**Full body motion capture using Vive Tracker**

Hey, my friend! Here you are: [code](https://github.com/JamesBear/vive_ik_demo), [application](https://github.com/JamesBear/vive_ik_demo/raw/master/vive_ik_test.rar). Thanks! Bye.

...

Yes? Oh, you want to know how to implement your own motion capture in 5 minutes? Read on then.

Contents：

What is full body motion capture?

Hardware and software

How it works

     Non-IK part

**IK part**

          What is IK

          Three-point IK is more accurate

          Selecting the "bend plane"

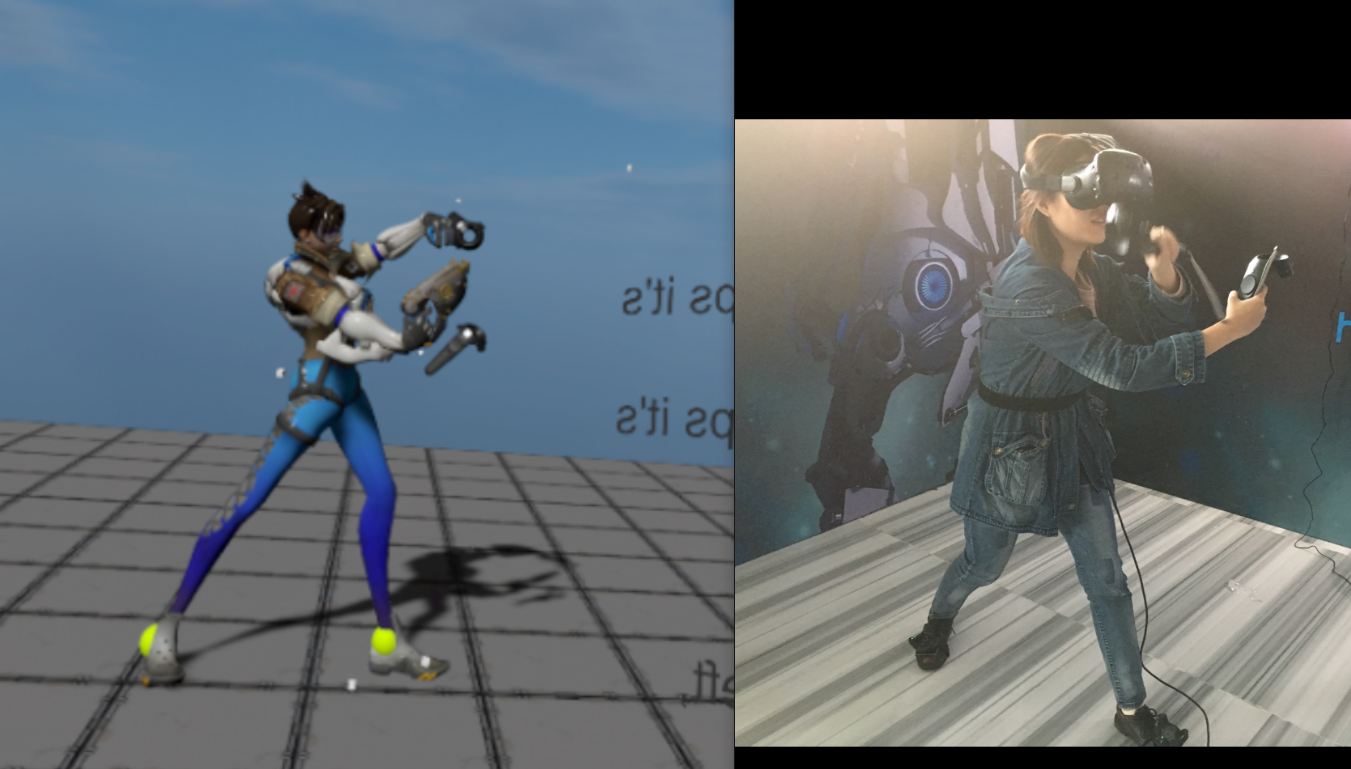
     Calibration

How to attach trackers to your body

**What is full body motion capture?**

In a word, full body motion capture is capturing your motion and restoring it in the virtual world.





**Hardware and software**

Software: unity, steamvr plugin, IK plugin(in our case, Final IK), a 3D model with bones

Hardware: Vive\*1, Tracker\*3, \*accessories to help attach trackers

**How it works**

The goal of our program is to make the model pose exactly the same as the player ... by moving her bones.

How can we achieve this?

**TL; DR**: track the player's feet and waist with trackers, and hands with controllers, then calculate the positions of the knees and the elbows using IK.

Let's see how each of them works in detail.

**Non-IK part**：

We can derive the positions of feet, the waist and hands from positions of trackers and controllers.

**Feet**：



Just place a tracker on top of each foot, and you'll know the pose (position and rotation) of the foot - because the tracker-to-foot offset is fixed. Let's say that their offset is matrix "trans", then we have

     pose\_foot = pose\_tracker \* trans;

Of course in Unity we don't need to write the equation ourselves. Simply set the Tracker as the foot's parent object. This way, when the Tracker moves, the foot will follow.

> This is only how it would work in theory. In reality, we can't program this way. We'll break the model's bone structure if we change the parent object of her foot. The right way to do this is using an agent GameObject. Let the GameObject be the child object of Tracker, and then assign the GameObject's position and rotation to the model's foot in every frame.

**Hands**：

What's unique about hand tracking is that almost everyone holds the controller in the same way. Thus it's safe to assume the hand-controller offset remains unchanged whoever is using the program. So we don't need calibration for hand tracking.

Similar to foot tracking, setting the hand as a child object of the controller will enable hand tracking.

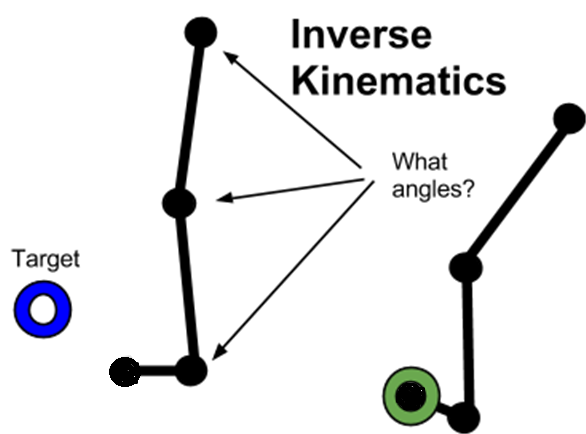
**Body (torso)**：



Like foot tracking, just set the torso as a child object of the tracker.

**IK**：

**What is IK?**



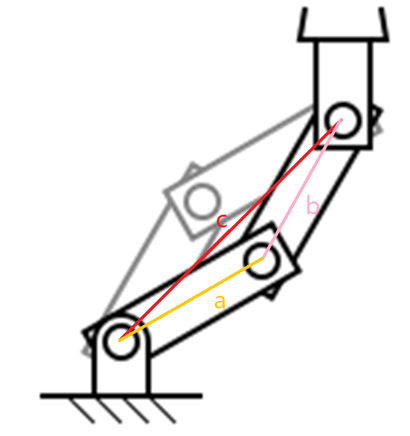
IK is short for Inverse Kinematics.

An IK problem is the calculation of the angles and positions of the joints in between the two end points which are known. For example, calculate the position of the eldow when the positions of the wrist and the shoulder are given. Most of the time an IK problem has more than one solution, which means using one of the many solutions as a prediction of the reality is probably inaccurate.

The reason we use IK despite the inaccuracy is that it can reduce the number of trackers we need. Fewer trackers means our motion capture system is easier to use. We don't have trackers bound to calves, or thighs, or arms, or forearms, thus we need IK to "guess" the positions of the knees and the elbows.

**Three-point IK**

A three-point IK is IK with three joints. The good thing about a three-point IK problem is that it has only one solution of the angle. This way it can improve the accuracy of the prediction. The reason is shown below:



Connect all the three joints and we'll get a triangle, where a,b are the bones thus fix-lengthed, and c is the connection between two known points and thus also fix-lengthed. Since all the three sides of the triangle are known, we know there's only one solution for the angles.

To exemplify with the human body, if the positions of the ankle and the hip (root of the thigh) are known (tracked), there's only one solution for the knee's angle.

So in order to use the three-point IK, aside from tracking hands and feet, **we also need a tracker on the waist**. This tracker will offer accurate tracking for the torso, and then help calculate positions of the knees and the elbows.

**Selecting the "bend plane"**

So, thanks to three-point IK, now we have the accurate **angle** of the knee. Does it mean we also know the **position** of the knee? The answer is no.



In the three pictures above, the model's ankles and hip stay at the same positions, and her knee is at 90 degrees in all three pictures. However we can see the knee is at different positions in three cases. Apparently, there are countless solutions even when the bending angle is fixed.

To solve this problem, we need a reasonably good "bend plane". Then we choose the solution that is on this plane.

Luckily we can take advantage of the features of human body. For instance, when you are standing, your thigh will most likely point at the same direction as your foot. You can try this by standing up and lifting one foot. Therefore choosing the plane that contains the foot as the "bend plane" will generates a desirable prediction.

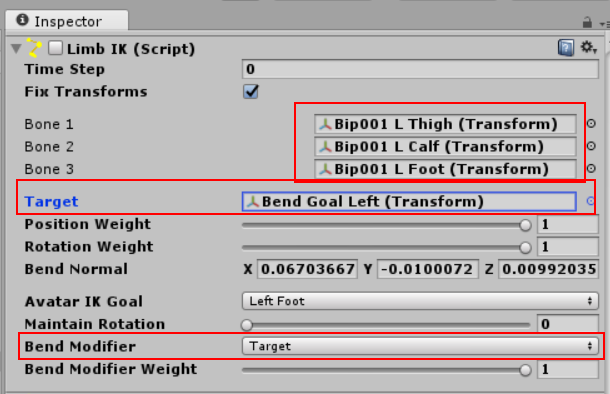
**\*Using Final IK**

Fields explained:

Bone1, Bone2, Bone3 stand for, respectively: root of the thigh, knee, ankle

Target: marker object that has the same pose as the foot

Bend Modifier = Target:  this means choosing the target (i.e. the foot) as the bend plane



**Calibration**：

The reason we need calibration is that players wear trackers in different ways and have different body shapes, which results into different offsets between the tracker and the body. I mentioned before that in the virtual world the foot needs to be a child object of the tracker. However, tracking only works when their offset (relative position and relative rotation) is equal or close to the offset between the tracker and the player's foot in reality.

In the sample program, the calibration is done by asking the player to go to the coordinate origin, to let his/her body overlap with the model's, and, most importantly, to step on her's feet. At this point, we can get the offset we want - the offset between the tracker and the model's foot in VR is equal to the offset between the tracker and player's foot in the real world.

Calibration for the torso is the same as the foot.

We don't need to calibrate the hands because people hold the controllers more or less the same way.

**How to attach Trackers to your body**

The are two important principles when attaching Trackers:

1. Stable
2. Easy to wear

If these two principles are satisfied, you can attach them in any way you want. Let's see an example:

Things we need: elastic hook & loop, 1/4'' hot shoe mount, an abitrary hard panel

Generic elastic hook & loop:



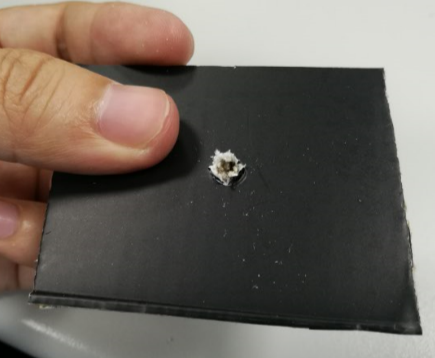
Or better yet, elastic hook & loop with a ring knot:



1/4'' hot shoe mount:



An arbitrary hard panel (hard paper or plastic):



Assembled:





Note that the function of the hard panels is to make the Tracker more stable. In my case, I get them by cutting the package box of the Tracker.

*Contact me: james\_xiong@htc.com*