COMPUTER SIMULATIONS

Important for research…

But also really **fun**

Case Study:

CONWAY'S GAME OF LIFE

GARLICSIM

Pythonic Platform for computer simulations

Scientist explores
a world...

http://www.flickr.com/photos/posk/3528783593/

Comes up with a model for that world

The scientist isn't sure at all if the model fits the real world. Maybe it fits the real world only in *some* situations?

A computer simulation can show a world governed by the model

The scientist can compare the results of his simulations to the real world, and know whether his model is good.

And after he discovers a model that works, he can use computer simulations to make experiments in that world.

Thinking up a

model

This should be a simple and straightforward step, right?

Running a

Simulation

of the model

It’s not!

http://www.flickr.com/photos/theyoungthousands/2482389516/

Maybe you can write a quick-n-dirty simulation easily enough…

But every time you'll want to change something, you'll have to do a lot of work.

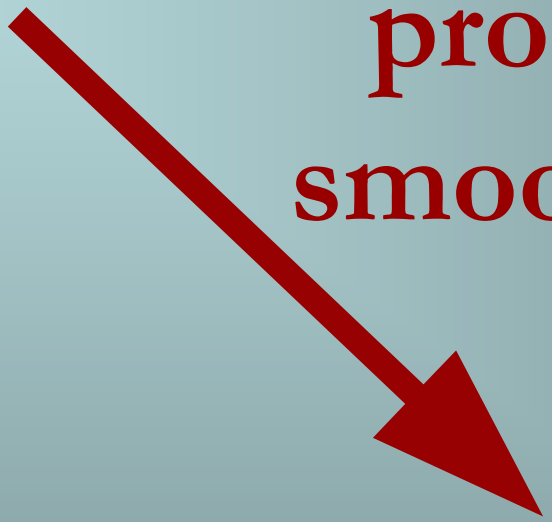
A scientist who wants to work efficiently with simulations needs a **simulations framework** in which he can easily:

- Make experiments
- Measure their results
- Make changes in the simulation world
- Tweak the world laws
- And more...

Thinking up a

model

GarlicSim's goal:
To make this process easy, smooth and fast



Running a

Simulation

of the model

To make a framework for simulations, one must answer the question:

What do all simulations have in common?

"Life" simulation

Physics simulation

Stock market simulation

These are some examples of different kinds of simulation. What do these simulations have in common?

"Life" simulation

Physics simulation

Stock market simulation

Every simulation has:

1. A concept of a **world state**

   and

2. A **step function**.

"Life" simulation

Physics simulation

Stock market simulation

World state:

A description of a frozen moment in time in the simulation. Contains all the information there is about the world at that single point in time.

"Life" simulation

World state:
2-dimensional array of cells, saying about each cell whether it's dead or alive.

Physics simulation

World state:
The position, velocity, acceleration, mass etc. for each body in the system.

Stock market simulation

World state:
The price of every stock, the amount of money and stock that every trader owns, etc.

"Life" simulation

Physics simulation

Stock market simulation

Step function:

A function that takes a world state, and outputs the next world state that follows it in time. This is where you put your "world laws".

"Life" simulation

Step function: Count the live neighbors for every cell, if it's 2, the cell will be alive, if it's 3... etc.

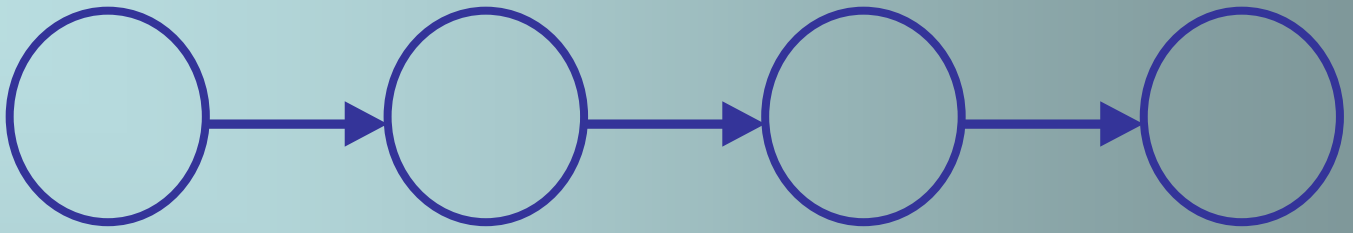
Physics simulation

Step function: Calculate the sum of forces on each body, use F=ma to get acceleration, use that to estimate position after Δt.

Stock market simulation

Step function: For every trader, calculate utility of buying/selling every stock, decide which stocks he will buy, transfer money/stocks accordingly.

Once you have a state and a step function, you're set; You can start crunching the timeline of your simulation.



You generate the first state.

Then you put it in the step function, and get the next state.

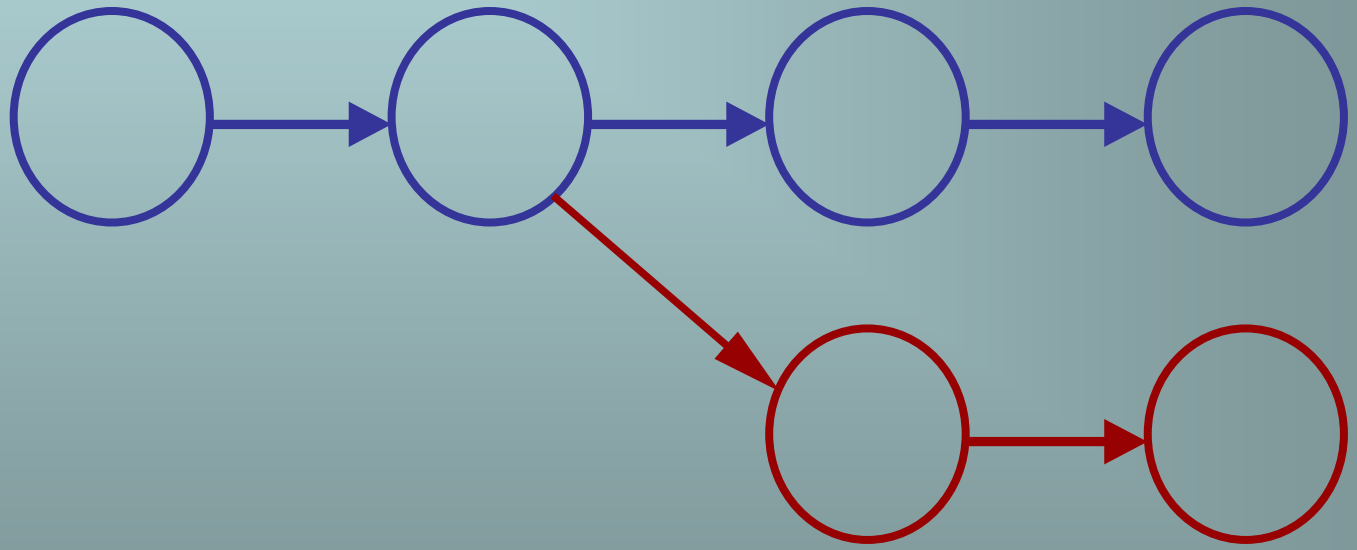
Then you put *that* state in the step function, and you're building your timeline.

`garlicsim_wx`
GarlicSim's graphical interface.

- Is awesome.
- Written with `wxPython`.
- Completely optional; You're free to just `import garlicsim` instead.
- Cross-platform: Windows/Mac/Linux.

What is a “time tree”?

A time tree is a generalization of a timeline.

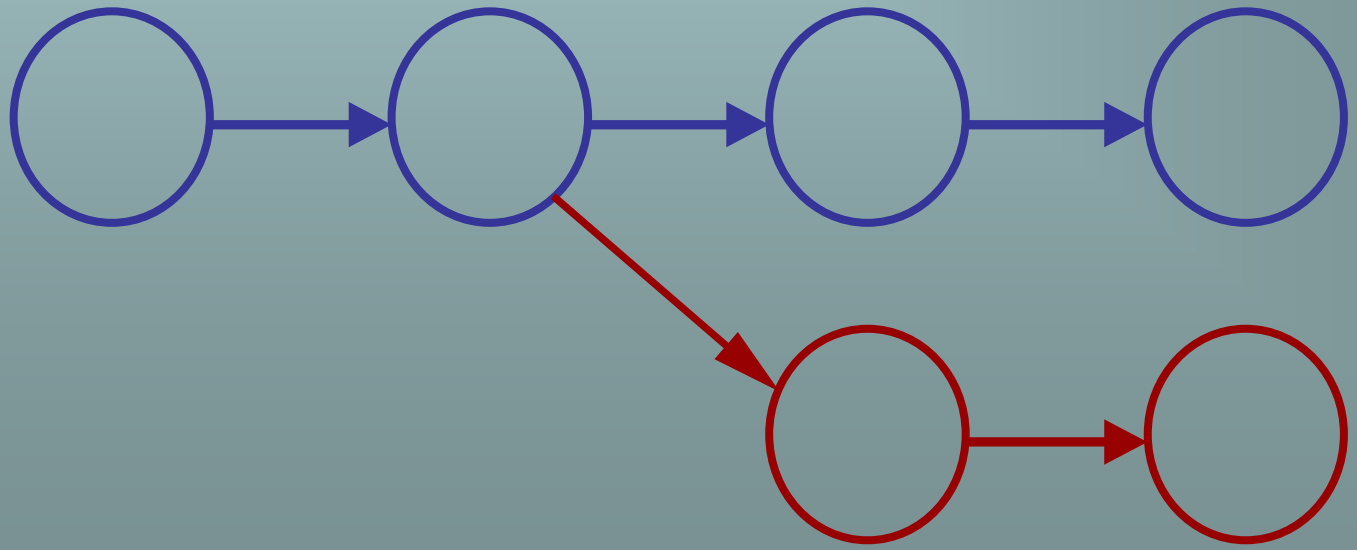


The difference: A time tree can be forked.

Why would you want to fork your time tree?

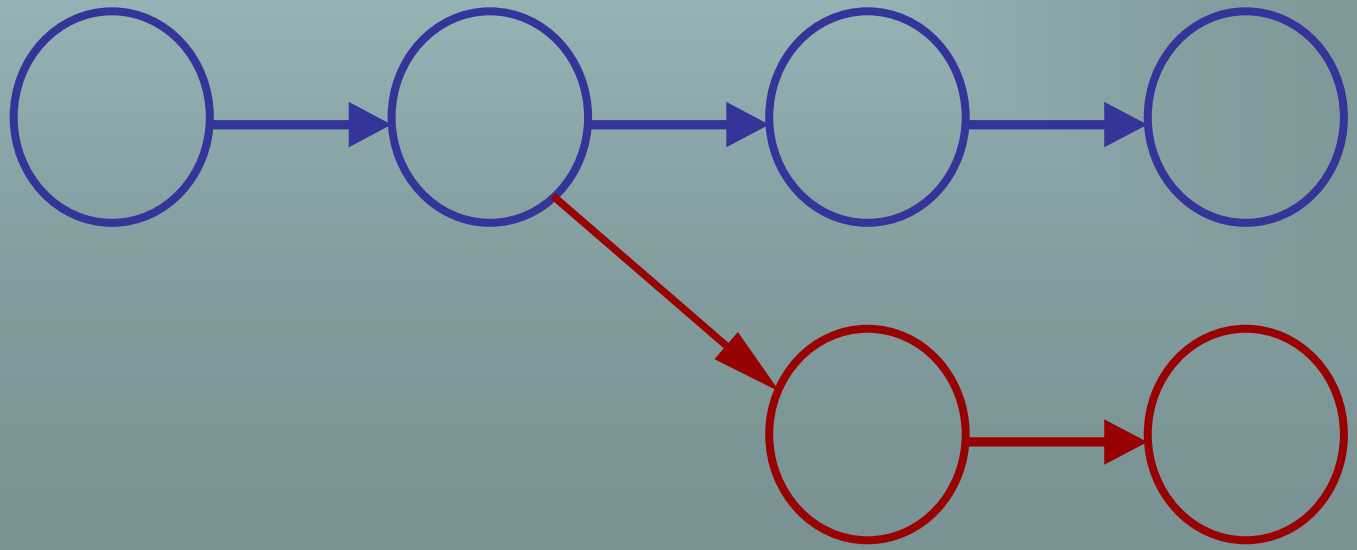
Usage 1: To ask, how could things have happened differently?

(Relevant only in non-deterministic simulations.)



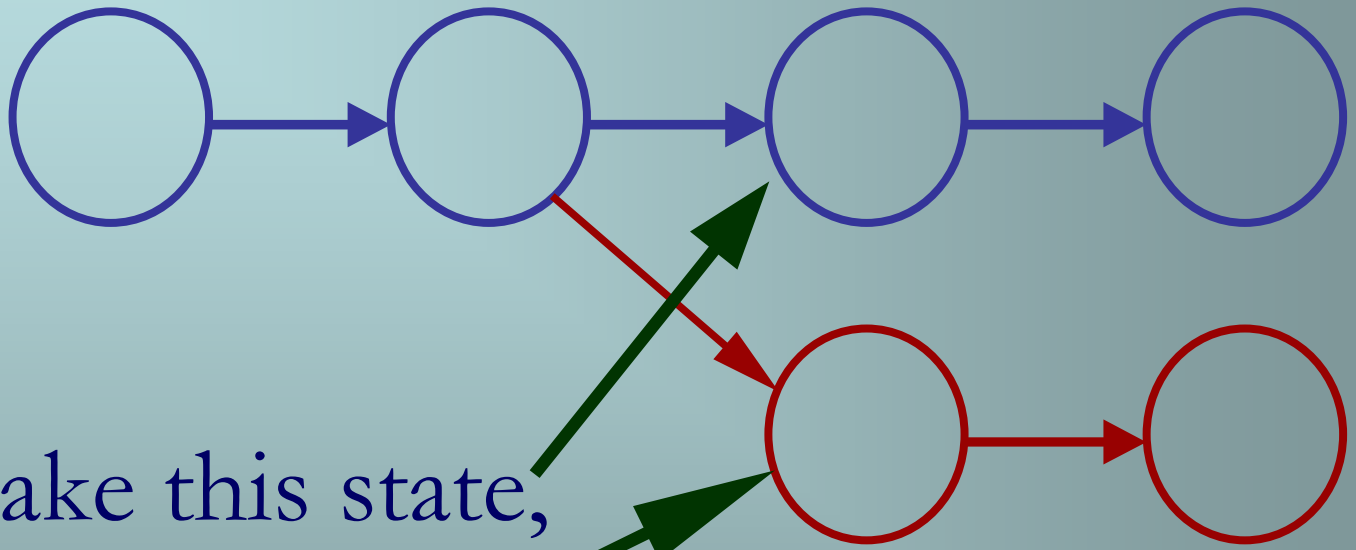
Why would you want to fork your time tree?

Usage 2: To ask, how could things have happened differently, if the world laws were different?
(Giving different arguments to the step function.)



These 2 usages were both examples of a **fork by crunching**: Making a fork in the time tree by telling GarlicSim to recalculate from a given state.

Usage number 3 is **fork by editing**.



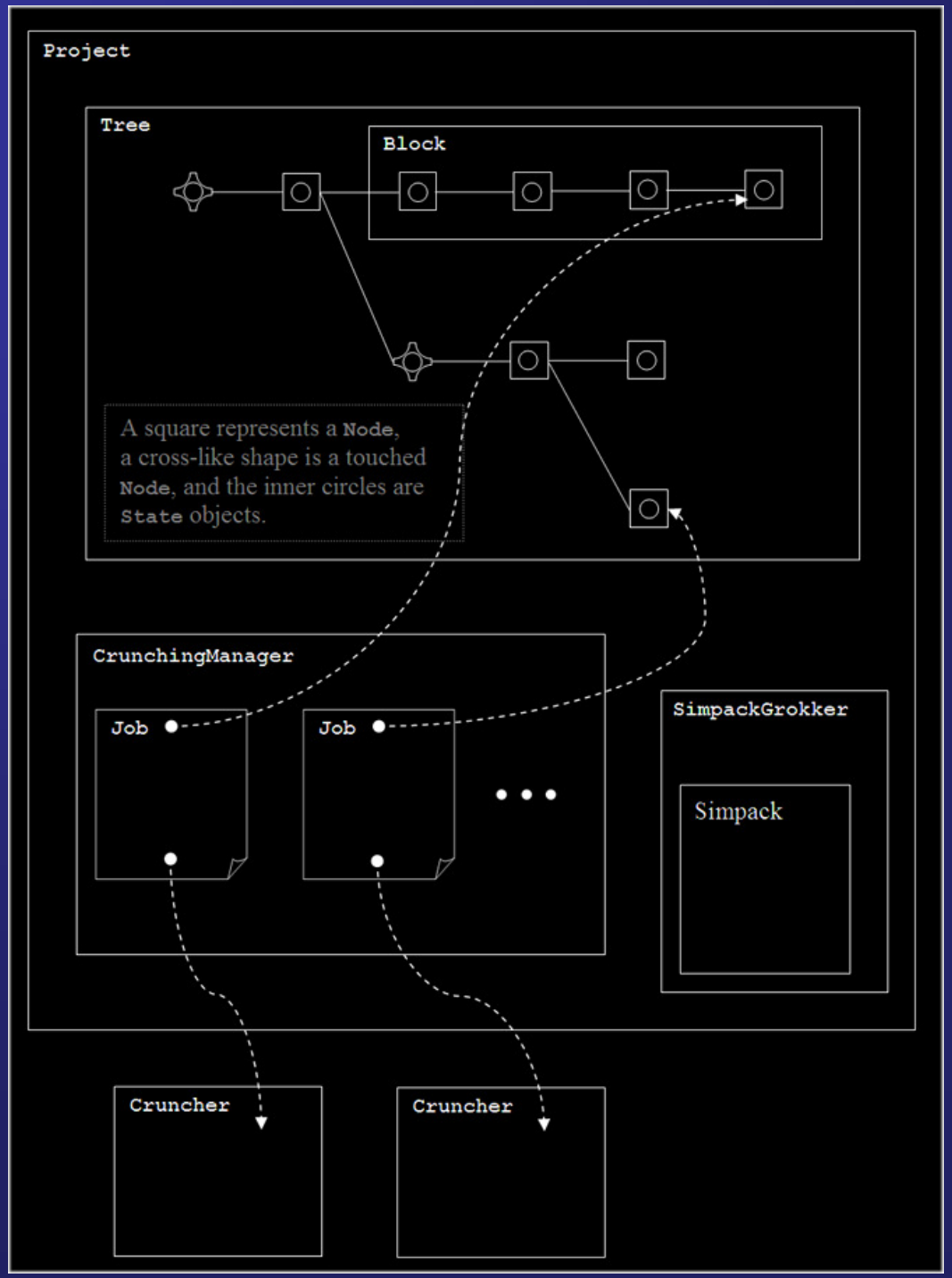
You take this state,
Modify it,
Put it here,
And see how the simulation evolves from there.

What is a simpack?

A simpack is the **type** of the simulation.

The simpack has all the code for the states and the step function.

A **simpack** is a package of code;
A **simulation**, which is also called a **project**, is an object created using the simpack.


Project
Tree
Block
A square represents a Node, a cross-like shape is a touched Node, and the inner circles are State objects.
CrunchingManager
Job
Job
SimpackGrokker
Simpack
Cruncher
Cruncher

There is an API for Cruncher types;
You can write new kinds of crunchers
and use them in your simulation.

Getting started with GarlicSim

- Go to the website: `http://garlicsim.org`
- Download and install. (No compiling!)
- Do tutorial 1, in which you'll run a simulation.
- Do tutorial 2, in which you'll write your first simpack.

Getting started with GarlicSim

If you have *any* questions or feedback, mail me: `cool-rr@cool-rr.com` or the mailing list: `garlicsim@librelist.org`

And please, give me feedback.

Every time you give me feedback, an angel gets a pair of wings.

Have fun simulating!