Multi-Touch- und Multi-User-Interfaces

Björn Frömmer

Primer

Goals of the third practical:

- Integration of the TUIO framework into your own project
- Normalization of the blob coordinates
- Transmit the tracked blobs (from practical 2) via network protocol utilizing TUIO-events
- Verification of sent event using TUIO testclients (TuioDump & TuioDemo)

Overview

- TUIO
- OSC
- Schematic structure
- Contents of the protocol
- TUIO API
- Test clients
- Normalization
- Some TUIO API components in detail

TUIO integration

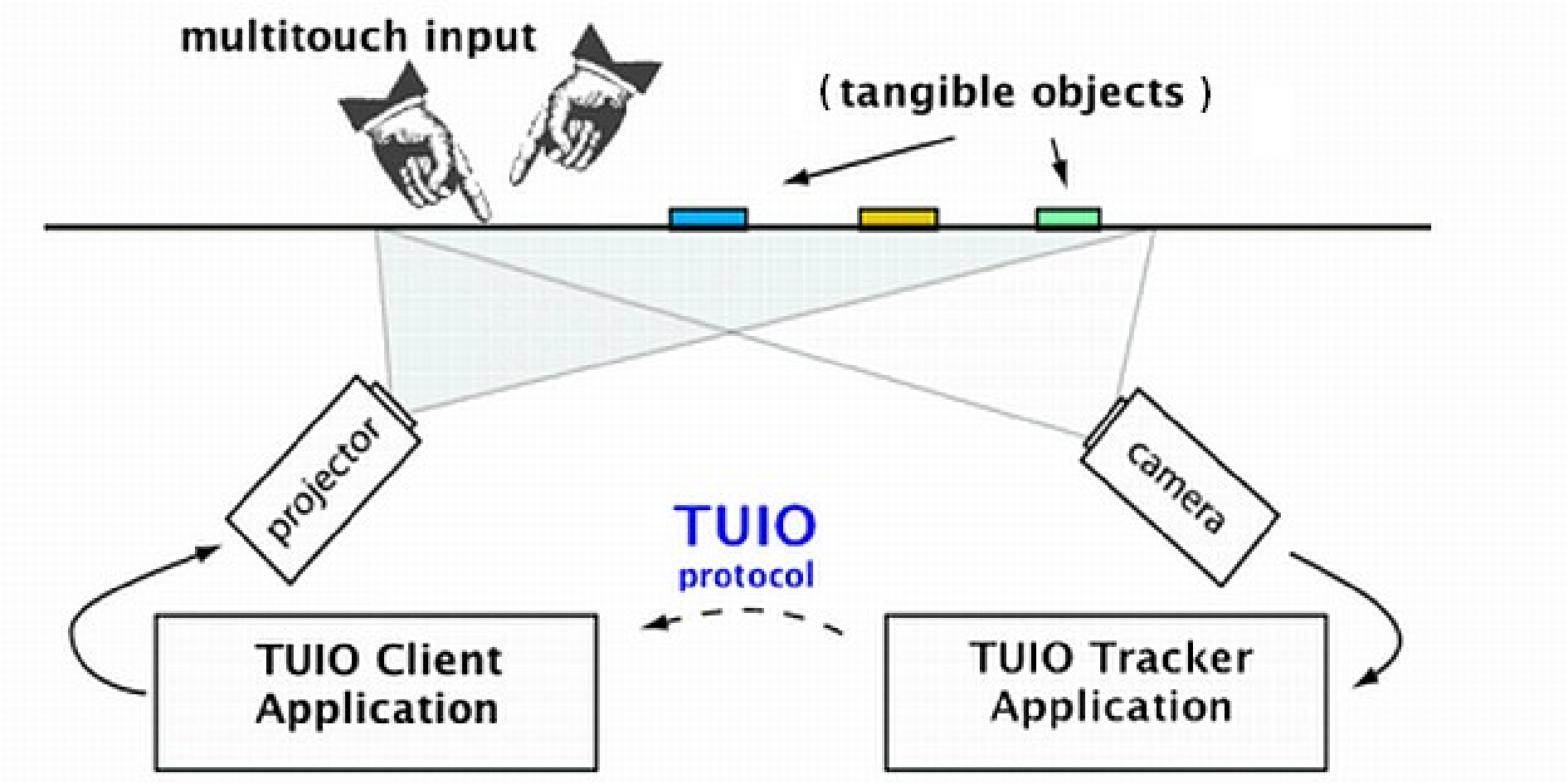
- What does the TUIO framework do?
- Why TUIO?
- Why a distributed system (via network traffic)?
- TUIO framework is based on the “Open Sound Control” protocol (OSC)

OSC

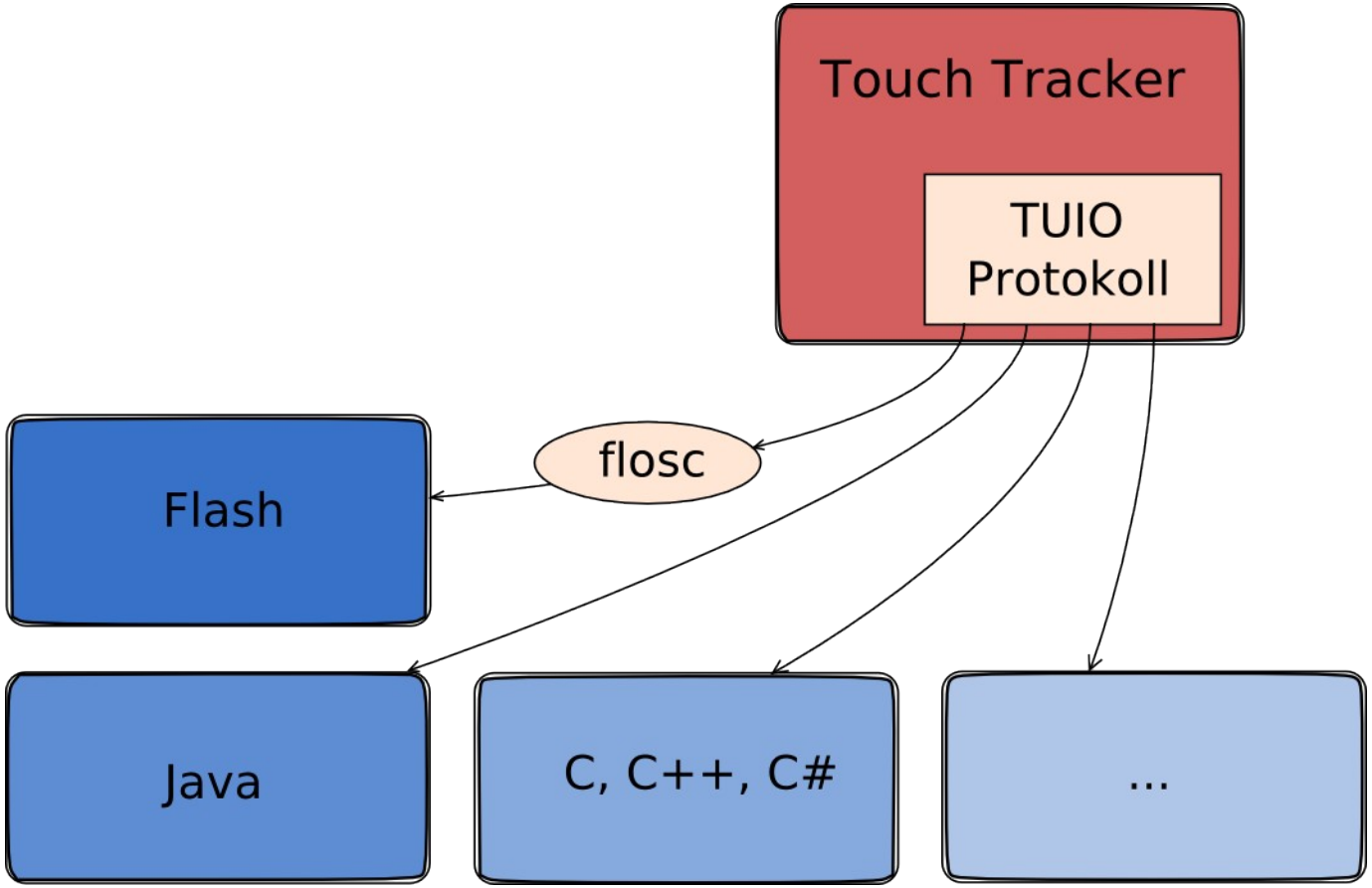
- **O**pen **S**ound **C**ontrol Protocol
  - Open-ended, dynamic, URL-style symbolic naming scheme
  - Symbolic and high-resolution numeric argument data
  - Pattern matching language to specify multiple recipients of a single message
  - High resolution time tags

This simple yet powerful protocol provides everything needed for real-time control of sound and other media processing while remaining flexible and easy to implement.

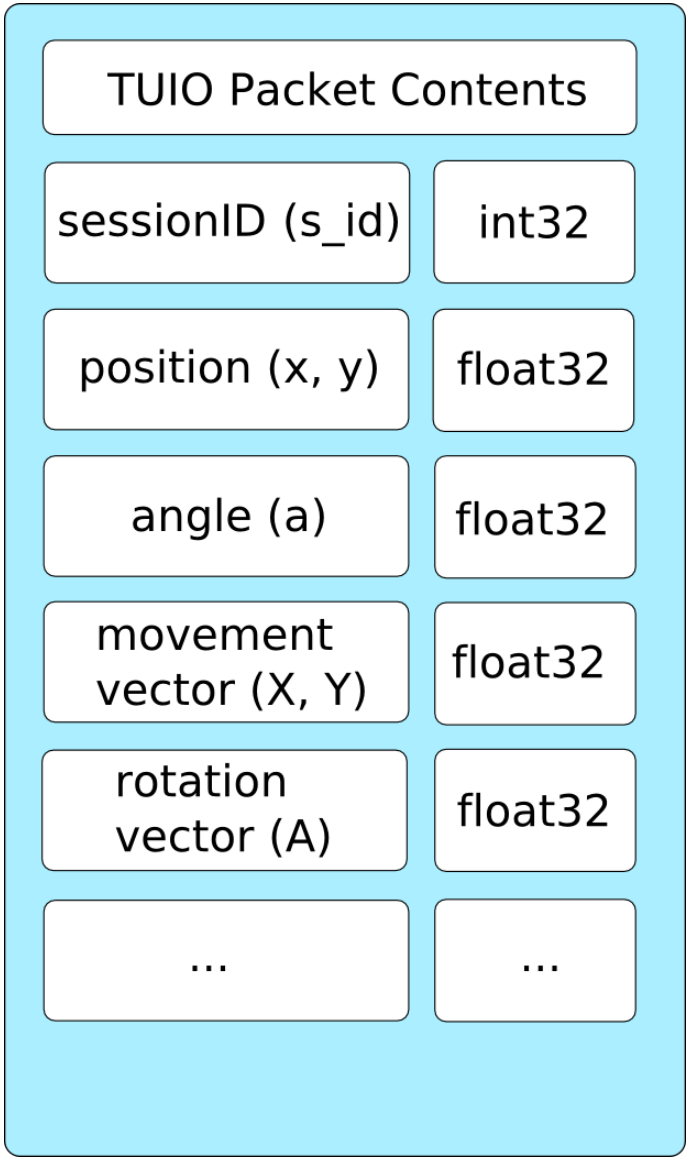
MT Table → TUIO → Application



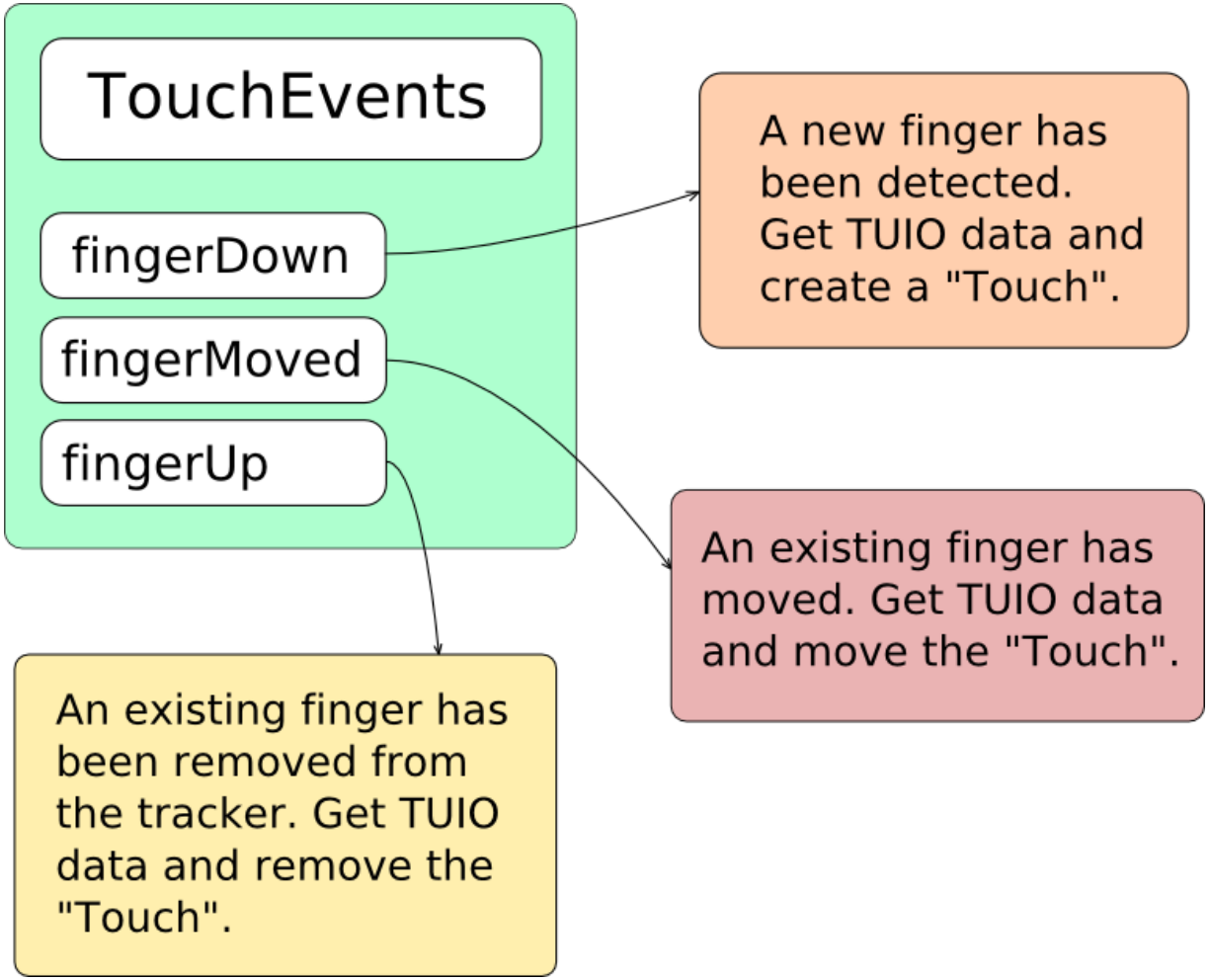
TUIO Communication



TUIO Packet Contents



TUIO Events


TouchEvents
fingerDown
fingerMoved
fingerUp
A new finger has been detected. Get TUIO data and create a "Touch".
An existing finger has moved. Get TUIO data and move the "Touch".
An existing finger has been removed from the tracker. Get TUIO data and remove the "Touch".

TUIO API

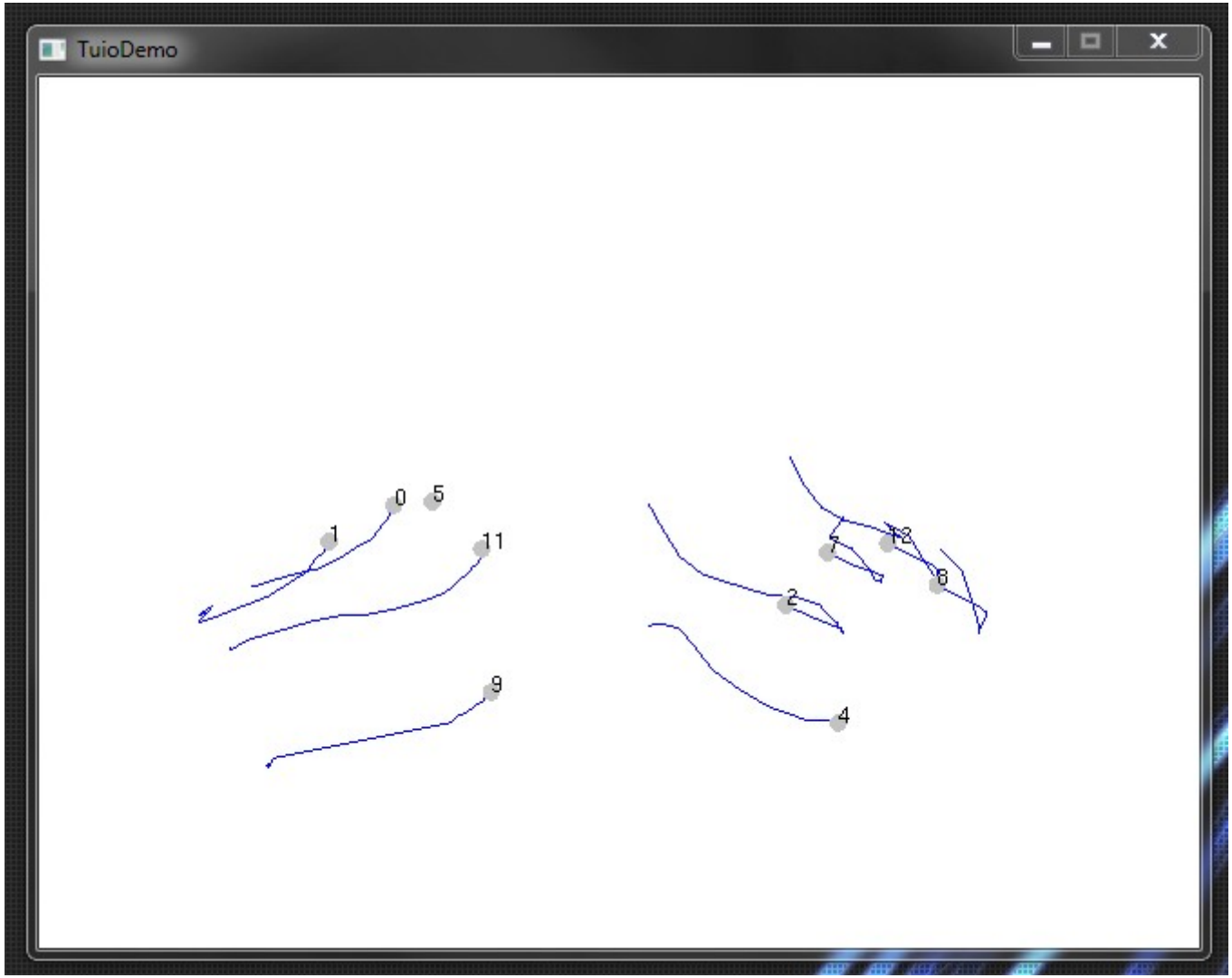
- API
  - TUIO::TuioClient
  - TUIO::TuioContainer
  - TUIO::TuioCursor
  - TUIO::TuioListener
  - TUIO::TuioObject
  - TUIO::TuioPoint
  - TUIO::TuioServer
  - TUIO::TuioTime

TuioDump

C:\Users\bjoern\Documents\Visual Studio 2008\Projects\finger_detection_tuio\TUIO_CPP\Debug\T...

```
listening to TUIO messages on UDP port 3333
add cur 0 (1) 0.909574 0.604167
add cur 1 (2) 0.805851 0.554167
add cur 2 (3) 0.882979 0.5375
add cur 3 (4) 0.851064 0.516667
set cur 2 (3) 0.881649 0.539583 0.0525857 1.11885
set cur 3 (4) 0.849734 0.516667 0.0282932 0.601983
set cur 0 (1) 0.908245 0.60625 0.0797268 2.57183
set cur 2 (3) 0.882979 0.539583 0.0428961 -0.312568
set cur 3 (4) 0.849734 0.516667 0 -0.912683
set cur 0 (1) 0.909574 0.60625 0.0415556 -1.19285
set cur 2 (3) 0.882979 0.539583 0 -1.3405
set cur 3 (4) 0.848404 0.516667 0.0415575 1.29867
set cur 0 (1) 0.908245 0.60625 0.000375008 -0.0116133
set cur 1 (2) 0.805851 0.560417 0.00176254 0.00049705
set cur 2 (3) 0.882979 0.539583 0 0
set cur 3 (4) 0.844415 0.51875 0.0012692 -0.0113616
set cur 0 (1) 0.904255 0.61875 0.00976276 0.00698493
set cur 1 (2) 0.804521 0.579167 0.0139859 0.0090948
set cur 2 (3) 0.880319 0.55625 0.0125577 0.00934351
set cur 3 (4) 0.837766 0.539583 0.0162713 0.0111623
```

TuioDemo



Normalization

- Applications can have different window sizes, therefore it's important to normalize the touch events
- Normalization means that coordinates are all between 0 and 1
- These coordinates can be re-calculated into screen coordinates on the client side
- All TUIO compatible multitouch demos (at the h_da) expect normalized touch coordinates
  - Example: The (graphical) TuioDemo does not work with native screen coordinates!

TUIO::TuioServer

```
TuioServer *server = new TuioServer();

TuioObject *tobj = server->addTuioObject(xpos, ypos, angle);
TuioCursor *tcur = server->addTuioCursor(xpos, ypos);

server->updateTuioObject(tobj, xpos, ypos, angle);
server->updateTuioCursor(tcur, xpos, ypos);

server->removeTuioObject(tobj);
server->removeTuioCursor(tcur);
```

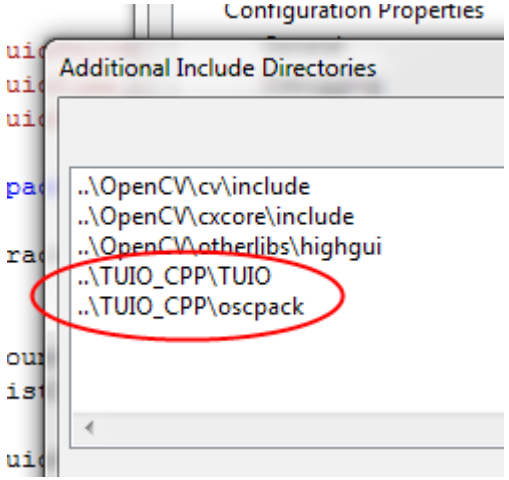
External TUIO Cursor

```
TUIO::TuioCursor *tcur = new TUIO::TuioCursor(session_id, cursor_id,
xpos, ypos);

server->addExternalTuioCursor(tcur);


// This way you can generate / pass a touch-ID to the server
    (otherwise (see previous slide) the server will assign an ID by
    itself)
```

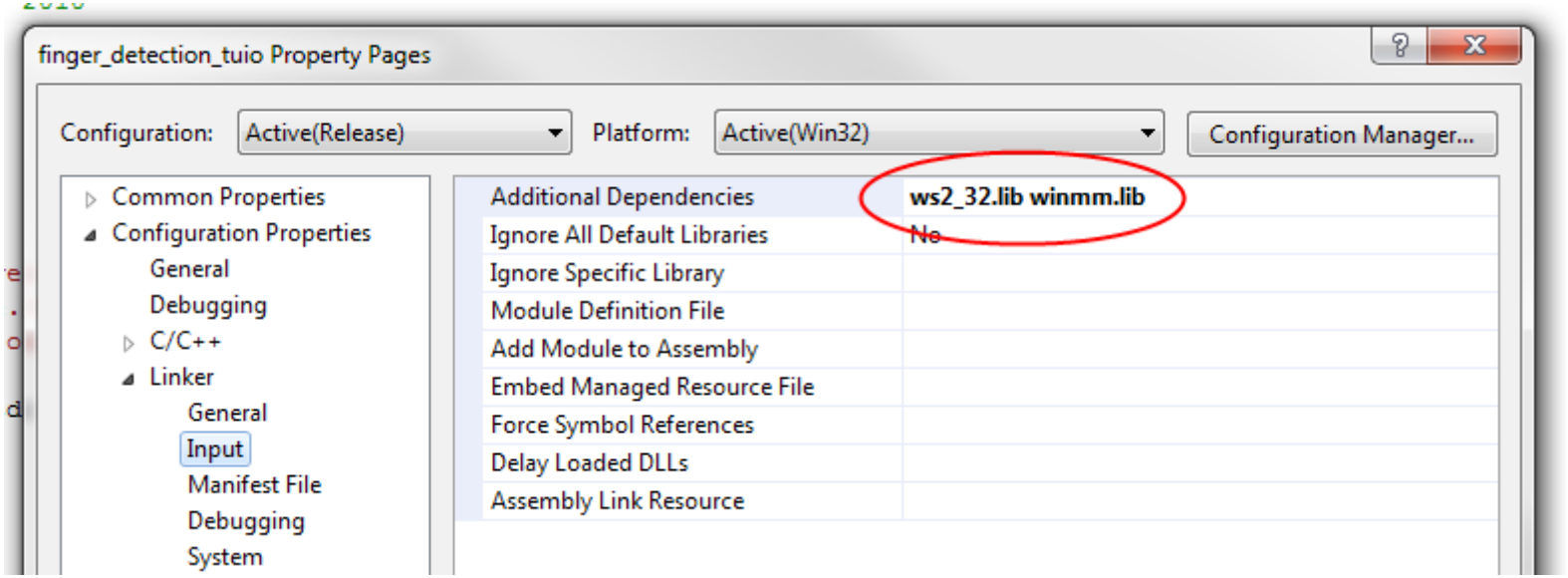
C++ Project Settings

- Needed TUIO header source files

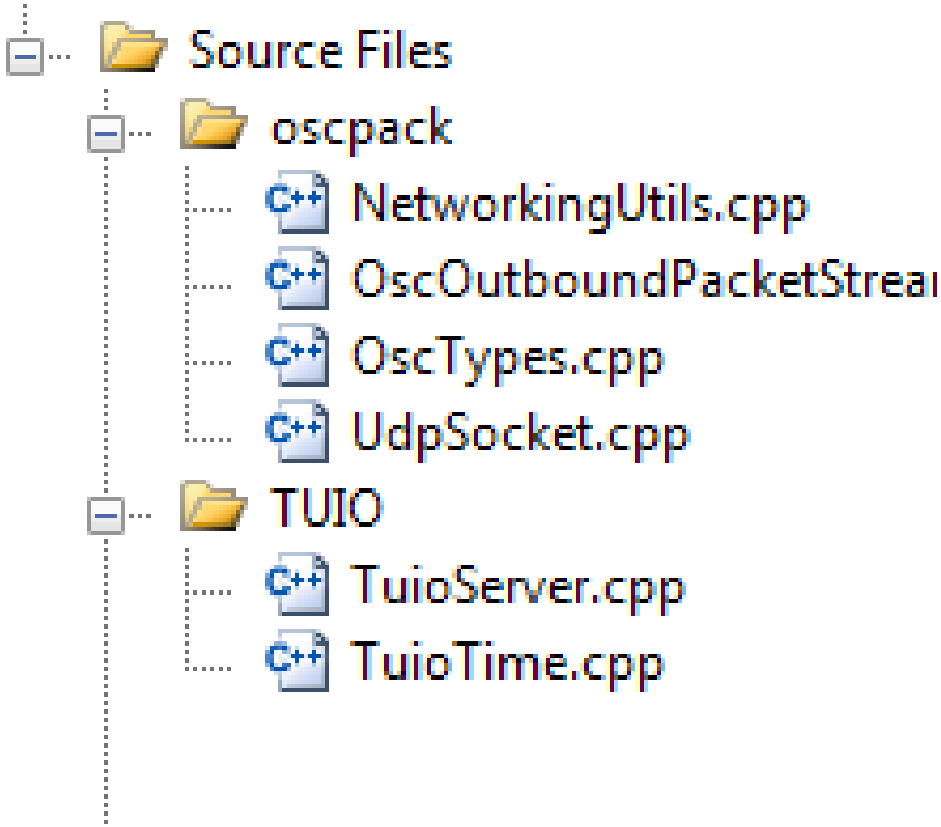
```
#include "TuioServer.h"
#include "TuioTime.h"
#include "TuioCursor.h"
```



- Necessary project settings



TUIO files


Source Files
oscpack
NetworkingUtils.cpp
OscOutboundPacketStrea
OscTypes.cpp
UdpSocket.cpp
TUIO
TuioServer.cpp
TuioTime.cpp

Tips

- Use an appropriate data structure to locally store the TUIO data (list, map, vector, …)

```
// tuio list - example
list<TUIO::TuioCursor*> tuioBlobs;
list<TUIO::TuioCursor*>::iterator tuioBlobsIter;
```

- Transmit the TUIO events every frame
  - Initialization of the time is important!

```
// send tuio events
TuioServer->initFrame(TUIO::TuioTime::getSessionTime());
tuioServer->sendFullMessages();
```