Kodu Module 4: Rule Ordering and Default Value

David S. Touretzky

Version of July 7, 2015

Learning Goals

- Indentation links to the rule above, not below.
- Rule order establishes priority for conflicting actions: the lower numbered rule wins.
- Rule order can be used to establish a default value for an action.
- In the Default Value idiom, the rule with the default value must come after the other rules for that action, so it can be overridden if one of the other rules applies.

Time Required: 2 hours

Worlds

- Indent1
- Apple2
- CountFish1

Tile Manipulatives

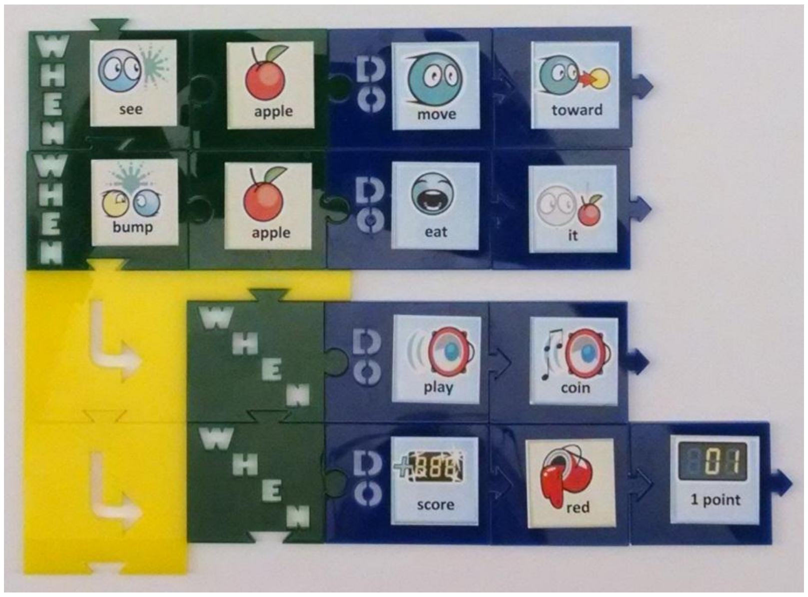
- WHEN-bump, Empty-WHEN (2 tiles)
- apple (two tiles)
- DO-eat, DO-play, DO-score
- it, coin, red, 1 point
- Indent, More-Indent

Flash Cards

- Do Two Things
- Count Actions
- Default Value (new to this lesson)

Part 1: Doing Three Things

1. Review the "Do Two Things" flash card. Suppose we want to do *three* things when we bump an apple: eat it, play a sound, and score a point. How many rules will this require? (Answer: three, because each action requires a separate rule.)
2. Use the tile manipulatives to model the Pursue and Consume pattern with three consume actions. You will need an Indent tile and a More-Indent tile:



3. Enter the rules into the Apple1 world. Run it and verify that all three actions occur with each bump.
   [1] WHEN see apple DO move toward
   [2] WHEN bump apple DO eat it
      ↳ [3] WHEN DO play coin
      ↳ [4] WHEN DO score red 1 point

4. Class discussion:
   a. Does the order of rules 3 and 4 matter? (Answer: no, because their actions are independent.)
   b. When will the kodu score a point? (Answer: when it bumps an apple.)
   c. How do the tiles show you this? Starting at rule 4's WHEN tile, move left to the arrow and then trace the arrows upward until you arrive at the parent rule's WHEN tile. You'll see that rule 4 can only run when "bump apple" is true (rule 2's condition).

Part 2: An Un-Indented Rule Can Follow an Indented One

1. An indented rule always attaches to the rule above, but there can also be a rule below.
2. Load and run the Indent1 world. You will see a series of turtles and octopuses approaching the kodu.
3. We want the kodu to eat the turtles, add a point to the green score, and play a sound. We also want it to eat the octopuses. Enter the following rules for the Kodu:

```
[1] WHEN bump (bots II) turtle DO eat it
    ↳ [2] WHEN DO (actions) (more) play (event) (tower) star collect
    ↳ [3] WHEN DO score green 1 point
[4] WHEN bump (objects) octopus DO eat it
```

4. Run the program and observe what happens.
5. Class discussion:
   a. When does the "star collect" sound play? (When the kodu is bumped by a turtle.)
   b. What sound is heard when an octopus bumps the kodu? (Only the chomping "eat" sound.)
   c. When will rule 3's action run? (When the kodu is bumped by a turtle.)
   d. Go back into the rule editor. Can you indent rule 2 any further? (No, it can only indent by one level more than the rule above it.)
   e. Can you un-indent rule 2? (Yes, and that makes rule 2 become rule 3's parent. And since rule 2 no longer depends on rule 1, the meaning of the program changes.)
   f. What happens when you run the program with this modification? (The sound plays and the green score runs up all the time.)
   g. What happens when you re-indent rule 2? (Rule 3 must then also be un-indented.)

Part 3: Rule Ordering

1. Make a new Kodu world: hit the back button, select the Home icon, and select "New World".
2. Select the Object Tool (kodu icon), click on the ground, and create a kodu in one corner of the world.
3. Create a tree in another corner and a castle in a third corner.
4. Program the kodu with the following rules:

```
[1] WHEN see tree DO move toward
[2] WHEN see castle DO move toward
```

5. Class discussion: When you run the program, what do you think the kodu will do? Try it and see.
6. Does it matter where the kodu starts from? Try picking up the kodu and moving it to different starting positions: closer to the tree, or closer to the castle. (Put the cursor on the kodu, press A to pick it up, move the cursor to a new location and press A to put the kodu down there.)
7. Try switching the order of the two rules. To do this, go into the rule editor, put the pencil on the number "1", press A to pick up the rule, use the left stick to slide the rule down, and press A to release the rule.
8. Class discussion: What conclusions did you reach from this experiment?
   General principle: when two rules have conflicting actions, the *earlier* (lower numbered) rule wins.

Part 4: First Eat the Red Apples, then Grab the Blue Ones

1. Load the Apple2 world.
2. Suppose we want the kodu to first eat all the red apples, and then grab all the blue ones. How would you make it do that?
3. Enter the following program for the kodu:

    [1] WHEN see red apple DO move toward

    [2] WHEN bump red apple DO eat it

    [3] WHEN see blue apple DO move toward

    [4] WHEN bump blue apple DO grab it

4. Run this program and see what the kodu does.
5. Does the ordering of the two consume rules (eat/grab) matter? Try changing their order and see. (It doesn't matter because the rules are independent; the kodu will never bump a red apple and a blue apple at the same time in this world.)
6. How would you make the kodu grab all the blue apples first, and then eat all the red ones? (Answer: move the "see blue apple" rule before the "see red apple" rule.)
7. Make this change to the program and see what happens

Part 5: Default Value

1. Study the Default Value flash card.
2. What does "default" mean? (It means the value assumed when there is no explicit statement of a different value. Example: "A McDonald's quarter pounder with cheese comes with <u>two</u> pickles." So the default number of pickles is 2; you should say something if you want more or fewer pickles.
3. Let's practice Default Value in the Indent1 world. Reload the Indent1 world.
4. Give the kodu the following rules from the Default Value flash card:

    [1] WHEN see *(objects)* octopus DO *(actions)* color me red

    [2] WHEN DO color me blue

5. Run the world and watch the kodu change colors.

Part 6: The CountFish1 World

1. Distribute the CountFish1 handout, and have students load and run the CountFish1 world.
2. In this world the blue score goes up as long as the kodu is colored blue. We want the kodu to stay blue in order to score more points, but it should turn red or green whenever it sees a flying fish, because if the fish see a kodu they will eat it unless it's the same color as them. In other words, blue should be the kodu's *default* color, used except when a flying fish is present.
3. Verify that the students come up with the correct solution:

   [1] WHEN see *(bots I)* fly fish red DO color me red

   [2] WHEN see fly fish green DO color me green

   [3] WHEN DO color me blue

4. Class discussion: Suppose we make the rule with the empty WHEN be the first rule. What do you think will happen?

   [1] WHEN DO color me blue

   [2] WHEN see flying fish red DO color me red

   [3] WHEN see flying fish green DO color me green

5. Try running this incorrect version. Why did the kodu not turn red or green? (Because rule 1 set the color to blue, and it has priority.) Remind the class: *"The earlier rule wins!"*

Part 7: Review and Assessment

Have students complete the questionnaire for this lesson.

Answers to last part of the questionnaire: a-1, b-2, c-1, d-4, e-3, f-4.

Part 8: Default Value And Move Wander

Distribute the Castles1 handout and have students do the exercise on their own.

The solution to this world is:

[1] WHEN see coin DO move toward
[2] WHEN bump coin DO eat it
[3] WHEN DO move wander

Answers to questions in the Castles1 handout:

a. Does the action (of rule 3) conflict with another rule's action? Which one?
   Rule 3 conflicts with rule 1: both have a "move" action, but with different parameters ("toward" vs. "wander").

b. Does the default rule (rule 3) have an empty WHEN part? **Yes**

c. Does the default rule appear *after* the rules it conflicts with? **Yes**

d. Is this program an example of the Default Value idiom? **Yes**

e. What is the default value? **The default value is "wander".**