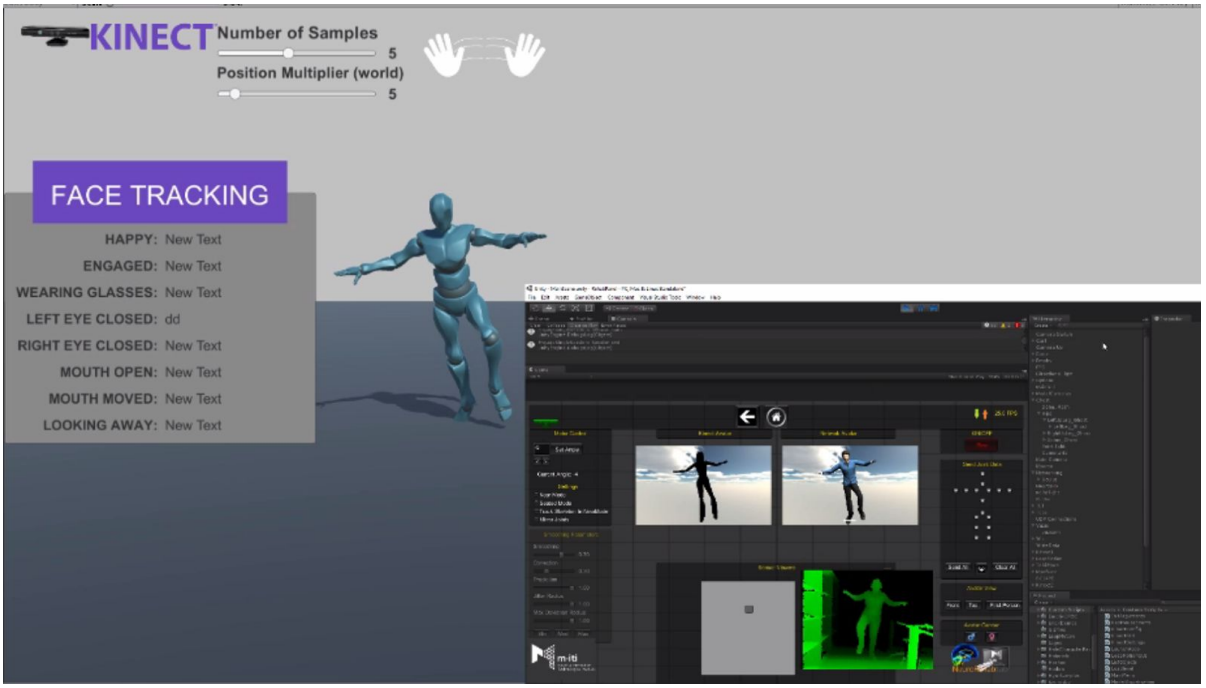
Kinect
Description

Any questions please contact mobile.neurorehablab@gmail.com

The kinect Demo package is a package to be used in Unity3D. This demo allows you to fully integrate your kinect with unity3D as long as you use the Reh@panel protocol to send its information. There is already a tool (Reh@panel) that sends the kinect information using this protocol.



User Manual

Importing the package

1. Create a new project
2. Import the package KinectDemo.unitypackage
    a. Assets -> Import package -> Custom Package

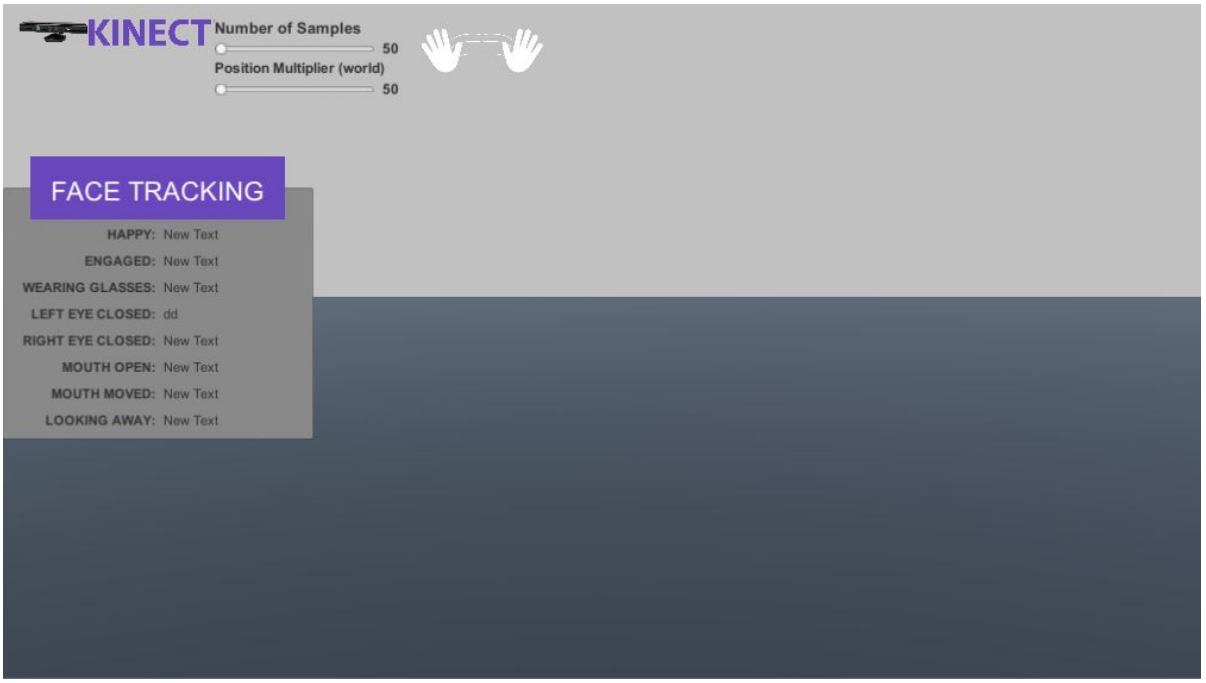
Requirements to use the package

1. Make sure you have the Kinect SDK installed - Kinect v1 or Kinect v2.
2. Plug in your Kinect to the PC
3. Make sure that it is sending information by UDP to port 1202 (To change this port see below).
    a. You can use the Reh@panel to do this

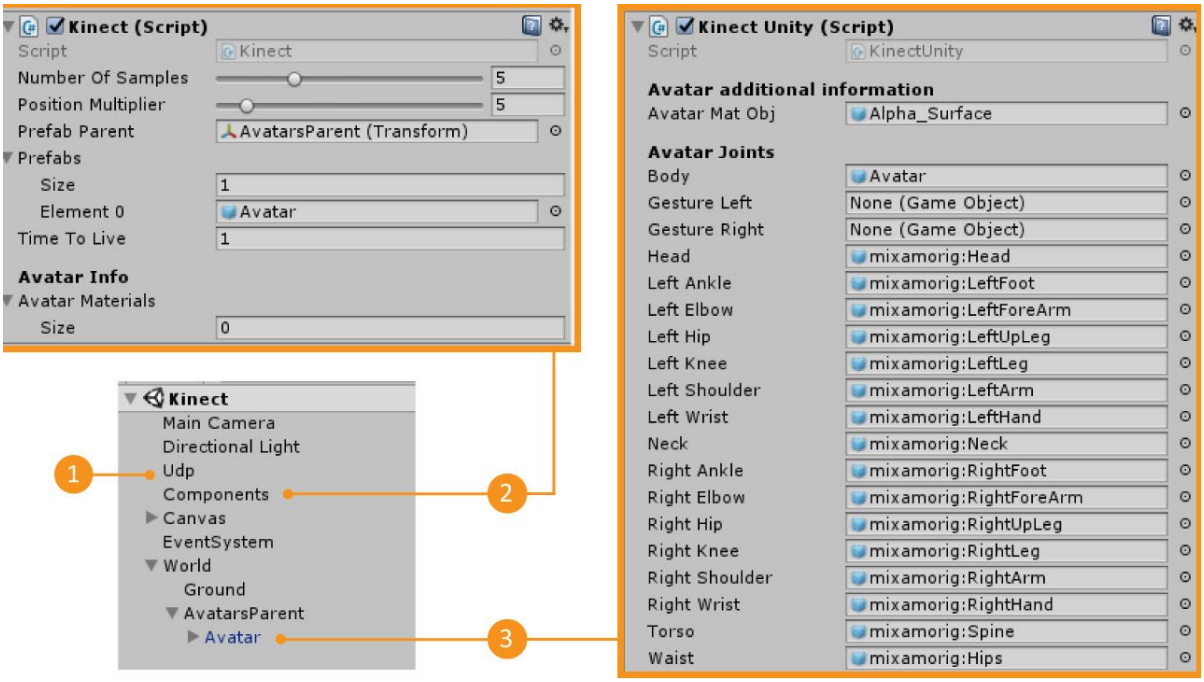
Testing the scene

1. Open the Scene
    a. Neurehab -> Demo Kinect -> Scenes -> KinectDemo

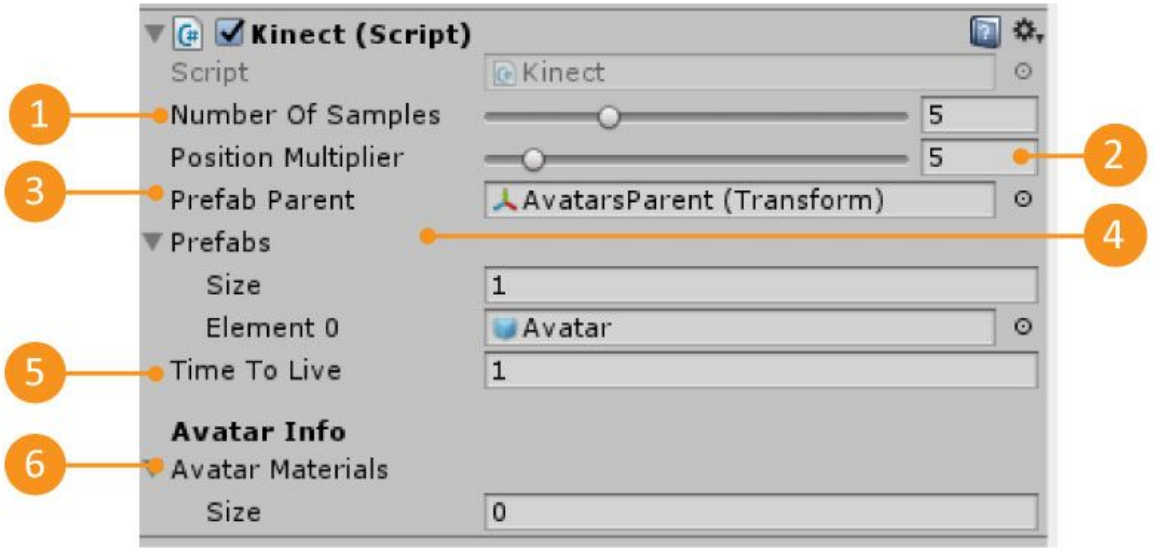
2. Make sure you fulfill all the requirements above
3. Press Play



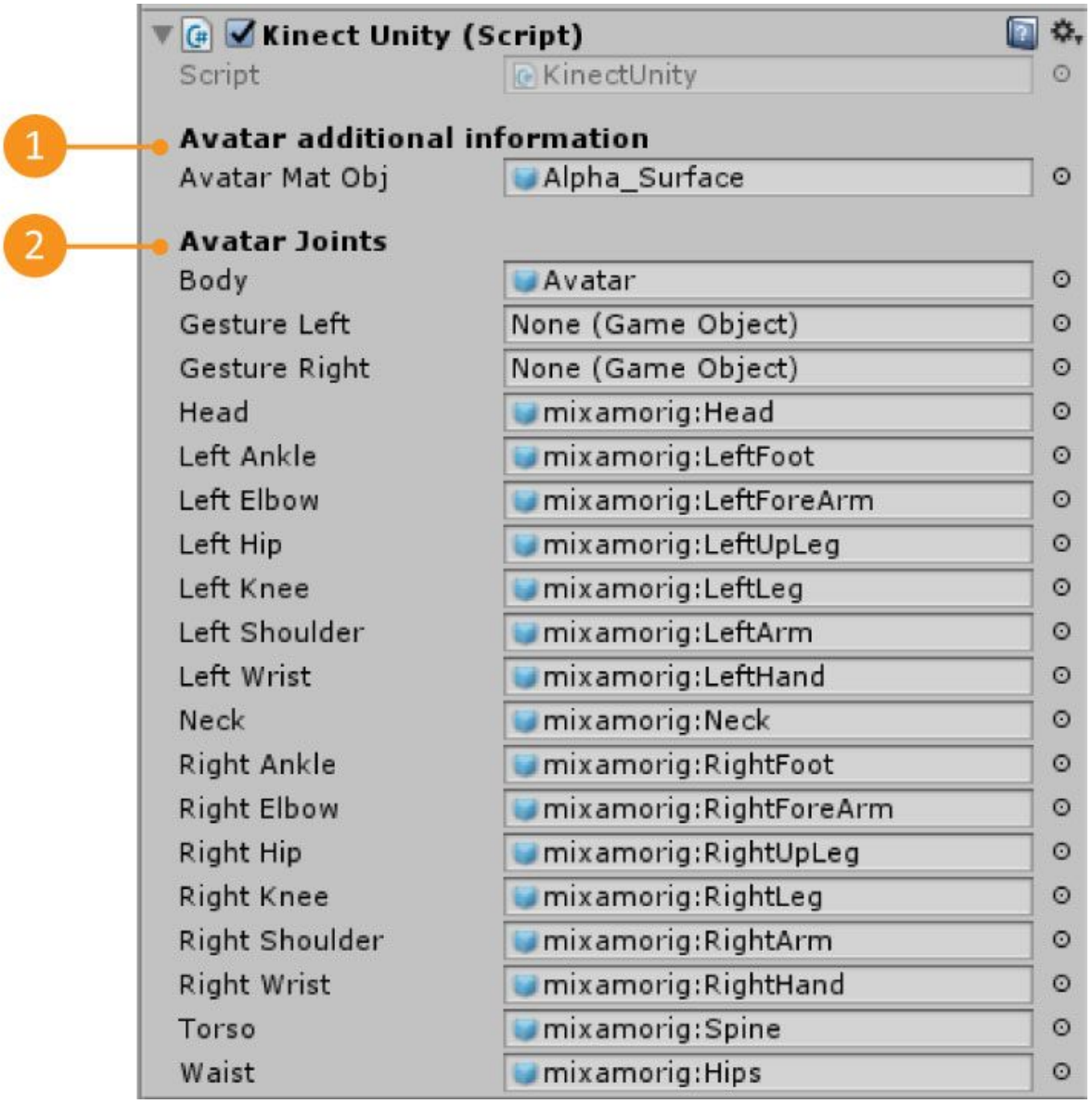
Understanding the Demo Scene



1 - UDP	Gameobject that has the UDP connection Information. You can change the UDP port here
2 - Components	Gameobject that has the Kinect.cs script
3 - Kinect Prefab	Prefab that has the KinectUnity.cs script. This prefab will only appear in the scene when you have pressed Play



1 - Number Of Samples	The number of data samples to save for each data input. In the end returns the average of that data input.
2 - Position Multiplier	Multiplies the position that is receiving for this value. This is useful when you have a game world scale that differs from the real world.
3 - Prefab parent	The Gameobject parent where the device prefabs are going to be instantiated Each instantiated prefab represents a Kinect that is sending data by UDP
4 - Prefabs	The list of prefabs that can be instantiated.
5 - Time to Live	The maximum time in seconds that a prefab can wait for new data before it is destroyed
6 - List of Materials	This is used if you want different materials for each avatar. Add a list of materials here and each avatar will pick a random material to add to its mesh (see below)



1 - Avatar Material	Which mesh is the material to be applied to. This is used if you want different materials for each avatar
2 - Device Data	All the kinect joints data. You will notice that kinect sends data from the joints and not from the bones themselves. For example, the Shoulder is the articulation but the bone that is referenced is the upper arm. If you want to add a new avatar, you should always take this into account.

How to access the device data information

The data is accessed through the KinectUnity.cs script. If you open this script, in the UpdateAvatar function you will notice that the Kinect prefab values are updated there.

```
 public void UpdateAvatar()
{
   if (Head != null)
        Head.transform.rotation = Body.transform.rotation *  //reference object
        GenericDeviceData.GetRotation(KinectBones.head.ToString()) * //rotation that comes from
kinect (world rotation)
        _initialRotations[(int) KinectBones.head]; //initial rotation of the head inside unity

...
}
```

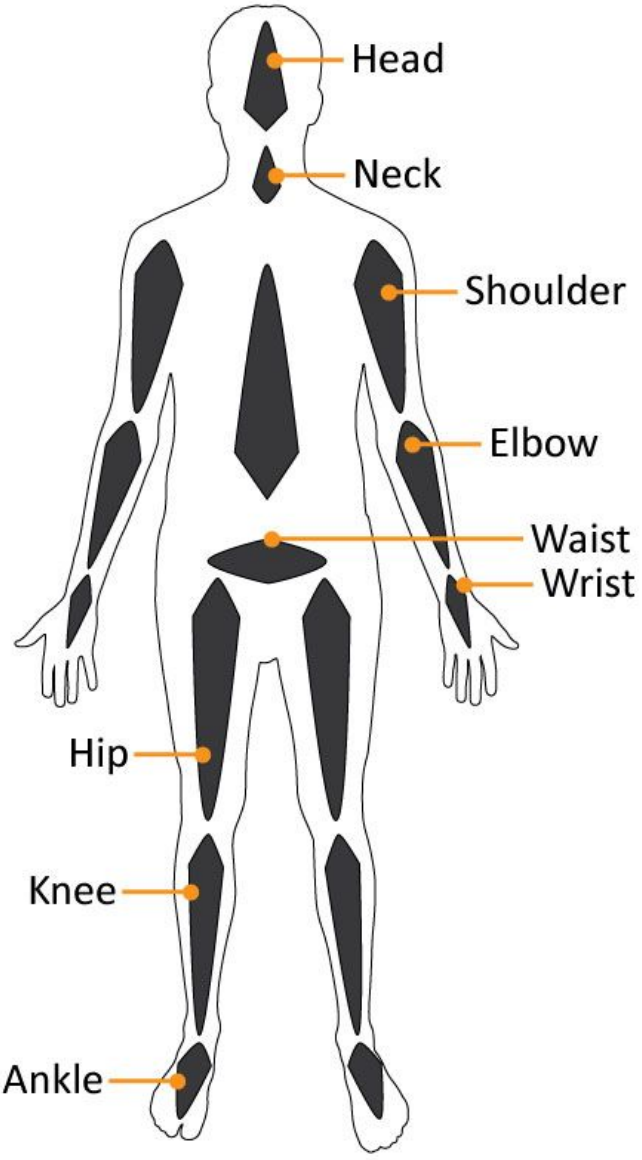
To be able to access any data that the Kinect is sending, you will need to know two things: the label of the information you want to access and the type of that information. Then, using the KinectUnity.cs script, you can find all this information by accessing the GenericDeviceData Dictionaries as shown below:

`GenericDeviceData.Get[INFORMATION_TYPE](“[INFORMATION_LABEL]”)`

For example, to access the Kinect leftshoulder rotation value which is labeled as ‘leftshoulder’ and is of the type rotation, you can:
`GenericDeviceData.GetRotation(“leftshoulder”)`

Current Kinect protocol values

Information Label	Information Type
body	position
head	position
	rotation
neck	position
	rotation
torso	position
	rotation
waist	position
	rotation
leftshoulder	position
	rotation
leftelbow	position
	rotation
leftwrist	position
	rotation
rightshoulder	position
	rotation
rightelbow	position
	rotation
rightwrist	position
	rotation
lefthip	position
	rotation
leftknee	position
	rotation



leftankle	position
	rotation
righthip	position
	rotation
rightknee	position
	rotation
rightankle	position
	rotation

Accessing the parameters

In the Rehapanel protocol, parameters are things that always present no matter if it is sending a position, rotation, value or bool. For more information on the Rehapanel read here.

To be able to access any parameters data that the Kinect is sending, you will need to know the parameter name. Then, using the KinectUnity.cs script, you can find all this information by accessing the GenericDeviceData.GetParameters("parameterNameHere") as shown below:

`GenericDeviceData.GetParameters("[PARAMETER_NAME]")`

For example, to access the Kinect closest body parameter which is labeled as "main", you can do:

```
var isClosestBody = GenericDeviceData.GetParameter("main");
```

Current Kinect parameters

Name	**Description**
Id	The GenericDeviceData ID.
Main	Tells if this GenericDeviceData is representing the closest body to the kinect Can be **true** or **false**