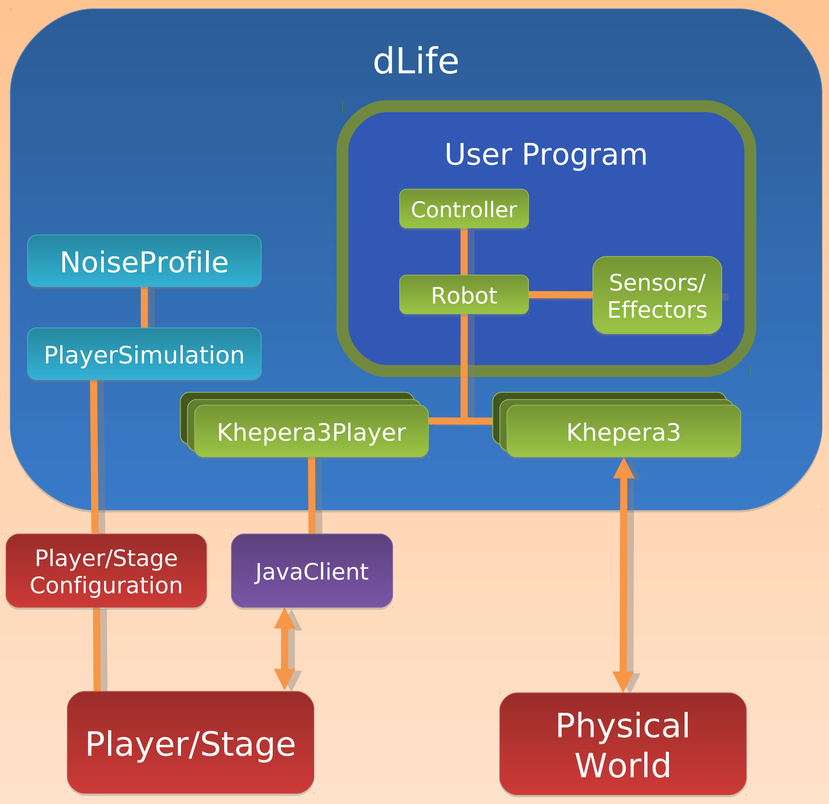
Integration Between dLife and the Player/Stage Robotics Simulation System

Russell Toris, Dickinson College ‘11
Advisor: Prof. Grant Braught

Abstract

The areas of artificial life, artificial intelligence, and robotics are progressive fields with nearly limitless possibilities. The dLife software package is a Java library developed by Professor Grant Braught of Dickinson College aimed specifically at these three fields. dLife includes extensive packages for key components such as neural networks, genetic algorithms, basic computer vision and robot control. The packages for robot control, however have been limited to working with physical robots. At times working with physical robots can be impractical (e.g. too time consuming / cost prohibitive). To address these situations I have developed additional packages in dLife that create a bridge between its resources, and those of Player/Stage (an open source robotics platform used for simulations). These additional packages produce a near seamless integration between the physical robots that dLife already supports (the **Pioneer**, **Hemisson**, and **Khepera III** robots), and simulated versions in Player/Stage.

dLife Model with Player/Stage



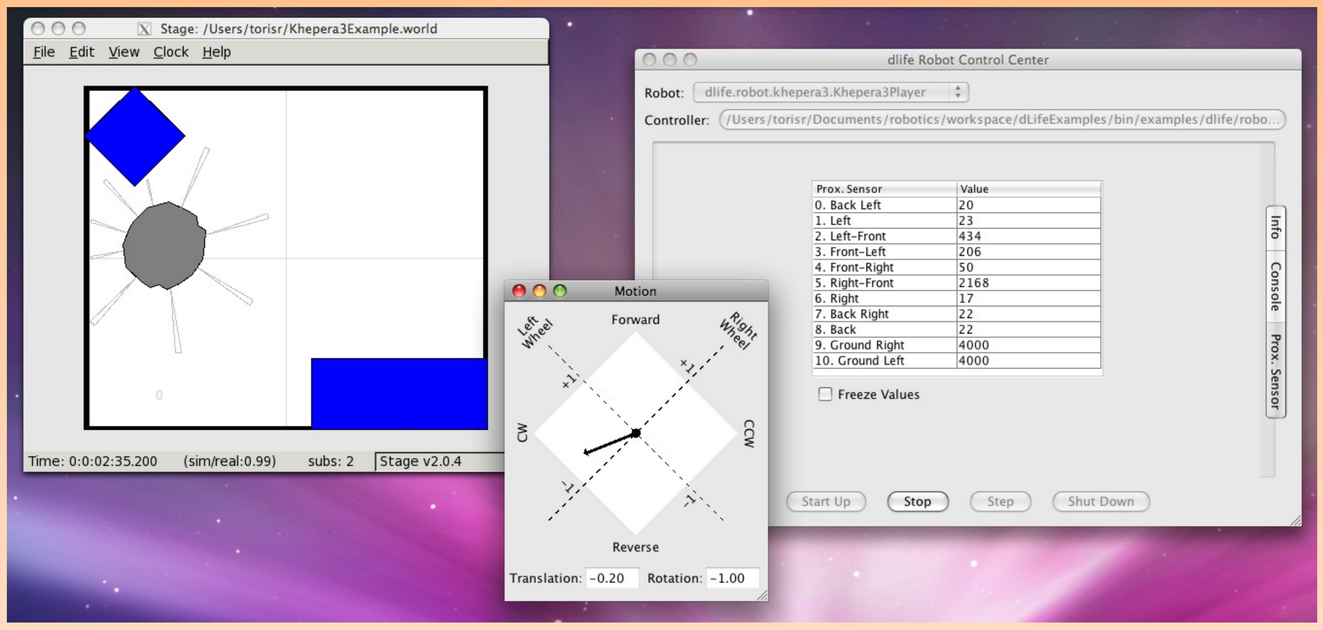
dLife & Player/Stage Integration

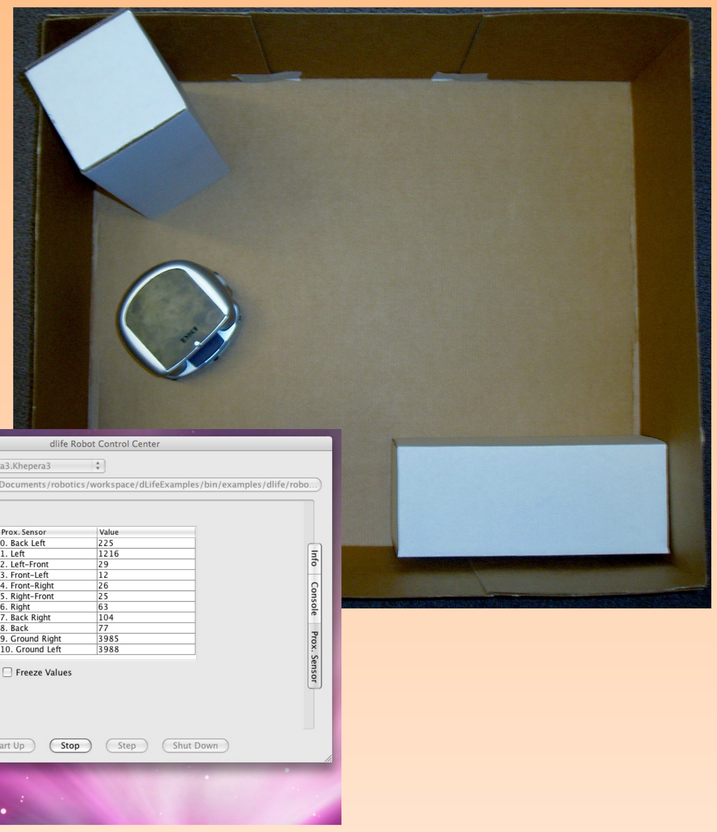
PlayerRobot objects (e.g. **Khepera3Player**) will receive formatted commands from **Sensor** and **Effector** objects. They then interpret these commands and make appropriate calls to Player/Stage using the **JavaClient** package. When returning sensor data, the **PlayerRobot** class formats the data in the format its physical equivalent would return. Optionally, noise can be added using a **NoiseProfile** object provided during simulation creation.

Simulation Creation

The capability to build Player/Stage simulations has been added to dLife using the **PlayerSimulation** class. This creates all the necessary Player/Stage configuration with simple method calls. This allows the user to easily customize the dimensions of the world, which image will be used to define the world, add and place pucks and robots, as well as attach specific sensors to individual robots. Furthermore, dLife also allows each simulated sensor to have a custom **NoiseProfile**, allowing the physical robot's sensors to be modeled more accurately.

Physical vs. Simulated Environments





dLife can now use Player/Stage (above) to model real world scenarios (right). The simulated and physical robots are being controlled by the same **Controller** class.

API Consistency

Care has been taken to ensure that there is no change in API between interacting with a simulated robot in Player/Stage and its physical equivalent. Thus, any user program interacting with a simulated robot via dLife can also interact with the corresponding physical robot. When interacting with any robot, the user will be interacting with the **Controller** and **Robot** classes. The user will communicate with the robot by making calls to its appropriate **Sensor** and **Effector** classes.

Integration Significance

The addition of this Player/Stage interface makes dLife an ideal platform for conducting multiple Player/Stage simulations in parallel. The availability of parallel simulations makes it practical to study the use simulated evolution for the design of controllers for the physical robots. Such simulations may also provide a viable means for testing theories and hypothesis about biological evolution.

Sources

JavaClient2 (2007). Javaclient for Player/Stage. Accessed 3/1/2010: http://java-player.sourceforge.net/

Player/Stage (2010). Player Project. Accessed 3/1/2010: http://playerstage.sourceforge.net/

Acknowledgements

The funding needed to help support this project was graciously awarded by the Dickinson College Research and Development Committee. I would like to thank Prof. Grant Braught of Dickinson College for his guidance and support throughout this project. Additionally, I would like to thank the developers of both Player/Stage and JavaClient, without which this project would not be possible.