Outline

What is geppetto?
Where does it come from?
What does it do?
Why should you care?
How does it work?
How is it being developed?
How do I use it?
What happens next?
What is Geppetto?

Geppetto is an open-source simulation platform aimed at multi-algorithm and multi-scale simulation of biological systems and their environment.
The Hodgkin-Huxley model (or conductance-based model) is a mathematical model that describes how action potentials in neurons are initiated and propagated. It is a set of nonlinear differential equations that approximates the electrical characteristics of excitable cells such as neurons.
Where does it come from?

- Developed in the context of the OpenWorm project
- Requirements gathering started in 2011
  - Full scale C. elegans simulation used to drive use-cases
  - Generic engine not tied to C. elegans and reusable for different models
- 2.5 years of coding, 10 active contributors
Approaches to building software

Academia
"Monolithic" approach

Industry
"Building blocks" approach
What does it do? (1/2)

- Provides a Web application to control the simulation through a web browser
- Provides a way to define what needs to be simulated in a data driven way
- Defines interfaces to allow third party simulators and model interpreters to extend the platform
- Provides a way to stream simulated data from the backend to the frontend
What does it do? (2/2)

- Provides a 3D engine to visualize in the browser what is being simulated
- Provides an API to interact with the simulators in real time
- Provides an extensible widgets infrastructure to visualize and interact with the data
Why should I care?

- It will make me access my simulations from anywhere through a web browser.
- It will give me a way to visualize and interact with my models without installing any software.
- It will make me model complex systems that require using of different simulators and algorithms.
How does it work?

JS Console | 2D Plotting Widget | Popup Widget | Your Widgets
Web based access and visualisation
API Layer
Geppetto Simulation
NeuroML Module | jLEMS Simulator Module | Sibernetic Simulator Module | Recordings Replayer | Your Model Interpreter | Your Simulator
Geppetto Core
Standardized Data (NeuroML, LEMS, Recordings, Your standard.)
Demo
How is it being developed?

Open Source, **MIT** License
Growing community (**Join us**)  
Bi-weekly Google Hangout open meetings (**Join us**)  
Open process  
Regular Monthly Releases  
(latest 0.1.2 alpha)
Geppetto

Website | Documentation | Wiki | Install Instructions | Releases

Contribution guidelines | Development progress

Geppetto is a web-based multi-algorithm, multi-scale simulation platform engineered to support the simulation of complex biological systems and their surrounding environment.

Although Geppetto was designed with systems biology in mind, thanks to its generic architecture Geppetto can be used anywhere there is need to rely on a backend to perform any kind of simulation which then needs to be streamed to a web client, allowing the user to interact with the simulation remotely and through an API (accessible from an embedded Javascript console) and a set of customisable widget which allows visualising data in different ways.

Geppetto is a modular platform based on Java, OSGi and Spring and different modules (also named bundles) provide different functionalities.

This is the umbrella project that keeps together all the different modules currently available:

- **Essential**
  - org.geppetto.core [build passing]
  - org.geppetto.simulation [build passing]
  - org.geppetto.frontend [build passing]

- **Domain Specific**
  - Neuronal simulation
    - org.geppetto.model.neuromodules [build passing]
    - org.geppetto.simulator.items [build passing]
  - Fluid mechanics simulation
    - org.geppetto.model.sph [build passing]
    - org.geppetto.solver.sph [build passing]
    - org.geppetto.simulator.sph [build passing]
How do I use it?

https://live.geppetto.org

http://www.geppetto.org

http://docs.geppetto.org

https://github.com/openworm/org.geppetto
What technologies are behind it?

Java
Virgo from EclipseRT
OpenCL
OSGi Alliance
Backbone.js
Maven
Spring
HTML5
jQuery
Web Services
WebGL
How are you testing it?

- **Automated**
  - JUnit tests for every module run through Travis build
  - QUnit tests for frontend and GUI
  - QUnit tests for integration testing

- **Manual**
  - Smoke test when every pull request is merged
  - Full manual testing every release
GEPPETTO Client Unit Test

- Hide passed tests
- Check for Globals
- No try-catch

Mozilla/5.0 (Macintosh; Intel Mac OS X 10_6_8) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/34.0.1847.131 Safari/537.36

Tests completed in 7242 milliseconds.
40 assertions of 40 passed, 0 failed.

1. Global Scope Test: Global scope Test (0, 1, 1) Rerun
2. G Object Test: Test Get Current Simulation (0, 1, 1) Rerun
3. G Object Test: Test Debug Mode (0, 2, 2) Rerun
4. G Object Test: Test G Object help method (0, 1, 1) Rerun
5. G Object Test: Test Clear Console (0, 1, 1) Rerun
6. G Object Test: Test Copy History To Clipboard (0, 2, 2) Rerun
7. G Object Test: Test Add Widget (0, 1, 1) Rerun
8. G Object Test: Test Remove Widget (0, 2, 2) Rerun
9. G Object Test: Test Widget (0, 5, 5) Rerun
10. G Object Test: Test Commands (0, 8, 8) Rerun
11. Run Script Test: Run Script Test 1 (0, 1, 1) Rerun
12. Simulation Load From Content Tests: Test Load Simulation from content (0, 1, 1) Rerun
13. Simulation Load From Content Tests 2: Test Load Simulation (0, 1, 1) Rerun
14. Simulation with Scripts: Test Simulation with Script (0, 3, 3) Rerun
15. Simulation controls Test: Test Simulation Controls (0, 1, 1) Rerun
16. Simulation controls Test 2: Test Variable Watch in Plot (0, 4, 4) Rerun
17. Get simulation variables test: Test list simulation variables no crash - SPH (0, 2, 2) Rerun
18. Watch variables test 1: Test add / get watchlists no crash - SPH (0, 1, 1) Rerun
19. Watch variables test 2: Test watch Simulation variables (0, 1, 1) Rerun
How do I contribute?

- Get in touch via info@geppetto.org or through the OpenWorm discus list
- Fork any of the repositories and issue a pull request
- Join one of our bi-weekly meetings
- Support OpenWorm on Kickstarter! (3 days left!)
- Tell your friends and colleagues about it <3
What happens next?

- Replay of recorded simulations
  - Google Summer of Code student through INCF

- Open Source Brain integration

- WormSim development

- Check our development board!
Acknowledgments

● OpenWorm Contributors
  ○ Jesus Martinez
  ○ Giovanni Idili
  ○ Sergey Khayrulin
  ○ Andrey Palyanov
  ○ Padraig Gleeson
  ○ Boris Marin
  ○ Matt Olson
  ○ Mike Vella
  ○ Stephen Larson
  ○ Everyone else

● Silver Lab
  ○ Angus Silver

● Textensor Ltd
  ○ Robert Cannon
Thanks for the attention!
The multiscale problem

Different spatial scale

Diverse models and algorithms
The multiscale problem

Different time scale
How do I model my simulation?

- **Entities**
  - An Entity is the basic building block of the simulation.
  - An Entity can aggregate other Entities (imagine a tree data structure).
  - An Entity can contain one or multiple Aspects.
  - Examples of entities span from a single cell to an entire organism.

- **Aspects**
  - An Aspect defines a particular characterization of an Entity which is specified through a Model and a Simulator.
  - Aspects can be thought of as domain specific descriptions of an Entity.
  - A muscle cell for instance can be described by multiple aspects, one defining its electrical properties, one defining its mechanical structure, one thermodynamics, etc.
Goals

- General enterprise architecture goals
  - Effectiveness
  - Efficiency
  - Flexibility
  - Durability

- Biggest scientific challenge is lack of data and models
  - Geppetto goal is to be completely data driven: new data and models can be fed to a simulation as they become available in a standard format
Architecture Features (1/3)

- **Modular**
  - Geppetto architecture allows separation of functionality into independent, interchangeable modules such that each contains everything necessary to execute only one aspect of the desired functionality.

- **Scalable**
  - Geppetto architecture handles load in a robust manner, being able to be scaled up to accommodate growth.
Architecture Features (2/3)

- **Extensible**
  - Geppetto architecture takes into account future growth by including hooks and mechanisms for expanding/enhancing the system with anticipated capabilities without having to make major changes to the system infrastructure.

- **Generic**
  - Geppetto architecture is not tied to any specific biological simulation, neither to the model being simulated or the aspects being simulated.
Architecture Features (3/3)

● **Client-Server**
  ○ Geppetto architecture is based on the client-server model where the simulation is controlled by a client through a web interface.

● **Distributed**
  ○ Geppetto architecture allows separation of the execution of a simulation into multiple processes executable on different servers and capable of communicating with each other.

● **Dynamic deployment**
  ○ Geppetto architecture allows deploy, re-deploy, and undeploy of modules without a server restart.