Activity 2.3.2 Robot Behaviors and Writing Pseudocode

Introduction

A behavior is anything your robot does: turning on a single motor is a behavior, moving forward is a behavior, tracking a line is a behavior, navigating a maze is a behavior. We are concerned with three main types of behaviors: basic behaviors, simple behaviors, and complex behaviors.

Basic Behaviors

Example: Turn on Motor Port 3 at half power

At the most basic level, everything in a program must be broken down into tiny behaviors that your robot can understand and perform directly. In ROBOTC these are behaviors the size of single statements, like turning on a single motor.

Simple Behaviors

Example: Move forward for 2 seconds

Simple behaviors are small, bite-sized behaviors that allow your robot to perform a simple yet significant task, like moving forward for a certain amount of time. These are the most useful behaviors because they are big enough that you can describe useful actions but small enough that you can program them easily using basic ROBOTC commands.

Complex Behaviors

Example: Follow a defined path through an entire maze

These are behaviors at the highest levels, such as navigating an entire maze. Though they may seem complicated, one nice property of complex behaviors is that they are always composed of smaller behaviors.

```c
#include <ROBOTC.h>

void task_main()
{
    startMotor(rightMotor, 63);
    startMotor(leftMotor, 63);
    wait(2);
    startMotor(rightMotor, -63);
    startMotor(leftMotor, 63);
    wait(.5);
    startMotor(rightMotor, 63);
    startMotor(leftMotor, 63);
    wait(2);
}
```

Basic Behavior – This code turns the right motor on at half power.

Simple Behavior – This code makes both the right and left motor turn on for 2 seconds

Complex Behavior – This code makes a robot with 2 motors travel straight, take a right turn, and continue traveling straight.
The most important idea in behaviors is that they can be built up or broken down into other behaviors. Complex behaviors, like going through a maze, can always be broken down into smaller, simpler behaviors. These in turn can be broken down further and further until you reach simple or basic behaviors that you recognize and can program.

Sometimes it can be difficult to tell whether a behavior is simple or complex. Some programs are so complex they need multiple layers of simple behaviors before they reach the basic ones! Basic, Simple, and Complex are categories of behaviors which are meant to help you think about the structure of programs. They are points of reference in the world of behaviors. Use these distinctions to help you, but don’t worry if your complex behavior suddenly becomes a simple part of your program.

Pseudocode is a compact and informal description of a computer program. It is a hybrid language which combines the features of the programming language with the native language of the person writing the program. Emphasis is placed on expressing the behavior or outcome of each portion of code rather than on strictly correct syntax (it does still need to be reasonable, though). In general pseudocode is used to outline a program before translating it into proper syntax. This helps in the initial planning of a program by creating the logical framework and sequence of the
code. It captures the logic and flow of a solution without the bulk of strict syntax rules.

Below is some pseudocode written for a program which moves a chair up when a touch sensor is pressed and then determines whether it should move up or down based on its current location. This pseudocode example includes elements of both the programming language and the English language. Curly braces are used as a visual aid indicating where portions of code need to be placed when they are finally written out in full and proper syntax.

```c
/*
  while
  {
    if chair pushbutton pressed
    {
      if bottom floor limit switch is pressed
        {
          Chair moves up
          Top limit switch pressed
          Chair stops
        }
    else (either the chair is at the top or midway)
    {
      Chair moves down
      Bottom limit switch pressed
      Chair stops
    }
  }
*/
```

In this activity you will practice writing behaviors and pseudocode for projects you will build in future classes.

**Equipment**

- Engineering notebook
- Pencil
Procedure

Complete the chart below by writing the basic, simple, and complex behaviors for this mobile robot: When a pushbutton is pressed, a mobile robot will go as fast as possible for 20 feet (takes 10 seconds) and then stop.

<table>
<thead>
<tr>
<th>Complex Behaviors</th>
<th>Simple Behaviors</th>
<th>Basic Behaviors</th>
</tr>
</thead>
</table>

Now fill in the template below to write the pseudocode for this mobile robot: When a pushbutton is pressed, a mobile robot will go as fast as possible for 20 feet (takes 10 seconds) and then stop.

Pseudocode:

```c
/*
   {

   */
```
Conclusion

1. List at least five smaller behaviors you could break the complex behavior “brushing my teeth” into.

2. Why is it important to think of a computer program as a set of basic, simple, and complex behaviors that a robot needs to follow?

3. What is the purpose of a set of curly braces { } in a ROBOTC computer program?

4. What is the role of a programmer?