



Lively Simulation

Leonardo Hübscher, Juliane Kleinknecht

End User Development Seminar SS 20
Software Architectures

14.07.2020

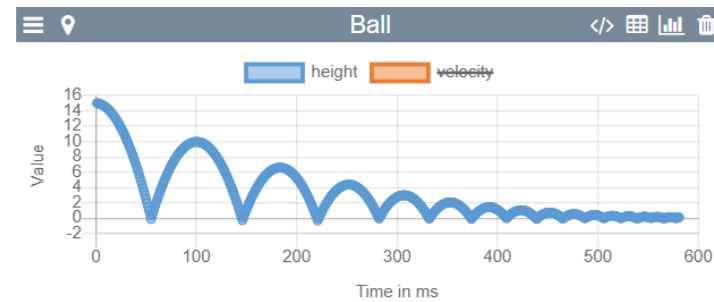
Outline

1. Simulations
2. Motivation
3. Project Goal
4. Energy Simulation
5. Demo
6. Improve End User Experience
7. What's Next?

Simulations

Real system 🌐 → Too complex ? 😞 → Model 📄 → Simulation ⚙️

- Experimenting in simulation → finding hypothesis
- Understanding the behavior of systems
- **Dynamic simulations:** time dependent 🕒
→ observe processes and flows



Bouncing Ball Example

Demo Bouncing Ball

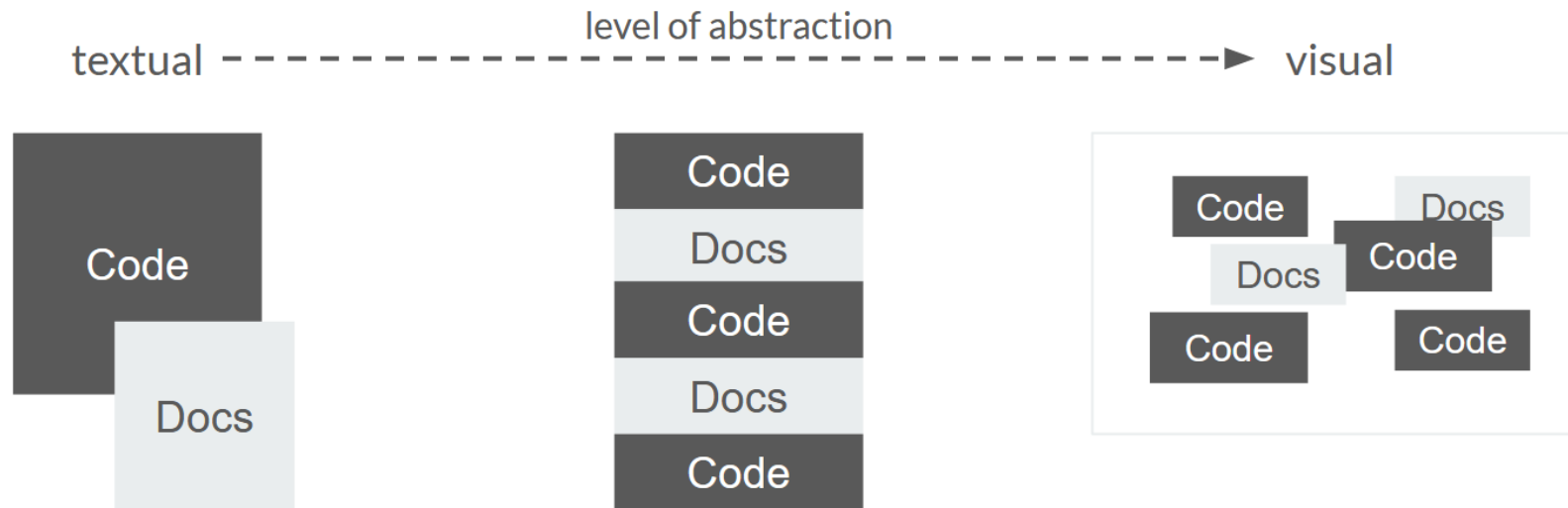
The screenshot displays a web browser window titled "bouncing-ball.html" with the URL <https://lively-kernel.org/lively4/lively4-livelyenergy/demos/engery-sim/bouncing-ball.html>. The simulation interface includes a control bar with a "Step" button, "Start", "Steps per Second: 1", "Time delta [s] 0.001", "Current time 0 ms", and a "Reset" button. Below the control bar are three code blocks:

- Gravity:** `g -9.81`, `#Ball.velocity += g * dt;`
- Friction:** `frictic 0.2`, `floor 0`, and a conditional code block:

```
if (#Ball.velocity < 0 && #Ball.height <= floor) { #Ball.velocity *= -(1 - frictic); }
```
- Reset:** `#Ball.height = 15;` and `#Ball.velocity = 0;`

A "Ball" graph is also visible, showing a coordinate system with "Value" on the y-axis (ranging from 0 to 1.0) and "Time in ms" on the x-axis (ranging from 0 to 1.0). A video player overlay at the bottom indicates a duration of 0:00 / 1:02.

Motivation



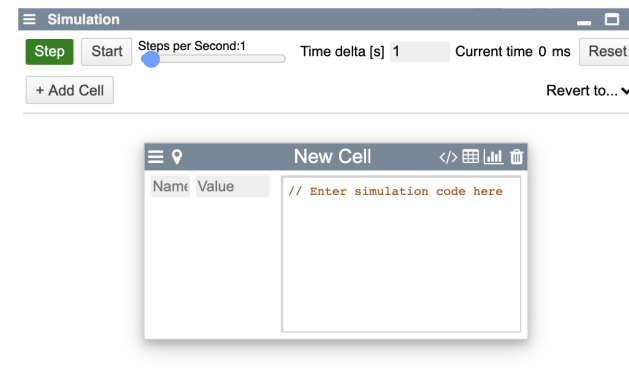
We want to enable end-users to create simulations in a visually structured way

Project Goal

We want to enable end-users to create simulations in a visually structured way

Idea Build a semi-graphical simulation framework

- ✿ Starting point: PoC* in Lively Webwerkstatt (legacy)
- 📄 Allow documentation to be part of simulation
- 🔍 Provide a way that allows simulation result investigation
- 🔗 Proper Lively integration
- 🔧 Port existing *energy simulation* from legacy



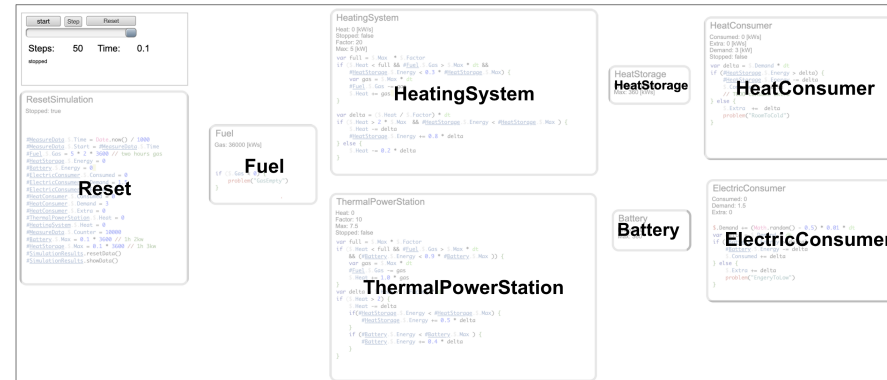
Simulation Framework

*PoC...Proof of concept

Energy Simulation

- Simulate energy flow
- Fuel → transformer → consumer
- Possible hypothesis:
 - Is heat storage big enough?
 - Is a backup system required?

Lively ThermalPowerStation Simulation



Energy simulation in Lively Webwerkstatt [1]

[1] <https://lively-kernel.org/repository/webwerkstatt/demos/EnergySimulationScripted.xhtml>



Demo

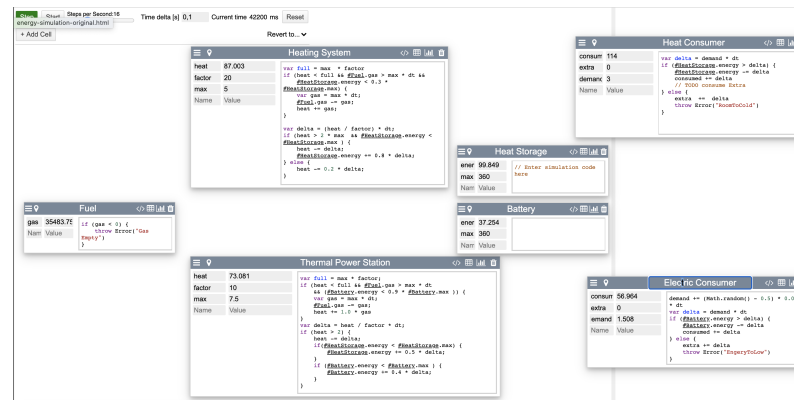
Energy Simulation Port

Demo

Let's use our framework for the energy simulation!

What we want to show:

1. Ported simulation is working
2. How the simulation is being edited
3. Cell locality
4. Cell state visualization



Energy simulation in Lively4

Demo

The screenshot displays the Lively Simulation interface with a central simulation area and several floating cell windows. At the top, the simulation controls include a 'Step' button, a 'Start' button, a 'Steps per Second:10' slider, 'Time delta [s] 0,1', 'Current time 0 ms', and a 'Reset' button. Below these are '+ Add Cell' and 'Revert to...' options.

The simulation area contains several cells:

- Fuel:** A cell with a gas icon. It contains a table with 'gas' at 36000 and a code block:

```
if (gas < 0) {
  throw Error("Gas Empty");
}
```
- Heating System:** A cell with a radiator icon. It contains a table with 'heat' at 0, 'factor' at 20, and 'max' at 5. Below is a code block:

```
var full = max * factor;
if (heat < full && #Fuel.gas > max * dt &&
  #HeatStorage.energy < 0.3 *
  #HeatStorage.max) {
  var gas = max * dt;
  #Fuel.gas -= gas;
  heat += gas;
}
var delta = (heat / factor) * dt;
if (heat > 2 * max && #HeatStorage.energy <
  #HeatStorage.max) {
```
- Heat Storage:** A cell with a flame icon. It contains a table with 'ener' at 0 and a text input field labeled '// Enter simulation code'.
- Battery:** A cell with a battery icon. It contains a table with 'ener' at 0 and a text input field.
- Heat Consumer:** A cell with a person icon. It contains a table with 'consum' at 0, 'extra' at 0, and 'demand' at 3. Below is a code block:

```
var delta = demand * dt;
if (#HeatStorage.energy > delta) {
  #HeatStorage.energy -= delta;
  consumed += delta;
  // TODO consume Extra
} else {
  extra += delta;
  throw Error("RoomTooCold");
}
```
- Electric Consumer:** A cell with a lightbulb icon. It contains a table with 'consum' at 0, 'extra' at 0, and 'demand' at 1.5. Below is a code block:

```
demand += (Math.random() - 0.5) * 0.
* dt;
var delta = demand * dt;
if (#Battery.energy > delta) {
  #Battery.energy -= delta;
  consumed += delta;
}
```
- Thermal Power Station:** A cell with a power plant icon. It contains a table with 'heat' at 0, 'factor' at 10, and 'max' at 7.5. Below is a code block:

```
var full = max * factor;
if (heat < full && #Fuel.gas > max * dt
  && (#Battery.energy < 0.9 * #Battery.max ))
{
  var gas = max * dt;
  #Fuel.gas -= gas;
  heat += 1.0 * gas;
}
var delta = heat / factor * dt;
if (heat > 2) {
  heat -= delta;
  if (#HeatStorage.energy < #HeatStorage.max) {
    #HeatStorage.energy += 0.5 * delta;
```

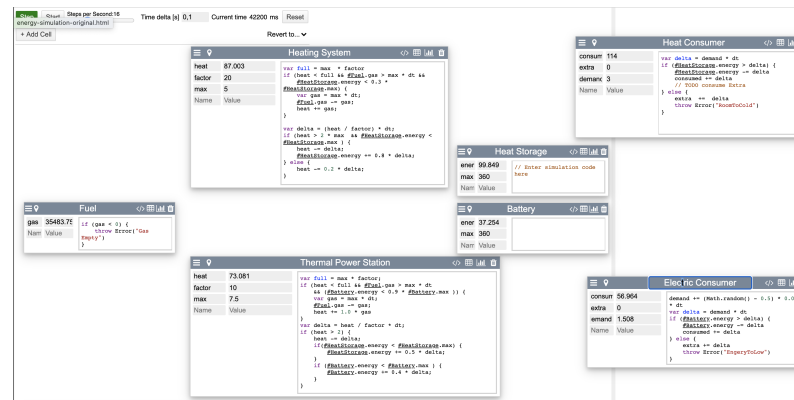
At the bottom left, there is a play button and a progress indicator showing '0:00 / 2:00'. At the bottom right, there are volume, full screen, and menu icons.

Demo

Let's use our framework for the energy simulation!

What we have seen:

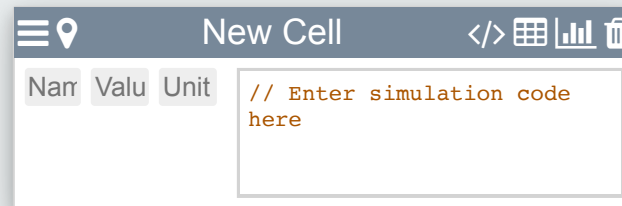
1. Ported simulation works ✓
2. Loading a simulation and adding a cell ✓
3. Independent cells/ global state access ✓
4. Log tables and charts ✓



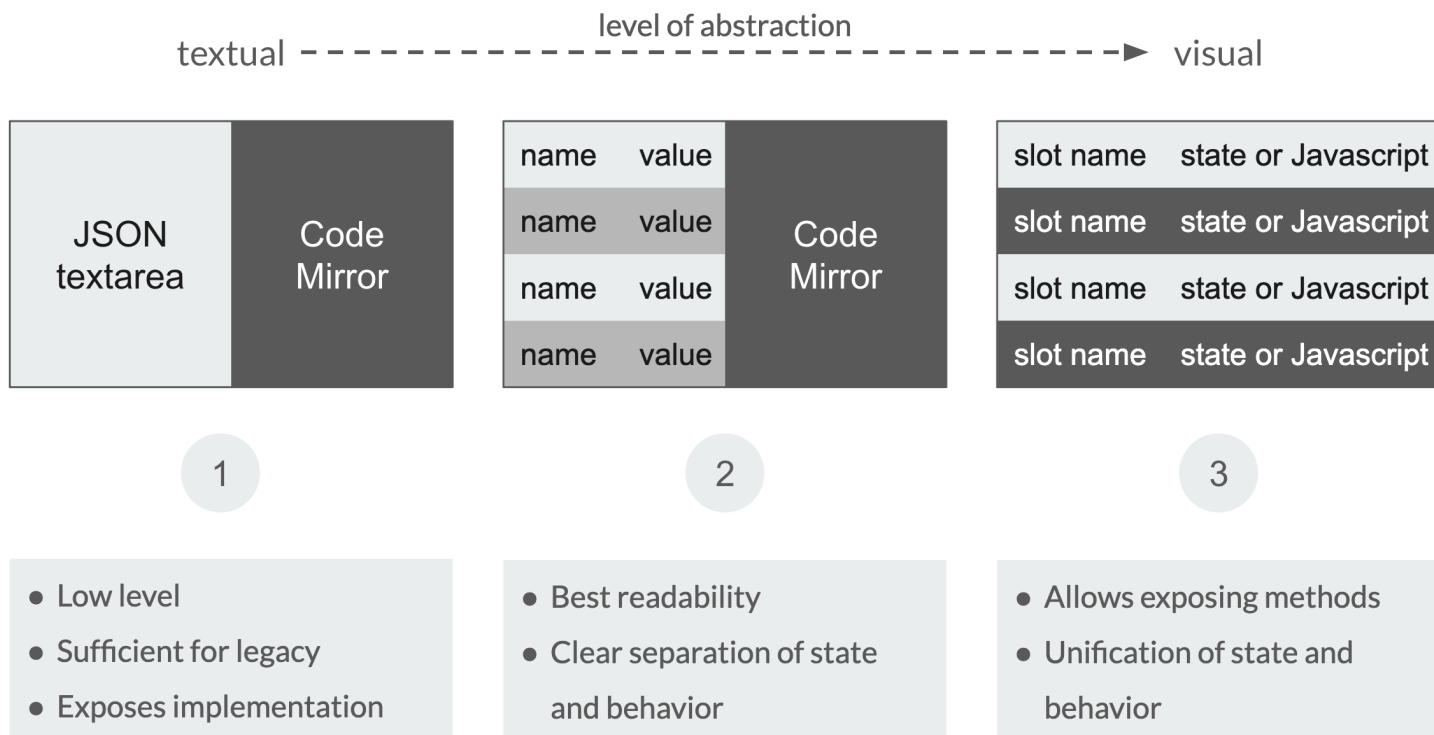
Energy simulation in Lively4

Design Improvements

Improve end user experience



Managing State and Behavior



Managing State and Behavior

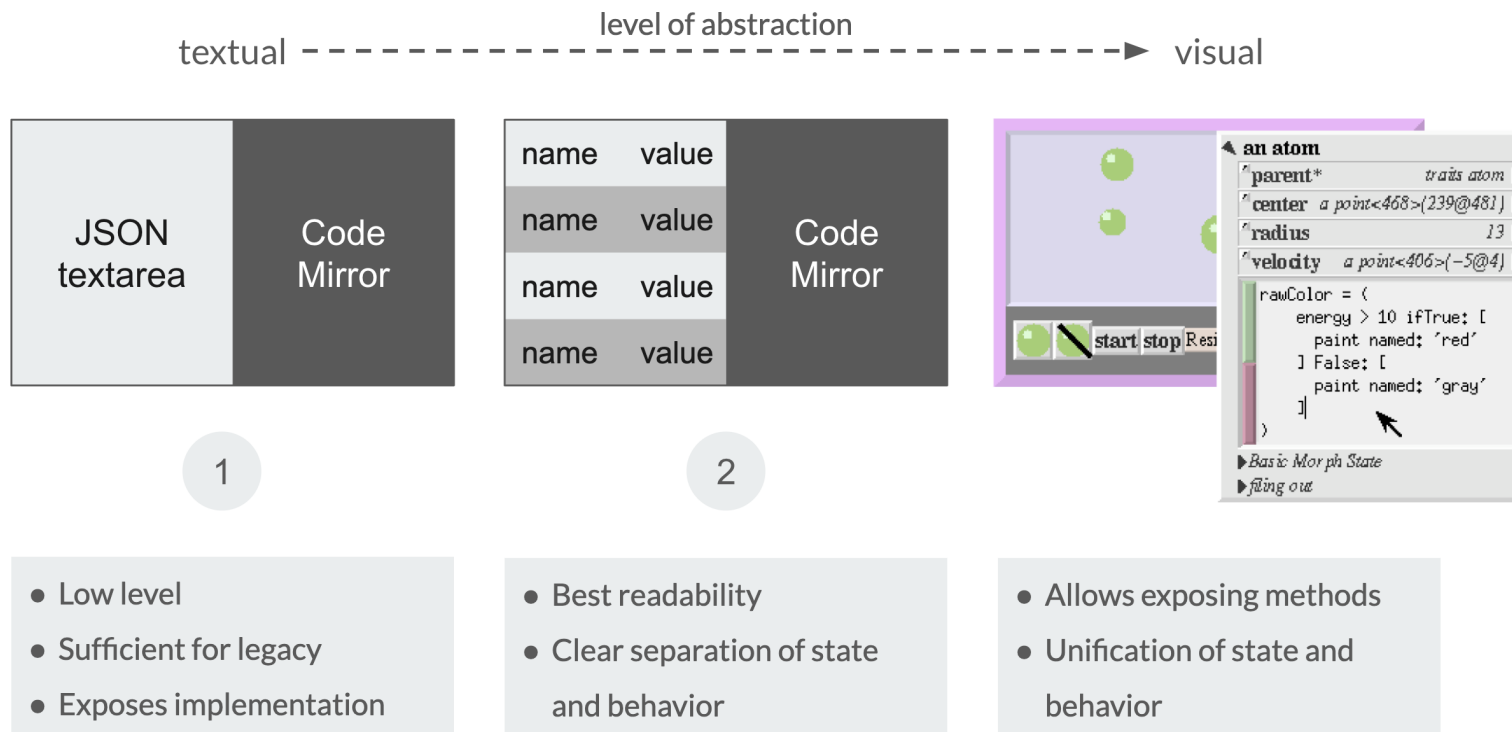


Image based on: Smith, R. B., Maloney, J., & Ungar, D. (1995, October). The Self-4.0 user interface: manifesting a system-wide vision of concreteness, uniformity, and flexibility. In Proceedings of the tenth annual conference on Object-oriented programming systems (pp. 47-60).

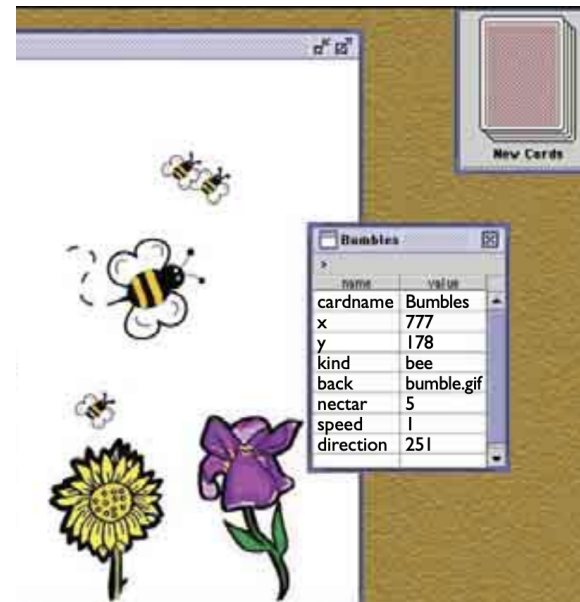
14

Cell Views

- Legacy had one centralized log/ chart
- Each cell represents one simulation unit

“ *An environment that works the way nonprogrammers expect is more inviting and helps users become more confident and productive. [2]*

→ each unit should have it's own state/ behavior, log, chart



each entity has different views [2]

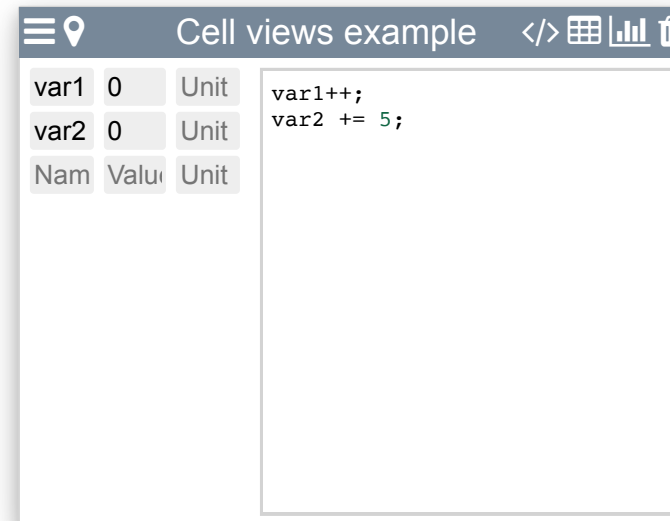
[2] Myers, B. A., Pane, J. F., & Ko, A. (2004). Natural programming languages and environments. Communications of the ACM, 47(9), 47-52.

Cell Views

- Legacy had one centralized log/ chart
- Each cell represents one simulation unit

An environment that works the way nonprogrammers expect is more inviting and helps users become more confident and productive. [2]

→ each unit should have it's own state/ behavior, log, chart, ...**custom view?!**



[2] Myers, B. A., Pane, J. F., & Ko, A. (2004). Natural programming languages and environments. Communications of the ACM, 47(9), 47-52.

Unit Support for DSL

Goal Allow usage of units in DSL and support automatic conversion

Legacy Add units as comment to state only

Solution Use `mathjs` [3] to parse the code and work with units



[Mathjs] features a flexible expression parser [...] and offers an integrated solution to work with [...] units, and matrices. Powerful and easy to use. [3]

```
math.evaluate('5.08 cm + 2 inch') // 10.16 cm
```

[3] <https://mathjs.org/>

Outlook: Unit Support for DSL

Goal Allow usage of units in DSL and support automatic conversion

Legacy Add units as comment to state only

Solution Use `mathjs` [3] to parse the code and work with units

The logo for mathjs, featuring the word "mathjs" in a white, sans-serif font on a red rectangular background.

[Mathjs] features a flexible expression parser [...] and offers an integrated solution to work with [...] units, and matrices. Powerful and easy to use. [3]

```
math.evaluate('5.08 cm + 2 inch') // 10.16 cm
```

Mathjs does not fully support Javascript (f.e. loops, if/else)
User would have to learn mathjs to use it

[3] <https://mathjs.org/>

What's Next?

Learnings 📖

- Lively integration is easier than expected
- Plenty tools & features → keeping overview is hard for newbies
- Stable internet connection required
- Dataflow for nested components is not specified

Next Steps 🏃

- Mirroring of cells

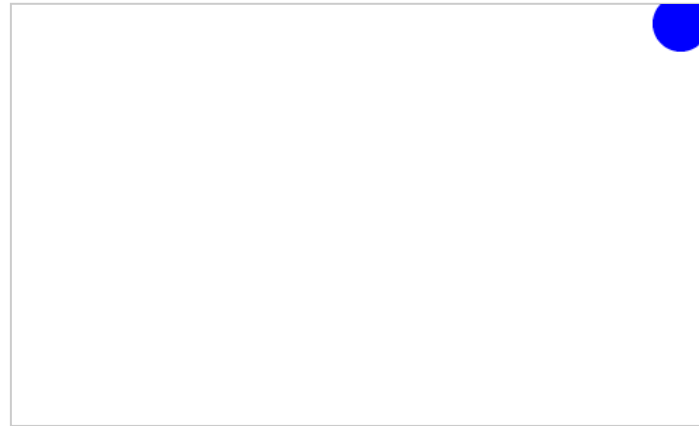
What's Next?

Future Work

→ *Gas tank simulation* shows functionality for further development

- **Variable number** of entities of a cell
- Creating a **visualization** component

Hits: 5

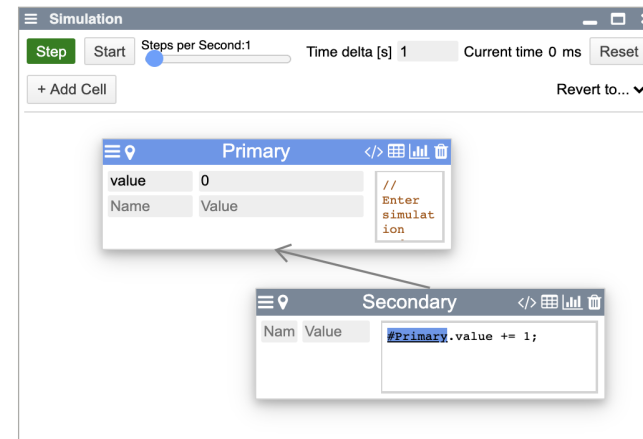


Project Goal

We want to enable end-users to create simulations in a visually structured way

Idea Build a semi-graphical simulation framework

- ✿ Starting point: PoC* in Lively Webwerkstatt (legacy)
- 📄 Allow documentation to be part of simulation
- 🔍 Provide a way that allows simulation result investigation
- 🔗 Proper Lively integration
- 🔧 Port existing *energy simulation* from legacy



Simulation Framework