详细设计

目录

[1. 引言 1](#_Toc13057527)

[1.1 编写目的 1](#_Toc13057528)

[1.2 背景 2](#_Toc13057529)

[1.3 术语表 2](#_Toc13057530)

[1.4 参考资料 2](#_Toc13057531)

[2. 设计概述 2](#_Toc13057532)

[2.1 需求概述 2](#_Toc13057533)

[2.2 运行环境概述 2](#_Toc13057534)

[2.3 条件与限制 3](#_Toc13057535)

[2.4 详细设计方法和工具 3](#_Toc13057536)

[3. 程序描述 3](#_Toc13057537)

[3.1 动作文件导出模块 3](#_Toc13057538)

[3.1.1 功能 3](#_Toc13057539)

[3.1.2 性能 3](#_Toc13057540)

[3.1.3 输入项 3](#_Toc13057541)

[3.1.4 输出项 3](#_Toc13057542)

[3.1.5 算法 4](#_Toc13057543)

[3.1.6 程序逻辑 4](#_Toc13057544)

[3.2 姿态对比模块 6](#_Toc13057545)

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 |  |
| 3 |  | 王恺，张熠鹏 | 2019.7.11 | 1.2 |  |
| 4 |  | 吴伟豪、黄锟城 | 2019.7.13 | 1.3 |  |

# 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

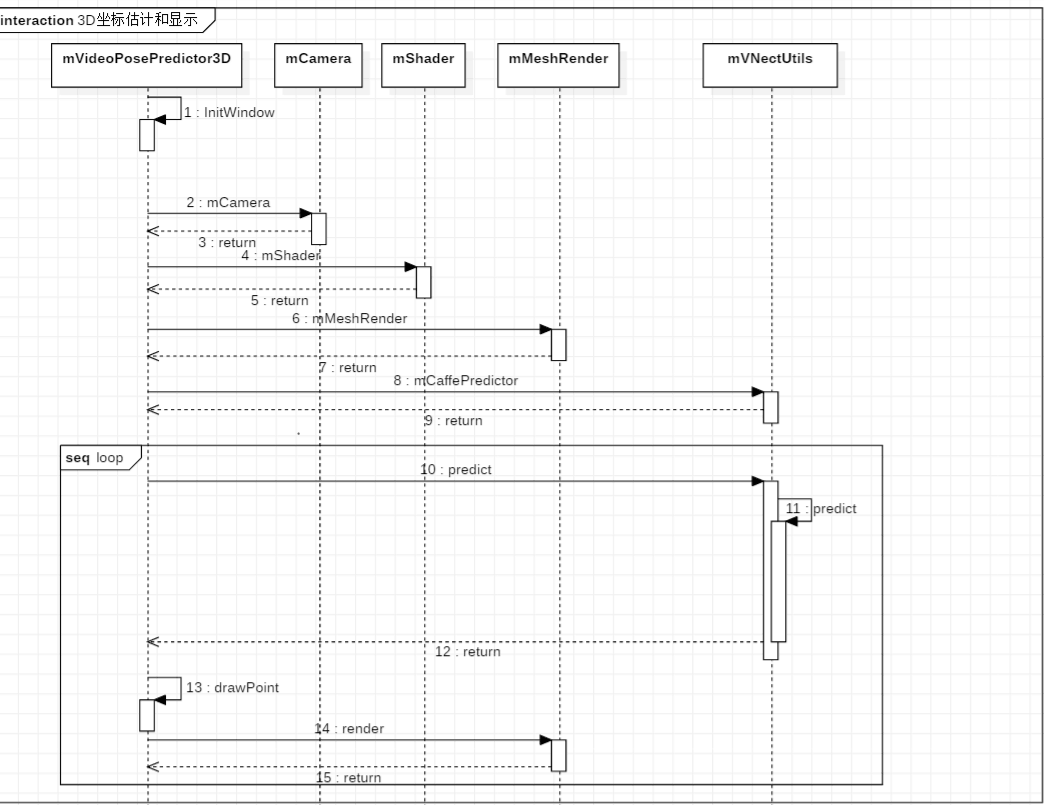
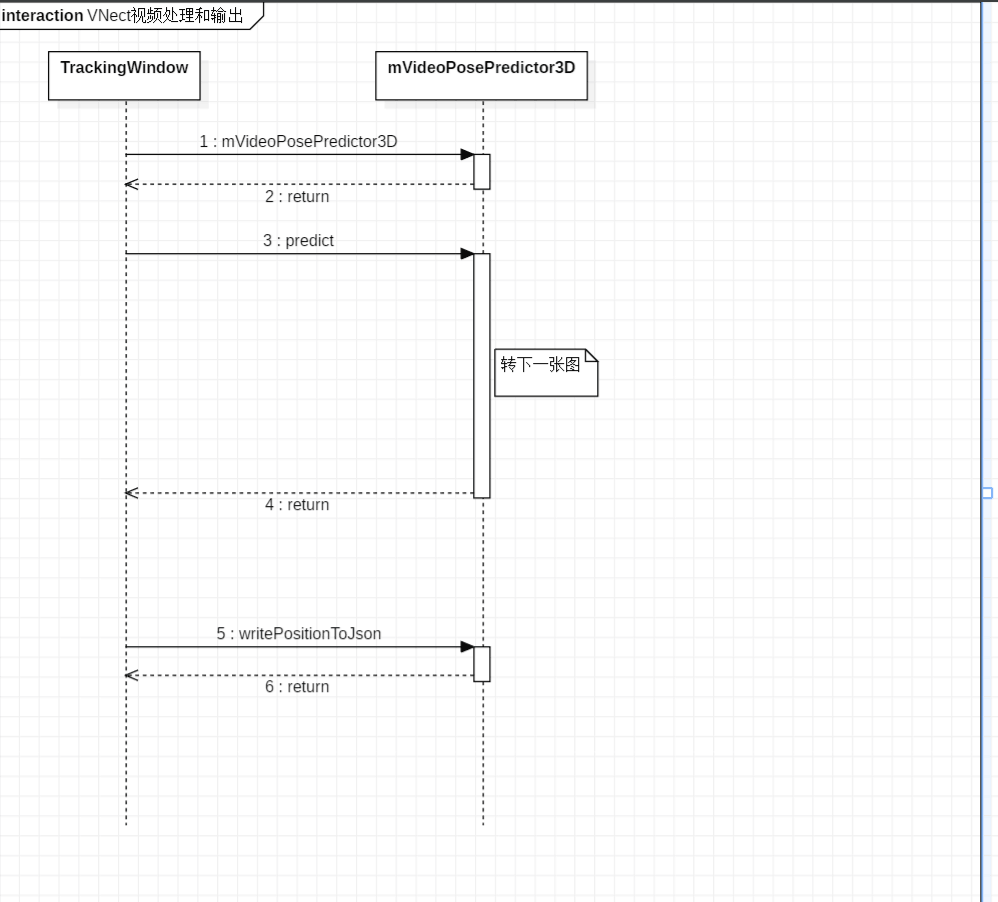
该模块中主要采用四种算法，Vnect，OpenMMD，3D->fbx，3D->vmd，opengl动画。

### 3.1.6 程序逻辑

#### 3.1.6.1 Vnect：

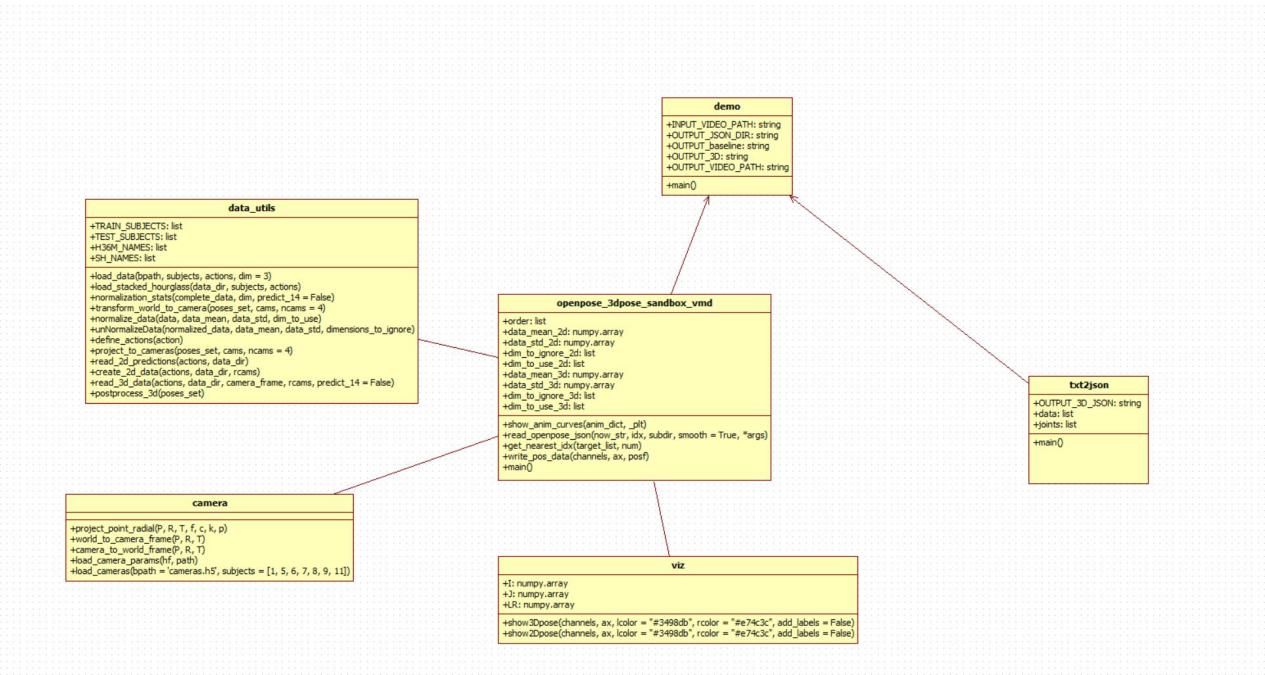
类图：

顺序图：

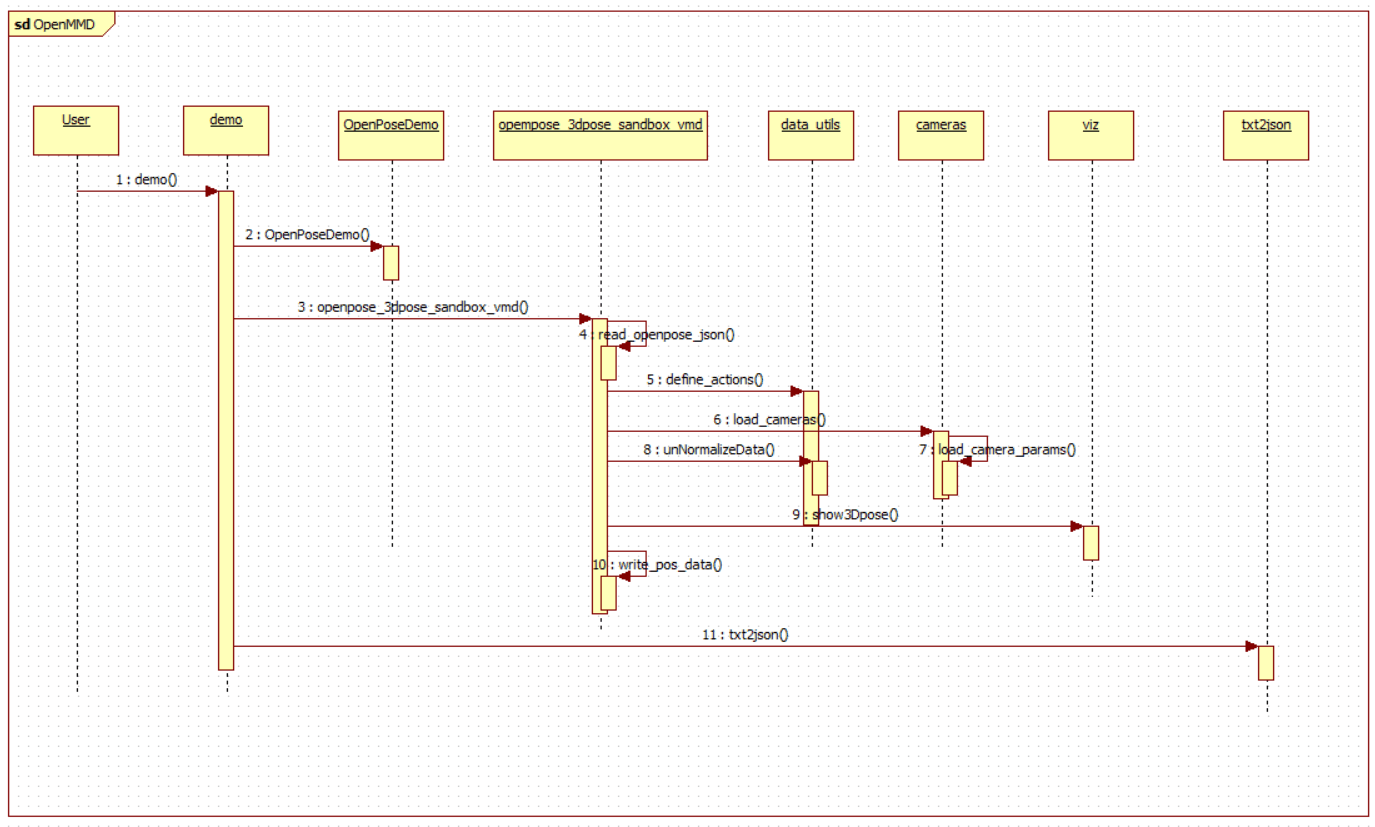


#### 3.1.6.2 OpenMMD：

类图：

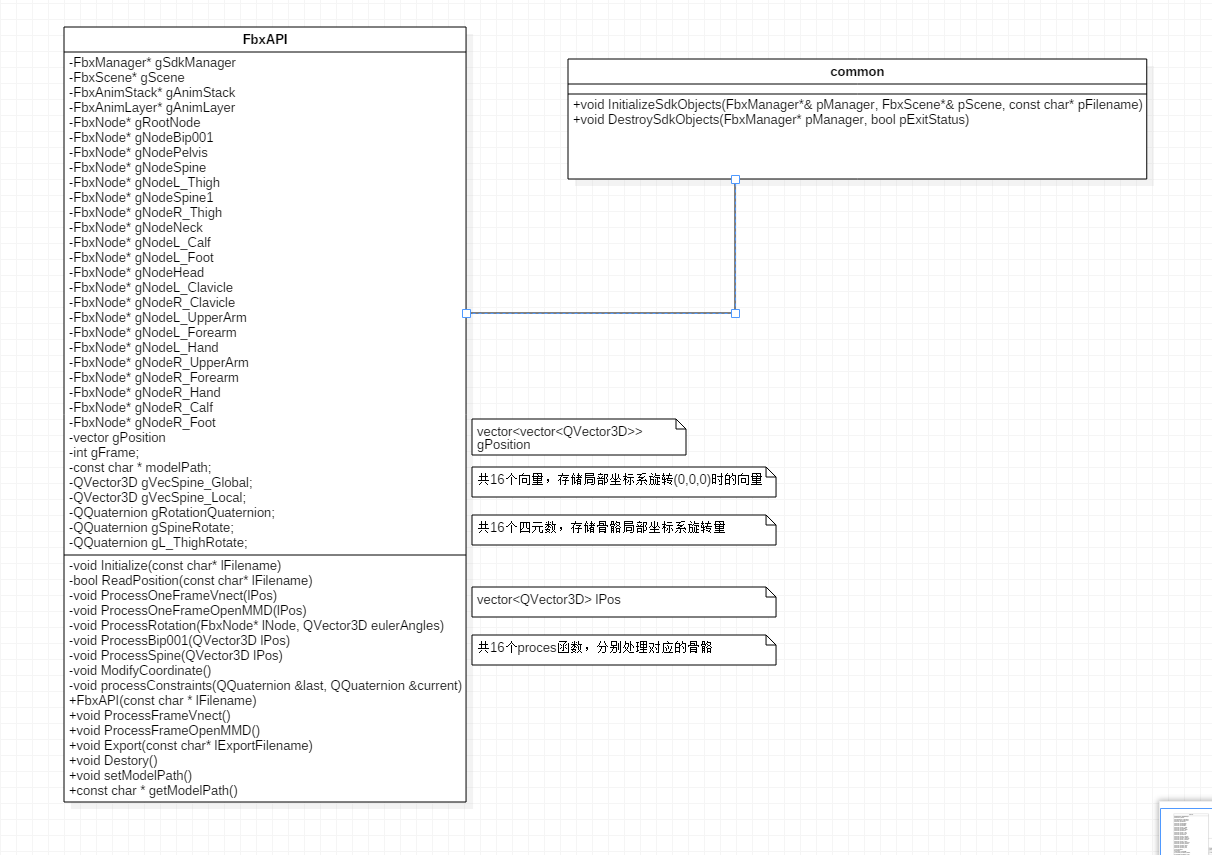


顺序图：

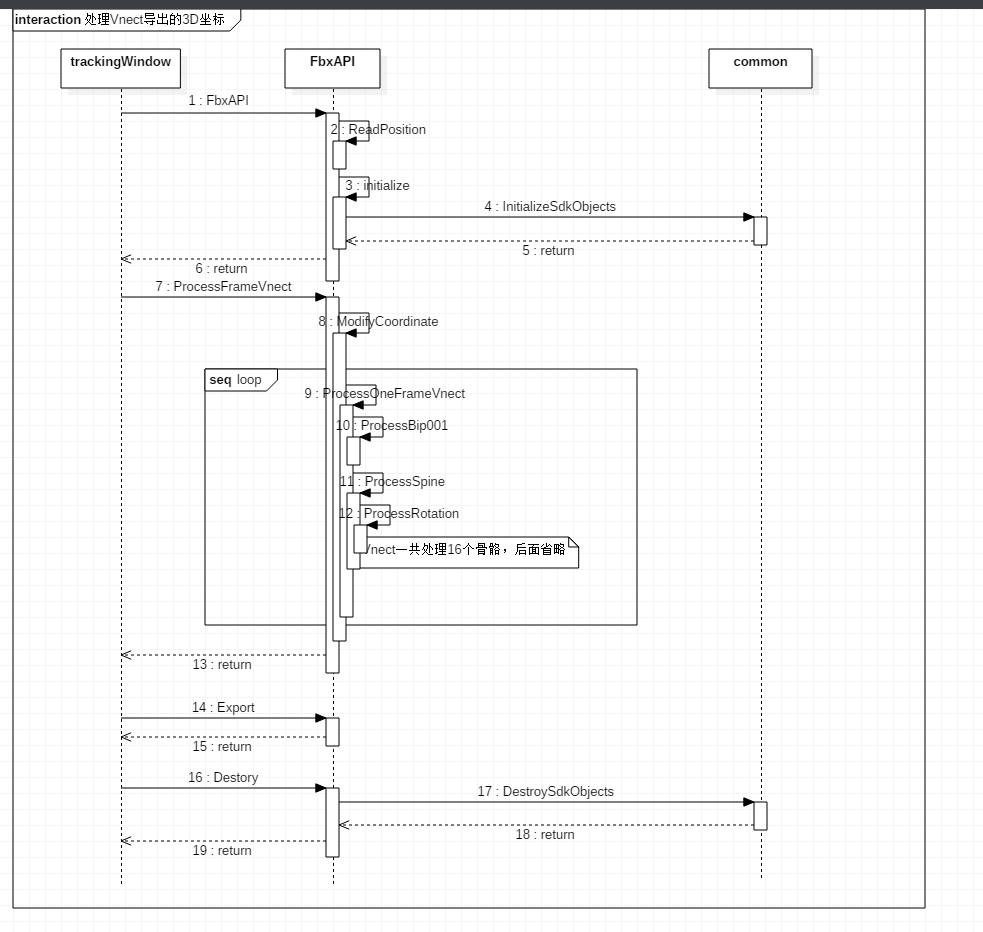


#### 3.1.6.3 3D->fbx:

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。

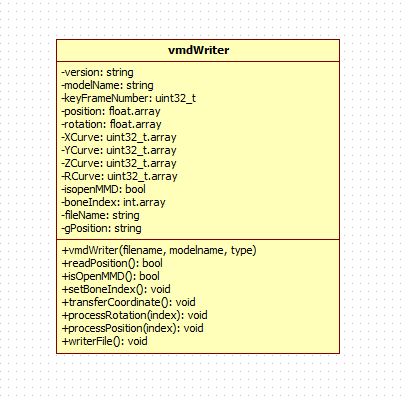
另外，类FbxAPI提供setModelPath函数，可以设置所处理的FBX原型，默认的Fbx原型只有骨骼没有模型。

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。

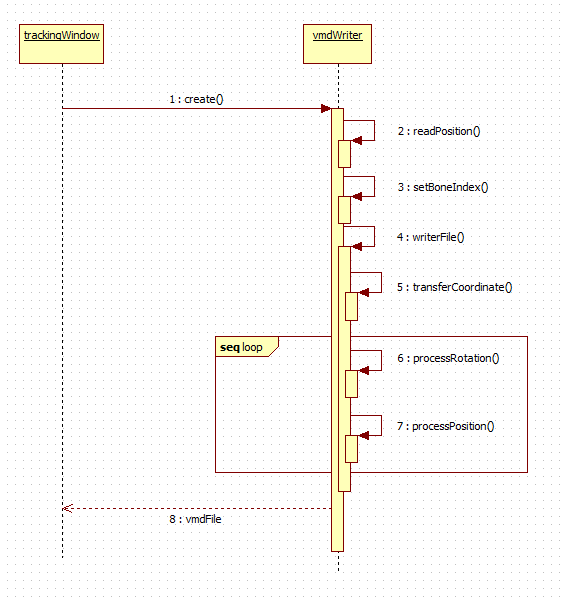
一般流程为：构造函数->选择处理方法入口->导出fbx文件->释放fbxSDK。

#### 3.1.6.4 3D->vmd:

类图：

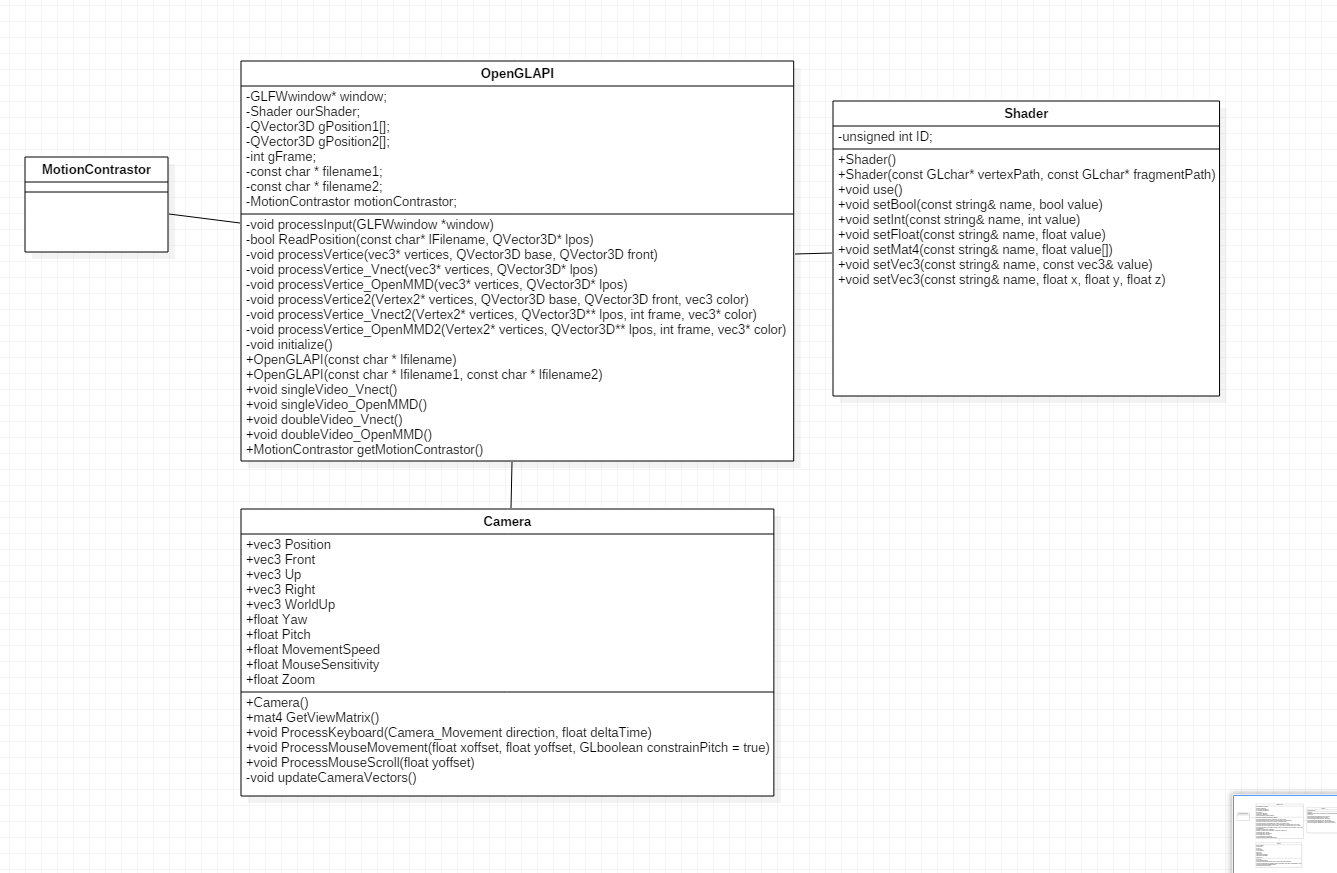


顺序图：

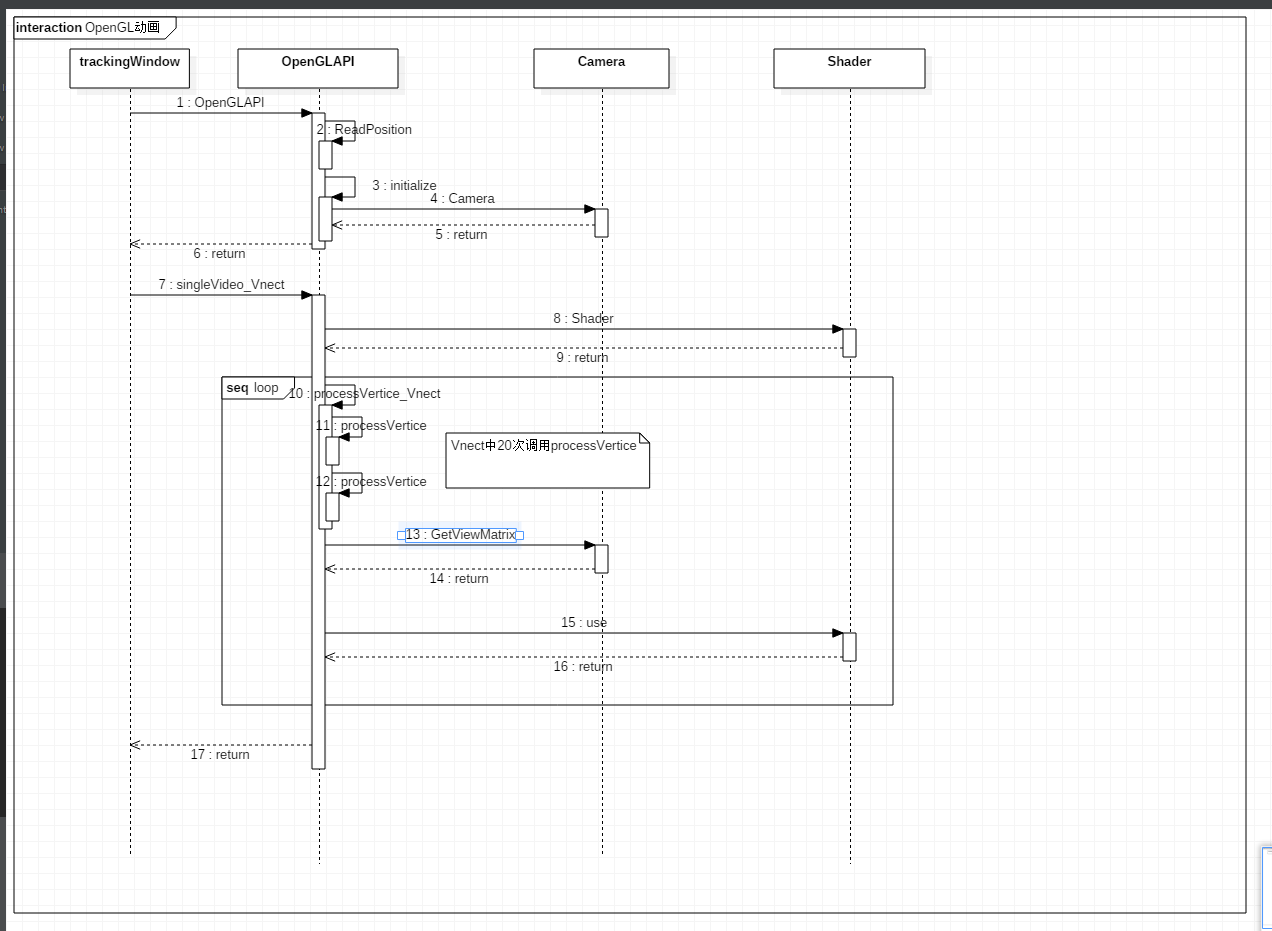


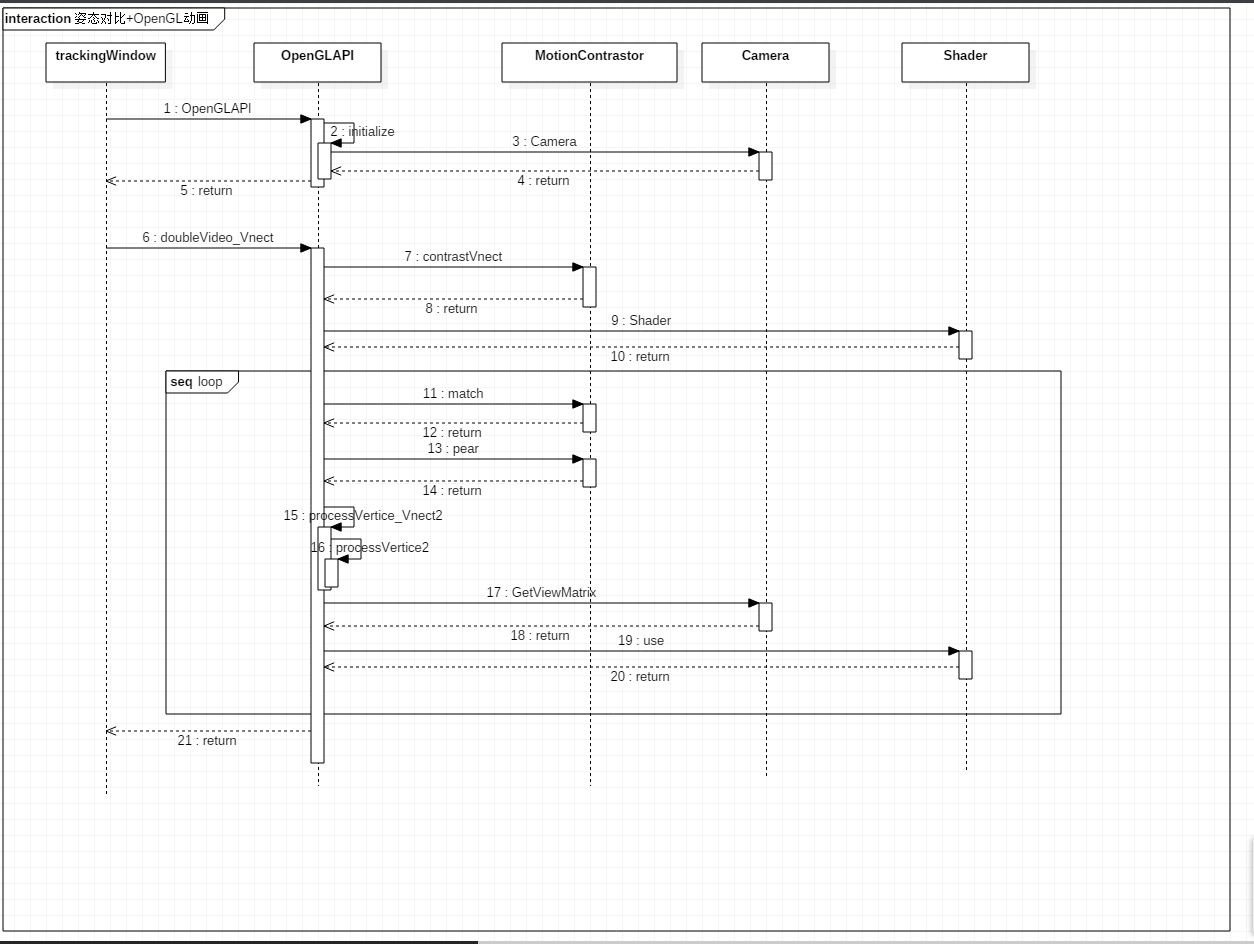
#### 3.1.6.5 OpenGL动画

类图：



顺序图：





## 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。

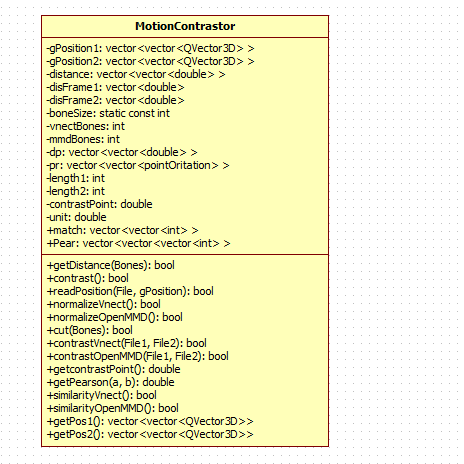
3D姿态评估主要包括：骨骼长度标准化、动作剪切、DTW算法获取对应帧、动作相识度判定

### 3.2.6 程序逻辑

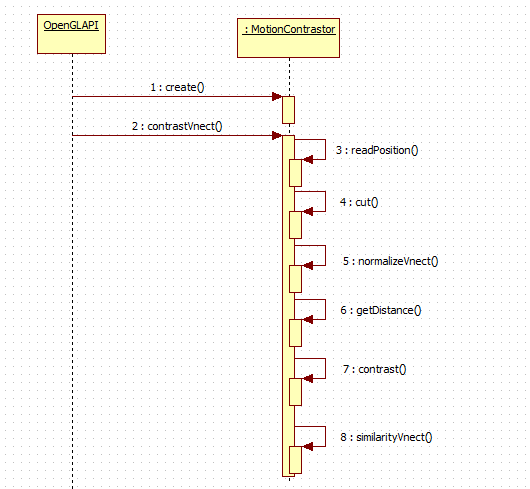
Vnect与OpenMMD的算法程序逻辑与上一模块相同。

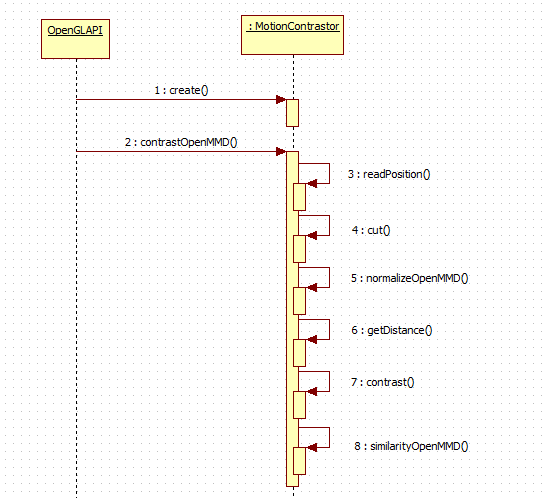
#### 3.2.6.1 3D姿态评估

类图：



顺序图：





## 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。

4 系统详细设计

4.1 系统结构设计及子系统划分

4.2 系统功能模块详细设计

4.3 系统界面详细设计