

Python SDK 说明

文档信息：

当前版本测试环境：

Python 3.10.5

CMAvatar 1.1.0.1744

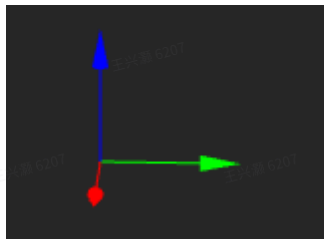
Windows 10

最后测试时间

2022-12-13

坐标系说明

chingmu 发送 vrpn 的的坐标系是按照 chingmu 坐标系，如下图坐标系显示，红色为 x 轴，绿色为 y 轴，蓝色为 z 轴。其中位置是按照 x,y,z 的顺序，旋转信息发送的是四元数，按照 x,y,z,r 的顺序。

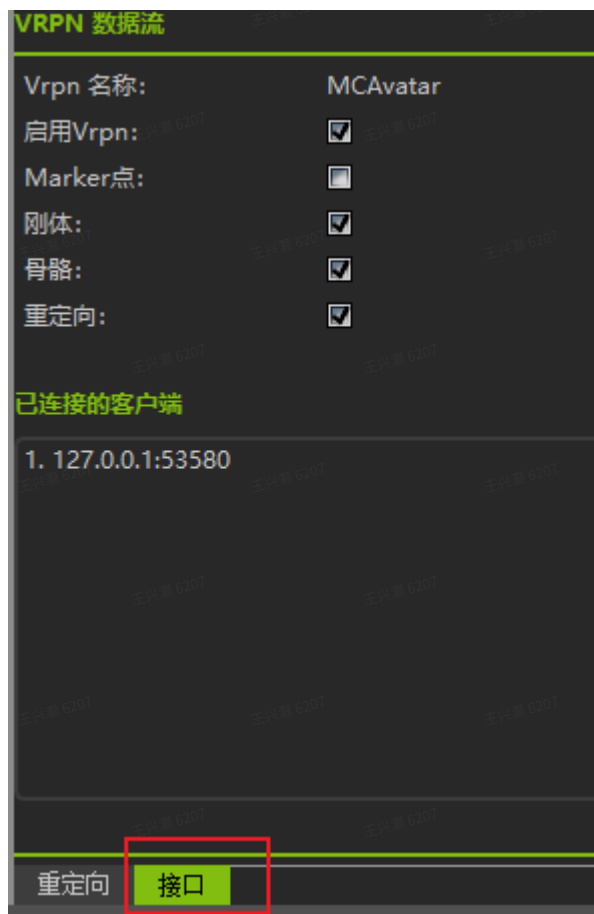


Vrpn数据说明

根据按照动捕服务器可分为两种类型。

CMAvatar:

vrpn 的地址为 MCAvatar@动捕服务器地址，默认的端口采用3883。其中，刚体数据的范围为 0~300，和动捕服务器中的刚体 id 一一对应。骨骼数据从 300开始，间隔为 150，即第一个人的第一个关节为 300，第二个人的第一个关节为 450，依此类推。重定向骨骼数据和非重定向的骨骼数据的定义一致，根据重定向发送的层级信息来确定骨骼 id与模型关节的对应关系。

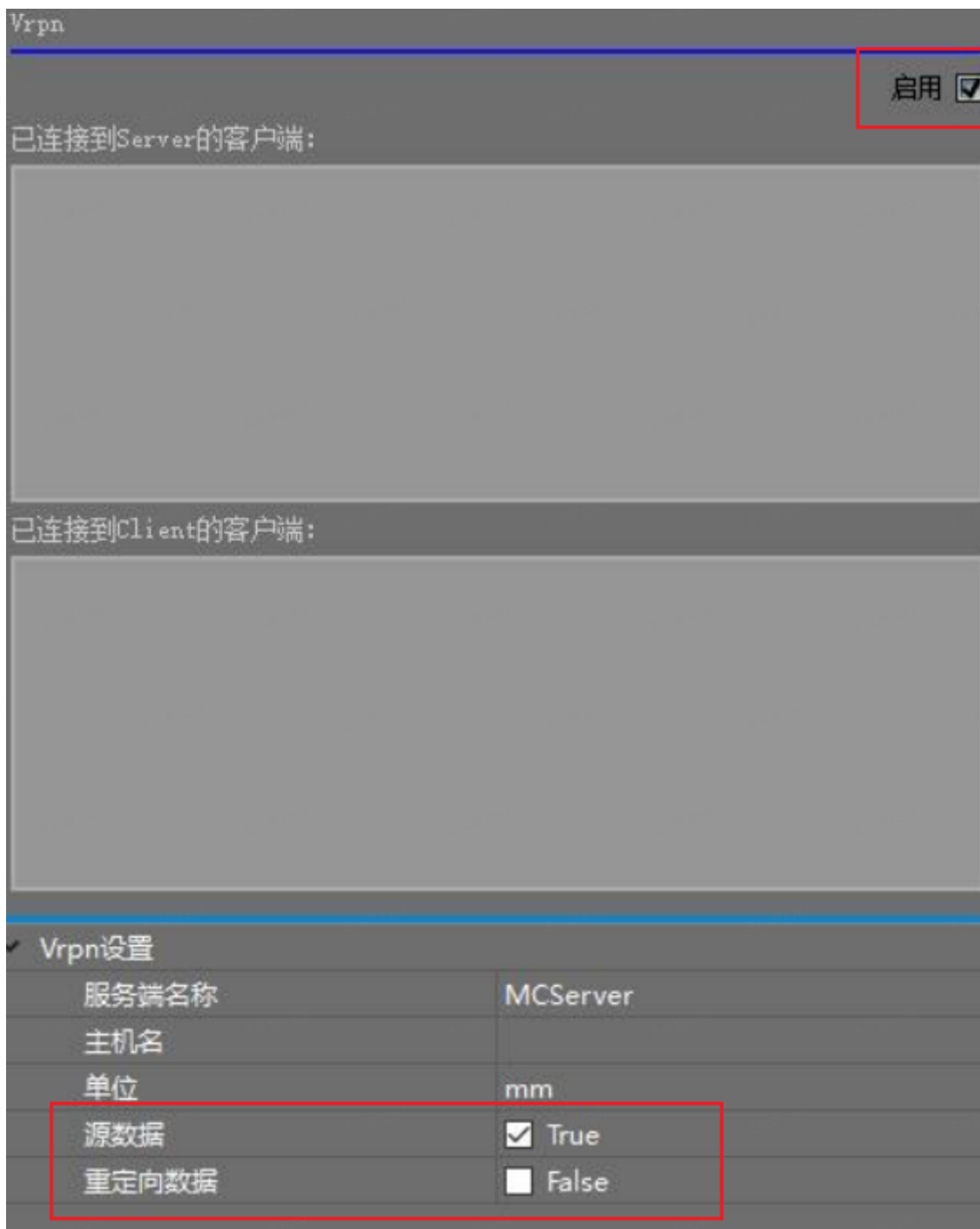


VRPN 数据流模块主要有 VRPN 服务器名称显示，控制数据流和传输数据类型的“启用/禁用”，通过右侧的复选框勾选来进行“禁用/启用”操作。若有客户端主机通过 VRPN 数据流连接上系统接收数据的话，则下方“已连接客户端”窗口会显示已连接上的客户端主机的 IP 地址。

（注：Avatar 中重定向包含重定向骨骼和重定向刚体。勾选重定向和骨骼时，会发送重定向的骨骼数据，而不是都发送）

CMTracker:

tracker server 和 tracker client 均有 vrpn server 存在。MCServer@动捕服务器地址为 tracker server，而 MCServer@动捕服务器地址:3884 为 tracker client。其中，刚体数据的范围为 0~100，和动捕服务器中的刚体 id 一一对应。骨骼的数据为 100 开始，3883 端口下，间隔为 23，3884 端口下，间隔为 150。3883 端口仅支持发送非重定向数据。3884 端口可以通过界面勾选来发送非重定向数据或者非重定向数据。



CMTracker Client 界面如上图所示。界面上可以查看到连接到 server 的 vrpn 客户端和连接到 client 的 vrpn 客户端。下方 vrpn 设置可以勾选源数据和重定向数据来设置 CMTracker Client 来发送非重定向数据还是重定向数据。

骨骼间层级关系

青瞳视觉单个人物模型的数据构成由23段骨骼和40段手指信息组成。

23段骨骼组成



index	name	index	name	index	name	index	name	index	name
0	Hips	7	LeftShoulder	11	RightShoulder	15	LeftUpLeg	19	RightUpLeg
1	Spine	8	LeftArm	12	RightArm	16	LeftLeg	20	RightLeg
2	Spine1	9	LeftForeArm	13	RightForeArm	17	LeftFoot	21	RightFoot
3	Spine2	10	LeftHand	14	RightHand	18	LeftToeBase	22	RightToeBase
4	Spine3								
5	Neck								
6	Head								

40段手指组成

index	name	index	name
23	LeftHandThumb1	43	RightHandThumb1
24	LeftHandThumb2	44	RightHandThumb2
25	LeftHandThumb3	45	RightHandThumb3
26	LeftHandThumb4	46	RightHandThumb4
27	LeftHandIndex1	47	RightHandIndex1
28	LeftHandIndex2	48	RightHandIndex2

29	LeftHandIndex3	49	RightHandIndex3
30	LeftHandIndex4	50	RightHandIndex4
31	LeftHandMiddle1	51	RightHandMiddle1
32	LeftHandMiddle2	52	RightHandMiddle2
33	_LeftHandMiddle3	53	RightHandMiddle3
34	LeftHandMiddle4	54	RightHandMiddle4
35	LeftHandRing1	55	RightHandRing1
36	LeftHandRing2	56	RightHandRing2
37	LeftHandRing3	57	RightHandRing3
38	LeftHandRing4	58	RightHandRing4
39	LeftHandPinky1	59	RightHandPinky1
40	LeftHandPinky2	60	RightHandPinky2
41	LeftHandPinky3	61	RightHandPinky3
42	LeftHandPinky4	62	RightHandPinky4

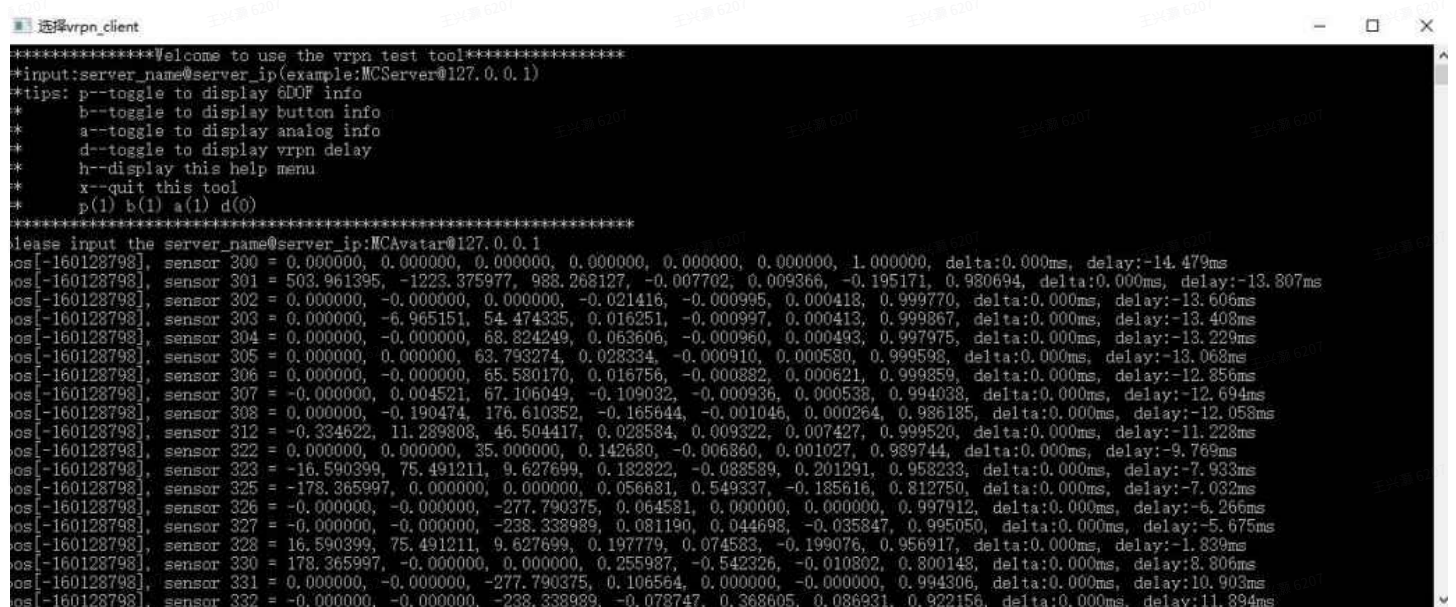
重定向说明

接收 vrpn 重定向的数据，需要先注册回调函数来获取当前模型的层级信息。层级信息 中包含模型骨骼名称，模型骨骼的 id，以及模型骨骼的 parent id，下图为某个模型的层级示例。在动捕软件中，模型所有被指认的关节的数据都会通过 vrpn 发送层级信息并发送重定向数据。

```
segment name:Dongguixue_Cloth_01Rig_all_root segment parent id:-1 segment id:300
segment name:Hips segment parent id:300 segment id:301
segment name:Spine segment parent id:301 segment id:302
segment name:Spine1 segment parent id:302 segment id:303
segment name:Spine2 segment parent id:303 segment id:304
segment name:Spine3 segment parent id:304 segment id:305
segment name:Spine4 segment parent id:305 segment id:306
segment name:Chest segment parent id:306 segment id:307
segment name:ChestMid segment parent id:307 segment id:308
segment name:Neck segment parent id:308 segment id:312
segment name:Head segment parent id:312 segment id:322
segment name:LeftShoulder segment parent id:308 segment id:323
segment name:LeftArm segment parent id:323 segment id:325
segment name:LeftForeArm segment parent id:325 segment id:326
segment name:LeftHand segment parent id:326 segment id:327
segment name:RightShoulder segment parent id:308 segment id:328
segment name:RightArm segment parent id:328 segment id:330
segment name:RightForeArm segment parent id:330 segment id:331
segment name:RightHand segment parent id:331 segment id:332
```

测试工具vrpn_client

下图为 vrpn_client 的界面图。可以通过输入对应的 server_name@server_ip 来打印 vrpn 数据，判断连接状态。数据的顺序为位置 xyz，旋转 xyzr。



```
*****Welcome to use the vrpn test tool*****
*input:server_name@server_ip(example:MCAvatar@127.0.0.1)
*tips: p--toggle to display 6DOF info
*      b--toggle to display button info
*      a--toggle to display analog info
*      d--toggle to display vrpn delay
*      h--display this help menu
*      x--quit this tool
*      p(1) b(1) a(1) d(0)
*****
Please input the server_name@server_ip:MCAvatar@127.0.0.1
os[-160128798], sensor 300 = 0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 1.000000, delta:0.000ms, delay:-14.479ms
os[-160128798], sensor 301 = 503.961395, -1223.375977, 988.268127, -0.007702, 0.009366, -0.195171, 0.980694, delta:0.000ms, delay:-13.807ms
os[-160128798], sensor 302 = 0.000000, -0.000000, 0.000000, -0.021416, -0.000995, 0.000418, 0.999770, delta:0.000ms, delay:-13.606ms
os[-160128798], sensor 303 = 0.000000, -6.965151, 54.474335, 0.016251, -0.000997, 0.000413, 0.999867, delta:0.000ms, delay:-13.408ms
os[-160128798], sensor 304 = 0.000000, -0.000000, 68.824249, 0.063606, -0.000960, 0.000493, 0.997975, delta:0.000ms, delay:-13.229ms
os[-160128798], sensor 305 = 0.000000, 0.000000, 63.793274, 0.028334, -0.000910, 0.000580, 0.999598, delta:0.000ms, delay:-13.068ms
os[-160128798], sensor 306 = 0.000000, -0.000000, 65.580170, 0.016756, -0.000882, 0.000621, 0.999859, delta:0.000ms, delay:-12.856ms
os[-160128798], sensor 307 = -0.000000, 0.004521, 67.106049, -0.109032, -0.000936, 0.000538, 0.994038, delta:0.000ms, delay:-12.694ms
os[-160128798], sensor 308 = 0.000000, -0.190474, 176.610352, -0.165644, -0.001046, 0.000264, 0.986185, delta:0.000ms, delay:-12.053ms
os[-160128798], sensor 312 = -0.334622, 11.289808, 46.504417, 0.028584, 0.009322, 0.007427, 0.999520, delta:0.000ms, delay:-11.228ms
os[-160128798], sensor 322 = 0.000000, 0.000000, 35.000000, 0.142680, -0.006860, 0.001027, 0.989744, delta:0.000ms, delay:-9.769ms
os[-160128798], sensor 323 = -16.590399, 75.491211, 9.627699, 0.182822, -0.088589, 0.201291, 0.958233, delta:0.000ms, delay:-7.933ms
os[-160128798], sensor 325 = -178.365997, 0.000000, 0.000000, 0.056681, 0.549337, -0.185616, 0.812750, delta:0.000ms, delay:-7.032ms
os[-160128798], sensor 326 = -0.000000, -0.000000, -277.790375, 0.064581, 0.000000, -0.000000, 0.997912, delta:0.000ms, delay:-6.266ms
os[-160128798], sensor 327 = -0.000000, -0.000000, -238.338989, 0.081190, 0.044698, -0.035847, 0.995050, delta:0.000ms, delay:-5.675ms
os[-160128798], sensor 328 = 16.590399, 75.491211, 9.627699, 0.197779, 0.074583, -0.199076, 0.956917, delta:0.000ms, delay:-1.839ms
os[-160128798], sensor 330 = 178.365997, -0.000000, 0.000000, 0.255987, -0.542326, -0.010802, 0.800148, delta:0.000ms, delay:8.806ms
os[-160128798], sensor 331 = 0.000000, -0.000000, -277.790375, 0.106564, 0.000000, -0.000000, 0.994306, delta:0.000ms, delay:10.903ms
os[-160128798], sensor 332 = -0.000000, -0.000000, -238.338989, -0.078747, 0.368605, 0.086931, 0.922156, delta:0.000ms, delay:11.894ms
```

API说明

CMVrpnStartExtern

开启Vrpn接口

C++:

`void CMVrpnStartExtern()`

参数:

无

返回值:

无

说明:

用于开启Vrpn模块，程序开始时调用一次。

CMVrpnQuitExtern

关闭Vrpn接口

C++:

```
void CMVrpnQuitExtern()
```

参数:

无

返回值:

无

说明:

用于关闭Vrpn模块，退出程序时调用一次。

CMVrpnEnableLog

启用/禁用日志

C++:

```
void CMVrpnEnableLog(bool enable)
```

参数:

enable为true表示启用，false表示禁用

返回值:

无

说明:

用于禁用启用日志文件记录，如果启用日志文件，则会在项目的根目录下生成追捕追踪日志文件 tracker_log.txt。

CMTrackerExtern

获取刚体信息（带预测）

C++:

```
double CMTrackerExtern(char* _address, int channel, int component, int frameCount, bool lockUpRotation = false)
```

参数:

char* _address: server端的地址

int channel: 刚体ID

int component: 获取的类型0表示X,1表示Y, 2表示Z, 3表示Qx, 4表示Qy, 5表示Qz, 6表示Qw, 7表示delay

int frameCount: 当前帧序号

bool lockUpRotation: 是否锁定旋转轴

返回值:

返回对应component的值

说明:

获取带预测的刚体数据信息

CMTrackerExternTC

获取刚体信息（非预测）

C++:

```
bool CMTrackerExternTC(char* _address, int channel, VrpnTimeCode *timecode, double T[3], double R_Quat[4]);
```

参数:

char* _address: server端的地址

int channel: 刚体ID

VrpnTimeCode *timecode: 当前时码

double T[3]: 刚体XYZ位置

double R_Quat[4]: 刚体旋转四元数XYZR

返回值:

刚体是否检测到

说明:

获取不带预测的刚体数据信息

CMHumanExtern

获取Human信息(hips全局位置+所有骨骼的局部旋转)

C++:

```
bool CMHumanExtern(char* _address, int channel, int frameCount, double* attitude, int* segmentIsDetected);
```

参数:

`char* _address`: server端的地址

`int channel`: human ID

`int frameCount`: 帧序号

`double* attribute`: 获取到的human的3+骨骼数*3属性, 包含hips的全局位置XYZ和骨骼的局部旋转的四元数XYZR

`int* segmentIsDetected`: 每段骨骼检测状态

返回值:

`bool return`: 人物检测到返回true, 否则为false.

说明:

获取human的实时姿态信息。

CMHumanGlobalTLocalRTC

获取Human信息 (所有骨骼的全局位置+局部旋转)

C++:

```
bool CMHumanGlobalTLocalRTC(char* _address, int channel, VrpnTimeCode *timecode, double* T, double* R, int* segmentIsDetected)
```

参数:

`char* _address`: server端的地址

`int channel`: human ID

`VrpnTimeCode* timecode`: 当前时码

`double* T`: 获取到的human的3+骨骼数*3属性, 包含hips的全局位置XYZ和骨骼的局部旋转的四元数XYZR

`double* R`: 获取到的骨骼数*3的局部旋转四元数XYZR

`int* segmentIsDetected`: 各个关节是否检测到

返回值:

bool return: 人物检测到返回true,否则为false.

说明:

获取Human信息 (所有骨骼的全局位置+局部旋转)

CMRetargetHumanExternTC

获取重定向Human信息 (所有模型骨骼的全局位置+局部旋转)

C++:

```
bool CMRetargetHumanExternTC(char* _address, int channel, int  
frameCount, VrpnTimeCode *timecode, double *position, double *quaternion, int*  
segmentIsDetected)
```

参数:

char* _address: server端的地址

int channel: human ID

int frameCount: 当前帧序号

VrpnTimeCode* timecode :当前时码

double *position: 获取到的骨骼数*3的全局位置信息XYZ

double *quaternion: 获取到的骨骼数*3的局部旋转四元数XYZR

int* segmentIsDetected: 各个关节是否检测到

返回值:

重定向human是否检测到

说明:

获取重定向human的信息。需要先进行重定向骨骼层级的注册，具体见第12条。

CMPluginRegisterUpdateHierarchy

获取重定向骨骼层级信息

C++:

```
bool CMPluginRegisterUpdateHierarchy(char* _address, void* userdata,  
UpdateHierarchyCallback func)
```

说明：更新骨骼层级信息时，会连续回调。每次调用只发送一个层级信息，一个模型通常会调用多次。

CMPluginRegisterResetHierarchy

重置重定向骨骼层级信息

C++:

```
bool CMPluginRegisterResetHierarchy(char* _address, void* userdata, ResetHierarchyCallback func)
```

说明：程序重新加载模型时会触发回调

CMPluginRegisterEndHierarchy

C++:

```
bool CMPluginRegisterEndHierarchy(char* _address, void* userdata, EndHierarchyCallback func)
```

说明：当层级信息发送完成后，会触发回调函数。

Demo程序说明

刚体数据演示

示例一：CMTrackerExtern

获取刚体的预测位姿信息

```
Rigid body 0 not detected
pos: X:149.611379 Y:43.328861 Z:453.462947
quaternion: rx:0.000000 ry:0.000000 rz:0.061627 rw:0.998099
pos: X:150.674592 Y:59.495115 Z:449.653584
quaternion: rx:0.000000 ry:-0.000000 rz:0.032308 rw:0.999478
pos: X:154.729669 Y:84.002130 Z:450.374086
quaternion: rx:-0.000000 ry:0.000000 rz:0.001128 rw:0.999999
pos: X:157.577326 Y:107.384253 Z:468.835195
quaternion: rx:-0.000000 ry:0.000000 rz:0.013339 rw:0.999911
```

```
1 import os
2 import sys
```

```

3 import time
4 from ctypes import *
5
6 # Load dynamic Library
7 def LoadDll(dllPath):
8     if(os.path.exists(dllPath)):
9         return CDLL(dllPath)
10    else:
11        print("Chingmu's dynamic Library does not exist \n")
12        sys.exit()
13
14 # get body(id:0) component 0:x,1:y,2:z,3:rx,4:ry,5:rz,6:rw with time predict
15 # warning : When using this function, the 'frameCount' must increase with the nu
16 def CMTrackerExtern(host,bodyID,frameCount):
17     bodyPos = (c_double * 3)()
18     bodyRot = (c_double * 4)()
19     trackerExtern = cmVrpn.CMTrackerExtern
20     trackerExtern.restype = c_double
21
22     bodyPos[0] = cmVrpn.CMTrackerExtern(host, bodyID, 0, frameCount)
23     bodyPos[1] = cmVrpn.CMTrackerExtern(host, bodyID, 1, frameCount)
24     bodyPos[2] = cmVrpn.CMTrackerExtern(host, bodyID, 2, frameCount)
25
26     bodyRot[0] = cmVrpn.CMTrackerExtern(host, bodyID, 3, frameCount)
27     bodyRot[1] = cmVrpn.CMTrackerExtern(host, bodyID, 4, frameCount)
28     bodyRot[2] = cmVrpn.CMTrackerExtern(host, bodyID, 5, frameCount)
29     bodyRot[3] = cmVrpn.CMTrackerExtern(host, bodyID, 6, frameCount)
30
31     # check body detected, must call after CMTrackerExtern
32     isBodyDetected = cmVrpn.CMTrackerExternIsDetected(host, bodyID, frameCount)
33     if (isBodyDetected):
34         print("pos: X:%f Y:%f Z:%f"%(bodyPos[0], bodyPos[1], bodyPos[2]))
35         print("quaternion: rx:%f ry:%f rz:%f rw:%f"%(bodyRot[0], bodyRot[1], bod
36     else:
37         print("Rigid body %d not detected"%(bodyID))
38     return isBodyDetected
39
40 if __name__ == '__main__':
41     # Set dynamic Library path
42     dllPath = os.path.dirname(os.path.dirname(__file__)) + "\\ChingmuDLL\\CMVrpn
43
44     # Load dynamic Library
45     cmVrpn = LoadDll(dllPath)
46
47     # set server address
48     host = bytes("MCAvatar@127.0.0.1", "gbk")
49

```

```

50 # start vrpn thread
51 cmVrpn.CMVrpnStartExtern()
52
53 # enable write trace_log.txt
54 cmVrpn.CMVrpnEnableLog(True)
55
56 # Person ID displayed on the server
57 bodyID = 0
58
59 frameCount = 0
60
61 getfailed = 0
62 loopState = True
63 while(loopState):
64     if(frameCount > 100):
65         loopState = False
66
67     # Control acquisition frequency.parameters can be customized.
68     time.sleep(0.2)
69
70     isDataDetected = CMTrackerExtern(host, bodyID, frameCount)
71     if(isDataDetected):
72         getfailed = 0
73     else:
74         getfailed += 1
75
76     if(getfailed > 10):
77         print("Continuous acquisition failed, exit the program.")
78         loopState = False
79
80     frameCount += 1
81
82 # quit vrpn thread
83 cmVrpn.CMVrpnQuitExtern()

```

示例二：CMTrackerExternTC

演示，获取刚体位姿信息

```

Rigid body 0 not detected
server frame num: 18109374
pos: X:149.956665 Y:99.899139 Z:33.217464
quaternion: rx:-0.020764 ry:-0.004075 rz:-0.000673 rw:0.999776
server frame num: 18109398
pos: X:142.796829 Y:88.820969 Z:31.554661
quaternion: rx:-0.016852 ry:-0.004820 rz:0.009810 rw:0.999798

```

```

1  import os
2  import sys
3  import time
4  from ctypes import *
5
6  # Load dynamic Library
7  def LoadDll(dllPath):
8      if(os.path.exists(dllPath)):
9          return CDLL(dllPath)
10     else:
11         print("Chingmu's dynamic Library does not exist \n")
12         sys.exit()
13
14  def PrintTimecode(timecode):
15      standard = ((timecode & 0x60000000) >> 29)
16      hours = ((timecode & 0x1f000000) >> 24)
17      minutes = ((timecode & 0x00fc0000) >> 18)
18      seconds = ((timecode & 0x0003f000) >> 12)
19      frames = ((timecode & 0x00000fe0) >> 5)
20      subframes = (timecode & 0x0000001f)
21      print("Time code: %d:%d:%d:%d" % (hours, minutes, seconds, frames))
22
23  # get body(id:0) T(XYZ 3), body Quaternion(XYZR 4), timecode without time predic
24  def CMTrackerExternTC(host, bodyID, frameCount):
25      bodyPos = (c_double * 3)()
26      bodyRot = (c_double * 4)()
27      timecodeData = (c_int * 1)()
28      isBodyDetected = cmVrpn.CMTrackerExternTC(host, bodyID, timecodeData, bodyPo
29
30      if (isBodyDetected):
31          timecode = timecodeData[0]
32          valid = ((timecode & 0x80000000) >> 31)
33          if (True == valid):
34              PrintTimecode(timecode)
35          else:
36              print("server frame num: %d" % (timecode))
37
38          print("pos: X:%f Y:%f Z:%f"%(bodyPos[0], bodyPos[1], bodyPos[2]))
39          print("quaternion: rx:%f ry:%f rz:%f rw:%f"%(bodyRot[0], bodyRot[1], bod
40      else:
41          print("Rigid body %d not detected" % (bodyID))
42      return isBodyDetected
43
44  if __name__ == '__main__':
45      # Set dynamic Library path
46      dllPath = os.path.dirname(os.path.dirname(__file__)) + "\\ChingmuDLL\\CMVrpn
47

```

```

48     # Load dynamic Library
49     cmVrpn = LoadDll(dllPath)
50
51     # set server address
52     host = bytes("MCAvatar@127.0.0.1", "gbk")
53
54     # start vrpn thread
55     cmVrpn.CMVrpnStartExtern()
56
57     # enable write trace_log.txt
58     cmVrpn.CMVrpnEnableLog(True)
59
60     # Person ID displayed on the server
61     bodyID = 0
62
63     frameCount = 0
64
65     getfailed = 0
66     loopState = True
67     while(loopState):
68         if(frameCount > 100):
69             loopState = False
70
71         # Control acquisition frequency.parameters can be customized.
72         time.sleep(0.2)
73
74         isDataDetected = CMTrackerExternTC(host, bodyID, frameCount)
75         if(isDataDetected):
76             getfailed = 0
77         else:
78             getfailed += 1
79
80         if(getfailed > 10):
81             print("Continuous acquisition failed, exit the program.")
82             loopState = False
83
84         frameCount += 1
85
86     # quit vrpn thread
87     cmVrpn.CMVrpnQuitExtern()

```

人物数据演示

示例三：CMHumanExtern

获取Human信息(hips全局位置+所有骨骼的局部旋转)

```
Human 0 not detected
pos: X:902.871704 Y:-2031.988281 Z:915.726074
quaternion: rx:0.033834 ry:-0.003111 rz:-0.457686 rw:0.888464
pos: X:1003.315796 Y:-1945.988647 Z:912.664673
quaternion: rx:0.025472 ry:0.013526 rz:-0.349599 rw:0.936456
pos: X:1136.766357 Y:-1849.657104 Z:906.059143
quaternion: rx:0.023703 ry:-0.002822 rz:-0.201595 rw:0.979178
```

```
1 import os
2 import sys
3 import time
4 from ctypes import *
5
6 MAX_SEGMENT_NUM = 150
7
8 # Load dynamic Library
9 def LoadDll(dllPath):
10     if(os.path.exists(dllPath)):
11         return CDLL(dllPath)
12     else:
13         print("Chingmu's dynamic Library does not exist \n")
14         sys.exit()
15
16 # get human(id:0) attitude(hips pos XYZ(3) + segments quaternion XYZW(segmentNum * 4)), segmentIsDetected(segmentNum)
17 def CMHumanExtern(host, humanID, frameCount):
18     attitude = (c_double * (3 + MAX_SEGMENT_NUM * 4))()
19     isDetected = (c_int * MAX_SEGMENT_NUM)()
20     isHumanDetected = cmVrpn.CMHumanExtern(host, humanID, frameCount, attitude, isDetected)
21     if(True == isHumanDetected):
22         print("pos: X:%f Y:%f Z:%f"%(attitude[0], attitude[1], attitude[2]))
23         print("quaternion: rx:%f ry:%f rz:%f rw:%f"%(attitude[3], attitude[4], attitude[5], attitude[6]))
24     else:
25         print("Human %d not detected" % (humanID))
26     return isHumanDetected
27
28 if __name__ == '__main__':
29     # Set dynamic Library path
30     dllPath = os.path.dirname(os.path.dirname(__file__)) + "\\ChingmuDLL\\CMVrpn.dll"
31
32     # Load dynamic Library
33     cmVrpn = LoadDll(dllPath)
34
35     # set server address
```



```

36     host = bytes("MCAvatar@127.0.0.1", "gbk")
37
38     # start vrpn thread
39     cmVrpn.CMVrpnStartExtern()
40
41     # enable write trace_log.txt
42     cmVrpn.CMVrpnEnableLog(True)
43
44     # Person ID displayed on the server
45     humanID = 0
46
47     frameCount = 0
48     getfailed = 0
49     loopState = True
50     while(loopState):
51         # Control acquisition frequency.parameters can be customized.
52         time.sleep(0.2)
53
54         if(frameCount > 100):
55             loopState = False
56
57         isDataDetected = CMHumanExtern(host, humanID, frameCount)
58
59         if(isDataDetected):
60             getfailed = 0
61             frameCount += 1
62         else:
63             getfailed += 1
64
65         if(getfailed > 10):
66             print("Continuous acquisition failed, exit the program.")
67             loopState = False
68
69     # quit vrpn thread
70     cmVrpn.CMVrpnQuitExtern()

```

示例四：CMHumanGlobalTLocalRTC

获取Human信息（所有骨骼的全局位置+局部旋转）

```

Human 0 not detected
Time code: 7:56:32:26
pos: X:972.828674 Y:-1161.645386 Z:930.468201
quaternion: rx:0.033592 ry:-0.001078 rz:0.768324 rw:0.639179
Time code: 7:56:33:2
pos: X:923.324341 Y:-1090.500977 Z:931.297424
quaternion: rx:0.012409 ry:-0.014296 rz:0.823992 rw:0.566285

```

```

1  import os
2  import sys
3  import time
4  from ctypes import *
5
6  MAX_SEGMENT_NUM = 150
7
8  # Load dynamic Library
9  def LoadDll(dllPath):
10     if(os.path.exists(dllPath)):
11         return CDLL(dllPath)
12     else:
13         print("Chingmu's dynamic Library does not exist \n")
14         sys.exit()
15
16  def PrintTimecode(timecode):
17     standard = ((timecode & 0x60000000) >> 29)
18     hours = ((timecode & 0x1f000000) >> 24)
19     minutes = ((timecode & 0x00fc0000) >> 18)
20     seconds = ((timecode & 0x0003f000) >> 12)
21     frames = ((timecode & 0x00000fe0) >> 5)
22     subframes = (timecode & 0x0000001f)
23     print("Time code: %d:%d:%d:%d" % (hours, minutes, seconds, frames))
24
25  # get human(id: 0) T(segmentNum * 3), localR(segmentNum * 4), segmentIsDetected(
26  def CMHumanGlobalTLocalRTC(host, humanID, frameCount):
27     humanT = (c_double * (MAX_SEGMENT_NUM * 3))()
28     humanLocalR = (c_double * (MAX_SEGMENT_NUM * 4))()
29     segmentIsDetected = (c_int * MAX_SEGMENT_NUM)()
30     timecodeData = (c_int * 1)()
31     isHumanDetected = cmVrpn.CMHumanGlobalTLocalRTC(host, humanID, timecodeData,
32
33     if (True == isHumanDetected):
34         timecode = timecodeData[0]
35         valid = ((timecode & 0x80000000) >> 31)
36         if (True == valid):
37             PrintTimecode(timecode)
38         else:
39             print("server frame num: %d"%(timecode))
40
41         print("pos: X:%f Y:%f Z:%f" % (humanT[0], humanT[1], humanT[2]))
42         print("quaternion: rx:%f ry:%f rz:%f rw:%f" % (humanLocalR[0], humanLocalR[1], humanLocalR[2], humanLocalR[3]))
43     else:
44         print("Human %d not detected" % (humanID))
45     return isHumanDetected
46

```

```

47 if __name__ == '__main__':
48     # Set dynamic Library path
49     dllPath = os.path.dirname(os.path.dirname(__file__)) + "\\ChingmuDLL\\CMVrpn
50
51     # Load dynamic Library
52     cmVrpn = LoadDll(dllPath)
53
54     # set server address
55     host = bytes("MCAvatar@127.0.0.1", "gbk")
56
57     # start vrpn thread
58     cmVrpn.CMVrpnStartExtern()
59
60     # enable write trace_log.txt
61     cmVrpn.CMVrpnEnableLog(True)
62
63     # Person ID displayed on the server
64     humanID = 0
65
66     frameCount = 0
67     getfailed = 0
68     loopState = True
69     while(loopState):
70         # Control acquisition frequency.parameters can be customized.
71         time.sleep(0.2)
72
73         if(frameCount > 100):
74             loopState = False
75
76         isDataDetected = CMHumanGlobalToLocalRTC(host,humanID,frameCount)
77
78         if(isDataDetected):
79             getfailed = 0
80             frameCount += 1
81         else:
82             getfailed += 1
83
84         if(getfailed > 10):
85             print("Continuous acquisition failed, exit the program.")
86             loopState = False
87
88     # quit vrpn thread
89     cmVrpn.CMVrpnQuitExtern()

```

示例五：CMRetargetHumanExternTC

获取重定向Human信息（所有模型骨骼的全局位置+局部旋转）

```
Human 0 not detected
Time code: 7:56:32:7
pos: X:1258.805542 Y:-1339.968262 Z:994.510010
quaternion: rx:-0.017994 ry:0.009228 rz:0.007112 rw:0.999770
Time code: 7:56:32:13
pos: X:1158.502686 Y:-1296.250122 Z:990.395203
quaternion: rx:-0.026228 ry:-0.007229 rz:0.079801 rw:0.996439
```

```
1 import os
2 import sys
3 import time
4 from ctypes import *
5
6 MAX_SEGMENT_NUM = 150
7
8 # Load dynamic Library
9 def LoadDll(dllPath):
10     if(os.path.exists(dllPath)):
11         return CDLL(dllPath)
12     else:
13         print("Chingmu's dynamic Library does not exist \n")
14         sys.exit()
15
16 def PrintTimecode(timecode):
17     standard = ((timecode & 0x60000000) >> 29)
18     hours = ((timecode & 0x1f000000) >> 24)
19     minutes = ((timecode & 0x00fc0000) >> 18)
20     seconds = ((timecode & 0x0003f000) >> 12)
21     frames = ((timecode & 0x00000fe0) >> 5)
22     subframes = (timecode & 0x0000001f)
23     print("Time code: %d:%d:%d:%d" % (hours, minutes, seconds, frames))
24
25 # get human(id:0) humanT XYZ(segmentNum * 3), humanLocalR XYZW(segmentNum * 4),
26 # humanT[0:3] is root node position, humanLocalR[0:4] is root node rotation.
27 def CMRetargetHumanExternTC(host, humanID, frameCount):
28     humanT = (c_double * (MAX_SEGMENT_NUM * 3))()
29     humanLocalR = (c_double * (MAX_SEGMENT_NUM * 4))()
30     segmentIsDetected = (c_int * MAX_SEGMENT_NUM)()
31     timecodeData = (c_int * 1)()
32     isHumanDetected = cmVrpn.CMRetargetHumanExternTC(host, humanID, frameCount,
33
34     if (True == isHumanDetected):
35         timecode = timecodeData[0]
36         valid = ((timecode & 0x80000000) >> 31)
```

```
37         if (True == valid):
38             PrintTimecode(timecode)
39         else:
40             print("server frame num: %d" % (timecode))
41
42     print("pos: X:%f Y:%f Z:%f" % (humanT[3], humanT[4], humanT[5]))
43     print("quaternion: rx:%f ry:%f rz:%f rw:%f" % (humanLocalR[4], humanLocalR[5], humanLocalR[6], humanLocalR[7]))
44 else:
45     print("Human %d not detected" % (humanID))
46     return isHumanDetected
47
48 if __name__ == '__main__':
49     # Set dynamic Library path
50     dllPath = os.path.dirname(os.path.dirname(__file__)) + "\\ChingmuDLL\\CMVrpn.dll"
51
52     # Load dynamic Library
53     cmVrpn = LoadDll(dllPath)
54
55     # set server address
56     host = bytes("MCAvatar@127.0.0.1", "gbk")
57
58     # start vrpn thread
59     cmVrpn.CMVrpnStartExtern()
60
61     # enable write trace_log.txt
62     cmVrpn.CMVrpnEnableLog(True)
63
64     # Person ID displayed on the server
65     humanID = 0
66
67     frameCount = 0
68     getfailed = 0
69     loopState = True
70     while(loopState):
71         # Control acquisition frequency.parameters can be customized.
72         time.sleep(0.2)
73
74         if(frameCount > 100):
75             loopState = False
76
77         isDataDetected = CMRetargetHumanExternTC(host, humanID, frameCount)
78
79         if(isDataDetected):
80             getfailed = 0
81             frameCount += 1
82         else:
83             getfailed += 1
```

```

84
85         if(getfailed > 10):
86             print("Continuous acquisition failed, exit the program.")
87             loopState = False
88
89     # quit vrpn thread
90     cmVrpn.CMVrpnQuitExtern()

```

层级信息演示

示例六：CMPluginRegisterUpdateHierarchy

```

segment name:Skeleton0_Hips segment parent id:-1 segment id:300
segment name:Skeleton0_Spine segment parent id:300 segment id:301
segment name:Skeleton0_Spine1 segment parent id:301 segment id:302
segment name:Skeleton0_Spine2 segment parent id:302 segment id:303
segment name:Skeleton0_Spine3 segment parent id:303 segment id:304
segment name:Skeleton0_Neck segment parent id:304 segment id:305
segment name:Skeleton0_Head segment parent id:305 segment id:306
segment name:Skeleton0_LeftShoulder segment parent id:304 segment id:307
segment name:Skeleton0_LeftArm segment parent id:307 segment id:308
segment name:Skeleton0_LeftForeArm segment parent id:308 segment id:309
segment name:Skeleton0_LeftHand segment parent id:309 segment id:310
segment name:Skeleton0_RightShoulder segment parent id:304 segment id:311
segment name:Skeleton0_RightArm segment parent id:311 segment id:312
segment name:Skeleton0_RightForeArm segment parent id:312 segment id:313
segment name:Skeleton0_RightHand segment parent id:313 segment id:314
segment name:Skeleton0_LeftUpLeg segment parent id:300 segment id:315
segment name:Skeleton0_LeftLeg segment parent id:315 segment id:316
segment name:Skeleton0_LeftFoot segment parent id:316 segment id:317
segment name:Skeleton0_LeftToeBase segment parent id:317 segment id:318
segment name:Skeleton0_RightUpLeg segment parent id:300 segment id:319
segment name:Skeleton0_RightLeg segment parent id:319 segment id:320
segment name:Skeleton0_RightFoot segment parent id:320 segment id:321
segment name:Skeleton0_RightToeBase segment parent id:321 segment id:322

```

```

1  import os
2  import sys
3  import time
4  from ctypes import *
5
6  class timeval(Structure):
7      _fields_ = [("tv_sec",c_long),
8                  ("tv_usec",c_long)]
9
10
11 class VrpHierarchy(Structure):

```

```

12     _fields_ = [("msg_time", timeval),
13                 ("sensor", c_int),
14                 ("parent", c_int),
15                 ("name", c_char*127)]
16
17 # Load dynamic Library
18 def LoadDll(dllPath):
19     if os.path.exists(dllPath):
20         return CDLL(dllPath)
21     else:
22
23         print("Chingmu's dynamic Library does not exist \n")
24         sys.exit()
25
26 # get hierarchy info
27 def CallbackUpdateHierarchy(voidPtr, hierarchy):
28     print("segment name:%s segment parent id:%d segment id:%d"%(hierarchy.name.
29         decode(), hierarchy.parent, hierarchy.sensor))
30
31 if __name__ == '__main__':
32     # Set dynamic Library path
33     dllPath = os.path.dirname(os.path.dirname(__file__)) + "\\ChingmuDLL\\CMVrp
34     n.dll"
35
36     # Load dynamic Library
37     cmVrpn = LoadDll(dllPath)
38
39     # set server address
40     host = bytes("MCAvatar@127.0.0.1", "gbk")
41
42     # start vrpn thread
43     cmVrpn.CMVrpnStartExtern()
44
45     # enable write trace_log.txt
46     cmVrpn.CMVrpnEnableLog(True)
47
48     userData = VrpnHierarchy(timeval(c_int(0), c_int(0)), c_int(0), c_int(0), b"0"
49         *127)
50
51     callbackUpdata = CFUNCTYPE(None, c_char_p, VrpnHierarchy)(CallbackUpdateHiera
52         rchy)
53
54     # register human model hierarchy
55     cmVrpn.CMPluginRegisterUpdateHierarchy(host, byref(userData), callbackUpdat
56         a)
57
58     loopState = True

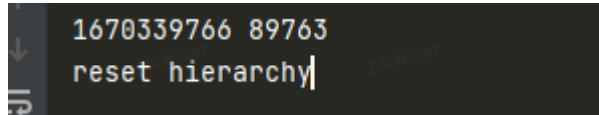
```

```

54     while(loopState):
55         time.sleep(0.2)
56
57     cmVrpn.CMPluginUnRegisterUpdateHierarchy(host, byref(userData), CallbackUpd
ateHierarchy)
58
59     # quit vrpn thread
60     cmVrpn.CMVrpnQuitExtern()

```

示例七：CMPluginRegisterResetHierarchy



```

1670339766 89763
↓
reset hierarchy|

```

```

1  import os
2  import sys
3  import time
4  from ctypes import *
5
6  class timeval(Structure):
7      _fields_ = [("tv_sec",c_long),
8                  ("tv_usec",c_long)]
9
10
11 class VrpnHierarchy(Structure):
12     _fields_ = [("msg_time", timeval),
13                 ("sensor", c_int),
14                 ("parent", c_int),
15                 ("name", c_char*123)]
16
17 initHierarchy = [timeval(c_int(0),c_int(0)), c_int(0),c_int(0),b"0"*123]
18
19 # Load dynamic Library
20 def LoadDll(dllPath):
21     if(os.path.exists(dllPath)):
22         return CDLL(dllPath)
23     else:
24         print("Chingmu's dynamic Library does not exist \n")
25         sys.exit()
26
27 def CallbackResetHierarchy(voidPtr,msg):
28     print(msg.tv_sec,msg.tv_usec)
29     print("reset hierarchy\n")
30
31 if __name__ == '__main__':

```



```

32 # Set dynamic Library path
33 dllPath = os.path.dirname(os.path.dirname(__file__)) + "\\ChingmuDLL\\CMVrpn
34
35 # Load dynamic Library
36 cmVrpn = LoadDll(dllPath)
37
38 # set server address
39 host = bytes("MCAvatar@127.0.0.1", "gbk")
40
41 # start vrpn thread
42 cmVrpn.CMVrpnStartExtern()
43
44 # enable write trace_log.txt
45 cmVrpn.CMVrpnEnableLog(True)
46
47 userData = VrpnHierarchy(*initHierarchy)
48
49 callbackReset = CFUNCTYPE(None, c_char_p, timeval)(CallbackResetHierarchy)
50
51 # register human model reset hierarchy
52 cmVrpn.CMPluginRegisterResetHierarchy(host, byref(userData), callbackReset)
53
54 loopState = True
55 while(loopState):
56     time.sleep(0.2)
57
58 cmVrpn.CMPluginUnRegisterResetHierarchy(host, byref(userData), CallbackReset
59
60 # quit vrpn thread
61 cmVrpn.CMVrpnQuitExtern()

```

示例八：CMPluginRegisterEndHierarchy

```

sec:1670959510,usec306341
0 - source, 1 - retarget:1

```

```

1 import os
2 import sys
3 import time
4 from ctypes import *
5
6 class timeval(Structure):
7     _fields_ = [("tv_sec", c_long),
8                 ("tv_usec", c_long)]
9

```

```

10 class VrpnEndHierarchyMsg(Structure):
11     _fields_ = [("msg_time",timeval),
12                 ("retarget_flag",c_int)] # 0 - source, 1 - retarget
13
14 # Load dynamic Library
15 def LoadDll(dllPath):
16     if(os.path.exists(dllPath)):
17         return CDLL(dllPath)
18     else:
19         print("Chingmu's dynamic Library does not exist \n")
20         sys.exit()
21
22 def CallbackVrpnEndHierarchy(voidPtr, endMsg):
23     print("sec:%d,usec%d"%(endMsg.msg_time.tv_sec,endMsg.msg_time.tv_usec))
24     print("0 - source, 1 - retarget:%d"%(endMsg.retarget_flag))
25
26 if __name__ == '__main__':
27     # Set dynamic Library path
28     dllPath = os.path.dirname(os.path.dirname(__file__)) + "\\ChingmuDLL\\CMVrpn
29
30     # Load dynamic Library
31     cmVrpn = LoadDll(dllPath)
32
33     # set server address
34     host = bytes("MCAvatar@127.0.0.1", "gbk")
35
36     # start vrpn thread
37     cmVrpn.CMVrpnStartExtern()
38
39     # enable write trace_log.txt
40     cmVrpn.CMVrpnEnableLog(True)
41
42     userData = VrpnHierarchy(timeval(c_int(0),c_int(0)), c_int(0),c_int(0),b"0"*
43
44     callbackFinishe = CFUNCTYPE(None, c_char_p, VrpnEndHierarchyMsg)(CallbackVrp
45
46     # Register the callback function of sending completion signal
47     cmVrpn.CMPluginRegisterEndHierarchy(host, byref(userData), callbackFinishe)
48
49     loopState = True
50     while(loopState):
51         time.sleep(0.2)
52
53     cmVrpn.CMPluginUnRegisterEndHierarchy(host, byref(userData), callbackFinishe)
54     # quit vrpn thread
55     cmVrpn.CMVrpnQuitExtern()

```

