The background is a dark blue gradient. In the corners, there are various colorful geometric shapes: triangles, circles, and lines. Some are solid colors like yellow, orange, and pink, while others are dashed or dotted lines. A large blue triangle and a large yellow circle overlap in the center, serving as a backdrop for the title.

AMUSE: Taking python to the stars

BY: MARIA CAMILA
REMOLINA-GUTIÉRREZ



Hello!

I am **Maria Camila**

Physicist. Undergrad senior of Systems & Computing Engineering @ Uniandes, CO

GitHub: [@mariacamilaremolinagutierrez](https://github.com/mariacamilaremolinagutierrez)

LinkedIn: [/in/mariacamilaremolinagutierrez](https://www.linkedin.com/in/mariacamilaremolinagutierrez)



1

What is AMUSE?







AMUSE

**Astrophysical
MULTipurpose
Software
Environment**

What is AMUSE?



- ✓ Software framework for astrophysical simulations.
 - ✓ Existing codes from different domains can be easily coupled.
 - ✓ Trustworthy incorporated modules.
 - ✓ Python interface.
 - ✓ Open source (under GNU).
- 
- 

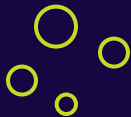
4 Main Domains



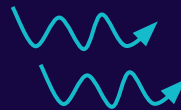
Stellar Evolution



Hydrodynamics



Gravitational Dynamics



Radiative Transport

AMUSE Objective



“To do science, to explore scientific questions, to seek new boundaries and explore new physics by means of simulation. [...] To enjoy the joyous parts of computational astrophysics, to worry about the non-linear couplings between physical domains without having to worry too much about the details of the energy conservation in a close three-body interaction.”

- **Simon Portegies Zwart**





2

History of AMUSE

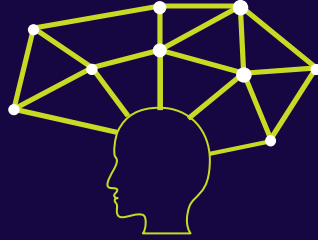
How was it born?



History of AMUSE



Island of Capri,
ITALY



WHY REDESIGN CODE?

Why should we rewrite any software if we have the perfect tools at hand?

History of AMUSE



THE NETHERLANDS

History of AMUSE



Workshop in Amsterdam with several computational astrophysicists.





A PhD Student that attended was a python fanatic.



First framework version called MUSE (MultiPurpose Software Environment)



Creation of a team, hiring of software engineers and scientific programmers => AMUSE





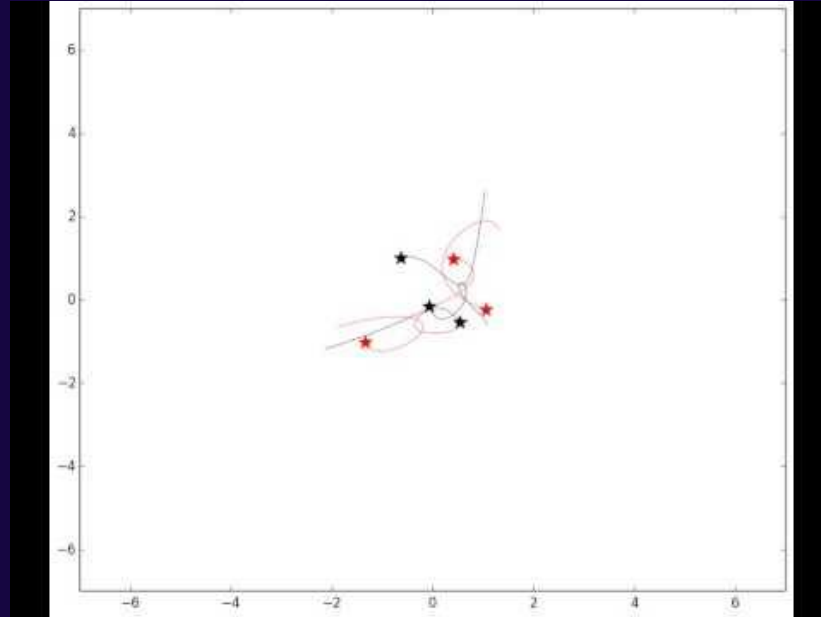
3

What can you do with AMUSE?

Examples of its use



a) Pythagorean Problem



Red = Hermite & Black = Brutus. [CODE](#)

<https://arxiv.org/abs/1402.6713>

Short Code

'''

Calculates the Pythagorean 3-body problem using different values for the smoothing length in the n-body code.

'''

```
import numpy
import time
```

```
from amuse.community.phigRAPE.interface import PhiGRAPE
from amuse.community.hermite0.interface import Hermite
from amuse.community.huayno.interface import Huayno
from amuse.community.bhtree.interface import BHTree
from amuse.units import nbody_system
from amuse.units.quantities import AdaptingVectorQuantity
from matplotlib import pyplot
from amuse.datamodel import Particles
```

```
def new_particles():
    particles = Particles(3)

    particles.mass=[3.,4.,5.] | nbody_system.mass
    particles.position = [
        [1, 3, 0],
        [-2, -1, 0],
        [1, -1, 0],
    ] | nbody_system.length

    particles.velocity = [0.,0.,0.] | nbody_system.speed
    particles.radius = 0 | nbody_system.length

    return particles
```

Short Code



```
def run_pyth(interface,tend=100,dt=0.125,parameters=[]):  
  
    code = interface()  
  
    for name,value in parameters:  
        setattr(code.parameters, name, value)  
  
    code.particles.add_particles(new_particles())  
    code.commit_particles()  
  
    x = AdaptingVectorQuantity()  
    y = AdaptingVectorQuantity()  
    t=0. | nbody_system.time  
    while(t < tend-dt/2):  
        t=t+dt  
        code.evolve_model(t)  
        x.append(code.particles.x)  
        y.append(code.particles.y)  
    code.stop()  
  
    return x,y
```

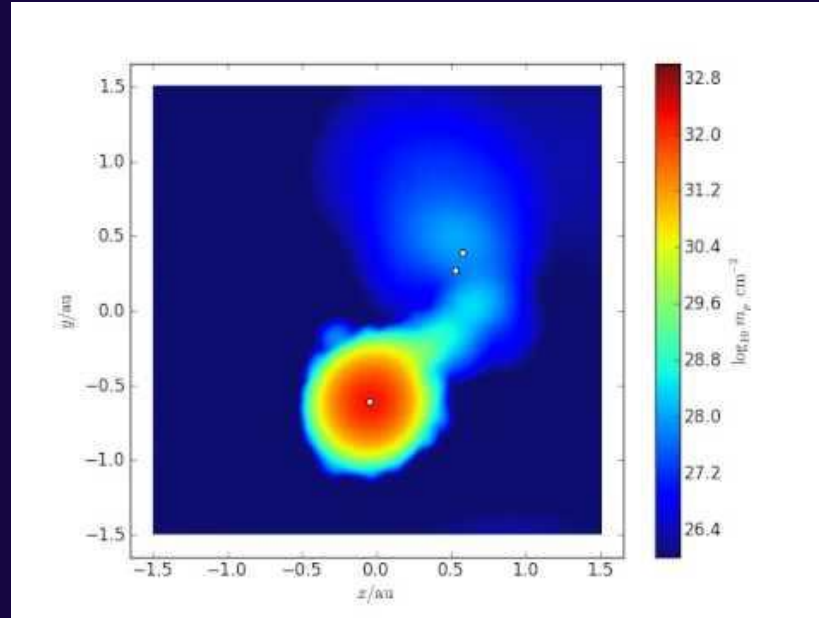

Short Code

```
if __name__ == "__main__":
    codes_to_run=[ ('Hermite0', '$\eta=0.03$', Hermite, [ ("dt_param",0.03)] ),
                   ('Hermite0', '$\eta=0.01$', Hermite, [ ("dt_param",0.01)] ),
                   ('Hermite0', '$\eta=0.003$', Hermite, [ ("dt_param",0.003)] ),
                   ('Hermite0', '$\eta=0.001$', Hermite, [ ("dt_param",0.001)] ) ]

    N=(len(codes_to_run)-1)/2+1
    f=pyplot.figure(figsize=(8,4*N))

    for i,(label,interface,parameters) in enumerate(codes_to_run):
        x,y=run_pyth(interface,tend=100 | nbody_system.time ,dt=0.0625 | nbody_system.time,parameters=parameters)
        x = x.value_in(nbody_system.length)
        y = y.value_in(nbody_system.length)
        subplot=f.add_subplot(N,2,i+1)
        subplot.plot(x[:,0],y[:,0], 'r')
        subplot.plot(x[:,1],y[:,1], 'b')
        subplot.plot(x[:,2],y[:,2], 'g')
        subplot.set_title(label)
        subplot.set_xlim(-8,8)
        subplot.set_ylim(-6,6)
    pyplot.show()
```

b) Triple Star in Xi Taurus



Taurus constellation. [CODE](#)

<https://arxiv.org/abs/1309.1475>



4

Structure of AMUSE

How is it built?



2 Top Level Components



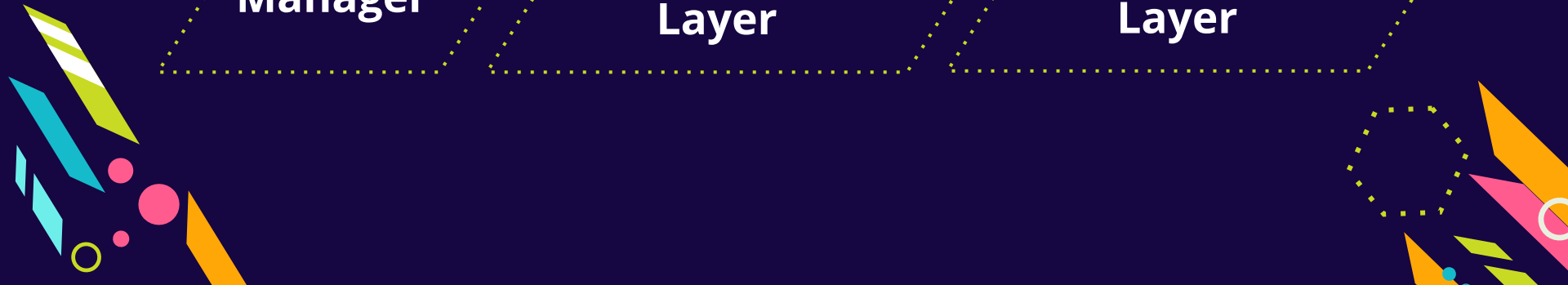
User Script (Python)

Community Module

Manager

**Communication
Layer**

**Community Code
Layer**



2 Top Level Components



User Script (Python)

- Implements a specific physical problem or set of problems, in the form of system-provided or user-written scripts.
- Serves as the user interface to the AMUSE framework.
- Implements the coupling between community codes with the help of support classes provided by the manager.

2 Top Level Components



- Suite of system-provided utility functions.
- Object-oriented interface onto the communication layer.
- Handles **unit conversion**.
- Contains the **state engine** and the associated **data repository**, to guarantee consistency of data across modules.
- Handles exceptions.

2 Top Level Components

Community Module

Communication Layer

- Bi-directional communication between the manager and the community code layer.
- Uses MPI (Message Passing Interface) to communicate layers.
- Implemented via a **proxy (converts python cmds to MPI msgs)** and an associated **partner (decodes MPI msgs into community code cmds)**.

2 Top Level Components



Community Module

Communication Layer

- MPI is widespread accepted in the computational science community and has broad support by many programming languages and computing platforms.
- AMUSE naturally accommodates inherently parallel community modules and allows simultaneous execution of independent modules from the user script.

2 Top Level Components

Community Module

Community Code Layer

- Contains the community codes .
- Implements control and data management operations on the codes.
- Each piece of code in this layer is domain-specific, and general within its particular physical domain.
- Accepts any computer language that supports smart sockets or bindings to the MPI protocol. Ex: C, C++, Java, FORTRAN, Python.



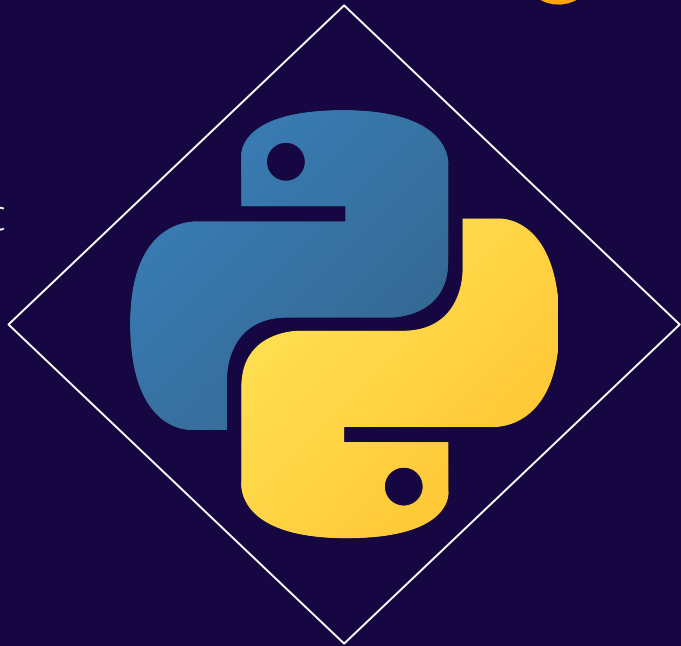
5

Why Python?



Python:

- is broadly accepted and used in the scientific community.
- has an object oriented design.
- allows rapid prototyping, shortening the software development cycle.
- enables easy access to the community code.



A person is camping in a dark, rocky landscape under a starry night sky. A white tent is illuminated from within, casting a warm glow. A person in a yellow jacket stands near a body of water in the background. The sky is filled with stars and a faint Milky Way. The text "Let bits help us To explore our Cosmos" is overlaid on the image.

**Let bits help us
To explore our
Cosmos**



Thanks!

Any questions?

You can find me at:

mariacamilaremolinagutierrez@gmail.com



References



- Amuse Website amusecode.org
- Computational Astrophysics with the Astronomical Multipurpose Software Environment by [Simon Portegies Zwart](#)
- Presentation template designed by [Slidesmash](#)