Mesh-based Application

小组成员及分工

组员	学号	班别	分工
颜府	18342113	软工教务二班	协作完成每部分
郑卓民	18342138	软工教务二班	协作完成每部分

设想（Vision）

简介：

我们希望开发一个基于网格划分（Mesh）模拟与仿真的系统（MeshHelper），其广泛用于航天、水文、制造业、物理学等领域。我们设想 MeshHelper 是下一代 Mesh 应用，能够容错，具有灵活性以支持各种客户的不同业务规则，能够在各种终端系统中兼容性运作，并且能够与各种第三方支持软件进行整合。

定位：

1. 商业机遇：

就各种业务规则和功能需求而言，现有的 Mesh 产品无法适应客户的业务。此外，在业务和终端增长时，现有产品不能很好地扩展。不能动态地根据错误进行调整。现有产品都无法方便地与大量第三方系统集成。以上这些不灵活地情况无法满足市场地需求，因此需要一款新型 Mesh 软件来改变这种情况。

2. 问题综述：

传统的 Mesh 应用灵活性、容错性差，并且难以与第三方系统集成。

3. 产品定位综述：

通过实现一个基于 C++语言的用户帮助库，供领域用户用以协助开发基于 mesh 的应用以及在已离散化的物理模型上的各种变化的模拟与仿真。领域用户可以使用我们的库载入一种/若干种常见格式的 mesh 文件，并能方便、高效地对该 mesh 进行各种操作，例如修改结点信息、自定义核运算（kernel operation）等，最终能将处理后的结果输出出来并保存为文件。

涉众描述：

1. 用户概要：

从事 Mesh 软件开发的专业人员、普通体验 Mesh 功能的尝鲜用户、学习开发 Mesh 的学生。

2. 用户级目标：

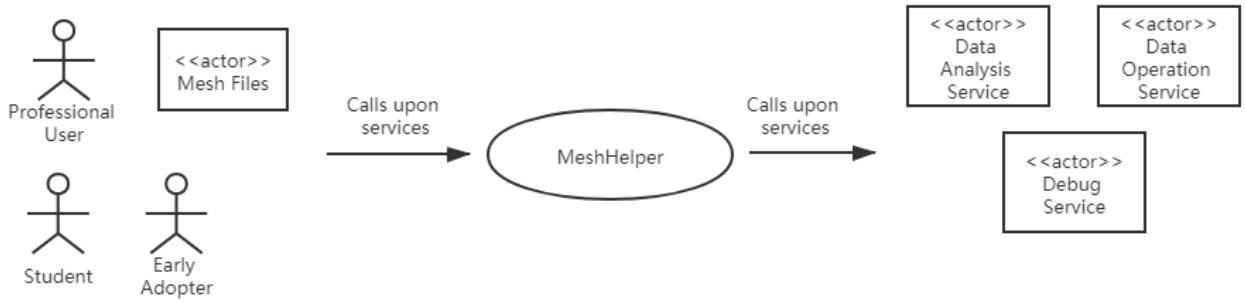
- 从事 Mesh 软件开发的专业人员：协助分析 Mesh 文件，操作 Mesh 文件。
- 普通体验 Mesh 功能的尝鲜用户：简单的使用接口帮助分析 Mesh 文件。
- 学习开发 Mesh 的学生：简单的分析与过程输出来帮助分析数据。

产品概况：

1. 产品展望：

MeshHelper 应用通常作为一个协助库在 C++语言中被引用使用，该库能协助用户完成各种需求服务，对 Mesh 文件进行分析与操作。

MeshHelper 系统语境图：



2. 假设和依赖：

本应用需要运行于安装了 C++编译器的 Linux 系统上。

其他需求和约束：

参见补充性规格说明和用例部分。

详述用例（Fully dressed use cases）

用例 1：创建 Mesh 对象

范围：MeshHelper 库

级别：用户目标

主要参与者：领域用户

涉众及其关注点：

- 领域用户：希望能够准确、快速地产生一个想要的 Mesh 对象，并能输出到指定文件中。

前置条件：无特殊要求，只需能正确引入本库并能运行程序。

成功保证：在内存里生成正确的 Mesh 对象，并可输出符合格式的 Mesh 文件。

主成功场景：

1. 领域用户使用 C++语言正确导入本协助库。
2. 领域用户创建一个系统实例对象。
3. 用户自己通过一定算法生成一定符合格式的数据。
4. 系统根据用户生成的数据转换成符合 Mesh 对象格式的数据结构。
5. 用户可选择调用保存功能将前面所生成的 Mesh 对象保存输出到指定的文件地址的 Mesh 文件中。

扩展：

*a：用户使用的数据有越界等问题：

1. 返回错误代码，并在日志文件中记录错误类型与内容，防止系统崩溃。
2. 用户应停止后续操作，纠正错误内容后重启系统。

*b：系统运行过程中遇到不可避免的硬件错误或系统中断：

1. 在日志文件中记录当前运行状态，系统关闭。
2. 用户应尽快查明外部原因后重启系统恢复错误前状态或重新初始化系统。

发生频率：可能会偶尔发生。

用例 2：导入 Mesh 对象

范围：MeshHelper 库

级别：用户目标

主要参与者：领域用户

涉众及其关注点：

- 领域用户：希望能准确、快速、便捷地导入已有的 Mesh 对象到内存中。

前置条件：正确引入本库并能运行程序，用户提供符合格式的 Mesh 文件。

成功保证：在内存里生成文件对应的 Mesh 对象。

主成功场景：

1. 领域用户使用 C++语言正确导入本协助库。
2. 领域用户创建一个系统实例对象。
3. 调用系统导入数据功能导入指定文件路径下地 Mesh 文件并进行转化。
4. 在内存中生成我们设计的 Mesh 对象数据结构。

扩展：

*a: 用户导入的 Mesh 文件中的数据有错误或 Mesh 文件不符合标准格式：

1. 返回错误代码，并在日志文件中记录错误类型与内容。
2. 用户应停止后续操作，纠正错误内容，并重新启动系统。

*b：系统运行过程中遇到不可避免的硬件错误或系统中断：

1. 在日志文件中记录当前运行状态，系统关闭。
2. 用户应尽快查明外部原因后重启系统恢复错误前状态或重新初始化系统。

发生频率：可能会偶尔发生。

用例 3：模拟仿真核运算

范围： MeshHelper 库

级别：用户目标

主要参与者：领域用户

涉众及其关注点：

- 领域用户：希望能准确、快速、便捷地将核运算在 Mesh 对象上进行模拟仿真。

前置条件：正确引入本库并能运行程序，已经生成或导入了 Mesh 对象，用户提供核运算方法。并在此对象上进行操作。

成功保证：在 Mesh 对象上应用用户给定的核运算方法模拟仿真计算出正确的结果，并可以或输出到屏幕上或存储为文件。

主成功场景：

1. 领域用户使用 C++语言正确导入本协助库。并已经创建了一个系统实例对象，生成或导入了 Mesh 对象到内存中。
2. 用户提供核运算方法和模拟变化的次数，系统将用户自定义的符合规范的核运算方程传入到模拟仿真方法中，返回对每个结点进行该核运算后的新 Mesh 对象的信息。
3. 调用相关功能可以选择输出 Mesh 对象的信息，或者将 Mesh 对象存储到指定文件中。

扩展：

*a：用户提供的核运算方法不符合实际格式要求：

1. 返回相应错误代码，并在日志文件中记录错误类型与内容，防止系统崩溃。
2. 系统提醒用户修正核运算方程后重新执行相关功能。
3. 用户应停止后续操作，纠正错误内容，最后重启系统服务。

*b：系统运行过程中遇到不可避免的硬件错误或系统中断：

1. 在日志文件中记录当前运行状态，系统关闭。
2. 用户应尽快查明外部原因后重启系统恢复错误前状态或重新初始化系统。

发生频率：可能会不断地发生。

用例 4：输出 Mesh 对象信息

范围： MeshHelper 库

级别：用户目标

主要参与者：领域用户

涉众及其关注点：

- 领域用户：希望能准确、快速、便捷地输出指定 Mesh 对象的数据信息。

前置条件：正确引入本库并能运行程序，已经生成或导入了 mesh 对象，并在此对象上进行操作。

成功保证：能获取到正确的信息（如结点信息，边的信息）。

主成功场景：

1. 用户使用 C++语言正确导入本协助库。并已经创建了一个系统实例对象，生成或导入了 Mesh 对象到内存中。
2. 用户调用打印信息功能将当前 Mesh 对象的数据打印到屏幕上。

扩展：

*a：系统运行过程中遇到不可避免的硬件错误或系统中断：

1. 在日志文件中记录当前运行状态，系统关闭。
2. 用户应尽快查明外部原因后重启系统恢复错误前状态或重新初始化系统。

发生频率：可能会偶尔发生。

用例 5：导出 Mesh 对象

范围： MeshHelper 库

级别：用户目标

主要参与者：领域用户

涉众及其关注点：

- 领域用户：希望能准确、快速、便捷地导出指定 Mesh 对象的数据信息。

前置条件：正确引入本库并能运行程序，已经生成或导入了 mesh 对象，并在此对象上进行操作。

成功保证：能获取到正确的信息（如结点信息，边的信息）。

主成功场景：

3. 用户使用 C++语言正确导入本协助库。并已经创建了一个系统实例对象，生成或导入了 Mesh 对象到内存中。
4. 用户调用导出对象功能将当前内存中的 Mesh 对象以标准 Mesh File 格式导出到指定文件下。

扩展：

*a：系统运行过程中遇到不可避免的硬件错误或系统中断：

1. 在日志文件中记录当前运行状态，系统关闭。
2. 用户应尽快查明外部原因后重启系统恢复错误前状态或重新初始化系统。

发生频率：可能会偶尔发生。

用例 6：跟踪模拟仿真过程

范围： MeshHelper 库

级别：用户目标

主要参与者：领域用户

涉众及其关注点：

- 领域用户：希望能准确、快速、便捷地跟踪到模拟仿真过程中给的每一个状态的 Mesh 对象信息。

前置条件：正确引入本库并能运行程序，已经生成或导入了 mesh 对象，并在此对象上进行操作。

成功保证：获取到对象上每一单位时间下的变化信息，并进行输出。

主成功场景：

1. 用户使用 C++语言正确导入本协助库。并已经创建了一个系统实例对象，生成或导入了 Mesh 对象到内存中。
2. 用户在执行核运算模拟仿真的过程中可选择调用输出信息功能打印相关 Mesh 对象信息。

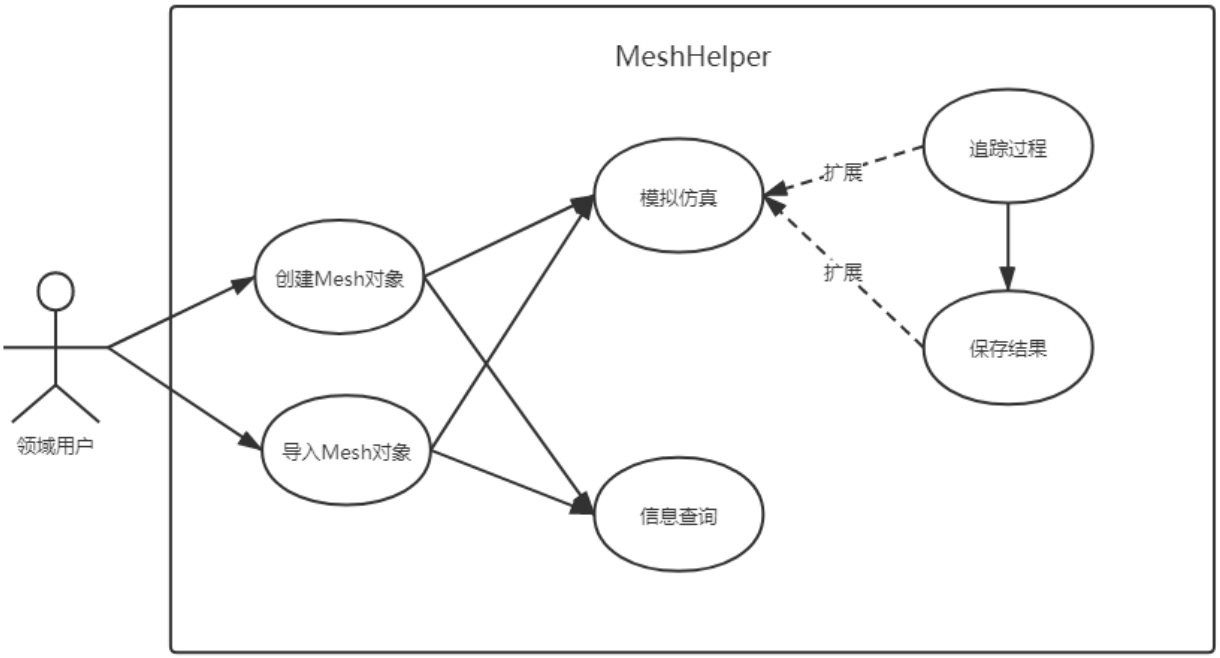
扩展：

*a：系统运行过程中遇到不可避免的硬件错误或系统中断：

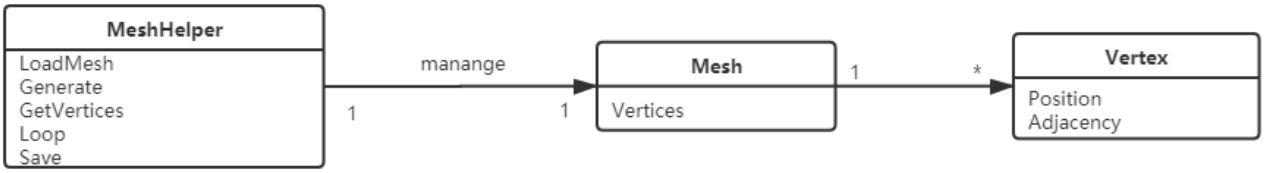
1. 在日志文件中记录当前运行状态，系统关闭。
2. 用户应尽快查明外部原因后重启系统恢复错误前状态或重新初始化系统。

发生频率：可能会偶尔发生。

用例图（Use case diagrams）



领域模型（Domain Model）



补充性规格说明（Supplementary specifications）

简介：

本节记录了未在用例中描述的需求。

功能性

1. 日志和错误处理。

   进行的每一次操作和错误都会被记录在日志文件里。

2. 从内存读取数据。

   读取数据时不一定从磁盘上的文件读取，可以直接从内存读取。

3. 使用该库不应导致程序崩溃。

   如果用户进行了错误操作，应返回相应错误代码并妥善处理错误，不应使程序崩溃。

可用性

1. 该 MeshHelper 库的模型应简单易懂。
2. 代码命名风格保持一致，且用直观的单词命名。
3. 方便用户将该库嵌入到自己的程序。我们将提供动态库文件，用户仅需编译时链接该动态库即可。

可靠性

1. 可恢复性

   如果程序意外关闭，可通过某种手段尽量恢复。

2. 安全性

   如果程序运行出错，不会对原输入文件进行修改。

3. 性能

   在绝大部分仿真用例中，用户等待的时间不会过长。

可支持性

1. 可适应性：所实现的 C++协助库可在多系统环境下运行。

实现约束

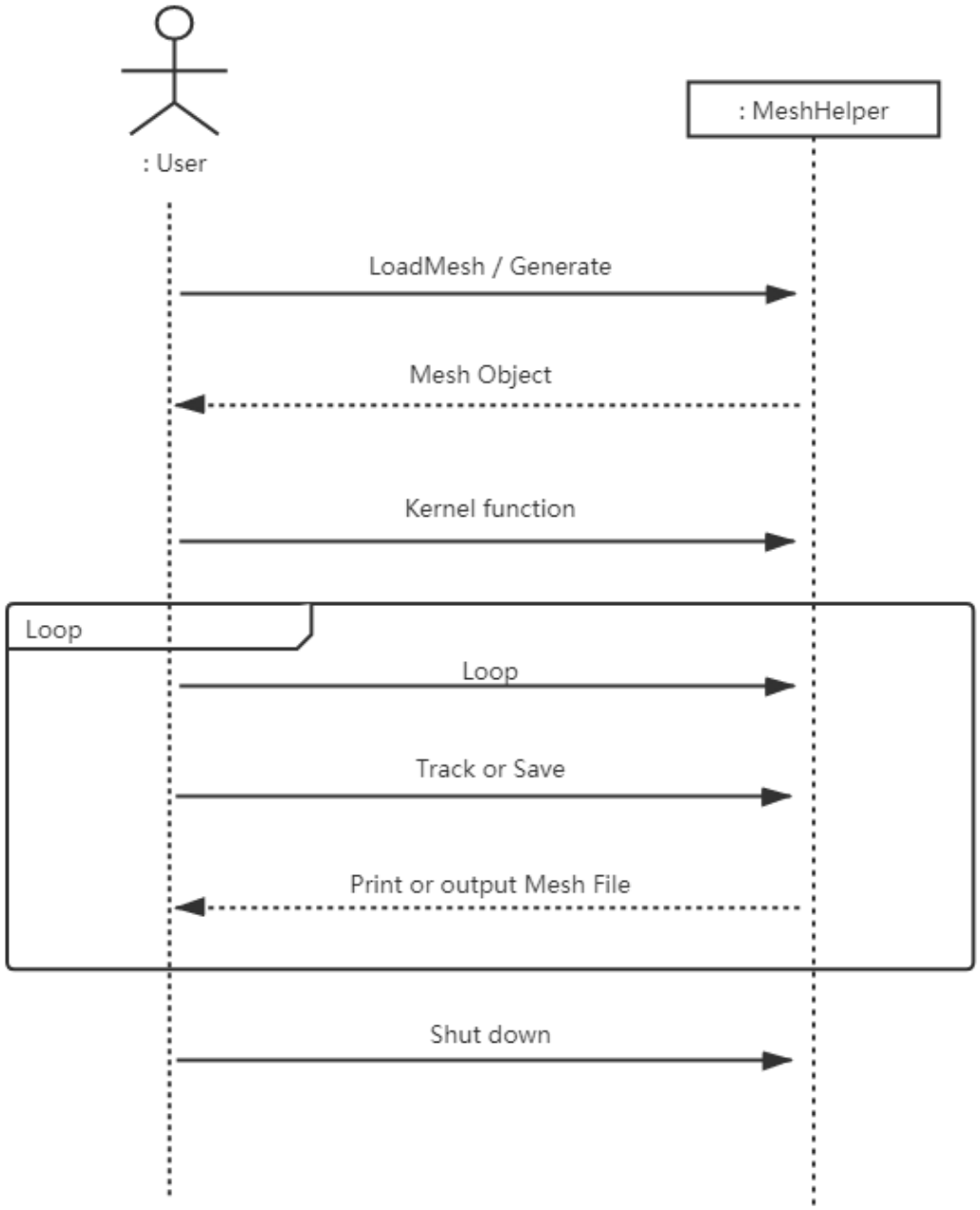
出于性能考虑，我们认为 C++更适合作为本项目的编程语言，本项目不会使用到其他

类型的编程语言。

词汇表（Glossary）

术语	定义和信息
细胞（Cell）	物理模型离散化之后的一块体积单元
节点（Node）	物理模型离散化之后网格的节点
细胞中心（Cell Center）	细胞的中心
表面（Face）	细胞的边界
边缘（Edge）	细胞表面的边界
区域（Zone）	节点、表面、细胞的集合
范围、域（Domain）	节点、表面、细胞域的集合

系统顺序图（SSD）


: User
: MeshHelper
LoadMesh / Generate
Mesh Object
Kernel function
Loop
Loop
Track or Save
Print or output Mesh File
Shut down

操作契约（System Operation Contracts）

契约 1：LoadMesh

操作：LoadMesh(Path : string)

交叉引用：导入 Mesh 对象用例

前置条件：指定的路径存在且为合法 mesh 文件。

后置条件：将一个 MeshHelper 对象绑定了该文件所描述的 Mesh，即将该文件所描述的 Mesh 转化为内存中预先定义的 Mesh 数据结构，后续操作对象皆为该 Mesh 对象。

契约 2：Save

操作：Save(Path : string)

交叉引用：创建 Mesh 用例、模拟仿真核运算用例、导出 Mesh 对象用例。

前置条件：指定的 Path 的每级目录都存在且可访问。

后置条件：将绑定的 Mesh 从内存中保存为符合 Mesh File Format 的文件。

契约 3：Loop

操作：Loop(KernelFunc : void*, infoSize : int)

交叉引用：模拟仿真核运算用例，跟踪模拟仿真过程用例。

前置条件：核运算状态变化方程需要符合我们预先提供的固定格式。

后置条件：通过循环进行状态转化，Mesh 对象的属性随定义的核运算方程合理变化。

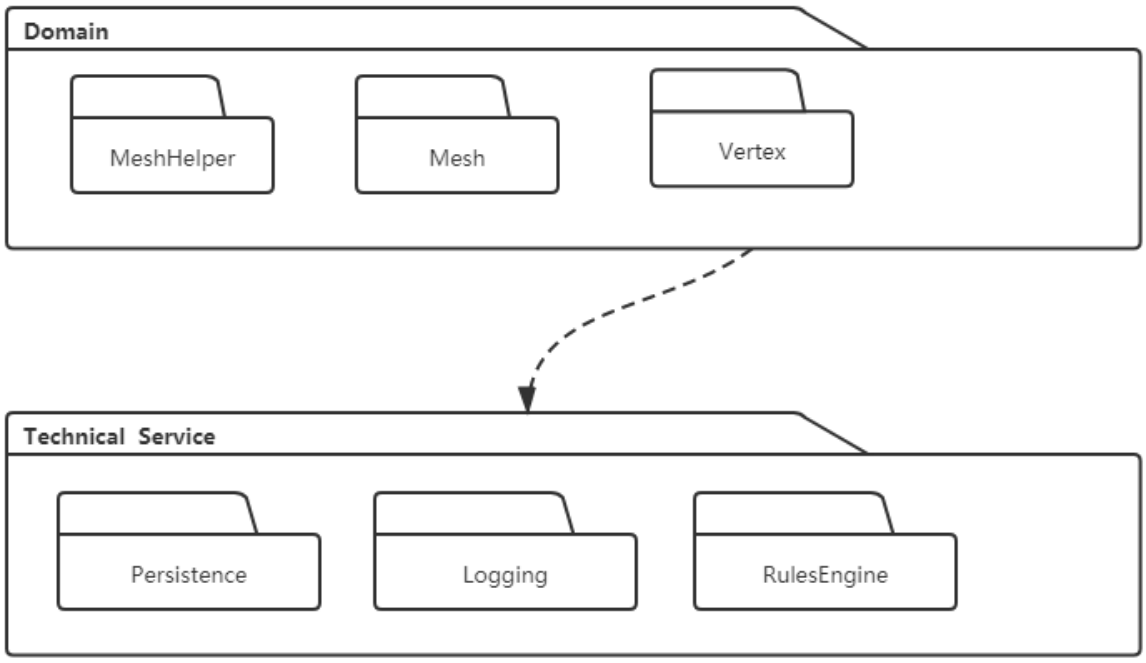
逻辑架构（Logical software architecture）

在本系统设计过程中，逻辑架构由两个层组成，分别是领域对象层和技术服务层。其中领域对象层严格依赖于技术服务层。

领域对象层：表示领域概念的软件对象，这些对象实现了系统需求。

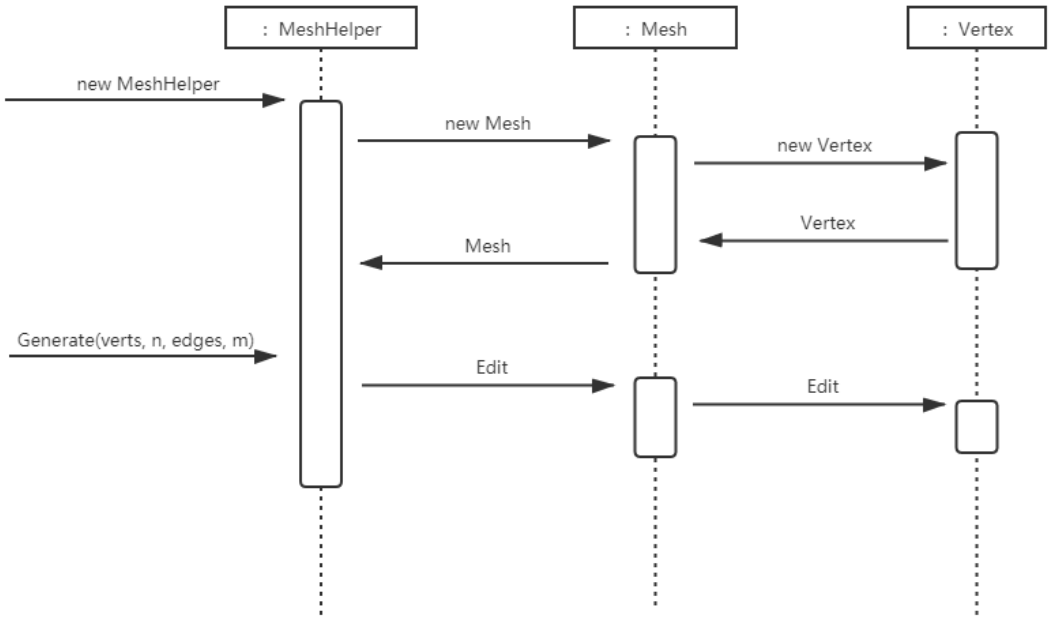
技术服务层：提供支持性技术服务的常用对象和子系统，例如错误日志。这些服务通常独立于应用系统，也可复用。

包图（Packet diagram）

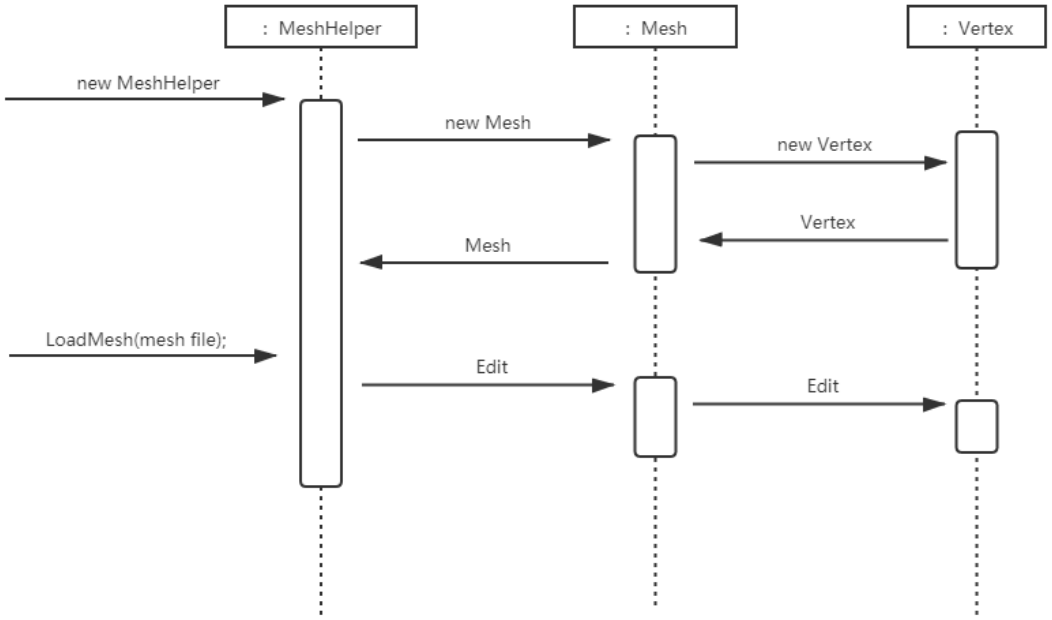


交互图（Interaction diagram）

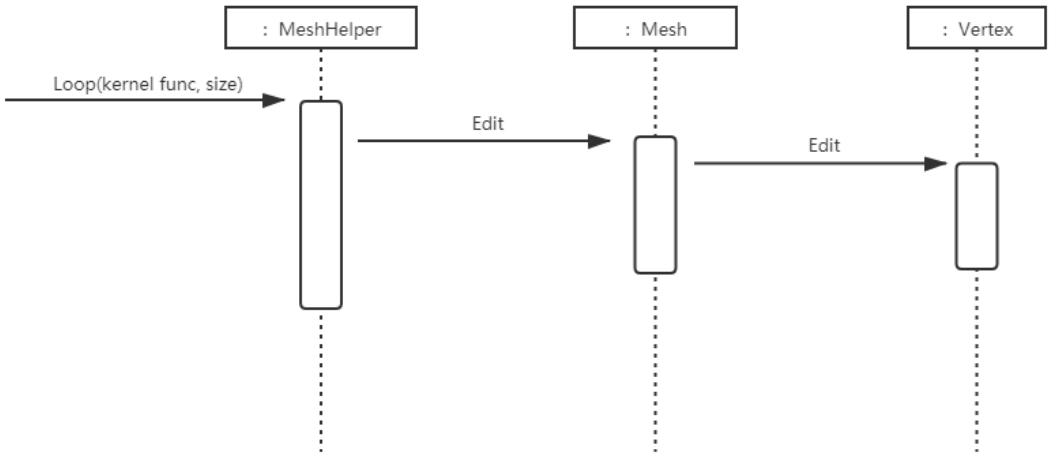
顺序图：创建 Mesh 对象



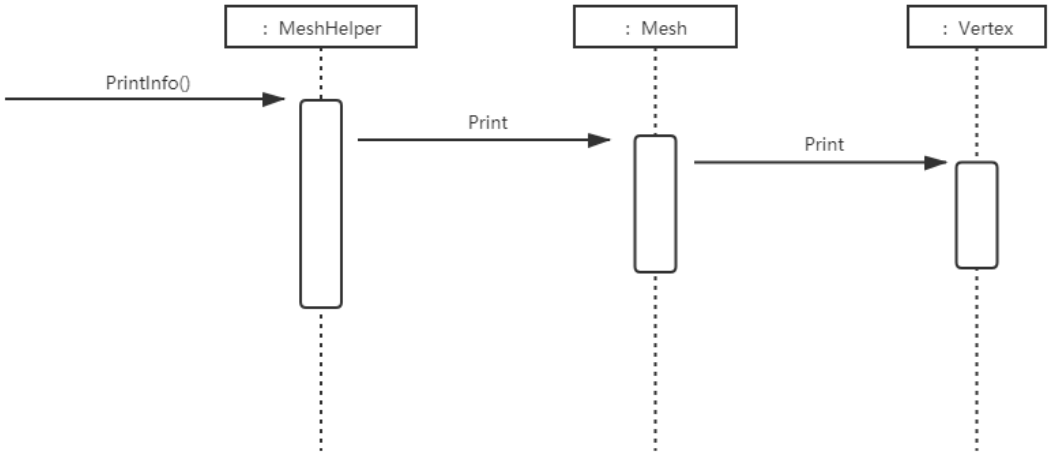
顺序图：导入 Mesh 对象



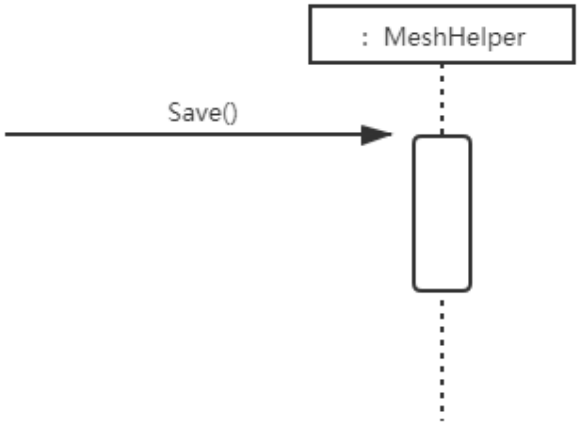
顺序图：模拟仿真核运算



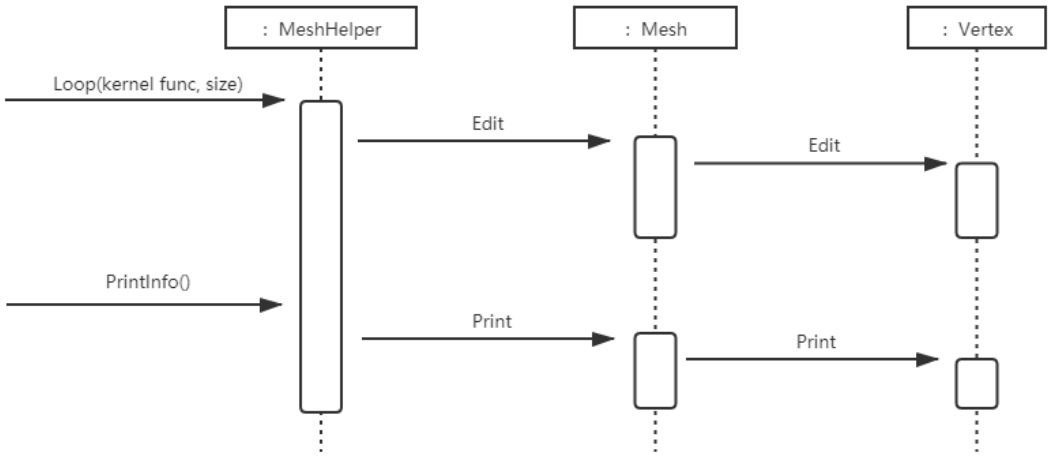
顺序图：输出 Mesh 对象信息



顺序图：导出 Mesh 对象



顺序图：跟踪模拟仿真过程



类图（Class diagram）

使用到的 GRASP 设计模式：创造者（Creator）模式

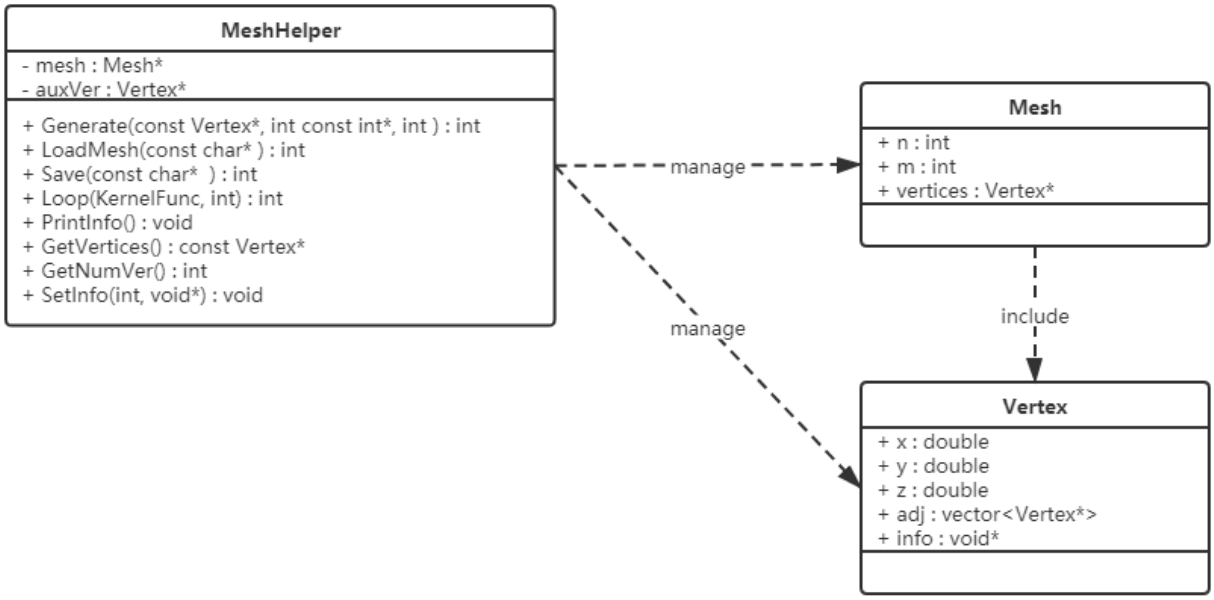
本次 UML 类图建模中，有三个类，分别是 Mesh 系统(MeshHelper)、Mesh 对象(Mesh)、结点（Vertex)。

其中 Mesh 系统可以创造实例包含 Mesh 对象，Mesh 对象可创造实例包含结点 Vertex。

如何解决系统的功能需求问题：

- 用户调用 C++new 功能创造一个 MeshHelper 对象（MeshHelper 系统)。
- 使用 MeshHelper 对象中的 LoadMesh 方法导入 Mesh 文件。
- 使用 MeshHelper 对象中的 Generate 方法生成 Mesh 对象。
- 使用 MeshHelper 对象中的 Save 方法导出 Mesh 对象到 Mesh 文件。
- 使用 MeshHelper 对象中的 Loop 方法模拟仿真核运算。
- 使用 MeshHelper 对象中的 PrintInfo 方法输出 Mesh 对象信息。

Class diagram：


MeshHelper
- mesh : Mesh*
- auxVer : Vertex*
+ Generate(const Vertex*, int const int*, int) : int
+ LoadMesh(const char*) : int
+ Save(const char*) : int
+ Loop(KernelFunc, int) : int
+ PrintInfo() : void
+ GetVertices() : const Vertex*
+ GetNumVer() : int
+ SetInfo(int, void*) : void
manage
manage
Mesh
+ n : int
+ m : int
+ vertices : Vertex*
include
Vertex
+ x : double
+ y : double
+ z : double
+ adj : vector<Vertex*>
+ info : void*