OCAF GPT

Chen Chen

January 16, 2024

Contents

1 Introduction

1.1 Overview

Open CASCADE Technology (OCCT) 是一个开源的软件开发平台，用于三维 CAD、CAM、CAE 系统的开发。它提供了广泛的功能，涵盖了几何建模、图形可视化、数据交换和更多方面。

Open CASCADE Application Framework (OCAF) 是 OCCT 中的一个重要模块。OCAF 是一个应用程序框架，用于简化复杂工程图形应用程序的开发。它提供了一种有效的方式来组织、存储、检索和操作复杂的工程数据。OCAF 特别适合于需要处理复杂的装配结构、历史记录、参数化设计等场景的应用程序。

1.2 OCAF 的主要功能

- 数据管理
  - OCAF 提供了一套工具来有效地管理和组织数据。
  - 这包括用于创建、管理和修改数据结构的 API
- 历史记录和撤销/重做机制

  - OCAF 支持记录用户的操作历史，使得可以方便地实现撤销和重做功能。

- 属性和关系管理
  - OCAF 允许开发者为数据元素定义属性（如颜色材料等），并管理数据元素间的关系。

- 事务管理
  - OCAF 支持事务管理，这对于保证数据的一致性和完整性非常重要。

- 扩展性
  - OCAF 设计灵活，易于扩展，开发者可以根据特定应用需求添加新的功能。

通过 OCAF，开发者可以更专注于应用的核心功能，而不是底层的数据管理和操作。

1.3 OCAF 的主要 Packages

下面列出了一些在 OCAF 中常用且重要的 packages:

- **TDF (Topological Data Framework)**
  - 用于管理和存储拓扑数据的结构和信息。
  - TDF 提供了一个层次化的数据组织方式，通过 `Label`, `Attribute` 等来存储和管理数据

- **TDocStd (Document Standard)**
  - `TDocStd` 提供了创建、管理和保存文档的功能
  - 一个文档可以包含一个或多个 TDF 数据结构

- **XCAF (eXtended CA Framework)**
  - 用于更高级的 CAD 数据处理，如装配结构、颜色和层次信息。
  - XCAF 扩展了 OCAF 的功能，使其能够处理更复杂的 CAD 模型和数据

- **TNaming (Naming)**

  - 提供了一个命名服务，用于在模型中标识和追踪对象。
  - `TNaming` 使得在模型变更过程中可以保持对特定对象的引用。

- **TPrsStd (Presentation Standard)**
  - 用于关联数据模型和其图形表示
  - `TPrsStd` 允许开发者定义如何将模型数据转换为可视化的图形表示

- **TDataStd (Data Standard)**
  - 包含了一系列的 Attribute 类型，如字符串、整数、实数、枚举类型等。
  - TDataStd 提供了基础的数据类型，用于存储和处理常规属性。

- **AppStdL (Application Standard Library)**
  - 提供了一组标准的应用程序功能和服务，如历史管理、撤销/重做机制等。

- **BinTObj**
  - 用于持久化存储和加载 OCAF 对象的包。
  - 支持二进制格式，适用于大型数据集。

- **XmlTObj**
  - 类似于 BinTObj，但用于处理基于 XML 的持久化存储和加载。

1.4 如何学习掌握 OCCT 及 OCAF

- 基础了解
  - 了解 OCCT 的基本概念，包括主要组件、功能和应用场景
  - 熟悉 OCAF 模块的基本概念，如 Label, Tag, Attribute 等。

- 阅读官方文档

- 学习示例代码
  - 查看和分析 OCCT 提供的示例代码，尤其是设计 OCAF 的示例。
  - 通过理解代码，你可以更好地了解如何在实际项目中使用 OCAF 模块。

- 小型项目实践
  - 开始一个小型的项目，使用 OCAF 模块来实现一些基本功能。比如一个简单的 CAD 工具或任何需要数据组织和管理的应用。
  - 在实践中尝试创建、修改和管理 Label、Tag 和 Attribute，以及处理事务和历史记录。
- 深入学习高级特性
  - 当你对基本功能有一定理解后，开始学习 OCAF 的高级特性，如复杂的数据关系管理、历史版本控制、自定义属性类型等。
- 参与社区讨论
- 查阅相关书籍和资源
- 实际项目经验

2 TDF Package

2.1 Overview

TDF (Topology Data Framework) 是 OCAF 的核心组件，用于管理和组织复杂的工程数据（其中拓扑数据是几何建模的基础）。TDF 提供了一个结构化的方式来存储和操作与拓扑相关的信息，如点、线、面、实体等几何元素及其之间的关系。

2.2 TDF 的功能与职责

- 数据组织

  TDF 提供了一种层次化的数据结构，使得对复杂拓扑数据的管理和访问更加直观灵活。

- 事务管理

  通过 TDF，可以实现对拓扑数据的事务管理，支持撤销/重做操作，保证数据一致性。

- 属性管理

  TDF 允许为拓扑元素附加属性（颜色材料等），并管理这些属性。

- 关系管理

  TDF 支持管理拓扑元素之间的关系，如约束、连接等。

- 版本控制

  TDF 支持数据的版本控制，这对于跟踪数据的历史变更非常有用。

- 灵活性和扩展性

  TDF 设计灵活，易于扩展，可以根据特定的应用需求进行定制。

2.3 TDF 的主要接口及功能

- `TDF_Label` class

  功能: 代表数据结构中的一个节点，可包含多个 sub-Labels 和 Attribute。

  主要接口: `FindChild`, `NewChild`, `HashAttribute`, `AddAttribute`, `FindAttribute`, `ForEach` 等。

  - `FindChild` 查找或创建 sub-label
  - `NewChild` 创建一个新的 sub-label
  - `HasAttribute` 检查是否存在特定类型的 Attribute
  - `AddAttribute` 添加一个新的 Attribute
  - `FindAttribute` 查找特定类型的 Attribute

- `TDF_Attribute` class

  功能: 附加在 Label 上的数据单元，用于存储特定类型的信息，如几何数据、颜色、文本等。

  主要接口: `Set`, `Get`, `NewEmpty`, `Restore`, `Paste` 等

  - `NewEmpty` 创建一个新的空 Attribute 实例
  - `Restore` 从备份中恢复 Attribute 的状态
  - `Paste` 复制或转移 Attribute 的内容

- `TDF_Data` class

  功能: 代表整个数据集合，包含一个或多个 `TDF_Label` 树

  主要接口:

  - `Root` 获取数据几何的 root label
  - `TransactionStart`, `TransactionCommit` 开始和提交事务
  - `Undo`, `Redo` 支持撤销和重做操作

- `TDF_TagSource` class

  功能: 用于自动生成唯一的 Tag (标签号)。

  主要接口: `NewTag` 生成一个新的唯一 Tag。

- `TDF_RelocationTable` class

  功能: 在数据复制和粘贴操作中使用，管理 Label 和 Attribute 之间的关系映射。

  主要接口:

  - `SetRelocation` 设置新旧 Label 或 Attribute 之间的映射
  - `HasRelocation` 检查是否存在特定的映射
  - `Relocation` 获取映射的目标

3 TDocStd Package

3.1 Overview

`TDocStd` 主要用于处理和管理文档 (Document)，这些文档用于存储和组织复杂的 CAD 数据结构。一个文档通常代表一个工程项目或一个 CAD 模型，它包含了所有相关的数据和信息。`TDocStd` 提供了一套工具和接口来创建、管理和存储这些文档。

3.2 TDocStd 的功能与职责

- 文档管理

  `TDocStd` 提供了创建和管理文档的基本机制。文档可以包含多种类型的数据，如几何形状、装配信息、属性等。

- 文档结构

  文档中的数据通过 OCAF 的 `TDF_Label` 结构进行组织。每个文档都有一个 root Label, 从 root Label 开始可以创建一个层次化的数据结构。

- 事务管理

  `TDocStd` 支持事务管理，允许用户对文档进行修改操作，同时支持 Undo/Redo 功能。这对于保持数据的一致性和完整性至关重要。

- 存储和加载

  TDocStd 提供了将文档保存到文件系统和从文件系统加载文档的功能。支持多种格式，包括自定义格式。

- 版本控制

  文档可以支持版本控制，允许跟踪文档的历史变更。

- 扩展性

  TDocStd 的设计允许开发者根据需要扩展和定制文档的功能，以适应特定的应用需求。

3.3 TDocStd 与 TDF package 的关系

- TDocStd 依赖于 TDF 来组织文档内的数据。

  每个 TDocStd_Document 包含一个根 TDF_Label, 这个 root label 是文档所有数据的起点。通过 root label, 可以访问和操作文档中的所有数据。

- 在 TDF 基础上，TDocStd 提供了文档级别的管理，如创建/保存/加载文档、事务处理 (Undo/Redo) 等。

3.4 TDocStd 的主要接口及功能

- TDocStd_Document class

  功能: 代表一个文档，是管理和组织 CAD 数据的主要实体。

  主要接口:

  - NewCommand() 开始一个新的命令或操作
  - CommitCommand() 提交当前命令，使其更改称为文档的一部分
  - Undo(), Redo() 撤销和重做
  - Save(), Open() 文档的存储和加载

- TDocStd_Application class

  功能: 处理文档的创建、加载和保存，管理文档集合。

  主要接口:

  - NewDocument() 创建一个新的文档
  - SaveAs(), Open() 保存和打开文档

  - `GetFormats()` 获取支持的文档格式列表
  - `Close()` 关闭文档

- `TDocStd_Owner` class

  功能: 作为文档所有者的角色，管理文档的状态和事务。

  主要接口:

  - `SetDocument` 设置或关联文档
  - `BeforeUndo`, `AfterUndo` 撤销操作前后的处理函数。

4 XCAF Package (属于 DataExchange Module)

4.1 Overview

XCAF (eXtended CA Framework) 用于处理更高级别的 CAD 数据，尤其是那些涉及到复杂装配结构的数据。XCAF 提供了一些列工具和接口，用于管理和操作包括颜色、材料、元数据、层级关系等在内的复杂 CAD 模型数据。

4.2 XCAF 主要功能与职责

- 复杂装配结构管理

  XCAF 提供了工具来创建和管理复杂的 CAD 装配结构，包括定义装配体、子装配体和零件之间的层级关系。

- 颜色和图层管理

  支持为模型的不同部分指定颜色和图层，帮助改善模型的可视化和组织。

- 高级属性管理

  XCAF 允许为模型元素添加和管理高级属性，如材料属性、PMI(产品和制造信息)、注释和元数据。

- 形状标识和追踪

  提供工具来唯一标识和追踪模型中的形状，尤其在模型的变更或更新过程中，保持对特定形状的引用。

- 数据交换支持

  支持与其他 CAD 系统间的数据交换，特别是在处理 STEP 和 IGES 文件格式时，能够导入和导出中配信息和属性。

- 扩展性和定制
  XCAF 设计灵活，可以根据特定应用需求进行扩展和定制。

4.3 XCAF 的主要接口与功能

- `XCAFDoc_ShapeTool` class
  功能: 用于管理装配结构和形状。
  主要接口:
  - `GetShape` 获取形状
  - `AddShape` 添加新的形状
  - `GetSubShapes`, `GetSubShapeExt` 获取子形状
  - `GetAssembly` 获取装配体
- `XCAFDoc_ColorTool` class
  功能: 管理颜色属性
  主要接口:
  - `SetColor` 为形状设置颜色
  - `GetColor` 获取形状的颜色
  - `RemoveColor` 移除形状的颜色
- `XCAFDoc_LayerTool` class
  功能: 管理图层属性
  主要接口:
  - `SetLayer` 为形状设置图层
  - `GetLayers` 获取形状的图层
- `XCAFDoc_MaterialTool` class
  功能: 管理材料属性
  主要接口:
  - `SetMaterial` 为形状设置材料
  - `GetMaterial` 获取形状的材料

- XCAFDoc_DatumTool, XCAFDoc_DimTolTool classes

  功能: 管理标注和公差。

  主要接口:

  - AddDatum, AddDimTol 添加新的标注或公差
  - GetDatum, GetDimTol 获取标注或公差

- XCAFDoc_AreaStyleTool class

  功能: 管理区域样式

  主要接口

  - SetAreaStyle 为形状设置区域样式
  - GetAreaStyle 获取形状的区域样式

5 TNaming package

5.1 Overview

TNaming 提供了命名服务，以便在复杂的 CAD 模型和数据结构中标识和追踪对象。这对于在模型变更过程中保持对特定对象的引用非常重要。

5.2 TNaming 的主要功能与职责

- 对象标识和追踪

  TNaming 允许用户为模型中的对象 (如形状、特征等) 赋予唯一的名称，从而在整个模型的生命周期中追踪和引用这些对象。

- 历史追踪

  支持记录和跟踪对象随时间的变化。这使得即使在模型被修改或更新后，也能够识别和访问原始对象。

- 版本控制

  TNaming 提供了一种机制来处理模型中对象的版本控制，保证在多次修改和迭代中对象的一致性。

- 复杂操作支持

  对于复杂的操作（如布尔运算、分割、修剪等）, TNaming 能够帮助保持对影响的对象的引用，确保数据的准确性和完整性。

- 与 TDF 协同工作

  TNaming 与 TDF 紧密协作，利用 TDF_Label 和 TDF_Attribute 来存储和管理命名信息。

- 撤销/重做机制支持

  支持与 OCAF 的撤销/重做机制结合使用，确保在运行这些操作时保持命名信息的一致性。

5.3 TNaming 的主要接口与功能

- TNaming_NamedShape class

  功能: 用于关联形状 (Shape) 与名称，实现形状的命名和追踪。

  主要接口

  - Get 获取与名称关联的形状
  - Set 设置或更新名称与形状的关联

- TNaming_Builder class

  功能: 用于构建和修改命名关系

  主要接口

  - Select 为给定的形状选择或创建一个名称
  - Generate 生成一个新的名称

- TNaming_Tool class

  功能: 提供一系列静态方法来操作和查询命名信息

  主要接口

  - GetShape 根据名称获取形状
  - GetLabel 获取与特定形状关联的标签

- TNaming_Naming class

  功能: 存储和管理命名操作的历史记录。

  主要接口

  - GetName 获取命名操作的名称
  - GetShapes 获取命名操作影响的形状列表

- TNaming_NamingTool class

  功能: 提供用于执行复杂命名操作的高级方法

  主要接口

  - Solve 解决命名冲突
  - LoadNamedShapes 加载命名形状

6 TPrsStd package

6.1 Overview

TPrsStd package 用于将工程数据（如存储在 OCAF 文档中的数据）与其图形表示相关联。它为开发者提供了一系列工具和接口，以便在图形界面中展示和交互复杂的工程模型。

6.2 TPrsStd 主要功能与职责

- 图形表示管理

  TPrsStd 使得开发者可以将工程数据（如形状、属性等）与其在图形用户界面中的视觉表示相关联。这包括形状的渲染、颜色、纹理等。

- 交互和选择支持

  提供了工具来支持用户与图形表示的交互，包括选择、高亮显示和编辑操作。

- 属性与视觉同步

  确保工程数据的更改能够实时反映在图形表示上，例如当形状发生变化时，其视觉表示也会相应更新。

- 高级显示功能

  支持高级的显示功能，如透明度、阴影和纹理映射，使得工程模型的视觉表示更加逼真和详细。

- 自定义显示属性

  允许开发者定义自己的显示属性和表示方式，以满足特定应用的需求。

- 与 OCAF 结合使用

  TPrsStd 与 OCAF 的其他组件（如 TDF_Label, TDF_Attribute）紧密集成，使得开发者可以方便地管理和同步数据与其图形表示。

- 支持多种渲染引擎

  可以与 OCCT 提供的不同渲染引擎（如 OpenGL）协同工作，提供高质量的图形输出。

6.3 TPrsStd 的主要接口与功能

- `TPrsStd_AISPresentation` class

  功能: 用于管理工程数据的图形表示，如形状在图形界面中的显示。

  主要接口

  - `SetDriver` 设置用于显示的驱动程序。
  - `Update` 更新图形表示以反映数据的更改。
  - `Display`, `Erase` 控制对象的显示和隐藏。

- `TPrsStd_AISViewer` class

  功能: 提供一个视图环境，用于显示和管理多个图形表示。

  主要接口

  - `Update` 更新视图中的所有表示
  - `SetView` 设置或更改关联的视图

- `TPrsStd_Presentation` class

  功能: 作为数据和其图形表示之间的桥梁。

  主要接口

  - `Set` 关联数据和图形表示。
  - `Get` 获取与数据关联的图形表示。

- `TPrsStd_Driver` class

  功能: 为具体的数据类型提供图形表示的生成和更新逻辑。

  主要接口

  - `Update` 根据数据更新图形表示。
  - `Create` 根据给定的数据创建新的图形表示。

7 TDataStd package

7.1 Overview

TDataStd 主要提供了一系列标准的数据属性 (Attributes)，这些属性可以附加到 OCAF 文档中的 Labels 上，用于存储和管理各种类型的数据。

7.2 TDataStd 主要功能与职责

- 基本数据类型的管理

  TDataStd 提供了用于存储基本数据类型（如字符串、整数、实数、布尔值等）的属性。这些属性用于存储和检索与标签相关联的基本信息。

- 集合和列表的管理

  提供了管理数据集合（如数组、列表）的属性，用于存储多个数据项。

- 命名和标识符管理

  支持为标签分配名称和标识符，方便数据的识别与引用。

- 枚举和状态管理

  提供了用于管理枚举值和状态的属性，可以用于表示有限的选择集或状态机。

- 文档的元数据管理

  支持存储文档级别的元数据，如作者、版本信息、注释等。

- 与 TDF_Label 结合使用

  TDataStd 的属性与 TDF_Label 紧密集成，使得数据可以方便地附加到标签上，并在 OCAF 文档的层次化结构中进行管理。

7.3 TDataStd 的主要接口与功能

- TDataStd_Integer
  - 功能: 用于存储和管理整数值
  - 主要接口: Set, Get
- TDataStd_Real
  - 功能: 用于存储和管理实数值
  - 主要接口: Set, Get

- TDataStd_String
  - 功能: 用于存储和管理字符串
  - 主要接口: Set, Get
- TDataStd_UAttribute
  - 功能: 作为用户自定义数据的基类，可以派生出用于存储特定类型数据的类。
  - 主要接口: SetID 设置属性的唯一标识符
- TDataStd_Name
  - 功能: 用于存储和管理对象的名称
  - 主要接口: Set, Get
- TDataStd_Boolean
  - 功能: 用于存储和管理布尔值
  - 主要接口: Set, Get
- TDataStd_Enum
  - 功能: 用于存储和管理枚举值
  - 主要接口: Set, Get

8 AppStdL package

8.1 Overview

AppStdL (Application Standard Library) 包提供了标准化的应用程序级功能和服务，旨在简化复杂工程应用程序的开发过程。

8.2 AppStdL 的功能与职责

- 标准文档管理

  提供标准的文档管理功能，包括文档的创建、打开、保存、关闭等操作。这些操作通常是大多数工程应用程序的基础。

- 事务管理

  支持事务处理机制，允许对文档进行修改操作，并支持 Undo/Redo 功能。这对于保证数据的一致性和完整性非常重要。

- 用户界面交互
  提供与用户界面交互相关的标准功能，如命令处理、事件响应等。
- 数据存储和加载
  支持标准化的数据存储和加载机制，包括对不同格式的文件的处理，如 XML、二进制等。
- 应用程序配置
  提供应用程序配置的管理，允许存储和检索应用程序设置和参数。
- 资源管理
  管理应用程序所需的资源，如图像、图标、样式表等。

8.3 AppStdL 主要接口与功能

- `AppStd_Application` class
  功能: 作为应用程序的核心类，负责管理文档和用户界面的交互。
  主要接口: `NewDocument`, `OpenDocument`, `SaveDocument`, `CloseDocument`
- `AppStd_Document` class
  功能: 代表应用程序中的单个文档，负责管理文档中的数据和事务。
  主要接口:
  - `InitNew` 初始化一个新的文档
  - `Open`, `Save` 打开/保存文档
  - `Undo`, `Redo` 撤销/重做操作
- `AppStd_Command` class
  功能: 封装用户界面中的命令操作，用于处理事件和执行特定的动作。
  主要接口
  - `Execute` 执行命令
  - `Undo`, `Redo` 命令的撤销和重做
- `AppStd_Preferences` class
  功能: 管理应用程序的配置和偏好设置。
  主要接口:
  - `ReadPreferences` 读取配置设置
  - `WritePreferences` 写入配置设置

9 BinTObj package

10 XmlTObj packages