pst

Software Development Kit (SDK)

License Agreement

The products of PS-Tech B.V. come with a software license agreement. This END USER LICENSE AGREEMENT (EULA) is shipped with each product.

In no event shall PS-Tech be held liable for any incidental, indirect, or consequential damages whatsoever (including, without limitation, damages for loss of business profits, business interruption, loss of business information, or any other pecuniary loss) arising out of the use of or inability to use the software or hardware.

Patent Liability

No patent liability is assumed with respect to the use of the products of PS-Tech.

Errors in Manual

While every precaution has been taken in the preparation of this manual, PS-Tech assumes no responsibility for errors or omissions.

Copyright Information

PS-Tech B.V.
Falckstraat 53 hs, NL 1017VV Amsterdam
The Netherlands
Call: 858 764 4411 (USA)
Call: +31 (0)20 331 1214 (Global markets)
Fax : +31 (0)20 524 8797

info@ps-tech.com

http://www.ps-tech.com

1 Software Development Kit

The PST Software Development Kit (SDK) provides an interface between the PST tracking system and your own software applications.

Usage

To use the PST SDK in your own software, include the header file pstapi.h in your project. The PST SDK library is dynamically linked with your program. Please install the file pst.dll in your system directory or program directory and link your project with pstdll.lib.

Note that the PST SDK communicates with the PST client software that is included with your PST installation. If this application is not running you will not receive tracker events in your application, even if the tracker unit itself is running.

The PST SDK contains two data types to describe tracker data events: PSTSensor and PSTPoint.

PSTSensor

Description

Sensor events are generated when an interaction device is visible and has been identified by the PST.

Member Documentation

name	*char[80]*	Name of the interaction device as listed in the PST client software.
id	*int*	Identifier of the interaction device as listed in the PST client software.
pose	*float[16]*	Row-major 4x4 transformation matrix describing the pose of the device in the coordinate system as defined by the PST client software (see the PST manual). The pose is defined as: $\begin{pmatrix} p_0 & p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 & p_7 \\ p_8 & p_9 & p_{10} & p_{11} \\ p_{12} & p_{13} & p_{14} & p_{15} \end{pmatrix} = \begin{pmatrix} U_x & V_x & W_x & T_x \\ U_y & V_y & W_y & T_y \\ U_z & V_z & W_z & T_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$ where p_i represents the elements from the pose, the vectors *U, V, W* represent the 3x3 rotation matrix, and *T* represents the translation vector in meters.
timestamp	*double*	Timestamp of the moment the data was recorded since the start of the PST client application (in seconds).

PSTPoint

Description

Point events are generated for single visible 3D points that have not been identified as part of an interaction device.

Member Documentation

id	*int*	Identifier of the 3D point. As a single 3D has no features to distinguish it from another, points are given an identifier based on their previous motion. Note that there is no guarantee that the identifier is consistent or correct between sensor updates.
pos	*float[3]*	The 3D position of the point in meters.
timestamp	*double*	Timestamp of the moment the data was recorded since the start of the PST client application (in seconds).

Header pstapi.h

Description

Represents the interface to the PST client software.

Member Function Documentation

int pst_connect();
Connect to the PST.

Return value	*int*	one on success, zero on failure

int pst_disconnect();
Disconnect from the PST.

Return value	*int*	one on success, zero on failure

int pst_sensor_changed();
Check if any PST sensor has been updated since the last time it was read by the SDK.

Return value	*int*	one if new data is available, zero if no new data is available

int pst_sensor_changed_by_id(int id);
Check if the PST sensor indicated by *id* has been updated since the last time it was read by the SDK.

Parameters	*id*	The identifier of the device (0-99)
Return value	*int*	one if new data is available, zero if no new data is available

int pst_get_sensor(struct PSTSensor* sensor);
Get the last PST sensor event if a new event is available.

Parameters	*sensor*	A pointer to an allocated PSTSensor struct to receive a new event
Return value	*int*	one if a new event is returned, zero if no new data is available

int pst_get_sensor_by_id(int id, struct PSTSensor* sensor);
Get the last PST sensor event with the given *id* if a new event is available.

Parameters	*id*	The identifier of the device (0-99)
	sensor	A pointer to an allocated PSTSensor struct to receive a new event
Return value	*int*	one if a new event is returned, zero if no new data is available

int pst_point_changed();
Check if any PST point has been updated since the last time it was read by the SDK.

Return value	*int*	one if a new point is available, zero if no new point is available

int pst_get_point(struct PSTPoint* point);
Get the last PST point event if a new event is available.

Parameters	*point*	A pointer to an allocated PSTPoint struct to receive a new event
Return value	*int*	one if a new event is returned, zero if no new data is available

Example

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include "pstapi.h"

int main (int ac, char **av)
{
    int i,j;
    struct PSTSensor sensor;

    // connect to the PST
    if (!pst_connect())
        exit (1);

    // infinite loop...
    while (1)
    {
        // loop over all new sensor events
        while (pst_get_sensor(&sensor))
        {
            // print out the name and id
            printf("Device: \"%s\", id: %d\n", sensor.name,
                    sensor.id);

            // print the rotation matrix
            printf("  Orientation:\n");
            for (i=0; i<3; ++i)
            {
                printf("  ");
                for (j=0; j<3; ++j)
                    printf("%.2f ", sensor.pose[i*4+j]);

                printf("\n");
            }

            // print the translation vector
            printf("\n  Translation:\n  ");
            for (i=0; i<3; ++i)
                printf("%.2f ", sensor.pose[i*4+3]);

            printf("\n\n");
```

```
        }
    }

    // disconnect from the PST
    pst_disconnect();

    return 0;
}
```