A Decentralized Approach for Programming Interactive Applications with JavaScript and Blockly

Assaf Marron
Gera Weiss
Guy Wiener

October 2012

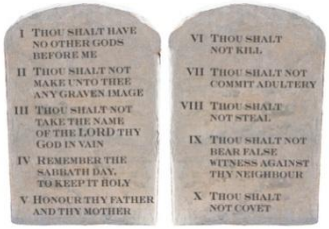
Weizmann Institute of Science

Ben Gurion University

Basic idea of behavioral programming

Develop complex software systems

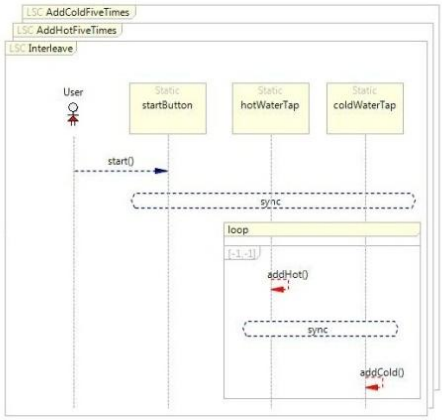
from simple specifications of required behavior



- ✓ You shall do …
- ✓ You shall not do …
- ✓ . . .

that are interwoven automatically

From required behavior to code



LSC: A visual language for scenario specification

- Damm and Harel 2001, Harel and Marelly 2003
- Natural yet executable
- For requirement specification and for programming
- IDEs – PlayGo, Play Engine

BPJ: A package for programming scenarios in Java

```
class AddHotFiveTimes extends BThread {
    public void runBThread() {
        for (int i=1; i<=5; i++) {
            bSync(addHot, none, none);
        }
    }
}
```

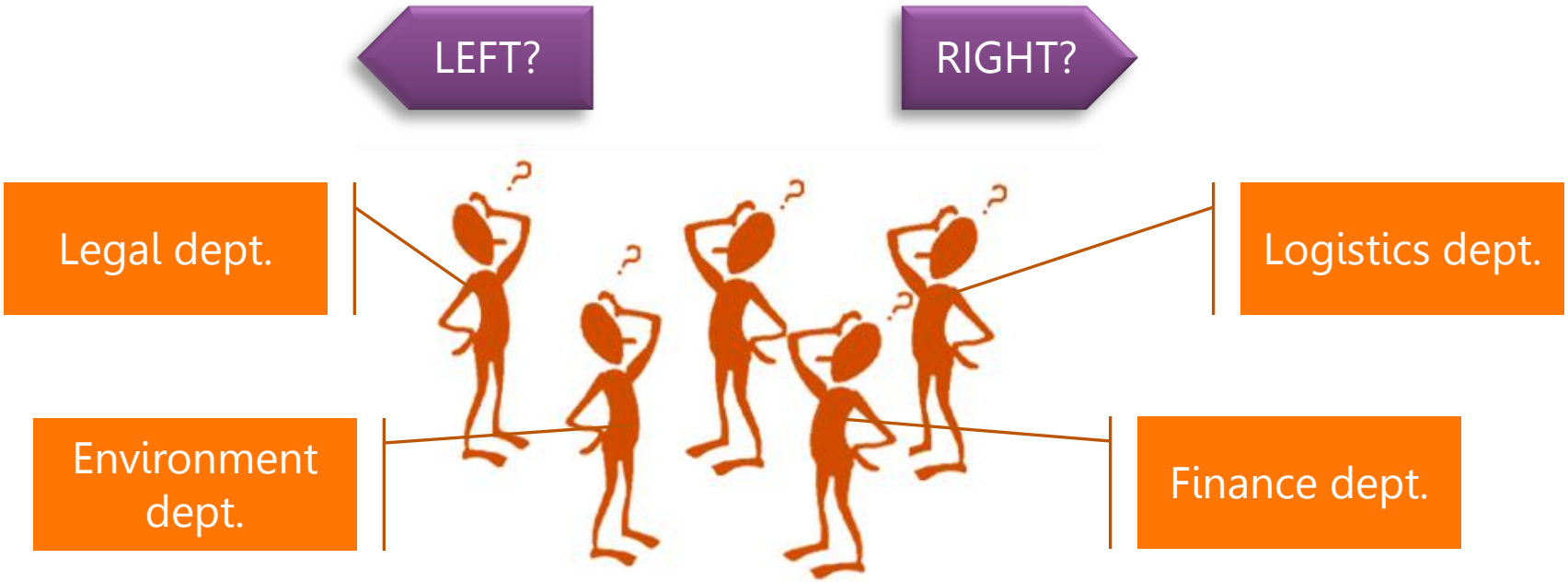
- Harel, Marron, and Weiss 2010
- Programming scenarios in standard languages
- Integrate with & complement other paradigms (OOP, aspects, rule-based, agile, …).

Now also in **C, C++, Erlang, JavaScript and Blockly**

The basic idea of BP

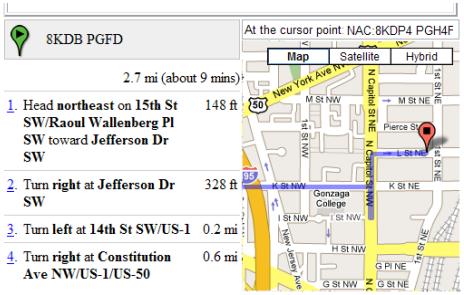
At every decision point in execution different components represent different facets of the next decision

A general interweaving mechanism collects and evaluates the information and then makes a decision



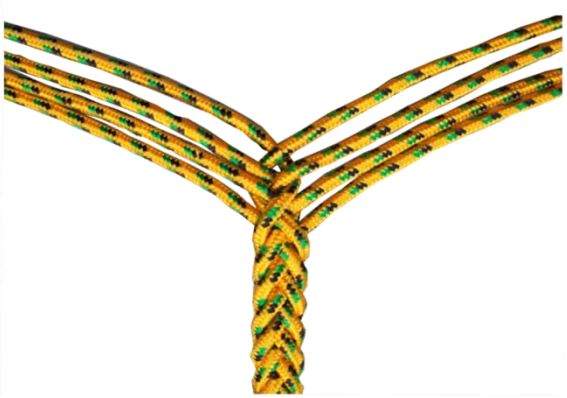
Requesting and blocking allow humans to interweave processes in their head

Route



- Go for 90 km on road A1
- Turn left to road A4
- Go for 70 km on road A4
- . . .

+



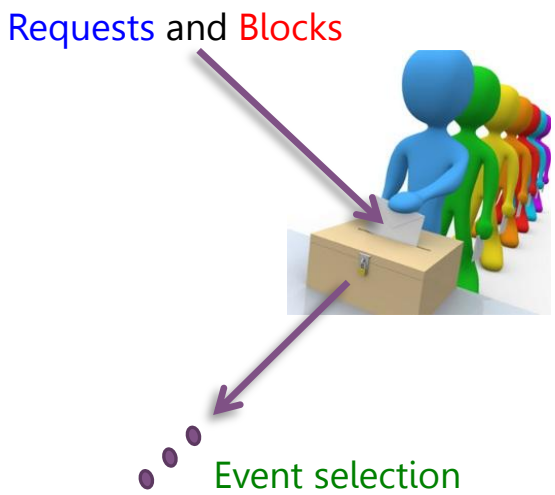
Schedule

- . . .
- Drive for 4 hours
- Stop for lunch
- Drive for 3 hours
- Look for a hotel
- Stop for the night

= Trip

Behavior execution cycle

1. Behavior threads (b-threads) post declarations:
   - Request events
   - Wait for events
   - Block events

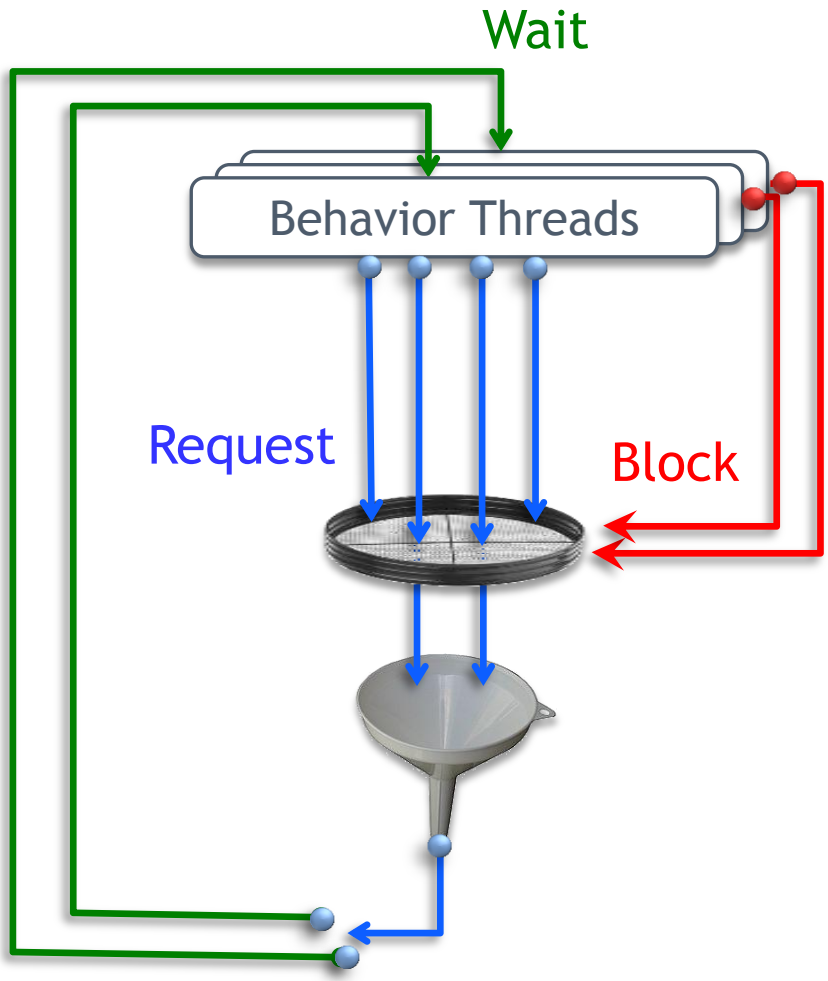


2. When all declarations are collected:

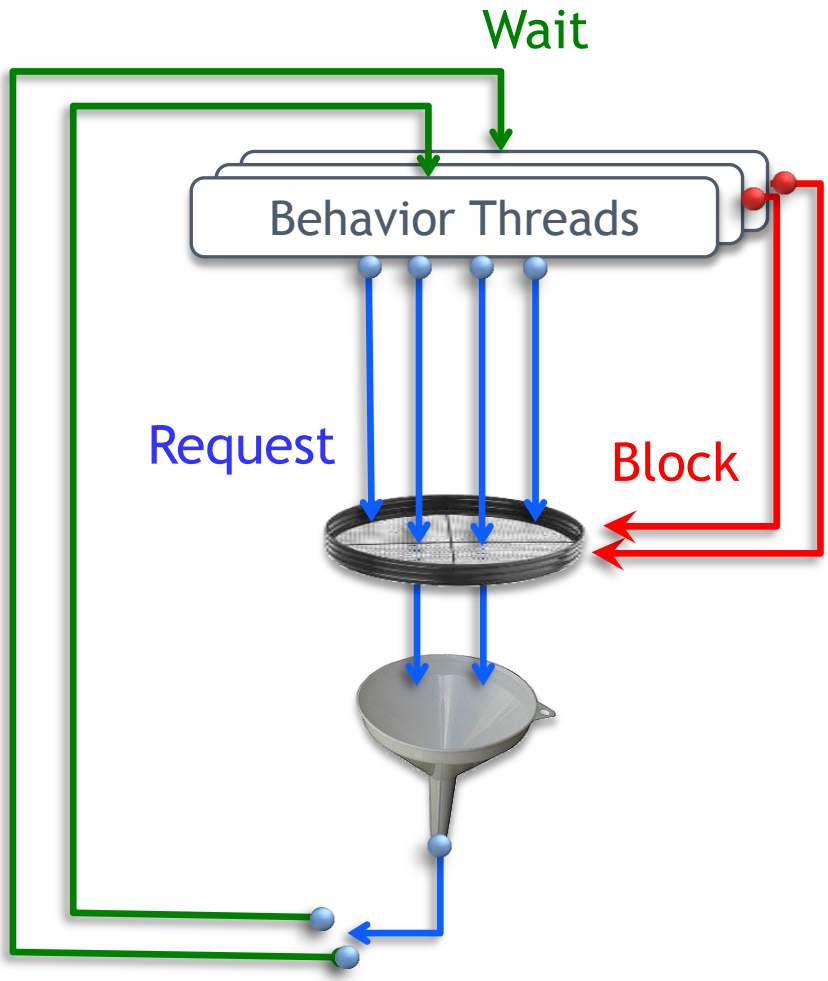
   An event **is selected** that is **requested** and not **blocked**.

   B-threads **waiting** for or **requesting** the event are **notified**

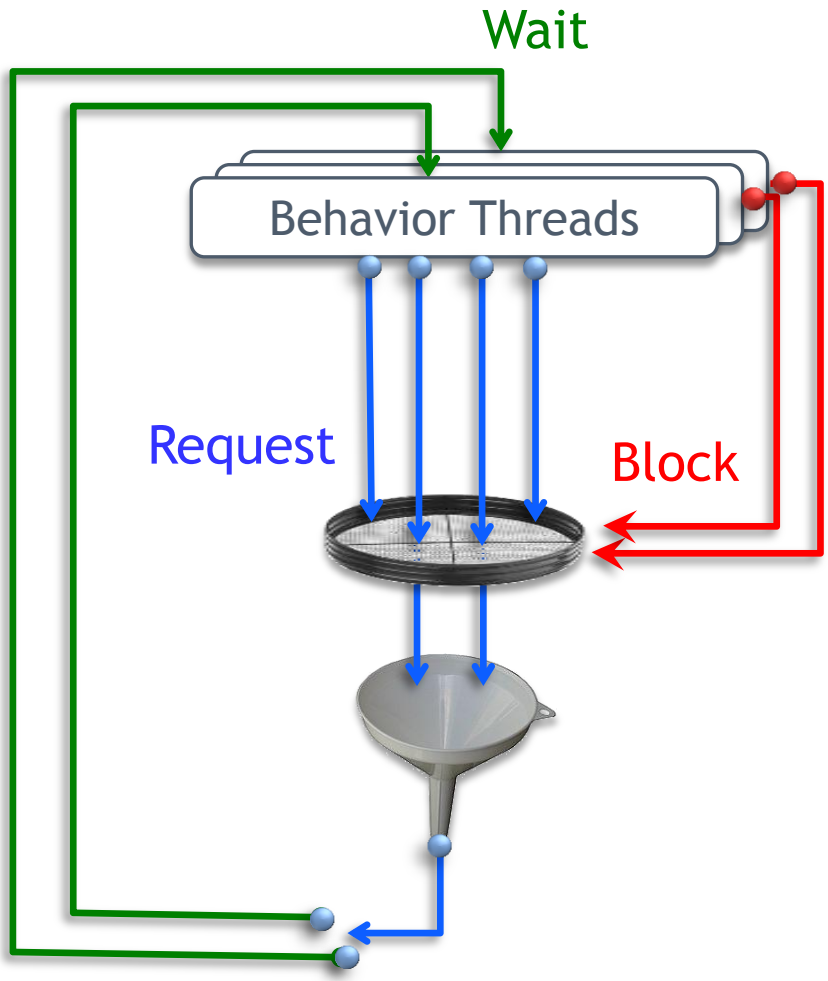
Behavior execution cycle



Behavior execution cycle



Behavior execution cycle



Interweaving programmed scenarios

```
AddHotFiveTimes() {
   for i=1 to 5 {
      bSync(request=addHot, wait-for=none, block=none);
   }
}
```

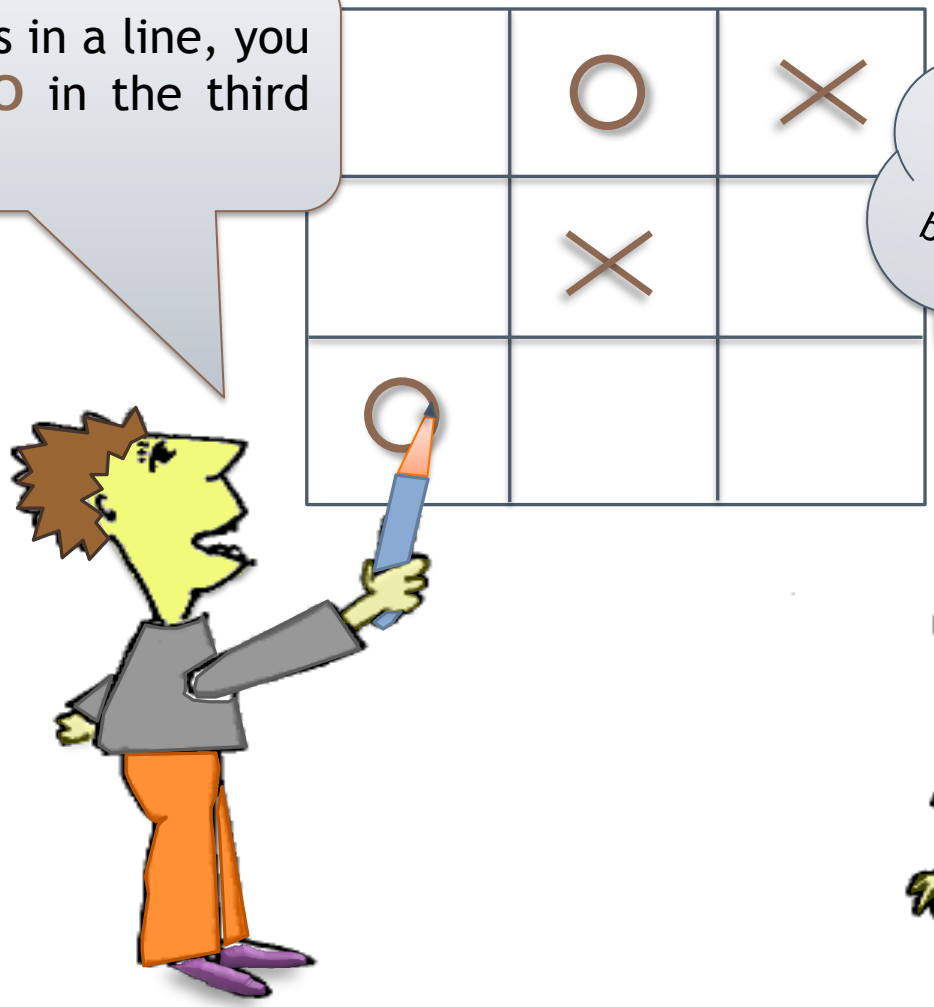
```
AddColdFiveTimes() {
   for i=1 to 5 {
      bSync(request=addCold, wait-for=none, block=none);
   }
}
```

```
Interleave() {
   forever {
      bSync(request=none, wait-for=addHot, block=addCold);
      bSync(request=none, wait-for=addCold, block=addHot);
   }
}
```

addHot
addCold
addHot
addCold
addHot
addCold
addHot
addCold
addHot
addCold

Alignment of code modules with requirements

When I put two Xs in a line, you need to put an O in the third square



A program playing Tic-Tac-Toe

```
EnforceTurns() {
  forever {
    bSync(request=none, wait-for=XMove, block=OMove);
    bSync(request=none, wait-for=OMove, block=XMove);
  }
}
```

⋮

```
DetectTie() {
  for i = 1 to 9 {
      bSync(request=none, wait-for=AnyMove, block=none);
  }
  bSync(request=DeclareTie,  wait-for=none, block=none);
}
```

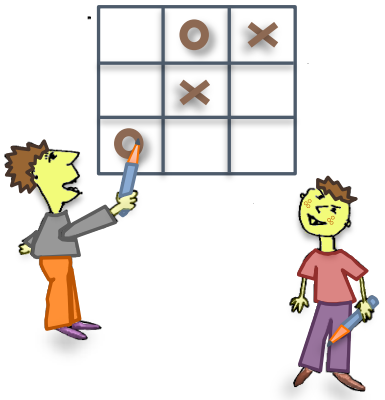
Rules of the game

⋮

```
AddThirdO () {
      bSync(request=none, wait-for=O⟨1,3⟩, block=none);
      bSync(request=none, wait-for=O⟨2,2⟩, block=none);
      bSync(request=O⟨3,1⟩, wait-for=none, block=none);
}
```

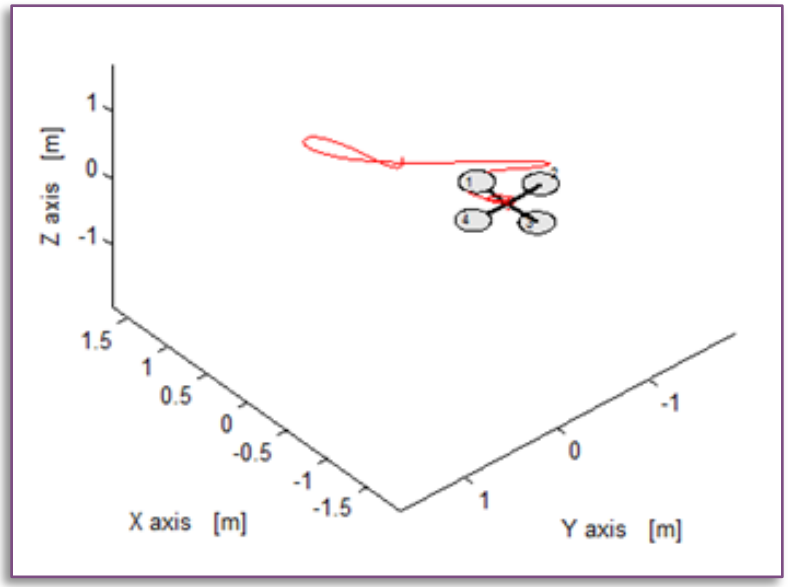
⋮

```
PreventThirdX() {
      bSync(request=none, wait-for=X⟨1,3⟩, block=none);
      bSync(request=none, wait-for=X⟨2,2⟩, block=none);
      bSync(request=O⟨3,1⟩, wait-for=none, block=none);
}
```



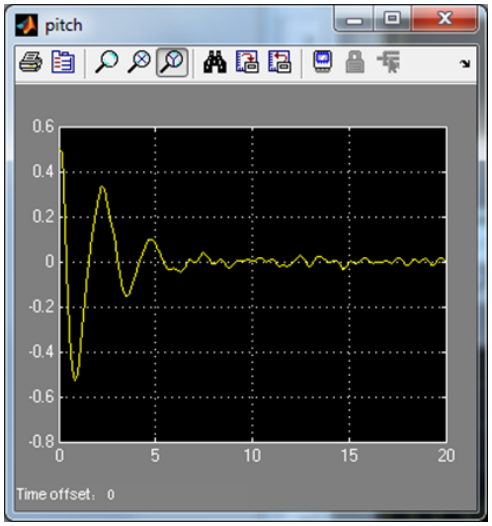
Strategies

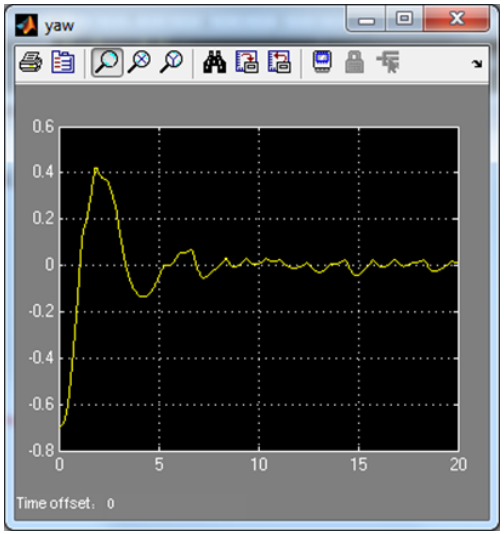
Stabilizing a quadrotor - behaviorally

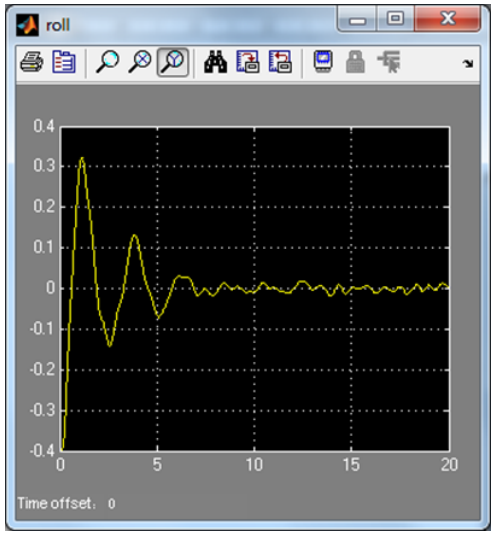


Results of applying the quadrotor Simulink model of Bouabdalla et al where a linear transformation box is replaced with behavior threads.

Pitch, yaw and roll angles are stabilized after a few seconds.







Behavioral Scripts in JavaScript
Synchronization and Event Selection

bSync implemented as `yield` - available in JS 1.7 (on Firefox)

```
bp.addBThread('Add hot five times',
priority++, function() {
        for (x = 1; x <= 5; x++) {
                yield({ request: ['addHot'],
                        wait: [],
                        block: []
                });
        }
});
```

Behavioral Scripts in JavaScript
Sensors and Actuators

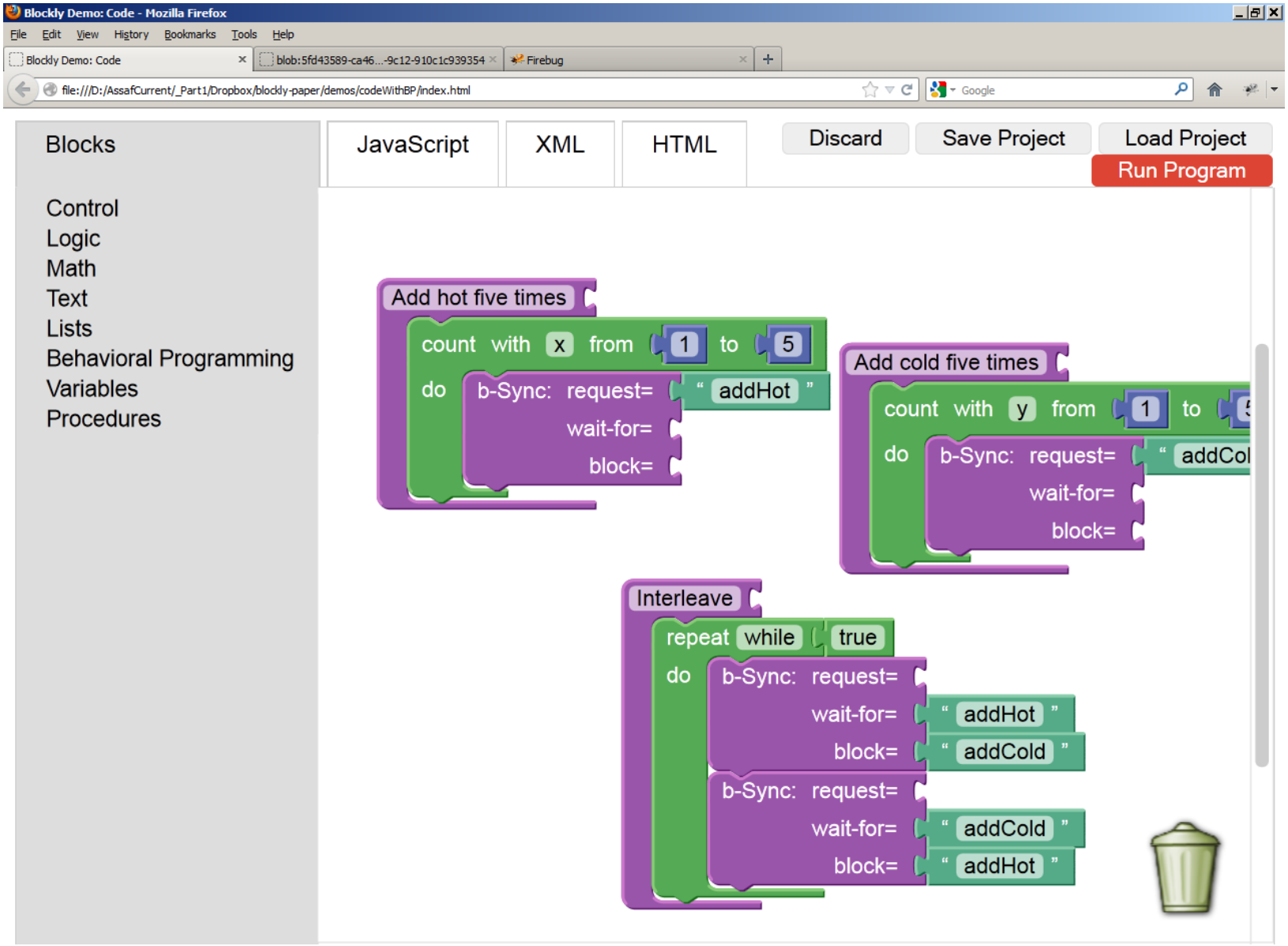
```
<input
    value="StartGame"
    type="button"
    onclick="trigger('StartGame');"
    style="position:relative;background-color:LightPink;"
/>
```

```
<center style="font-size: x-large"

 when_addHot  = 'innerHTML+="<span …>addHot</span><br>"'

 when_addCold = 'innerHTML += "<span …">addCold</span><br>"'
>
```

BP in Google's Blockly



DEMO

Benefits of Programming behaviorally

NATURAL DEVELOPMENT

Small parallel scenarios are natural

Incrementality developed

No need for complex state management

Implementing scenario-based programming in languages such as JavaScript allows for integration with other programming metaphors

Each new required behavior

(or game rule, or strategy, etc.)

is added in a separate b-thread

without changing existing code

Research

- Programming languages and idioms
- Compositional model-checking and verification
- Adaptivity and learning
- Automatic repair
- Comprehension and visualization
- Programming platforms
- Applications in robotics and hybrid control
- Natural language
- Performance and scalability
- . . .

Team members

Past and present – more or less in chronological order – updated 6/2012.

- **David Harel (*)**
- Rami Marelly
- Hillel Kugler
- Shahar Maoz
- Itai Segall
- Yoram Atir
- Asaf Kleinbrot
- Dan Barak
- Avital Sadot
- Amir Kantor
- Michal Gordon
- Tal Berger
- Smadar Szekely (*)
- Assaf Marron (*)
- Gera Weiss (*)
- Daniel Barkan
- Guy Weiss
- Yaarit Natan
- Amir Nissim
- Robby Lampert
- Guy Wiener
- Yaniv Sa'ar
- Nir Eitan
- Guy Katz
- Michael Bar-Sinai
- Moshe Weinstock
- Ronen Brafman

Weizmann Institute of Science

Ben Gurion University

(*) contact persons for the team

Behavioral Programming
A language-independent approach to natural incremental development

For systems where complexity stems from the need to coordinate multiple simple behaviors