Sign Language Synthesis and Interaction lab (http://slsi.dfki.de/)

Kinect 2 Broadcaster

A stand-alone Kinect 2 data network broadcaster for Windows 8

Authors

- Fabrizio Nunnari <fabrizio.nunnari@dfki.de>
- David Tavárez <david@aholab.ehu.es>

Motivation

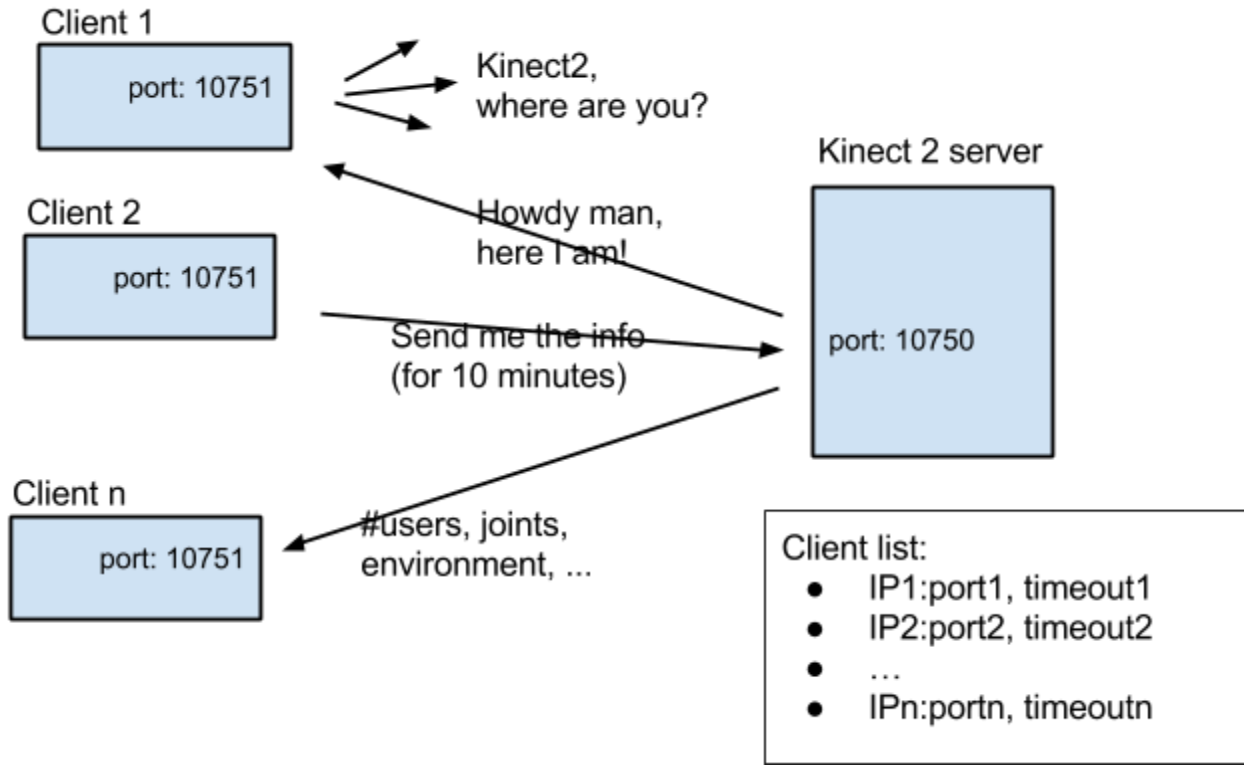
Kinect 2 is bound to Windows 8 and modern PCs with USB 3.0 cards and plenty of RAM. You cannot just take an old PC and plug the Kinect 2 on it. It means that you must invest resources to have a machine to dedicate to the Kinect 2.
Our intent is to create a service application able to broadcast the Kinect 2 information over a local network so that multiple machines can use the kinect 2 information from different machines, at the same time, on different platforms.
The target environment is a lab with a dedicated machine, connected to the kinect 2, with no keyboard, nor mouse, nor monitor. Just turn it on to access the Kinect 2 information across the network on different machines.
The information will be exchanged using the OSC protocol (http://opensoundcontrol.org/, http://en.wikipedia.org/wiki/Open_Sound_Control). A full description of the protocol is found in a later section.

Protocol design

The protocol has been designed for simplicity. In order to keep it simple, we have to accept some limitations. Currently, the main limitation of the protocol is that each client can receive information about only one character at a time.
Supporting the possibility to track whatever user is entering or leaving the scene would have made the protocol much more complex. And, because of the event-based nature of this information, the need to open a reliable TCP-based communication between client and server.

So, **we decided to move the user selection logic in the server side**. The client asks which user it is interested to, according to a predefined set of selection logics (closest to the center, firat at left from the center, first at right from the center). More selection logics can be added in future protocol versions.

Architecture



The idea is that multiple clients can request a centralized server to send them information about the kinect.
A discovery mechanism allows computers on the same local network to broadcast a message looking for a server. The server answers so that the client can store information about the IP address of the server.
Afterwards, a client can request the server to send him information about the Kinect 2. The client IP:port is stored in a list and associated to a 10 minutes timeout. During this time, information are sent to the client. If a client wants to renew the service, another request must be sent. The timer will be reset on the server side. If the timeout expires, the client is deleted from the list.

Requirements

Hardware (server)

- A windows 8 machine with 32 bit (x86) or 64 bit (x64) processor, dual-core 2.66-GHz or faster processor, 2 GB RAM, DirectX 11 compatible graphics card and USB 3 port to run the client (see hardware requirements for the kinect v2 sensor: http://www.microsoft.com/en-us/kinectforwindows/purchase/sensor_setup.aspx).

Software (client)

- An OSC implementation **supporting bundles**.

Installation

An installer for the server executable has been created to make sure that the configuration is properly performed. Once you have downloaded the installer please follow the next steps:

- Double-click on **setup.exe** to run the installer.
- In the welcome screen click Next.
- Select the installation folder and click Next to continue.
- In the Confirm screen click Next to start the installation.

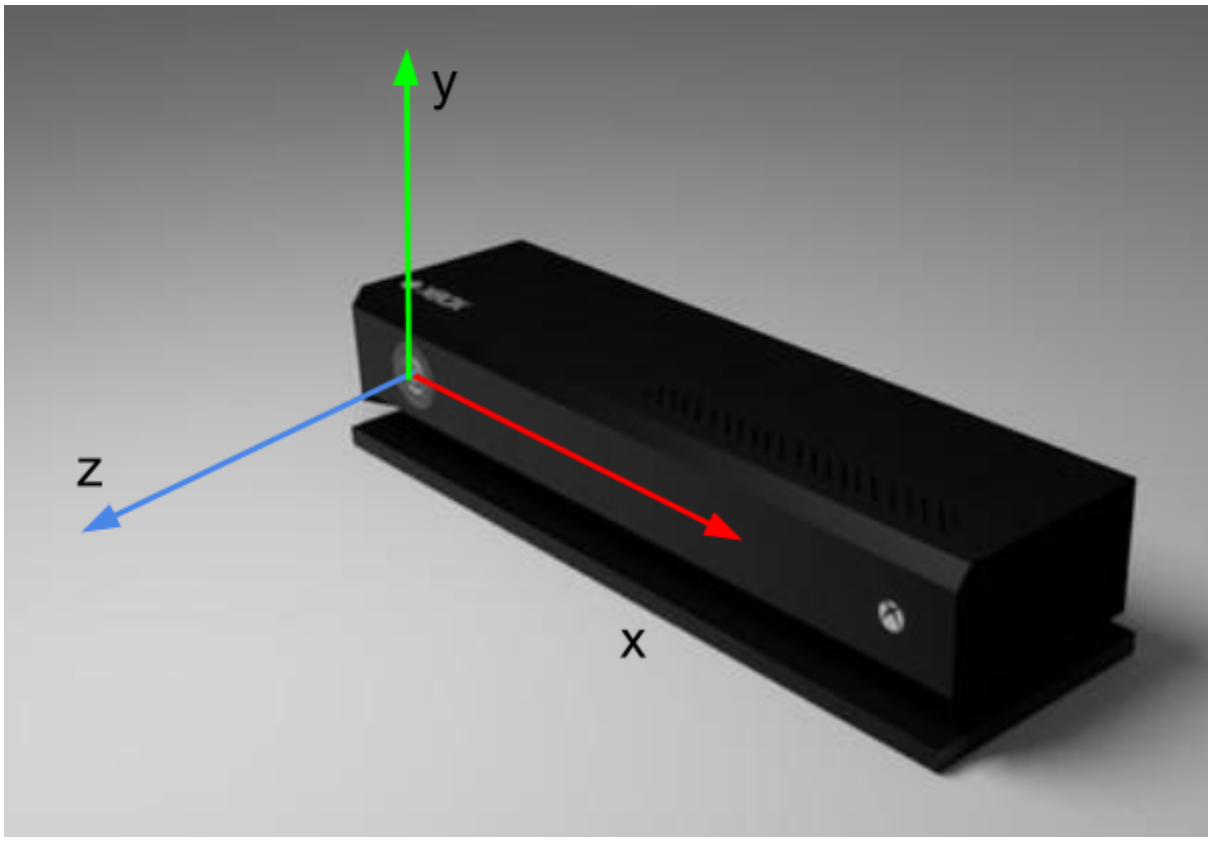
Usage

No special actions are required. The Kinect2Broadcaster will open a window showing a preview of the recognized skeletons. No preferences or menus are present.
Details of the communication protocol are in a later section.

Development Environment (Compile it yourself)

The Kinect2Broadcaster has been currently implemented as a modification of a basic example distributed with the Kinect 2 SDK. For this reason no source code is available, yet.

Coordinate system



OSC protocol

For easier message routing, all messages will start with the “/kinect2” prefix name space.
The server listen on port 10750. It is composed by the ascii decimal codes of letters *k* (107) and *2* (50). All the clients listen for answers and information messages on port 10751.

Discovery messages

When a client needs to look for a server, it can broadcast a message to the broadcast address of the local network, to port 10750:

→ `/kinect2/where_are_you`

The server will store client IP and answers to port 10751 with:

← `/kinect2/here_I_am`

Control messages

The client can request to be inserted in the list of clients receiving the information

→ `/kinect2/send_me_info s`

The server will start sending info to the client that requested it for 10 minutes.
The client will use the same message to renew the request, before the 10 minutes are over, in order to keep on receiving messages.
The server will send information about only one subject.
The first parameter specifies to which subject the client is interested in. Possible choices are:

- “CLOSEST_TO_CENTER”: the subject whose center x coordinates is closest to x=0
- “FIRST_LEFT_FROM_CENTER”: the first subject whose center x coordinates are > 0 (left when looking at the Kinect)
- “FIRST_RIGHT_FROM_CENTER”: the first subject whose center x coordinates are < 0 (right when looking at the Kinect)

In the last two modes, if there are not users at the left/right of the center, no data is sent. The philosophy is: “better not data than wrong data”.

Information messages

The server will send at high ratio (30 messages per second) the information to all the clients registered in the list.

← `/kinect2/joint ifffffffi`

Parameters are, in order:

- (int) joint_id
- (float) translation_x
- (float) translation_y
- (float) translation_z
- (float) rotation_quaternion_w
- (float) rotation_quaternion_x
- (float) rotation_quaternion_y
- (float) rotation_quaternion_z
- (int) confidence, 0 tracked, 1 inferred, 2 untracked

The bone name associated to the id can be retrieved from the code extract reported in Appendix.

← /kinect2/pose s
The server will send this message when a specific user pose is detected. The different choices for user poses are application domain dependent and will depend on the server implementation.

Appendix

JointType enumeration. This is an extract from the Kinect2 SDK. This enumeration can be used to infer the joint name from its integer ID.

```
enum _JointType
      {
      JointType_SpineBase    = 0,
      JointType_SpineMid    = 1,
      JointType_Neck    = 2,
      JointType_Head    = 3,
      JointType_ShoulderLeft    = 4,
      JointType_ElbowLeft    = 5,
      JointType_WristLeft    = 6,
      JointType_HandLeft    = 7,
      JointType_ShoulderRight    = 8,
      JointType_ElbowRight    = 9,
      JointType_WristRight    = 10,
      JointType_HandRight    = 11,
      JointType_HipLeft    = 12,
      JointType_KneeLeft    = 13,
      JointType_AnkleLeft    = 14,
      JointType_FootLeft    = 15,
      JointType_HipRight    = 16,
      JointType_KneeRight    = 17,
      JointType_AnkleRight    = 18,
      JointType_FootRight    = 19,
      JointType_SpineShoulder    = 20,
      JointType_HandTipLeft    = 21,
      JointType_ThumbLeft    = 22,
      JointType_HandTipRight    = 23,
      JointType_ThumbRight    = 24,
      JointType_Count    = ( JointType_ThumbRight + 1 )
} ;
```