Projects on the Move

Building robots is an expensive hobby but too interesting to leave to a small group of elite researchers. Peekabot saves you the investment in hardware and simulates a brave new robot world. Also, we'll look at Gbrainy, a program to train your gray matter. *By Carsten Schnober*

Utopia or reality? Science fiction or reality report? Plans are to have the current world soccer champions play against a team of robots in the year 2050, if the initiators of the annual world championship in robot soccer, the Robocup, have their way. The game is played by FIFA rules, and the Robocup Federation is quite certain that the mechanical players will win against the human team.

Critics might reply that humans have more serious problems that robots could help solve, but soccer is very good for experimenting in many respects. Ball games require strategic planning and dynamic learning, the ability to identify various objects, fast and agile movements, reactions to external conditions that change quickly, and team coordination. All of these requirements are typical for daily life – and not just on the soccer pitch.

Virtualizing the Virtual

Software developers try to control the given hardware in the best possible way to solve the problem at hand, whereas hardware just tries to put together the available parts to create a machine that is capable of solving any problems that might occur. A combination of the two is the quintessence of the academic discipline of robotics.

One drawback is that hardware changes always involve some costs. Even minor changes can consume valuable time and resources if the robotics scientist wants to try out various combinations of sub-assemblies. To save time and money, developers rely on simulations and test their software on virtual hardware, and in a similarly virtual environment.

The free Peekabot [1] software shows what the robot world could look like,

with no need to own a real robot for experiments (Figure 1). The program visualizes completely simulated environments and reflects the movement of recorded actions by physical robots. Thanks to the program's client-server architecture, you can even monitor genuine robots in action. In this case, the virtual world simply has to model the robot's physical environment. The server provides the virtual world and also handles graphics output.

In the Peekabot main window, the user navigates a three-dimensional space using a camera. The client is a single robot: It sets up a TCP/IP connection to the server and goes about its business in the virtual world on the basis of the instructions given by the control software. The benefit is that it doesn't matter whether the server is running on the same computer or on another computer with a network connection.

Do-it-Yourself Robots

The robots, which can act individually or in groups, are not made of chips, sensors, wheels, plastic, and metal; they are made of a comparatively cheap and easily obtained material: C++ code. The Peekabot package includes a `peekabot.hh` header file that gives programmers the required class and function definitions.

If you are designing a client, the `peekabot::PeekabotClient` class is vital. When called with the `-lpeekabot` option, GCC creates a robot from the source code that can be launched as an independent client program. A tutorial [2] provides an excellent introduction to Peekabot programming.

The source code archive contains a great selection of sample material. Developers will find many examples in the `examples/bo-slam` subdirectory, including the simple `bo-slam` robot, which is ready to run immediately after building. A simple `make` call, followed by `./bo-slam` will dump the robot into a virtual world running on your Peekabot server. Unfortunately, bo-slam is a fairly dumb artificial buddy, who mainly runs around in circles without any real targets or orientation (Figures 2 and 3).

To set the scene, you don't need any knowledge of programming. World builders simply describe the robot homes in an XML-based configuration file (scene file). Additionally, you can define objects and their positions and design the appearance of the robots.

The XML configuration properties relate to 3D models that Peekabot needs to have in its own format. Users can create models of robots or other objects in external programs, the only condition being that the software can output Wavefront files. Blender [3], another free program, can do this, for example. The `obj2pbmf` from the Peekabot package then converts the models into the program's own format. A collection of 3D models is available [4].

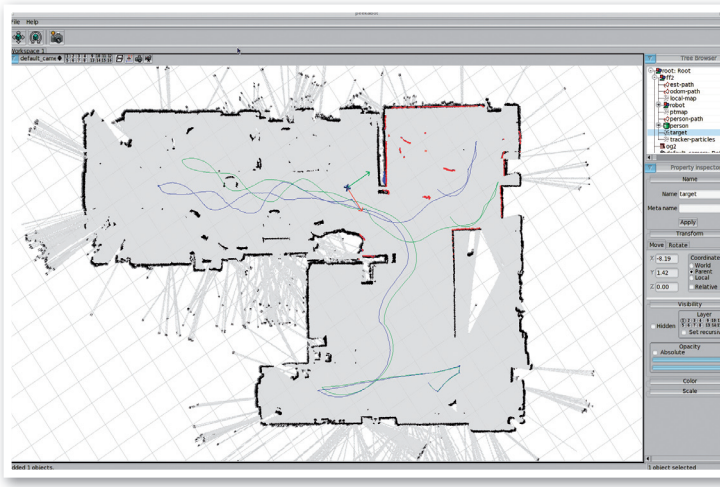

Figure 1: Robot programming without a robot. Peekabot visualizes events in a virtual 3D space. The environment itself is defined by XML configuration files.

Robot Fans Wanted

Staffan Gimåker and Anders Boberg developed Peekabot back in 2006 as part of their university course at the Royal University of Technology in Stockholm. Gimåker has maintained the GPL'd project ever since and is now looking for people to help out.

To develop a Peekabot robot, you need a sound working knowledge of C++ – this is also a requirement for improving the program by defining your own patches. However, if you don't feel up to the complexity of the Peekabot code, you can still help the project by returning bug reports, providing feedback, and making suggestions for improvements. The developer will be happy to accept your input, and he is interested in finding out who uses Peekabot in a production environment and the kind of scenarios in which the virtual robots act.

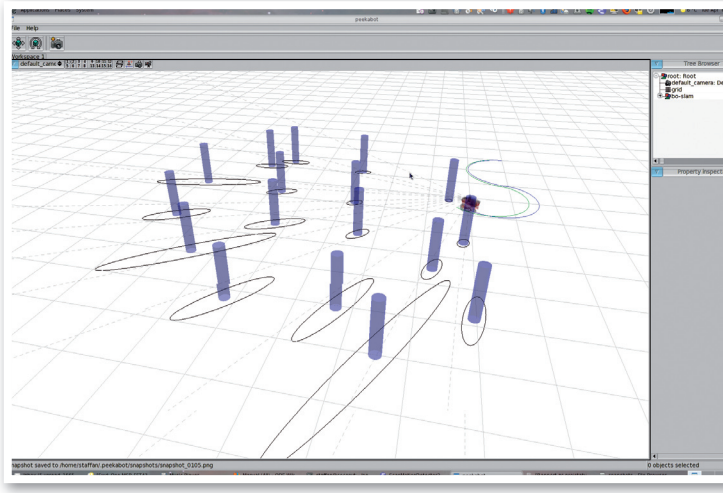

Figure 2: A bird's eye view of the bo-slam robot. The camera is in the center; the lines show the path the robot has taken and the influence that external objects have had on it.

Brain Jogging

The human brain is pretty much like any other organ: It needs

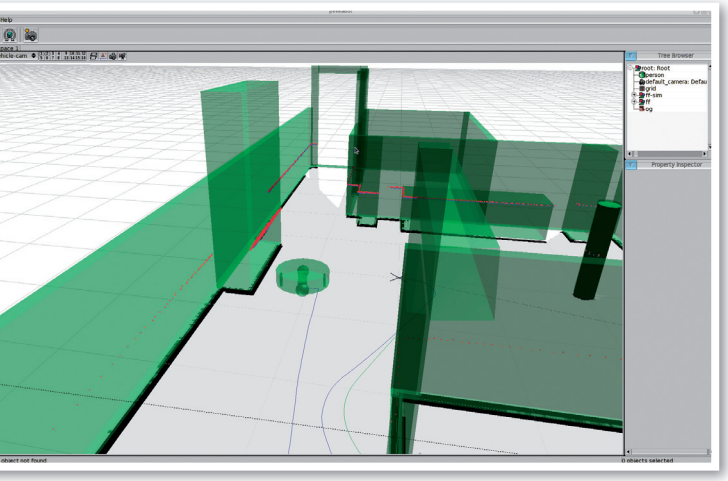

Figure 3: Peekabot offers various camera views. The alignment perspective shown here displays two lines for the expected and actual route taken by the robot.

constant exercise to keep it fit.

Spanish developer Jordi Mas is helping keep users of the Gnome desktop fit and providing a distraction from the daily grind. His Gbrainy [5] program keeps your gray matter on the move (see Figure 4).

The software-based trainer challenges your brain in three categories. *Logic puzzles* challenge the user to identify sequences of numbers. Figure 5 shows an example in which the user is asked to provide the correct answer to a logic problem.

Mental arithmetic turns out to be more complex than the innocent-sounding name would suggest. Gbrainy doesn't just ask you to complete various mathematical operations with given numbers but also poses tricky sums that go well beyond the rule of three.

When it comes to *memory*, Gbrainy starts by presenting various types of objects, such as numbers and geometric shapes. The user is asked to memorize these in a fairly short time. But again, the tasks posed are not exactly easy.

Instead of simply prompting the user to name the objects in their original view, Gbrainy might ask you to state the number of two-figure integers or describe the way a specific shape is positioned.

Future Plans for the Trainer

After developing mental skills, Gbrainy players can give some of their newly won brain power back to the project. The developer asks users for help if they find any bugs in the program and for direct help in the form of feedback on the game, drafts for new puzzles, or even full implementations of new puzzles. Translators are welcome, too. The manual is available in five languages right now.

For the future, Jordi Mas is planning to add text-based puzzles and to add Sudoku and IQ tests similar to those used by Mensa. Additionally, Gbrainy will also run on portable platforms like Maemo [6] and Hildon [7], let you select the correct solution by clicking, and be accompanied by documentation. The print function that Mas plans would also make sense; after all, solving all these puzzles on screen isn't going to help your concentration.

Incidentally, Gbrainy developer Jordi Mas is no newcomer to the open source scene. He has contributed programming work to the AbiWord project and helped develop the Catalan spellchecker for previous versions of OpenOffice.

Besides Gbrainy, Mas has also developed the Mistelix [8] DVD creator, and he contributed to Mono, the free implementation of DotNet, for many years on a volunteer basis. Little surprise then that he implemented Gbrainy in Mono and C#. ■■■

Figure 4: Gbrainy trains your gray matter. The tool trains logical thinking, mental arithmetic and your memory.

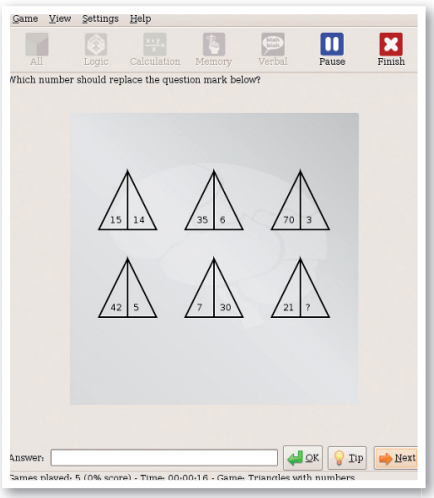

Figure 5: Logic problems exercise your ability to identify patterns and extrapolate series.

INFO

[1] Peekabot: *http://www.peekabot.org*
[2] Peekabot documentation: *http://www.peekabot.org/doc/latest*
[3] Blender: *http://www.blender.org*
[4] Peekabot model repository: *http://sourceforge.net/apps/mediawiki/peekabot/index.php?title=Model_repository*
[5] Gbrainy: *http://live.gnome.org/gbrainy*
[6] Maemo: *http://maemo.org*
[7] Hildon: *http://live.gnome.org/Hildon*
[8] Mistelix: *http://www.mistelix.org*