该项目已使用MIT开源协议（https://opensource.org/licenses/mit-license.php）开源，软件开源地址：<https://github.com/ijrys/WorldCreater>

# 文档规范

## 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)

## 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

# 软件简介

## 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。

## 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。

## 开发技术及软件运行目标环境

### 开发技术

* C# （C#7.0）
* WPF （WPF4.5）
* .NET Framework （.NET Framework 4.7.2）

### 开发环境

* Visual Studio 2017 Community

### 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

## 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）

## 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>

# 软件逻辑设计

## 工程管理逻辑

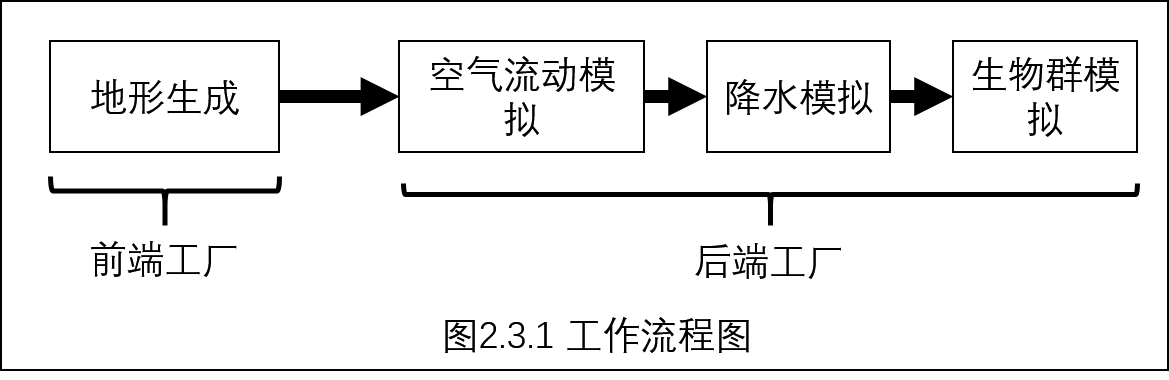
软件工程逻辑为树形分层逻辑，根节点为工程节点（Project）。工程节点下分为多个工作节点（Work），每一个工作节点对应一个地形从生成到最终的环境模拟演变结束的整个过程。在同一工程内的各个工作是相互关联而又相互独立的，他们既可以是同一个项目的不同部分的地形，也可以通过更强大的合成器将多个地形合成到一起，实现更加复杂的效果控制。

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。

## 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：



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。

# 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。

## 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。

### 工程文件

工程文件后缀名为mriwcpro（MiRaI 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>

### 工作文件

工作文件后缀名为mriwcw（MiRaI 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>

## 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  
|-……

## 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】

# 模块简介及功能概览

## WorldCreaterStudio

### 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。

## WorldCreaterStudio\_Core

### 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。

## WorldCreaterStudio\_Resouses

### 模块功能

存储 WorldCreaterStudio 所使用到的资源。

## [FE]FrontEndFactorys.RandomTend

### 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。

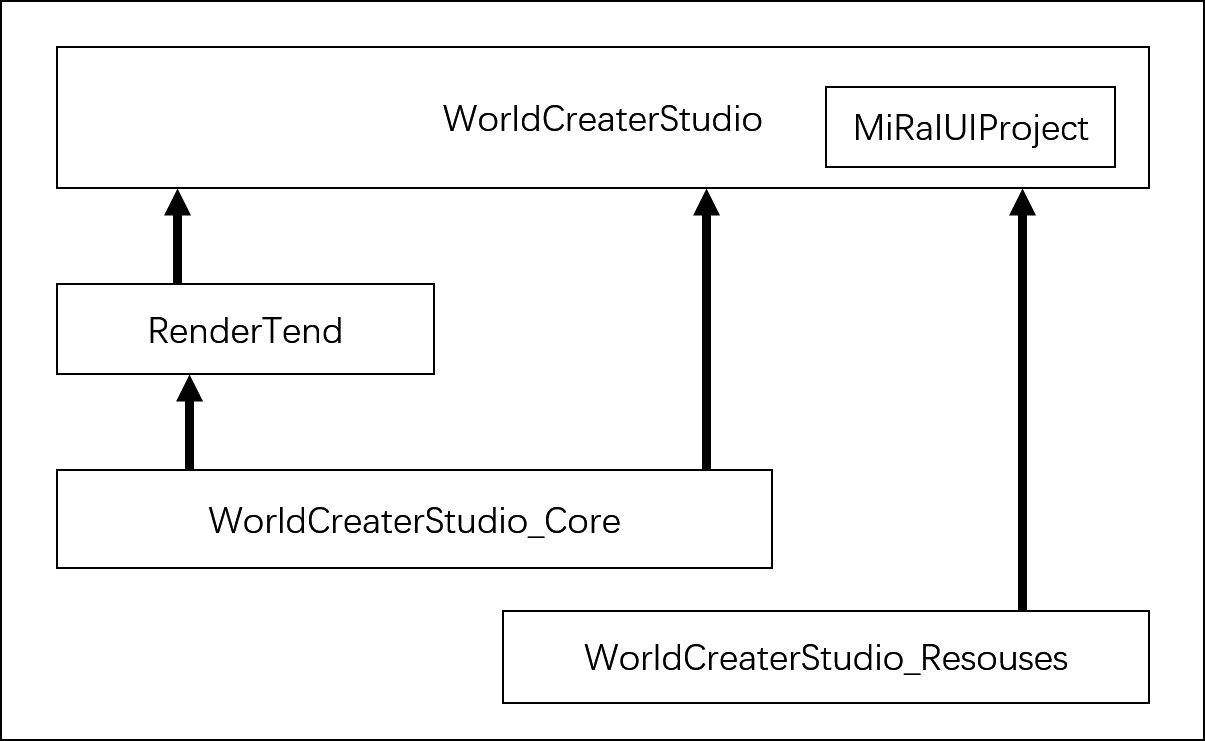
## MiRaIUIProject

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>

### 模块功能

提供WPF程序的的UI样式。

## 模块依赖关系图



# WorldCreaterStudio模块功能细则

## [static]Commands

Studio使用到的自定义命令

### 构造函数

static Commands()

### 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 |

## [static]Resouses/StoreRoom

### 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 |

### 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 |
| public static WorldCreaterStudio\_Core.MapCreater.MapCreaterFactory GetACreaterFactory  (string programSet) | 根据programSet获取一个CreaterFactory  programSet：所根据的programSet字符串 |

## [static]Resouses/Icons

提供所有用到的的图标资源。

### 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 |

## [WPF]Resouses/ControlTemplates <br> [WPF]Resouses/Theme

MiRaIUIProject模块提供的内容

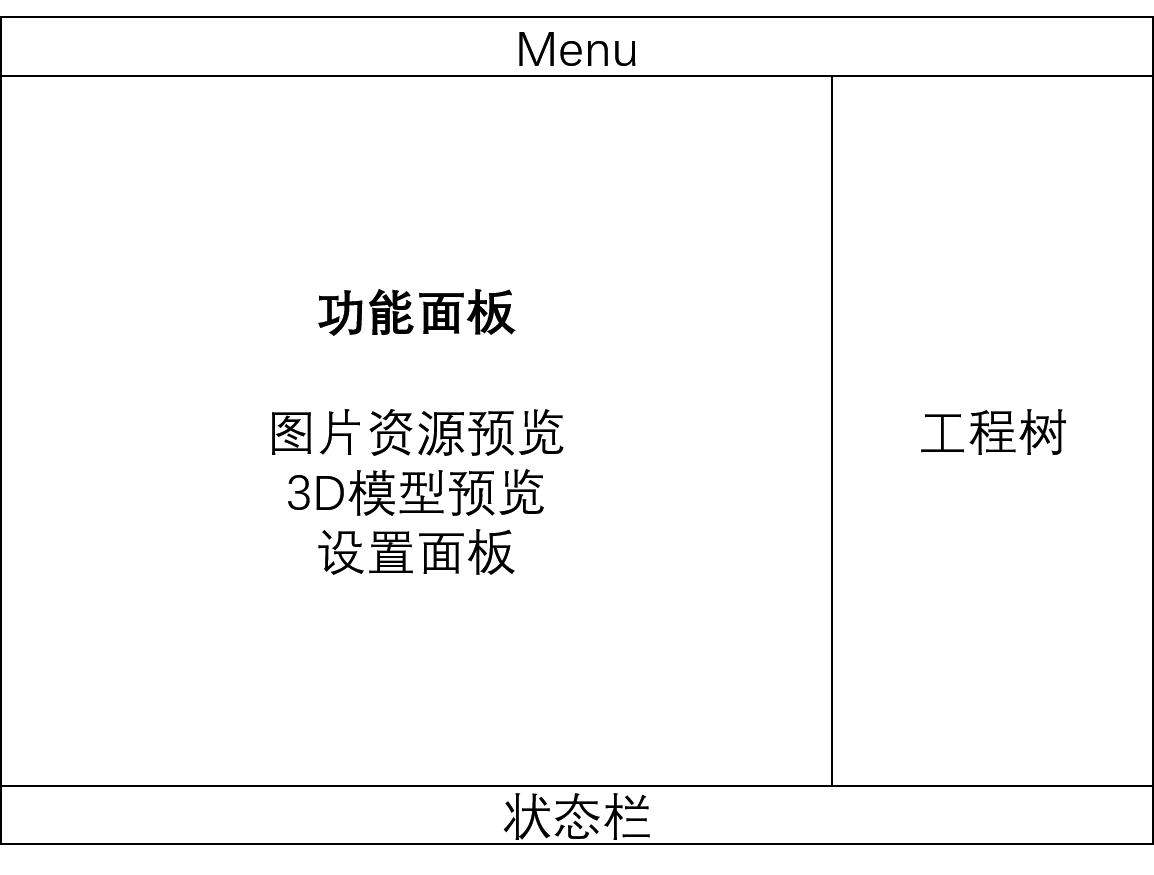
## [WPF]Windows/MainWindow

主功能窗体

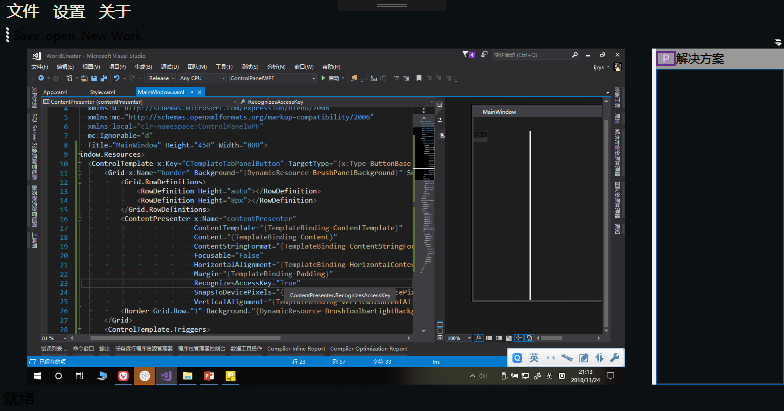
* 继承：Window

### UI设计

#### 原型设计图



#### 最终产品图



### 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 |

### 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） |

### 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 |

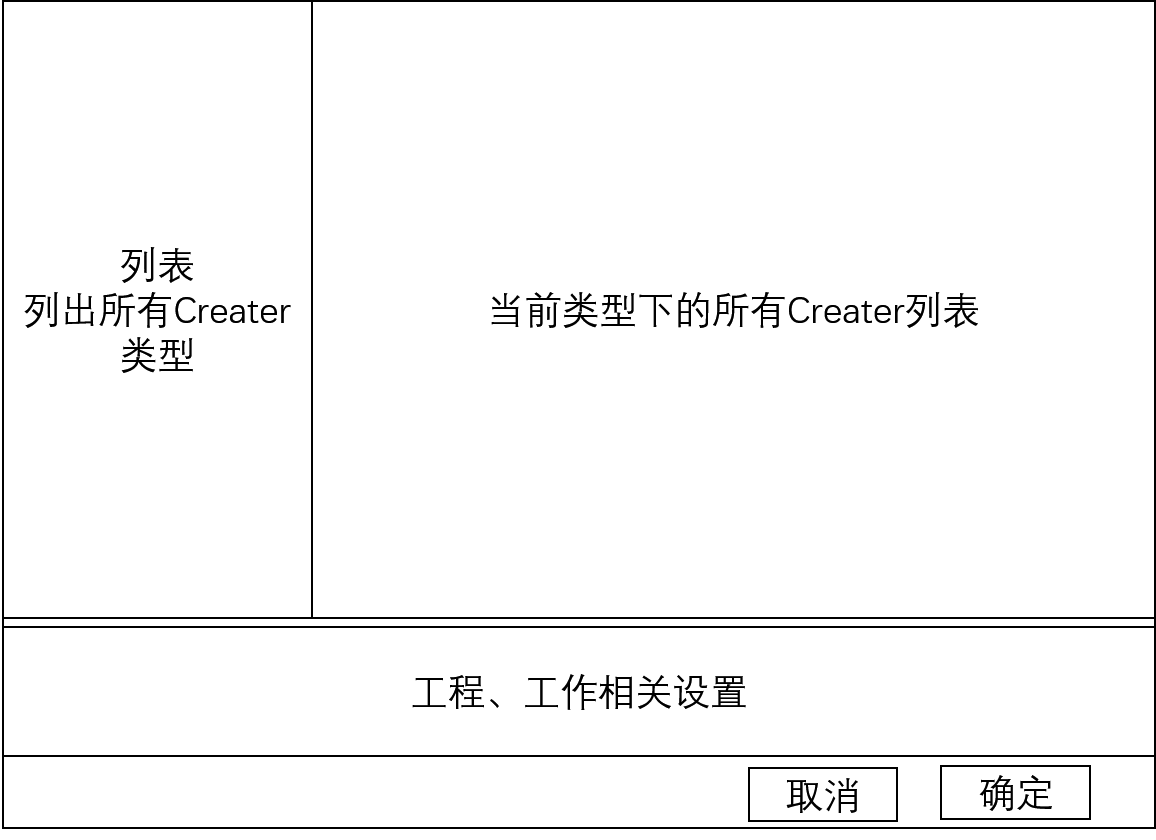
## [WPF]Windows/NewProject

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

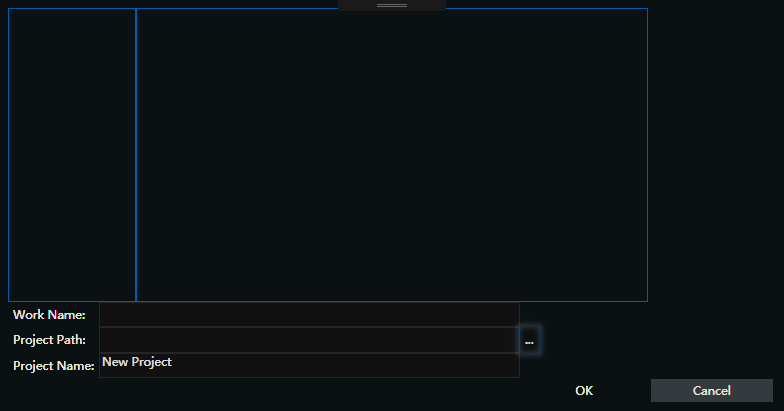
* 继承：Window

### UI设计

* 原型设计图



* 最终产品图



### 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 |

### 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, SelectionChangedEventArgs e) | 前端列表List\_CreaterType选择项目改变事件 |

# WorldCreaterStudio\_Core模块功能细则

## 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。

## [I]ListNode/IWorkLogicNodeAble

表示在程序中的逻辑节点，用以搭建层级逻辑关系。

### 字段和属性

|  |  |
| --- | --- |
| 名称 | 描述 |
| Work Work | 获取节点所属的工作 |
| UIElement 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 |

### 方法

|  |  |
| --- | --- |
| 名称 | 描述 |
| XmlElement XmlNode  (XmlDocument xmlDocument) | 获取节点的XML节点 |

## 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

### 字段和属性

|  |  |
| --- | --- |
| 名称 | 描述 |
| [IWorkLogicNodeAble]public Work Work | 获取节点所属的工作 |
| [IWorkLogicNodeAble]public UIElement 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 |

### 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  (string projectPath, string filename) | 打开一个存在于存储器上的工程  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 |

### 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 |

## ListNode/Work

表示一个工作

* 继承：IWorkLogicNodeAble, InotifyPropertyChanged

### 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。 |

### 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 UIElement 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 |

### 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 |

### 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 |

## ListNode/FrontEndFactory

前端工厂管理节点。

* 继承：IWorkLogicNodeAble

### 委托和事件

|  |  |
| --- | --- |
| 名称 | 描述 |

### 属性和字段

|  |  |
| --- | --- |
| 名称 | 描述 |
| public Work Work |  |
| public UIElement ShowPanel |  |
| public UIElement 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 |  |
|  |  |
|  |  |

### 构造函数和静态获取方法

|  |  |
| --- | --- |
| 名称 | 描述 |

### 方法

|  |  |
| --- | --- |
| 名称 | 描述 |