```
public AtmosphericMotionResault GetAtmosphericMotionDatas
(IAtmosphericMotionConfigAble config, int[,] heightMap, Work work)
```

使用一个 IAtmosphericMotionConfigAble 类型的配置项进行模拟。

若不为 SingleValueConfig 类型，抛出 IncongruentConfigurationException

config	配置设置
heightMap	高度数据
work	执行方法的工作集

2. SingleValueConfig

记录 SingleValue 进行模拟所需要的配置数据

继承：IRainfallMotionConfigAble

a) 属性

名称	类型	可赋值	可获取	默认值	描述
RainfallIntensity	int	T	T		获取或设置降水强度。每单位时间降水 0.01 个全局高度单位
DisplayRainfallIntensity	double	F	T	RainfallIntensity / 100	用于展示的真实降水强度
SeaLevel	int	T	T		获取或设置海平面高度，与高度图高度同单位
ShowPanel	ControlTemplate	F	T		获取相配套的界面模板

b) 委托

- **ValueChanged**

```
public event NodeValueChangedEventType ValueChanged
```

内容发生更改时事件

- **PropertyChanged**

```
public event PropertyChangedEventHandler PropertyChanged
```

属性发生更改时通知绑定目标

c) 方法

- **LoadFromXMLNode**

```
public void LoadFromXMLNode (XmlElement xmlnode)
```

从 XML 节点中加载配置数据

xmlnode	数据来源的 XML 节点

- **XmlNode**

```
public XmlElement XmlNode (XmlDocument xmlDocument, bool save = false)
```

获取配置的 XML 描述节点

xmlDocument	XML 节点所在文档的根节点
save	是否为保存动作

3. SingleValueFactory

用于创建、获取相应的 Calculater 和 Config 的工厂类

继承：IRainfallMotionCalculaterFactoryAble

a) 属性

名称	类型	可赋值	可获取	默认值	描述
DisplayName	string	F	T	"单一值快速设定"	展示用的名称，与 Calculater 的 CreaterName 相同
DisplayType	string	F	T	"BE.RM"	展示所属类型
CalculaterProgramSet	string	F	T	"MiRaI.BE.RM.SV|0.1"	获取模拟器的程序集，与 Calculater 的 CreaterProgramSet 相同
CalculaterGuid	Guid	F	T	typeof (SingleValue).GUID	获取模拟器的 GUID，与 Calculater 的 CreaterGuid 相同

b) 方法

- **GetACalculater**

```
public IAtmosphericMotionCalculaterAble GetACalculater ()
```

获取一个模拟器

- **GetAConfiguration**

```
public IAtmosphericMotionConfigAble GetAConfiguration ()
```

获取一个相应的配置对象

九、 BackEndFactorys/SolarIlluminance/MiRaI.BE.SI.QuickCalculating 模块功能细则

1. QuickCalculating

利用法线快速模拟光照强度模拟器。

继承：ISolarIlluminanceCalculaterAble

a) 属性

名称	类型	可赋值	可获取	默认值	描述
CreaterName	string	F	T	"反射面法线快速计算"	获取模拟器名称
CreaterProgramSet	string	F	T	"MiRaI.BE.SI.QC|0.1"	获取模拟器程序集
CreaterGuid	Guid	F	T	typeof (QuickCalculating).GUID	获取模拟器 GUID

b) 委托

- **OnProcessingChanged**

```
public event DataCalculatingProcessingEventType OnProcessingChanged
```

模拟过程产生新进度消息时事件

c) 方法

- **GetSolarIlluminanceResaultDatasBySpecialConfig**

```
public SolarIlluminanceResault GetSolarIlluminanceResaultDatasBySpecialConfig
(QuickCalculatingConfig config, int[,] heightMap, Work work)
```

使用 QuickCalculatingConfig 类型的配置项进行模拟

config	配置设置
heightMap	高度数据
work	执行方法的工作集

- **GetAtmosphericMotionDatas**

```
public SolarIlluminanceResault GetSolarIlluminanceDatas
(ISolarIlluminanceConfigAble config, int[,] heightMap, Work work)
```

使用一个 ISolarIlluminanceConfigAble 类型的配置项进行模拟。

若不为 ISolarIlluminanceConfigAble 类型，抛出 IncongruentConfigurationException

config	配置设置
heightMap	高度数据
work	执行方法的工作集

2. QuickCalculatingConfig

记录 QuickCalculating 进行模拟所需要的配置数据

继承：ISolarIlluminanceConfigAble

a) 属性

名称	类型	可赋值	可获取	默认值	描述

Angle	double	T	T		获取或设置光照角度
ShowPanel	ControlTemplate	F	T		获取相配套的界面模板

b) 委托

- **ValueChanged**

```
public event NodeValueChangedEventType ValueChanged
```

内容发生更改时事件

- **PropertyChanged**

```
public event PropertyChangedEventHandler PropertyChanged
```

属性发生更改时通知绑定目标

c) 方法

- **LoadFromXMLNode**

```
public void LoadFromXMLNode (XmlElement xmlnode)
```

从 XML 节点中加载配置数据

xmlnode	数据来源的 XML 节点

- **XmlNode**

```
public XmlElement XmlNode (XmlDocument xmlDocument, bool save = false)
```

获取配置的 XML 描述节点

xmlDocument	XML 节点所在文档的根节点
save	是否为保存动作

3. QuickCalculatingFactory

用于创建、获取相应的 Calculater 和 Config 的工厂类

继承：IRainfallMotionCalculaterFactoryAble

a) 属性

名称	类型	赋值	获取	默认值	描述
DisplayName	string	F	T	"反射面法线快速计算"	展示用的名称，与 Calculater 的 CreaterName 相同
DisplayType	string	F	T	"BE.SI"	展示所属类型
CalculaterProgramSet	string	F	T	"MiRaI.BE.SI.QC|0.1"	获取模拟器的程序集，与 Calculater 的 CreaterProgramSet 相同
CalculaterGuid	Guid	F	T	typeof (QuickCalculating).GUID	获取模拟器的 GUID，与 Calculater 的 CreaterGuid 相同

b) 方法

- **GetACalculater**

```
public IsolarIlluminanceCalculaterAble GetACalculater ()
```

获取一个模拟器

- **GetAConfiguration**

```
public IsolarIlluminanceConfigAble GetAConfiguration ()
```

获取一个相应的配置对象

十、 WorldCreaterStudio 模块功能细则

1. [static]Commands

Studio 使用到的自定义命令

a) 构造函数

```
static Commands()
```

b) 属性

声明	描述
public static RoutedUICommand **NewWork**	命令・新建工作
public static RoutedUICommand **NewProject**	命令・新建工程

2. [static]Resouses/StoreRoom

a) 属性

声明	描述
public static List<NewWork.MapCreaterTypeNode> **MapCreaterCollection**	获取注册到程序中的获取 MapCreaterTypeNode 列表
private static Dictionary<string, NewWork.MapCreaterTypeNode> ***_mapCreaterTypeMayToCollection***	【私有】存储 MapCreaterTypeNode 与其 Type 的关联规则，方便根据 Type 查找 MapCreaterTypeNode
private static Dictionary<string, WorldCreaterStudioCore.MapCreater.MapCreaterFactory> ***programSetToCreaterFactory***	【私有】存储 Creater 与其 ProgramSet 的对应规则，方便根据 ProgramSet 查找 Creater

b) 方法

声明	描述
public static void **RegisterACreaterFactory** (WorldCreaterStudio_Core.MapCreater.MapCreaterFactory createrFactory)	注册一个 CreaterFactory createrFactory：要注册的 CreaterFactory
[DEL]public static WorldCreaterStudio_Core.MapCreater.MapCreaterFactory **GetACreaterFactory** (string programSet)	【已删除】根据 programSet 获取一个 CreaterFactory programSet：所根据的 programSet 字符串

3. [static]Resouses/Icons

提供所有用到的的图标资源。

a) 属性

名称	描述
`public static BitmapSource DarkIconPro`	暗色主题 工程图标
`public static BitmapSource DarkIconWork`	暗色主题 工作图标
`public static BitmapSource DarkIconRes`	暗色主题 资源图标
`public static BitmapSource DarkIconResLib`	暗色主题 资源库图标
`public static BitmapSource DarkIconBackEndWork`	暗色主题 后端工作图标

4. [WPF]Resouses/ControlTemplates

[WPF]Resouses/Theme

MiRaIUIProject 模块提供的内容

5. [WPF]Windows/MainWindow

主功能窗体

- 继承：Window

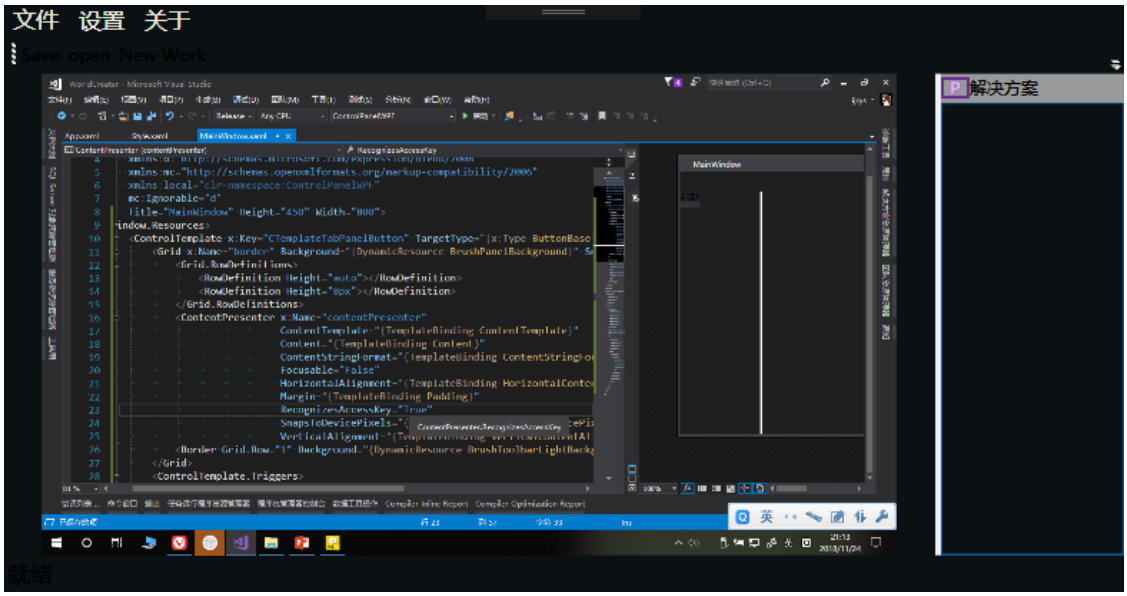
a) UI 设计

- **原型设计图**

<table>
<tr><td colspan="2">Menu</td></tr>
<tr><td>功能面板

图片资源预览
3D模型预览
设置面板</td><td>工程树</td></tr>
<tr><td colspan="2">状态栏</td></tr>
</table>

- **最终产品图**



b) 委托和事件

名称	描述
`public delegate void ProjectChangedEventType (object sender, Project value);`	属性 Project 改变的 Event 基本类 sender：触发改变的对象 value：新的 Project
`public event ProjectChangedEventType ProjectChanged;`	属性 Project 改变后触发的事件

c) 字段和属性

名称	描述
`private Project _project`	【私有|字段】保存当前工程对象
`public Project Project`	获取和设置当前工程，并在更改后执行相应事件（**ProjectChanged**）
`private FrameworkElement _showingFunctionPanel`	**[功能面板相关]**表示正在展示的面板

d) 方法

名称	描述
`private void CommandNewWorkExecuted (object sender, ExecutedRoutedEventArgs e)`	命令 **NewWork** 执行的方法 sender：调用者 e：参数
`private void CommandNewProjectExecuted (object sender, ExecutedRoutedEventArgs e)`	命令 **NewProject** 执行的方法 sender：调用者 e：参数
`private void ProjectChangedFunction(Project value)`	更改属性 **Project** 后必然执行的函数，用于更新前端的数据绑定 Value：新的 Project
`private void ShowFunctionPanel (object dataprovider, FrameworkElement panel)`	**[功能面板相关]**展示一个功能面板
`private void Tree_Project_Item_DoubleClick (object sender, MouseButtonEventArgs e)`	**[功能面板相关]**Tree_Project_Item 的双击时间

`private void RegShowPanel()`	**[功能面板相关]**向 Core 注册展示板

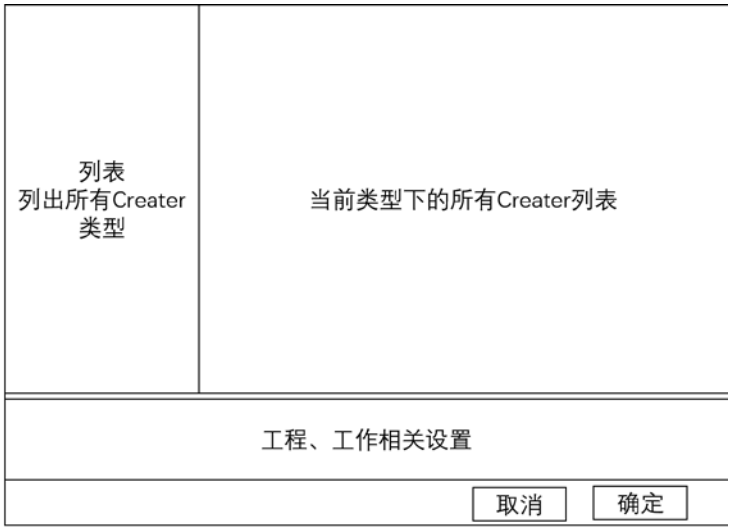
6. [WPF]Windows/NewProject

新建工程、工作窗体 用于选择新建的工程、工作的类型，确定名称和存储位置等

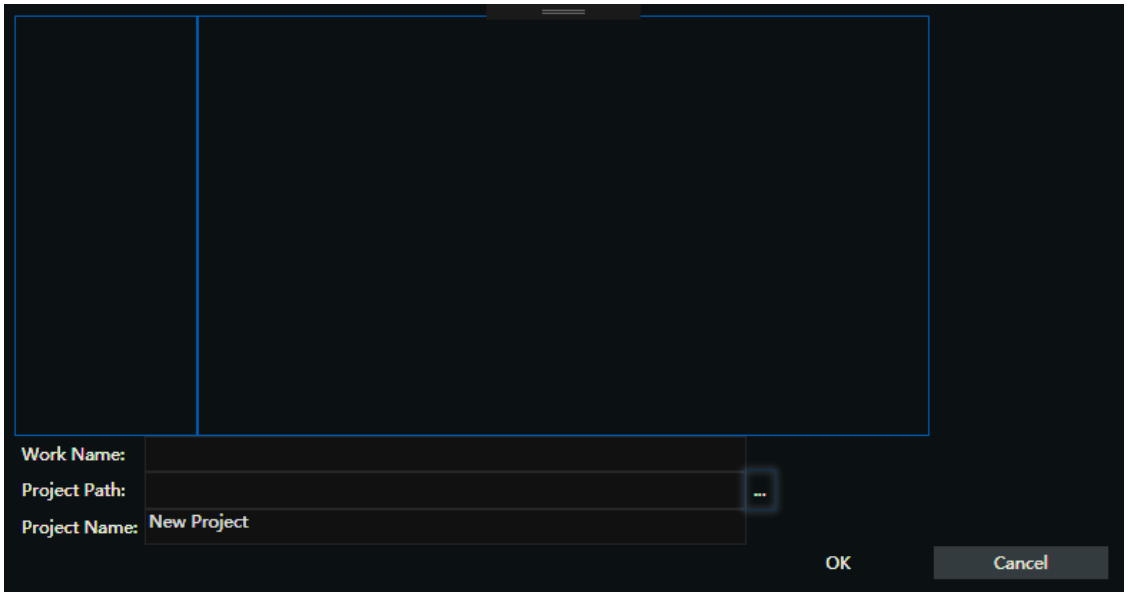
- 继承：Window

a) UI 设计

- 原型设计图



- 最终产品图



b) 字段和属性

名称	描述
private Project ***Project***	【私有】用于保存新创建的工程
private Work ***Work***	【私有】用于保存新创建的工作
private bool ***CreateNewProject***	【私有】
public DialogResult **WindowResult**	用于调用者获取用户最终选择的结果

c) 方法

名称	描述
public new void **Show**()	重写了父类的 Show 方法，自动调用 ShowDialog() **不建议使用，请合理选择使用 GetNewProject() 与 GetNewWork()**
public new void **ShowDialog**()	重写了父类的 ShowDialog 方法，用于以独占方式弹出窗体 **不建议使用，请合理选择使用 GetNewProject() 与 GetNewWork()**
private void **ShowDialog** (Project project)	用于以独占方式弹出窗体，并记录传入的 Project 对象。用于创建一个新工作时确定工作所属的工程 Project：所属的工程
public Project **GetNewProject**()	获取一个新的工程，会根据需要自动弹出窗体
public Work **GetNewWork** (Project project)	在已有的工程中创建一个新工作，会根据需要自动弹出窗体。 project：目标工程
private void ***BtnDirSelect_Click*** *(object sender, RoutedEventArgs e)*	前端按钮 **BtnDirSelect** 点击事件
private void ***BtnOk_Click*** *(object sender, RoutedEventArgs e)*	前端按钮 **BtnOk** 点击事件
private void ***BtnCancel_Click*** *(object sender, RoutedEventArgs e)*	前端按钮 **BtnCancel** 点击事件
private void ***ListCreaterTypeSelectionChanged*** *(object sender,*	前端列表 **List_CreaterType** 选择项目改变事件

SelectionChangedEventArgs e)	

十一、 WorldCreaterStudio_Core 模块功能细则

1.子命名空间或逻辑文件夹功能描述

- ListNode

逻辑文件夹，内部元素直接存在于 WorldCreaterStudio_Core 命名空间下。 逻辑节点类型，提供主功能节点和逻辑节点接口。

- MapCreater

Creater 相关基类的定义。

- Resouses

资源节点和资源管理管理节点的定义。

- StoreRoom

应用储藏室，存储各种注册到应用中的数据。

- Tools

各种工具类。

2. [I]ListNode/IWorkLogicNodeAble

表示在程序中的逻辑节点，用以搭建层级逻辑关系。

a) 字段和属性

名称	描述
Work **Work**	获取节点所属的工作
FrameworkElement **ShowPanel**	节点可展示的面板
string **NodeName**	节点在工程树形图中的展示名称
ImageSource **Icon**	节点在工程树形图中的展示图标
ObservableCollection<IWorkLogicNodeAble> **Childrens**	节点的子节点

b) 方法

名称	描述

`XmlElement XmlNode (XmlDocument xmlDocument)`	获取节点的 XML 节点

3. ListNode/Project

表示一个工程

对于工程文件的存储信息详见**工程文件格式及存储结构**=>**工程文件格式**

- 继承：IWorkLogicNodeAble

a) 字段和属性

名称	描述
`[IWorkLogicNodeAble]public Work Work`	获取节点所属的工作
`[IWorkLogicNodeAble]public FrameworkElement ShowPanel`	节点可展示的面板
`[IWorkLogicNodeAble]public string NodeName`	节点在工程树形图中的展示名称
`[IWorkLogicNodeAble]public ImageSource Icon`	节点在工程树形图中的展示图标
`[IWorkLogicNodeAble]public ObservableCollection<IWorkLogicNodeAble> Childrens`	节点的子节点
`public Guid Guid`	工程的 GUID
`public DirectoryInfo ProjectDirectionary`	工程所在文件夹信息
`public FileInfo ProjectFile`	工程的文件信息
`public bool Changed`	工程是否发生了更改

b) 构造函数和静态获取方法

名称	描述
`private Project (string projectPath, string filename, string proName)`	构造函数 projectPath：工程文件夹路径 filename：工程文件名 proName：工程名
`public static Project OpenProject`	打开一个存在于存储器上的工程

(string projectPath, string filename)	projectPath：工程文件夹路径 filename：工程文件名
public static Project **NewProject** (string projectPath, string filename, string projectName)	在存储器上新建一个工程 projectPath：工程文件夹路径 filename：工程文件名 projectName：工程名

c) 方法

名称	描述
[IWorkLogicNodeAble]public XmlElement **XmlNode** (XmlDocument xmlDocument)	获取节点的 XML 存储节点
public void **Save** (bool saveEvenUnchanged = false)	保存工程到本地文件 saveEvenUnchanged：指示在工程没有发生更改时是否强制保存。false：不强制保存,true：强制保存。
public Work **NewWork** (string workPath, string filename, string workName)	在工程内新建一个工作 workPath：工作所在目录 filename：工作文件名 workName：工作名

4. ListNode/Work

表示一个工作

- 继承：IWorkLogicNodeAble, InotifyPropertyChanged

a) 委托和事件

名称	描述
[InotifyPropertyChanged]public event PropertyChangedEventHandler **PropertyChanged**;	接口 InotifyPropertyChanged 的实现，用以在属性发生更改时通知绑定的对象。

b) 属性和字段

名称	描述
DirectoryInfo ***_workDirectionary***	记录工作目录
DirectoryInfo ***_workResousesDirectionary***	记录工作的图片资源存储目录
FileInfo ***_workFile***	记录工作文件
Guid ***_guid***	工作的 GUID
bool ***_changed***	记录工作是否发生了更改
ImageSource ***_icon***	记录节点的图标
Config ***_config***	已移除
[IWorkLogicNodeAble]Work **IWorkLogicNodeAble.Work** => this	获取节点所在的工作，固定为当前对象
[IWorkLogicNodeAble]public FrameworkElement **ShowPanel**	获取节点的展示面板
[IWorkLogicNodeAble]public string **NodeName**	获取节点的展示名称
[IWorkLogicNodeAble]public ImageSource **Icon**	获取节点的图标
public Resouses.ImageResourceManager **Images**	获取工作的图片资源管理器
public FrontEndFactory **FrontEndNodes**	获取工作的前端节点
public BackEndFactory **BackEndNodes**	获取工作的后端节点
[IWorkLogicNodeAble]public ObservableCollection<IWorkLogicNodeAble> **Childrens**	获取工作的所有子节点
public Guid **Guid**	获取工作的 GUID

c) 构造函数和静态获取方法

名称	描述
private ***Work*** *(string workPath, string filename, string workName)*	【私有】构造函数
public static Work **NewWork** (string workPath, string filename, string workName)	新建一个工作 workPath：工作所在文件夹 filename：工作文件名

	workName：工作名
public static Work **OpenWork** (string workPath, string filename, string workName)	打开一个工作 workPath：工作所在文件夹 filename：工作文件名 workName：工作名

d) 方法

名称	描述
public XmlElement **XmlNode** (XmlDocument xmlDocument)	获取节点的 XML 节点，用于工程文件的保存
public void **Save** (bool saveEvenUnchanged = false)	保存工作 saveEvenUnchanged：确定是否强制保存，false：在工作没有发生更改时不保存，true：在工作没有发生更改时依旧重新保存。
private static BitmapImage ***GetBitmapFromFile*** *(string path)*	工具方法，打开指定路径的图片

5. ListNode/FrontEndFactory

前端工厂管理节点。

- 继承：IWorkLogicNodeAble

a) 委托和事件

名称	描述

b) 属性和字段

名称	描述
public Work **Work**	
public FrameworkElement **ShowPanel**	
public FrameworkElement **ConfigurationPanel**	

public string **NodeName** => "前端工厂"	
public ImageSource **Icon**	
public MapCreater.Configuration **Configuration**	
public MapCreater.MapCreater **Creater**	
public Resouses.ImageResourceReferenceManager **ImageResourceReferenceManager**	
public ObservableCollection<IWorkLogicNodeAble> **Childrens**	

c) 构造函数和静态获取方法

名称	描述

d) 方法

名称	描述