

News from MOC Lib Rust, MOCPy and other derivatives

F.-X. Pineau¹, M. Baumann¹, T. Boch¹, T. Dumortier¹, P. Fernique¹, M. Marchand¹

¹Centre de Données astronomiques de Strasbourg

11th May 2023



□ MOC: an IVOA standard

- MOC = Multi Order Coverage Map
- IVOA standard (see P. Fernique's talk)
- **v1.0: discretized sky regions** description
 - HEALPix based
- **v2.0: add Time and Space-Time**
 - time: discretized quantity (microsec since $JD = 0$)
 - **uniform precision** (int64; microsec)
- **vx.x: add Frequency** (and Space-Frequency, ...)
 - frequency: continuous quantity
 - **relative precision** (float64; Hz)
- Main interests:
 - fixed systems (ICRS/TCB/Hz): **easy comparison**
 - describe **any region** (possibly complex)
 - **fast** operations (union/intersection/contains/...)

□ MOC Lib Rust

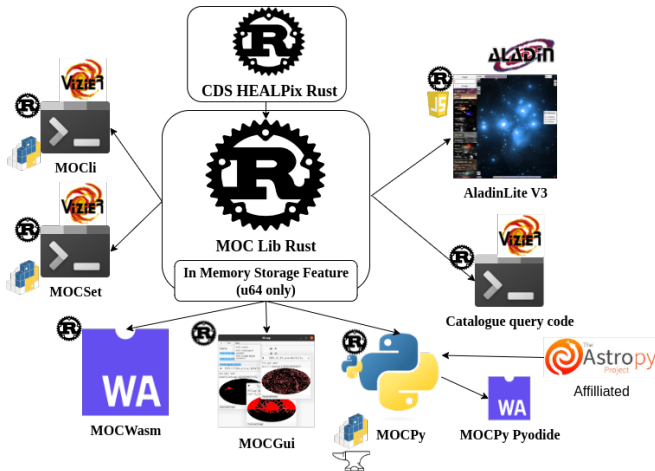


Figure 1: MOC Lib Rust ecosystem, see [IVOA April 2022 slides](#)

□ MOC Lib Rust functionalities

- Serialize/deserialize S/T/F/ST-MOCs
 - ASCII, JSON, (gzipped) FITS
- Build S-MOCs from:
 - cone, elliptical cone, box, zone, polygon, ring
 - list of positions, list of large/small cones
 - multi-order map, skymap
- Build T-MOCs, **F-MOCs**, ST-MOCs
- Logical operations (both standard and lazy):
 - complement, intersection, union, sym. diff, minus
 - multi union and intersection
 - lazy = operations on iterators => streaming mode
 - generic: same code for Space u32 or **Frequency** u64
- Other S-MOC operations
 - degrade, extend, contract, external/internal border
 - **split** into disjoint MOCs, **fill holes**, **find barycenter**, ...
- **S-MOC visualization**

□ New storage

- Before
 - MOCPy: **heterogeneous solutions** (historical reasons)
 - MOC, T-MOC: data (ranges) stored on Python side
 - ST-MOC: data stores on Rust side
 - MOCWasm: **MOC** data stored in WASM, **accessed by name**
 - MOCGui: **large duplication** of MOCWasm code
- Now
 - **One generic storage (u64) in MOC Lib Rust**
 - R/W protected **slab**
 - **Same storage used in MOCPy, MOCWasm and MOCGui**
 - MOC objects (Python or JS) only store a **slab** index
 - remove duplicate code (DRY principle)
 - smaller and more uniform Python and JS/WASM wrappers

MOCPy v0.12

- **Python library** to load, parse and manipulate MOCs
 - M. Baumann, T. Boch, F.-X. Pineau, M. Marchand
- Based on **PyO3**, available in both **pypi** and **conda-forge**
- Rust and Python code **completely restructured from v0.12**
 - **breaking changes** introduced (we do have users :))
 - use the MOC Lib Rust storage feature
 - fix MOC pre v2.0 / v2.0 inconsistencies
 - less Python code (contains, ...)

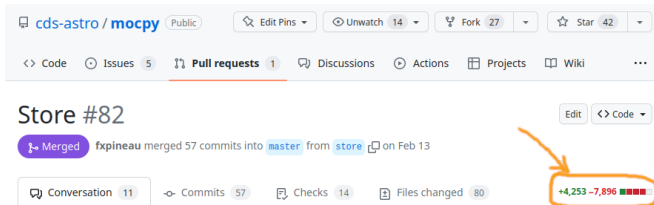


Figure 2: Large refactoring for MOCPy v0.12

□ MOCPy v0.12

- **Improvements: doc + quality checks (Manon Marchand)**
 - `git pre-commit`
 - `Black`: check code style
 - `Ruff`: Python linter (in Rust ;))
 - `nbQA`: notebook quality assurance to check notebooks
 - docsting improved following:
 - `PEP 257`
 - some of the `numpy` rules
- Experiments to run **MOCPy in web browser** (M. Marchand)
 - `Jupyter Lite`, `REPL`, `Retrolab`
 - plugged in `Pyodide` (WebAssembly port of CPython)
 - Need MOCPy `Emscripten wheels` to run in `Pyodide`
 - Try it! cds-astro.github.io/jupyterlite/lab/index.html
 - Pros
 - nothing to install locally
 - do not depends on server resources
 - multi-mode: keep client notebook, reset, live collaboration
 - Cons: not “code near data”, local resources

MOCPy v0.12.3 in JupyterLite

The screenshot shows a JupyterLite notebook interface. The browser address bar is `https://cds-astro.github.io/jupyterlite/lab/index.html`. The notebook title is `02-mocpy.ipynb`. The left sidebar shows a file explorer with a table of files:

Name	Last Modified
/	
data	23 days ago
pyodide	23 days ago
01-ipyaladi...	23 days ago
02-mocpy.i...	23 days ago
03-draw_...	23 days ago
README....	23 days ago

The main content area contains the following text and code:

Illustration of the contains function in MOCPy

Install libraries locally because they are not yet on micropip

```
[ ]: import micropip
from pathlib import Path

[ ]: whl_paths = list(Path.cwd().glob(pattern="pyodide/wheels/**"))
whl_paths

[ ]: await micropip.install("emfs:" + str(path) for path in whl_paths)
```

Imports

Libraries that are in the requirement files can imported just like in "normal" python

```
[ ]: from astropy.coordinates import SkyCoord, Latitude, Longitude
import astropy.units as u
import numpy as np
from mocpy import MOC
import matplotlib.pyplot as plt
from astropy.wcs.utils import skycoord_to_pixel
```

Our python tests

□ MOCWasm v0.7

- **MOC library in JS/WebAssembly for your web apps**
 - dual-licensing MIT/Apache-2.0 (GPLv2-compatible)
 - **100% Rust**, uses `wasm-bindgen`
- Sub-project in `moc-lib-rust`: see `crates/wasm`
- Available in github `release` section
 - `moc-wasm-v0.7.0.tar.gz`
 - main files: `moc.js`, calling `moc_bg.wasm` (~800 KB)
- **New**: published on `npm`: `@fxpineau/moc-wasm`
 - WARNING: not tested (feedback welcome)!
- **Breaking changes from v0.6 to v0.7**
 - Large `code refactoring`
 - use MOC Lib Rust storage
 - **Now resort on (T/F/ST)MOC JavaScript objects**
 - v0.6: `fromCone(name, depth, lon, lat, r)`
 - v0.7: `let mymoc = MOC.fromCone(depth, lon, lat, r)`
- **Try it!** (from your Web Browser console)
 - `http://cdsxmatch.u-strasbg.fr/lab/mocwasm/`

MOCWasm v0.7

```
🗑️ Filter Output Errors Warnings Logs Info Debug CSS
>> // Load 2MASS and SDSS DR12 MOCs from CDS
    let moc2mass = await moc.MOC.fromFitsUrl('http://alasky.u-strasbg.fr/footprints/tables/vizier/II_246_out/MOC');
    console.log("2MASS MOC depth: " + moc2mass.getDepth());
    console.log("2MASS MOC coverage: " + moc2mass.coveragePercentage() + "%");
    let mocsdss = await moc.MOC.fromFitsUrl('http://alasky.u-strasbg.fr/footprints/tables/vizier/V_147_sdss12/MOC');
    console.log("SDSS DR12 MOC depth: " + mocsdss.getDepth());
    console.log("SDSS DR12 MOC coverage: " + mocsdss.coveragePercentage() + "%");

2MASS MOC depth: 9
2MASS MOC coverage: 99.99980926513672%
SDSS DR12 MOC depth: 7
SDSS DR12 MOC coverage: 38.7420654296875%
← undefined
>> // Performs MOC intersection
    let tmass_inter_sdss12 = moc2mass.and(mocsdss);
← undefined
>> // Print the ASCII and JSON serializations of '2mass_inter_sdss12_d2'
    console.log(tmass_inter_sdss12.toAscii());


▶ 1/5 8 10 27 2/16-17 19 25 28 36 38 70 73-78 103 107 3/10 32 34 72-73 75 97 116 150 176-178 184 197 208 271
275-278 284 286-287 291 316-318 341 405-407 411 421 423 427 491 494-495 506 4/0 2 34 46-47 106 123 132-134 140 160
162-163 166 176 296-297 299 385 396-397 399 468-469 472 592 594-595 624 626 629 631-632 634 639 716 718 720 722 740
744 746 768-769 771-774 785 787 797 799 836-838 840-841 848 1078-1079 1083 1095 1099 1116-1118 1140 1142-1143 1159
1163 1276-1277 1369 1372-1373 1375 1397 1399 1431 1635 1637-1639 1641-1643 1689-1691 1701-1703 1705-1707 1877 1958
1961-1963 2018 2030-2031 2042 2299 2302 3063 5/5-7 12 27 30-32 69 71 76-78 85-88 97 99 104 106 131 140 142-143 149
151 154-156 158 177-181 183 205 207 227-228 230-231 233-234 249 252-253 256-258 260-262 264-265 267-268 292 295
312-313 391 396-398 417 419-420 422 428-429 440-441 444-445 488-489 504-505 508-509 540-542 568-569 574 576 599
605-607 627-629 644 661 668-669 672-673 676 708-709 712 723 726-727 768 770 776 898-899 909 9_

>>
```

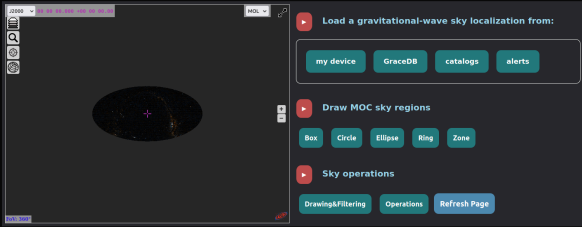
MOCWasm use case

**Gravitational-Wave Sky Localizations:
Online Calculator and Interactive Viewer of Credible Areas**

Version beta 0.6

The tool provides the credible areas of gravitational-wave sky localizations issued by the LIGO-Virgo-KAGRA collaborations (LVK). The resulting credible area is encoded with the data-structures **Multi Order Coverage map (MOC)**. MOC is a Virtual Observatory standard approved by the IVOA (International Virtual Observatory Alliance) to manage sky coverage. Each MOC is visualized in the **Aladin Lite** with various background image surveys. The whole list and the image surveys are accessible by clicking the icon  manage layers located at the top left. The MOC maps are created and manipulated with the WebAssembly library **MOCWasm**. The tool accepts the two LVK sky map formats: the **multiorder format** (with .fits extension) and the **unflattened skymap** (with .fits.gz extension). Better performances are achieved with the multiorder format.

Latest LVK Public Alert: S200316a



The screenshot shows the web interface with a central sky map displaying a localized area. To the right, there are three main control sections: 'Load a gravitational-wave sky localization from:' with buttons for 'my device', 'GraceDB', 'catalogs', and 'alerts'; 'Draw MOC sky regions:' with buttons for 'Box', 'Circle', 'Ellipse', 'Ring', and 'Zone'; and 'Sky operations:' with buttons for 'Drawing&Filtering', 'Operations', and 'Refresh Page'. The bottom of the page features logos for INFN (Istituto Nazionale di Fisica Nucleare) and UNIPG (Università degli Studi di Perugia).

Figure 3: Web page <https://virgo.pg.infn.it/maps/index.html> by Giuseppe Greco; Mateusz Bawaj; Roberto De Pietri

□ MOCGui v0.1.0-alpha

- T. Dumortier, supervised by F.-X. Pineau and M. Baumann
- Purpose: provide a **simple GUI to manipulate MOCs**
 - **both native** (to be distributed/installed)
 - and in **web browsers** (in wasm; simple URL)
- Based on [egui](#): simple immediate mode GUI library

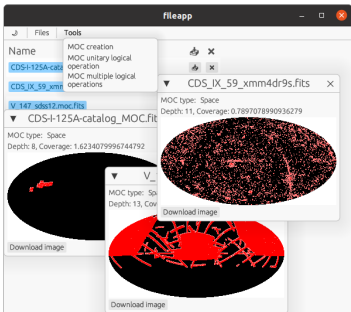


Figure 4: Native GUI on Ubuntu

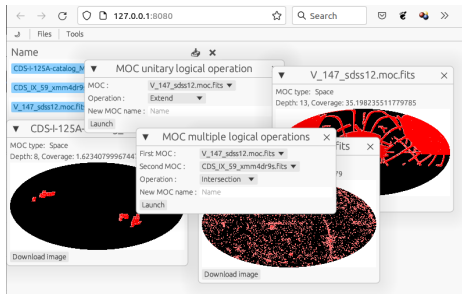


Figure 5: WASM GUI in Firefox

MOCCli and MOCSet

- 2 **command line tools** (scriptable, only 1 binary each)
 - **MOCCli**: manipulate MOCs
 - **MOCSet**: build, update and query a set of MOCs from the disk
- **Distribution**
 - pre-compiled binaries and .deb packages in [github releases](#)
 - **New!** available on [pypi](#)
 - <https://pypi.org/project/moc-cli/>
 - <https://pypi.org/project/moc-set/>

```
pineau@ocds-dev:~$ noc
noc 0.5.0
Create, manipulate and filter files using HEALPIX Multi-Order Coverage maps (MOCs).
See the man page for more information.

USAGE:
  noc <SUBCOMMAND>

FLAGS:
  -h, --help      Prints help information
  -v, --version   Prints version information

SUBCOMMANDS:
  convert  Converts an input format to the (most recent versions of) an output format
  filter   Filter file rows using a MOC
  from     Create a MOC from given parameters
  help     Prints this message or the help of the given subcommand(s)
  hprint   Print a MOC to a human readable form
  info     Prints information on the given MOC
  op       Perform operations on MOCs
  table    Prints MOC constants
  view     Save a PNG of a S-MOC and visualize it
```

```
pineau@ocds-dev:~$ mocset
moc-set 0.5.0
F.-X. Pineau <francois-xavier.pineau@astro.unistra.fr>
mocset is a command-line tool to build, update and query a persistent set of
Multi-Order Coverages maps (MOCs), see https://lvoa.net/documents/MOC/.

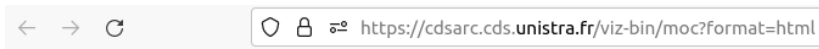
USAGE:
  mocset <SUBCOMMAND>

OPTIONS:
  -h, --help      Print help information
  -v, --version   Print version information

SUBCOMMANDS:
  append      Append the given MOCs to an existing mocset
  chgstatus   Change the status flag of the given MOCs identifiers (valid, deprecated,
              removed)
  extract     Extracts a MOC from the given noc-set
  help        Print this message or the help of the given subcommand(s)
  list        Provide the list of the MOCs in the mocset and the associated flags
  make        Make a new mocset
  purge       Purge the mocset removing physically the MOCs flagged as 'removed'
  query       Query the mocset
```

VizieR indexation by MOC

- MOCs of all VizieR catalogues
 - computed by **MOCCLi**: from coos, union of tables, ...
 - put in a **MOCSet**
- New service (by Alicia Flint and Gilles Landais)
 - Python **CGI** on top of **MOCCLi** and **MOCSet**
 - base URL: <https://cdsarc.cds.unistra.fr/viz-bin/moc>
 - get a MOC
 - **MOC of V/146**: [base URL/V/146?order=9&format=ascii](https://cdsarc.cds.unistra.fr/viz-bin/moc?format=html)
 - or a **list of MOCs**
 - **more recent than 2023-05-04**: [base URL?date=2022-10-0](https://cdsarc.cds.unistra.fr/viz-bin/moc?format=html)
 - **by cone**: [base URL?ra=2&dec=2&sr=0.01&format=html](https://cdsarc.cds.unistra.fr/viz-bin/moc?format=html)



Catalog name	MOC file	Date modified
B/assocdata	cat_102036.moc.fits	2023-02-03
B/avo.rad	cat_102013.moc.fits	2022-09-27
R/hav	cat_102017.moc.fits	2022-09-27

VizieR indexation by MOC

New way to pre-filter VizieR global queries (26K tables).

The screenshot shows the VizieR web interface. The browser address bar displays `https://vizier.cds.unistra.fr/viz-bin/VizieR?c=M1&-c.rs=10`. The page title is "VizieR". The main content area shows search results for "Crab" pulsars. The results are displayed in a table with columns: Full, name, RA2000, DE2000, title, and text. The first three rows of results are:

Full	name	RA2000	DE2000	title	text
1	J/ApJS/157/324	05 34 32.0	+22 00 52	EGRET high-energy cosmic photons (E>10GeV) (Thompson+, 2005)	[AladinJava] Crab = M1
2	J/A+A/640/A43	05 34 31.9	+22 00 52	10-year Fermi LAT results for the Crab pulsar (Yeung, 2020)	[AladinJava] Crab pulsar = V* CM Tau
3	J/A+A/565/L12	05 34 31.9	+22 00 52	Crab pulsar 50-400GeV light curve (Aleksic+, 2014)	[AladinJava] Crab pulsar = V* CM Tau - (gamma-ray (50-400GeV) light curve)

Below the first table, there is a note: "Note: The Naval Observatory Merged Astrometric Dataset (NOMAD) contains astrometric and photometric data for over 1 billion stars derived from the Hipparcos (I/239), Tycho-2 (I/259), UCAC2 (I/289), and USNO-B1.0 (I/284) catalogs for astrometry and optical photometry, supplemented by 2MASS (I/244) near-infrared photometry. An efficient remote query program `findnomad1` is available in the `cdscient` package, for Unix/Linux platforms".

Below the note, there is a section for "I/297/out" with the title "NOMAD Catalog (Zacharias+ 2005)" and "The NOMAD-1 Catalog (1117612732 rows)". To the right of this section are links for "2004AAS...205.4815Z" and "ReadMe+ftp".

Below the second section, there is a table with columns: Full, NOMAD1, YM, RAJ2000, DEJ2000, r, pmRA, e, pmDE, e, Bmag, r, Ymag, r, Rmag, r, Imag, Hmag, Kmag, R. The first three rows of data are:

Full	NOMAD1	YM	RAJ2000	DEJ2000	r	pmRA	e	pmDE	e	Bmag	r	Ymag	r	Rmag	r	Imag	Hmag	Kmag	R
1	1120-0089111	M	083.6328569	+22.0127939	M	0.0	0.0	0.0	0.0							16.655	15.778	16.510	
2	1120-0089112	M	083.6331069	+22.0144858	M	0.0	0.0	0.0	0.0							14.576	14.054	13.500	
3	1120-0089116	M	083.6327764	+22.0118322	V	15.6	0.0	63.7	0.0			7.26	V			16.263	14.801	14.764	

Figure 6: Look for all sources in VizieR at 10 arcsec around M1

□ MOCSet/Py in AMORA

AMORA: Asynchronous Multi Observation Region-based Analysis

- Purpose: run XMM reduction tasks on regions drawn by the user on Aladin Lite
 - MOCSet use to select observations matching the region
 - 3 x 14 000 Obs
 - by cone
 - by polygon
 - (+ SQL lite to store obs. metadata)

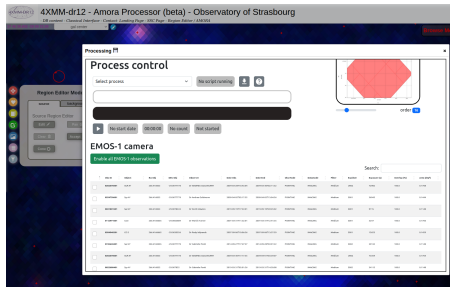
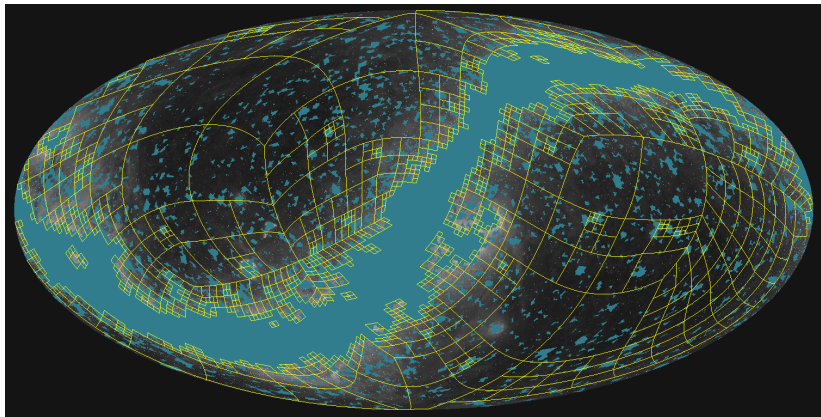


Figure 7: AMORA Web page

By XMM team in Strasbourg: ask **Laurent Michel** for more details

Aladin Lite v3

- MOC Lib Rust in Aladin Lite v3 (Matthieu Baumann)
 - load MOCs (FITS/JSON/ASCII), iterate on cells
 - MOC from polygon / cone
 - MOC intersection



□ MOC Lib Rust

New functionalities apart from the **storage** feature

□ New: F-MOCs

- S-MOC index:
 - 12 base cells (4 bits b)
 - plus 2-D z-order curve of depth 29 in each base cell $((i, j))$
 - $ipix_{29} = 00$

$b_3 b_2 b_1 b_0$	$j_1 i_1 j_2 i_2 \dots j_{29} i_{29}$
-------------------	---------------------------------------
- T-MOC index
 - microsec since $JD = 0, \in [0, 2^{62}[$ (1-D Z-order curve)
 - $ipix_{61} = 00$

$t_0 t_1 t_2 t_3 \dots t_{61}$

- **F-MOC** (see also P. Fernique talk in App 1)
 - **simply re-encode f64 Hz value**
 - regular f64:

s	$e_{10} \dots e_0$	$f_{51} \dots f_0$
-----	--------------------	--------------------

 - $v = (-1)^s (1.f_{51} \dots f_0)_2 \times 2^{e-1023}$
 - 1 sign bit (not needed!)
 - 11 exponent bits (smaller range ok)
 - 52 fraction bits (keep full precision)
 - $ipix_{59} = 0000$

$e_7 \dots e_0$	$f_{51} \dots f_0$
-----------------	--------------------

 - **nice properties** (order, ...) \Rightarrow operations on int64 representation \Leftrightarrow **operations on f64 Hz ranges**

□ New: F-MOCs

- For F-MOC order $\in [7, 53]$:
 - relative precision in Hz: $\epsilon \in [1/2^{1+order-7}, 1/2^{order-7}]$
 - number of significant digits: $n \approx \ln_{10} \frac{1}{prec}$
 - at order 59: $\epsilon \in [1.1 \times 10^{-16}, 2.2 \times 10^{-16}]$, $n \approx 16$
- **New!** Human print option **hprint** in MOCCLI

```
> echo "25/3047" | moc hprint -f ascii -t fmoc -  
# Frequency ranges in Hz  
from_inclusive,to_exclusive  
5.114326226619749e-29,5.114345485919193e-29
```

```
> echo "150000000.0" | \  
> moc from freqval 59 - ascii | \  
> moc hprint -f ascii -t fmoc -  
# Frequency ranges in Hz  
from_inclusive,to_exclusive  
1.5e8,1.50000000000000003e8
```

□ New: quick view (allsky)

- MOCCLI: `> moc view moc.fits moc.png allsky 600`
- MOCpy: `MOC.display_preview(y_size=600)`
- MOCWasm: TBD (ask if you need it)

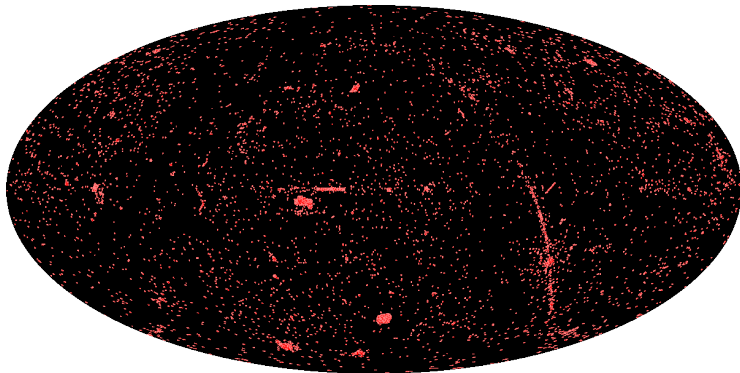


Figure 8: XMM MOC in 1200x600

□ New: quick view algo

- Algo different from Aladin/Aladin Lite
 - not real time \Rightarrow not the same perf. needs
- Algo details:
 - for each image pixel
 - de-project to find sky coordinates
 - compute HEALPix cell
 - switch on (100% opacity) pixel if HEALPix cell is in MOC
 - for each MOC cell
 - compute the projected coordinate of the cell center
 - switch on (50% opacity) the image pixel
- Why?
 - support case of sparse MOCs made of cells smaller than image pixel
 - better visualization of MOC borders
 - performances are ok ($\sim 1s$ for 1600x800 image preview)

□ New: custom quick view

- Support various projections
 - `mapproj` Rust library, also used in Aladin Lite v3
- Implemented in *MOC Lib Rust* and *MOCCLI*
 - ```
> moc view moc.fits moc.png custom -l 80.0 -b -45.0 sfl 1600 800
```

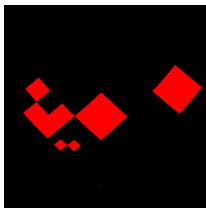
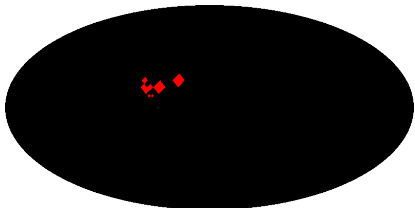


# □ New: autodetect MOC FOV

- MOC “barycenter” and “radius” (nothing fancy)
  - barycenter: mean cell center ponderated by cell area
  - radius: distance from center to furthest vertex
  - motivation: allow viewers to center the FOV around a MOC
- MOCCLI: `> moc view moc.fits moc.png auto 800`
- MOCpy (feature requested by Laurent Michel):
  - `MOC.barycenter()`
  - `MOC.largest_distance_from_coo_to_vertices(coo)`

```
> moc view moc.fits moc.png allsky 400
```

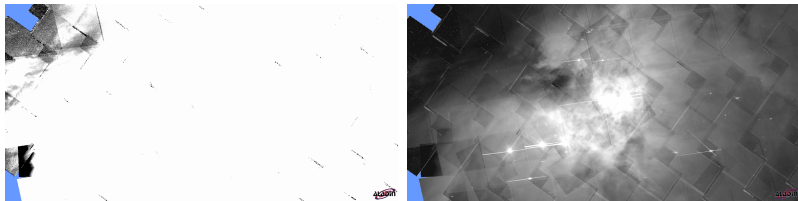
```
> moc view moc.fits moc.png auto 400
```





# □ New use case for split

- **Split** a MOC into disjoint sub-MOCs
  - feature request by **G. Greco** (grav. waves credible regions)
- New **use case for HiPS**, see T. Boch talk (App 1)
  - split a HiPS into independent regions
  - apply one cut by region (instead of a global cut)
  - also implemented in **HiPS gen Java now** (P. Fernique)



# Summary

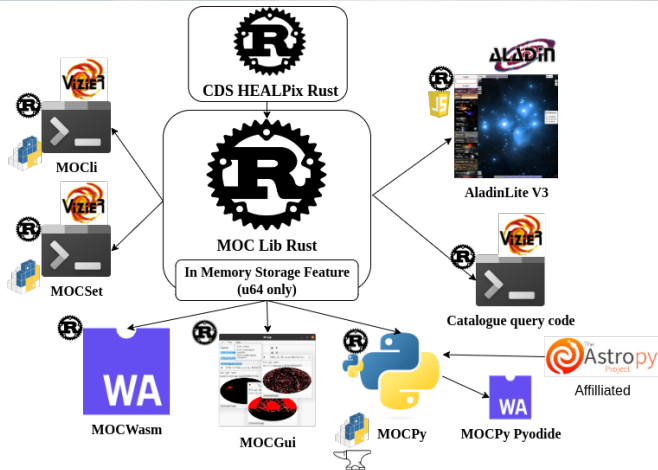


Figure 9: one Rust core lib (still evolving) + thin wrappers: many benefits (it is fast!)