**3D人体动作跟踪与对比详细设计**

目录

[1. 引言 2](#_Toc14266171)

[1.1 编写目的 2](#_Toc14266172)

[1.2 背景 3](#_Toc14266173)

[1.3 术语表 3](#_Toc14266174)

[1.4 参考资料 3](#_Toc14266175)

[2. 设计概述 3](#_Toc14266176)

[2.1 需求概述 3](#_Toc14266177)

[2.2 运行环境概述 3](#_Toc14266178)

[2.3 条件与限制 4](#_Toc14266179)

[2.4 详细设计方法和工具 4](#_Toc14266180)

[3. 程序描述 4](#_Toc14266181)

[3.1 动作文件导出模块 4](#_Toc14266182)

[3.1.1 功能 4](#_Toc14266183)

[3.1.2 性能 4](#_Toc14266184)

[3.1.3 输入项 4](#_Toc14266185)

[3.1.4 输出项 4](#_Toc14266186)

[3.1.5 算法 5](#_Toc14266187)

[3.1.6 程序逻辑 5](#_Toc14266188)

[3.2 姿态对比模块 5](#_Toc14266189)

[3.2.1 功能 5](#_Toc14266190)

[3.2.2 性能 5](#_Toc14266191)

[3.2.3 输入项 5](#_Toc14266192)

[3.2.4 输出项 5](#_Toc14266193)

[3.2.5 算法 5](#_Toc14266194)

[3.2.6 程序逻辑 5](#_Toc14266195)

[3.3 界面 6](#_Toc14266196)

[3.3.1 功能 6](#_Toc14266197)

[3.3.2 初始菜单界面 7](#_Toc14266198)

[3.3.3动作捕捉界面 7](#_Toc14266199)

[3.3.4姿态对比界面 9](#_Toc14266200)

[4. 系统详细设计 9](#_Toc14266201)

[4.1 系统功能模块详细设计 9](#_Toc14266202)

[4.1.1 功能名：Vnect视频输出和处理 9](#_Toc14266203)

[4.1.2 功能名：Vnect3D坐标估计和显示 10](#_Toc14266204)

[4.1.3功能名：OpenMMD视频输出和处理 11](#_Toc14266205)

[4.1.4 功能名：3D坐标文件格式 12](#_Toc14266206)

[4.1.4功能名：3D坐标文件转FBX文件 13](#_Toc14266207)

[4.1.5功能名：3D坐标文件转vmd文件 16](#_Toc14266208)

[4.1.6功能名：OpenGL动画 18](#_Toc14266209)

[4.1.7功能名：3D姿态评估 19](#_Toc14266210)

修订版本记录

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 修订内容 | 修订人 | 修订日期 | 版本 | 备注 |
| 1 | 初版 | 马学坤 | 2019.6.14 | 1.0 |  |
| 2 |  | 王恺，何建航 | 2019.7.3 | 1.1 | 添加vnect，fbx模块详细设计 |
| 3 |  | 王恺，张熠鹏 | 2019.7.8 | 1.2 | 添加opengl动画，openmmd模块详细设计 |
| 4 |  | 马学坤，吴伟豪，黄锟城 | 2019.7.11 | 1.3 | 添加界面，vmd，姿态对比模块详细设计 |
| 5 |  | 王恺 | 2019.7.15 | 1.4 | 添加具体实现细节 |
| 6 | 添加具体实现 | 吴伟豪、黄锟城 | 2019.7.17 | 1.5 | 无 |

# 引言

## 1.1 编写目的

本详细设计文档为软件系统各个模块以及功能进行了详细的划分，并对各功能模块执行流程进行了设计，为程序员编码提供依据。

## 1.2 背景

目前，无论在游戏行业还是影视行业中，人物模型的动作制作仍是一项繁琐、昂贵的工作，或需要雇佣专业的人员的进行制作，或需要购买昂贵的专业设备进行人物的动作捕捉，这大大提高了制作人物模型动作的门槛，目前也还没有支持动作捕捉功能的软件。为此，我们决定制作一个基于windows’平台低门槛的动作捕捉与动作文件输出软件。在我们的应用中，用户可用一般的RGB摄像头对动作进行拍摄，如手机等，拍摄的视频经过系统处理将变成对应的动作文件。

## 1.3 术语表

1) 动作文件：我们提供两种输出文件格式：.fbx和.vmd。如果用户选择了3ds Max或Maya的输出格式，我们输出的文件格式为.fbx，该文件为AutoDesk提供的模型文件格式，包含了模型的顶点、动画信息等。如果用户选择了MMD的输出格式，我们输出的文件格式为.vmd，该文件为MikuMikuDance使用的三维动画文件，包含人物骨骼的关键帧信息。

2) 中间文件：系统分析用户输入的视频后，用来保存视频中的人物动作的骨骼点的三维空间坐标与时间信息的文件，还不是最终的输出格式。

## 1.4 参考资料

# 2. 设计概述

## 2.1 需求概述

1. 读取视频文件，对视频文件进行预处理，输出视频文件中单个人体的姿态动作，以fbx或vmd动作文件格式输出。
2. 读取两个视频文件，对其中的动作姿态进行对比分析，输出分析结果。
3. 读取视频文件，将视频文件中的人进行服装纹理替换。

## 2.2 运行环境概述

Windows7及以上

## 2.3 条件与限制

1）目前对二维图像序列进行三维重建的算法有很多，但考虑到算法效率，对视频文件大小做出限制。

2）软件运行效率受到所选用的三维重建算法效率制约。

3）OpenMMD算法的使用需要用户自己去配置python环境，包括所使用的库的配置。

## 详细设计方法和工具

使用starUML绘制类图以及模块时序图

# 3. 程序描述

## 3.1 动作文件导出模块

### 3.1.1 功能

系统接收用输入的视频，经过用户选择的3d重构算法，得到视频中人体的三维坐标，之后再写入到fbx/vmd文件中，是模型与视频中人体做相同的动作。

### 3.1.2 性能

该模块的性能主要取决于所选用的算法，Vnect大概fps：7~8，OpenMMD中的OpenPose大概fps：5~6，（显卡GTX960M）但是OpenMMD在OpenPose之后得到的是二维坐标，需要使用3D Pose BaseLine方法将二维坐标转换为三维坐标，此方法是深度神经网络方法，带有已经训练好的模型，速度很快。得到三维坐标后需要将其写入fbx/vmd文件中，此方法速度较快，取决于CPU。

### 3.1.3 输入项

用户在界面的提示下输入人体动作视频。

### 3.1.4 输出项

程序根据用户所选择的方法和输出文件格式，导出相应的输出文件。

### 3.1.5 算法

该模块中主要采用四种算法，Vnect，OpenMMD，3D->fbx，3D->vmd，opengl动画。

### 3.1.6 程序逻辑

首先使用Vnect或OpenMMD算法得到人体的3D坐标动作序列，之后根据用户的选择导出成相应的FBX或vmd格式，若用户选择播放OpenGL动画，则在最后播放OpenGL动画

## 3.2 姿态对比模块

### 3.2.1 功能

该模块接收两段用户输入的人体姿态视频，对比其在3D坐标中的差异，最后评价两段视频的相似度。

### 3.2.2 性能

该模块的性能主要取决于所选用的3D姿态算法，算法的准确度直接决定了后面真正对比时的准确度，时间也大多是消耗在得到3D坐标。

### 3.2.3 输入项

用户根据界面提示输入两段人体动作视频

### 3.2.4 输出项

输出两个人体的骨架模型，最后输出相似度评分

### 3.2.5 算法

该模块主要使用三种算法：Vnect，OpenMMD，3D姿态评估。3D姿态评估主要包括：骨骼长度标准化、动作剪切、DTW算法获取对应帧、动作相似度判定

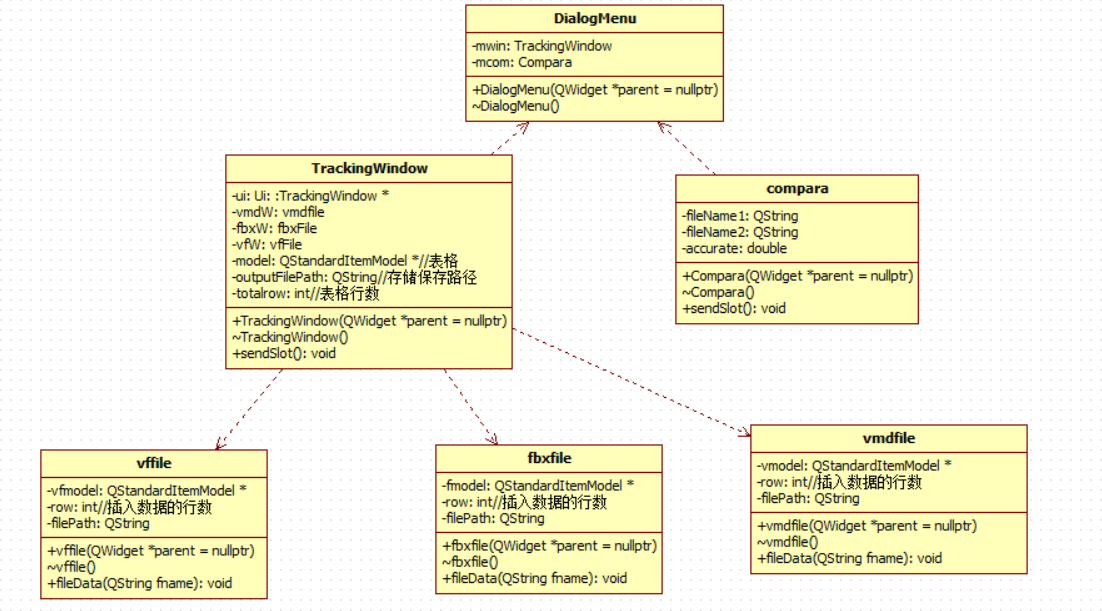
### 3.2.6 程序逻辑

首先使用Vnect或OpenMMD算法得到人体的3D坐标动作序列，匹配完对应帧后再进行动作相似度评估，之后播放OpenGL动画，用户可以较为直观的看出两个姿态的差别，最后得到整体的相似度。

## 3.3 界面

### 3.3.1 功能

该模块主要负责人机交互，是用户与系统交互的入口。界面大致框架如下类图：



### 3.3.2 初始菜单界面



用户可以选择动作捕捉以及姿态对比两种功能，并且可以返回菜单

### 3.3.3动作捕捉界面

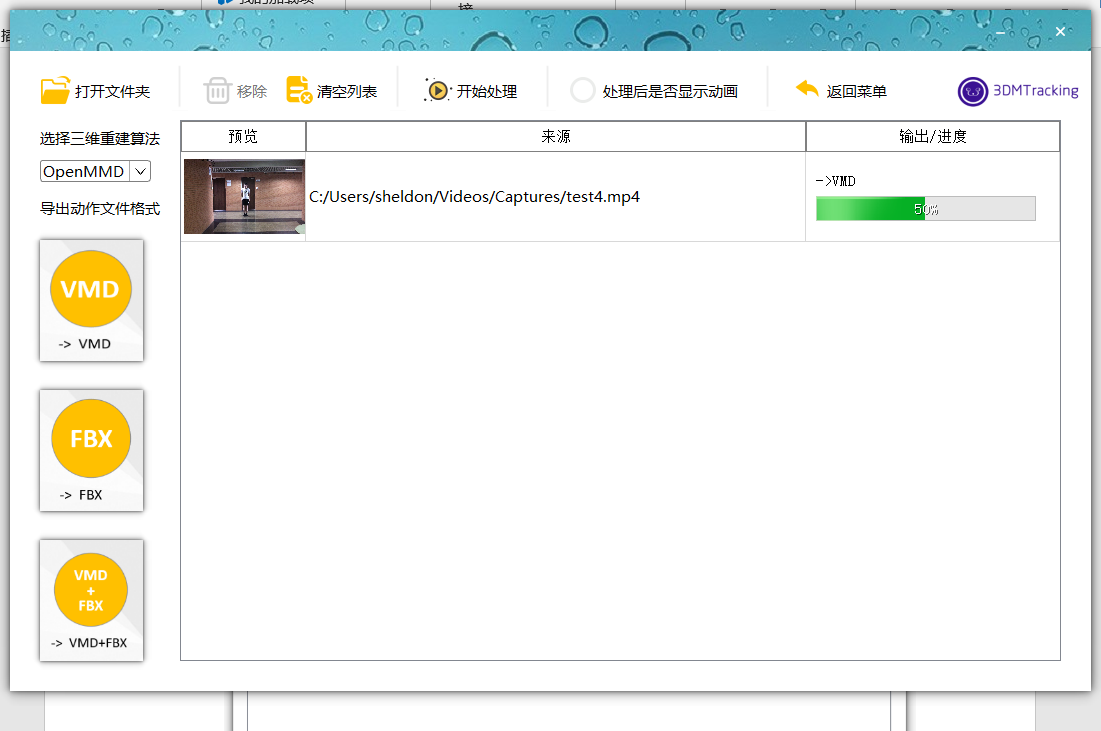


1. 左栏用于选择导出不同的文件格式，选择导出vmd+fbx可以导出两种格式的文件

点击后界面如图：

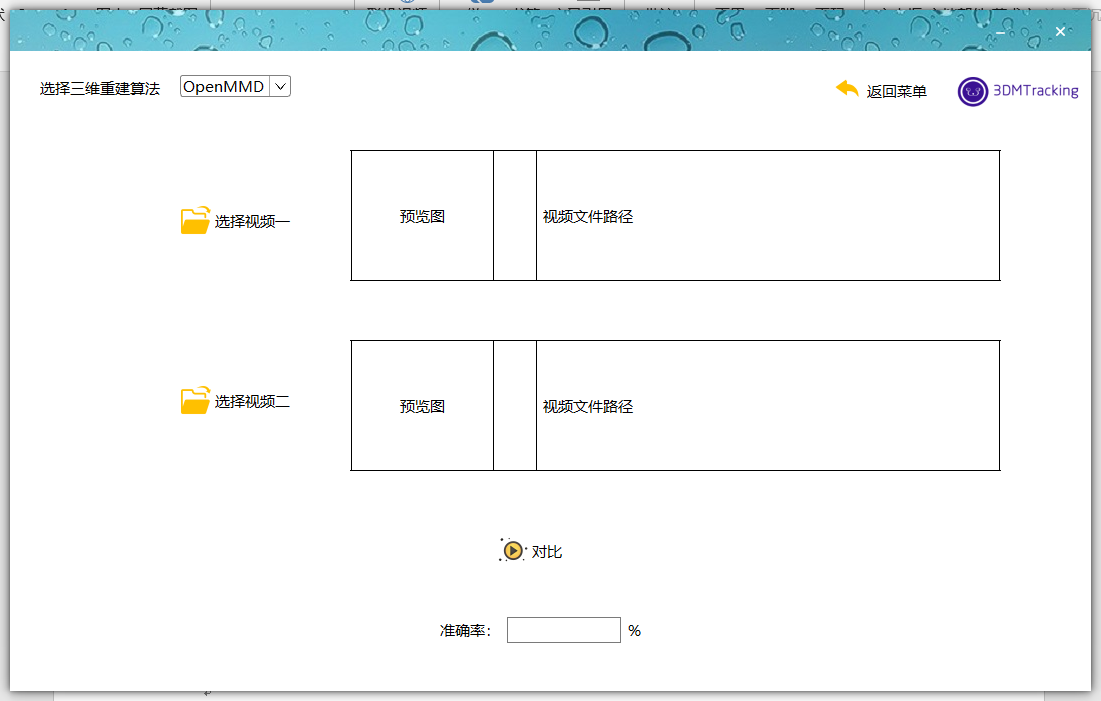


通过添加文件添加用户需要处理的视频文件支持MP4，AVI等格式。点击确定回到动作捕捉界面，点击开始处理，处理视频



1. 可以选择处理视频的算法，分为openmmd以及vnect
2. 打开文件夹用于选择输出路径
3. 勾选处理后显示动画后，用户可以在处理完视频后看到相应的骨骼动画。

### 3.3.4姿态对比界面



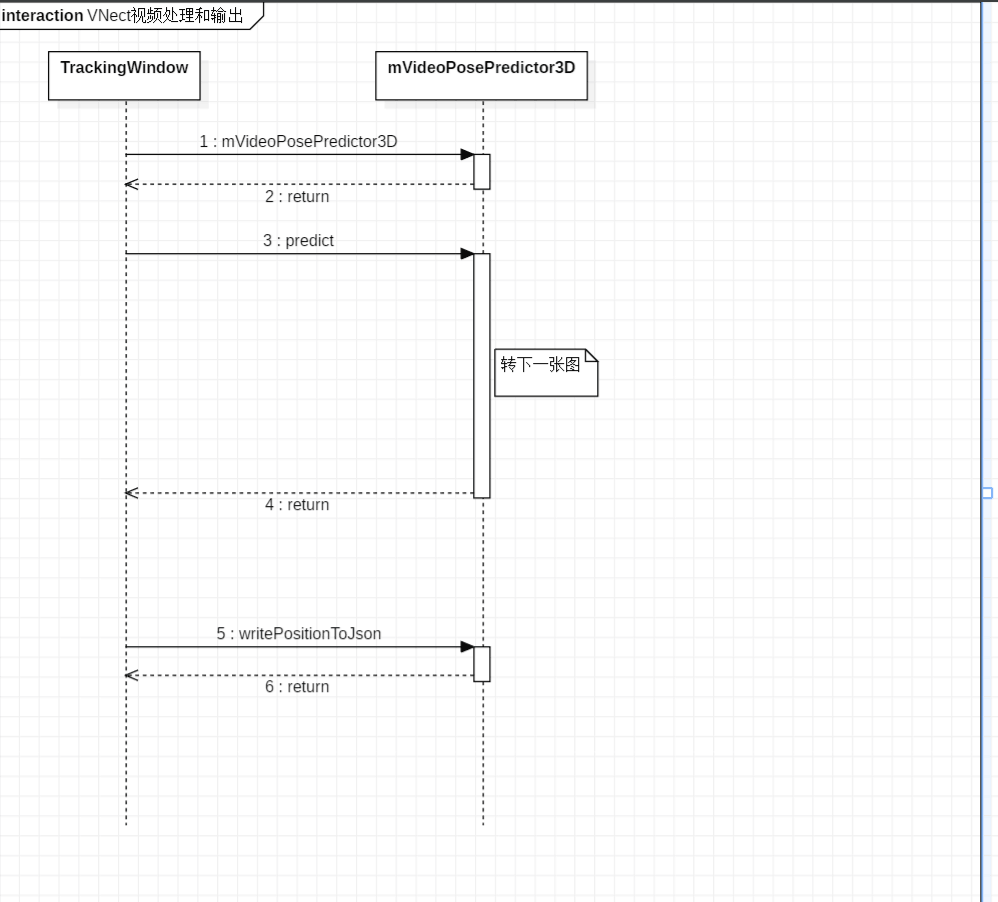
用户可选择对比算法，添加两个视频文件，点击对比，会进入对比画面，最后输出准确率。

# 4. 系统详细设计

## 4.1 系统功能模块详细设计

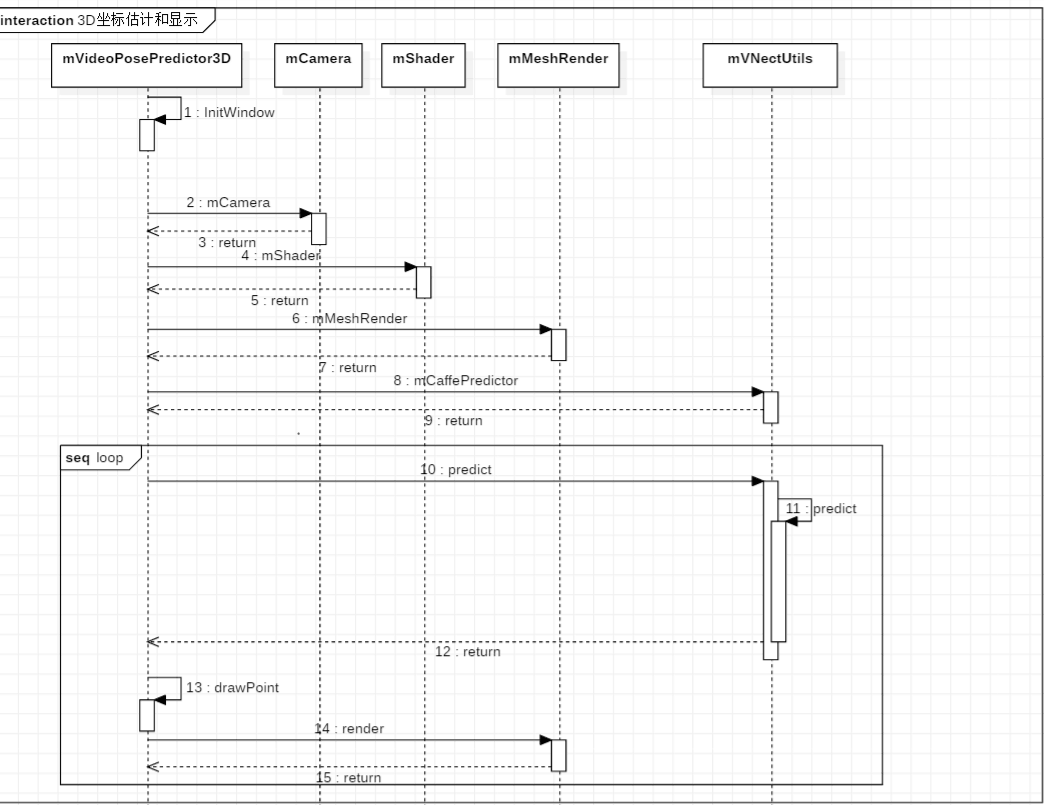
### 4.1.1 功能名：Vnect视频输出和处理

顺序图：



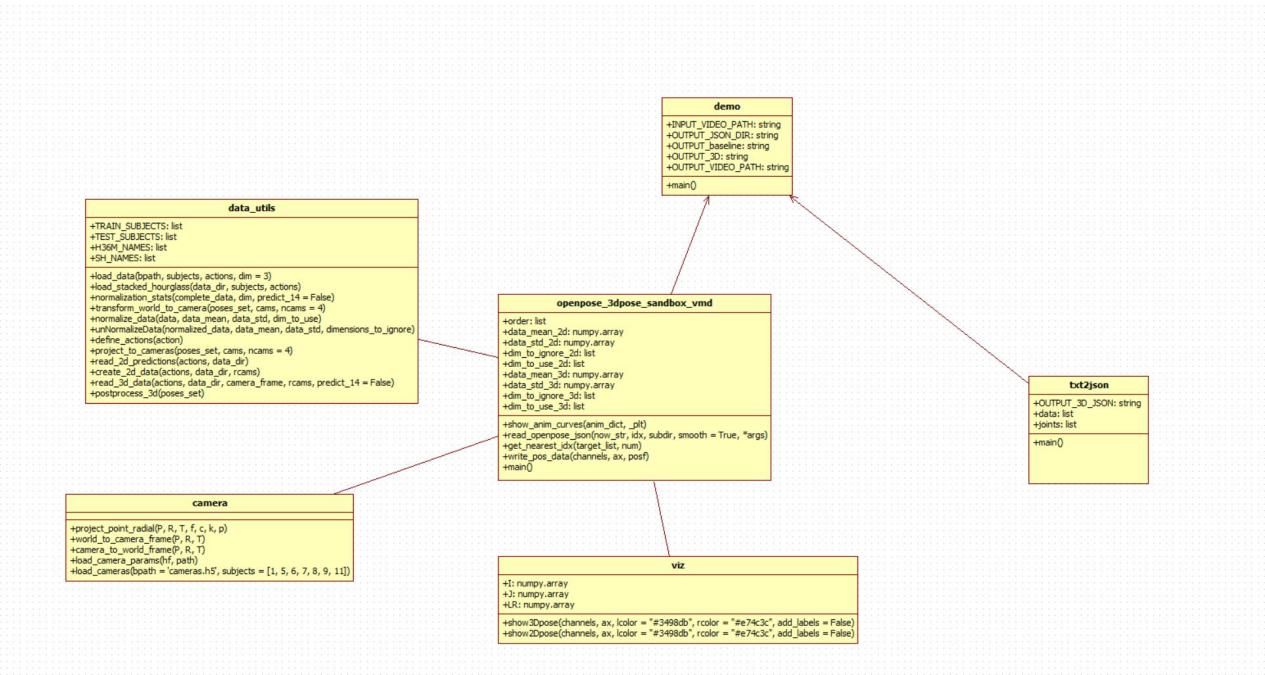
### 4.1.2 功能名：Vnect3D坐标估计和显示

顺序图：

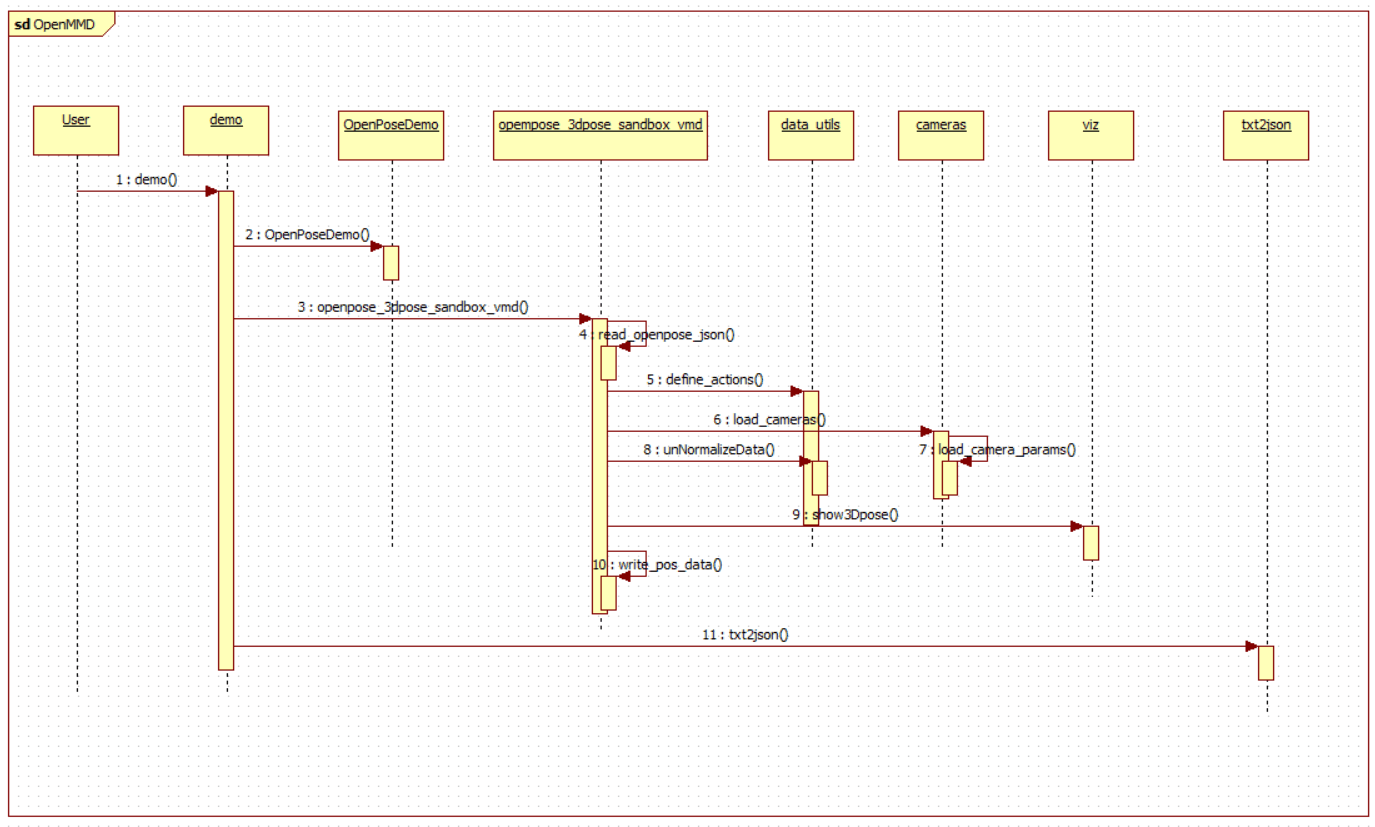


### 4.1.3功能名：OpenMMD视频输出和处理

类图：



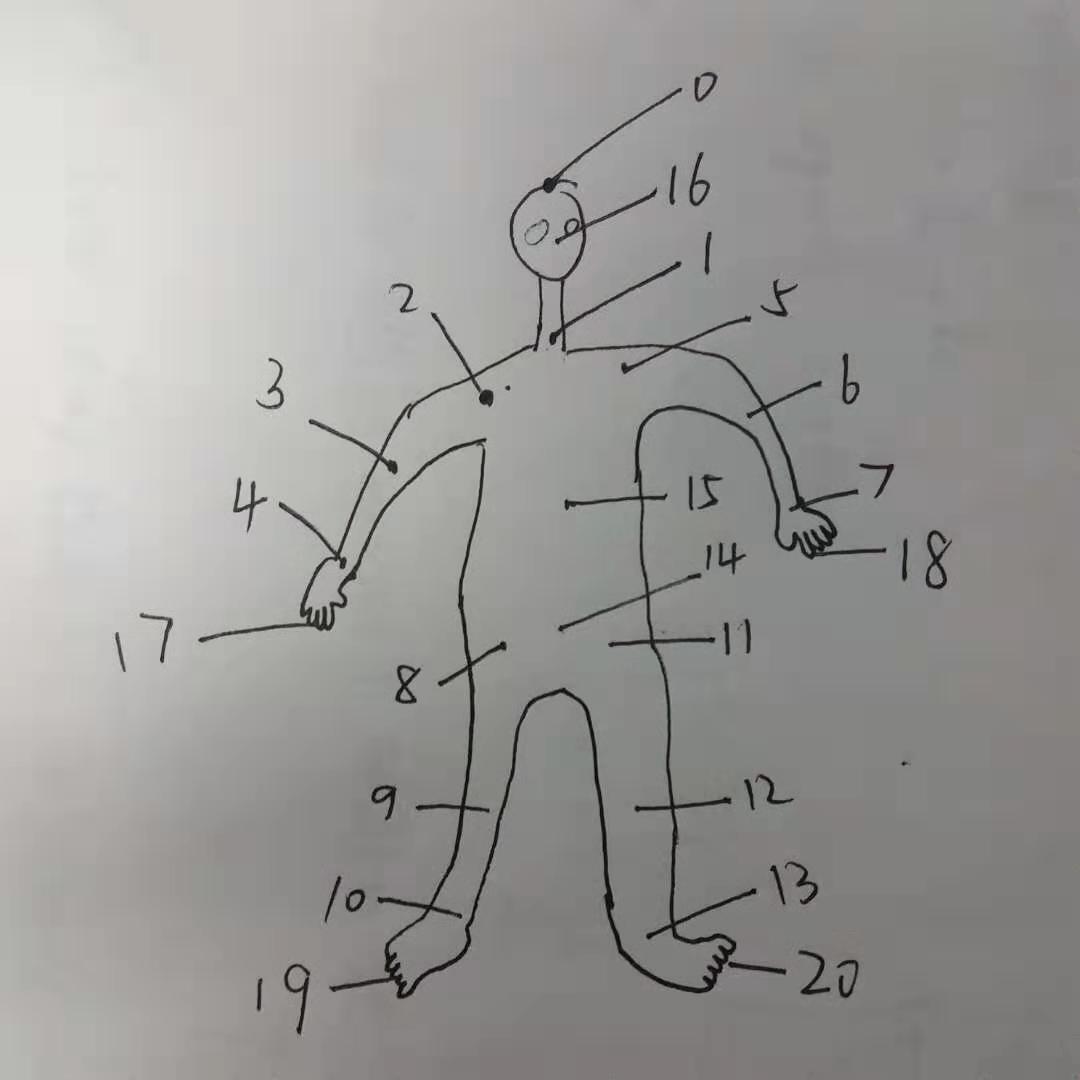
顺序图：



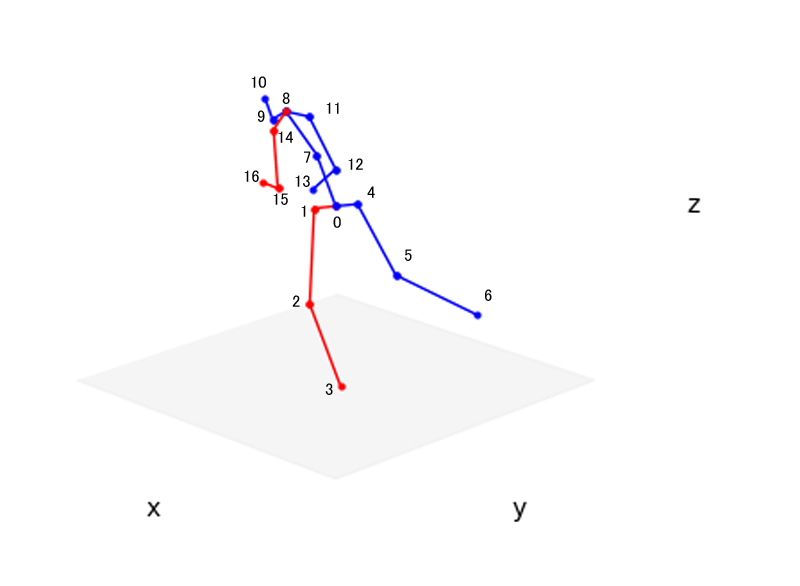
### 4.1.4 功能名：3D坐标文件格式

系统提供4.1.1和4.1.3两种3D重建算法来获取人体骨骼位置，

Vnect算法可以得到21个人体骨骼的位置信息，如下图



OpenMMD算法可以得到17个人体骨骼的位置信息，如下图



为了统一各个模块的接口我们制定了统一的3D坐标文件的json格式，如下图所示：



videopath为视频文件的绝对路径，

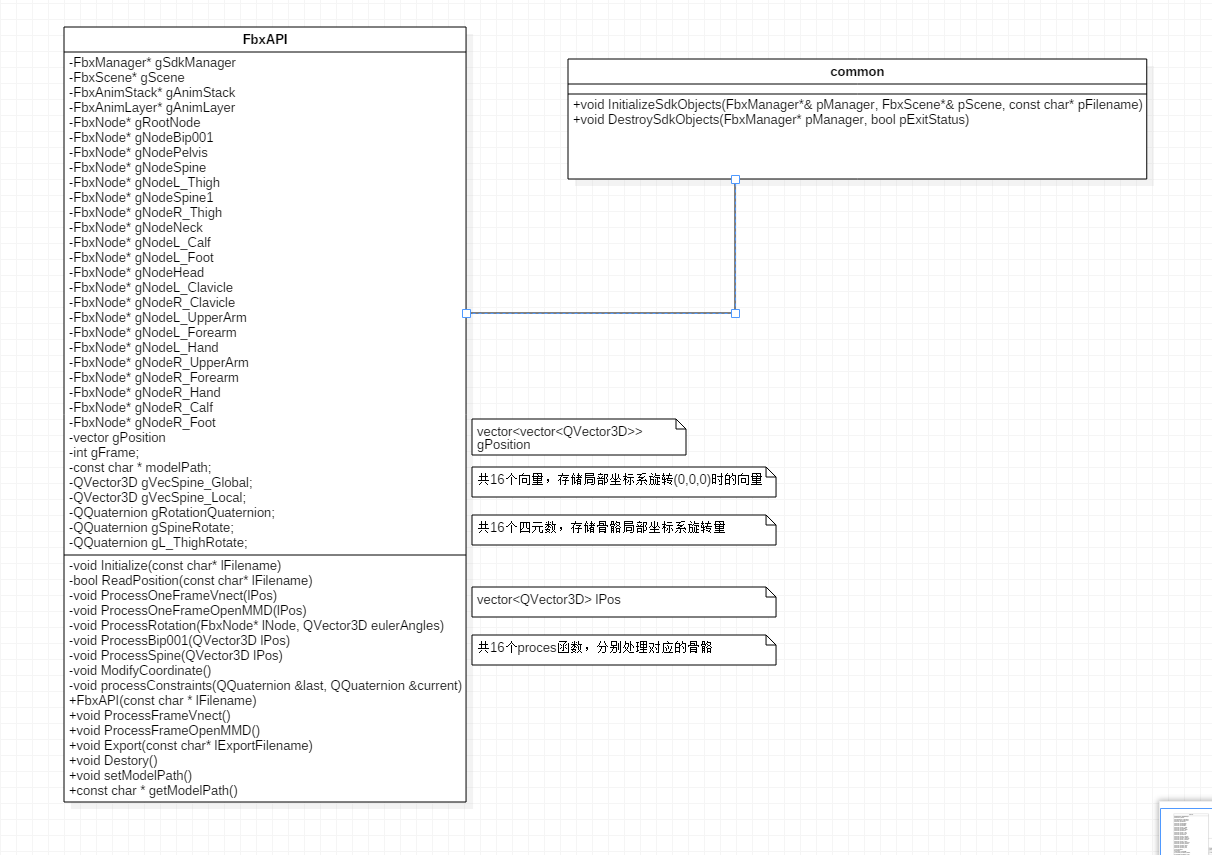
total\_frame为视频文件总帧数，

bodies内，frame为当前帧，position为骨骼位置信息，direction为骨骼法向量信息

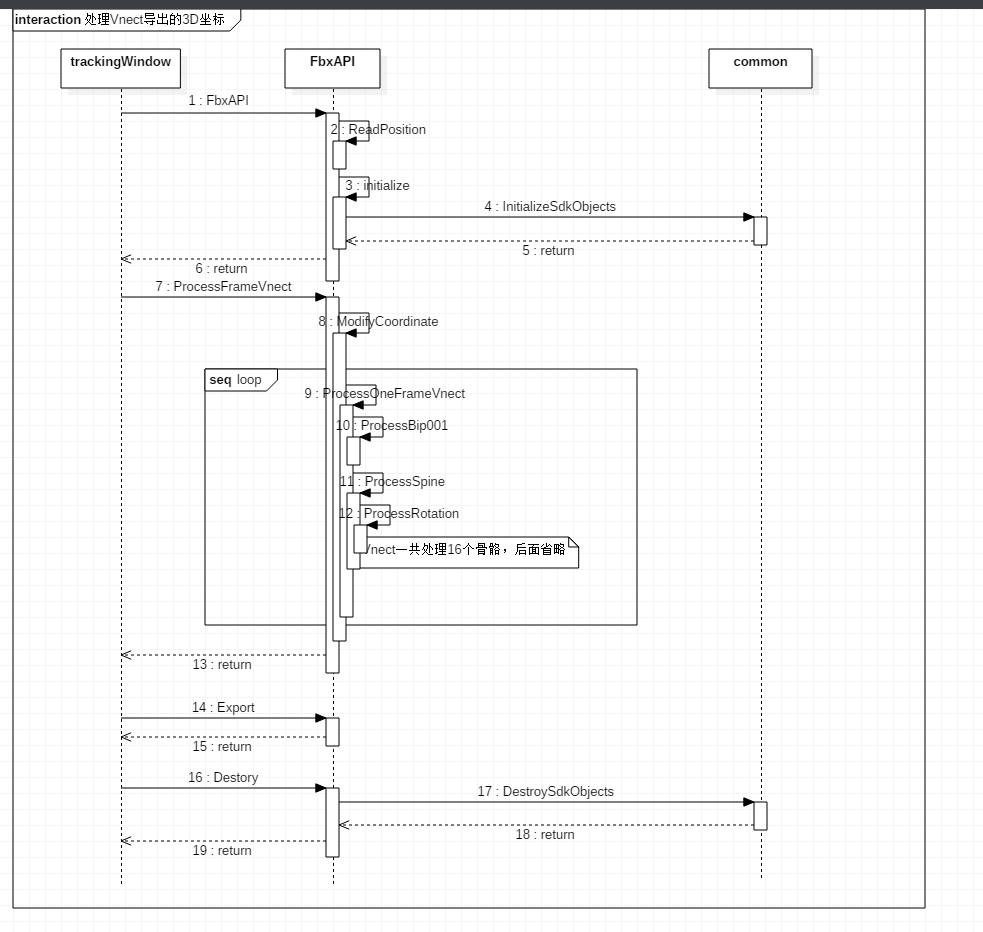
以Vnect为例，每一帧的position应该有21\*3行，第一行为0位置的x坐标，第二行为0位置的y坐标，第三行为0位置的z坐标，以此类推。

### 4.1.4功能名：3D坐标文件转FBX文件

类图：



顺序图

注：处理OpenMMD导出的3D坐标与上图大同小异，只是在7.调用ProcessOneFrameOpenMMD，中间循环的OpemMMD只需处理14个骨骼即可。

另外，类FbxAPI提供setModelPath函数，可以设置所处理的FBX原型，默认的Fbx原型只有骨骼没有模型。

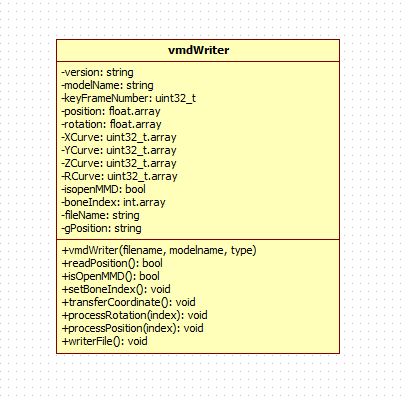
具体实现：首先需要的输入是我们规定好的3D坐标文件，然后读取每一帧的骨骼位置，之后要做的是坐标系的统一，我们假定（以vnect为例）14 –> 15这个骨骼位置相同，可以得到从跑出来的3D坐标到fbx文件默认骨骼坐标的四元数，将所有的坐标进行转换，得到fbx世界坐标系中的坐标，14号点（根节点）的位置决定整体位置，其他所有骨骼都是在根节点基础上进行骨骼旋转和缩放得到的，旋转过程中我们只使用了3个自由度，直接找到最小的旋转路径，并没有保证骨骼法向量位置正确，原因是无法只从空间位置信息得到法向量信息。解决方法目前我们想到的有两个：1. 某关节的法向量由特定两个关节的向量叉乘得到，此方法缺点是无法保证法向量的正确率，导致骨骼的错位，抖动十分严重，置信度不高。2. 在输入信息中增加法向量信息。 在得到每个骨骼的旋转信息后使用FBXSDK读入默认骨骼模型，写入，然后导出。

接口：FbxAPI类中一共留有7个public函数，FbxAPI(const char \* lFilename);，构造函数，包括3D坐标文件读取，FBXSKD的初始化. void ProcessFrameVnect();处理Vnect3D坐标入口，void ProcessFrameOpenMMD();处理OpenMMD3D坐标入口，void Export(const char\* lExportFilename); 导出FBX文件，void Destory();释放FBXSDK，void setModelPath(const char \* lModelPath); 设置模型路径，const char \* getModelPath();获取模型路径。

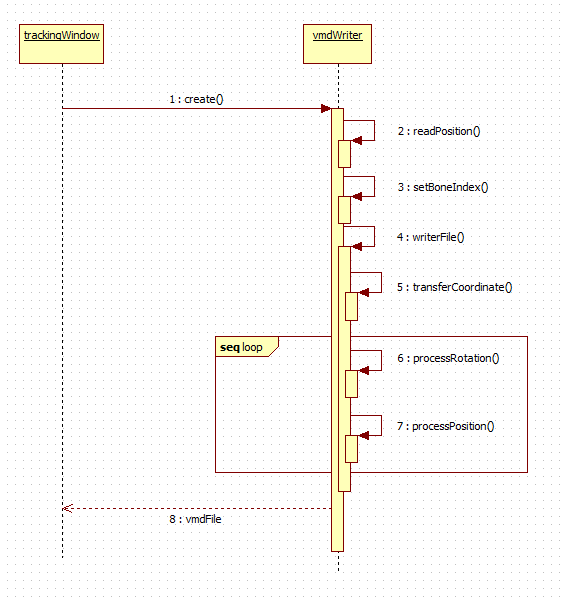
一般流程为：构造函数->选择处理方法入口->导出fbx文件->释放fbxSDK。

### 4.1.5功能名：3D坐标文件转vmd文件

类图：



顺序图：



函数介绍：

ReadPosition()：读取并储存json文件中的骨骼三维坐标

SetBoneIndex()：该函数的目的是设置一个骨骼序列以方便程序处理openMMD和VNect导出的不同人体骨骼序列

WriteFile()：将计算好的数据写入到二进制vmd文件中

TransferCoordinate()：将openMMD或VNect中的坐标系变换到MMD中的坐标系

ProcessRotation()：计算每个关节点的旋转变量

ProcessPosition()：计算IK骨的位置

需要注意的点：

1. MMD中使用的是左手坐标系，在计算旋转角度时候需要转换坐标系
2. Vmd文件中的骨骼名称需用shift-JIS编码写入，不能用unicode编码写入
3. MMD不止用关节点的旋转角度来控制人体的运动，还用了IK骨来控制，IK骨的计算比较复杂

IK骨的计算难点：

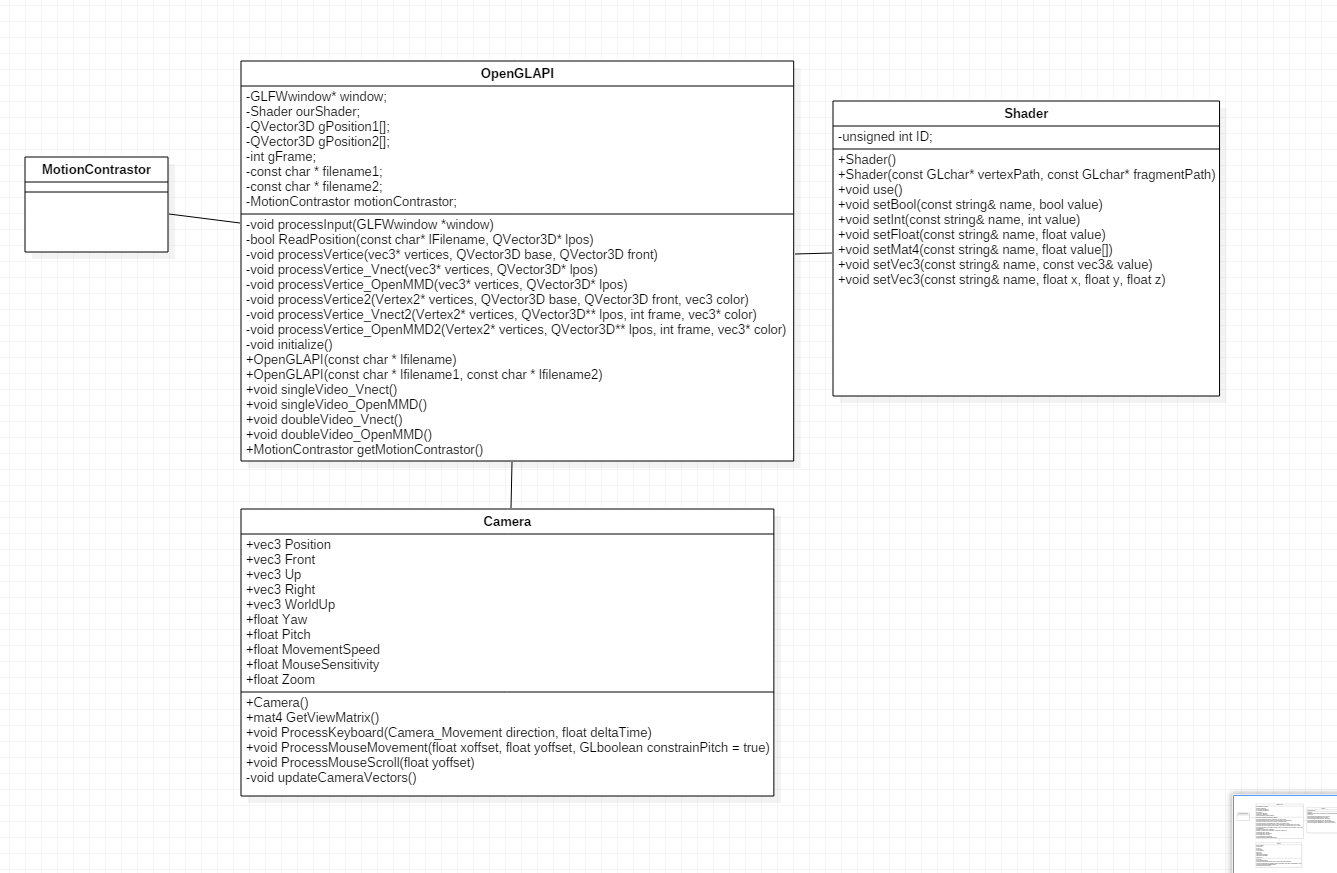
1. 计算视频中人物到MMD人物模型的缩放比例
2. 解决openMMD或VNect导出的“突兀数据”

对IK骨的计算设想：

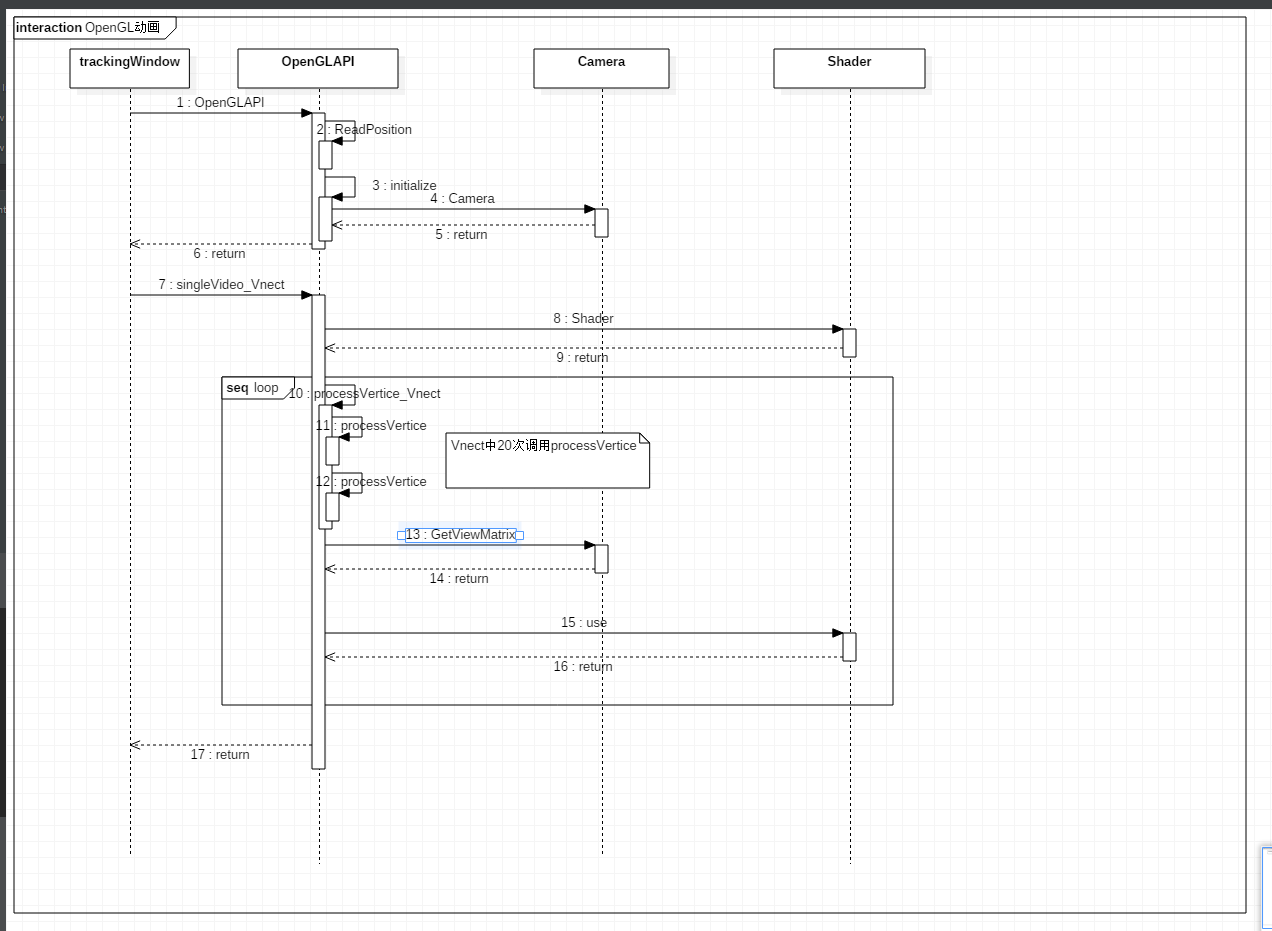
1. 通过计算膝盖的旋转角度来辅助计算
2. 通过人体的整体骨骼数据推断人体将要进行的运动，然后往vmd中写入相应的预值

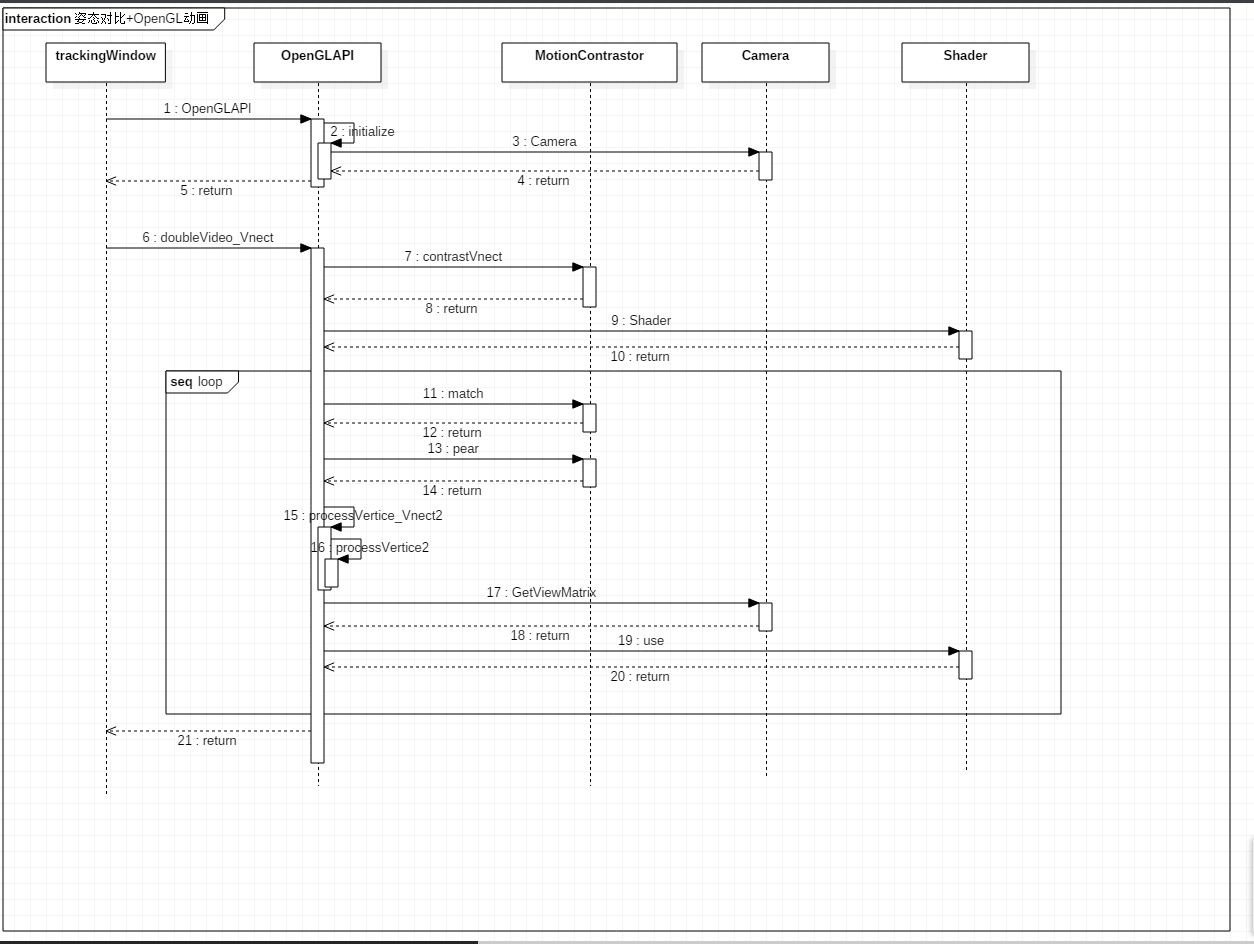
### 4.1.6功能名：OpenGL动画

类图：



顺序图：





具体实现：OpenGL动画模块预留了四个接口供选择，

singleVideo\_Vnect：处理单个Vnect跑出来的3D坐标

singleVideo\_OpenMMD：处理单个OpenMMD跑出来的3D坐标

doubleVideo\_Vnect：处理两个Vnect跑出来的3D坐标

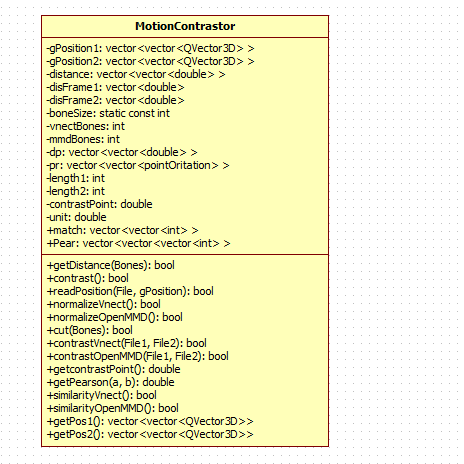
doubleVideo\_OpenMMD：处理两个OpenMMD跑出来的3D坐标

读取3D坐标->创建窗口->设置好shader->写入索引信息->循环内每一帧更新顶点信息（根据读取的顶点坐标，在相应的关节点之间画圆柱体）->循环过所有帧，播放结束。

大体上是上面的流程，在处理两个3D坐标序列时使用了4.1.7模块提供的序列配对，以及配对帧各个骨骼的相似度（比较了手部脚部的8个骨骼），播放时是播放配对帧，根据相似度，骨骼颜色会变化（只有红绿两种颜色，颜色渐变的话效果并不好，大部分都是黄绿色）。

### 4.1.7功能名：3D姿态评估

类图：



注：

length1、length2指两个视频的长度

gPosition1和gPosition2存储两个视频的关键点坐标

disFrame1、disFrame2指两个视频前后帧的关键点距离之和，用于剪切无用帧

vnectBones、mmdBones存储需要对比的关键点在vnect、mmd两种骨架模型中的下标，

distance、dp、pr是帧匹配算法DTW使用的中间变量，distance(i,j)指视频1第i帧和视频2第j帧的关键点距离之和，dp(i,j)指视频1前i帧与视频2前j帧最小的帧匹配距离之和，pr

存储的是匹配后的对应帧下标

unit指标准骨骼长度

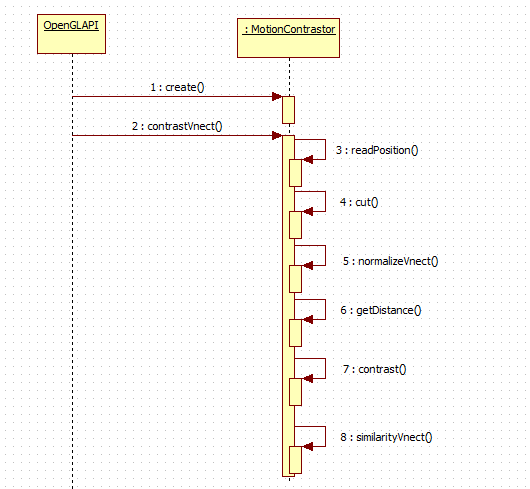
contrastPoint指两组动作的相似度，在0~100之间

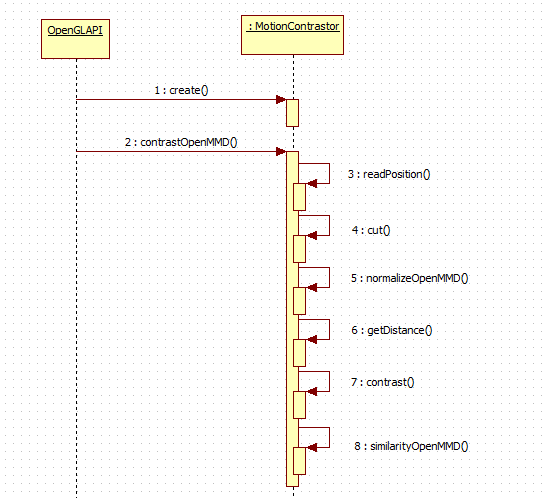
contrastPoint=100-5\*dp(length1,length2)/length1/unit

match用于存储对应帧的下标

Pear存的是匹配帧对应骨骼向量的相似度

顺序图：





具体实现：

readPostion用于获得关键点坐标，使用读取json的方式，

cut是剪切算法，这里将前后帧距离之差少于平均值一半的动作前部分去掉，这是为了剪掉视频开始时人物处于静止的状态，而Vnect算法得到的动作前几帧会出现动作混乱而不可使用的状况，所以将前后帧距离之差多于平均值1.5倍的动作前部分去掉，

normalizeVnect、normalizeOpenMMD是将人物骨骼长度统一，这里长度取为视频1骨骼长度的平均值，

getDistance是获取DTW算法需要的distance，

contrast是使用DTW算法获取匹配帧和相似度分数，

similarityVnect、similarityOpenMMD是用于获取对应帧的动作匹配度，这里使用pearson相关系数来比较，它的范围为-1~1，越接近1越相似

最后将得到的数据传回OpenGLAPI进行动作演示