Overview

AMUSE Introduction

Purpose

- Purpose of GPUnit
- Target Audiences

Components and Features

- Experiment Management
- Diagnostics
- Command Line Interface
- Parallelism and Distribution

User Interface

Final Goals

Astrophysical Multipurpose Software Environment (AMUSE)

- Start with modules, move upwards -> MPI -> AMUSE python code.
- AMUSE is state of the art for software for simulation of HUGE stellar systems, emphasize this (millions of stars in the galaxy or gas cloud).

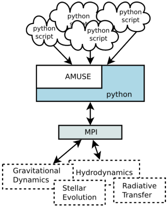

Figure: AMUSE Architecture

State of AMUSE

- Partnership between Drexel and the Leiden Observatory in the Netherlands, sponsored by NOVA.
- NOVA = Netherlands Research School for Astronomy AMUSE
- mention large scale again
- written by hand = hard to share
- waste of work to replicate someone else's diagnostics to fit your exact circumstances.

Purpose of GPUnit

- Ease the creation, execution, and analysis of experiments with AMUSE
- Create experiments with minimal to no programming
- Repeatability
- Sharing Experiments
- Results/ Diagnostics

Target Audiences

- Physics Student
  - A user with minimal to no programming experience and minimal knowledge of astronomy
  - Has an interest in learning, performing, and observing experiments
- Observational Astrophysicists
  - Little programming experience
  - High understanding of astronomy Interested in specific experiments and calculations
- Theoretical Astrophysicists
  - Large programming experience
  - High understanding of astronomy
  - Interested in creating own experiments with custom diagnostic tools

Experiment Management

- Diagnostics on next slide.
- Initial conditions control the state of the stellar system before the experiment starts.
- Logging: what modules are running, when a timestep is taken, how long the timestep took.
- Diagnostics: periodic output of the physical state of the stellar system ex: mass, velocity, position etc...
- User configures how often logs are written, what kind of output is generated

Diagnostics

- Visual diagnostics accomplished through plugins, uses OpenGL

Command Line Interface

- small changes via flags
- headless = without X windows

Parallelism and Distribution

- Modules have a list of parts of the model that they modify, there should be no overlap.
- After the time interval ends, modules send their results back to the central model where they are merged.

User Interface

- Modules are defined by specification files, as are initial condition models.
- There are categories of modules, an experiment can have at most one of each category of module.

Cluster Control

- Nodes are detected automatically so long as they are running a small piece of network code.
- Node usage is polled periodically.

Final Goals

- Integration into AMUSE
- Users can download AMUSE and create/run experiments right away.
- Open AMUSE up for use by a non-programmer audience.