Artworks narrating a story:
a modular framework for the integrated presentation of three-dimensional and textual contents

MARCO CALLIERI
CHIARA LEONI
MATTEO DELLEPIANE
ROBERTO SCOPIGNO

VISUAL COMPUTING LAB ISTI-CNR PISA, ITALY

HTTP://VCG.ISTI.CNR.IT

Who am I / Who are we

Marco Callieri

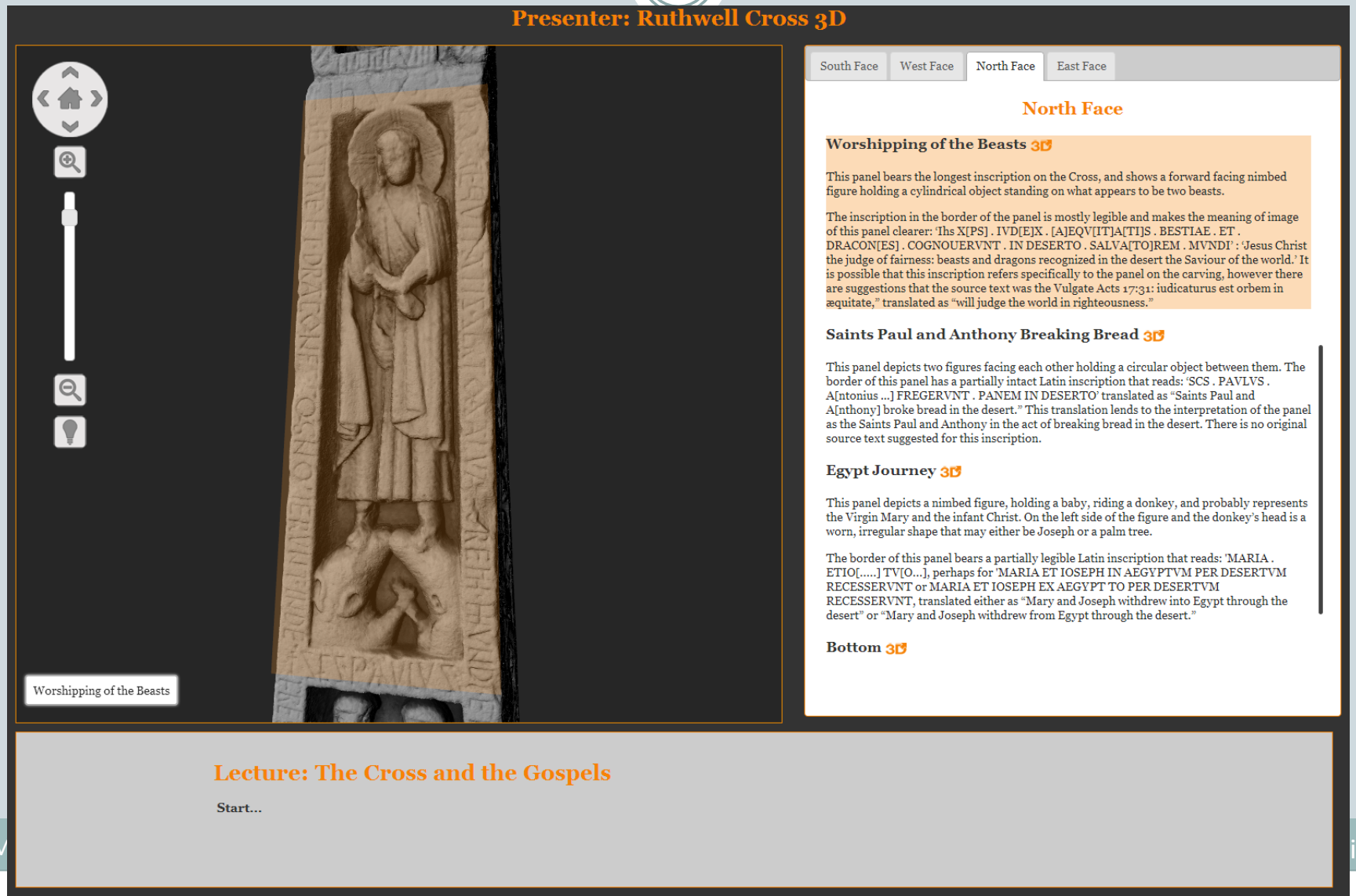
- 3D scanning, color acquisition, 3D model processing
- 3D for Cultural Heritage

Visual Computing Lab

~20 people, National Research Council of Italy, located in Pisa

- Basic 3D graphics algorithms, Interactive visualization (PC/Web/Mobile), CG for Cultural Heritage, Geometry processing,
- You may have heard/used our tools: **MeshLab**, **SpiderGL**

What am I here to present?



How all this work

ever started

The Visionary Cross Project

People from the **Visionary Cross Project** involved us in the 3D scanning of an important Anglo-Saxon artifact…

The dreamers of the "Dream of the Rood"

Interest of the project is to study the relationship of the ancient poem the *Dream of the Rood* with different CH artifacts, especially this cross

- Present the cross at different levels (as a ***cross***, as a ***narrative*** carved art piece, as the substrate for the ***poem***)
- Wide access => online
- Focusing on didactic purposes

http://visionarycross.org

The need for

~~speed~~ triangles

We need a 3D model…

… and a pretty complex one too! We are talking about millions of triangles…

It is difficult, for SceneGraph-oriented engines/libraries/viewers to manage highly complex, unstructured objects…

Solution:

Let us go **multiresolution**, CG style…

WebGL / SpiderGL

No need to explain *here* what WebGL is… powerful, but scarily low-level.

SpiderGL is a CG-oriented library, wrapping WebGL… It provides higher level structures AND low level control, making possible for CG programmers to write their own visualization code as in a normal software development

http://spidergl.org/

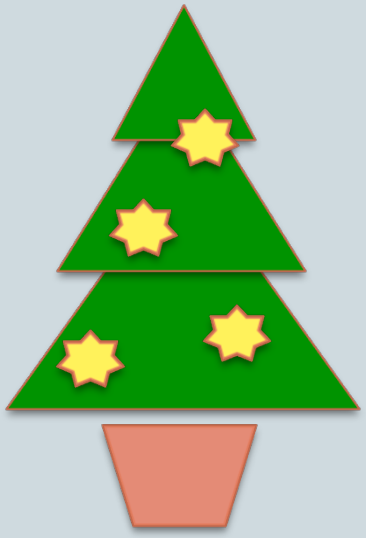
Our multiresolution structure has been ported from C++ to JavaScript using this library

The basic idea

of peer exploration

Some media *are* created equal…

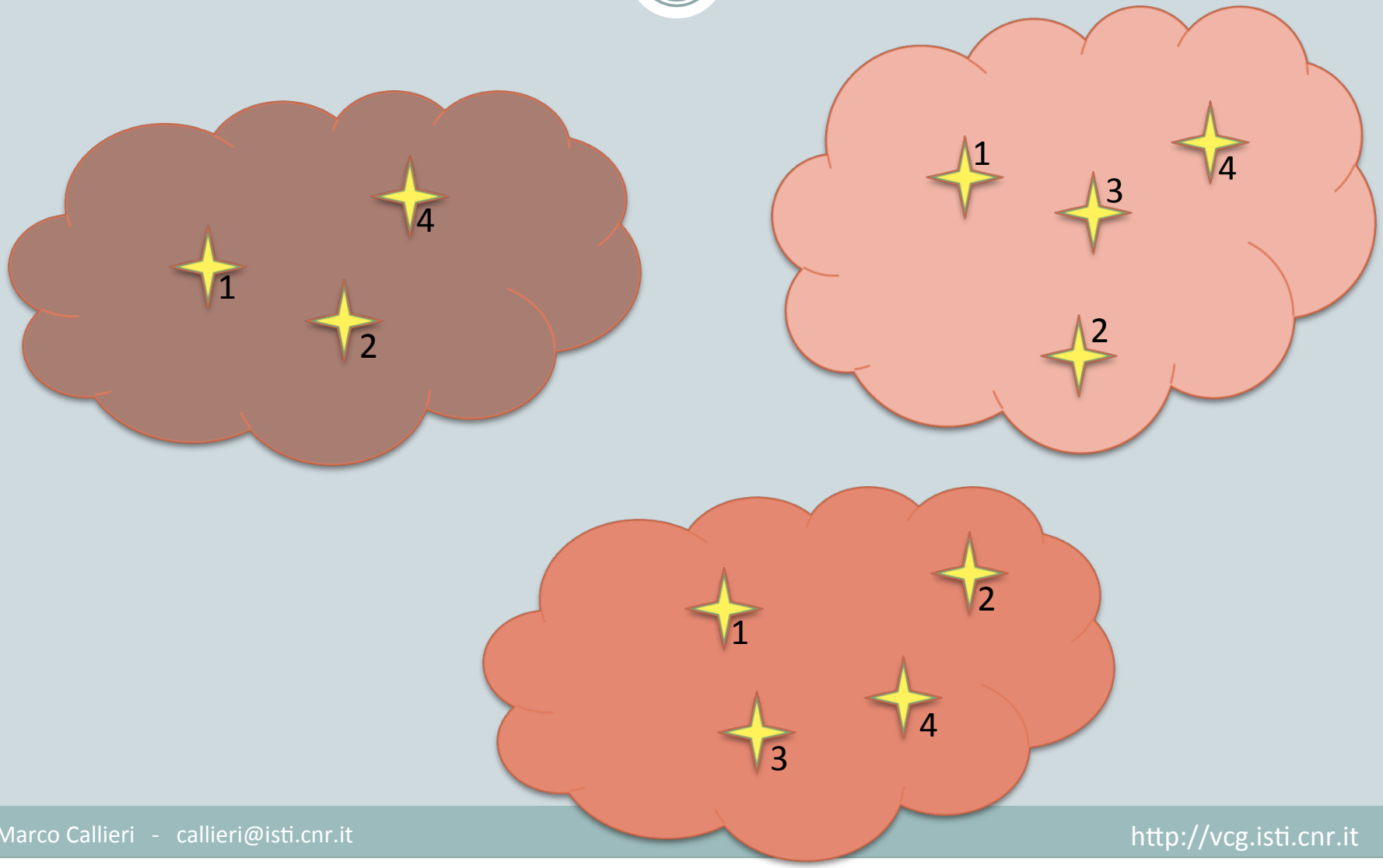
Most "multimedia" visualization schemes are just like a Christmas tree… there is a strong, complete main media, with other stuff dangling from its branches…



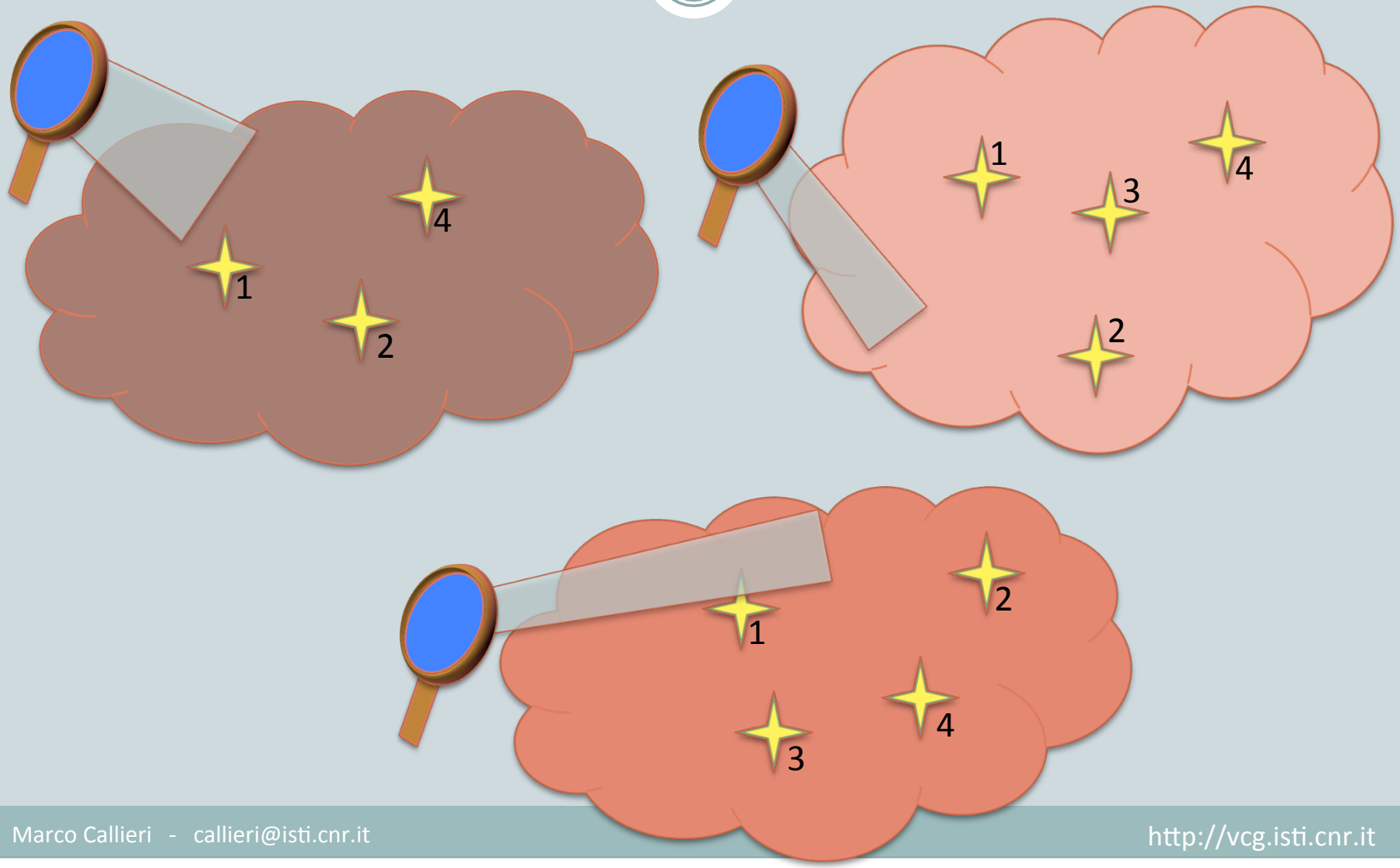
Nothing wrong with this… but in some cases, like this one, it is important to put _all_ the different media at the **same** level…

It should be possible to explore each dataset independently, and be able to switch between dataset in a meaningful way

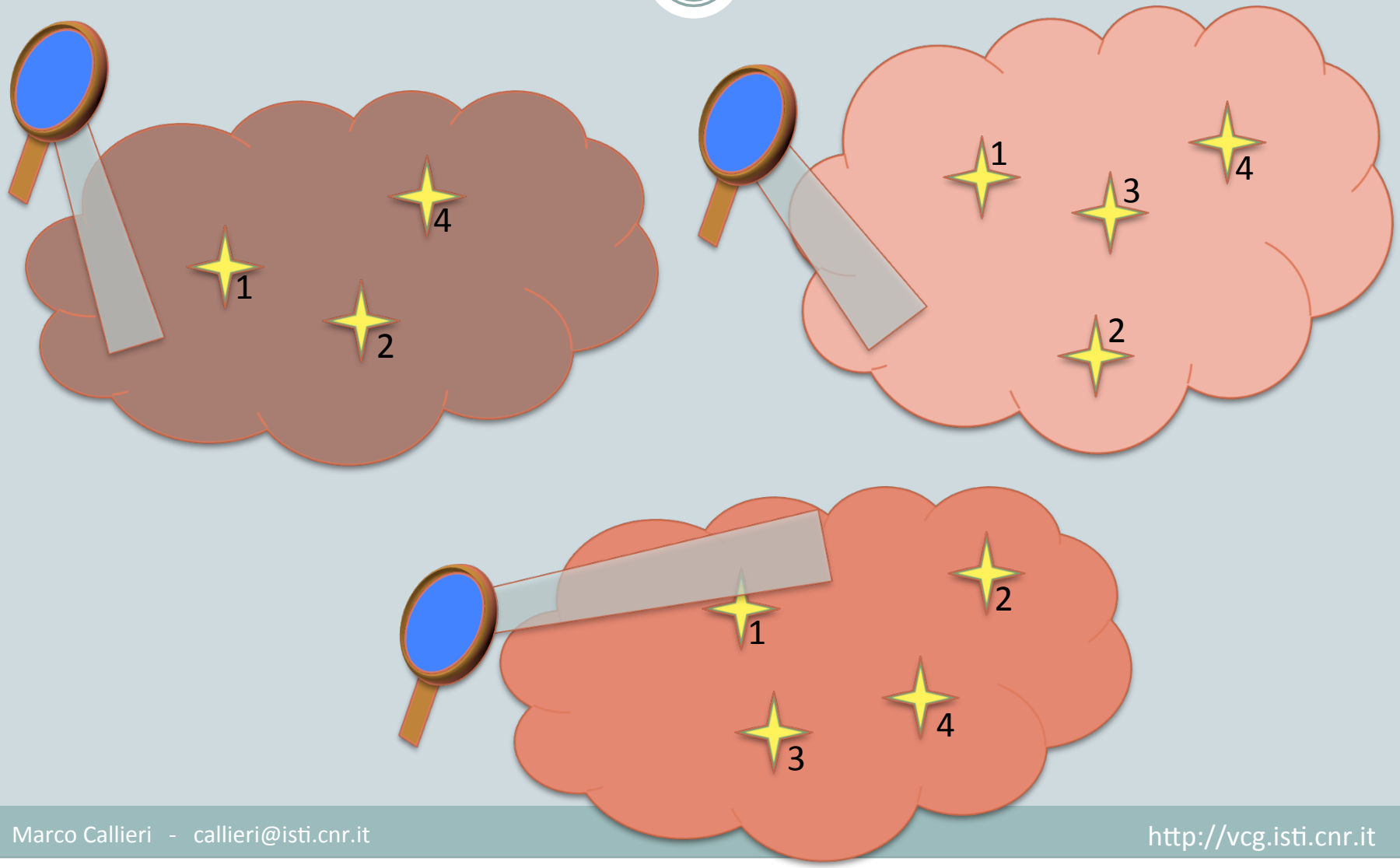
Viewer <-> Synch



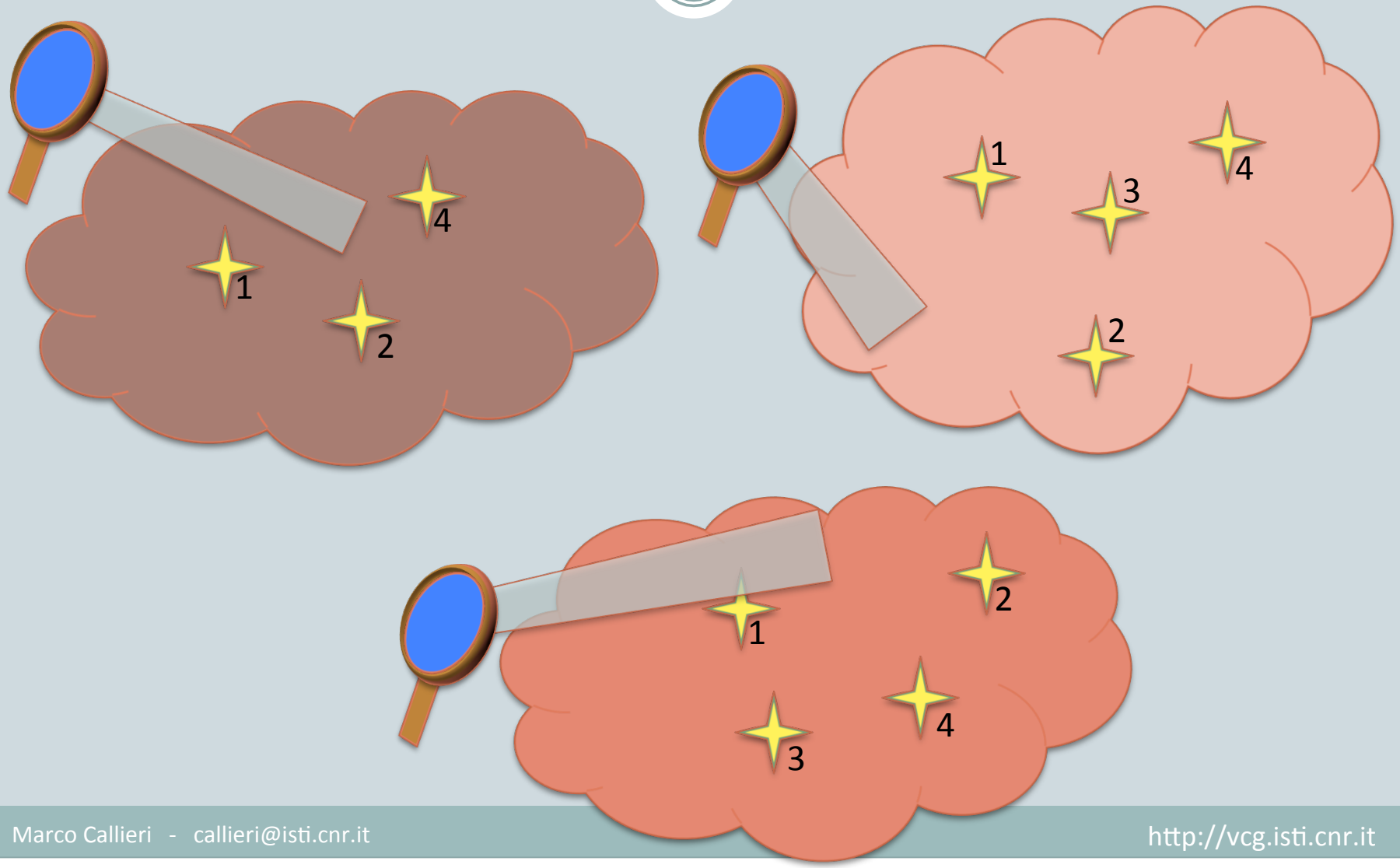
Viewer <-> Synch



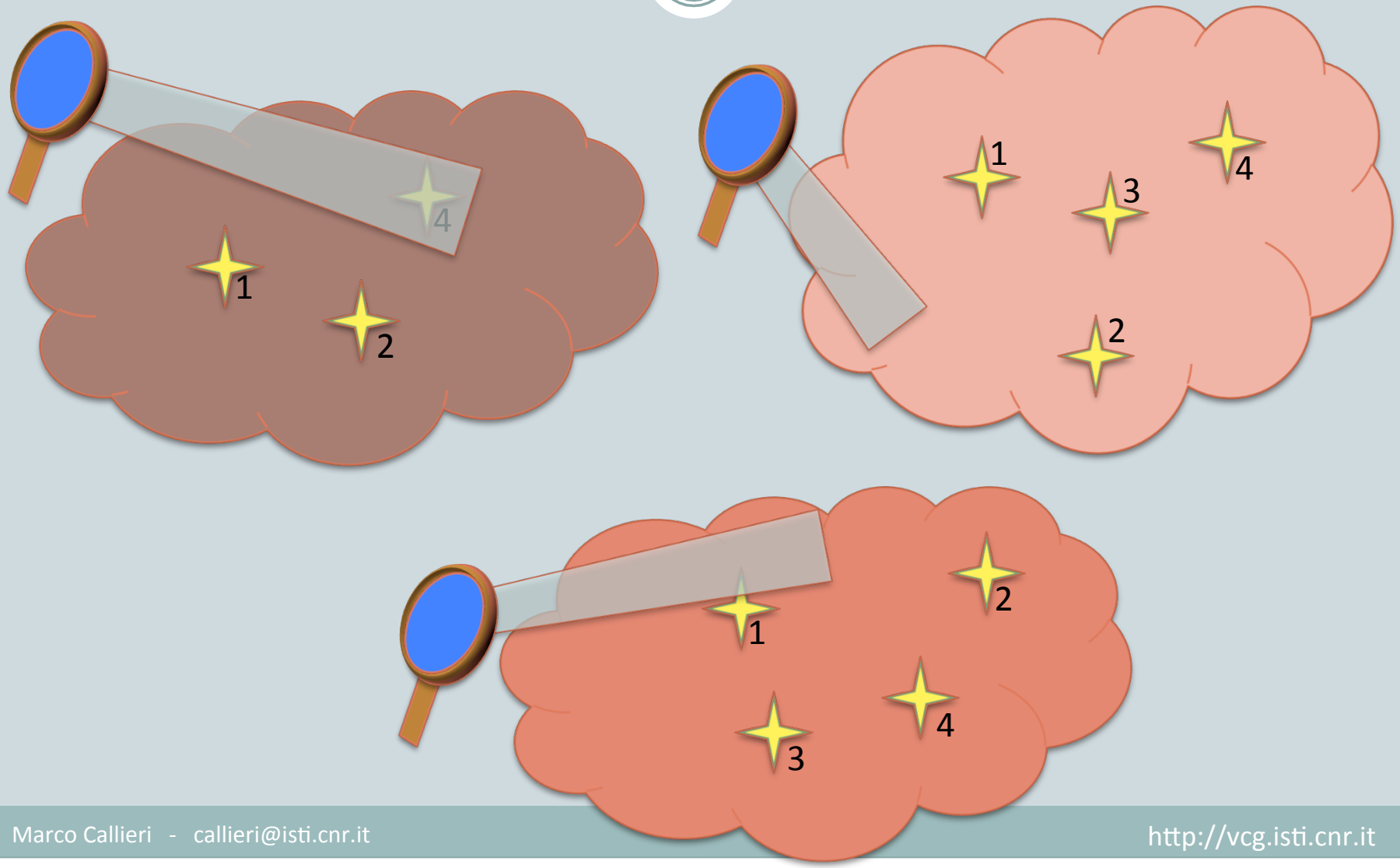
Viewer <-> Synch



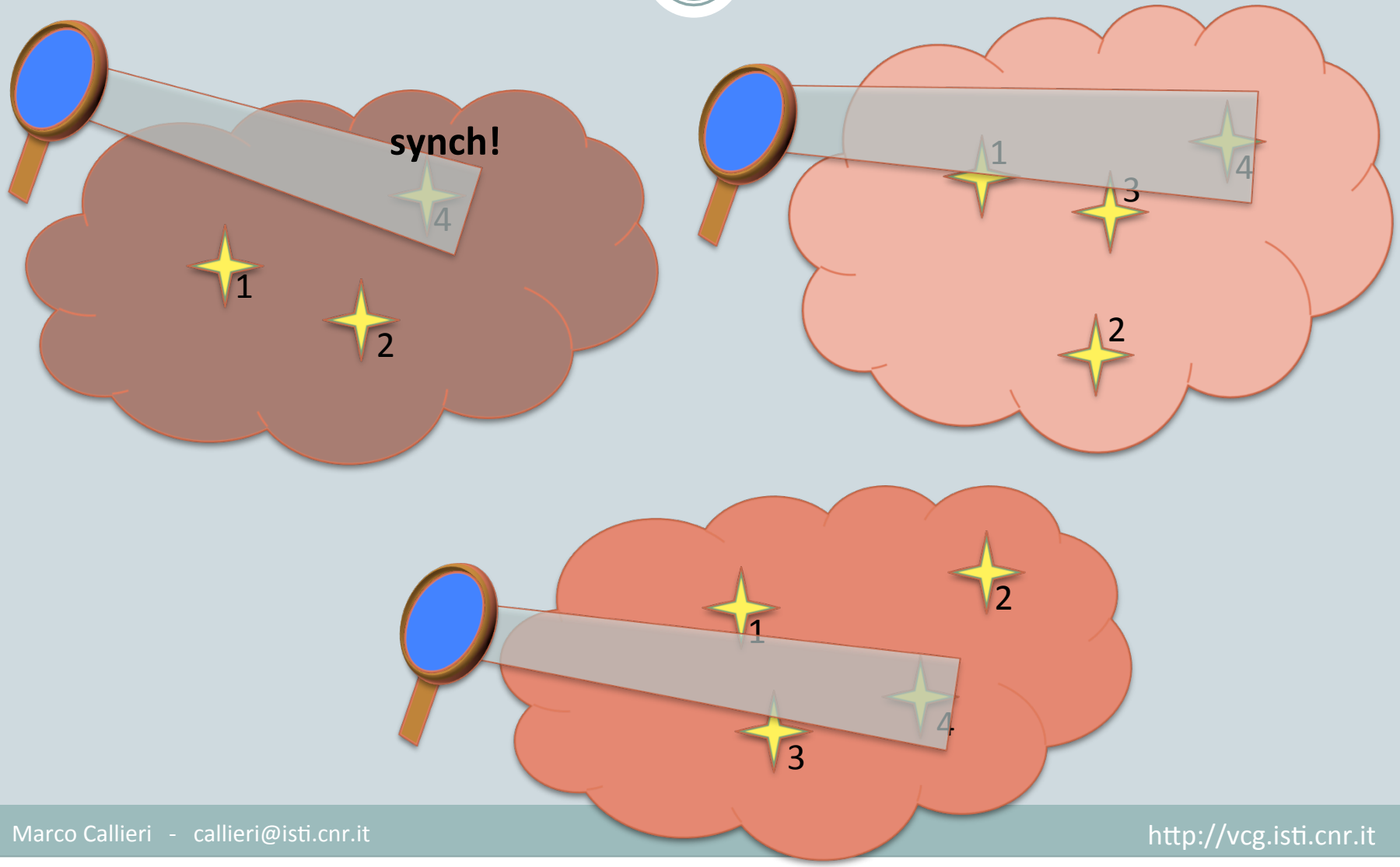
Viewer <-> Synch



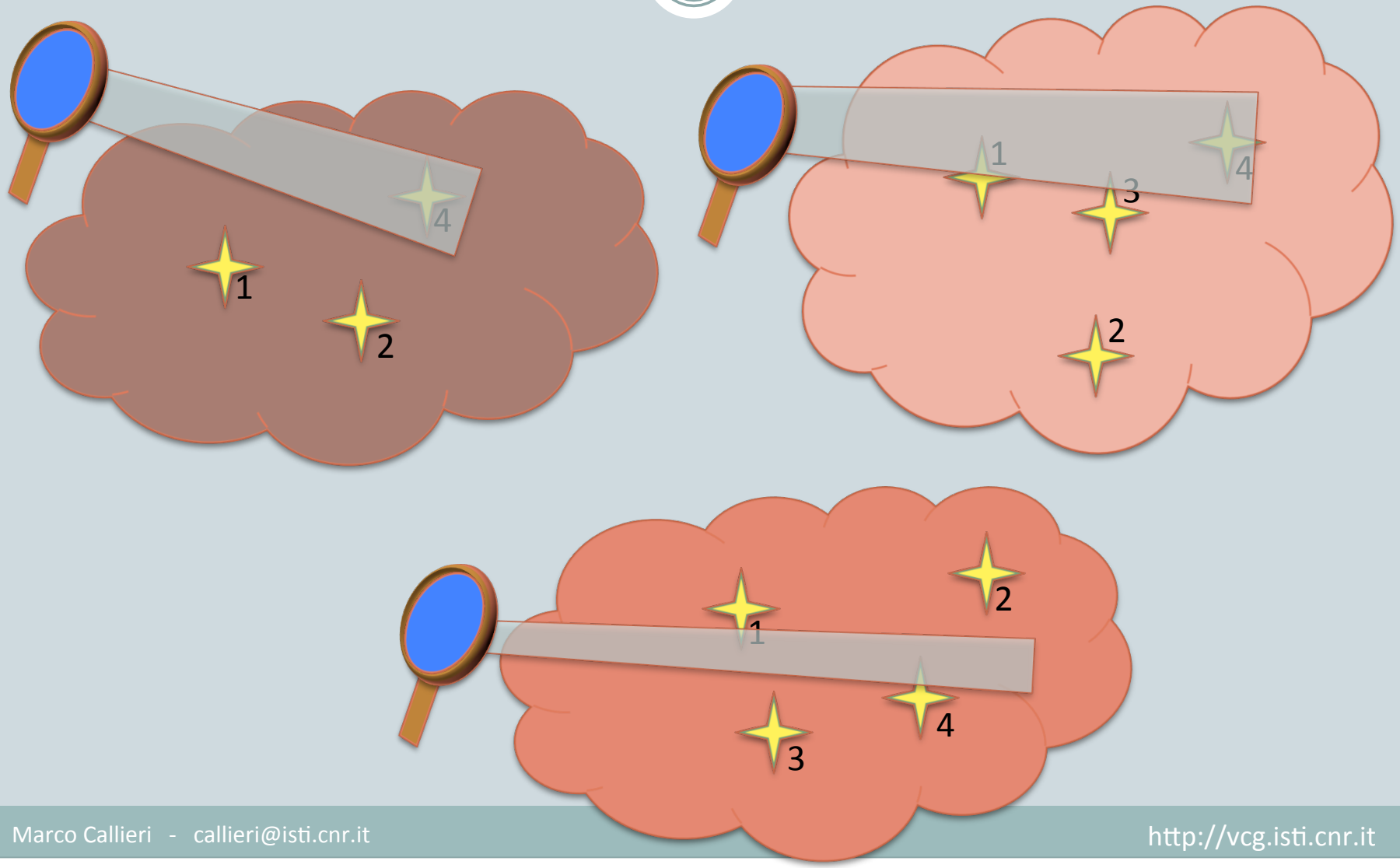
Viewer <-> Synch



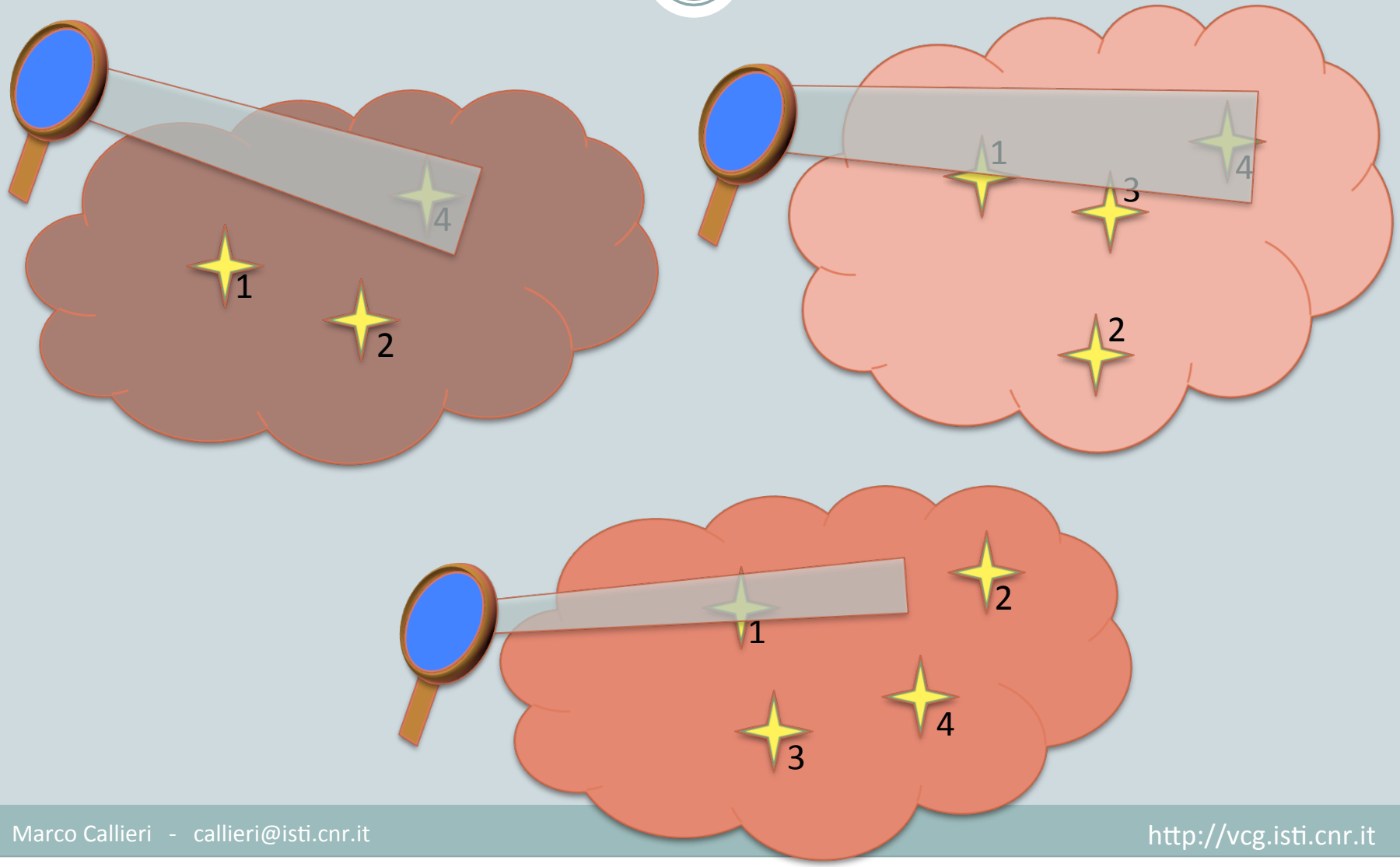
Viewer <-> Synch



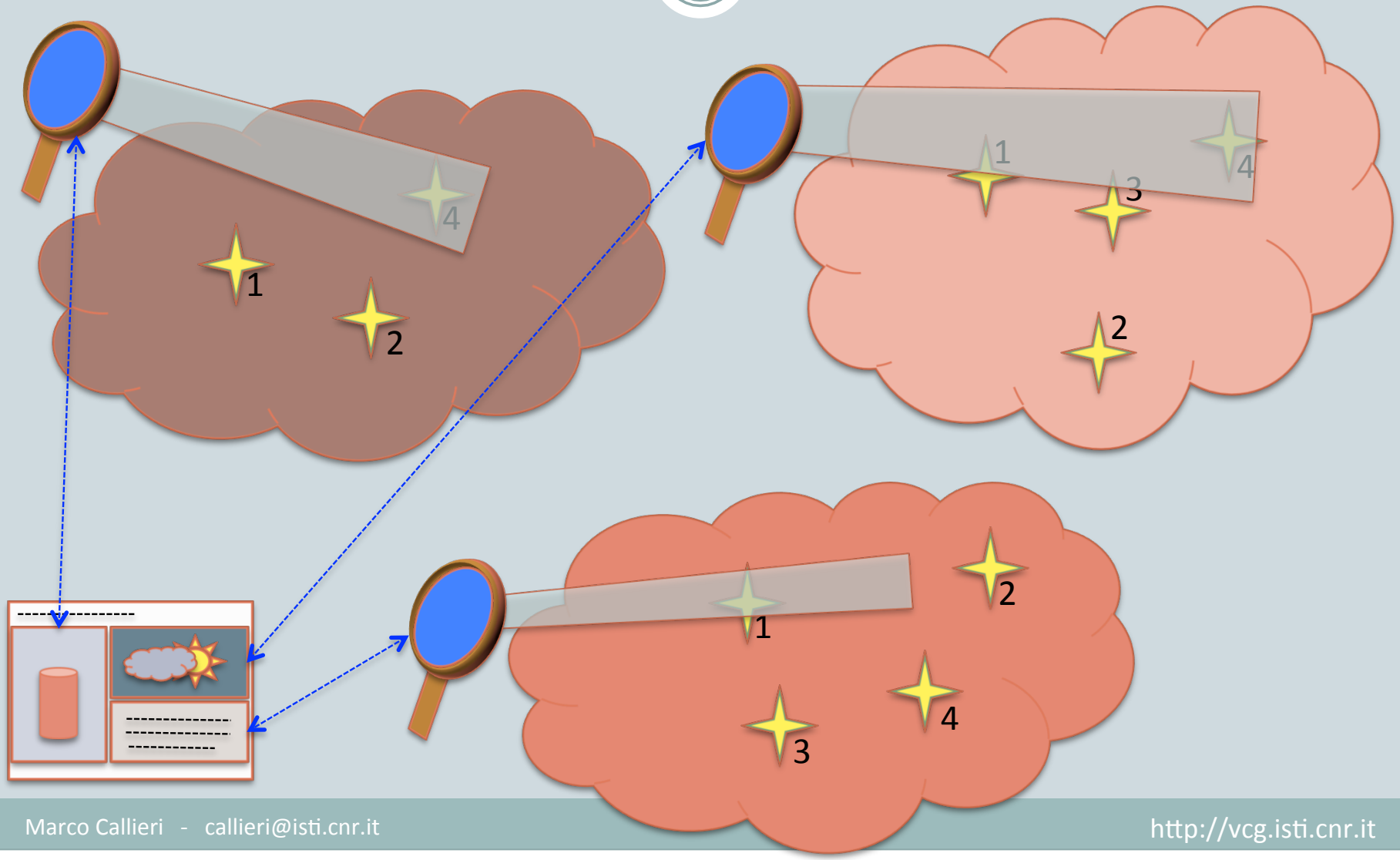
Viewer <-> Synch



Viewer <-> Synch



Viewer <-> Synch



How to make

things run

The 3D viewer

For each media involved, there is an associated “viewer”, as independent as possible from everything else…

Each viewer does need:

- A way to display and navigate the dataset
- A way to display “spots” and let user select them [synch(spot_id)=>]
- A way to focus to a specific position in the dataset [=>goto(spot_id)]

A global entity is used to keep a table of all spots, with utility data and dispatch messages between viewers…

The 3D viewer

Almost entirely SpiderGL, with some interface elements in HTML:

- Rendering of the dataset is done using multiresolution in SpiderGL
- A specific trackball, to cope with the shape of the object, is used to navigate the object
- Spots are rendered as simple geometries, and the user selects them using double-click
- The trackball can be animated to reach a specific point

The Structured Text viewer

almost completely HTML (with a little Javascript + JQuery):

- Dataset is arranged in tabs, scrolling panels and sections
- Spots are links (as every user would expect)
- Change of tab and scrolling is possible thanks to JQuery

Why all this hassle

for a single object

From single-purpose to a system

Step 1) decouple **data** from viewer code

Step 2) establish a suitable data format

Step 3) have someone (else) enter new data

Step 4) profit…

Things would be even better with an authoring tool…

The visualization page starts empty… placeholders for the various components (viewers, and interface elements) are placed by the author as it normally would editing the page HTML

XML configuration

All the components are configured using XML files, read at the loading of the page, and used to populate the page…

Text-based data is parsed and rendered in HTML inside empty DIVs in the main page, other elements, like the 3D viewer, are created on the fly inside elements of the page.

XML has been chosen because it's simple, easy to edit by non-programmers (CH people), and can be automatically generated from many existing tools.

Enter the Runes,

enter the problems

XML – XSLT

Instead of using the “programmer’s approach” of doing

[XML] => parsing into **[JSON]** => render into **[HTML]**

We wanted to follow the conventions of Digital Editions

- Encoding comes first!
- Encoding uses XML compliant with **TEI** schemas/ conventions/guidelines
- XML is transformed in HTML using **XSL Transformation**

XML – XSLT looks good but…

As a programmer, I found much easier the code-based parsing/rendering…

XSL transformation can be invoked dynamically, but nothing remains in memory of the transformed structure… sometimes the XML has to be parsed twice (one with XSL, one with code)

No support for XSLT 2.0 in browsers… This mean that some "tricks" are needed for more complex XML structures

What comes

next

Future was yesterday

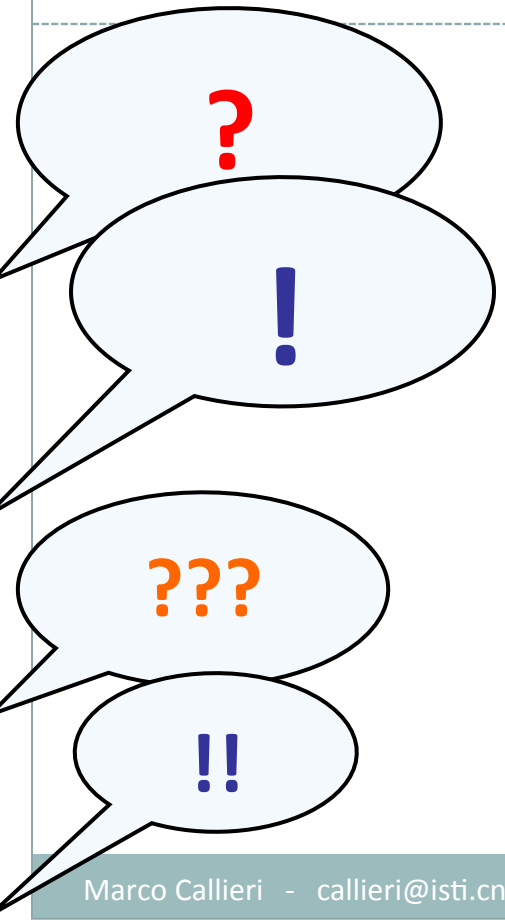
1. We do need an authoring tool… we need it badly!
2. We do need to completely explore the XML-XSLT solution, for digital editions compliant with existing standards
3. We do need to add more media viewers (images, videos)

(1) is in design stage…

We are currently working on (2) with a student for its thesis

For (3), we have a lot of code (in c++ / opengl) for managing and rendering large image collections

Thanks for your attention… Any question?



MORE INFO:

http://vcg.isti.cnr.it

http://visionarycross.org

CONTACT:

callieri@isti.cnr.it

roberto.scopigno@isti.cnr.it