Experiences Building The Largest Multitouch Screen in Latin America

Ariel Molina Rueda
ariel@edis.mx
EDIS Interactive

@ariel_mr
edisinteractive
Building The Largest Multitouch Screen in Latin America

Agenda

A talk is to share my experience building a huge multitouch screen!
... and how Qt helped along the adventure

• Why this monster multitouch screen?
  • What we started doing & what we ended doing
  • Where & how was it done
• What's inside?
  • Qt at the tracker & sensor multiplexor
  • Qt Quick at the UI level

Qt truly everywhere and we really pushed some limits.
Building The Largest Multitouch Screen in Latin America

Ariel Molina (PhD Cand)

- Founder & lead developer at EDIS Interactive
- Incessant bug reporter (desert) at QtC Bugtracker
- Qt Ambassador, Evangelist at academia & industry at México

EDIS Interactive

- Creates huge interactive surfaces, either touch or other
- Academy, education & fun, and recently alongside medical partners
- We love what we do
Building The Largest Multitouch Screen in Latin America

This is what we started doing:

**QGLWidget for Qt 4.x** and lot's of custom animation
Building The Largest Multitouch Screen in Latin America

This is what we ended doing:
Qt 5 C++, OpenCV, Qt Quick, QPA Plugins, Custom Affectors, C++ Quick
Building The Largest Multitouch Screen in Latin America

Where was it done?
- IBERO University at Puebla, México
  *(That's 90km southwest of Mexico City)*

CONACYT Supported Federal Project
- Mexican Council for Science and Technology
- EDIS Interactive led two small teams at two universities,
  - IBERO and UPSLP
- Evolve LayerFX Interactive Bars for fun into a huge multi touchscreen
- Kickstart EDIS Interactive Startup
Building The Largest Multitouch Screen in Latin America

How was it done?

Hardware:

- **Two** 4 meter tempered glass units (currently the only in México)
- Laser light plane and **four** infrared-filtered cameras
- **Three** projectors (Will upgrade to four)
- Accelerated Xinerama over two nVidia GTX 650, an Intel Core i3
Building The Largest Multitouch Screen in Latin America

How was it done?

Software

- Qt 4.8, then upgraded to Qt 5
- There are several pieces of Qt software
  - High-speed finger tracker using OpenCV *(no reliable CCV by then)*
  - Console multitracker-muxer, “unlimited” mosaiced cameras *(CCV still severely lags behind at this moment)*
- And a homemade *qmlscene*-like viewer
Building The Largest Multitouch Screen in Latin America

Software (1/3). The tracker.

- Originally written in C++ Qt Widgets for ~Qt 4.6, with OpenCV lib
- Eventually ported to Qt 5
- Interesting bits:
  - IplImage ↔ QImage translators
  - Split Core & UI
  - Core runs in own thread (shared mem with UI)
  - Originally wrapped official TUIO_CPP
    (100% rewritten with QtNetwork, faster, cleaner)
  - Tangible & Silhouette data via OSC extensions
  - We've got it to run really fast, > 200 FPS
Software (1/3). The tracker interesting bits

• Interesting bits:
  
  • **IplImage ↔ QImage translators**
  
  • Split Core & UI
  
  • Core runs in own thread (shared mem with UI)
  
  • Originally wrapped official TUIO_CPP
    
    *(100% rewritten with QtNetwork, faster, cleaner)*
  
  • Tangible & Silhouette data via OSC extensions
  
  • We've got it to run really fast, > 200 FPS
Software (1/3). The tracker interesting bits

- Interesting bits:
  - IplImage ↔ QImage fast translators
  - **Split Core & UI**
    - Core runs in own thread (shared mem with UI)
    - Originally wrapped official TUIO_CPP
      - *(100% rewritten with QtNetwork, faster, cleaner)*
    - Tangible & Silhouette data via OSC extensions
  - We've got it to run really fast, > 200 FPS
Software (1/3). The tracker interesting bits

- Interesting bits:
  - `IplImage ↔ QImage` fast translators
  - Split Core & UI
  - **Core runs in own thread** (shared mem with UI)
  - Originally wrapped official TUIO_CPP
    - *(100% rewritten with QtNetwork, faster, cleaner)*
  - Tangible & Silhouette data via OSC extensions
  - We've got it to run really fast, > 200 FPS
Software (2/3).

The “tarengo” general-mosaic tracker multiplexor.

- Takes any amount of TUIO inputs → Delivers a single output
- Qt Console App with own QtNetwork OSC parser
- Simple JSON configuration
- Launches slave listeners and mixes into a single TUIO output
- EXTREMELY fast
  - Tested with up to 10 trackers @ 120 FPS each
- Delivers normal 120FPS flow
- CONS: Still lacks a couple of useful things
Software (2/3).

The "tarengo" general-mosaic tracker multiplexor.

- Takes any amount of TUIO inputs → Delivers a single output
- Any possible mosaic: N x M

With a QPA plugin it is now a normal multitouch screen, so everything works as expected, including QtQuick.
Software (2/3).

The “tarengos” general-mosaic tracker multiplexor.

- Qt Console App with own QtNetwork-based OSC parser
- Simple JSON configuration
- Launches slave listeners and mixes into a single TUIO output

```json
{
    "port": 3333,
    "target": "127.0.0.1",
    "verbose": true,
    "slaves": [
        {
            "port": "3340",
            "mapX1": "0.0, 0.0",
            "mapX2": "0.5, 1.0"
        },
        {
            "port": "3341",
            "mapX1": "0.5, 0.0",
            "mapX2": "1.0, 1.0"
        }
    ]
}
```
Building The Largest Multitouch Screen in Latin America

Software (3/3). The User Interface.

- **Target user A: College student**
  - Designers, close to zero technical know-how
  - Some of them, just college rookies
  - Strong QtQuick preference
  - Works at first shot or hate it!

- **Target user B: Senior Developer**
  - Wizard-level technical knowledge
  - No problems with these
  - How to deploy apps, easy... no really easy
  - First idea: QtCreator Deployment...

I can haz multitouch app!
... magically uploading: OK

Image: Courtesy of emanueleferonato.com
Software (3/3). The User Interface.

- Target user A: College student
- How to deploy apps, easy... no really easy: 1) **QtCreator Deployment**...

1) Select compiler
2) Create Linux Generic Device, Set IP address, password, test
3) Create New Kit, Duplicate Settings
4) Select Generic Device, Select your device
5) Select New Kit
6) Oh, and don't forget to be in the LAN
Software (3/3). The User Interface.

- **Target user A: College student**
- **How to deploy apps, easy... no really easy:** 1) **QtCreator Deployment**...
Software (3/3). The User Interface.

- **Target user A: College student**
- **How to deploy apps, easy... no really easy:** 1) **QtCreator Deployment**...
Software (3/3). The User Interface.

- **Target user A: College student**
- How to deploy apps, easy.

- **Idea #2**
  - We setup Samba shared folder
  - Create Qt Quick UI (they already have some)
  - Move Folder
  - We make UI load it magically

You promised it to be easy
Software (3/3). The User Interface.

- Target user A: College student deploying apps
- **Idea #2: Samba shared folder**
  - Create Qt Quick UI (they already have some)
  - Move Folder
  - We make UI load it magically, so...
- ... we created a `qmlscene` inspired viewer
- ... monitoring a local folder,
- ... and it worked like a charm

I can haz multitouch app!
...magically-samba uploading: OK
Software (3/3). The User Interface.

• Target user A: College student deploying apps

• Idea #2: Samba shared folder
  • Create Qt Quick UI (they already have some)
  • Move Folder
  • We make UI load it magically, so...
  • ... we created a qmlscene inspired viewer
    ... monitoring a local folder,
    ... and it worked like a charm

I can haz multitouch app!
...magically-samba uploading: OK

PROBLEM SOLVED
Building The Largest Multitouch Screen in Latin America

Software (3/3). The User Interface.

- Target user A: College student deploying apps
- Idea #2: Samba shared folder
  - ...
- We created a qmlscene inspired viewer
- And it worked like a charm
- But it was not charming, need to make it “Cute”

I can haz multitouch app!

...magically-samba uploading: OK
Software (3/3). The User Interface.

- Target user A: College student deploying apps
- Idea #2: Samba shared folder
  - ...
- We created a qmlscene inspired viewer
- And it worked like a charm
- But it was not charming, need to make it “Cute”

I can haz multitouch app!

...magically-samba uploading: OK
Building The Largest Multitouch Screen in Latin America

Software (3/3). The User Interface, making it “Cute”.

• Uglyness due to projector non-blending

• More visible for expectators → unprofessional look
Software (3/3). The User Interface, making it “Cute”.

- **Uglyness** due to lack of projector blending
- More visible for expectators → unprofessional look
Software (3/3).

The User Interface, making it “Cute”.

But this is a whole different problem
Building The Largest Multitouch Screen in Latin America

**Software (3/3).**

**The User Interface, Edge Blending**

*Non trivial techniques, commercial software is expensive, see Watchout or Rhino*

**General Idea:**
1) Overlap projector instead of align
2) Split projector segments
3) Darken edges to compensate brightness
4) Rewarp mouse/touch coordinates
5) …
6) Fun
Building The Largest Multitouch Screen in Latin America

Software (3/3).

The User Interface, Edge Blending

**General Idea:**

1) Overlap projector instead of align
   1) Past: we aligned carefully projector
   2) Now: we overlap them **carelessly**

**easy part!**
Building The Largest Multitouch Screen in Latin America

Software (3/3).

The User Interface, Edge Blending

General Idea Step 2 & 3:
2) Split projector segments
3) Darken edges to compensate brightness

QtQuick is a Texture
1) Put whole scene as ShaderSource
2) Render just the needed slice
3) While being there, darken edges
Software (3/3).

The User Interface, Edge Blending

**General Idea Step 2 & 3:** Split & Darken

QtQuick Tree as a Texture

1) Put whole scene as ShaderEffectSource
2) Render just the needed slice using ShaderEffect (N-times)
3) While being there, darken edges
Software (3/3).

The User Interface, Edge Blending

**General Idea Step 2 & 3:** Split & Darken

QtQuick Tree as a Texture

1) Put **whole scene as ShaderEffectSource**

**Here be dragons!**

Found a couple of QTBUG's, like Webkit refused to render... now solved
Software (3/3).

The User Interface, Edge Blending using QtQuick and GLSL

Bonus points: We implemented generic homography warping
Software (3/3).
Warp each image, because axis-alignment is a nightmare.

Simple Quad Warping
Ugly transformation, incorrect coords

Homography Warping
Perfect perspective correction
Software (3/3).

Before homography, projectors carefully aligned
Software (3/3).

After homography, projectors awfully aligned, ShaderEffect fixes it.
Software (3/3).
The User Interface, Edge Blending

General Idea Step 2 & 3

QtQuick Tree as a Texture

1) Put whole scene as ShaderEffectSource
2) Render just the needed slice using ShaderEffect (N-times)
3) While being there, darken edges
4) Bonus: Homography warping
5) **Finally**: Put projectors however fits your taste, align them with QtQuick
Software (3/3).
The User Interface, Edge Blending
Software (3/3).

The User Interface, Edge Blending RESULTS
Software (3/3).

The User Interface, Edge Blending RESULTS

Really hard to see any imperfections
Software (3/3).

The User Interface, Edge Blending RESULTS

Warp carefully and get a perfect image
Software (3/3).
The User Interface
So, how was it, remember?

General Idea:
1) Overlap projector instead of align
2) Split projector segments
3) Darken edges to compensate brightness (and warp)
4) Rewarp mouse/touch coordinates
5) ...
6) Fun

Just one last step to do!
Building The Largest Multitouch Screen in Latin America

Software (3/3).

The User Interface

So, how was it, remember?

**General Idea:**
1) Overlap projector instead of align
2) Split projector segments
3) Darken edges to compensate brightness (**and warp**)
4) **Rewarp mouse/touch coordinates**
5) ...
6) Fun

Just one last step to do!
Software (3/3).

The User Interface

- Last step is easy, we just used coordinate remapping
- Our multiplexor, “tarengo” did it automatically
- Define a TUIO tracker mapping at slices
- It works

```javascript
{
  "port": 3333,
  "target": "127.0.0.1",
  "verbose": true,
  "slaves": [
    {
      "port": "3340",
      "mapX1": "0.0, 0.0",
      "mapX2": "0.5, 1.0"
    },
    {
      "port": "3341",
      "mapX1": "0.5, 0.0",
      "mapX2": "1.0, 1.0"
    }
  ]
}
```
Software (3/3).

The User Interface

- Last step is easy, we just used coordinate remapping
- Our multiplexor, “tarengo” did it automagically
- Define a TUIO tracker mapping at slices
- It works

... but that's TUIO network traffic, how does Qt handles TUIO?
Software (3/3).

The User Interface

- Last step is easy, we just used coordinate remapping
- Our multiplexor, “tarengo” did it automagically
- Define a TUIO tracker mapping at slices
- It works

... but that's TUIO network traffic, how does Qt handles TUIO?

Qt Platform Abstraction Plugin

```json
{
    "port": 3333,
    "target": "127.0.0.1",
    "verbose": true,
    "slaves": [
        {
            "port": "3340",
            "mapX1": "0.0, 0.0",
            "mapX2": "0.5, 1.0"
        },
        {
            "port": "3341",
            "mapX1": "0.5, 0.0",
            "mapX2": "1.0, 1.0"
        }
    ]
}
```
Software (3/3).

The User Interface is Native Touch

- Platform Abstraction Plugin
- Listen for TUIO input at launch
- Setup a TouchDevice
- Map TUIO Cursors almost without effort, they map nicely to QTouchPoints

- And then PinchArea, MultipointArea and friends work perfect!
Resume:

1. Create QtQuick UI with Qt Creator
2. Copy the folder to remote server (Samba)
3. Use it
Building The Largest Multitouch Screen in Latin America

Thank you

ariel@edis.mx
@ariel_mr
edisinteractive

Experiences Building The Largest Multitouch Screen in Latin America