Marcel Stimberg, ENS Paris
marcel.stimberg@ens.fr

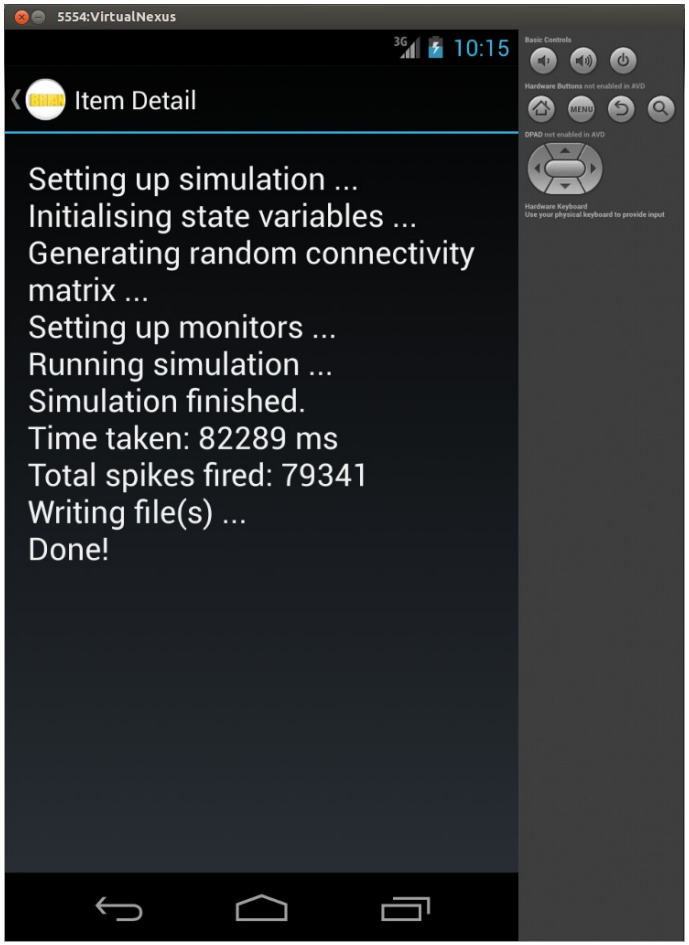
Motivation for Brian²

- Learn from the experiences made in Brian 1 and incorporate them in the design from the beginning
- Core principles:
  - Explicit model descriptions with strings
  - Code generation
  - Modularity
- Allow for some backward incompatibility

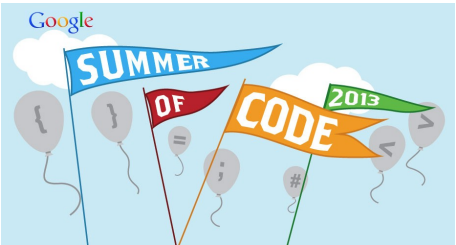
New targets for Brian

- Separating **preparatory work** (applying numerical integration methods to model equations …) from **computational work** (state update step, setting variable values, …)
- Will allow "standalone" Brian code, e.g. completely Brian-independent C++ code
- and then…

Robots and phones!



Achilles Koutsou,
University of Cyprus



Code generation step 1

Model equations

```
dv/dt = -(-v + w) / tau_v : volt
dw/dt = -w / tau_w : volt
```

+

Numerical integration

```
x_new = x + dt*f(x, t)
```

=

Abstract code

```
_v = dt*(-v + w)/tau_v + v
_w = -dt*w/tau_w + w
v = _v
w = _w
```

Code generation step 2 (C++)

general template

```
for(int _n_idx=0; _n_idx<_num_neurons; _n_idx++)
{
    double v = _ptr_array_neurongroup_v[_n_idx];
    double w = _ptr_array_neurongroup_w[_n_idx];
    const double _w = -(dt) * w / tau_w + w;
    const double _v = dt * (-(v) + w) / tau_v + v;
    w = _w;
    v = _v;
    _ptr_array_neurongroup_v[_n_idx] = v;
    _ptr_array_neurongroup_w[_n_idx] = w;
}
```

model-specific code