Module 001 Project: Scribbler2 Dance
CSC 161, “Imperative Problem Solving and Data Structures”
Department of Computer Science · Grinnell College
February 12, 2016

The Pairs
• Ella Nicolson and Tanner Tufto
• Faizaan Ali and Ying Zhang
• Yuyin Sun and Dennis Chan
• Eli Salm and Sanjay Sudhir
• Josh Lavin and Zachary Susag
• Lex Martin and Colin Greenman
• Gemma Nash and Sophie Gaschott
• Mattori Birnbaum and Lilly Webster
• Jong Hoon Bae and Jae Hong Shin
• Anna Blindermann and Cory McCartan
• Tyler Williams and Saung Thuya

Erhaan Ahmad will replace anyone who is absent on Friday.

The Assignment

Working in pairs, develop a program that causes the Scribbler2 robot to perform a dance consisting of three short sections, each involving a different patterned and regular sequence of motions. You may make the three sections completely independent or present them as “variations on a theme,” but no two of them should be identical.

Your design for the robot’s dance and the program that implements it should satisfy these constraints:

• Throughout the dance, the center of the robot should remain within a square sixteen inches on each side.
• Within the program, eSpeak should be used to deliver a text announcement at the beginning of each section of the dance, describing what the robot is about to do.
• Each of the three sections should be written as a separate C function, called from the main program.
• The definitions of those three functions, taken together, must include at least one for-statement, at least one while-statement, at least one if-statement, and at least one switch-statement.

You may use as many more of each kind of statement as you need, and you may also define additional “helper functions” to execute short sequences of motions that are repeated, perhaps with parameterizable variations, in the course of the dance.

The program may punctuate the robot’s motions by having it beep between them, or (with non-blocking motions) may direct the robot to provide its own musical accompaniment by producing tones while the motions are in progress.

To prove that your choreography satisfies the first of the constraints listed above, create a hard-copy record of the robot’s movements: If you tape an eighteen-by-twenty-four-inch sheet of newsprint to one of the tables in our classroom, position and orient the robot on that sheet, drop an uncapped marker into the “pen port” that runs vertically through the middle of the robot, and run the program, the marker will record the robot’s movements on the sheet. When you submit your program turn in the newsprint along with the source code.
Notes and Hints

If you’re having trouble thinking of a use for a switch-statement, here’s one possible application: You could declare and regularly update a variable that stores an integer value that encodes the direction of the robot’s most recent motion (using 0 to represent “forward,” 1 for “backward,” 2 for “left,” and 3 for “right,” for instance). and use a switch-statement to select different subsequent motion patterns or differently pitched beeps depending on the value of the variable.

It would be natural to try to use the four principal movement functions (coderForward, rBackward, rTurnLeft, and rTurnRight) for all of the robot’s motions, but you may find that the robot makes such wide turns that it is almost impossible to create dances that are elaborate enough to be interesting without violating the stay-within-the-wuare constraint. To make the robot pirouette in place (or nearly), you can use the rMotors function, giving it arguments that are equal in magnitude but opposite in sign. This causes the wheels to turn equally quickly, but in opposite directions, so that the robot twists around in the direction of the backward-moving wheel. Note, however, that there is no time parameter for the rMotors function. The wheels keep turning until the rStop or rHardStop function stops them.

If you need inspiration for the choreography, you may want to plan the dance so that the line drawing created by the marker is aesthetically pleasing or achieves some dramatic effect, working backward from the desired appearance of the drawing to the movements that will generate it.

Bear in mind, however, that the robots move differently when rolling over newsprint than when moving around on the hard surface of a desk. They also move differently when they are dragging the marker around than when they are moving without the marker friction. You may want to take these differences into account as well when designing the dance.

I’ll collect the programs and movement records in hard copy at the beginning of class on Wednesday, February 17. Be sure to write your names on the movement records.

We’re still working on a mechanism for submitting programs electronically without sharing them to the world. Until that’s ready, just include the full pathname of your program in the opening comment, and I’ll pull a copy directly from inside your home directory.