Step-by-step instructions on how to set up your creature

Allen Parseghian
November 15, 1999

1) Print out the creature library tutorial file (cl.ps) which is located in /home/ll/doc/cl_tutorial directory.
2) Make sure that you belong to group "leg", by typing "groups" at your Xterm prompt. If you type "groups" and it doesn't say "leg", then you need to send an e-mail to allens@ai.mit.edu and ask to be added to group "leg".
3) For the creature library to run faster, you need to use a linux machine. But, before you do that, open up the file ".rhosts" which is in your HOME directory and add the linux machine names that you will be logging into to compile the creature library in that file. If you do not have a ".rhosts" file in your home directory, create one. To add a machine name in the ".rhosts" file, type: "*machine_name username*" e.g. "lv08 hacker" where lv08 is a sample linux machine name and "hacker" is the username of the person who has logged into the machine. You have to make sure that you always add the machine "soleus" to ".rhosts" i.e. have the line: "soleus *username*" in the ".rhosts" file or you might get compilation error later. Also, make sure that you hit return in the last line of the ".rhosts" file. Some linux machines to try are: lv01, lv02, lv03, lv04, lv05, lv06, lv07, lv08, and lv09, where l=L, and 0=zero.
4) Now log into one of the linux machines that you have added to your ".rhosts" file by typing: "slogin *machine_name*" . In your home directory, create a directory and call it any random name that you want e.g. "test_directory". From now on, this will be your creature directory for your particular creature.
5) Go to your creature directory and create your creature and call it anything that you want e.g. "test_creature". Save that file as "create_test_creature.c". Note that whatever name you give to your creature, you have to make sure that you add "create_" before the name of the creature when you save your creating creature file. To learn about how to create a creature, refer to the creature library tutorial.
6) In your creature directory, run the following command to generate a Makefile: "setup_simulation *creature_name*" e.g. "setup_simulation test_creature". Now if you type "ls", there should be 2 files in your creature directory:
   "create_test_creature.c", "Makefile"
7) Before you compile your creature, you might need to run the following command:
   "source /home/ll/.login"
   then compile by typing "make". This will generate all the files that will be needed by creature library to run your simulation. It might take awhile until the process is over, and you might also see some warning messages which you should ignore. As long as you do not get an error message, it means that the compilation process is going smoothly.
8) Every time you change the looks of your creature, i.e. you modify your "create_*creature_name*.c" file, you need to do the following:
   a) In your creature directory, type "make graphics"

b) Open up another Xterm window and “slogin” one of the following machines: “purple”, “dino-pebbles”, “amygdala”, “abacus”, “beowulf”, or “boo-berry”
c) In your new Xterm window, type “cd $LEGPLOT_SOURCE” and type “make update”.
d) In your original Xterm window, in your creature directory, type “make” in case it is not up to date.

Once you are happy with the looks of your creature, you will be skipping this step and you will only stick to your creature directory and only run the compilation in there (by typing “make” and you do that every time your “control.c” file gets modified). Note: sometimes when you try to compile your creature, you get a message that says “make is up to date” even though you have made changes to your “control.c” file and saved it. To force creature library to compile your creature, type “touch control.c” and then try to compile again (“make”) again.

9) After the creature is compiled successfully, type name of your creature e.g. “test_creature”. You will be taken to a window that says “No Save file found in *your creature directory*” This is when you are supposed to create a state file for your creature. If it says “Command not found”, type “source /home/ll/.login” and try again.
10) A state file contains all the variables associated with your creature e.g. “q.joint1” which is the relative position of the two links connected by joint “joint1”. To create a sample state file, type “record q.*”, “record qd.*”, and “record t”. If you get a message saying “no comprende”, that means that you have typed something wrong or you are trying to save a state file that already exists (ignore this message when you are using the commands “graph”, “gsave”, and “gload” in GUI). The reason we record variables is that we want to be able to view their values/plots later on in legplot which is a graphical program to view what the creature does as time progresses. In your state file, you can also type: “show *variable*” which will display the *variable* in the state file, but you will not be able to view it in legplot. You can also set any variable you want to a desired value by typing “set *variable_name new_value*”. The very first time you create a state file, you need to type “firstrun” before you save the state file. After you have recorded some variables, and typed “firstrun”, save the sample state file by typing “ssave *state_name*”. To see how to use the state file, type “q” which will quit the simulator and take you back to the Xterm. Now, instead of just typing the creature name at the prompt, type: “*creature_name statefile_name*” where *statefile_name* is name of the state file that you saved your state file as. You will see that after executing this command, you will be taken to the state file window that you just saved.
11) If you have defined ground contact points in your “create_*creature_name*.c” file, in order to activate them, you need to set all “*ground_contact_point_names*.model” variables to 1 by typing: “set gc*.model 1” otherwise, your creature will go right through the ground when you simulate it. Note: ground contact points are the only points that cannot go through the ground (of course, after setting the *.model variables to 1) i.e. if you have four ground contact points in the four corners of bottom of each foot of a

biped, and your robot suddenly falls over after taking a few steps, the rest of the body of the robot might go through the ground which will result in your robot hanging at its ground contact points from the ground while the robot is inside the ground.

12) To run the simulation, type “run *run_time*” e.g. “run 2” which will run the Simulation for 2 seconds and will stop. You can stop the simulation at any time by typing “stop”. There are a few more functions that can be used in the simulation window such as “stop”, “hide”, “quit”, etc. which are all listed in page 25 and page 26 of the creature library tutorial under “Simulation Commands”.
13) If your simulation window crashes (you might see the error message “segmentation fault”) or it displays “NAN” (Not A Number) for all the variables, this means that something fatal is happening due to excessive torque. You need to modify your controller in order to prevent that.
14) The first time you run your simulation, you might notice that your creature starts inside the ground. To set the creature’s height to a desired value, type “set q.z *desired_height*” or refer to the GUI’s tutorial to do this visually. Note: In every state file, there exist “q.x”, “q.y”, “q.z”, “q.roll”, “q.pitch”, and “q.yaw” variables. These variables are measured from the center of the bottom base of the VERY FIRST LINK that you created in your “create_*creature_name*.c” file.
15) In creature library, the ground is modeled as combination of springs and dampers. You need to model the stiffness and damping of the ground in order to get successful simulation results.
16) Since this step is mostly visual, first learn how to use GUI and legplot then come back to this step. To model the ground contact point gains (assuming that you have already defined ground contact points in your “create_*creature_name*.c” file; refer to page 15 of the creature library tutorial to learn how to define ground contact points), first find the ground contact point gains in your state file by typing “show gc*.k*” and “show gc*.b*” which will pop up all the ground contact point gains. Let your creature fall from a certain height in the air and see how realistically it reacts based on what your goal is, then keep tuning the gains until you get your desired results. For a 24 Kg biped (called “M2”) that walks on a normal, stiff ground, with four ground contact points in the four corners of bottom of its each foot, we have the following gains:

gc*.k_x = 4000.0,	gc*.b_x = 100.0
gc*.k_y = 4000.0,	gc*.b_y = 100.0
gc*.k_z = 40.0,	gc*.b_z = 500.0

17) After the simulation is stopped, you can save all the data generated during the time the simulation was running by typing “save *datafile_name*” e.g. “save data_test1.dat”. If you only type “save”, the simulator will give it a default name which is the number of the day of the year that the file was saved on with an extension 0, 1, or 2, etc. Note: Every time you save a data file, a state file will automatically be saved with the same name (with an “S” in front of it) as was given to the data file e.g. if your data file was called “data_test1.dat”, a state file will be generated with the name “Sdata_test1.dat”. You will probably not be using these state files, because these are the state files that have been saved after the simulation has been running for a certain amount of time. You rather use state

files that you have manually created before the simulation started running. The "garbage" state files can pile up quickly, so you might need to remove them from your directory before their count reaches few hundred.

18) You can now use legplot to view the data file that you saved after running the simulation. To be able to run legplot, open up another Xterm window and "slogin" one of the following machines: "purple", "dino-pebbles", 'amygdala", "abacus", "beowulf", or "boo-berry". After you logged into one of these machines, go to your creature directory and type "legplot *datafile* &" where *datafile* is the name of the data file that you saved after you ran your simulation. If you get an error, you might need to do the following:
    a) Type "source /home/ll/.login"
    b) Type "setenv DISPLAY *your_machine_name*.ai.mit.edu:0.0"
    c) If you are using a PC, right click on your desktop, the "Display Properties" window will pop up. Click on "Settings", in the "Screen area" section, set your screen resolution to 1280x1024 and in the "Colors" area, set your computer color to High Color (16 bit). Click on "Advanced" (in windows 98 case) and then click on the tab that says "general" and set your font size to small. In the windows NT case, there is no button that says "Advanced", you can set the font size in the "Settings" window. If you are unable to set the screen resolution to what was stated above, you might need to replace your computer video card with a more advanced version otherwise you will not be able to use legplot.
    d) If you had to do part c, you need to reboot your machine.

    Now run legplot again.
19) To learn how to use legplot, refer to the file "legplot_tutorial.doc" (MS Word format) that is located in /home/ll/doc/cl_tutorial directory.
20) There is another graphical user interface called "GUI" which displays the live action of the creature, so instead of saving a data file and viewing it using legplot later, you can use GUI to view your creature while you are simulating it (this is the main difference between legplot and GUI). To start GUI, instead of typing "*creature_name statefile_name*" as you did before, type: "*creature_name statefile_name* -gui" which will open up both the state file (the simulator) and GUI. If you get an error message, try typing:

    "source /home/ll/.login"
    "setenv DISPLAY *your_machine_name*.ai.mit.edu:0.0"

    and run GUI again.
21) To learn how to use "GUI", refer to the file "gui_tutorial.doc" (MS Word format) that is located in /home/ll/doc/cl_tutorial directory.
22) Now that you are ready to write a controller for your creature, look for the function "controller1()" in the "control.c" file which is located in your creature directory. Write your control code within the body of control1(). Every time you modify your "control.c" file, you need to compile the creature by typing "make" in the creature directory. Keep in mind that you do not need to type "make graphics" in your creature directory or "make update" in the

$LEGPLOT_SOURCE directory just because your controller has been modified. As was mentioned in step 8, you only need to run those two commands, if the looks of your creature changes.

23) One important thing to know about creature library is that everything has to be in decimal format. For example, never say 0 in creature library, say 0.0. Also, in creature library when you define a new variable (either in your "control.c" or "create_*creature*.c" file, you do not need to specify whether it is an integer, float, or character as you would in C/C++. All you need to do is to insert "ls." in front of your newly defined variable e.g. if I want to define a variable called "ke" in creature library, all I need to do is to say "ls.ke" and the creature library will set a default value of 0.0 to it unless otherwise specified. Note: Try only using lower case letters, otherwise creature library might have problems recognizing the upper case letters.
24) In every state file, there is a variable called "control_dt" (type "show control_dt" in your state file in order to see it). This variable indicated how often your controller runs (a loop) e.g. if "control_dt" is set to 0.0004, your controller (your code in "control.c" file) will run every 0.0004 seconds. There is another variable called "record_dt" (type "show record_dt" in your state file in order to see it). This variable indicates how often the data are recorded while you are running the simulation. The lower this number is, the more frequently the data will get recorded. Sometimes after running your simulation for a long time (say 30 sec) and viewing it in legplot, you might notice that the beginning portion (first few seconds) of the simulation has not been recorded, therefore legplot only shows a fraction of the simulation that you have run. In order to solve this problem, you need to set your "record_dt" value to a higher number, but be careful, because if this number is too high (0.016 is considered fairly high), legplot will be running the results extremely fast, so you will have to slow it down by using the "Skip" key (refer to legplot tutorial). For now, just start with the default values of "control_dt" and "record_dt".
25) Sometimes after you have added a new variable to your "control.c" file and recompiled, you might notice that either the new variable is not affecting the controller at all or it is totally messing up the results even though this new variable may not have anything to do with the output torque. To solve this problem, do the following:
    a) Make sure that you are in your creature directory
    b) Type "source /home/ll/.login"
    c) Type "echo $MACHTYPE" which should tell you what type of machine you are logged into.
    d) If it says: "pc", type "rm pc/*.o" and if it says "sun4", type "rm sun4/*.o" BE VERY CAREFUL!!! DO NOT type "rm *" instead of "rm *machine_type*/*.o" , because this will delete most of your files in your creature directory.
    e) Recompile by typing "make"