Behaviors

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. There are three main types of behaviors that we are concerned with: **basic** behaviors, **simple** behaviors, and **complex** behaviors.
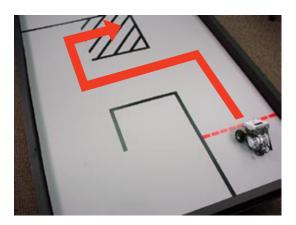
Basic Behaviors
Example: Turn on Motor C at 100% power

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**, or **resetting a timer**.

Simple Behaviors
Example: Move forward for 3 seconds

Simple behaviors are small, bite-size behaviors that allow your robot to perform a **simple, yet significant task**, like **moving forward for a certain amount of time**. These are perhaps the most useful behaviors to think about, because they are big enough that you can describe **useful actions** with them, but small enough that you can program them easily from 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. If you observe a complex behavior, you can always break it down into smaller and smaller behaviors until you eventually reach something you recognize.

```
task main()
{
    motor[motorC] = 50;
    motor[motorB] = 50;
    wait1Msec(2000);

    motor[motorC] = -50;
    motor[motorB] = 50;
    wait1Msec(800);

    motor[motorC] = 50;
    motor[motorB] = 50;
    wait1Msec(2000);
}
```

Basic behavior This code turns the left motor on at 50% power.

Simple behavior This code makes the robot go forward for 2 seconds at 50% power.

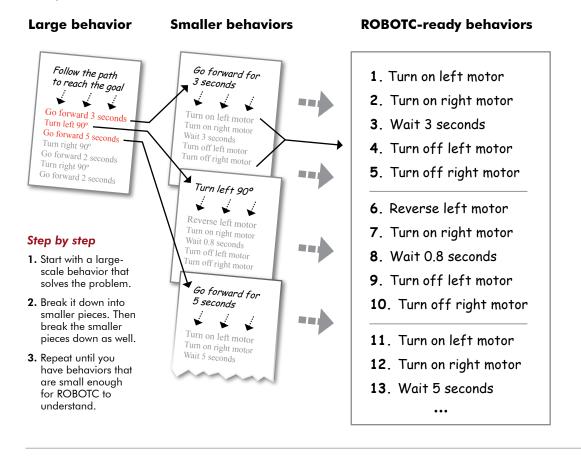***Complex behavior*** This code makes the robot move around a corner.

Behaviors

Composition and Analysis

Perhaps 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.

By looking back at the path of behaviors you broke down, you can also see how the smaller behaviors should be programmed so that they **combine back together**, and produce the larger behavior. In this way, analyzing a complex behavior **maps out the pieces** that need to be programmed, then allows you to **program them**, and **put them together** to build the final product.

Large behavior   **Smaller behaviors**   **ROBOTC-ready behaviors**

Follow the path
to reach the goal

Go forward 3 seconds
Turn left 90°
Go forward 5 seconds
Turn right 90°
Go forward 2 seconds
Turn right 90°
Go forward 2 seconds

Go forward for
3 seconds

Turn on left motor
Turn on right motor
Wait 3 seconds
Turn off left motor
Turn off right motor

Turn left 90°

Reverse left motor
Turn on right motor
Wait 0.8 seconds
Turn off left motor
Turn off right motor

Go forward for
5 seconds

Turn on left motor
Turn on right motor
Wait 5 seconds

1. Turn on left motor
2. Turn on right motor
3. Wait 3 seconds
4. Turn off left motor
5. Turn off right motor

6. Reverse left motor
7. Turn on right motor
8. Wait 0.8 seconds
9. Turn off left motor
10. Turn off right motor

11. Turn on left motor
12. Turn on right motor
13. Wait 5 seconds
...

Step by step

1. Start with a large-scale behavior that solves the problem.

2. Break it down into smaller pieces. Then break the smaller pieces down as well.

3. Repeat until you have behaviors that are small enough for ROBOTC to understand.

Sometimes it can be hard 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 next program… just pick the point of reference that's most useful for what you need.