

# GPUUnit

Gabriel Schwartz

Tim McJilton

Andrew Sherman

Rajkumar Jayachandran

Daniel Bagnell

Jason Economou

Advisor: Prof. Jeremy Johnson

Stakeholders: Prof. Steve McMillan

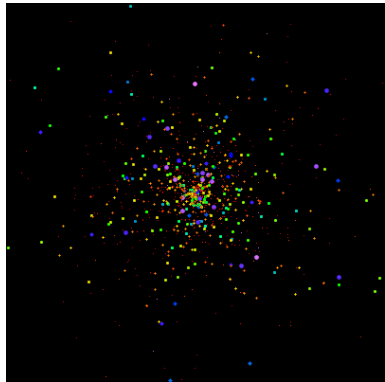
Alfred Whitehead

The Leiden Observatory

May 16, 2011

# Motivation

- ▶ Astrophysics researchers need to simulate movement and evolution of star clusters and galaxies.
- ▶ Every star pulls on all of the others.
  - ▶  $O(n^2)$
- ▶ Complex software exists to perform these computations efficiently.
- ▶ Our project exists to make running experiments with these tools feasible for a wider audience.



# Overview

Introduction

Purpose

Purpose of GPUUnit

Target Audiences

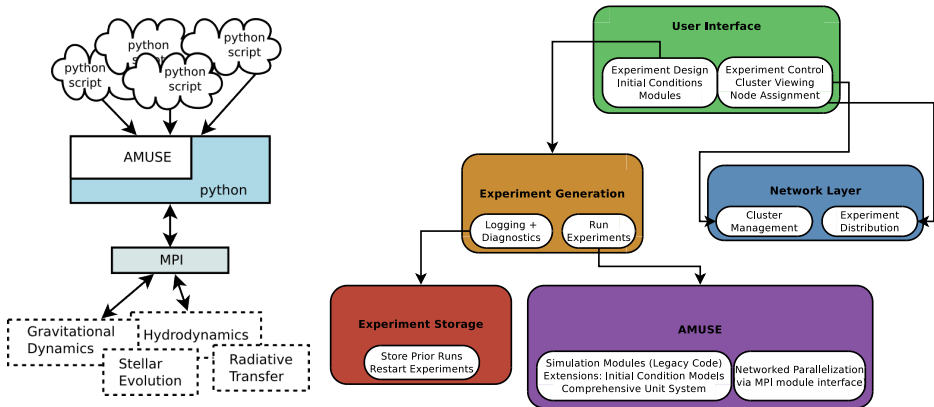
Features and Design

Software Engineering

Impact

Demo

# Astrophysical Multipurpose Software Environment (AMUSE)



<http://www.amusecode.org>

# State of AMUSE

- ▶ Currently used by researchers to run large-scale simulations.
- ▶ Scripts, diagnostics, logging are all written by hand.
- ▶ AMUSE API/programming knowledge is required to create experiments.
- ▶ Still better than separated and opaque FORTRAN codes.

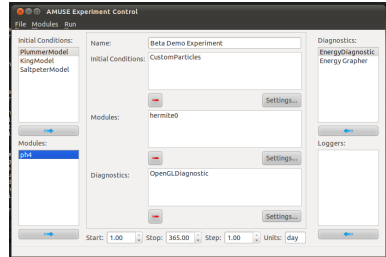
```

first_try = True
model_number = get_model_number(AMUSE_id, ierr)
if (evolve_failed('get_model_number', ierr, evolve, -3)) return
step_loop: do ! may need to repeat this loop for retry or backup
result = star_evolve_step(AMUSE_id, first_try)
if (result == keep_going) result = check_model(s, AMUSE_id, 0)
if (result == keep_going) result = star_pick_next_timestep(AMUSE_id)
if (result == keep_going) exit step_loop
model_number = get_model_number(AMUSE_id, ierr)
if (evolve_failed('get_model_number', ierr, evolve, -3)) return
result_reason = get_result_reason(AMUSE_id, ierr)
if (result == retry) then
! trying to spark interest... Why should I care and not fail as
if (evolve_failed('get_result_reason', ierr, evolve, -4)) return
if (report_retries) & summary of what the project is
write(*, '(i6,3x,a,/)') model_number, &
'retry reason', trim(result_reason_str(result_reason))
else if (result == backup) then
if (evolve_failed('get_result_reason', ierr, evolve, -4)) return
if (report_backups) & summary of AMUSE motivation for our project
write(*, '(i6,3x,a,/)') model_number, &
'backup reason', trim(result_reason_str(result_reason))
end if
if (result == retry) result = star_prepare_for_retry(AMUSE_id)
if (result == backup) result = star_do1_backup(AMUSE_id)
if (result == terminate) then
evolve = -11 ! Unspecified stop condition reached, or:
check if ($% number of backups in a row > $% max backups in a row ) then
! terminate
evolve = -14 ! max backups reached
endif
! Show ugly FORTRAN code, transition to simple Python API
if ($% max model number > 0 .and. $% model number >= 6) code is X lines
! ($% max model number) evolve = -13 ! max iterations reached

```

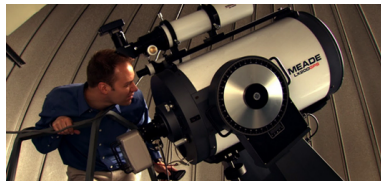
# Purpose of GUnit

- ▶ Ease the use of AMUSE
- ▶ Create/Design/Modify experiments
- ▶ Select, configure, swap out modules and initial conditions
- ▶ Store and restore progress of running experiments.



# Target Audiences

- ▶ Physics Students
- ▶ Observational Astrophysicists
- ▶ Theoretical Astrophysicists



$$\begin{aligned}
 \ddot{X} &= \frac{C}{M} \ddot{X} + \omega \ddot{\theta} \left[ 1 - \frac{f(\theta)}{2} (\Delta k_1 + \Delta k_2 \cos 2\theta) \right] \ddot{X} \\
 &\quad - \frac{\omega \ddot{f}(\theta) \Delta k_2 \sin 2\theta}{2} \left( Y_m - \frac{P}{K} \right) \\
 &= \varepsilon(\Omega + \phi)^2 \cos(\phi + \delta) + \varepsilon \phi \sin(\phi + \delta), \\
 \ddot{Y}_m &+ \frac{C}{M} \ddot{Y}_m - \frac{\omega \ddot{f}(\theta) \Delta k_2 \sin 2\theta}{2} \ddot{X} \\
 &\quad + \omega \ddot{\theta} \left[ 1 - \frac{f(\theta)}{2} (\Delta k_1 - \Delta k_2 \cos 2\theta) \right] \ddot{Y}_m \\
 &= \varepsilon(\Omega + \phi)^2 \sin(\phi + \delta) - \varepsilon \phi \cos(\phi + \delta) \\
 &\quad - \frac{P f(\theta)}{2M} (\Delta k_1 - \Delta k_2 \cos 2\theta), \\
 \ddot{\theta} &+ \frac{K_t + K_c}{I_0} \ddot{\theta} - \frac{K_t}{I_0} \ddot{\phi} = -\frac{C_t + C_c}{I_0} \ddot{\phi} + \frac{C_t}{I_0} \ddot{\phi}, \\
 \ddot{\phi} &+ \frac{C_t}{I_1} \ddot{\phi} - \frac{C_t}{I_1} \ddot{\theta} + \frac{K_t}{I_1} \ddot{\phi} - \frac{K_t}{I_1} \ddot{\theta} \\
 &= \frac{P \varepsilon f(\theta)}{2I_1} (\Delta k_1 \cos(\phi + \delta) - \Delta k_2 \cos(\phi - \delta)) \\
 &\quad + \frac{P^2}{2KI_1} \frac{1}{2} \frac{\varepsilon f(\theta)}{\phi} (\Delta k_1 - \Delta k_2 \cos 2\theta) \\
 &\quad + f(\theta) \Delta k_2 \sin 2\theta \} = \ddot{\theta}_c,
 \end{aligned}$$

# Features

- ▶ Configurable experiments that can be saved and shared.
- ▶ Diagnostic tools that compute useful metrics.
- ▶ Storage of experiment state in case of crashes.
- ▶ Custom diagnostics and code generation.
- ▶ Provides a display of cluster usage to aid in scheduling.





# Design

- ▶ Written in Python using the PyQt4 GUI toolkit.
- ▶ AMUSE is written in Python, streamlines interaction.
- ▶ C++ was considered as it supports Qt as well.
  - ▶ Communication w/AMUSE would be cumbersome.
  - ▶ AMUSE would be in a separate process.
- ▶ Designed APIs for diagnostics, logging and experiment persistence.
  - ▶ Users can create new diagnostics easily.
  - ▶ Experiments can be stored in a file structure, a remote DB etc...



# Tests

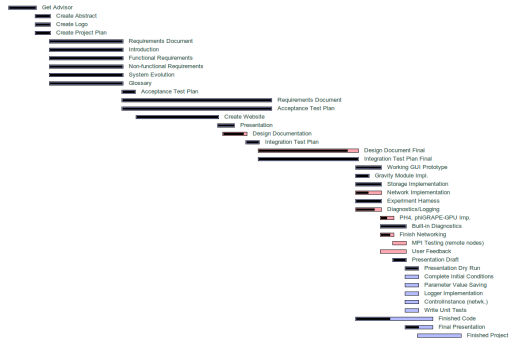
- ▶ Table of tests that pass.

# User Testing

- ▶ Tested with customers (Steve/Tim)

# Project Plan

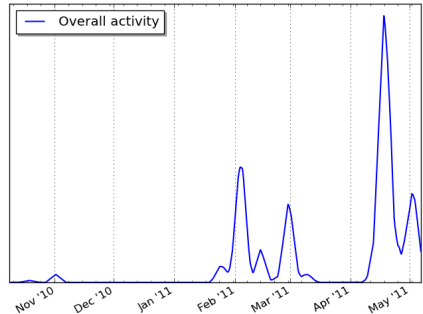
- ▶ Mostly waterfall design process.
- ▶ Initial phases were spent learning the domain (Physics/AMUSE).
- ▶ Roles
  - ▶ Tim: Physics reference, test subject
  - ▶ Andrew/Jason: Experiment and Module design.
  - ▶ Dan: Diagnostics
  - ▶ Raj: Logging
  - ▶ Gabe: Network, GUI.



# Team Management

- ▶ Used Mercurial as our version control system.
  - ▶ Distributed, allows off-line commits.
- ▶ Team met weekly.
  - ▶ Once to plan work, once to code.
- ▶ Bi-weekly advisor meetings.

GPUUnit Commit History



# Project Impact

- ▶ Gives students and physicists easy access to state-of-the-art tools.
- ▶ Simple experiment creation → faster turnaround on experiments.
- ▶ Faster experiments → more time to study them.
- ▶ Current state:
  - ▶ Software is usable to create simple experiments.
  - ▶ Comes with useful diagnostics, from real experimental setups.
  - ▶ Needs refinement before it will be useful to professional astrophysicists.

# Demo

- ▶ Demonstration of a simulation.

# Questions

- ▶ Questions?