



# Simulations et possibilités observationnelles de Gaia

Carine Babusiaux

GEPI - Observatoire de Paris



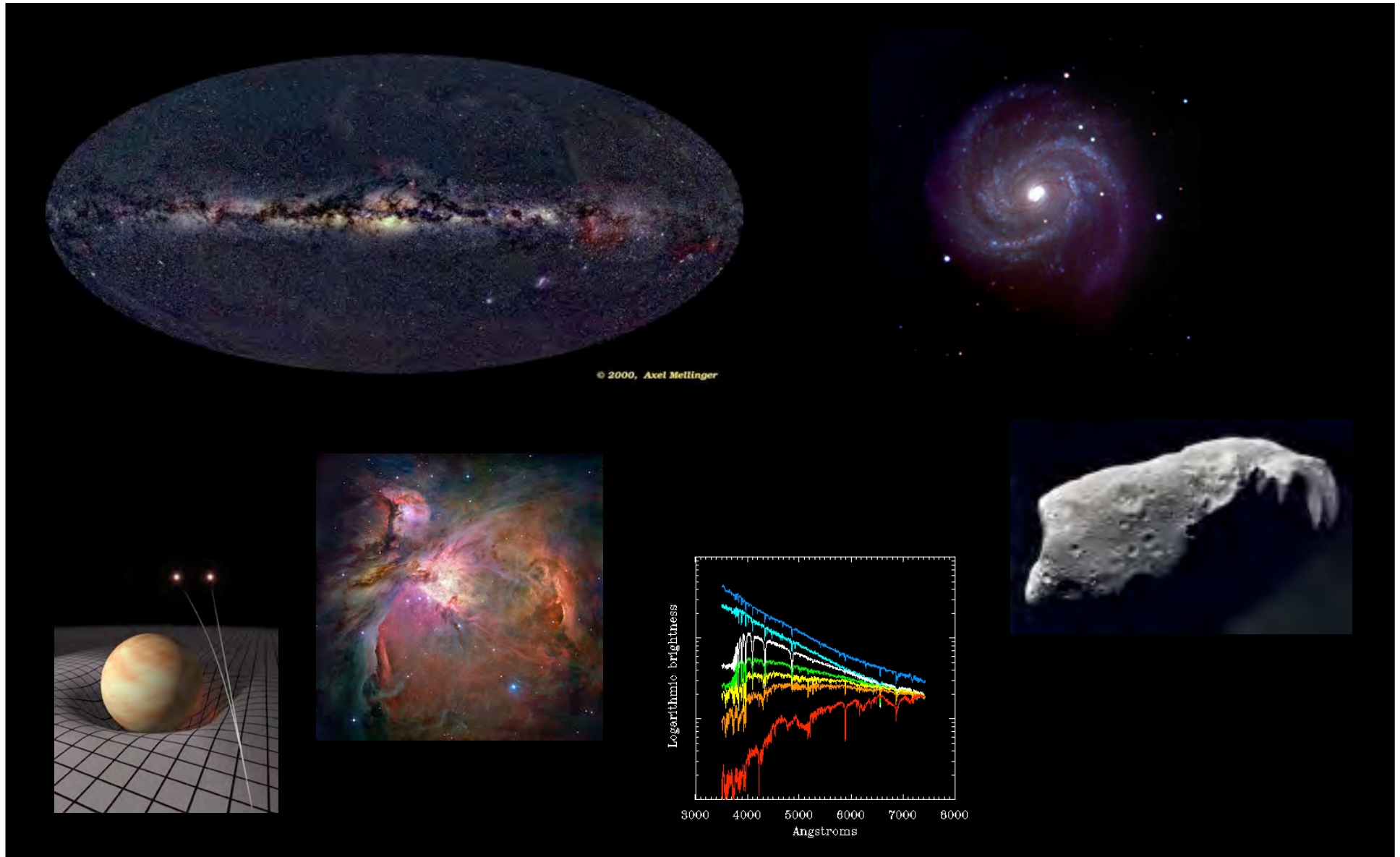
# Pourquoi des simulations ?

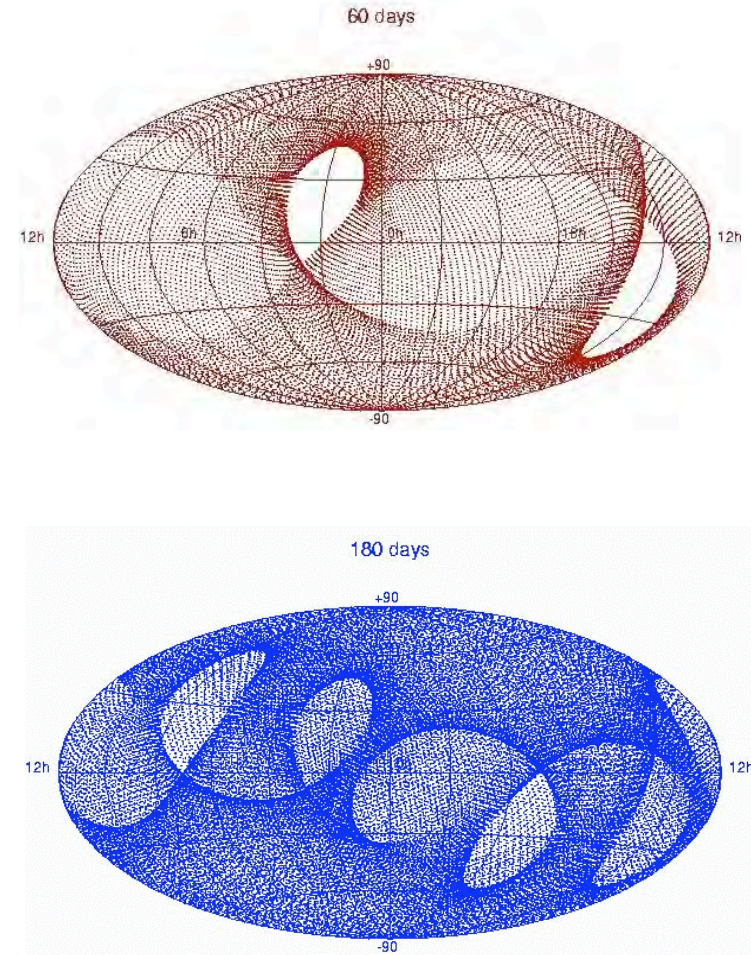
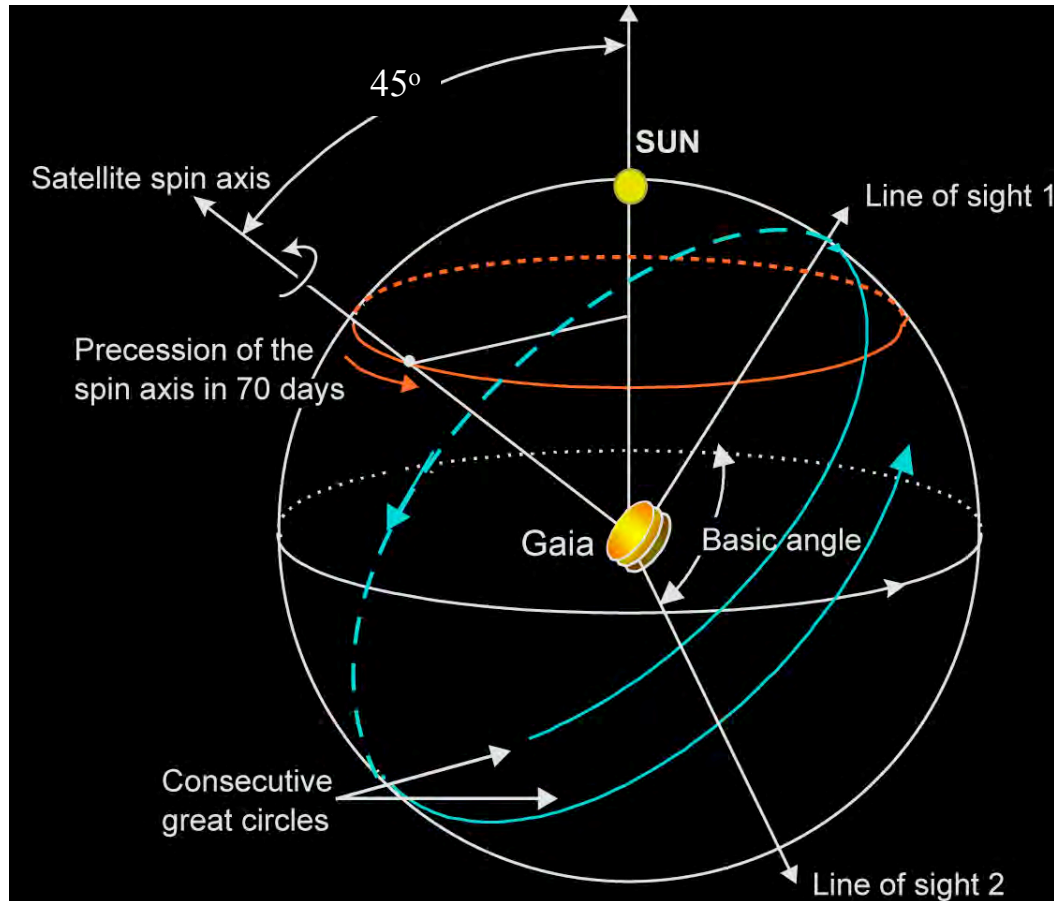
---

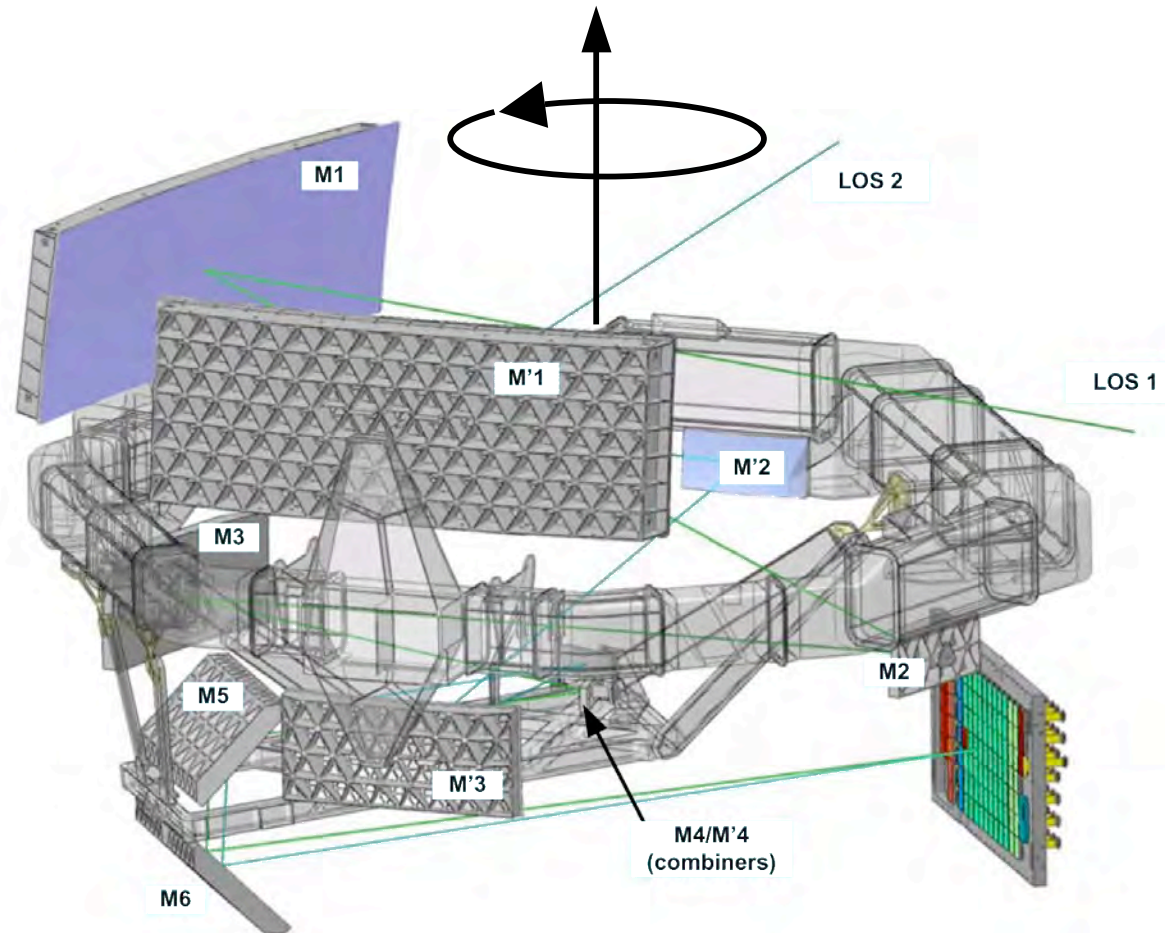
- Préparer la réduction de données
  - Mettre en place l'architecture du traitement des données
  - Développer les algorithmes
  - Vérifier les algorithmes et les modèles
  - Estimer les performances
- Pendant opération
  - Vérification des modèles scientifiques sur les données réelles

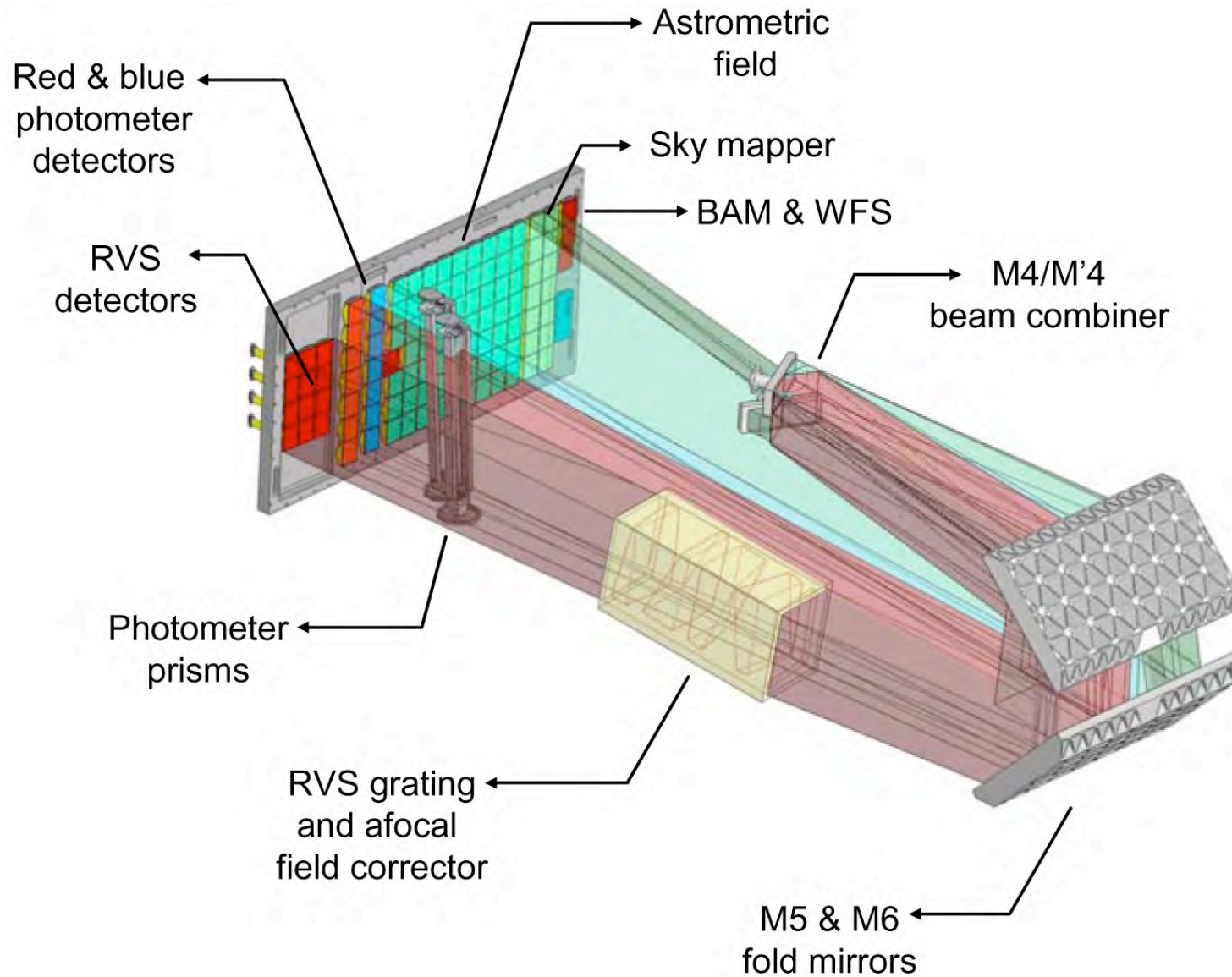


# L'univers visible par Gaia ⇒



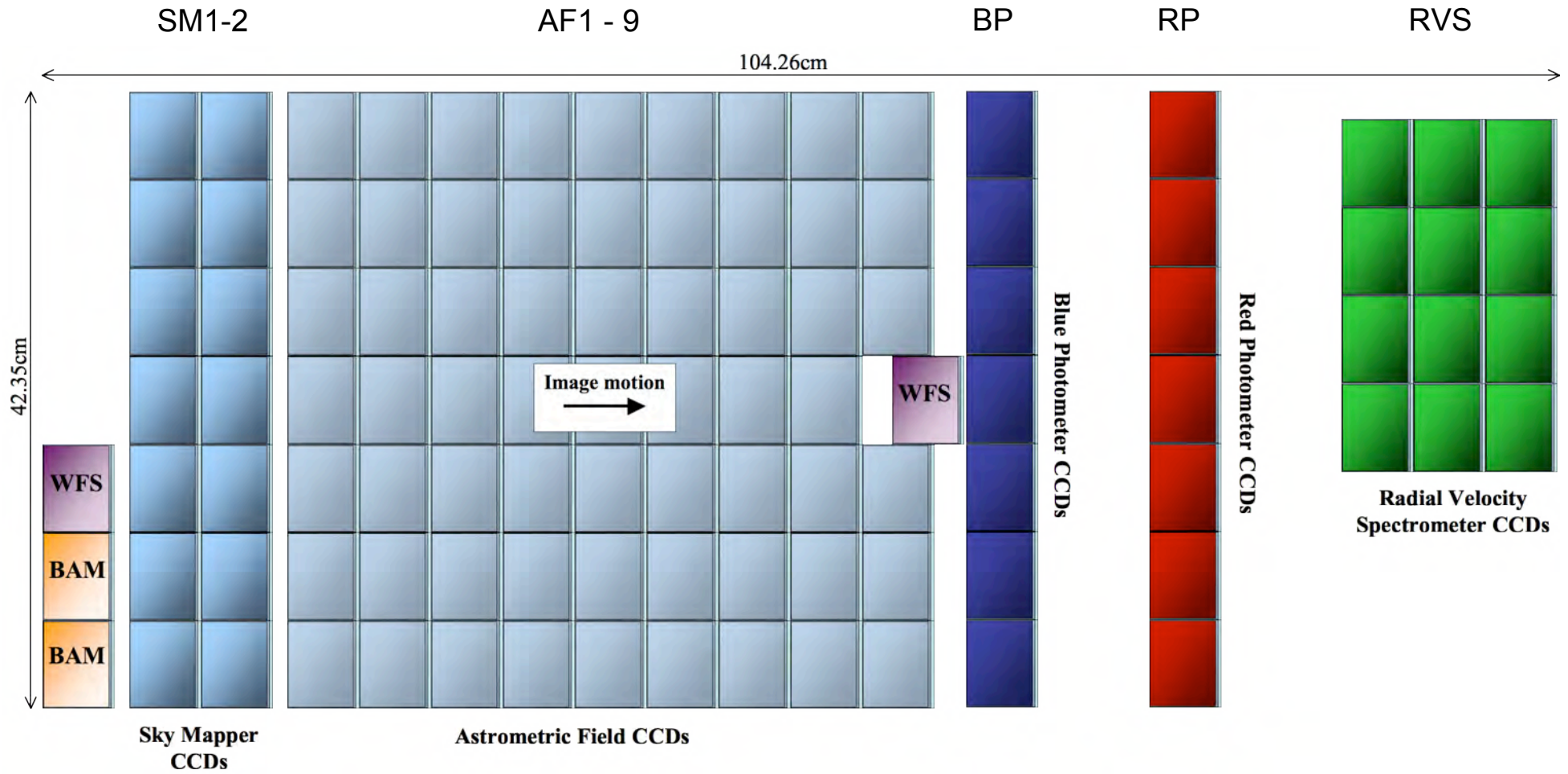






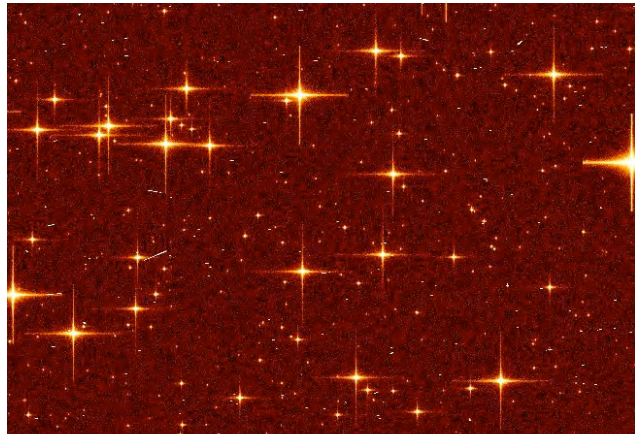


⇒ plan focal ⇒

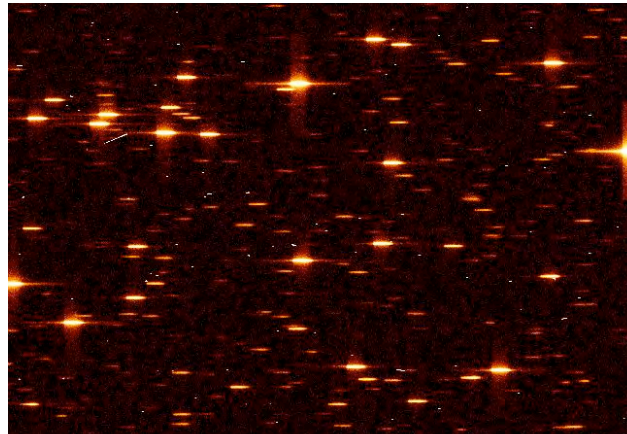




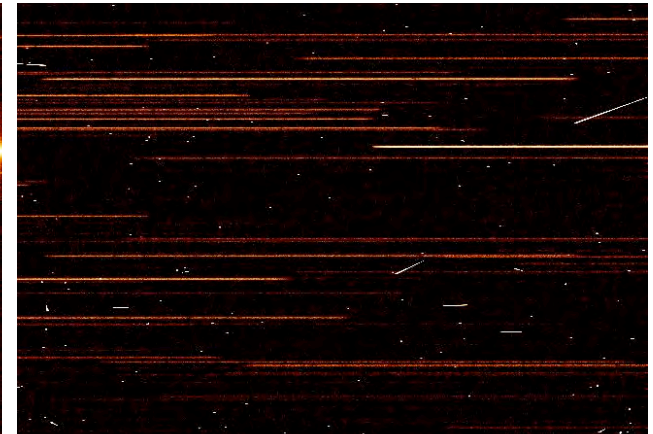
# ⇒ Collection des photons ⇒



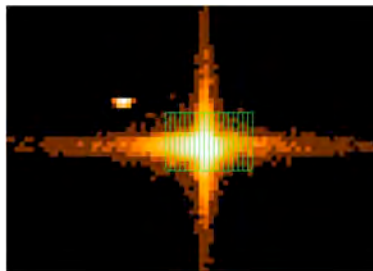
AF



RP



RVS



détection et fenêtrage à bord



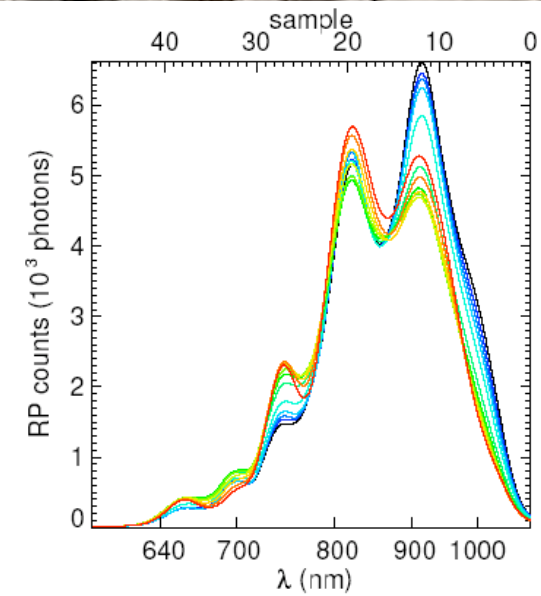
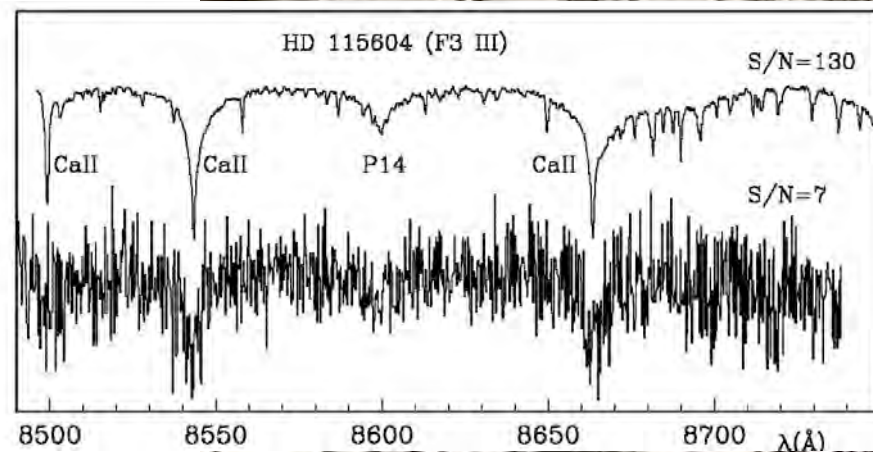
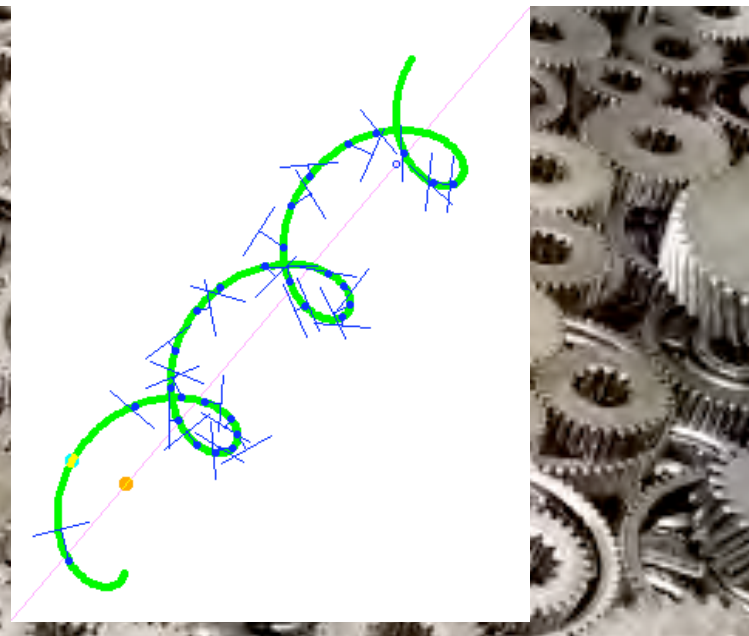
319255
1486 19264 78012 15508 1269
1781 23164 93825 18624 1553
111 1624 6603 1302 110
5 226 841 169 23
1 37 134 24 4
3703 14461 63430 106228 64261

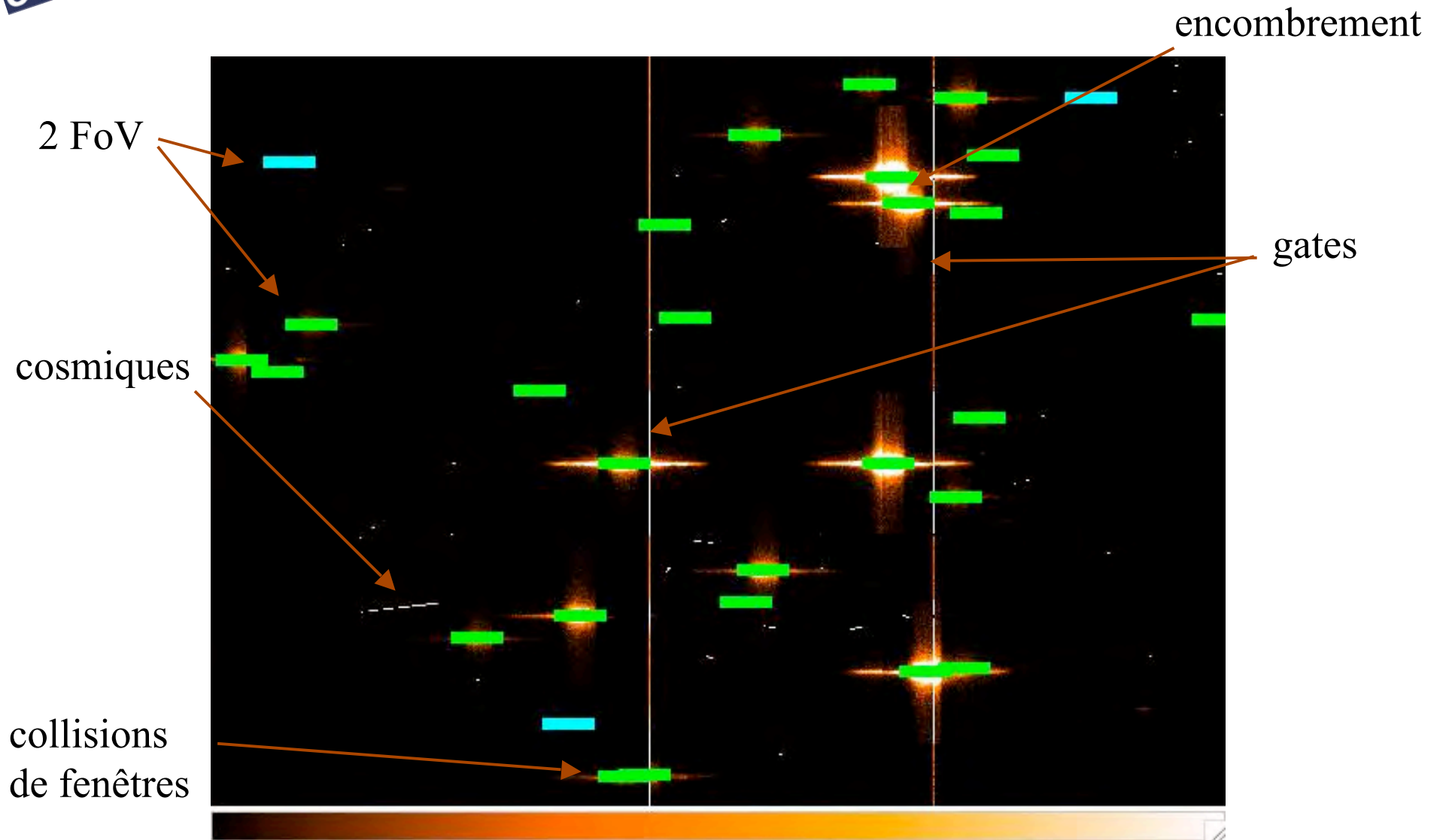
données envoyées au sol





⇒ DPAC !

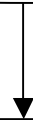
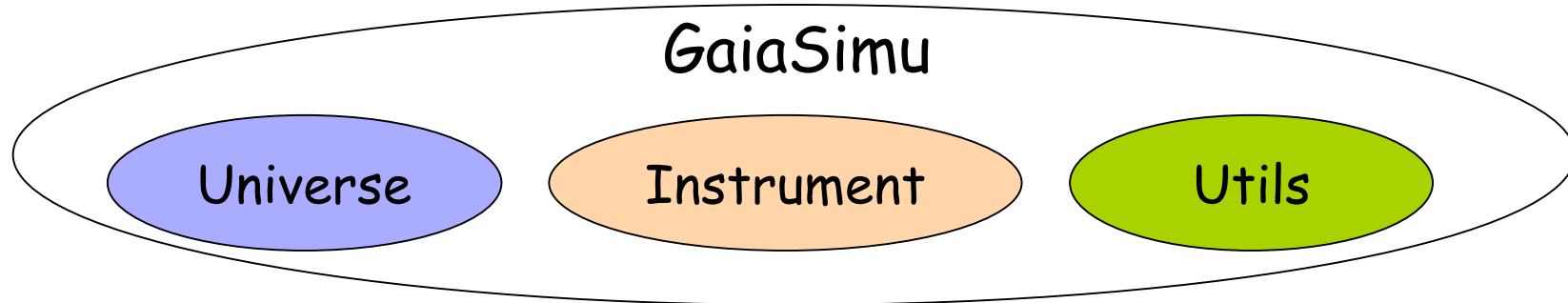




RP observation de l'amas ouvert ngc6231



# Le Simulateur Gaia



<b>GASS</b>	<b>GIBIS</b>	<b>GOG</b>
Telemetry	Pixel-level data	Gaia catalogue
<pre>319255 1486 19264 78012 15508 1269 1781 23164 93825 18624 1553 111 1624 6603 1302 110 5 226 841 169 23 1 37 134 24 4 3703 14461 63430 106228 64261</pre>		



# La participation française...

---

28 membres actifs (37%)

(co)-management de 6/8 DUs:

- DU1: **Management**
- DU2: **Software**
- DU3: **Modèle d'Univers**
- DU4: **Modèle d'Instrument**
- DU5: **GASS**
- DU6: **GIBIS**
- DU7: **GOG**
- DU8: **Validation**

X. Luri, **C. Babusiaux** (*Barcelone/Paris*)

**J.M. Wallut** (*CNES*)

**A. Robin** (*Besançon*)

D. Gardiol (*Turin*)

E. Masana (*Barcelone*)

**C. Babusiaux** (*Paris*)

X. Luri, **C. Babusiaux** (*Barcelone/Paris*)

**D. Egret** (*Paris*)

1/3 coordinateur technique:

- **Astrometry**
- **Photometry**
- **Spectroscopy**

**C. Fabricius** (*Barcelone*)

**C. Jordi** (*Barcelone*)

**P. Sartoretti** (*Paris*)



- Coordinateur technique
- Déploiement des interfaces web de GIBIS et GOG au CNES

**Gaia DPAC** **GIBIS**  
Gaia Instrument and Basic Image Simulator - v3.2

Home Run Monitoring Docs GIBIS-Wiki History Help Issues

**Application**

**GIBIS** is a pixel-level simulator of the **Gaia** mission. It is intended to simulate how the Gaia instruments will observe the sky, using realistic simulations of the astronomical sources and of the instrumental properties. It is a branch of the global **Gaia Simulator** under development within the **Gaia DPAC CU2 Group (Data Simulations)**.

**Run GIBIS simulation**

**Run GIBIS Simulation**

You must have an account to run GIBIS and you can subscribe [here](#).

You can follow up your simulations [here](#)

**Related links**

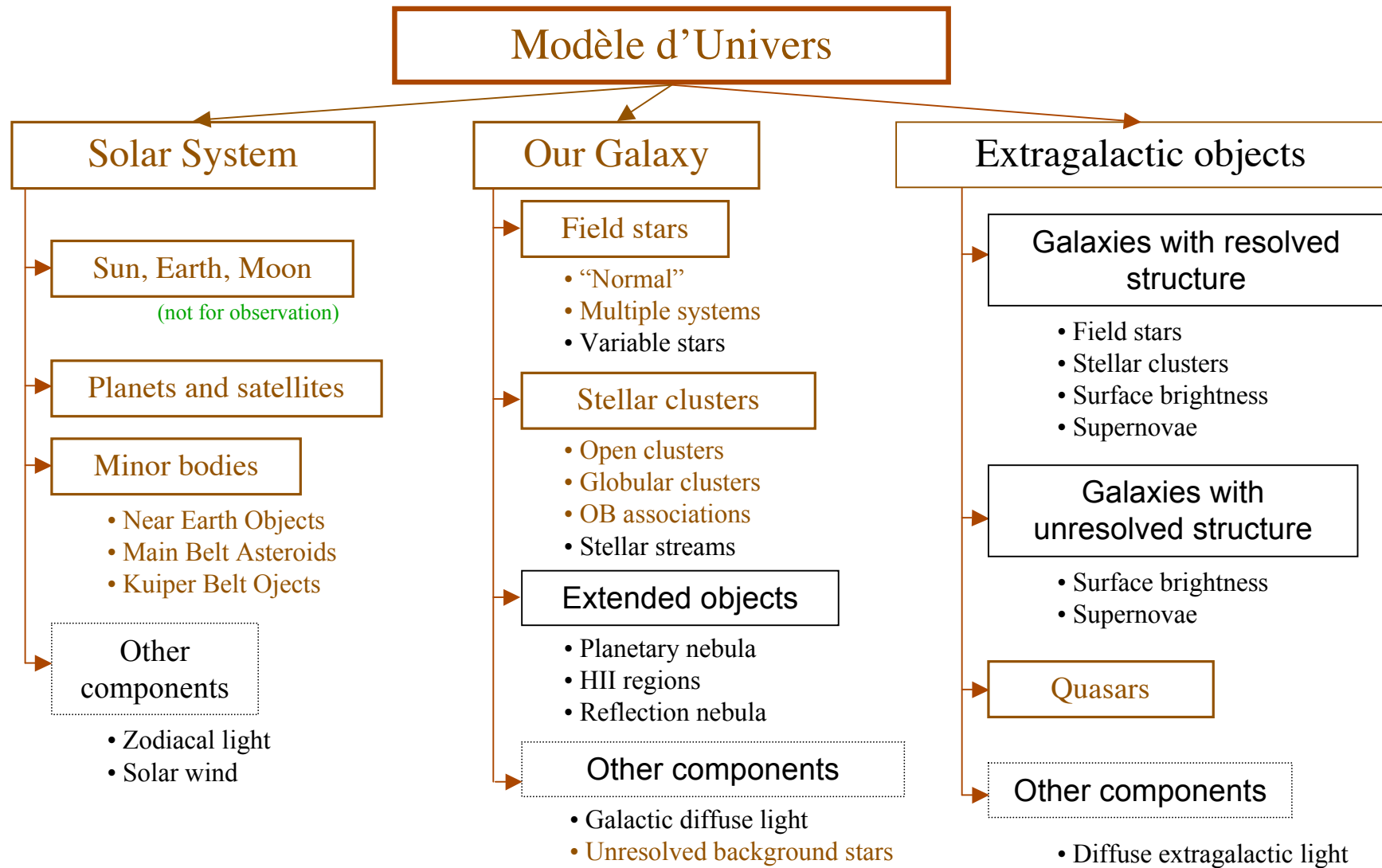
- [GAIA home Page](#)
- [CU2 Wiki Page](#)
- [SWG Home Page](#)

GIBIS v3.2  
(Last update : 05/10/2007)

Hosted by **cnes**  
CENTRE NATIONAL D'ETUDES SPATIALES

<http://gibis.cnes.fr/>

GOG soon on-line



[http://www.rssd.esa.int/SA-general/Projects/GAIA/wiki/index.php?title=CU2:\\_Data\\_Simulations](http://www.rssd.esa.int/SA-general/Projects/GAIA/wiki/index.php?title=CU2:_Data_Simulations)

```
// New time intervals for the NFIs
t1= NF1*(time.getGMTtime()/NF1);
t2= t1 + NF1;
t3= t2 + NF1;

// Arrays to build the splines
double[] t= new double[nPoints/4 +1];
double[] n1= new double[nPoints/4 +1];
double[] n2= new double[nPoints/4 +1];
double[] n3= new double[nPoints/4 +1];

// First spline

// Generate the three Markov chains
double[] c1 = generateNF1Chain(activeNF1.NF1, NoiseChain.x);
double[] c2 = generateNF1Chain(activeNF1.NF1, NoiseChain.y);
double[] c3 = generateNF1Chain(activeNF1.NF1, NoiseChain.z);

// Select one point in four to fill the splines
for(int i=0; i<nPoints+1; i=i+4){
    t[i/4] = (double)( t1 + i*tStep );
    n1[i/4]= 0.5 * c1[i];
    n2[i/4] = 0.5 * c2[i];
    n3[i/4] = 0.5 * c3[i];
    System.out.println("t[" + i/4 + "] = " + t[i/4] + " n1[" + i/4 + "] = " + n1[i/4] + " n2[" + i/4 + "] = " + n2[i/4] + " n3[" + i/4 + "] = " + n3[i/4]);
}

s1n1.setSample(t,n1);
s1n2.setSample(t,n2);
s1n3.setSample(t,n3);

// Second spline

// Generate the three Markov chains
c1 = generateNF1Chain(activeNF1.NF2, NoiseChain.x);
c2 = generateNF1Chain(activeNF1.NF2, NoiseChain.y);
c3 = generateNF1Chain(activeNF1.NF2, NoiseChain.z);

// Select one point in four to fill the splines
for(int i=0; i<nPoints+1; i=i+4){
    t[i/4] = (double)( t2 + i*tStep );
    n1[i/4] = 0.5 * c1[i];
    n2[i/4] = 0.5 * c2[i];
    n3[i/4] = 0.5 * c3[i];
}

s2n1.setSample(t,n1);
s2n2.setSample(t,n2);
s2n3.setSample(t,n3);
```