Open Spatial Service Instructions (v0.0.2)

Welcome to the Open Spatial Windows Service. Here are some basic instructions to get you started.

The package you received will contain 5 items

1. NodConfig.exe - The service configuration utility.
2. OpenSpatialDLL.dll - The client DLL
3. OpenSpatialDLL.lib - The library for the client DLL
4. OpenSpatialService.exe - The service
5. OpenSpatialServiceController.h - The header for the DLL
6. OpenSpatialExampleApplication - a Visual Studio 2013 example project, showcasing how to integrate with the Nod DLL

Note: most of these operations must be done as administrator. If you encounter errors please check you are running everything as an administrator before contacting us.

Part 1: Installing the service

1. Open an Administrator Command Prompt
2. Navigate to the folder containing OpenSpatialService.exe

   Note: I have seen windows have issues installing services from paths that have a “ “ (space) in the name. if you have issues installing please move OpenSpatialService.exe into your C:/ root folder.

3. Type: OpenSpatialService.exe -install
4. It should print a confirmation message saying the service is installed.

Part 2: Starting the Service and Using the NodConfig application

1. Connect all your rings to your PC via Bluetooth

   Note: If you connect new rings please close and reopen the NodConfig application to use them.

2. Open NodConfig.exe (Run as Administrator)
3. This will start the OpenSpatialService
4. Click Refresh to list your connected rings
5. If you select a ring and click settings you can change some settings of the ring

   Note: Updating firmware is not supported on windows yet. The UI elements for updating firmware will not do anything

<u>Part 3: Writing your own application with the DLL (using visual studio)</u>
<u>Note: currently only C++ is supported.</u>

1. Create a new visual studio project
2. Copy OpenSpatialDLL.dll, OpenSpatialDLL.lib and OpenSpatialController.h into your source folder
3. Add OpenSpatialController.h to your solution.

Now you are ready to start using the code. (Please look at the source of the provided visual studio example project if there is any confusion).

There are two C++ classes which are relevant to integrating with Nod.
1. OpenSpatialServiceController
2. OpenSpatialDelegate

<u>Step One: Instantiate OpenSpatial Controller</u>

OpenSpatialServiceController is the class which interacts with the OpenSpatialService to control your Nod and get data.

To use this class first create an instance of it. This will automatically setup the connection to the service.

The service works through a subscription / event model. There are 4 types of events
1. Pointer events - 2D X,Y data
2. Gesture Events - Swipe Up, Swipe Down etc. These gestures should be explained in your Nod Manual
3. Button Events - Touches, Tactiles etc, please refer to your Nod Manual for an explanation of the buttons
4. Pose6D Events - X, Y, Z, Yaw, Pitch, Roll - 3D rotational and translational data. (Currently only rotation euler angles are supported X, Y and Z will always be 0)

To subscribe to these events call the corresponding functions in the OpenSpatialControllerClass:

```
void subscribeToPointer(std::string ringName);
void subscribeToGesture(std::string ringName);
void subscribeToPose6D(std::string ringName);
void subscribeToButon(std::string ringName);
```

These functions take in a ring name string. There is a vector defined as public member of the OpenSpatialController class:

```
std::vector<std::string> names;
```

This vector will contain all the names of your connected rings, so for example to subscribe to Pointer events of the first ring:

```
openSpatialController->subscribeToPointer(openSpatialController->names.at(0));
```

<u>Step Two: Create a Delegate Class</u>

The OpenSpatialDelegate interface defined in OpenSpatialServiceController.h is the class which receives the data from the service.

Create a class which implements this interface. For an example please refer to the ExampleDelegate Class in the example app.

The interface has 4 methods which are called when data is sent from Nod. These functions are in the format:

```
virtual void pointerEventFired(PointerEvent event);
```

Each function will have one event argument which contains the event data. These arguments are structs defined in OpenSpatialServiceController.h

For example the PointerEvent struct contains 3 members. X, the x value of the pointer event. Y, the y value of the pointer event. id, the id of the ring sending the event. This id is an integer starting at 0. It corresponds to the ring at that index in the names vector mentioned earlier.

In these functions implement your handling of the data you need. Notice the example app just prints out the data in a readable format.

<u>Step Three: ASSIGN YOUR DELEGATE!</u>

call the setDelegate function of OpenSpatialServiceController to assign the delegate. Without this you will not receive any data.

e.x. *yourOpenSpatialController* -> setDelegate(*yourOpenSpatialDelegate);

<u>Step Four: Enjoy your data</u>

Data should now be flowing into your application once you subscribe to the appropriate events.

Note: Pointer and Gesture events will only occur in Pointer Mode. Button and Pose6D events will only occur in 3D Mode. Pose6D events will only occur after sliding once on the ring’s slider. Please refer to your Nod Manual to know how to slide.

Thank You for using Nod!

For any questions regarding this windows service or Nod in general please email: support@nod-labs.com