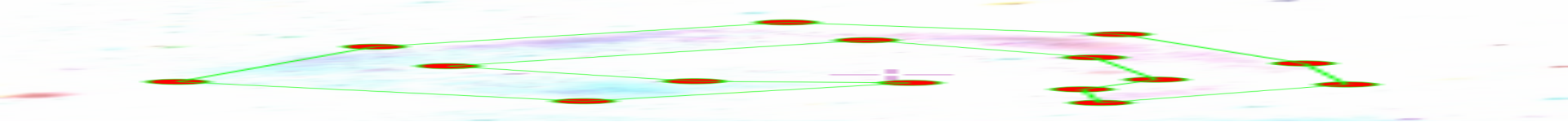


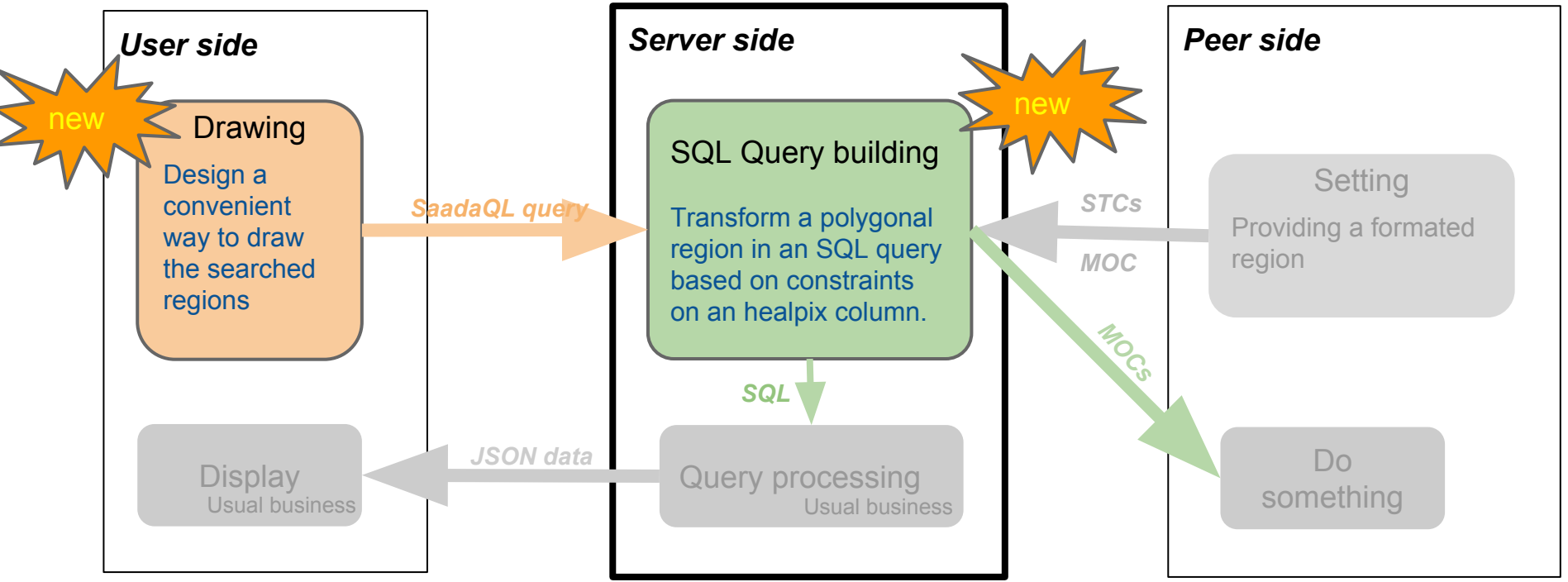
Querying By Region
From Drawing to SQL Queries

Laurent Michel - Thomas Boch
Julien Remy - Gerardo Irvin Campos Yah

The context of the project

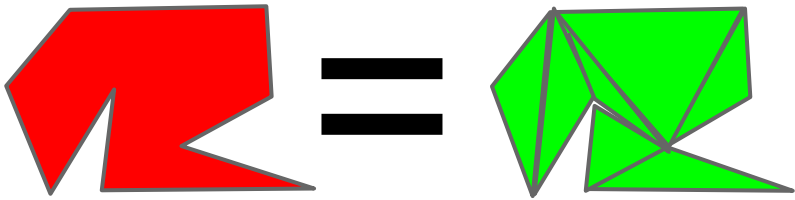
- Enabling the XCatDB to support search by region
  - ADASS poster 6.10
  - XMM catalogue interface: http://xcatdb.unistra.fr
- This development first targets Saada
  - Must work with PSQL, MySQL or SQLite
  - Must work with simple constraints on one Healpix column
- Loose coupling with Saada code
  - Java (MOC Healpix library)
  - Local Java packages
  - Javascript + JQuery
  - Easily exportable

2 facets, some use cases



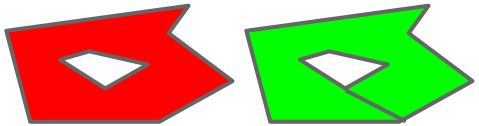
Server side key-points

- The Healpix library does not support concave polygons
- Validating the polygon
  - 3 vertices at least
  - A correct topology
- Splitting the polygon as a triangle set
  - Triangles are always convex
  - Work on an euclidean frame
    - Just interested on topology, not in size or distance
    - Make the vector computation quite easier



WHAT'S A GOOD POLYGON

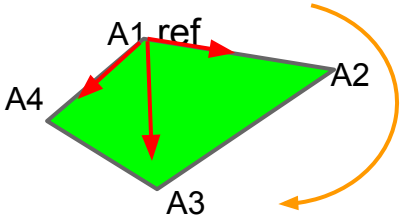
- The contour must be unique



- Must be oriented in the right direction

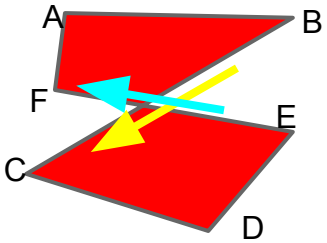
$$SUM\ (\ \overrightarrow{A_{ref}A_i}\ \wedge\ \overrightarrow{A_{ref}A_{i+1}}\ )\ <\ 0$$

Orientation changed as required



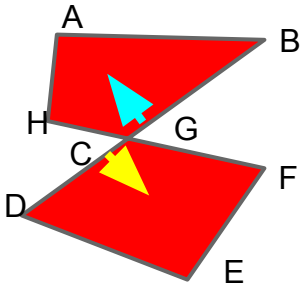
- Mustn't be self secant

abort (could be fixed by reordering the vertices)

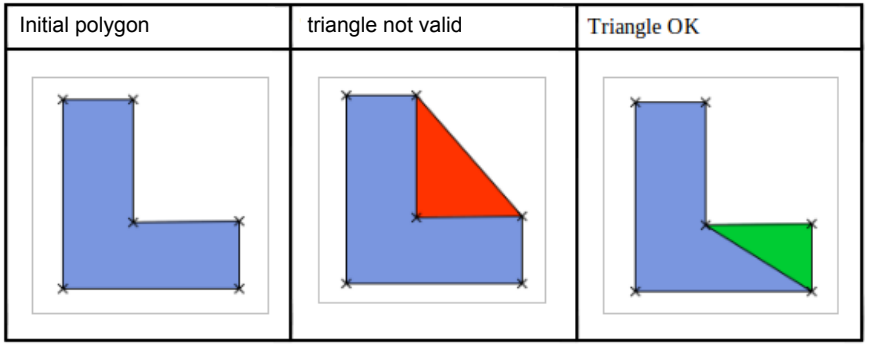


- Must be unfolded: inner side always on the right

abort (could be fixed by reordering the vertices)



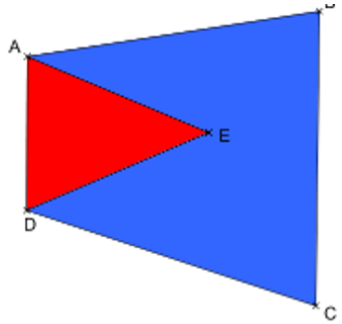
Triangulation process



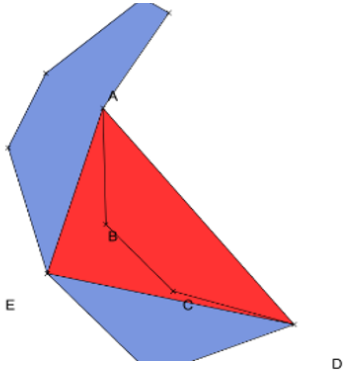
- **Cropping the polygon** with valid triangles containing at least 2 segments of the perimeter.
- The triangle vertex joining 2 contour segments is **removed**
- Job is over when the perimeter has 3 (or less) vertices

Triangulation traps

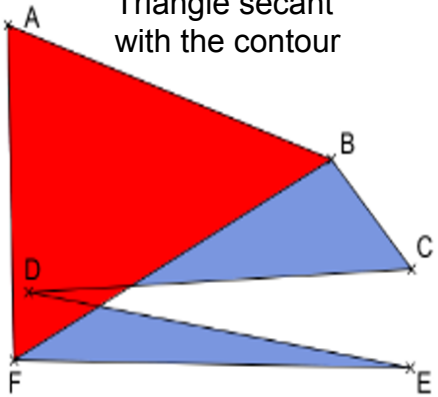
Outer triangle



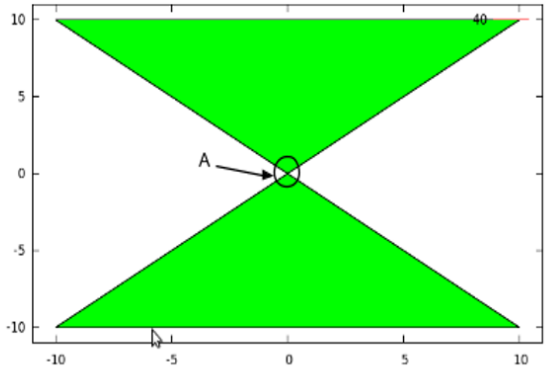
Other case of outer triangle



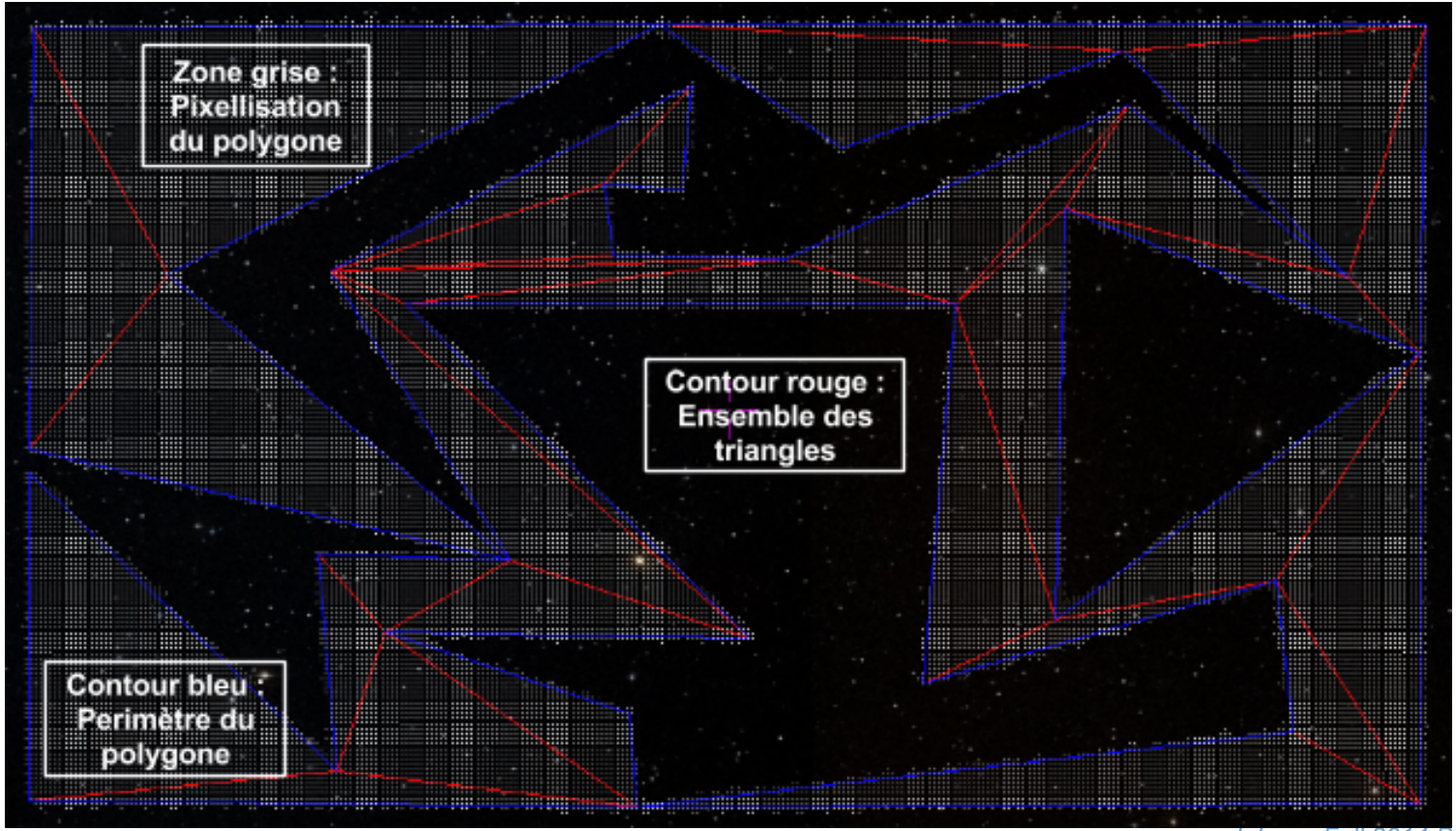
Triangle secant
with the contour



Hourglass Effect

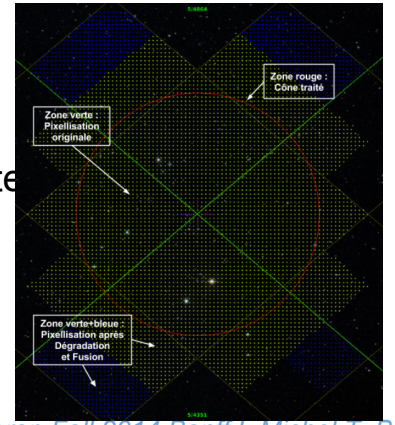


Validation (GNUPLOT)



HEALPIX SEGMENTATION

- **Healpixization**
  - The set of triangles is transformed into a set of Healpix pixels.
- **Segmentation**
  - [103] [104-106] [105-109] [110] → [103-110]
- **SQLization**
  - The segments are transformed in a WHERE statement
  - [103-109],[123],.. ⇒ (h BETWEEN 103 AND 109) OR h = 123 OR(..)
- **Adaptative resolution**
  - The resolution is degraded with a bit mask if there are too much ORed state
  - SQLite only supports 1000 constraints in one WHERE clause
  - [100111 - 110101] → [100100 - 110111]



SQL against SaadaQL

Input region (SaadaQL)

```
public static void ma (String[] args) throws Exception {
    Database.init("Th eXMM");
    Messenger.debug_m e = true;
    //WherePosition  new WherePosition("WherePosition {isInCircle(\"M33\",  ICRS) }");
    WherePositi wp = new WherePosition("WherePosition {isInRegion(\"202.45837,+47.29028, " +
          202.52390,+47.28777, 202.55595,+47.19132, 202.53863,+47.13598, " +
          "202.43629,+47.15609, 202.38558,+47.22314, 202.41519,+47.23658, " +
          "202.44488,+47.18796, 202.46833,+47.18378, 202.49177,+47.19301, " +
          "202.49177,+47.21481, 202.46336,+47.22738, 202.45590,+47.28860, " +
          "  -, ICRS) } ");


    System.out.println(wp.getSqlConstraint().replaceAll("OR", "\nOR"));
    Database.close();
}
```

SQL

```
[13/02/14 16:16:11]   DEBUG: Triangle number : 11
[13/02/14 16:16:11]   DEBUG: Current Healpix resolution: 15
[13/02/14 16:16:11]   DEBUG: Request Size : 154
 healpix_csa = 2890801663
OR healpix_csa BETWEEN 2890802004 AND 2890802007
OR healpix_csa = 2890802009
OR healpix_csa BETWEEN 2890802011 AND 2890802015
OR healpix_csa = 2890802023
OR healpix_csa BETWEEN 2890802029 AND 2890802047
OR healpix_csa = 2890802079
OR healpix_csa BETWEEN 2890802101 AND 2890802103
OR healpix_csa BETWEEN 2890802107 AND 2890802111
OR healpix_csa = 2890802113
OR healpix_csa BETWEEN 2890802115 AND 2890802175
OR healpix_csa BETWEEN 2890802346 AND 2890802347
OR healpix_csa BETWEEN 2890802350 AND 2890802351
OR healpix_csa BETWEEN 2890802362 AND 2890802363
OR healpix_csa BETWEEN 2890802366 AND 2890802367
OR healpix_csa = 2890802410
OR healpix_csa BETWEEN 2890802688 AND 2890802947
OR healpix_csa BETWEEN 2890802950 AND 2890802959
OR healpix_csa BETWEEN 2890802962 AND 2890802963
OR healpix_csa BETWEEN 2890802966 AND 2890803007
OR healpix_csa BETWEEN 2890803010 AND 2890803011
OR healpix_csa BETWEEN 2890803014 AND 2890803023
OR healpix_csa BETWEEN 2890803032 AND 2890803199
OR healpix_csa BETWEEN 2890803413 AND 2890803415
OR healpix_csa BETWEEN 2890803419 AND 2890803423
OR healpix_csa = 2890803437
OR healpix_csa = 2890803439
OR healpix_csa BETWEEN 2890803441 AND 2890803455
OR healpix_csa BETWEEN 2890803469 AND 2890803487
OR healpix_csa BETWEEN 2890803491 AND 2890803711
OR healpix_csa BETWEEN 2890803779 AND 2890803783
OR healpix_csa BETWEEN 2890803785 AND 2890803809
OR healpix_csa BETWEEN 2890803811 AND 2890803815
OR healpix_csa = 2890803817
OR healpix_csa BETWEEN 2890803820 AND 2890803839
OR healpix_csa BETWEEN 2890803908 AND 2890803909
OR healpix_csa = 2890803911
OR healpix_csa BETWEEN 2890803920 AND 2890803935
OR healpix_csa = 2890803953
OR healpix_csa BETWEEN 2890803955 AND 2890803959
```

A rather simple API

```
if( this.operator.equals("isInRegion")) {
    List<Point> pts = new ArrayList<Point>();
    for( int i=0 ; i<this.positions.size() ; i++ ) {
        pts.add(new Point(this.positions.getRa(i), this.positions.getDec(i)));
    }
    Polygone p = new Polygone(pts);
    Zone r = new Region(p, Database.getAstroframe());
    retour = r.getSQL();
    //System.out.println(r.getGnuplotScript());
```

Generating an SQL statement from a coordinate sequence
No DBMS dependency

```
public static String getWhereMoc(HealpixMoc hm, int resolution) throws Exception{
    long[] pix= hm.getPixLevel(resolution);
    ListeSegment ls = new ListeSegment(pix);
    RequeteCreator rc = new RequeteCreator(ls);
    return rc.getWhere();
}
```

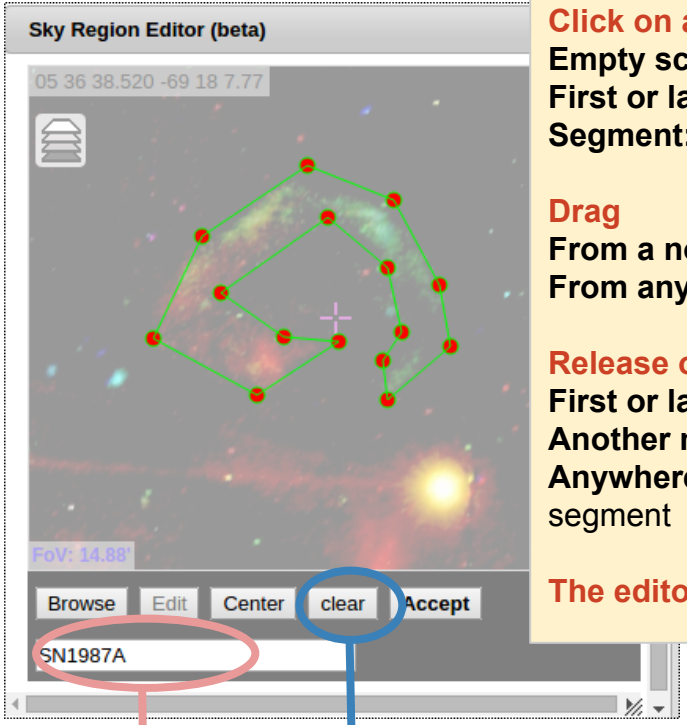
Generating an SQL statement from a MOC
Just bypass the triangulation.

- The best way to setup a region is to draw it by hand upon a sky view.
  - Need to identify the search region
  - The possibility of choosing the survey is welcome
- The obvious candidate for providing a sky view on a Web browser is Aladin Lite
  - 2 options
    - Including the drawing feature within Aladin Lite
    - Including Aladin Lite within the drawing tool.
  - 2nd solution safer while there is no standard plugin interface in Aladin Lite

Client side / Design key-points

- Need to keep the Aladin Lite controls working
  - click - drag - mouse wheel
- Need the same controls for drawing
  - click - drag at least
  - intuitive
- Overlaying Aladin Lite with a canvas
  - 2 modes (browse / draw) toggled by the user
    - **Browsing mode**
      - The polygon is overlayed on the Aladin image and the drawing canvas is hidden
    - **Drawing mode**
      - Aladin Lite is frozen and the overlayed polygon is hidden.
  - The drawing mode works with cartesian coordinates while Aladin Lite uses sky coordinates.
    - The polygon is imported/exported from/to the canvas thanks to the Aladin Lite API (`world2pix/pix2world` methods)

Drawing Panel



Click on a:
Empty screen: put the first node
First or last node: start new segment
Segment: add a node

Drag
From a node: move it
From anywhere else: nothing happens

Release on
First or last node: close the polygon
Another node: merge the 2 nodes
Anywhere else: ends or update the segment

The editor prevents crossing segments

Validate the drawing
Polygon must be closed

Clear the current polygon

Center the polygon view

Center Aladin Lite on a position or object
CR to validate

Mode selector

JavaScript API

```
Modalinfo.region(
        function(data){
            if( data.userAction )
                alert(JSON.stringify(data));
            }
        , {type: "array"
        , value: [84.24901652054093,-5.640882748140112,
                  83.34451837951998,-6.103216341255678,
                  83.60897420186223, -4.553808802262613]
           }
    );
```

Opening the region editor in the `jsresources` context

The widget initialisation needs
A **handler** processing the polygon completion
A **polygon** to be drawn first

The handler is called
When the **polygon is closed**
When the **user click** on *accept*

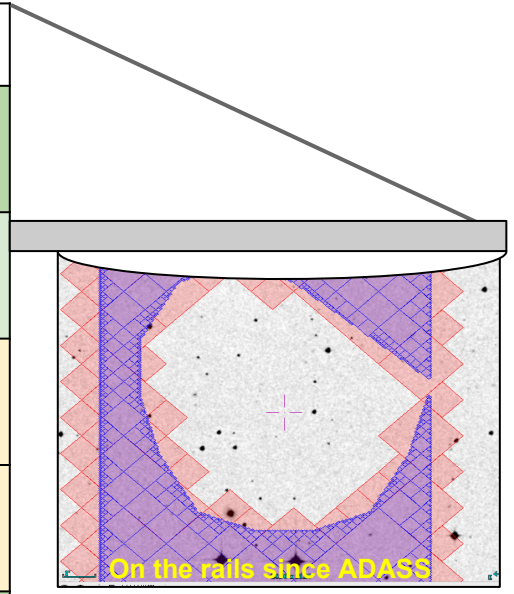
The Data passed to the handler give
The **status** of the polygon
The **action** at the origin of that callback
The sequence of **points**
The polygon **size**

```
{
    isReady: true,          // true if the polygon is really closed
    userAction: true,       // handler called after the user has clicked on
    region : {
              format: "array2dim",   // The only one format supported yet [[x, y]....]
              points: [[x,y],[...],..] ,   // array with structure matching the format
              size: {x: , y:}              // region size in deg
              }
}
```

JSON object passed to the handler

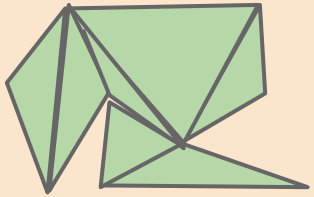
Toward a XYZ ⇔ SQL/MOC wrapper

input	Output	Purpose
Point sequence (By hand edited)	SQL	Implementing a search by region in a simple SQL database
Point sequence (By hand edited)	MOC	Providing a region editor for resources supporting the search by MOC
STCs (workflow, datalink, Obscore)	SQL	Implementing a search by region in a simple SQL database
STCs (workflow, datalink, Obscore)	MOC	Providing a region editor for resources supporting the search by MOC
MOC (workflow, Aladin, Topcat)	SQL	Implementing the search by MOC in a simple SQL database



Not packaged to be made publicly available yet
can however be distributed on demand

Toward a XYZ/SQL wrapper ?!



Sequence of coord.

MOC

STCs
next in the queue

[103] [104-106] [105-109] [109] → [103-109]

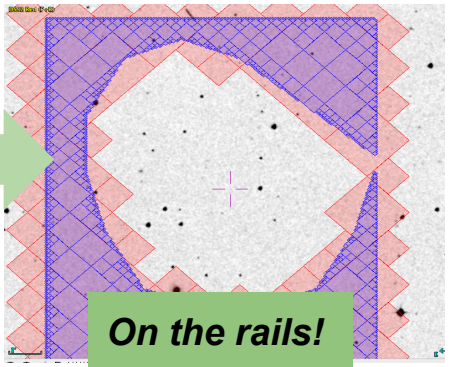
[100111 - 110101] → [100100 - 110111]

(h BETWEEN 103 AND 109) OR h = 123 OR(..)

Java package

SQL statement

MOC



On the rails!

Done

To be done