Kodu Module 3: Indentation and Scores

David S. Touretzky
Version of June 7, 2016

Learning Goals

- Effective range of the "eat" action.
- New idioms: "Do Two Things" and "Count Actions"
- Rule indentation.
- Empty WHEN predicates are always "true".
- Failing to indent a rule means its action occurs all the time.
- The "score" action adds points to a score.
- The "boom" action blows something up.
- The "give" action gives another character the object we are holding.

Worlds

- Measure1 – with Debug LOS turned on
- Apple1 – with Debug LOS turned on
- HeartCannon1X
- BoomPipes1
- Giver1

Handouts

- Measure1 world
- HeartCannon1X world
- BoomPipes1 world
- Giver1 world

Tile Manipulatives

- WHEN-see, WHEN-bumped (2 tiles), Empty-WHEN
- apple (2 tiles), heart (2 tiles), red (2 tiles)
- DO-move, DO-eat, DO-play, DO-score
- toward, it, coin sound, "1 point"
- Indent tile

Flash Cards

- Do Two Things
- Count Actions

Part 1: An Experiment Involving "See" And "Eat"

1. Ask the students: Why do we need to "pursue" before we can "consume"? Answer: because the kodu can't eat objects that are out of reach – and neither can you! (Can you eat an apple that's sitting on a table at the back of the room? No.)
2. Distribute the "Measure1 World" handout and let the students load and run the world. The kodu looks at the apple but nothing else happens.
3. Have the students examine the kodu's programming. There is only one rule:

   [1] WHEN see apple DO eat it

4. Tell the students to measure distance to the apple by counting the black dots on the floor.
5. Class discussion: How close does an apple have to be in order for the kodu to eat it? Within arm's reach? The kodu doesn't have arms, but what if it did have arms? (Let them guess how far the kodu can reach.)
6. Have them move the apple closer to the kodu by following these steps:
   a. Select the object tool
   b. Put the cursor on the apple
   c. Press the green "A" button to pick up the apple
   d. Move the apple closer to the Kodu but keep it on the pink strip with the black dots
   e. Press "A" to put down the apple
   f. Press the Back button twice to play
7. Let them experiment until they find the farthest the apple can be from the kodu and still be eaten. Answer: 5 to 6 squares away, depending on how you measure.
8. Guide the class to this conclusion: when an action has limited range (like eat), we need the pursue rule in order to get the kodu close enough to the object for the consume rule to apply.

Part 2: Doing Several Things At Once

1. Have students examine the Do Two Things flash card.
2. Explain that sometimes we want a rule to take two actions at the same time. For example, when we bump an apple we might want to eat it and also play a sound.
3. The Do Two Things flash card shows how to make this happen: we put the second action underneath the first one, and indent that second rule. Also, the "when" part of the second rule can be left blank. A blank means the same as "always". (There is actually an "always" tile you can use if you wish.)

   [1] WHEN see apple DO move toward
   [2] WHEN bumped apple DO eat it
   ↳ [3] WHEN DO play coin

4. Model this program using the tile manipulatives. Using a yellow indentation tile plus a blank WHEN tile, set up the following program:

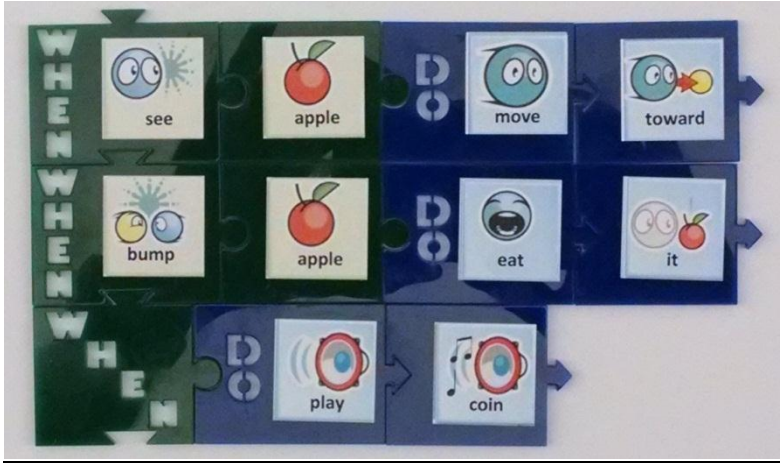
5. Load the Apple1 world and have the students program the kodu with just rules 1 and 2 above. Have them run the program so they can hear the kodu eating the apples (there is a chomping sound).
6. Now have them add rule 3, using the following inputs:

   ↳ [3] WHEN DO *(actions) (more)* play *(event) (arcade)* coin

7. To indent rule 3, use the left stick to put the pencil over the rule number, press "A" to pick up the rule, move the pencil to the right, and press "A" again to put the rule down.
8. Have the students run the program and verify that the kodu is both eating the apple and playing the coin sound at the same time.

Part 3: What Happens When We Don't Indent the Second Action?

1. Using the tile manipulatives, have the students remove the yellow indentation tile from rule 3. Now the tiles should look like this:



2. Ask them what they think about the empty WHEN part? What does it mean? Answer: an empty WHEN part is always true. Writing just "when" is the same as writing "when always".
3. What do they think rule 3 will do now? Answer: the kodu will constantly play coin sounds while the other rules lead it to pursue and consume the apples.
4. Have the students make the change to their program and test it out. Note: to remove indentation from a rule, pick it up as before using the "A" button, and slide the pencil to the left to "outdent" the rule.

   [1] WHEN see apple DO move toward

   [2] WHEN bumped apple DO eat it

   [3] WHEN DO play coin
5. The kodu should play the coin sound continuously. This quickly gets annoying.
6. Have the students "fix the bug" by re-indenting rule 3 and running the program again.

Part 4: Adding a Flying Fish That Launches Objects

1. Have students add a flying fish character to their Apple1 world. The flying fish's job is to get apples to the kodu so it can eat them more quickly.
2. Ask them to program the flying wish with Let Me Drive, but the drive rule should also play the "lock on" sound when it moves the kodu. Thus, they're using Do Two Things.
3. In addition, the "A" button should perform a "launch" action and play the "inkjet fire" sound. They can use this feature to toss apples (or the kodu). The code should look like this:

   [1] WHEN gamepad L-stick DO move

   ↳ [2] WHEN DO play *(event) (misc)* lock on

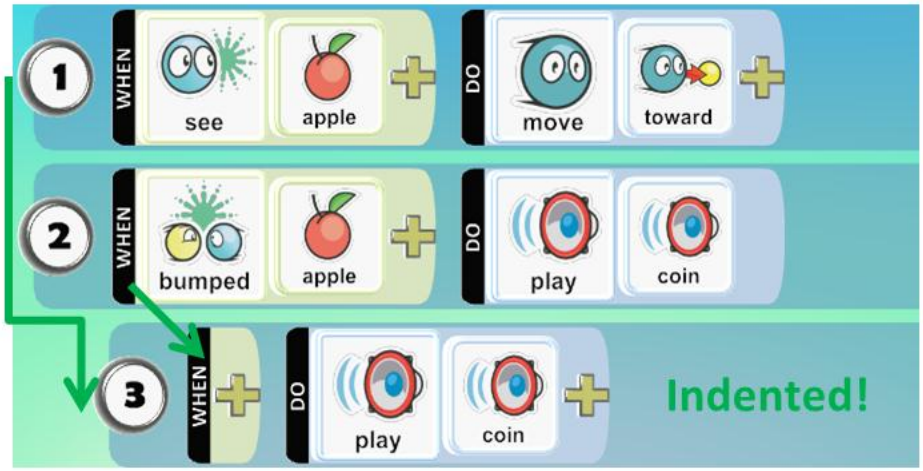
   [3] WHEN gamepad A-button DO launch

   ↳ [4] WHEN DO play *(event) (arcade)* inkjet fire

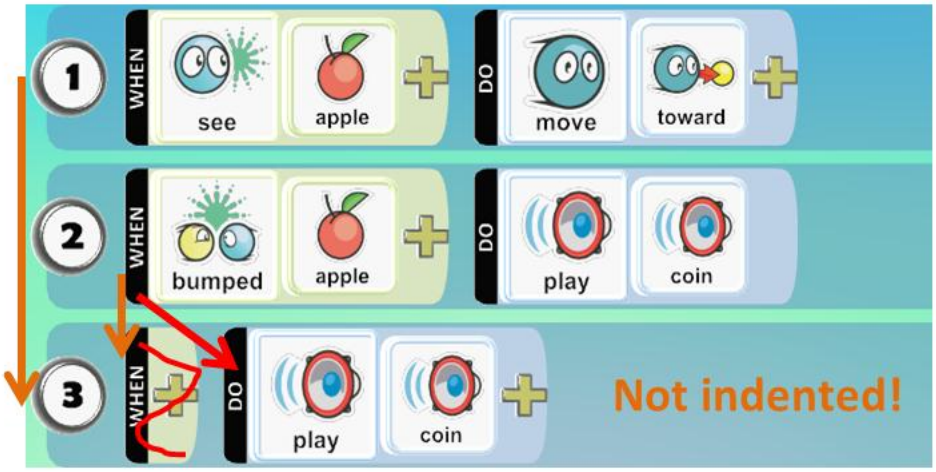
Part 5: Visual Confusion About Indentation

1. When a rule has an empty WHEN part, students are sometimes confused about whether the rule is indented or not. The examples below will help them make the distinction.

2. When a rule is indented, the rule number is displaced to the right (blue arrow), and the WHEN marker is also displaced to the right (orange arrow). Don't try to measure indentation by looking at the DO part.

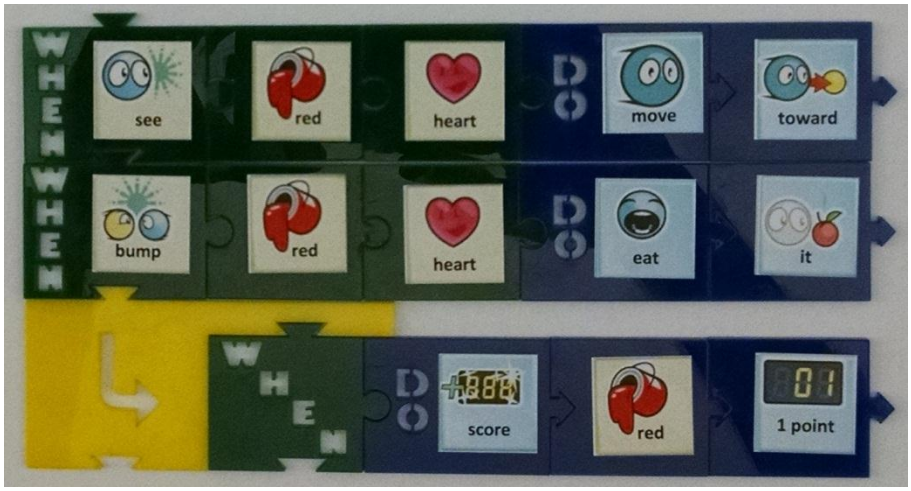


3. When a rule is not indented, the rule numbers are vertically aligned (blue arrow) and so are the WHEN markers (orange arrow). Students get confused when they ignore an empty WHEN marker (because it's tiny) and instead compare the indentations of the leftmost whole tiles (red arrow). This compares a WHEN with a DO, which is incorrect.



Part 6: Counting Things

1. Let's make the kodu count how many things it eats. Show students the Count Actions flash card.
2. Distribute the "HeartCannon1X World" handout and have students load and run the world. Notice that the cannon randomly shoots red or blue hearts, and the game is limited to 90 seconds.
3. Have them follow the instructions in the handout to make the kodu eat and count the red hearts. Optionally you can have them do this with the tiles first:



4. The code should look like this:

   [1] WHEN see red heart DO move toward

   [2] WHEN bumped red heart DO eat it

   ↳ [3] WHEN DO (game) score (scores) red 1 point

5. Experiment: try removing the "heart" tile from rule 1. (The kodu gets stuck at the red tree.)
6. Fix the bug by restoring the "heart" tile in rule 1.
7. When the fans turn on, the kodu has a hard time chasing down the red hearts. Program the blimp with the Let Me Drive idiom. The blimp can turn off a fan by bumping it.

Part 7: Removing Rule Indentation Again

1. Ask the students what they think would happen if they removed the indentation from rule 3. Answer: the score would run up by one point at a time, at a very fast rate.
2. Have the students try this in their program and observe the effect.
3. This world only runs for 60 seconds. How high can their score go in 60 seconds? Different students will get slightly different results; compare them and see. Faster computers will yield higher scores.
4. Have them "fix the bug" by restoring the indentation to rule 3 and running the program again.

Part 8: Counting Multiple Things

1. Distribute the BoomPipes1 handout and let students solve the problem on their own.
2. Here is the solution:

[1] WHEN see pipe DO move towards

[2] WHEN bumped pipe DO boom it

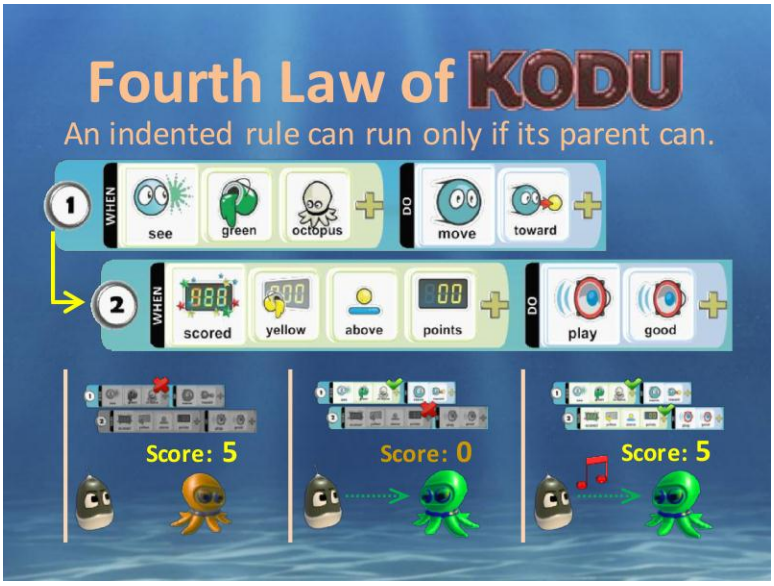
↳ [3] WHEN DO score blue 1 point

[4] WHEN bumped rock DO boom it

↳ [5] WHEN DO score gray 1 point

Part 9: The Fourth Law of Kodu (Indent Law)

1. Show students the Fourth Law of Kodu graphic and ask them to try to explain its meaning.



2. Here is the explanation:
   - In the first panel, rule 1 cannot run (no green octopus) so rule 2 is not even considered.
   - In the second panel, rule 1 runs but rule 2's WHEN part is not true because the yellow score is zero, so rule 2 cannot run.
   - In the third panel, both rule 1 and rule 2 can run, and the kodu both moves toward the octopus (rule 1's action) and plays a sound (rule 2's action).

Part 10: Saving the Octopuses

1. Have students load and run the Giver1 world. What are the characters in this world? (The kodu, the cycle, five octopi, and the pushpad.)
2. Let them solve the problem on their own. The final program should look like this:

[1] WHEN see pink octopus DO move toward

[2] WHEN bumped pink octopus DO give

↳ [3] WHEN DO *(actions) (more)* play *(events) (tower)* coin spit