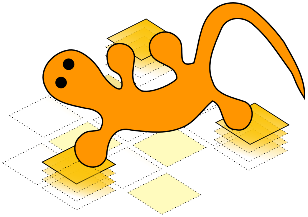


Towards an improved framework for YaMoR

Kevin Drapel - Cyril Jaquier

21 juin 2005

- The YaMoR project

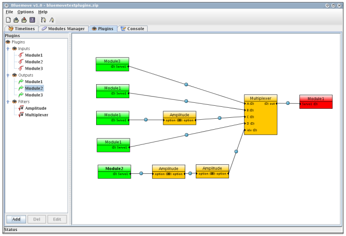
1. Bluemove
   - Real-time control
2. Simulator
   - Eve
3. Closing words
   - Future development
   - Conclusion
   - Questions

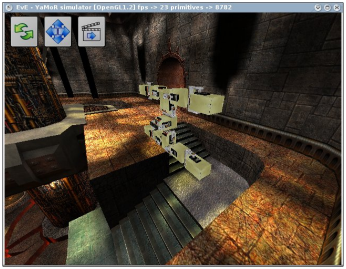
Previous work

YaMoR is a modular robot composed of wireless modules. A module contains three boards : a *Xilinx* FPGA, a *Zeevo* Bluetooth chip and a power board.

During the winter semester, we developed a Java software called Bluemove to remotely control the modules from a computer.

The YaMoR project




EvE - YaMoR simulator [OpenGL1.2] fps -> 23 primitives -> 8782

Project description

We had to :

- Improve Bluemove and add a real-time control system with scripting
- Design a simulator receiving orders from Bluemove
- Integrate the FPGA programming project of Jérôme Maye into Bluemove

Project description

We had to :

- Improve Bluemove and add a real-time control system with scripting
- Design a simulator receiving orders from Bluemove
- Integrate the FPGA programming project of Jérôme Maye into Bluemove

Project description

We had to :

- Improve Bluemove and add a real-time control system with scripting
- Design a simulator receiving orders from Bluemove
- Integrate the FPGA programming project of Jérôme Maye into Bluemove

Real-time control in Bluemove

Our real-time control uses plugins which act like generators, filters, modifiers. It behaves like an interactive signal processing box.

Main features :

- Organized in a graph with connected nodes representing the different plugins
- Each plugin can be fully configured, new options can be added
- Each plugin has its own script (BeanShell language)
- Plugins can receive keys events in real-time

Real-time control in Bluemove

Our real-time control uses plugins which act like generators, filters, modifiers. It behaves like an interactive signal processing box.

Main features :

- Organized in a graph with connected nodes representing the different plugins
- Each plugin can be fully configured, new options can be added
- Each plugin has its own script (BeanShell language)
- Plugins can receive keys events in real-time

Real-time control in Bluemove

Our real-time control uses plugins which act like generators, filters, modifiers. It behaves like an interactive signal processing box.

Main features :

- Organized in a graph with connected nodes representing the different plugins
- Each plugin can be fully configured, new options can be added
- Each plugin has its own script (BeanShell language)
- Plugins can receive keys events in real-time

Real-time control in Bluemove

Our real-time control uses plugins which act like generators, filters, modifiers. It behaves like an interactive signal processing box.

Main features :

- Organized in a graph with connected nodes representing the different plugins
- Each plugin can be fully configured, new options can be added
- Each plugin has its own script (BeanShell language)
- Plugins can receive keys events in real-time

Real-time control in Bluemove

Our real-time control uses plugins which act like generators, filters, modifiers. It behaves like an interactive signal processing box.

Main features :

- Organized in a graph with connected nodes representing the different plugins
- Each plugin can be fully configured, new options can be added
- Each plugin has its own script (BeanShell language)
- Plugins can receive keys events in real-time

Graph layout

The user adds new plugins on a panel and connects the nodes together. The values are transmitted along the links and finally sent to the modules.

Different types of plugins can be created :

- Inputs : generators, trajectories
- Filters : processors, modifiers, multiplexers, etc.
- Outputs : modules

Constraint : no cycles allowed in the graph !

Graph layout

The user adds new plugins on a panel and connects the nodes together. The values are transmitted along the links and finally sent to the modules.

Different types of plugins can be created :

- Inputs : generators, trajectories
- Filters : processors, modifiers, multiplexers, etc.
- Outputs : modules

Constraint : no cycles allowed in the graph !

Graph layout

The user adds new plugins on a panel and connects the nodes together. The values are transmitted along the links and finally sent to the modules.

Different types of plugins can be created :

- Inputs : generators, trajectories
- Filters : processors, modifiers, multiplexers, etc.
- Outputs : modules

Constraint : no cycles allowed in the graph !

Graph layout

The user adds new plugins on a panel and connects the nodes together. The values are transmitted along the links and finally sent to the modules.

Different types of plugins can be created :

- Inputs : generators, trajectories
- Filters : processors, modifiers, multiplexers, etc.
- Outputs : modules

Constraint : no cycles allowed in the graph !

Graph layout

The user adds new plugins on a panel and connects the nodes together. The values are transmitted along the links and finally sent to the modules.

Different types of plugins can be created :

- Inputs : generators, trajectories
- Filters : processors, modifiers, multiplexers, etc.
- Outputs : modules

Constraint : no cycles allowed in the graph !

BeanShell scripts

Each plugin can be fully configured and programmed using a BeanShell script. BeanShell is a subset of Java language, powerful and easy to learn.

- Options are limited to a set of common types
- Internal variables for state machines
- Scripts can be evaluated several times (for integration)
- Scripts have access to pressed keys and current frame

Recently accepted by the Java Community Process as JSR-274. Will be included in the J2SE.

BeanShell scripts

Each plugin can be fully configured and programmed using a BeanShell script. BeanShell is a subset of Java language, powerful and easy to learn.

- Options are limited to a set of common types
- Internal variables for state machines
- Scripts can be evaluated several times (for integration)
- Scripts have access to pressed keys and current frame

Recently accepted by the Java Community Process as JSR-274. Will be included in the J2SE.

BeanShell scripts

Each plugin can be fully configured and programmed using a BeanShell script. BeanShell is a subset of Java language, powerful and easy to learn.

- Options are limited to a set of common types
- Internal variables for state machines
- Scripts can be evaluated several times (for integration)
- Scripts have access to pressed keys and current frame

Recently accepted by the Java Community Process as JSR-274. Will be included in the J2SE.

BeanShell scripts

Each plugin can be fully configured and programmed using a BeanShell script. BeanShell is a subset of Java language, powerful and easy to learn.

- Options are limited to a set of common types
- Internal variables for state machines
- Scripts can be evaluated several times (for integration)
- Scripts have access to pressed keys and current frame

Recently accepted by the Java Community Process as JSR-274. Will be included in the J2SE.

BeanShell scripts

Each plugin can be fully configured and programmed using a BeanShell script. BeanShell is a subset of Java language, powerful and easy to learn.

- Options are limited to a set of common types
- Internal variables for state machines
- Scripts can be evaluated several times (for integration)
- Scripts have access to pressed keys and current frame

Recently accepted by the Java Community Process as JSR-274. Will be included in the J2SE.

BeanShell scripts

Each plugin can be fully configured and programmed using a BeanShell script. BeanShell is a subset of Java language, powerful and easy to learn.

- Options are limited to a set of common types
- Internal variables for state machines
- Scripts can be evaluated several times (for integration)
- Scripts have access to pressed keys and current frame

Recently accepted by the Java Community Process as JSR-274. Will be included in the J2SE.

Other features in Bluemove

We added other useful features in Bluemove :

- Function generator in the timelines manager
  For example : $sin(t*2)+exp(t)$
- Remote programming of the FPGA via Bluetooth (Jérome Maye project)

We also fixed bugs that we discovered during the inauguration and further tests.

Other features in Bluemove

We added other useful features in Bluemove :

- Function generator in the timelines manager
  For example : $sin(t * 2) + exp(t)$
- Remote programming of the FPGA via Bluetooth (Jérome Maye project)

We also fixed bugs that we discovered during the inauguration and further tests.

Other features in Bluemove

We added other useful features in Bluemove :

- Function generator in the timelines manager
  For example : $sin(t * 2) + exp(t)$
- Remote programming of the FPGA via Bluetooth (Jérome Maye project)

We also fixed bugs that we discovered during the inauguration and further tests.

Other features in Bluemove

We added other useful features in Bluemove :

- Function generator in the timelines manager
  For example : $sin(t * 2) + exp(t)$
- Remote programming of the FPGA via Bluetooth (Jérome Maye project)

We also fixed bugs that we discovered during the inauguration and further tests.

Eve - The YaMoR simulator

A simulator that could receive the positions sent by Bluemove would be useful to test and validate different configurations.
Main features of our simulator "Eve" :

- C++ code
- Physics and collisions (ODE)
- Polyvalent and portable 3D engine (Irrlicht)
- Interoperation with remote calls (XML-RPC)
- XML files for configuration

Eve - The YaMoR simulator

A simulator that could receive the positions sent by Bluemove would be useful to test and validate different configurations.
Main features of our simulator "Eve" :

- C++ code
- Physics and collisions (ODE)
- Polyvalent and portable 3D engine (Irrlicht)
- Interoperation with remote calls (XML-RPC)
- XML files for configuration

Eve - The YaMoR simulator

A simulator that could receive the positions sent by Bluemove would be useful to test and validate different configurations.
Main features of our simulator "Eve" :

- C++ code
- Physics and collisions (ODE)
- Polyvalent and portable 3D engine (Irrlicht)
- Interoperation with remote calls (XML-RPC)
- XML files for configuration

Eve - The YaMoR simulator

A simulator that could receive the positions sent by Bluemove would be useful to test and validate different configurations. Main features of our simulator "Eve" :

- C++ code
- Physics and collisions (ODE)
- Polyvalent and portable 3D engine (Irrlicht)
- Interoperation with remote calls (XML-RPC)
- XML files for configuration

Eve - The YaMoR simulator

A simulator that could receive the positions sent by Bluemove would be useful to test and validate different configurations.
Main features of our simulator "Eve" :

- C++ code
- Physics and collisions (ODE)
- Polyvalent and portable 3D engine (Irrlicht)
- Interoperation with remote calls (XML-RPC)
- XML files for configuration

3D world and physics

Thanks to Irrlicht and ODE, we tried to produce something realistic.

- No more flat floors : Quake III levels
- Other objects or obstacles can be added to the scene
- Collisions with the world, robot physics with ODE

Quake III : a good game and a good test for YaMoR (stairs, holes, bridges, etc)

3D world and physics

Thanks to Irrlicht and ODE, we tried to produce something realistic.

- No more flat floors : Quake III levels
- Other objects or obstacles can be added to the scene
- Collisions with the world, robot physics with ODE

Quake III : a good game and a good test for YaMoR (stairs, holes, bridges, etc)

3D world and physics

Thanks to Irrlicht and ODE, we tried to produce something realistic.

- No more flat floors : Quake III levels
- Other objects or obstacles can be added to the scene
- Collisions with the world, robot physics with ODE

Quake III : a good game and a good test for YaMoR (stairs, holes, bridges, etc)

3D world and physics

Thanks to Irrlicht and ODE, we tried to produce something realistic.

- No more flat floors : Quake III levels
- Other objects or obstacles can be added to the scene
- Collisions with the world, robot physics with ODE

Quake III : a good game and a good test for YaMoR (stairs, holes, bridges, etc)

Communication and configuration

The simulator can run on a computer while Bluemove sends data from another station. It is also easy to change the shape of the robot by giving the name of the two modules and the sides where they are connected.

- Bluemove uses remote calls to communicate with Eve.
- Used protocol : XML-RPC == XML files encoded and sent on HTTP
- Robot configuration : XML with modules and connections

Drawback : ping on network must be low and stable to ensure smooth motion

Communication and configuration

The simulator can run on a computer while Bluemove sends data from another station. It is also easy to change the shape of the robot by giving the name of the two modules and the sides where they are connected.

- Bluemove uses remote calls to communicate with Eve.
- Used protocol : XML-RPC == XML files encoded and sent on HTTP
- Robot configuration : XML with modules and connections

Drawback : ping on network must be low and stable to ensure smooth motion

Communication and configuration

The simulator can run on a computer while Bluemove sends data from another station. It is also easy to change the shape of the robot by giving the name of the two modules and the sides where they are connected.

- Bluemove uses remote calls to communicate with Eve.
- Used protocol : XML-RPC == XML files encoded and sent on HTTP
- Robot configuration : XML with modules and connections

Drawback : ping on network must be low and stable to ensure smooth motion

Communication and configuration

The simulator can run on a computer while Bluemove sends data from another station. It is also easy to change the shape of the robot by giving the name of the two modules and the sides where they are connected.

- Bluemove uses remote calls to communicate with Eve.
- Used protocol : XML-RPC == XML files encoded and sent on HTTP
- Robot configuration : XML with modules and connections

Drawback : ping on network must be low and stable to ensure smooth motion

Video

Future development

This first prototype clearly suffers of many problems. The 2nd generation should try to fix them.

- Modular robotics problem : weak velcro connectors
- Security problem : serial batteries
- Hardware problem : too many wires

And of course : add sensors

Future development

This first prototype clearly suffers of many problems. The 2nd generation should try to fix them.

- Modular robotics problem : weak velcro connectors
- Security problem : serial batteries
- Hardware problem : too many wires

And of course : add sensors

Future development

This first prototype clearly suffers of many problems. The 2nd generation should try to fix them.

- Modular robotics problem : weak velcro connectors
- Security problem : serial batteries
- Hardware problem : too many wires

And of course : add sensors

Future development

This first prototype clearly suffers of many problems. The 2nd generation should try to fix them.

- Modular robotics problem : weak velcro connectors
- Security problem : serial batteries
- Hardware problem : too many wires

And of course : add sensors

Conclusion

We have developed a complete framework to test and play around with YaMoR. Unfortunately, we always missed more working modules to fully test our system.

- Traveling modules : Yverdon, Zürich, EPFL, etc.
- Unreliable electronics : shortcuts, CRC errors, etc.

From the wireless point of view, YaMoR does a good job. However, we hope the 2nd generation will be more reliable than the first prototype.

Conclusion

We have developed a complete framework to test and play around with YaMoR. Unfortunately, we always missed more working modules to fully test our system.

- Traveling modules : Yverdon, Zürich, EPFL, etc.
- Unreliable electronics : shortcuts, CRC errors, etc.

From the wireless point of view, YaMoR does a good job. However, we hope the 2nd generation will be more reliable than the first prototype.

Conclusion

We have developed a complete framework to test and play around with YaMoR. Unfortunately, we always missed more working modules to fully test our system.

- Traveling modules : Yverdon, Zürich, EPFL, etc.
- Unreliable electronics : shortcuts, CRC errors, etc.

From the wireless point of view, YaMoR does a good job. However, we hope the 2nd generation will be more reliable than the first prototype.

Questions ?