CS295/CS395/CSYS395: Evolutionary Robotics

Programming Assignment 4 of 10

Assigned: Friday, September 23, 2011

Due: Friday, September 30, 2011 by midnight

Description: In this assignment you will download, install and make some changes to Open Dynamics Engine (ODE), the open source physics engine we will be using for the remainder of this class. After installation and compilation, you will run a test application that comes with ODE which simulates a buggy (Fig. 1a). You will then change the radii of the wheels in the ODE code, re-compile and re-run the code to produce Fig. 1b. You will then comment out all the ODE code that creates, simulates and draws the objects and joints in this example program, producing the 'empty world' shown in Fig. 1c. This will provide you with a blank canvas on which you will begin to build your robot in the next assignment.

Important: If you get stuck installing ODE, get on the ODE mailing list and ask your question; you will usually get a response in a couple of hours. **This means of course that you will have to start the assignment before the night before.** If you are still stuck, come see the instructor or teaching assistant.

1. **Back up your Python code from last week.** If you lose your laptop tomorrow, will you still have your code from assignments 1-3?

2. Download Open Dynamics Engine version 0.11.1. Download instructions can be found on ODE's main page. Make sure to download from Sourceforge.

3. Upzip the archive.

4. **If you are working on a Mac or on Linux**, follow the installation instructions found in INSTALL.txt. Once installed, skip to step 15.

5. **If you are working on Windows**, installing ODE is a bit trickier. Download and install the free C++ compiler Visual C++ 2005 Express Edition. **Note from the T.A.:** There is a 2010 version of Visual C++ that is free from Microsoft. Visual Studio 2010 Express (includes C++).

6. Download and install Windows Server 2003 Platform SDK. **Note from the T.A.:** There is an update to Windows Server 2003 Platform SDK which supports Windows 7, and earlier Windows versions including Vista and XP. Windows SDK for Windows 7 and .NET Framework 4

7. Open a Windows Command Prompt (in Windows XP, Start→Run...→cmd[enter]), and navigate to ode-0.11.1/build.

8. type `premake4 --with-demos --with-tests vs2005`

9. Close the Command Prompt and doubleclick on ode-0.11.1/build/vs2005/ode.sln, which will open ode, drawstuff (the drawing routines) and all of the demos in Visual C++.

10. in Visual C++ go to Tools/Options/Projects and Solutions/VC++ Directories.

11. Select 'Show directories for'/Executable files, and add a directory pointer to `C:/Program Files/Microsoft Visual Studio 8/VC/Microsoft Platform SDK for Windows Server 2003 R2/Bin`.

    Note from the T.A.: VC++ Directories editing in Tools→Options has been deprecated; directories are now available as a user property sheet that is added by default to all projects. Had to edit the output directory name in the linker property sheets for drawstuff and ode.

12. Select 'Show directories for'/Include files, and add a directory pointer to `C:/Program Files/Microsoft Visual Studio 8/VC/Microsoft Platform SDK for Windows Server 2003 R2/Include`

13. Select 'Show directories for'/Library files, and add a directory pointer to `C:/Program Files/Microsoft Visual Studio 8/VC/Microsoft Platform SDK for Windows Server 2003 R2/Lib`

14. Now Select Build/Build Solution to compile ODE, drawstuff and all the demo programs.

15. For windows users you should now find executables of the demo programs in `ode-0.11.1/lib/debugdoubledll`. For Linux and Mac users you should find the executables in `ode-0.11.1/ode/demo`. Run the demo_buggy executable.

16. You should see something like in Fig. 1a. Try driving the buggy up over the ramp, and get used to the mouse controls for moving the camera in ODE. Screencapture the simulation window, and paste the result into your document.

17. Open the code for demo_buggy, and find the line that specifies the radii of the buggie's wheels. Multiply that variable by two, recompile, and re-run the demo. Screencapture the simulation window (which should look like Fig. 1b) and paste the result into your document.

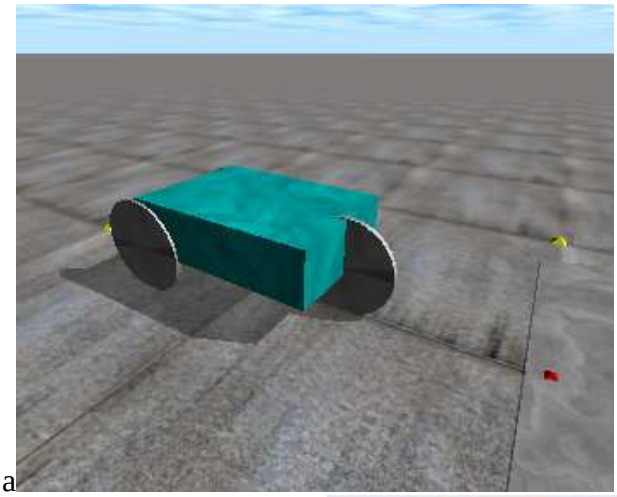
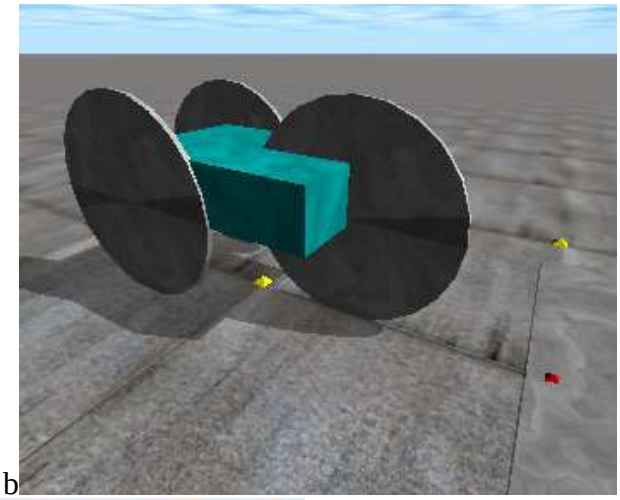
18. Re-open the code, and comment out all the code in the main function so that when you re-compile and re-run demo_buggy, the program does not nothing, but does not crash.

19. Now you will allow the application to only create and destroy data structures, but not start the simulation. Uncomment the seven lines at the beginning of the the main function that define and set the `fn` data structure (`fn` controls drawing). Do not compile or run yet.

20. Uncomment the six lines following that create the world and the ground plane.

21. Uncomment the last four lines in the main loop (beside the `return 0` line) that destroy the world.

22. Re-compile, re-run and ensure that the program still starts and ends without crashing.

a  b 

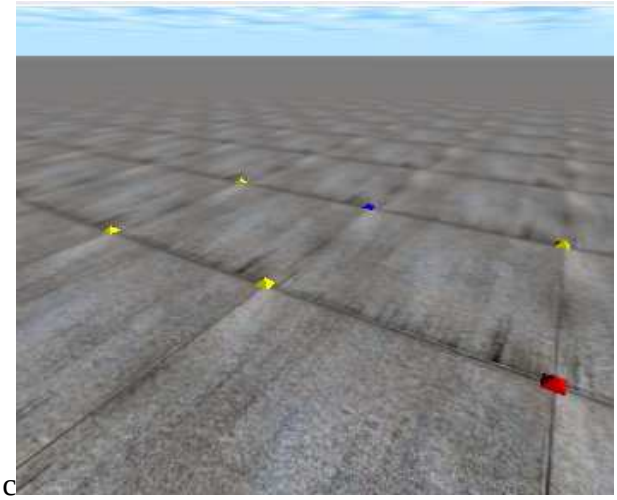
c 

Figure 1: Successful compilation and execution of the test_buggy program should produce **a**. Changing the radii of the buggy's wheels should produce **b**. Commenting out the construction, simulation and drawing of all objects and joints should produce **c**.

23. Now you will allow an 'empty' simulation to run. Uncomment the line that starts the simulation:
    `dsSimulationLoop (argc,argv,352,288,&fn);`
    This line creates a simulation window of 352x288 pixels and hands control to the simLoop function until the user stops the application.

24. Comment out all of the lines in the simLoop function. Recompile, re-run, and you should get an empty simulation as shown in Fig. 1c. Screencapture the window and copy it into your document.

25. Submit your document (containing three images) in .pdf format to the Teaching Assistant.