ZWSOFT

ZW3D永久命名（上）

主讲人：翁亦旸

自我介绍

姓名： 翁亦旸

教育背景： 电子科技大学 信息与软件工程

谢菲尔德大学 软件系统

工作经历

2021年4月 加入中望，一直在永久命名相关模块工作：

C O N T E N T S

什么是永久命名

- What?
- Why?
- ZW3D现有永久命名

永久命名结构

- Face Label
- Shell Label
- Edge Label

C O N T E N T S

永久命名匹配

- Vdata解析
- 永久命名查找
- 永久命名匹配

永久命名优化

- 缓存
- 并行
- 局部匹配

什么是永久命名

什么是永久命名

What?

能体现参数化设计的、对拓扑元素的一种永久性的语义描述

Why?

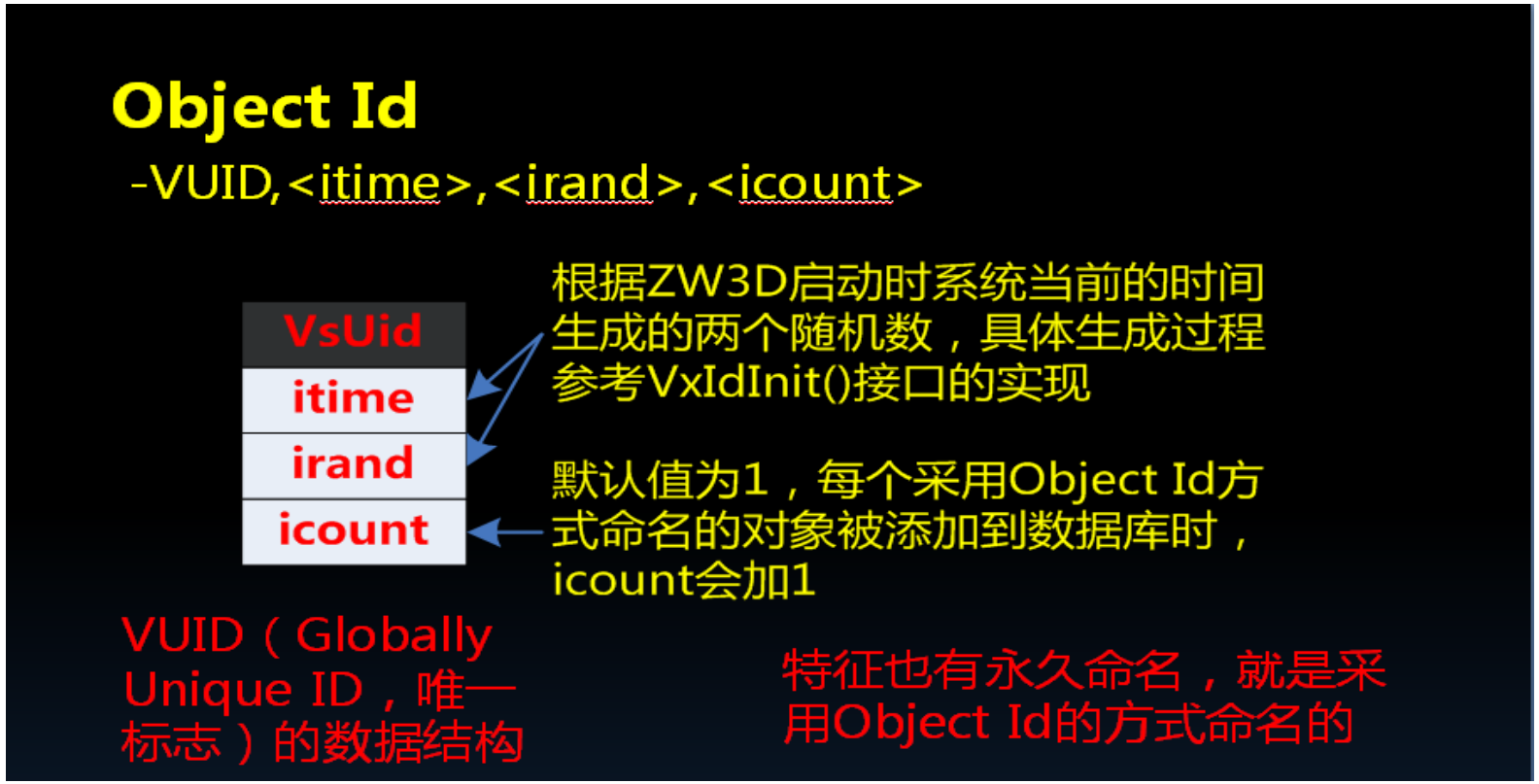
识别与追踪拓扑元素、维护设计意图

ZW3D现有永久命名基本规则

- 不同类型实体有各自的格式
- 以一串数字的形式进行存储在实体数据的末尾，每段的第一个数字表示类型
- 特征object ID （UID）是基础
- 有的label结尾会附上一个点坐标
- 有的label开头会记录文件名、part名（-2，-32开头）
- 属于递归型结构

永久命名结构

永久命名结构---UID

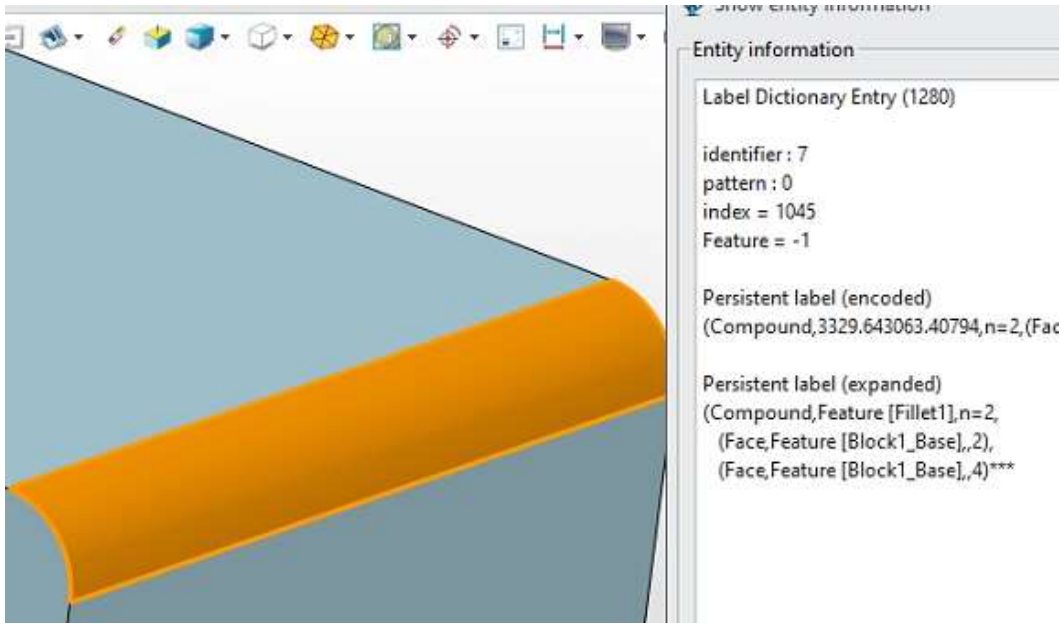


- 这是一张有些年头的图了
- 直接使用UID作为标识的对象：特征、草图、基准、curvelist、组件、草图中的几何对象和参考引用，即不会重新生成的对象

永久命名结构---VCOMPOUND

基本结构

```
-VCOMPOUND, <UID>, <label cnt>,
            <label 1>,
            <label 2>,
            ….
            <label N>
```



- Label中常见的一种组合方式（CdLblCompound）
- 可理解为position code的扩展，一般用于记录拓扑相关性
- Label复杂的根源之一
- 其语义不明确是label匹配不准确的原因之一
- 常见于合并面，圆角等地方。

永久命名--face

- Face label是永久命名体系的基础，shell和edge的label一般由face组合而成
- 一般格式：

  -VFACE, <UID>, <parent label>, <position code>

- 外部导入生成的explicit face:

  ~~-VFACE2, < shape ftr UID>,~~ <original face label>, ~~<position code>~~

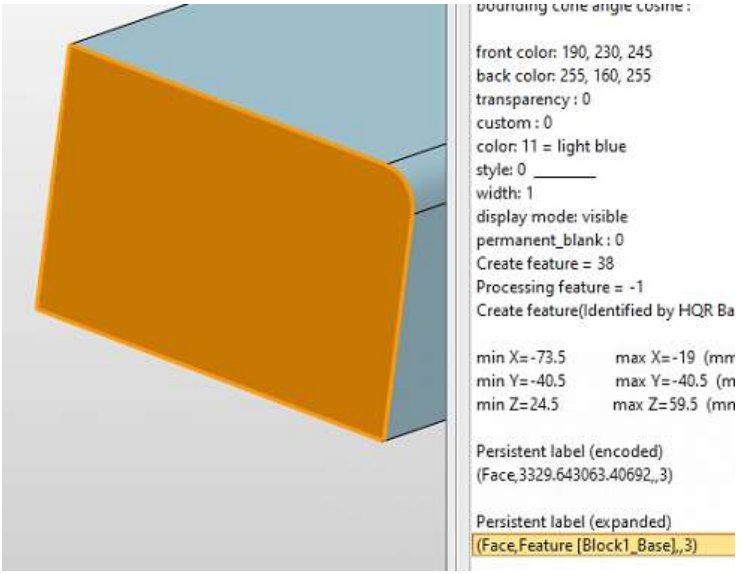
  -VFACE, <import ftr UID>, <parent label>, <position code>

- 封装、去参生成的explicit face:

  ~~-VFACE2, <shape ftr UID>,~~ <original face label>, ~~<position code>~~

- 阵列面的label

  -VFACE, <pattern ftr UID>, <original face label>, <pattern position code>



永久命名结构---分裂面

基本结构

首先得到Simple split face label: <simple split label>

-VFACE, <UID>, < original face label>, <position code>

进一步组合:

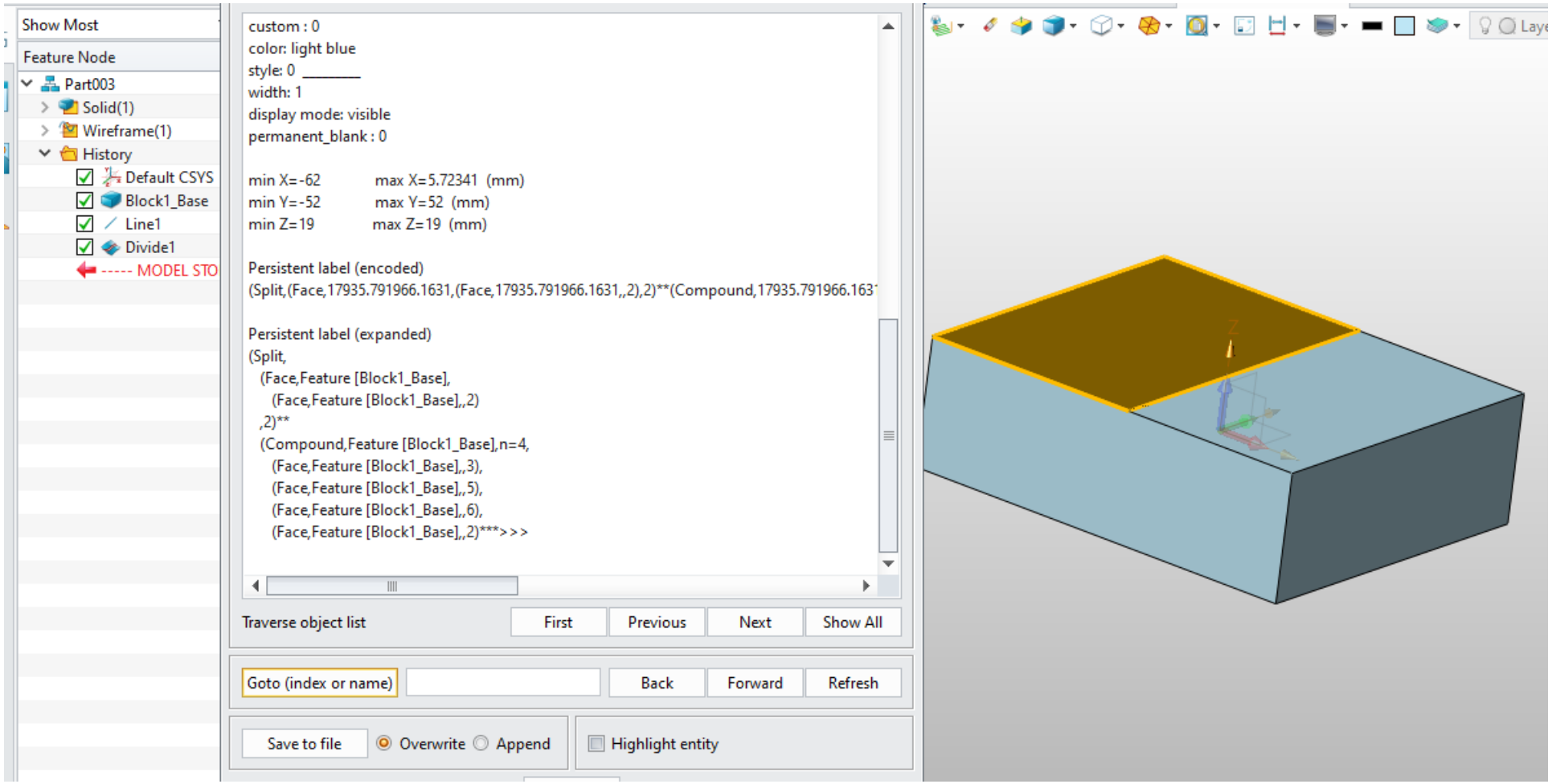
-VSPLIT,
<simple split label>
-VCOMPOUND, <feature UID>, <label cnt>,
<adjacent face1 label>,
<adjacent face2 label>,
…
original face label]

这里的VCOMPOUND表示拓扑邻接信息

储存位置

- Label中常见的一种组合方式（CdLblCompound）
- 可理解为position code的扩展，一般用于记录拓扑相关性
- Label复杂的根源之一
- 其语义不明确是label匹配不准确的原因之一
- 常见于合并面，圆角等地方。

永久命名结构---分裂面

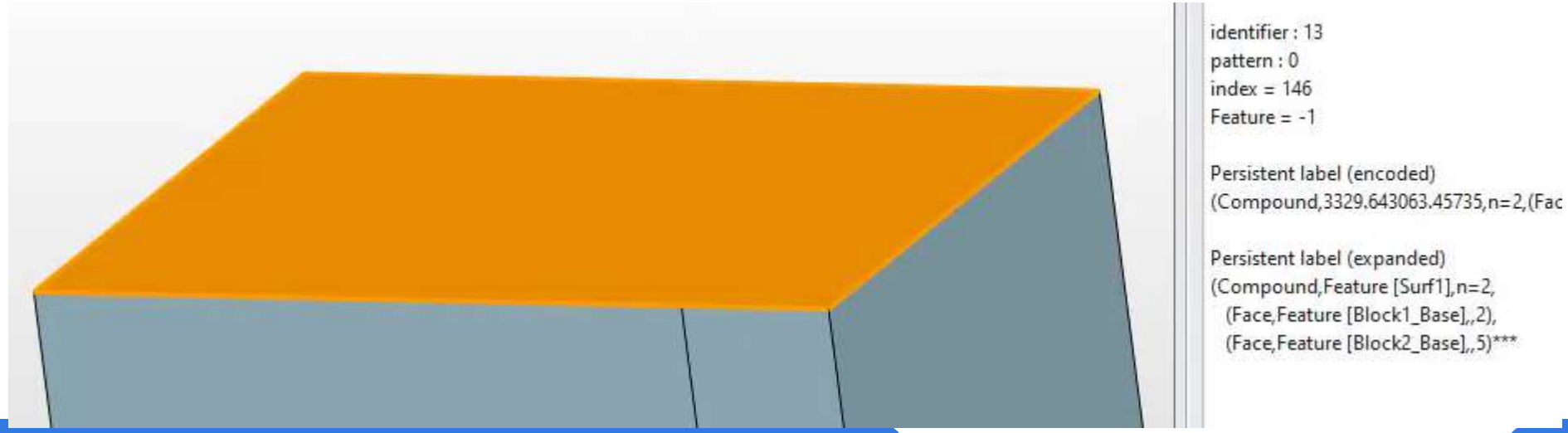

Show Most
Feature Node
Part003
Solid(1)
Wireframe(1)
History
Default CSYS
Block1_Base
Line1
Divide1
----- MODEL STO
custom : 0
color: light blue
style: 0 _____
width: 1
display mode: visible
permanent_blank : 0
min X=-62 max X=5.72341 (mm)
min Y=-52 max Y=52 (mm)
min Z=19 max Z=19 (mm)
Persistent label (encoded)
(Split,(Face,17935.791966.1631,(Face,17935.791966.1631,,2),2)**(Compound,17935.791966.163
Persistent label (expanded)
(Split,
(Face,Feature [Block1_Base],
(Face,Feature [Block1_Base],,2)
,2)**
(Compound,Feature [Block1_Base],n=4,
(Face,Feature [Block1_Base],,3),
(Face,Feature [Block1_Base],,5),
(Face,Feature [Block1_Base],,6),
(Face,Feature [Block1_Base],,2)***>>>
Traverse object list
First
Previous
Next
Show All
Goto (index or name)
Back
Forward
Refresh
Save to file
Overwrite
Append
Highlight entity

永久命名结构---合并面

基本结构

```
Face merge
-VCOMPOUND, <feature UID>, <label cnt N>,
              <child face1 label>,
              <child face2 label>,
              …
              <child faceN label>
```

这里的VCOMPOUND表示要素集合



永久命名结构---Edge

基本结构

- Edges: （CdEdgeGenLbl）

  -VEDGE, <face label 1>, <face label 2>, <loop code>, <edge code>,<-PNT3,X,Y,Z>

- Edge label一般不在edge创建的时候创建，而是在第一次用到该edge的label时创建，可能导致edge label不能及时更新（基于字典项目，已修改）
- VDATA中记录的edge label有时会附加一个点，辅助匹配

永久命名结构---Edge

字段解释

- Edges: （CdEdgeGenLbl）

  -VEDGE, <face label 1>, <face label 2>, <loop code>, <edge code>,<-PNT3,X,Y,Z>

- 关于每个字段的表达意图是什么：
- 首先，在永久命名的区分中，边有四个类型。
- 首先分为两个大类：开放边和闭合边
- 开放边又分为创建型开放边和炸开型开放边
- 闭合边又分为唯一闭合边和非唯一闭合边。
- 是否拥有face label 2是区分是否是闭合边的关键。
- Edge code用于分辨兄弟开放边与非唯一闭合边，Loop code则用于区分一个面内的多个loop

永久命名结构---Shell

基本结构

- 只包含一个面的shell：

  -VSHELL, <face label>

- 一般shell label（CdShellLblMk）

  -VSHELL2, -VCOMPOUND, feature UID, num,

  <Face 1 Label>,

  <Face 2 Label>

  ...

  <Face num-1 Label>

<-VSHELL, feature UID, pos>

- 面的选取原则：优先created by base feature, short。（gLblUidCache）
- 面label的顺序：label 排序
- 主特征的选取：当前特征、面的base feature

永久命名结构---Shell

Shell 命名简化

- Shell label简化（CdShellSimplifyLabels, from 2600）

  -VSHELL, feature UID, pos1, pos2

- 条件：
  - 只创建一个shell
  - 阵列、镜像、拷贝
  - 创建多个shell时简化第一个
- 优点：一定程度上解决匹配效率问题、更符合主特征设计原则
- 隐患：排查不完全，可能出现相同的shell label；是否简化、简化哪个的规则不清晰，导致不稳定

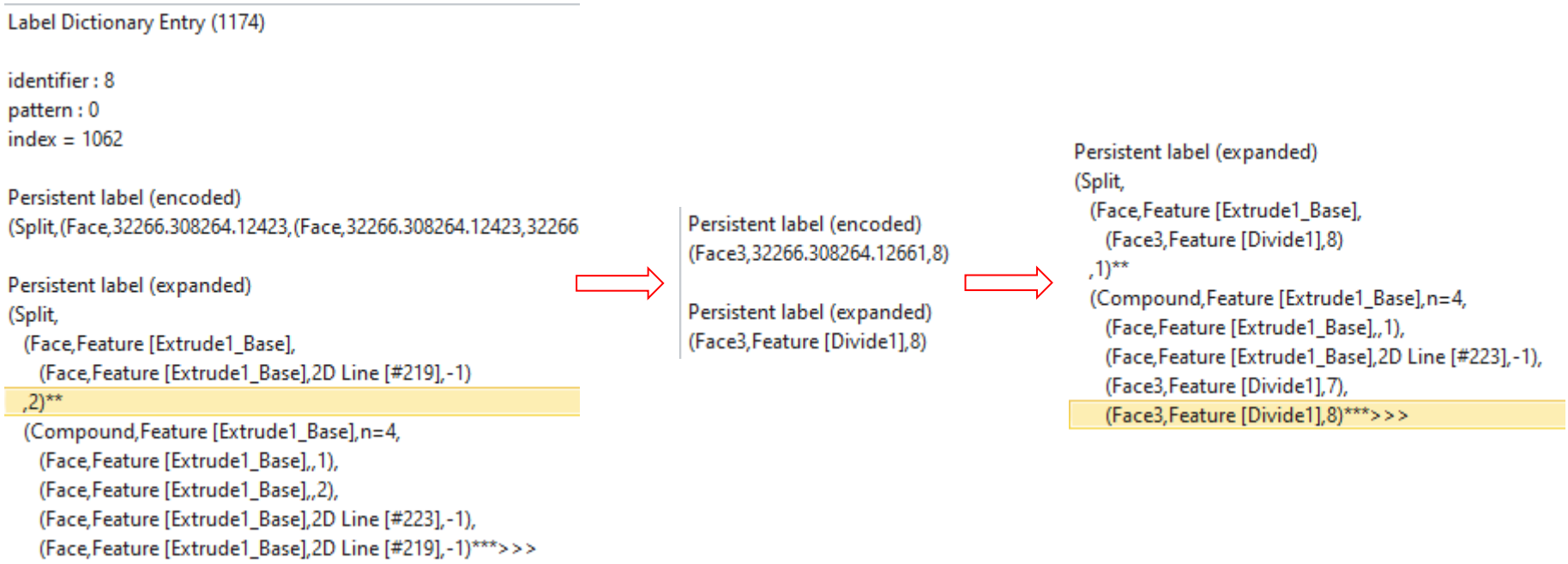
有什么问题？？

太冗余了！！！

永久命名结构---字典

永久命名字典

- 如果是simple label（IsLblSimple），则保留不变，如果不是，则创建label字典项，将原始的full label存储到字典，而将该字典项的“索引”作为一种新face label类型，VFACE3



永久命名结构---Parametric geometry

基本结构

- CdLblGeom、CdLblGeom3、CdLblGeomNew……
- -GEOM, <UID>, <parent label> <position code>
- -GEOM2, <UID>, <parent label> <position code>
- -GEOM3, <UID>, <parent label 1>, <parent label 2>, <position code1>, <position code2>
- -GEOM4,<UID>, <parent label1>, <parent label2>, <position code>

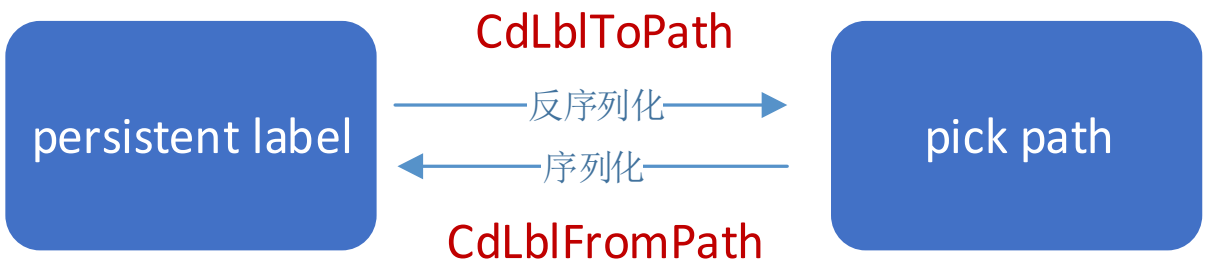
- 包括线框、annotation、拷贝的平面\草图\线框\annotation等

永久命名匹配

永久命名匹配

整体流程

VDATA的序列化与反序列化：V_CVT_LBL标准方法



label匹配标准流程：

1，VData解析流程：CdFtrRegen-CdDataRegen-CdPpathCvtLbl

2，Label查找流程：CdPpathCvtLbl-CdLblToPath-CdLblToPathImpl-CdLblFind-CdXXFindLbl

3，Label匹配流程：CdXXFindLbl-CdGenericFindLbl-CdLblCmp

匹配是分为精确匹配和非精确匹配的。

永久命名匹配---Vdata解析

1，VData解析流程：CdFtrRegen-CdDataRegen-CdPpathCvtLbl

- 这个阶段的目标就是解析vdata，将vdata中存储的实体（label）转换为pick path（db idx）
- 标准流程的走法可以查看接口CdDataRegenStd，这个是用在标准特征重生成过程中的。
- Dataregen是分层级的，所以我们存在模板套模板的情况时，会递归式解析vdata（比如圆角）
- Ppathcvtlbl是一个双向接口，换句话说，label持久化和label序列化都会从这里进行。
- 如果你希望脱离标准流程，单独去重生成某一个Vdata，你可以使用CdDataEval接口
- 简而言之，这个是一个n-n的接口

永久命名匹配---Label查找

2，Label查找流程：CdPpathCvtLbl-CdLblToPath-CdLblToPathImpl-CdLblFind-CdXXFindLbl

- Label 查找过程主要用途是确定我需要在哪个范围内对当前label进行检索。
- CdLblToPathImpl是功能的集合点，也是比较核心的代码。
- 这个解析过程也是一个循环过程，如果label路径中存在component，那么会在查找过程中不断调整target，直到最终的target
- 传入的参数不一定是label，也可以是uid，会通过uid cache检查对应的实体。
- CdXXFindLbl取决于targ的类型，常见流程为part-brep。
- 这里的目标，是将匹配转换为n-1。

永久命名匹配---Label匹配

3，Label匹配流程：CdXXFindLbl-CdGenericFindLbl-CdLblCmp

◆ 这个流程负责在label查找流程确定下来的匹配范围中，对每一个label进行比对，检查当前的待匹配label，和剩余所有的待比对label进行匹配，对每一对匹配结果进行评分。

◆ 那么在这个流程下，每一个label如何确定自己对应的实体呢？就是将所有的评分进行排序，然后选择最优结果。筛选结果在CdLblToPathImpl中的LblBestMatch中。

◆ 对于精确和非精确匹配，这里会有不同的匹配算法。

永久命名匹配---Further Info

如果你不希望使用常规流程进行匹配呢？

你可以使用CdLblEval进行查找，这个接口直接对接的是CdLblFind，所以你需要提供

1，你需要的精度（在标准流程中由CdLblToPath负责）

2，你需要的target（在标准流程中由CdDataRegenInit负责）

3，你需要的label（在标准流程中由CdDataRegen负责）

如果想知道的更多，请移步

《永久命名查找》 - 软件架构与生态部 - ZWiki (zwcax.com)

永久命名优化

永久命名优化---缓存

1. 一共三个缓存，Label Cache，FtrUidCache和LblUidCache
2. LabelCache用于缓存当前brep中的所有shell，face和geom（刚刚添加）。主要目的是**加速label匹配。**在缓存中会使用二分查找，而普通的label match是一个线性的字符串比对，更不要说模糊状态下的递归比对了。
3. FtrUidCache是用来映射uid到ftr idx的，可以理解为是一个简单的map，否则我们只能从历史链中进行完全遍历来查找内容。
4. LblUidCache是随开随用的，并不会长期保存在内存当中。这个东西是用于确定检查所有的uid到实体的映射的，包括基准面，线框等等。并且会在label排序中使用。

如果需要详情请移步

《永久命名缓存》 - 软件架构与生态部 - ZWiki (zwcax.com)

永久命名优化

- 现在在使用中的永久命名优化主要有两个，针对匹配效率的。
- 永久命名并行匹配
- 这个单纯的是一个技术性改进，将label比对过程进行并行比对。具体添加方式是在CdGenericFindLbl之前，同时将多个待比对label与待匹配label进行匹配。

- 永久命名局部匹配
- 这个改进旨在缩小匹配范围。从永久命名的角度很显而易见的可以看到，我们的label匹配是一个全局的匹配，而全局匹配会匹配大量的无关实体，这个改动就是提前缩小匹配范围，根据快速回滚备份中选定实体的创建特征，对label查找范围进行缩小。


感谢聆听
Q & A