<基于元宇宙平台的场景搭建系统>

软件架构文档

版本 <1.0>

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| <2/11/2022> | <1.0> | <第一版软件架构文档> | <花泉润，叶鹏鹏，封琪> |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目录

1. 简介 4

1.1 目的 4

1.2 参考资料 4

2. 用例视图 4

3. 逻辑视图 4

3.1 概述 4

3.2 在构架方面具有重要意义的设计包 4

4. 进程视图 4

5. 部署视图 4

6. 实现视图 5

7. 技术视图 5

8. 质量属性的设计 6

8.1 Attribute的使用 6

8.2 界面的Prefab化 6

软件架构文档

# 简介

## 目的

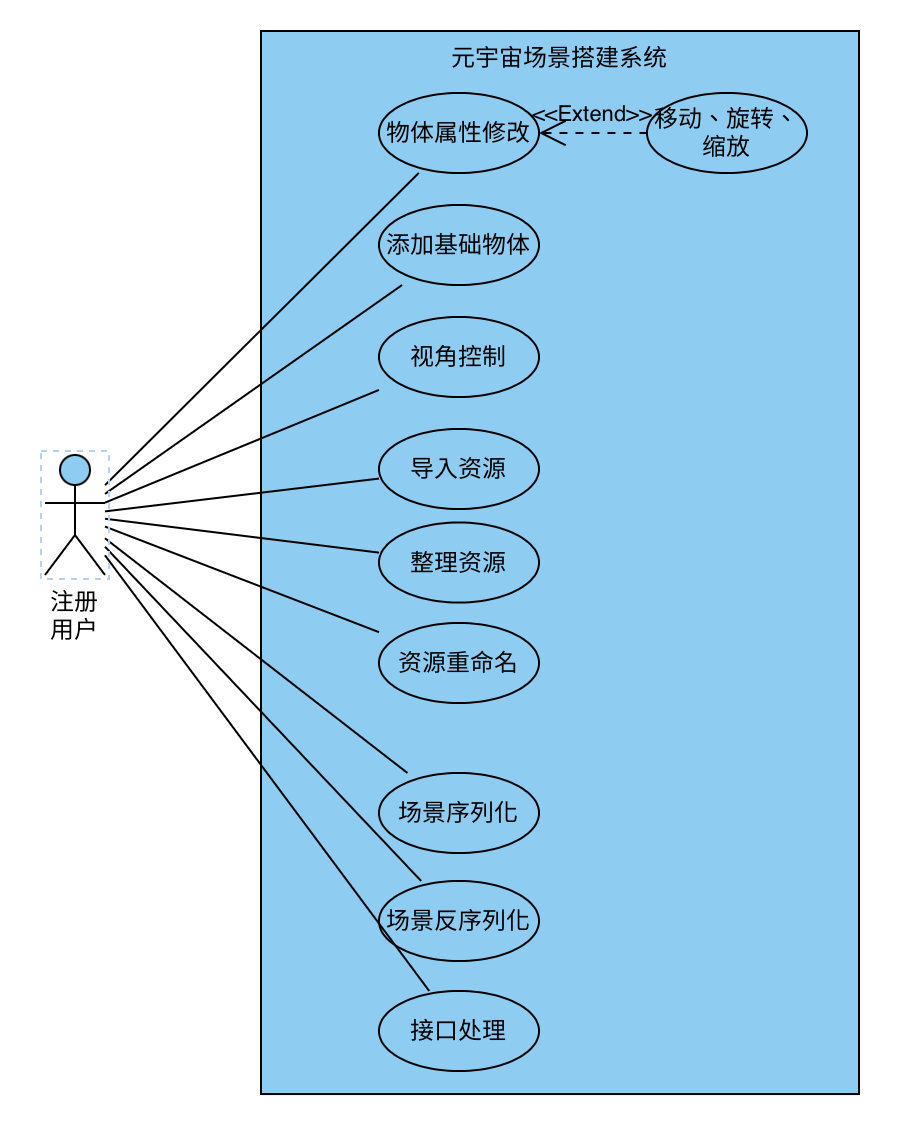
本文档将从构架方面对系统进行综合概述，其中会使用多种不同的构架视图来描述系统的各个方面。它用于记录并表述已对系统的构架方面作出的重要决策。

文档中详尽说明了本项目的多种构架视图以及质量属性的设计。此文档面向软件开发团队，通过多种构架视图为开发团队提供了清晰、统一的软件架构，明确了开发路径与开发方向，避免因架构模糊带来的无效工作，为团队开发提供了便利。

## 参考资料

1. 温昱. 软件架构设计[M]. 电子工业出版社, 2007.
2. DaivdM.Dikel, 迪克尔, Dikel,等. 软件架构[M]. 机械工业出版社, 2002.
3. 软件工程：面向对象和传统的方法（原书第8版），（美）Stephen R. Schach著，邓迎春、韩松等译，机械工业出版社，2018.

# 用例视图



## 物体属性修改

用户可以对物体进行移动、旋转和缩放操作。

## 添加基础物体

用户可以在物品栏添加提供的基础物体。

## 视角控制

用户可以通过鼠标左右键以及滚轮对视角进行操控。

## 导入资源

用户可以导入.obj等模型文件。

## 整理资源

用户导入的资源可以被整理为9宫格状。

## 资源重命名

用户可以对资源进行重命名。

## 场景序列化

用户导出场景时可以进行场景序列化操作。

## 场景反序列化

用户导入场景时可以进行场景反序列化操作。

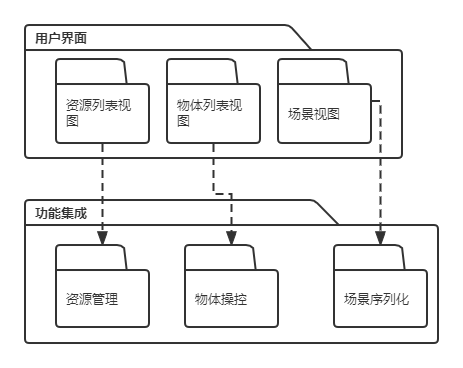
## 接口处理

在场景存储时，场景与unity系统的接口处理。

# 逻辑视图

## 概述

ASP项目的逻辑视图由2个主要部分组成：用户界面模块和功能集成模块。 其中用户界面包含资源列表视图、物体列表视图、场景视图三个组成部分。功能集成包含资源管理、物体操控、场景序列化三个模块。



## 在构架方面具有重要意义的设计包

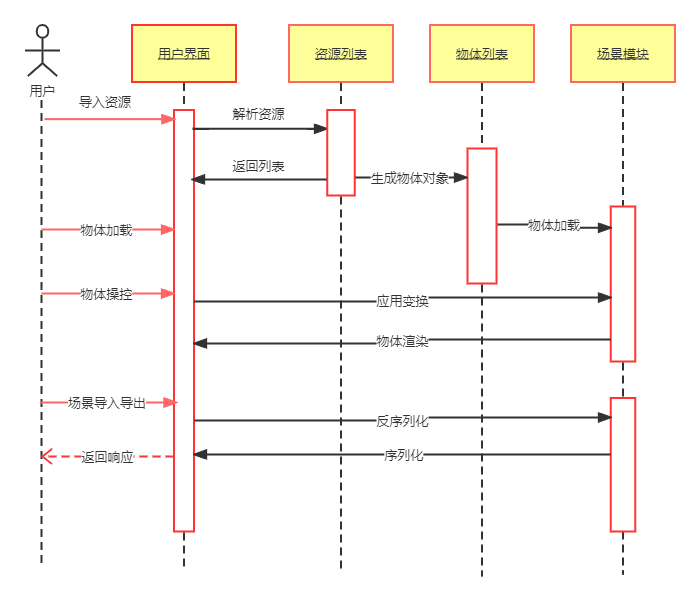
### 用户界面

1. 资源列表视图。包含了用户用于加载的每个文件的类实例。支持用户进行导入、重命名、拖拽排序、加载进场景等操作。
2. 物体列表视图。包含了该场景已加载的物体的类实例，支持重命名，选中，控制物体位置旋转缩放等方法。并支持显示物体的信息与绑定的行为组件等。
3. 场景视图。用于管理导入该场景的资源文件，以及场景内存在的物体，支持选中物体。并支持场景的序列化与反序列化以此进行导入导出操作。

### 功能集成

1. 资源管理。包含对资源文件对象中导入、重命名、拖拽排序、加载等方法的实现。
2. 物体操控。包含物体对象重命名，选中，控制物体位置旋转缩放等方法的实现。
3. 场景序列化。包含场景对象序列化反序列化方法的实现。

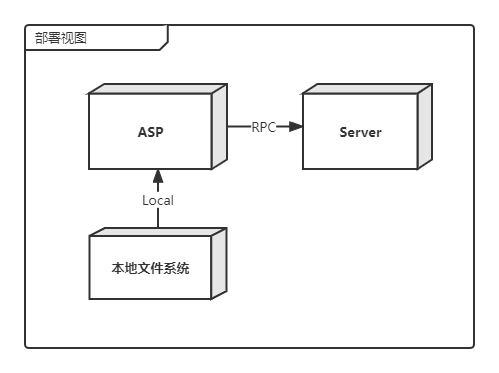
# 进程视图



本项目的进程情况如图所示。

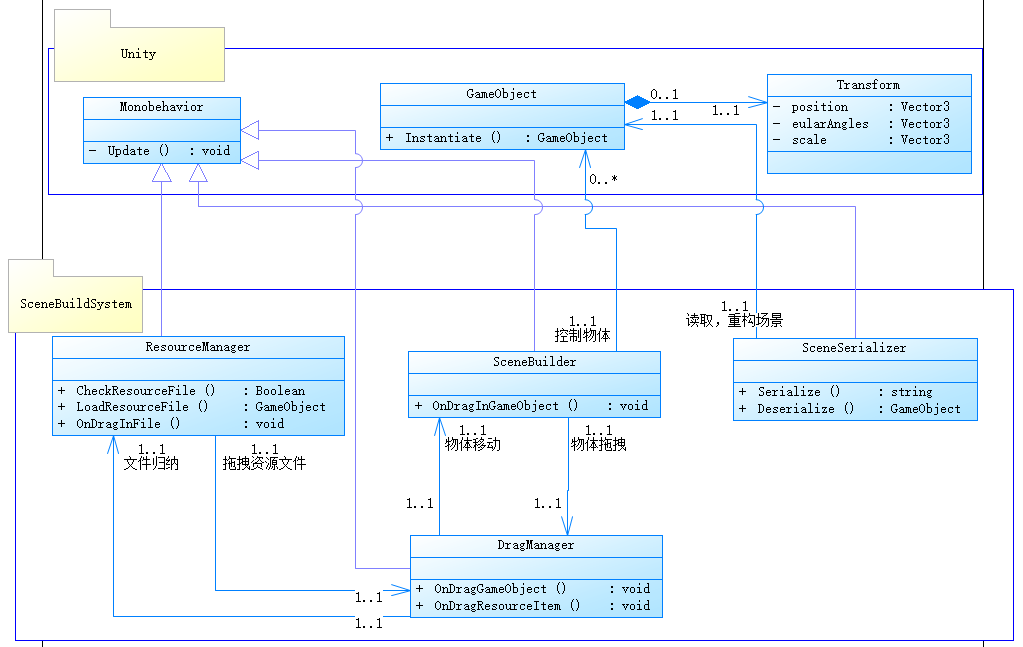
用户与UI界面直接交互，进行导入资源，物体加载等操作的触发。UI进一步调用资源列表、物体列表、场景模块等组件的方法进行数据处理。

# 部署视图



本项目的部署节点涉及如图三个部分。包括两个通信部分：ASP项目调用本地文件系统进行文件导入，并通过RPC（远程过程调用）方式访问服务端进行信息交互。

# 实现视图



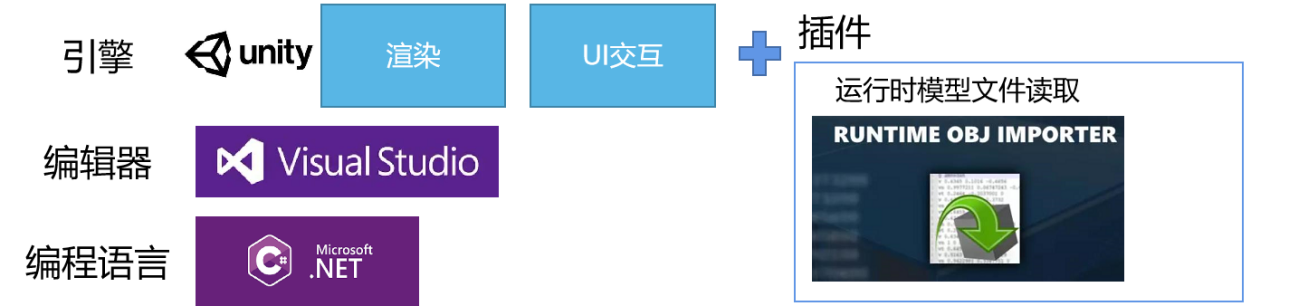
Unity包为Unity提供的API，SceneBuildSystem包为本项目的开发范围的包。

本项目的模块与类的对应关系图如下：

* 资源加载模块：ResourceManager
* 场景搭建模块：SceneBuilder
* 场景序列化、反序列化模块：SceneSerializer

每个类同时负责业务与对应模块视图中的UI处理，但对UI上的拖拽操作单独分离至DragManager类管理。

# 技术视图



本项目基于Unity引擎制作。使用VisutalStudio，用C#编写。

为实现模型文件的读取功能，Unity上加入第三方插件Runtime OBJ Importer为本系统赋予运行时从文件系统读取Mesh与材质数据的能力。

# 质量属性的设计

## Attribute的使用

Unity提供了多种字段（Attirbute）来提升显式编程的安全性，如[DisallowMultiComponent]，[RequireComponent()]，[Range()]等。由于Unity混合了显式编程的功能，开发人员可以在Unity的组件页面上进行组件类的安装与调参，而上述字段能为显式编程过程提供约束力，提高显式编程部分的安全性，避免开发人员的误操作。

## 界面的Prefab化

Unity提供了预制体（Prefab）作为程序界面、物体的预制物，可以省去用代码编写UI的生成流程。通过界面组件的低颗粒度的Prefab化，可以将大部分界面内容保持在显式编程内完成，有助于提高本系统的扩展，移植工作量的缩小。