目　　录

第一章 软件配置说明 1

1.1 开发系统配置 1

1.2 开发环境配置 1

第二章 主要功能和性能指标 2

2.1 主要功能 2

2.2 性能指标 2

第三章 用户手册和教程 2

3.1 交互方式与流程的概述 2

3.2 快速教程手册 3

3.2.1 运行 Sample Scene（可以跳过） 3

3.2.1 配置您自己的化学引擎 5

3.2.2 使用化学引擎服务于您的场景 7

3.2.3 进阶一：使用天气系统 12

3.2.4 进阶二：在现有基础上的改写 12

第一章 软件配置说明

1.1 开发系统配置

OS 名称: Microsoft Windows 10 专业版

OS 版本: 10.0.19043

系统类型: x64-based PC

处理器: Intel64 Family 6 Model 158 Stepping 10 GenuineIntel ~2201 Mhz

图形处理器： Intel(R) UHD Graphics 630

NVIDIA GeForce GTX 1060

1.2 开发环境配置

Unity： 2020.3.23f1c1 (.NET Standard 2.0)

VS Code: 1.66.2 (user setup)

Node.js: 16.13.0

表3.1 Unity 2020LTS 最低运行要求

	Windows	macOS	Linux（在预览中支持）
操作系统版本	Windows 7 (SP1+), Windows 10 and Windows 11, 64-bit versions only.	High Sierra 10.13+	Ubuntu 20.04、Ubuntu 18.04 和 CentOS 7
CPU	X64 架构（支持 SSE2 指令集）	X64 架构（支持 SSE2 指令集）	X64 架构（支持 SSE2 指令集）
图形 API	DX10、DX11 和 DX12 兼容型 GPU	Metal 兼容型 Intel 和 AMD GPU	OpenGL 3.2+ 或 Vulkan 兼容型 Nvidia 和 AMD GPU
其他要求	硬件供应商正式支持的驱动程序	Apple 正式支持的驱动程序	在 X11 窗口系统上运行的 Gnome 桌面环境、Nvidia 官方专有图形驱动程序或 AMD Mesa 图形驱动程序。提供的其他配置和使用者环境以及受支持的发行版（内核、合成器等）
对于所有操作系统，工作站或笔记本电脑机型均支持 Unity Editor，无需仿真、容器或兼容性层即可运行。			

第二章 主要功能和性能指标

2.1 主要功能

总的来说，本设计需要根据一定的法则，完成物体状态变化的计算。具体来说可以拆解为以下几点：

- 计算接收物在某一时刻是否符合反应条件
- 如果符合，计算应该发生什么反应
- 允许使用者自定义反应法则
- 允许使用者自定义元素和接收物种类
- 提供统一的环境条件管理

2.2 性能指标

本设计服务于游戏开发，主要性能指标为游戏帧率。帧率越高，性能越好，反之性能越差。

第三章 用户手册和教程

3.1 交互方式与流程的概述

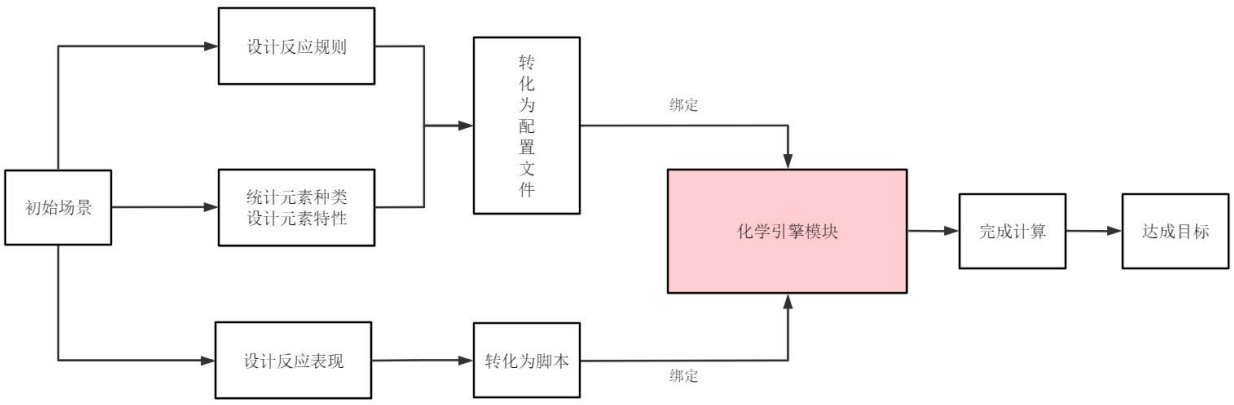

图 3.1 交互流程

使用者首先确定好对元素种类、元素特性、反应法则和反应表现，转化为对应的格式。其中，反应规则和元素类型以.json文件的格式存储，元素特性可以直接在Inspector面板中配置，反应表现需要转化为代码语言，使用约定好的函数名称。

之后，将以上文件绑定到模块中，启动游戏后，模块会进行自动计算和调用。

3.2 快速教程手册

3.2.1 运行 Sample Scene（可以跳过）

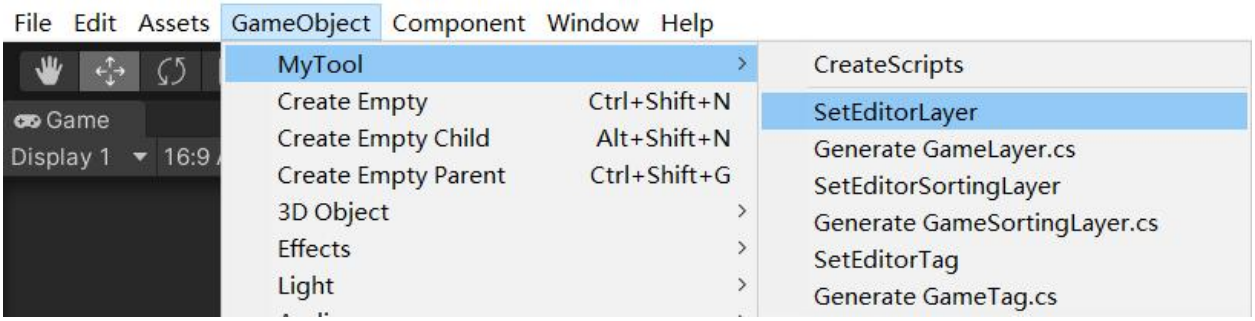

图 3.2 设置 Layer

导入完成后，点击GameObject-MyTool-SetEditorLayer。这一步将Demo场景中的Layer信息同步到您的工程中。注意，这一步会覆盖您现有的Layer信息，因此请在新的工程中试用；如果不需要试用Demo文件，您可以跳过这一步，从3.2.2开始。

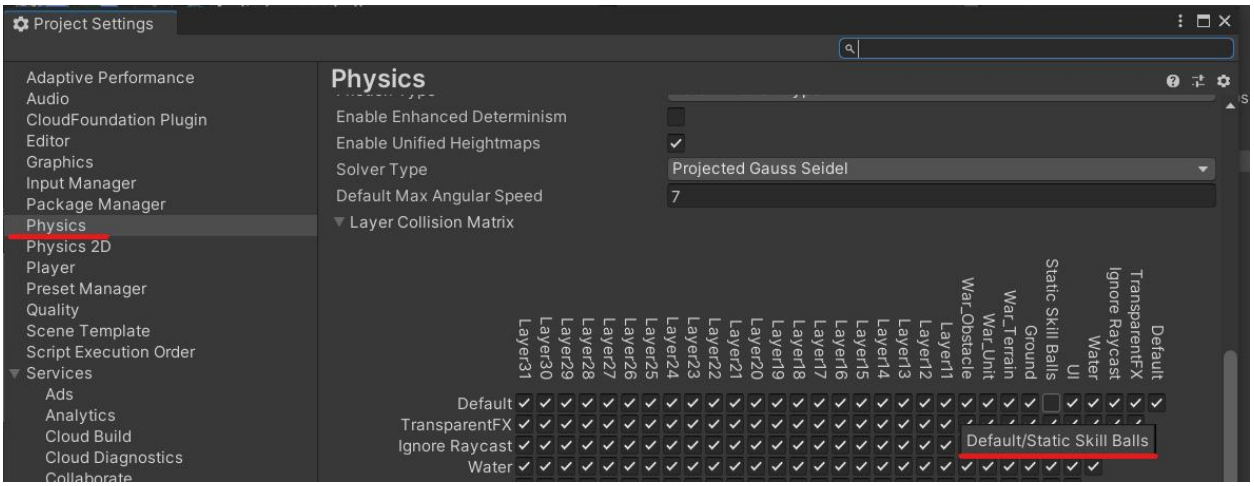

图 3.3 设置碰撞条件

接着，打开Edit-Project Settings，选择Physics选项卡，找到Layer Collision Matrix，取消勾选Default/Static Skill Balls。它在第一行的倒数第六列，请见上图。

接着请打开Scene文件夹下的Sample01。检查场景中Stylized Astronaut的Layer是否已经被标记为“Static Skill Balls”。如果是，说明现在我们已经完成了Sample Scene的设置，可以开始试用了！

点击运行按钮。您将看到游戏运行，摄像机随着玩家呼吸慢慢晃动。此时您可以按下键盘上方的数字键1或2发射含有火或水属性的武器。当您按下2时，水属性武器发射，但场景不会有变化；当您按下1时，火属性武器发射，您面前的树木等可燃物将被点燃，颜色发生变化并冒出红色的烟。

图 3.4 点燃树木

当树木未完成燃烧时，您可以按下2来发射水属性的武器。面前燃烧的树将被熄灭，颜色变暗并冒出暗蓝色的烟。

图 3.5 熄灭燃烧的树木

您可以使用WASD在场景中漫游，并尝试点燃和熄灭其他可燃物。

接着，我们打开Sample Scene02。您将看到空无一物的场景，但不要着急。点击运行按钮，游戏将生成50*50的简单木林。您可以按下M键，观察木林的引燃情况，红色代表木材被引燃。如果您看到红色区域扩散，说明配置正常。

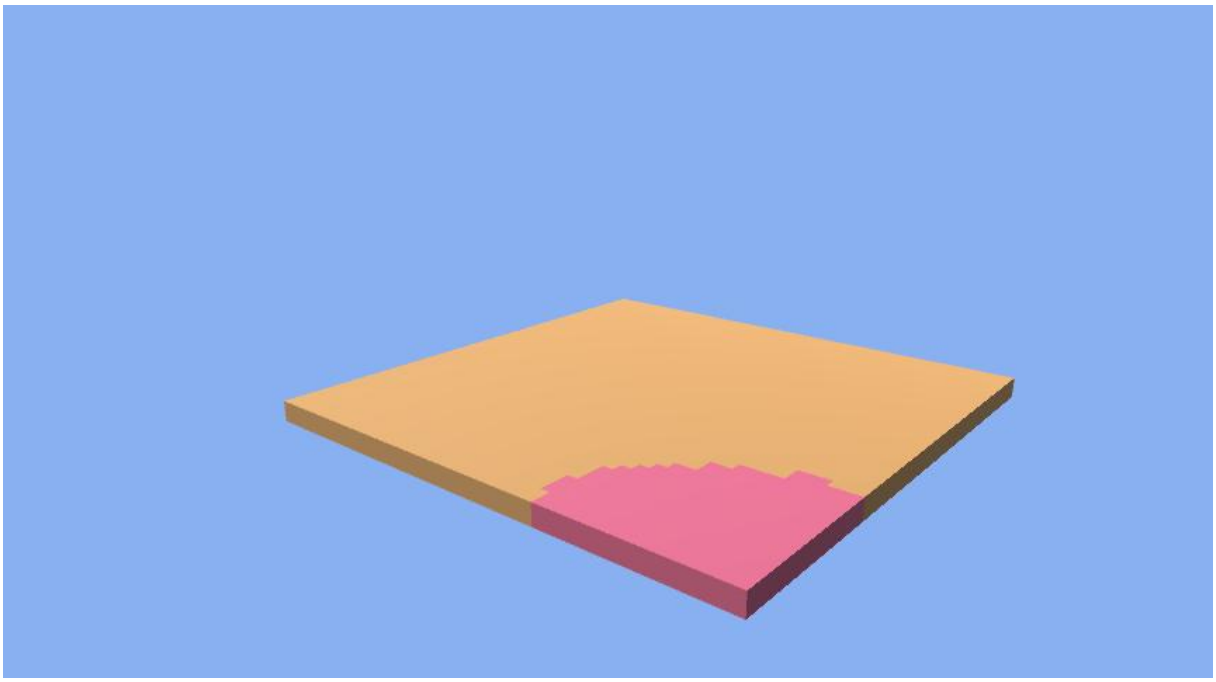

图 3.6 火元素的传递

以上步骤中如果出现问题，请重新导入工具包；或者将您的Unity版本、运行截图和Project Settings中的Physics部分截图保存下来，并联系开发者。

3.2.1 配置您自己的化学引擎

在文件资源管理器或访达中打开 ChemistryEngineScripts 文件夹，并打开Scripts-StreamingAssets-Config，您将看到两个.json 文件，用于配置您自己的化学引擎。开发者已经给出示例，**在教程中，您可以仅跟随步骤来学习配置，而不用自行更改。**

```
[
    {
        "EleType": "火"
    }
]
```

在ElementManager中，您可以使用EleType关键字，添加您自己的元素类型，这些元素类型将自动生成从0开始的ID，无需手动配置。类型将在元素设置面板中显示出来，用于区分不同的元素。在教程中，已经配置好ID=0的火元素，ID=1的冰元素和ID=2的水元素。

```
[
    {
        "FuncName": "OnFire",
        "PrevElementID": "-1",
        "NextElementID": "0"
    }
]
```

在RuleManager中，您可以使用以上格式定义您的反应规则：当接收物从PrevElementID的状态变为NextElementID的状态时，引擎将调用名字为FuncName的函数。ID为-1表示该接收物当前什么反应也没有进行，当它被ID=0的元素影响时，将调用OnFire函数。OnFire函数规定了反应的表现，您需要自行编写，并将这个脚本绑定在接收物上。在教程中，已经创建了以下规则：

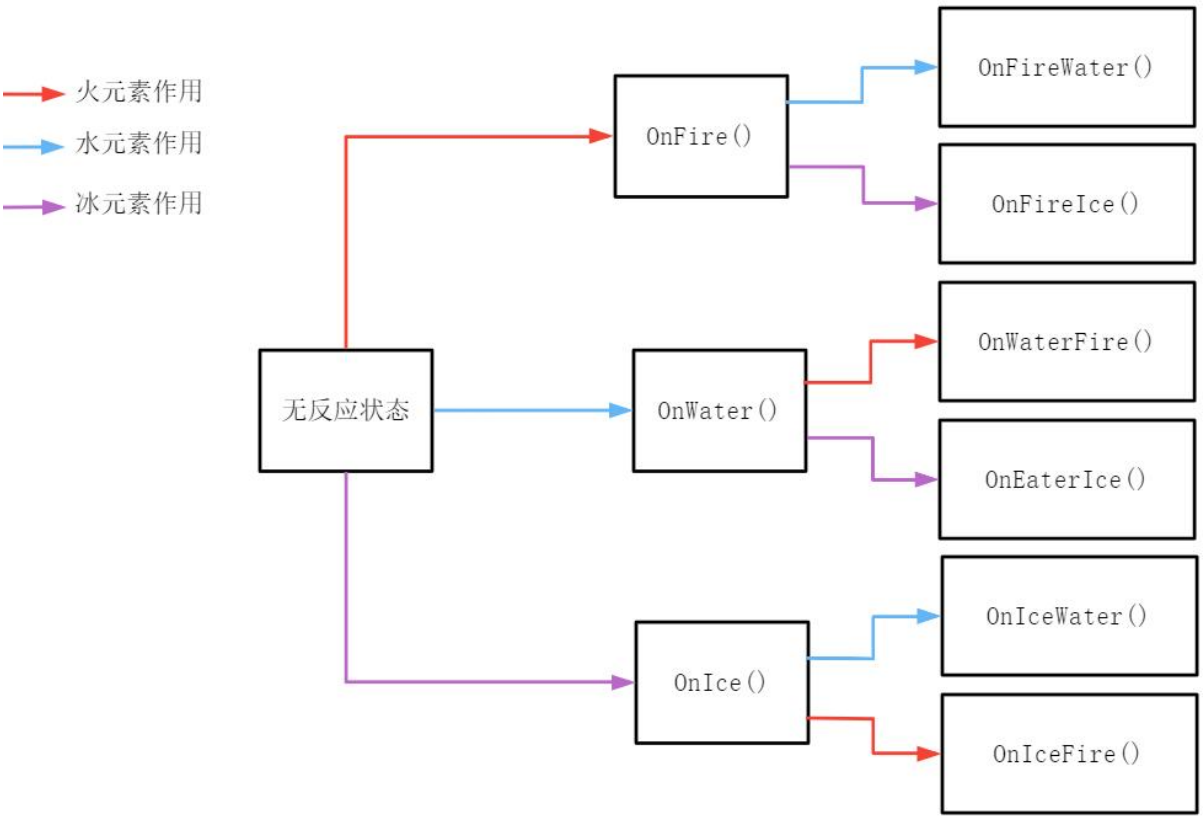

图 3.7 教程中设置的反应法则

3.2.2 使用化学引擎服务于您的场景

首先新建一个场景，创造一个空物体，并将它命名为CE。

为 CE 添加组件

单击Add Component，搜索“pool”，添加Active Reation Pool组件，并单击“更新接收物脚本”按钮，如果成功，会打印以下Log。

图 3.8 更新接收物脚本

创建元素和接收物

在场景中创建一个cube作为接收物，创建一个sphere作为元素。将它们命名为“Reciever”和“Element”。将Reciever的Transform.position设置为（-0.5，0，0），Element的Transform.position设置为（0.5，0，0）。确保您在摄像机预览画面中可以看到两个物体。

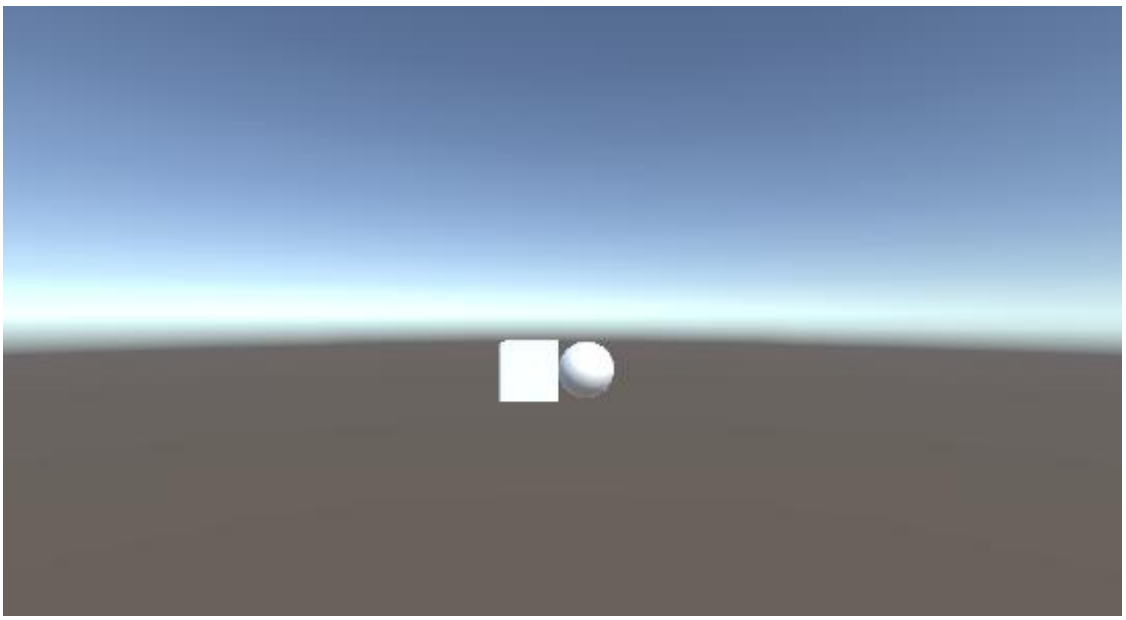

图 3.9 创建元素和接收物

设置元素属性

为Element添加元素组件。单击Add Component，搜索“ele”，添加An Element组件，正常界面如下图所示。如果您在此处发现Unity报错，请确认您是否已经单击过“更新接收物脚本”按钮。如果再次单击后仍然报错，请联系开发者。

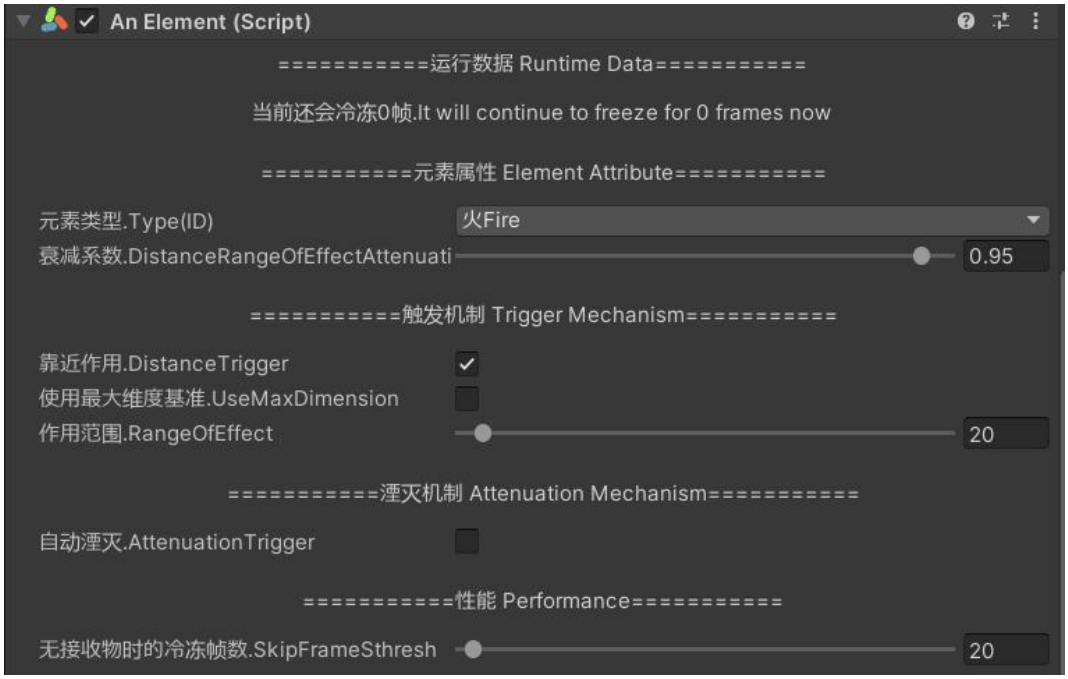

图 3.10 元素设置面板

首先我们对面板如何使用进行说明。面板由运行数据、元素属性、触发机制、湮灭机制和性能四个板块构成。

运行数据：游戏进入Play Mode时，会展示对应元素的冷冻情况和生命周期情况（如有）。

元素属性：

元素类型下拉菜单中可以选择我们在ElementManager.json中设置的元素种类。

传播衰减系数定义了每次传播时元素的影响范围的衰减程度。例如传播衰减系数为0.9、作用范围为100的元素，经过一次传播，接收物所携带的元素作用范围变为0.9*100=90。需要注意，传播衰减系数本身也会衰减，所以传播时的范围递减是平方的。

触发机制：定义该元素的作用方式。

如果不勾选靠近时作用，意味着该元素只有切实和接收物发生碰撞才能够发生作用。

如果元素勾选了靠近时作用，可以进一步设置元素的作用范围。这个范围是相对于物体的球形碰撞体而言向外扩张的半径，例如一个物体使用Box Collider，这个Collider的维度

大小分别为x 20，y 30，z 50，设置它的作用范围为40，那么引擎将在（x 20+40=）60的半径范围内寻找合法接收物。如果勾选“使用最大维度基准”，则会使用（z 50+40=）90的半径来寻找。最大维度基准是指物体的球形碰撞体半径是否采用当前碰撞体xyz轴中的的最大维度。

图 3.11 元素勾选靠近作用时的界面

湮灭机制：为了方便使用者设计简单元素，以便对游戏机制进行快速验证。但因为这项功能混淆了高层的使用者操作和中层的计算，可能造成意想不到的矛盾，因此除了验证游戏机制是否合理以外，在实际游戏中不建议使用。

勾选自动湮灭之后，可以进一步设置元素的生命时长，元素激活后会开启生命倒计时。

勾选“湮灭后是否也毁灭游戏对象”，则会在元素湮灭的同时销毁挂载了元素的物体。

初始强度为倒计时开始之前的强度，随着时间强度会不断减弱直到1以下，元素“湮灭”。

强度的下降方式有两种，一是线性递减，每秒下降固定的强度，使用者可以直接设置元素在多久之后湮灭；二是非线性递减，元素的强度会下降的越来越快，因此使用者可以通过设置衰减系数来间接设置元素的生命时间。

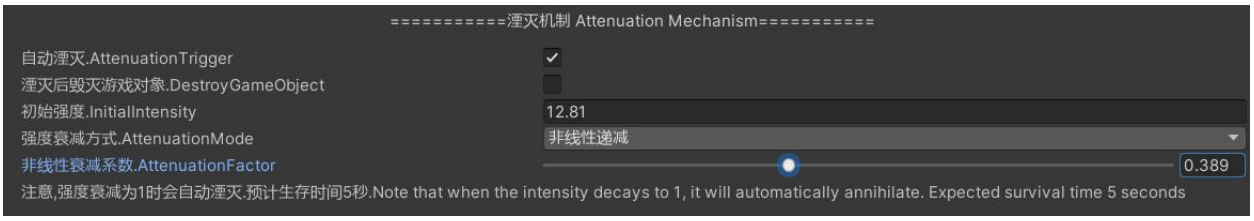

图 3.12 元素勾选自动湮灭时的界面

性能：本引擎使用类似计算机网络中慢启动的方式来逐渐减缓对活跃元素的监测。性能面板主要控制元素最开始时的冷冻门限。该门限为0时，元素每一帧都会被监测，消耗性能但监测及时、准确；该门限不为0时，元素的冷冻时间会在游戏中被动态更改以便达到性能最大化。建议设置为20，门限过高会导致元素作用发生延迟。

在教程中，请按图3.10设置元素属性。

设置接收物属性

为Reciever添加化学材质组件。单击Add Component，搜索“surr”，添加A Surrounding组件，正常界面如下图所示。如果您在此处发现Unity报错，请确认您是否已经单击过“更新接收物脚本”按钮。如果再次单击后仍然报错，请联系开发者。

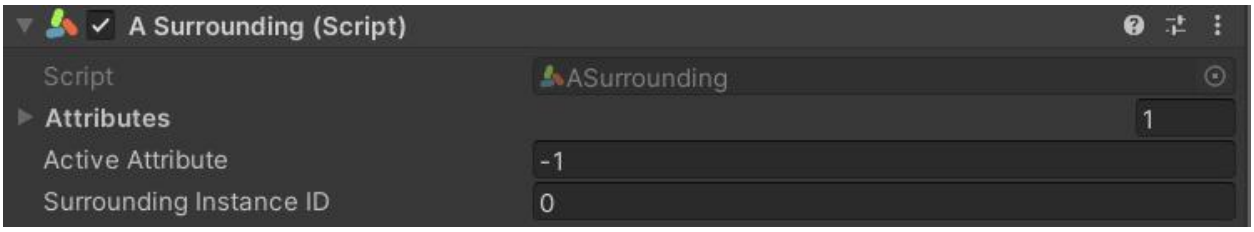

图 3.13 接收物属性面板

对于接收物，我们只需设置Attributes即可。Attributes标记了该接收物能够和什么元素反应。单击加号，添加和设置如下属性。

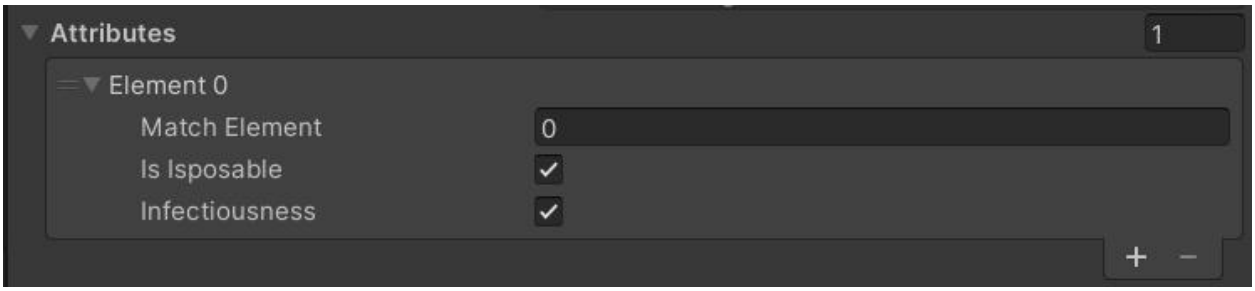

图 3.14 接收物属性面板

Match Element标记了元素的ID，Is Isposable标记了该属性是否一次性，如果勾选，意味着这一属性只能激活一次。例如，木材的点燃是一次性的，冰的融化不是一次性的。Infectiousness标记了该属性能否同时复制它所对应的元素。例如，木材被点燃可以携带新的火元素，但木材被打湿就仅仅是被打湿，不能成为新的水元素。

编写接收物脚本

在RuleManager中，我们已经规定了物体被火元素影响时，执行OnFire函数。新建一个脚本，命名为Cube.cs。接下来，您将学习如何让化学引擎和您的代码通信。

首先，让Cube类继承RecieverBase类。

接着我们编写一个OnFire函数，让Cube在被点燃时，打印出“OnFire”的文字。注意使用override 关键字，并包含一个AnElement类型的形参。

```
1.      public override void OnFire(AnElement e)
2.      {
3.          Debug.Log("OnFire");
4.      }
```

然后，我们在 Start() 中绑定这个函数：

```
1.      void Start()
2.      {
3.          Bound();
4.      }
```

现在您得到了一个完整的接收物脚本，将它挂载到Reciever上。

编写元素脚本

那么一个元素何时才能开始发挥作用呢？我们还需要编写元素脚本。新建一个脚本，将它命名为Sphere.cs。声明一个AnElement类型的成员变量，并在Start函数中初始化它。

```
1.      AnElement _thisElement;
2.      void Start()
3.      {
4.          _thisElement = gameObject.GetComponent<AnElement>();
5.      }
```

我们希望玩家按下M键时，元素被激活：

```
1.          if (Input.GetKeyDown(KeyCode.M))
2.          {
3.              this_element.ElementIfActive = true;
4.          }
```

现在您得到了一个完整的元素脚本，将它挂载到Element上。

运行您的游戏

点击运行按钮，并按下M键，Console窗口显示出“OnFire”字样。恭喜！您已经掌握了这个工具的基本使用方法！

3.2.3 进阶一：使用天气系统

我们知道，很多反应的剧烈程度，是依附于天气情况的。为此开发者提供了环境管理器，默认设置了湿度（wet），温度（temprature）和风力（wind）三个变量，默认值均为1.0f。

首先，在Cube.cs中声明管理器变量。接着在OnFire函数中，获取湿度的值并打印出来。

```
1.  public class Cube : RecieverBase
2.  {
3.      Environment _environment = Environment.EnvironmentInstance;
4.      void Start()
5.      {
6.          Bound();
7.      }
8.
9.      public override void OnFire(AnElement e)
10.     {
11.         Debug.Log(_environment.GetEnvironment("wet"));
12.         Debug.Log("OnFire");
13.     }
14. }
```

运行游戏，激活元素，Console窗口中打印出“1”，说明我们已经成功的获取了湿度数值！天气系统的其他API请参考文档：https://silmont2000.github.io/ChemistryEngineDoc/。

3.2.4 进阶二：在现有基础上的改写

该项目是作者本科期间的毕业设计，样本量尚不够大，也有很多不足，欢迎指正和改写。具体逻辑请参考文档：https://silmont2000.github.io/ChemistryEngineDoc/