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

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、软件需求说明

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）

b) 开发环境

- .NET Framework （.NET Framework 4.7.2）
- Visual Studio 2017 Community 及兼容产品

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 及 WPF 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

《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. 功能使用操作流程及方式

用户通过鼠标、键盘进行输入，并通过 ui 界面获得操作输出。

使用统一的 ui 平台对算法功能进行整合，用户通过控制 ui 平台来实现对算法的调用和数据的获取。

各算法按照算法目的分为五大部分，对应于工作的五个步骤。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：

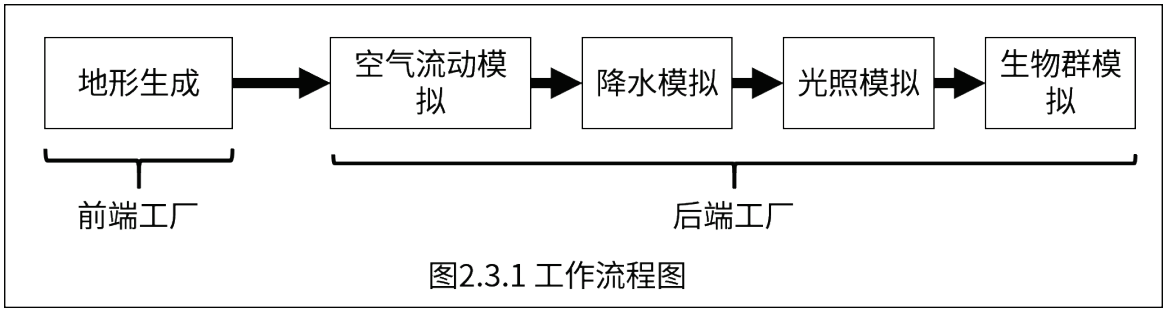


图2.3.1 工作流程图

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

四、软件数据存储结构及节点逻辑结构定义

1. 工程文件存储格式

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-47e4-1234-85cf72f19deb" name="DemoPro">
  <work guid="46a1d881-f42d-4a7e-4567-0123456789ab"
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>
        <Resault name="" value="[运算结果的存储文件]" imgrefkey="[用于展示的图片资源 key]" />
    </FrontEndFactory>

    <BackEndFactory>
        <AMNode creater="[后端工厂的 ProgramSet]" state="[节点的状态]">
            <Resault dataName="[运算结果的存储文件]" imgrefkey="[用于展示的图片资源 key]" />
            <Config>
                [该节点为相应类产生其具体内容]
            </Config>
        </AMNode>
    </BackEndFactory>
</work>
```

如

```
<work guid="54fe1651-5c29-45c5-9a31-379e4b18b100">
    <images>
        <image key="FE.RandomValue" file="eec6e1f3e0a64d4c97962b64bf67.png" description="前端工厂的随机值图" />
        <image key="FE.HeightValue" file="36089b82aacb4afa97544c5d5d8c.png" description="前端工厂高度值图" />
        <image key="BE.AM.Map" file="684083b24a3a4b08ab8aa9c6da1c.png" description="AtmosphericMotion Visual Map" />
    </images>
    <FrontEndFactory creater="MiRaI.RandomTend.RandomTend|0.1">
        <setting>
            <add key="blockSize" value="5" />
            <add key="wbn" value="1" />
            <add key="hbn" value="1" />
            <add key="seed" value="2019" />
        </setting>
        <Resault name="" value="HeightValue" imgrefkey="FE.HeightValue" />
    </FrontEndFactory>
    <BackEndFactory>
        <AMNode creater="MiRaI.BE.AM.SV|0.1" state="ok">
            <Resault dataName="AtmosphericMotion Visual Map" imgrefkey="BE.AM.Map" />
            <Config power="76" dir="NW" />
        </AMNode>
        <RMNode creater="MiRaI.BE.RM.SV|0.1" state="ok">
```

```
                <Resault dataName="RainfallMotion Visual Map"
imgrefkey="BE.RM.RIMap" riimgrefkey="BE.RM.RIMap"
atimgrefkey="BE.RM.ATMap" />
                <Config ri="4020" sl="0" />
            </RMNode>
            <SINode creater="" state="ready" />
            <BINode creater="" state="unable" />
        </BackEndFactory>
</work>
```

2. 工程文件的物理存储逻辑

```
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
| | |-……
| |-HeightValue [高度数据]
| |-AtmosphericMotion Visual Map [空气流动数据]
| |-RainfallMotion Visual Map [降数据]
| |- [其他产生的过程数据]
|-Geo2
|-……
```

3. 数据图结构定义

数据均为矩阵，内部操作时使用二维数组实现存储。

数组第一索引为行号，第二索引为列号，如下图(4.3.1)所示

0行	Map[0,1]	Map[0,2]	Map[0,3]	Map[0,4]	Map[0,5]	……
1行	Map[1,1]	Map[1,2]	Map[1,3]	Map[1,4]	Map[1,5]	……
2行	Map[2,1]	Map[2,2]	Map[2,3]	Map[2,4]	Map[2,5]	……
3行	Map[3,1]	Map[3,2]	Map[3,3]	Map[3,4]	Map[3,5]	……

图 4.2.1

数据存储时先存储数据图宽、高(int_32)，再按行存储其中每一列数据。伪代码如下：

```
Write (map.Width);
```

```
Write (map.Height);

for (int h = 0; h < map.Height; h ++) {
      for (int w = 0; w < map.Width; w ++) {
            Write (map[h, w]);
      }
}
```

数据图按 0 行在上，0 列在左处理。方位上按上方为北，左方为西处理。

为了兼容各个流程，对于图的数据意义和长宽进行限制，规定如下：

● 生成的地形边长应均为 $n * 2^k$，n、k 均为正整数，且 k>=4。

● 高度图中每个点代表的是一个正方形块的角的高度，则高度图的数据数量为(a + 1) * (b + 1)，a、b 分别为设定目标地图的长、宽。

● 后端工厂对数据的处理是基于块的，记录的是块的数据，则后端工厂生成的数据量应为 a * b，a、b 分别为设定目标地图的长、宽。

4. 工程的展示逻辑

软件中对用户展示一个逻辑树，来展示整个工程。用户通过对逻辑树的相应节点进行操作从而控制整个工程。

```
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、运算结果数据类型定义及限定

● 涉及到的类型定义

符号	描述	在 c#中的符号
int_32	有符号的 32 位整形	int
unsigned_int_8	无符号的 8 位整形	byte

1. 前端工厂

前端工厂产生数据为 int_32 类型的二维数组。

每一个数据指示相应点的高度。为了后续工作的需要，数据范围应为[-2^{29},2^{29}-1]，即[-536870912, 53687091]

2. 后端工厂 AtmosphericMotion

空气流动模拟生成的数据应表示两个属性：流动的方向和速度

为了简化需求，速度为 unsigned_int_8，与真实风力级别对应关系为真实风力=value/20

流动方向简化为 9 个方向，中心表示无风，枚举值为 0，其他各方向以向西北方开始顺时针按序排列。枚举值具体见下图：

↖（1）	↑（2）	↗（3）
←（8）	●（0）	→（4）
↙（7）	↓（6）	↘（5）

如空气向西北方流动（东南风），枚举值为 1；空气向南方流动（北风），枚举值为 6。

3. 后端工厂 RainfallMotion

降水模拟生成的数据应表示三个属性：降水强度、水深、水域类型。

降水强度为每单位时间降下 1 个全局高度单位的水，类型为 int_32，但数据范围应在[0,2^20]甚至更小范围内的合理值。

水深表示当前区域的水体水深，类型为 int_32，单位与全局同单位。

水域类型表示当前区域的水体类型，枚举值对应关系如下表：

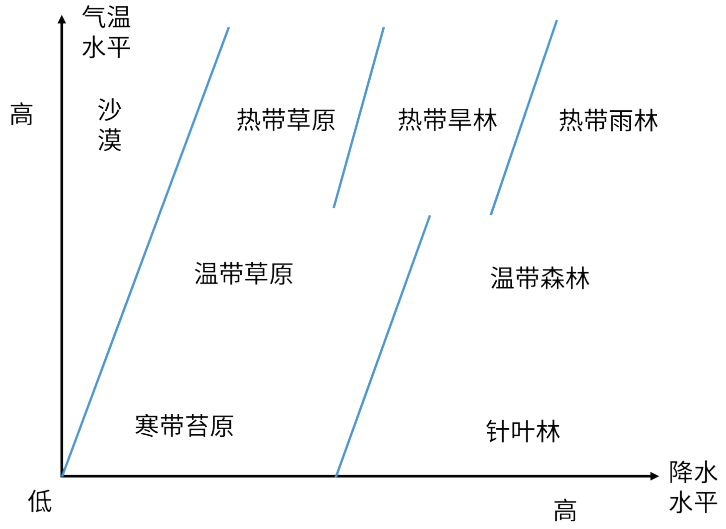
中文描述	枚举名	枚举值
陆地	land	0
江，河	river	1
湖	lake	2
海，洋	sea	3
沼泽，湿地	marsh	4

4. 后端工厂 SolarIlluminance

光照模拟输出为每个区域的光照辐射强度。数据类型为 unsigned_int_8，数据范围为[0,255]。

5. 后端工厂 Biomes

生物群模拟主要依靠之前的数据对区域内生物类型进行判别，按照降水和光照条件，可大致分为以下八种陆地生物群，同时水域生物群可分为淡水水域和咸水水域两种。



每种枚举值的关系如下表：

中文描述	枚举名	枚举值
淡水水域	FreshWater	0b1_000_0000

咸水水域	SaltWater	0b1_001_0000
沙漠	Desert	0b0_000_0000
炎热草原	TropicalSavanna	0b0_010_0001
炎热旱林	ThornForest	0b0_010_0010
炎热雨林	TropicalRainforest	0b0_010_0011
温和草原	TemperateGrassy	0b0_001_0001
温和森林	TemperateForest	0b0_001_0010
冰雪苔原	Tundra	0b0_000_0001
针叶林	ConiferousForest	0b0_000_0010

六、软件模块划分及关系

1. 模块功能简介

a) WorldCreaterStudio

软件 UI 提供模块，将各模块功能整合提供完整的操作流程，是用户与 WorldCreaterStudio_Core 模块的桥梁。

b) WorldCreaterStudio_Core

定义整个软件平台的核心算法、核心类，并为算法功能模块定义基本的开发接口，为整个平台的基础和各个组件联通的纽带。

c) WorldCreaterStudio_Resouses

存储 WorldCreaterStudio 所使用到的资源。

d) MiRaIUIProject

该模块为 WPF 框架的主题项目，基于 WPF 控件定义了一套视觉效果，现已独立为单独的项目，并已通过 MIT 协议开源。开原地址：https://github.com/ijrys/MiRaIUIProject

该模块为 WorldCreaterStudio 模块提供 WPF 程序的的 UI 样式。

e) [FE]FrontEndFactorys.RandomTend

前端工厂相关算法的实现。随机趋势化算法核心代码。

f) [BE]MiRaI.BE.AM.SingleValue

后端工厂空气运动模拟相关算法的实现。提供一个快速的、简单的实现，方便必要的需求和软件整体框架测试。

g) [BE]MiRaI.BE.RM.SingleValue

后端工厂降水运动模拟相关算法的实现。提供一个快速的、简单的实现，方便必要的需求和软件整体框架测试。

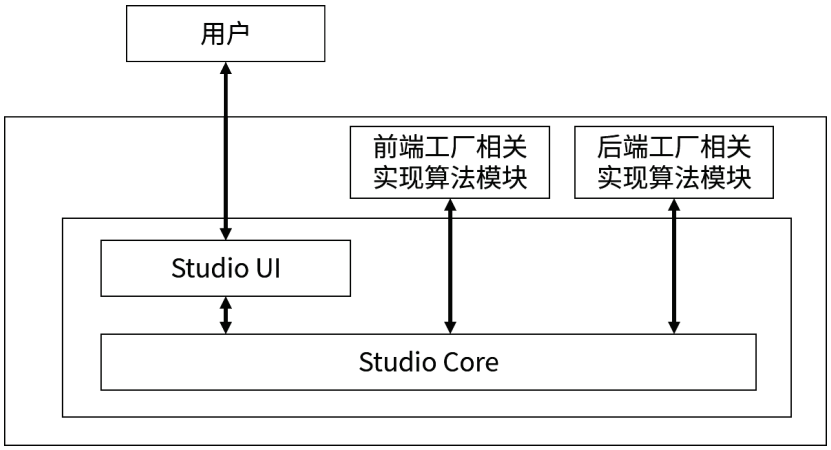
h) [BE]MiRaI.BE.SI.QuickCalculating

后端工厂光照强度模拟相关算法的实现。提供一个快速的、简单的实现，方便必要的需求和软件整体框架测试。

i) [BE]MiRaI.BE.BI.QuickCalc

后端工厂生物群模拟相关算法的实现。提供一个快速的、简单的实现，方便必要的需求和软件整体框架测试。

2. 模块依赖关系图



七、代码规范规约

1. 命名风格

● 程序集、命名空间、类名、方法名、属性名使用 UpperCamelCase 风格，如 FrontEndFactory

● 参数名、成员变量、局部变量使用 lowerCamelCase 风格，如 inputName

● 私有字段使用下划线开始，如_nodeName

● 异常类命名使用 Exception 结尾

● 接口以 I 开头，以 Able 结尾，如 IwriteToFileAble

2. 代码格式

● 大括号的使用约定。

如果是大括号内为空，则简洁地写成{}即可，不需要换行；

如果是非空代码块则 左大括号前不换行，左大括号后换行。右大括号前换行。右大括号后还有 else 等代码则不换行；表示终止的右大括号后必须换行。

● 左右小括号和字符之间不出现空格。

● 缩进

级别缩进使用 tab，统一级别的代码块应保证前边有相同数量的 tab

内容缩进在使用 tab 实现级别缩进后按需使用空格进行内容对齐

● 换行

在一行代码过长时进行换行保证阅读方便。换行时应保证语句块不被拆分（如调用方法时不将方法名与参数列表拆在两行），调用操作符应跟在新行，其他操作符保持在旧行。

● 代码格式示例

```
public static void Function (int id,
                             string name,
                             string nickName) {
    if (id < 10 &&
    ····name != null) {
    //·do·something
    } else {
    →    nickName.DoFun1('·')
    →    ·····.DoFun2("||")
    →    ·····.DoFun3();
    }
```

```
}
```

八、WorldCreaterStudio 模块功能细则

1. [static]Commands

Studio 使用到的自定义命令

- 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

存储 Studio 需要用到的一些资源，为前端绑定提供便利。

3. [static]Resouses/Icons

提供所有用到的的图标资源。

资源存储在 WorldCreaterStudio_Resouses 模块中。

4. [WPF]Resouses/ControlTemplates

[WPF]Resouses/Theme

MiRaIUIProject 模块提供的内容

5. [WPF]Windows/MainWindow

主功能窗体

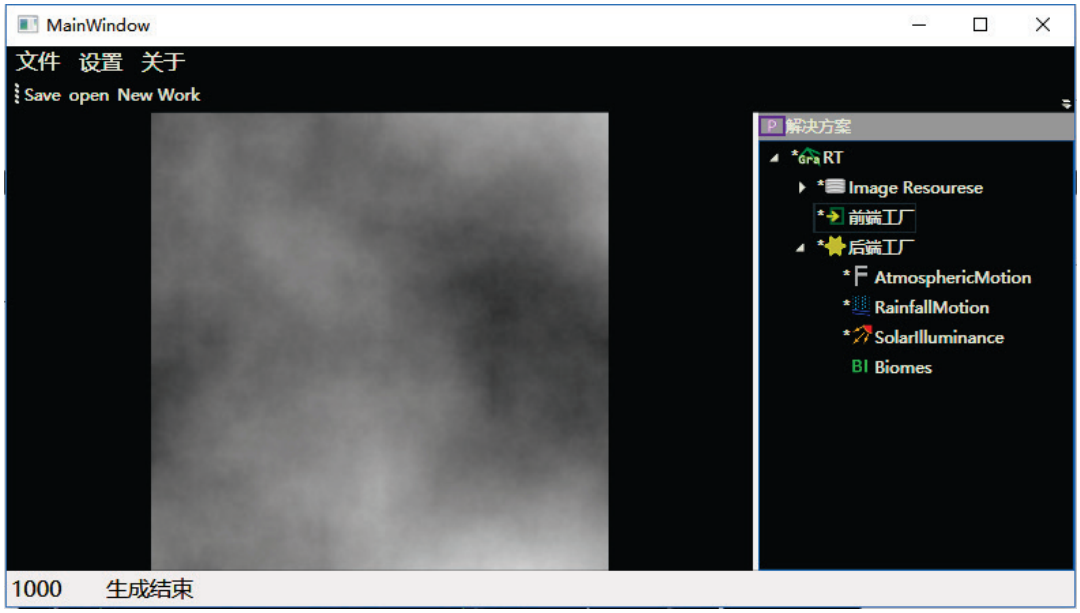
继承：Window

a) UI 设计

- 原型设计图

Menu	
功能面板 图片资源预览 3D模型预览 设置面板	工程树
状态栏	

- 最终产品图



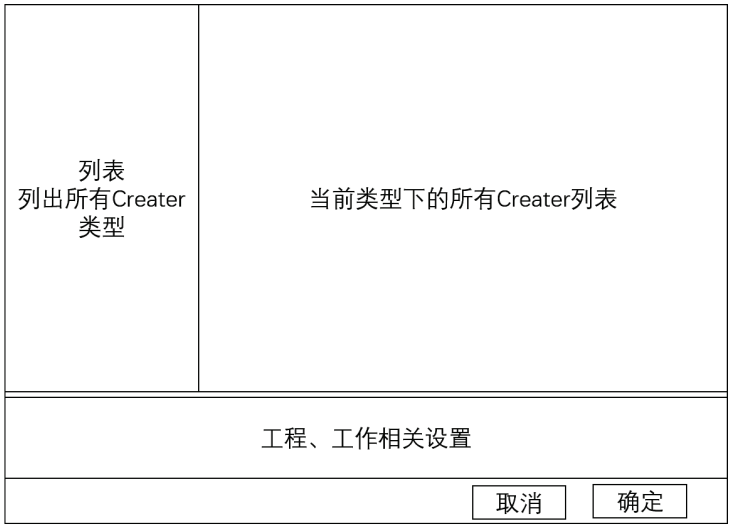
6. [WPF]Windows/NewProject

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

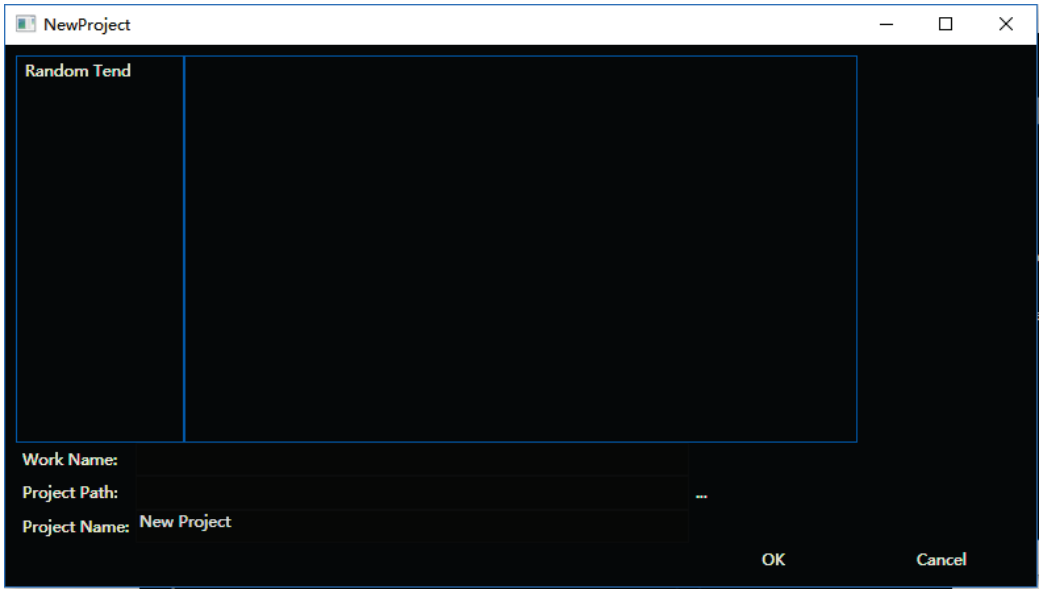
- 继承：Window

a) UI 设计

- 原型设计图



- 最终产品图



九、ValueToImage 模块功能细则

1. 关于常见图片格式的说明

图片均使用点阵图操作，资源存储时使用无损压缩的 png 格式。为减少内存占用，产生的数据内存使用量尽可能小，即每像素占用的比特数尽可能少。以下的格式指像素格式。

名称	像素比特数	图片类型	描述
BlackWhite	1	黑白	二值化图片，仅可表示黑、白两色

Gray2	2	灰度	灰度图像，随着每像素比特数增加可表示的中间色越多。
Gray4	4		
Gray8	8		
Index1	1	索引颜色	索引格式。可索引 2 种颜色。
Index2	2		索引格式。可索引 4 种颜色
Index4	4		索引格式。可索引 16 种颜色
Index8	8		索引格式。可索引 256 种颜色
BGR32	32	真彩色	按照 B、G、R、A 三通道存储的图片，每通道 256 种值。BGR32 会忽略 A 通道。因该软件需求不大，故不建议使用该种格式。
BGRA32	32	真彩色，带透明通道	

2. [static]ValueToGrayImage

将前端工厂生成的高度数据转换为可视的灰度图，输出为 Gray8 格式或是 Index8 格式的图片。

实现要求：通过设定 minValue、maxValue、minGray、maxGray 实现将 map 中的值映射到目标灰度值中。其中 map 中 value<=minValue 的映射为 minGray，value>=maxValue 的映射为 maxGray。

计算原理公式为：

$$\frac{\text{resGray} - \text{minGray}}{\text{maxGray} - \text{minGray}} = \frac{\text{height} - \text{minValue}}{\text{maxValue} - \text{minValue}}$$

得

```
resGray = minGray + (maxGray - minGray) * (height - minValue) / (maxValue - minValue)
```

3. [static]ValueToColorImage

将高度数据转换为可视的彩色图，用于不同高度的较精确的检查。

输出为 Index8 格式的图片。

数据计算方法见 ValueToGrayImage 的原理。

4. [static]AtmosphericMotionToImage

配合 AtmosphericMotion 的可视化值转换器。

数据分为若干边长为 16*16 的块，每个块内求平均值，展示每个块的平均值即可。

对每个块内的数据进行累加统计，得出 9 个方向的风力占比。按照对向相消的法则获得空气在两个正交的方向上的速度分量，进而算出块的风向和风力。

5. [static]RainfallMotionToImage

配合 RainfallMotion 的可视化值转换器。

因为 RainfallMotion 计算器生成的数据的三个分量在表示时并无相关性，故该部分需要分别实现对三个分量的可视化转换。

降雨强度的结果输出建议为 Gray8 格式、Index8 格式。

区域类型图输出建议为 Index4 格式。

水深图输出建议为 Gray8 格式、Index8 格式。

6. SolarIlluminance

配合 SolarIlluminance 的可视化值转换器。

输出建议为 Gray8 格式或是 Index8 格式。

7. BiomesToImage

配合 BiomesToImage 的可视化值转换器。

输出建议为 Index8 格式。

十、 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

● 基本算法：

使用用户设定的值填充所有区域，实现思路为双层 for 循环。

2. SingleValueConfig

记录 SingleValue 进行模拟所需要的配置数据

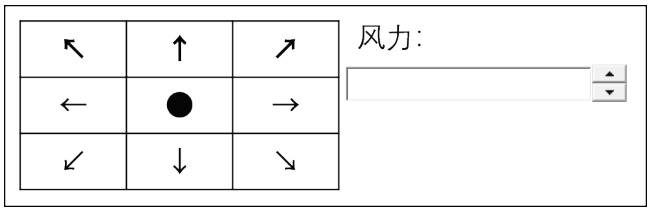
继承：IAtmosphericMotionConfigAble

3. SingleValueFactory

用于生成 SingleValue 和 SingleValueConfig 的工厂类。

● 配置面板

原型设计：



实现效果：

十一、 BackEndFactorys/RainfallMotion/MiRaI.BE.RM.SingleValue 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

● 基本算法

对于降水强度，按照用户设定填充所有区域

水深计算：若地形高度低于设定的水平面值，水深为水平面高度- 地形高度，否则水深为0。

根据水深设置水域类型。

2. SingleValueConfig

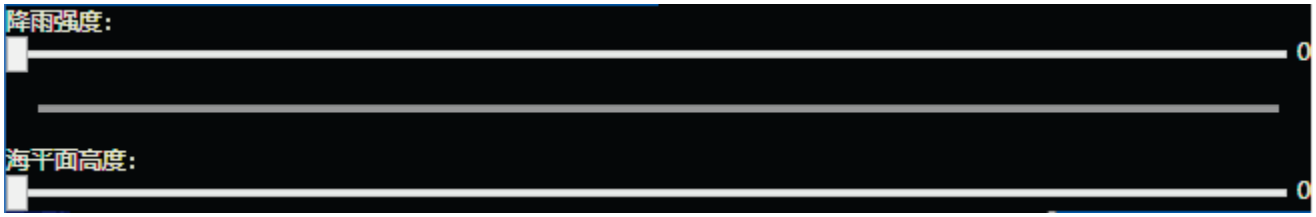
记录 SingleValue 进行模拟所需要的配置数据

继承：IRainfallMotionConfigAble

● 配置面板

原型设计：

实现效果：



3. SingleValueFactory

用于创建、获取相应的 Calculater 和 Config 的工厂类

继承：IRainfallMotionCalculaterFactoryAble

十二、BackEndFactorys/SolarIlluminance/MiRaI.BE.SI.Quick Calculating 模块功能细则

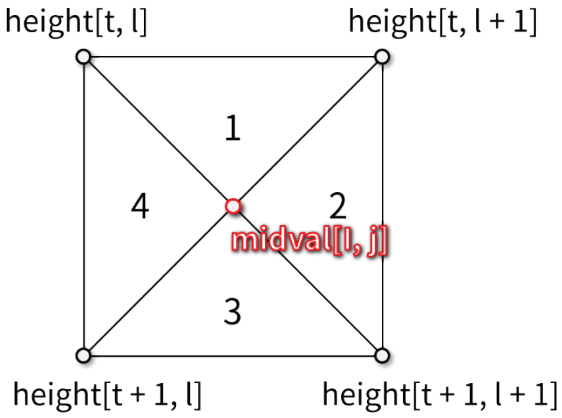
1. QuickCalculating

利用法线快速模拟光照强度模拟器。

继承：IsolarIlluminanceCalculaterAble

● 基本算法原理

对每个区域计算中点值（四角高度的平均值），将区域分为 4 块，分别计算四个区域的强度值，取强度值的平均值做最终结果。



每个区域的强度计算使用直线与平面夹角的计算方式，取夹角的 cos 值即可得到[0,1]之间的值，再乘 255 得出结果。

计算过程中使用 double 作为过程变量类型。

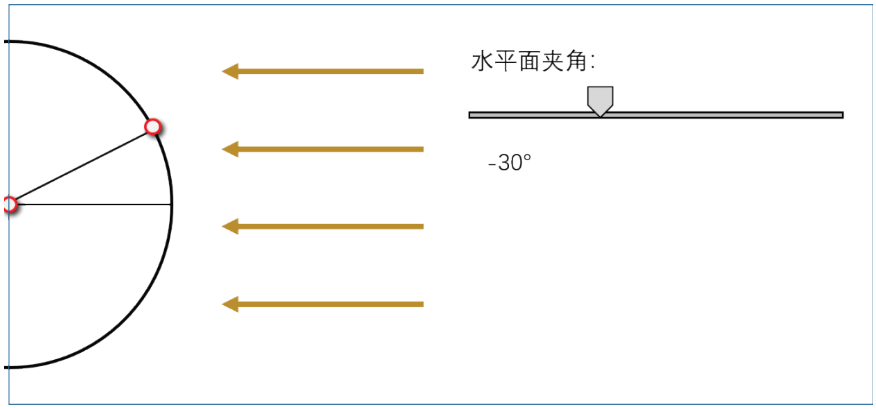
2. QuickCalculatingConfig

记录 QuickCalculating 进行模拟所需要的配置数据

继承：IsolarIlluminanceConfigAble

- 配置面板

原型设计：



3. QuickCalculatingFactory

用于创建、获取相应的 Calculater 和 Config 的工厂类

继承：IRainfallMotionCalculaterFactoryAble

十三、 BackEndFactorys/Biomes/MiRal.BE.BI.QuickCalc 模块功能细则

1. QuickCalc

根据已有快速模拟光照强度模拟器。

继承：IBiomesCalculaterAble

温度指数计算方式如下：

```
tempIndex = SIRes[i,j] – height[i,j] * scale
```

湿度指数计算方式如下：

```
RMRes[i, j].RainfallIntensity * scale - tempIndex
```

温度、降水与结果对应表：

温度指数	气候类型	湿度指数	生物群类型	枚举值
(-oo,10)	寒带	(-oo,0)	寒带沙漠	Desert
		[0,30)	冰雪苔原	Tundra
		[30,+oo)	针叶林	ConiferousForest
[10,25)	温带	(-oo,5)	沙漠	Desert
		[5,100)	温和草原	TemperateGrassy
		[100,+oo)	温和森林	TemperateForest
[25,+oo)	热带	(-oo,10)	沙漠	Desert
		[10,90)	炎热草原	TropicalSavanna
		[90,180)	炎热旱林	ThornForest
		[180,+oo)	雨林	TropicalRainforest

2. QuickCalcConfig

记录 QuickCalc 进行模拟所需要的配置数据

继承：IbiomesConfigAble

3. QuickCalcFactory

用于创建、获取相应的 Calculater 和 Config 的工厂类

继承：IbiomesCalculaterFactoryAble

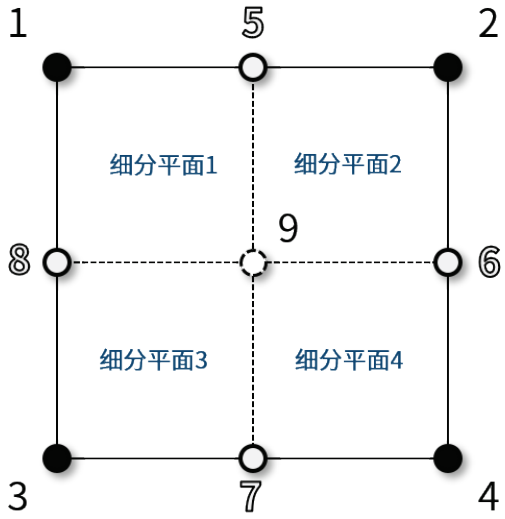
十四、FrontEndFactorys/RandomTend 模块功能细则

1. RandomTendCreater

随机趋势化地形生成算法

继承：MapCreater

● 算法基本原理



0. 如图，对区域的角点（点 1、2、3、4）进行随机，获得区域的大体高度趋势。

1. 对边上中点（点 5、6、7、8）进行高度值计算，方式为边上角点的平均值加随机值，即[5] = ([1] + [2]) / 2 + random，6、7、8 同理。

2. 对中心点利用于边上中点类似的的方式确定高度，公式为[9] = ([5] + [6] + [7] + [8]) / 4 + random。

这样一个平面会被细分为四个平面，每个平面再递归地进行步骤 1、2，直到细分区间达到要求。

因为递归算法会有数据初始化问题【如递归计算平面 1 时要先确定确定点 1、5、8、9 的值，递归平面 2 时要先确定点 5、2、9、6 的值，对于点 5、9 来说计算顺序并不容易确定】，故采用循环方式逐层计算。

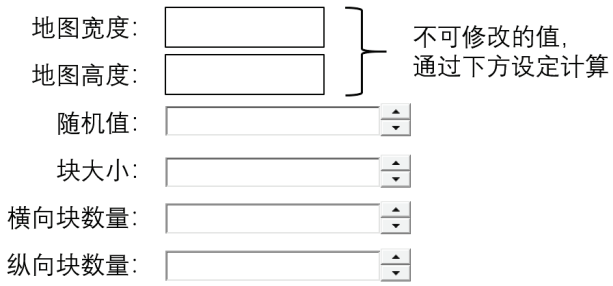
2. RTConfiguration

RandomTendCreater 的配置类型

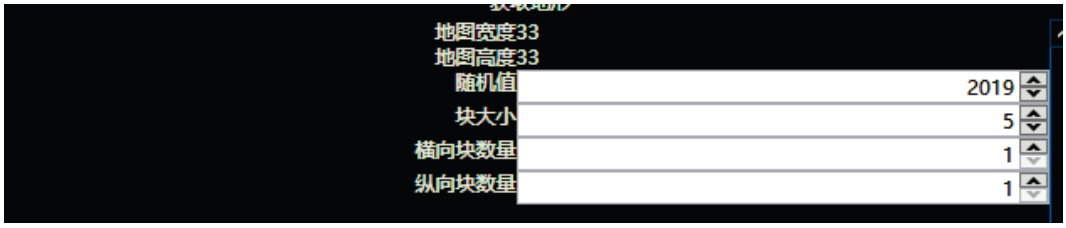
继承：Configuration

● 配置面板

原型设计



实际效果



3. RandomTendCreaterFactory

用于创建、获取相应的 MapCreater 和 Configuration 的工厂类

继承：MapCreaterFactory

十五、WorldCreaterStudio_Core 模块功能及重要模块细则

1. 子命名空间或逻辑文件夹功能描述

- ListNode

逻辑节点文件夹，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

FrontendFactory 的 Creater 相关基类的定义。

- FrontendNode

用于定义 FrontendFactory 的相关节点。

● BackendNode

BackendFactory 的逻辑节点的相关定义，内包含 AtmosphericMotion、RainfallMotion、SolarIlluminance、Biomes 四个文件夹，用以区分四个过程的节点。

● 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。

● Exceptions

用于定义所有所有应用可能用到的异常类型。

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。

继承：INotifyPropertyChanged

a) 字段和属性

名称	类型	可赋值	可获取	默认值	描述
Work	Work	N/A	T		获取节点所在的工作
ShowPanel	ControlTemplate	N/A	T		获取节点可展示的面板
NodeName	string	N/A	T		获取节点在工程树形图中的展示名称
Icon	ImageSource	N/A	T		获取节点在工程树形图中的展示图标
Childrens	ObservableCollection <IWorkLogicNodeAble>	N/A	T		获取节点的子节点
Changed	bool	N/A	T		获取节点是否有值发生了改变

b) 事件

● NodeValueChanged

```
event NodeValueChangedEventType NodeValueChanged
```

节点值发生改变事件，用于通知需要保存

c) 方法

● XmlNode

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

获取节点的 XML 节点

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

3. 主要节点规划

a) 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

● 该节点没有可用于展示的面板。

b) ListNode/Work

表示一个工作

继承：IWorkLogicNodeAble, InotifyPropertyChanged

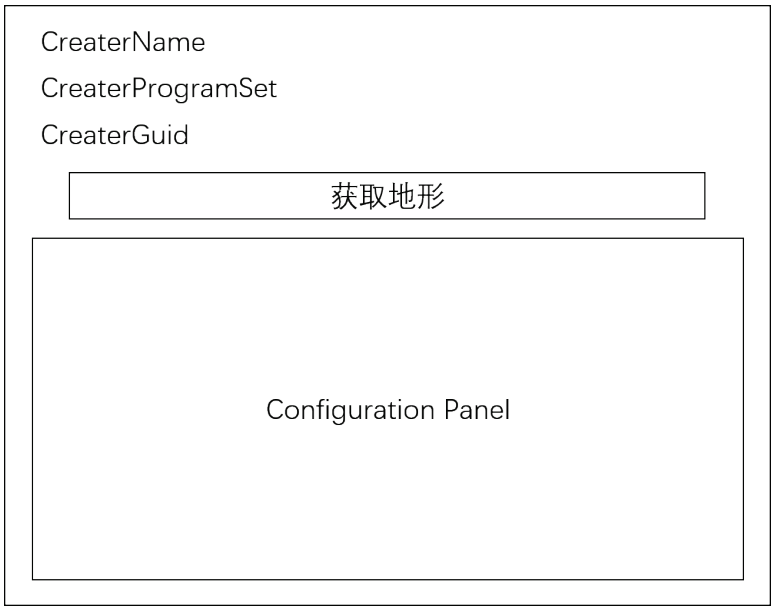
Work 是一个工作流程的最小完整管理节点，用于管理一个工作的所有内容，包括资源和流程。

c) ListNode/FrontEndFactory

前端工厂管理节点。

继承：IworkLogicNodeAble

- 展示面板原型设计

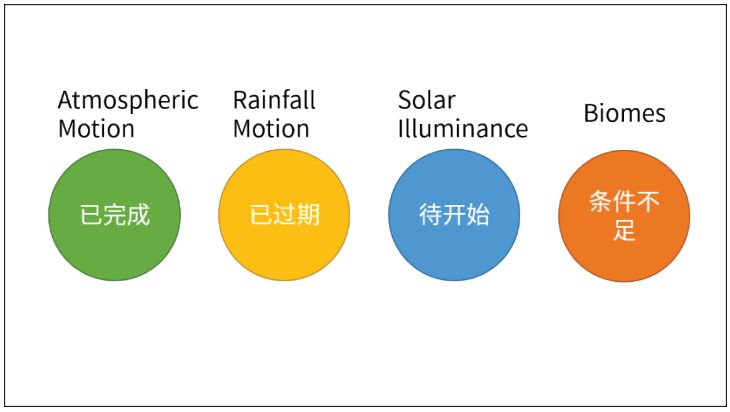


d) ListNode/BackEndFactory

后端工厂管理节点。

继承：IworkLogicNodeAble

- 展示面板原型设计



e) Resouses/ImageResourceManager

管理工作的全部图片资源，通过 key 进行图片索引和资源获取。

图片的 key 值在工作内唯一，与文件存储名称可以不相同。

- 该节点没有展示面板