Introduction

I was initially inspired by Steve Grand's "Creatures 3", an "artificial life" computer game featuring cute little "norn" animals that have to be carefully cared for – or they'll die. Later, looking into how the norns were programmed, I was impressed with the way that their attention's drawn to objects in their environment and the way that their emotions then drive their behaviors. So I decided to bring one to life in the form of a PC based AI robot with vision, speech recognition and text to speech capabilities – in fact, "Leaf" is named for the first norn that I studied and "raised".

AI Lisp

For me, Lisp was the higher-level programming language of choice. In large part because I'm familiar with the language, but also because Lisp is designed for symbol manipulation, object oriented programming and interactive program development. In addition, there is a community of Lisp programmers working in AI development with a large library of code available. I'm specifically using Xanalys LispWorks (free) Personal Edition which has language extensions that allow Lisp interfaces to Windows API's, C/C++ dll's and COM objects.

Vision

Leaf uses a standard webcam and Robin Hewitt's "Mavis" software for vision. At present, Leaf can visually recognize a white circle on the floor. And he can then go to the circle and stand on it. We plan to use this in a navigation contest so that Leaf can find his way back to his "home base". However, this might eventually also be used for him to find his recharging station – once we have one built! Leaf can also "see" horizontal lines in an image – and that helps him to detect objects like a table edge that his sonar might have missed. I'm still working on his "vision based obstacle avoidance" routines – but I hope to have that implemented soon. In addition, we're also implementing face recognition routines so that Leaf will be able to recognize his friends just by looking at them!

Speech Recognition and the "Listening Thread"

Leaf runs MicroSoft's Speech SAPI5 SDK in a separate "Listening Thread" to listen continuously for phrases and commands - the Lisp to SAPI5 interface was written by John D. Corbett. The speech recognition program has access to a 50,000 to 60,000 word dictionary, but usually restricts word searches to a "command library" of only a few dozen phrases – which significantly improves recognition accuracy… and is relatively speaker independent. Specific phrases, such as "Leaf forward", are then automatically passed on to the Main Control Loop for interpretation and processing. However, since Leaf is always listening, he might hear you talking in the background… and take off!

I've therefore built in a command – "Leaf pause listening" – that tells him to ignore anything that he hears until he's told "Leaf start listening again".

Main Control Loop

Leaf's Main Control Loop is really not much more than a series of conditional statements that then help to determine Leaf's next activity.

For example, if Leaf's Main Control Loop just matched the phrase "Leaf forward", Leaf will first halt any running "activity threads" and will then open a new activity thread to execute the forward command. Allowing only one activity thread at a time means that Leaf won't be trying to "go forward" and "go backward" at the same time!

In addition, the "activities" can be quite a bit more complex. For example, if Leaf hears "Leaf this is Bruce", then a "behavior script" is executed which changes Leaf's mood and emotions to reflect his memory of past experiences with me.

And finally, if no verbal commands have been given, the Main Control Loop checks on Leaf's current emotional state. So, if for example Leaf's getting tired, the Main Control Loop will execute a "tired" behavior – and Leaf might then take a little nap.

Emotion Driven Behaviors

Leaf in fact has several different emotions – happy, sad, crowded, bored, surprised, etc. And he can feel several different things at the same time – so he might be mostly happy, but slightly surprised and just a little crowded. His emotions can then drive his behaviors. So if he's feeling bored, he'll pick something at random to do – maybe turn around in a circle… or maybe make a noise. If he's feeling crowded, he'll try to move to a different – less crowded – location. So, in addition to responding to verbal commands, Leaf also then acts independently according to his emotional "needs" and "desires".

Emotion Thread and Personality

Leaf's emotions are in fact always changing. Sometimes in response to his interactions with people – ie. "happy". Sometimes in response to his interactions with the environment – ie. "crowded". And sometimes just in response to the passing of time – ie. "bored". This is all handled in the "Emotion Thread" as Lisp code running in parallel with the rest of his programming.

Interestingly, since Leaf's emotional state is constantly being shaped by his environment and by his interactions with people, his "personality" – which isn't specifically programmed in – then evolves as an "emergent" property of these interactions and experiences.

Emotion Based Reinforcement Learning

Leaf's emotions drive his behaviors. However, in addition, his emotions also help him learn about people. It works like this: As you interact with him, his emotions change – if you talk to him and do interesting things, he'll be happy and he'll like you… but if you're boring and repetitious, then he probably won't look forward to seeing you again. And Leaf remembers the emotions he was feeling the last time he saw you. So, the next time he's feeling lonely, he'll use his "imagination" to recall those previous feelings and then he'll decide who he might like to see again – and off he'll go looking for that person!

Simple Reinforcement Learning

Leaf also learns by simple reinforcement. If he's bored, he'll try doing something at random – maybe turn in a circle. If you choose to, you can then reinforce that behavior by praising him with "Leaf that's good". However, if he does something that you don't like – like making a rude noise – then you can scold him by saying "Leaf that's bad". As he's praised or scolded, he'll remember and change the likelihood of those behaviors – simple reinforcement learning. And soon you'll have him behaving like a good robot!

Face Animation / Text to Speech

Leaf uses the free CSLU Toolkit software package for face animation and text to speech. The CU Animate part of the package displays Leaf's blue dragon face on the laptop monitor complete with a range of emotions and expressions. The Festival program provides text to speech and lip-syncing. Of course, if you can't tell from Leaf's expression, you can always just ask "Leaf what are you feeling?" – and he'll tell you!

Many thanks to John D. Corbett for writing the Lisp interface to the CU Animate / Festival packages. And thanks to the CSLU group for the Toolkit software.

Navigation Control and the Microcontroller

Leaf 's Lisp code communicates with Alex Brown's Navigation Control code and the microcontroller using shared memory. Lisp passes commands to the microcontroller by simply passing data to slots in the shared memory. The Navigation Control code reads the slots in shared memory and passes the command data on to the microcontroller over a usb link. The process works in reverse for data sent by the microcontroller to the laptop's shared memory and on to Lisp.

At present, Lisp has access through the microcontroller to read IR, sonar, accelerometer and compass sensors and can command the drive motors, position servo motors and output digital IO commands.

In practice, this all works well enough for Lisp to command the robot to navigate a room while using sonar sensors to avoid obstacles – all in real time!

X10 Home Automation

Leaf can also send X10 commands through a FireCracker RF wireless transmitter connected to the laptop to a receiver plugged into a wall socket to control lights and appliances anywhere in your house. Now not only can you command him to turn lights on and off, Leaf can do it himself if he's left home alone and it's getting dark!

The X10 / FireCracker interface was written by Gary Malolepsy.

And Bill Foard is currently working on a more general version that will have many more options and features.

Conclusion

The Leaf project is the product of many talented contributors. And, as the project continues to grow, we hope that many more of you will join us in our efforts to create an artificial creature!