第一次系统与设计分析作业

组号	姓名	学号
0	郑有为	19335286

一、解释名词

1. System development case(系统开发案例)

系统开发案例就是将为项目选择的实践和UP制品编写为简短文档。

2. Project Vision(项目愿景)

它将指明业务目标。这一项目愿景为在软件开发生存期中做出决策提供了相关环境背景,这部分不应包括详细的功能需求和项目计划信息。项目愿景在陈述时将考虑权衡有不同需求客户的看法,它可能有点理想化,但必须以现有的或所期待的客户市场,企业框架,组织的战略方向和资源局限性为基础，需要描述新产品的主要特性和与用户相关的性能列表。

3. Business Case(业务用例)

业务用例刻画的是启动项目的原因，它的关键目的是提供证据和理由来让投资人继续投资。它的表达逻辑是：资源(如金钱或工作)的消耗支持了特定的业务需求。

业务用例可以是全面的，也可以是非正式的。正式的业务用例中可包含：项目的背景、预期的业务利益、考虑的选项(包括拒绝或结转每个选项的理由)、项目的预期成本、差距分析和预期风险，也应该考虑“不做”这个选择，包括“不做”的成本和风险。

我们能从这些信息中得到项目的合理性，一个好的业务案例报告可以为投资决策领域带来信心和责任感，它是在企业分析和业务案例过程中收集的所有信息的汇编。

4. UP Disciplines(UP科目)

是在一个主题域中的一组活动(及相关制品)，例如需求分析中的活动。UP，即Unified Process，统一过程，是一种流行的构造面向对象系统的迭代软件开发过程。UP科目主要包括：业务建模、需求、设计、实现、测试、部署、配置和变更管理、项目管理、环境等等。一个科目的相对工作量通常会在不同的UP阶段有所变化。

5. Iterative and evolutionary development(迭代进化开发)

迭代进化开发是一种与传统的瀑布式开发相反的软件开发过程，它弥补了传统开发方式中的一些弱点，具有更高的成功率和生产率。

与传统开发方法那种大规模的、不频繁的发布(通常以“季度”或“年”为单位)相比，敏捷方法大大提升了发布频率(通常以“天”或“周”为单位)。

在迭代式开发方法中，整个开发工作被组织为一系列的短小的、固定长度(如3周)的小项目，被称为一系列的迭代。每一次迭代都包括了需求分析、设计、实现与测试。采用这种方法，开发工作可以在需求被完整地确定之

前启动，并在一次迭代中完成系统的一部分功能或业务逻辑的开发工作。再通过客户的反馈来细化需求，并开始新一轮的迭代。

6. Project stakeholders(项目利益相关者)

指的是可能会因某个项目的决策，活动或结果而受到影响的个人，团体或组织；又或者是指可能影响到某个项目的决策，活动或结果的个人，团体或组织。

项目利益相关者，是对给定项目感兴趣的实体。这些利益相关者可以在组织内部或外部：

1. 赞助项目，或
2. 在成功完成项目后产生利益或收益；
3. 可能会对项目的完成产生正面或负面的影响。

项目利益相关者的一些举例：

- 项目负责人
- 高级管理人员
- 项目团队成员
- 项目客户
- 资源经理
- 直属经理
- 产品用户群
- 项目测试人员
- 在项目进行过程中受项目影响的任何小组
- 项目完成后受项目影响的任何小组
- 项目分包商
- 项目顾问

7.System Requirements(系统需求)

系统需求是系统级别的所有需求，它们描述了系统为满足利益相关者的需求和要求而应履行的全部功能，并以适当的文字陈述，观点和非功能性需求的组合来表示；后者表示必要的安全性，安全性，可靠性等级别。

系统需求在系统工程中扮演着重要角色，因为它们：

- 形成系统架构]和设计活动的基础。
- 形成系统集成和验证活动的基础。
- 用作验证和利益相关者接受的参考。
- 提供在整个项目中进行交互的各种技术人员之间的交流方式。

利益相关者需求的启发从概念定义开始，并将首先通过访谈和任务分析进行开发。系统定义期间会详细考虑系统要求。正如可追溯性所证明的那样，在实现两者之间的一致性之前，都不能认为两者都是完整的，为此可能需要进行多次迭代。

8.用例模型(Use case Model)

用例是文本形式的情节描述，用来说明某参与者如何使用系统以实现某些目标。例如如果商店是一个系统，那么如何处理顾客的购买请求就是一个用例。

而用例模型则是所有书面用例的集合，也是系统功能性和环境的模型。这种模型可以使得客户和开发人员更好的沟通，开发人员对客户的需求也有更直观的了解，因为用例强调了用户的目标和观点。

二、项目进展

Revise the collection of all your artifacts you have done for your inception phase of your project in the following.

概览

1. 设想
2. 业务规则
3. 词汇表
4. 用例模型

设想

简介

我们希望开发一个面向对象的类库，我们称之为网格应用程序开发系统，该类库能提供网格应用程序编辑、存储、执行、计算、可视化等基本功能。

用户概要

该产品面向的用户为使用c++创建类的开发人员、使用c++类库来建立完整应用程序的开发人员、需要应用网格图进行2D、3D建模的工程师和研究人员。

用户目标

- 医学工作者：研究某种疾病传播网络模型
- 机械工程师：研究热量在某物理设备中的传播
- 建筑师：建立一个建筑模型，测定其受力情况
- 网络管理：控制流量，确定数据流方向

产品概览

在网格应用程序系统中，我们提供文件输入接口和文件数据解析功能，用户可以将网格数据文件导入到程序中，并将这些数据以一定的数据结构存放在内存中。用户可以用一些系统内核功能定义一些函数以便处理数据和对网格进行计算，除此之外用户还可以利用一些算法来对数据结构进行处理。在系统设计中提供用户算法的循环、分支机制，使用户能够灵活定义和处理网格结构。

我们的系统以用户预定义的格式将模拟结果输出到一些处理工具。这些工具可以将模拟结果可视化并展示。

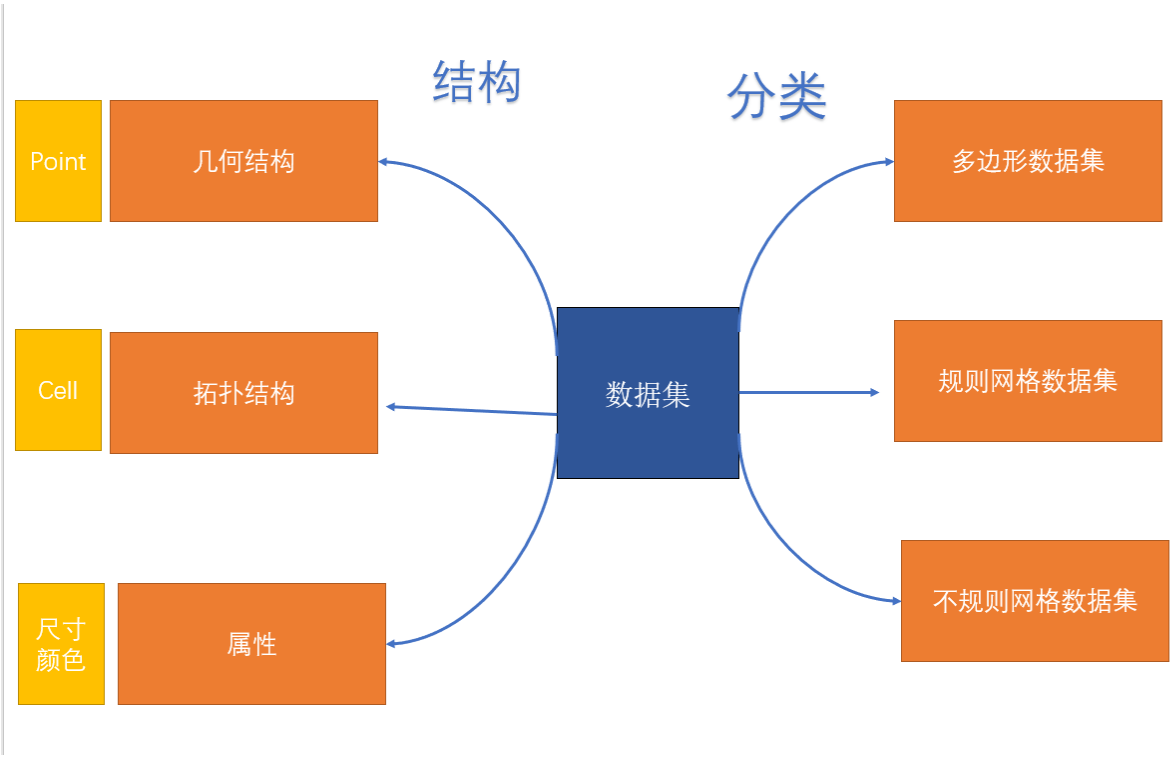
我们的系统提供基本的存储、IO等功能，支持某些平台和系统（windows/Linux、c/c++/python)下的网格应用程序的编程和执行。我们设计的系统允许用户用C/ c++之类的编程语言创建网格应用程序。并为用户程序提供一个执行环境。实现这个环境的工作主要是为网格程序实现一个运行时类库。

我们的系统可以应用于计算机图形学、图像处理和三维可视化等领域，独立于系统的图形界面接口，可嵌入到其它相关软件中。同时开发人员可以基于系统的独立类库开发自己的库函数，拓展应用范围。

系统特性概要

- 在windows/Linux下运行
- 允许使用c/c++/python进行编程
- 用户将文本文件、二进制文件作为输入，得到网格数据
- 以一种高效的数据结构在内存中存储网格结构
- 用户可以在代码中定义函数、算法，对网格做运算处理
- 系统提供内核循环、分支操作
- 系统由c/c++实现，可以在多核CPU上运行
- 系统允许用户跟踪进程性能

数据结构概况



业务规则（领域规则）

规则列表

ID	规则	可变性	来源
规则1	mesh类库是开源的放在gitee，但使用时需要声明原创及附上链接。	中，如果传播性高则不需要附上链接	专利权
规则2	存储mesh时需要由user确认存储格式，否则自动保存为text格式。	低	

ID	规则	可变性	来源
规则3	user不允许对程序进行破坏性攻击行为。	低	程序安全
规则4	我们应该为域程序员提供一个域应用程序模板。网格应用程序通常由网格声明、网格元素上的核函数定义、网格的所有结构上的操作和一些io操作组成	中，有时候用户会自己定义另一种操作	网格的基本操作
规则5	所有的$结束标记都必须在一个新的行上开始	低	格式要求
规则6	类库应该在网格上实现内核操作的循环	低	类库要求
规则7	类库应该输出到文本（或二进制)文件，以某些用户定义或预定义格式保存计算(模拟）结果。	低	类库要求

词汇表

术语	定义和信息	格式/验证规则	别名
MADS	Mesh Application Development System		网格应用程序开发系统
静态网格数据	该网格点、边等信息在定义之后不会再被改变，类似于一个常量		
动态网格数据	该网格点、边等信息在定义之后可以在程序执行过程中被改变，类似于一个变量		
核函数	若支持向量机的求解只用到内积运算，而在低维输入空间又存在某个函数 K(x, x′)，它恰好等于在高维空间中这个内积，那么这样的函数 K(x, x′) 称为核函数	支持向量机通过某非线性变换 φ(x)将输入空间映射到高维特征空间,而低维输入空间又存在某个函数K(x, x′)使得：K(x, x′) = <φ(x)·φ(x′) >	径向基函数（Radial Basis Function 简称 RBF)
聚类	在此开发过程中主要指基于网格的聚类，即利用属性空间的多维网格数据结构，将空间划分为有限数目的单元以构成网格结构		
错误代码	提供出错原因、位置	error:cause/filename/position	

用例模型

1. 系统参与者

1.1开发人员：调用MADS类库进行编程的人员

1.2类库：指提供底层Mesh操作的类库

2. 用例概览

版本与配置

- 2.1. MADS的版本与配置
    - 2.1.1 本机配置检查
        - 解释：例如检测当前OS架构和版本、编译器版本、是否安装/安装了哪些可用的 网格可视化程序和其他MADS需要使用的第三方类库或软件、网络的状况。
    - 2.1.2 版本更新检查
        - 解释：启动时检查MADS版本，询问是否升级（考虑到后续可能会打补丁什么的）

使用帮助

- 2.2. MADS的使用帮助
    - 2.2.1 查询使用教程（帮助文档）
        - 解释：虽然听起来怪怪的，但是对于第一次使用MADS的用户，这个过程是应该包含进"用户的软件使用过程中"。
    - 2.2.2 查询选项帮助
        - 解释：例如Linux上的很多软件会提--help，选项指程序编译、链接、运行等过程中使用的-参数。

使用日志

- 2.3 用户使用日志的记录和查询
    - 解释：例如用户使用此功能查询上次操作的文件、异常关闭前文件的状态。

数据文件操作

- 2.4. MADS与网格数据文件
    - 2.4.1 检查网格数据文件内容的（格式）正确性
    - 2.4.2 将特定字节串转换为特定数据类型
    - 2.4.3 以给定格式读取网格数据文件并写入内存

    - 2.4.4 以给定格式将内存中网格数据写入文件

数据可视化

- 2.5. MADS与可视化（似乎是专业网格可视化软件的工作）
    - 2.5.1 显示静态网格数据
    - 2.5.2 显示动态网格数据
        - 解释：例如一块材料温度随时间变化的热力图。

基本数据结构和基本操作

- 2.6. MADS类库操作（或许根本没有基本数据类型，所有网格数据的建立都是用户用Set、Map、Dat等定义的）
    - 2.6.1 使用系统提供的基本网格数据类型（2D/3D）进行建模（正方体、三角形、球等）
    - 2.6.2 使用MADS提供的通用函数库操作数据
        - 2.6.2.1 平移
        - 2.6.2.2旋转
        - 2.6.2.3 对称翻转
        - 2.6.2.4 缩放
        - 2.6.2.5 合并点
        - 2.6.2.6删除点、边、面和体
        - 2.6.2.7 增加点、边、面和体
    - 2.6.3 使用者使用类库的模板自定义一种网格数据结构。
        - 解释：自定义一种不在基本类型中的单元结构，可以是继承于基本结构、可以是基本结构的组合，也可以直接通过定义Set和Map等来创建结构
    - 2.6.4 使用者定义基于网格的操作（查找，遍历，比较，运算等）、核函数和特定算法（求权，排序，聚类等）
    - 2.6.5 使用者获取Mesh基本的属性数值
        - 2.6.5.1计算面的数量
        -
        - 2.6.5.2 点的度数

3. 系统用例

UC1：以给定格式读取网格数据文件并写入内存(2.4.1)

- 主要参与者：开发人员、类库
- 前置条件：格式合法且网格数据文件存在
- 主成功场景：

1. 开发人员调用类库的载入函数，传进参数：文件绝对地址、格式类型。
2. 查询文件是否存在，存在。
3. 尝试打开文件，成功。
4. 本函数调用同类中的确认文件格式合法的函数，确认文件格式无误。
5. 从数据段开始读取。
6. 读入一个字节串，将其作为参数调用同类中的串转换函数，并将返回值赋给类中对应成员。
7. 重复5.直至读入结束符。
8. 关闭文件。
9. 返回成功值给调用函数

- 扩展：
    - a.文件访问失败 处理过程：
        1. 调用stat函数查询文件状态，获得对应的错误代码。
        2. 根据错误代码返回对应的错误代码给开发人员。
    - b.文件打开失败 处理过程：
        1. 返回对应的错误代码给开发人员
    - c.格式错误 可能情况：
        1. 指定的格式和文件格式不符。
        2. 文件格式部分已经被破坏
        3. 指定的格式不合法 处理过程： 1.返回对应错误代码给开发人员
    - d.数据出错 可能情况：
        1. 数据段残缺
        2. 非法比特串 处理过程：
        3. 记录下路径、文件名、出错位置行标号以及错误类型
        4. 打印出错误信息
        5. 返回错误代码给调用者

UC2: 平移(2.6.2.1)

- 主要参与者：开发人员、类库
- 主成功场景：
    1. 编程者确定/选中需要平移的Mesh主体。
    2. 编程者输入平移的主体、主体维度信息、平移的方向以及距离等参数，并调用系统类库中的平移函数。
    3. 系统查询类库，并对被操作主体执行平移操作。
    4. 平移成功，修改后的主体被返回，控制台打印出操作成功的信息。

UC3: 旋转(2.6.2.2)

- 主要参与者：开发人员、类库
- 主成功场景：
    1. 编程者确定/选中需要旋转的Mesh主体。
    2. 编程者输入旋转的主体、主体维度信息、旋转的方向以及角度等参数，并调用系统类库中的旋转函数。

3. 系统查询类库，并对被操作主体执行旋转操作。
4. 旋转成功，修改后的主体被返回，控制台打印出操作成功的信息。

UC4: 对称翻转(2.6.2.3)

- 主要参与者：开发人员、类库
- 主成功场景：

1. 编程者确定/选中需要翻转的Mesh主体。
2. 编程者输入旋转的主体、主体维度信息、翻转的对称轴位置、方向等参数，并调用系统类库中的翻转函数。
3. 系统查询类库，并对被操作主体执行翻转操作。
4. 翻转成功，修改后的主体被返回，控制台打印出操作成功的信息。

UC5: 缩放(2.6.2.4)

- 主要参与者：开发人员、类库
- 主成功场景：

1. 编程者确定/选中需要缩放的Mesh主体。
2. 编程者输入缩放的主体、主体维度信息、缩小或放大的选择、缩放中心点的位置、缩放倍率等参数，并调用系统类库中的缩放函数。
3. 系统查询类库，并对被操作主体执行缩放操作。
4. 缩放成功，修改后的主体被返回，控制台打印出操作成功的信息。

UC6: 合并点(2.6.2.5)

- 主要参与者：开发人员、类库
- 主成功场景：

1. 编程者确定/选中需要合并的点集。
2. 编程者输入该点集、点集所在的维度信息、合并后点的位置等参数，并调用系统类库中的合并点函数。
3. 系统查询类库，并对被选中的点集执行合并操作。
4. 合并成功，返回合并点，控制台打印出操作成功的信息。

UC7: 删除点、边、面和体(2.6.2.6)

- 主要参与者：开发人员、类库
- 主成功场景：

1. 编程者确定/选中需要删除的点、边、面或体（为方便描述，统称为被删除主体）
2. 编程者输入被删除主体、该主体所在的维度信息等参数，并调用系统类库中删除主体的函数。
3. 系统查询类库，并对被删除主体执行删除操作。
4. 删除成功，控制台打印出操作成功的信息。

UC8: 增加点、边、面和体(2.6.2.7)

- 主要参与者：开发人员、类库
- 主成功场景：

    1. 编程者确定/选中需要增加的点、边、面或体（为方便描述，统称为新添主体）。
    2. 编程者输入新添主体、该主体所在的维度信息、主体将要放置的位置和方位信息等参数，并调用系统类库中新添主体的函数。
    3. 系统查询类库，并对新添主体执行添加操作。
    4. 添加成功，控制台打印出操作成功的信息。

UC9: 计算面的数量(2.6.5.1)

- 主要参与者：开发人员、类库
- 主成功场景：

    1. 编程者确定/选中需要计算的主体（维度在2维及以上，包含一个或多个面）。
    2. 编程者输入主体、主体所在的维度信息等参数，并调用系统类库中计算面数量的函数。
    3. 系统查询类库，并根据主体内部数据结构信息执行计算其面的数量的操作。
    4. 返回计算结果，该结果输出到编程者指定的文件中或者直接输出到控制台。

UC10: 点的度数(2.6.5.2)

- 主要参与者：开发人员、类库
- 主成功场景：

    1. 编程者确定/选中需要计算的主体（包含一个或多个顶点）。
    2. 编程者输入主体、主体所在的维度信息等参数，并调用系统类库中计算点的度数的函数。
    3. 系统查询类库，并执行计算主体点的度数的操作。
    4. 返回计算结果，该结果输出到编程者指定的文件中或者直接输出到控制台。
