“Laws of Kodu” Teacher’s Guide

David S. Touretzky, June 15, 2016

A Few Words About Terminology and Notation

We say that a rule “can run” if its WHEN part is true (green check mark). 

According to the Second Law, any rule that can run will run, so saying that a rule “can” run is the same as saying that it “will” run or “does” run or just “runs”.

If the WHEN part is false (red cross)  it is grayed out. In that case the rule cannot run, so its action will not be taken.

Even if a rule does run, its action might not be taken. The action could be blocked by a conflicting action of a lower numbered rule (Third Law).

If an action is not taken, either because the rule cannot run or because the action is overridden, by a lower numbered rule, the action is shown as grayed out.

It is also possible for a rule to run and its action is attempted but fails. For example, the rule “WHEN see apple DO eat it” can run whenever an apple is visible, but the eat action will fail if the apple is too far away to eat. The rule is still said to have run.

An indented rule can run only if its WHEN part is true and its parent can run (Fourth Law). If its parent cannot run, the entire indented rule is grayed out.

A green check mark indicates that the WHEN part of the rule is true, so the rule does run.

The dotted arrow shows which apple the kodu is seeing (and moving toward).

How to teach the First Law

The First Law is called the "Closest Law".

The First Law can be taught using any world that contains a Pursue rule, such as Apple1X.

You can lead students to discover the law by having them run a Pursue and Consume program and note the order in which the objects are visited. Orbit the camera to get a bird's eye view of the terrain and draw a map on the board showing the kodu's starting location and the positions of the objects. Then draw the path the kodu takes.

Ask students how the kodu decides which object to visit next. Guide them to the hypothesis that it visits the closest matching object.

To test the hypothesis, first change the kodu's starting position on the map you drew and ask the students to predict the path it will take from this new starting location. Then move the actual kodu to this new position and run the program to see what path the kodu takes.

Each rule independently chooses a closest matching object, so if there are multiple pursue rules on a page, several objects might be chosen, but the earliest move action will be the one chosen.

In the diagram, the kodu isn't currently bumping anything, so the WHEN part of the Consume rule is false. This is indicated by the red cross and also by the WHEN part being grayed out.

Since the Consume rule can't run, its DO part is also grayed out.

The Pursue rule has a true WHEN part since the kodu sees an apple, so this rule can run. It doesn't matter whether the Pursue rule comes before or after the Consume rule.

How to teach the Second Law

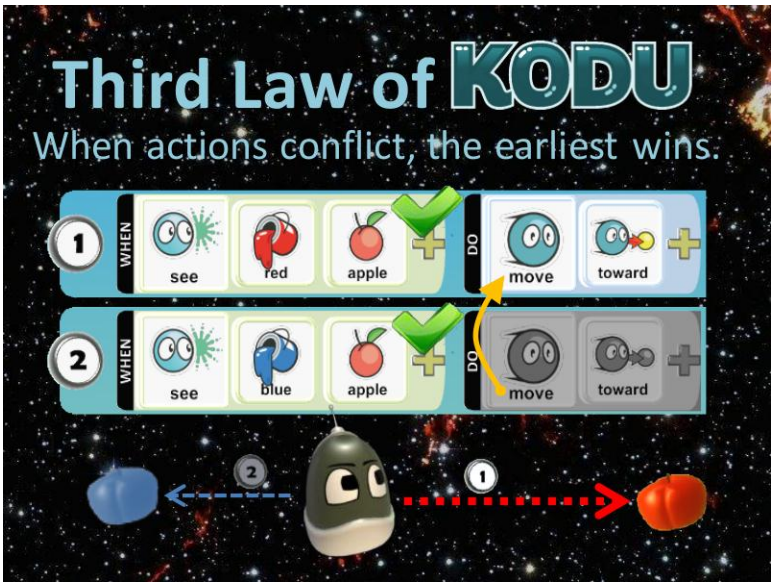
The Second Law is called the "Any Rule Law".

The Second Law can be taught using any world that contains Pursue and Consume, such as Apple1X. The point of this law is that rule order doesn't matter in determining whether a rule can run. Either order will work.

Have students program the normal Pursue and Consume rule for the world you're using, and run the program to verify that it works as expected.

Then have students switch the order of the rules and run the program again. The behavior will be the same.

This exercise can help clear up the common misconception that a page of rules is a sequential procedure (as it would be in Scratch or Python), and you cannot run rule 2 until rule 1 has run successfully. Rule order does not determine when a rule can run.



Since the kodu sees both a red apple and a blue one, both rules have true WHEN parts, as shown by the green check marks. And both rules run. But their actions conflict. In this situation the Third Law says that the lower numbered rule wins, so rule 1 causes the kodu to move toward the red apple. Since rule 2's DO part is overridden by rule 1, it is shown as grayed out even though the WHEN part has a green check mark.

How to teach the Third Law

The Third Law is called the "Conflict Law".

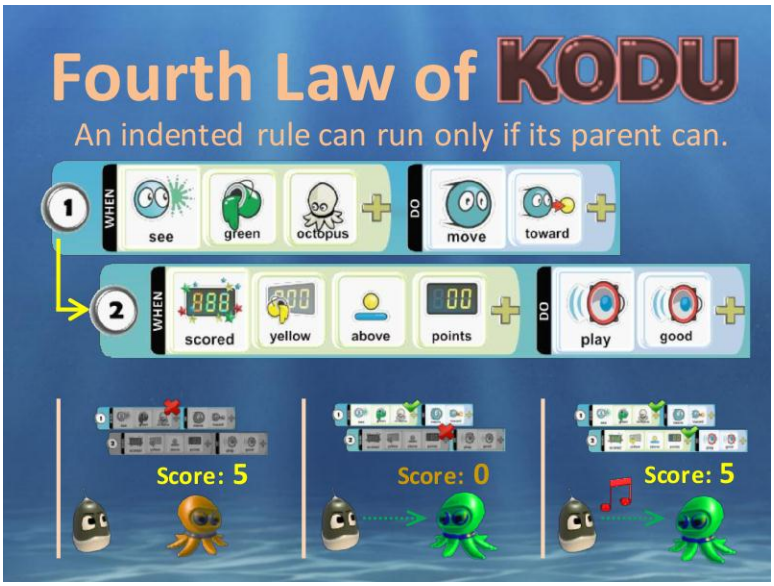
The Third Law introduces rule arbitration. It can be taught in any world containing multiple Pursue rules, such as Apple2 or Star2. The Star2 instruction sheet contains an exercise like the one below.

Have students run the world and write down the sequence of objects consumed, e.g., for Apple2 it would be red vs. blue apples; for Star2 it would be hearts vs. coins. With two Pursue rules, the kodu will consume all the objects of the first type before moving on to objects of the second type.

Then have students switch the order of the Pursue rules and ask them to predict what the effect will be. Then have them run the program again, noting the order in which objects are consumed. The order should be the reverse of the previous experiment.

Next ask students what will happen if you switch the order of the Consume rules. Answer: it makes no difference. Since Consume rules are never in conflict, the Third Law doesn't apply, and the Second Law tells us that they can appear in any order.

If you're pursuing two different colors of the same object, such as red and blue apples, you can replace the two Pursue rules with a single more abstract rule that says "WHEN see apple". Ask the students to predict the order in which apples will be consumed in this case. According to the First Law, they should see a mixed sequence of red and blue apples if they're using Apple2.



The Fourth Law introduces rule dependency: rule 2 is indented and thus dependent on its parent, rule 1. The rules are repeated in miniature in the three scenarios at bottom.

Case 1: rule 1's WHEN part is false because there is no green octopus visible. In this case the indented rule is not eligible to run; the entire second line is grayed out. The fact that the yellow score is above zero is never even considered.

Case 2: rule 1's WHEN part is true and its move toward action is taken. But rule 2's WHEN part is false because the yellow score is zero, which is not above zero. Rule 2's WHEN part is therefore marked with a red cross: this rule cannot run.

Case 3: rule 1's WHEN part is true, and so is rule 2's, since the yellow score is above zero. Both WHEN parts have green check marks and both rules' actions are taken.

How to teach the Fourth Law

The Fourth Law is called the "Dependency Law".

Rule Indentation is usually introduced using the Do Two Things or Count Actions idioms. In both of these the indented rules have empty WHEN parts, which is simpler than the diagram above. The diagram illustrates the more general case where the parent rule and the indented rule each has a condition that must be satisfied. With Count Actions or Do Two Things the parent rule alone determines whether both rules can run, so we have Case 1 if the parent's WHEN part is false or Case 3 if it is true.

To help students understand the effect of indentation, take a world where Do Two Things or Count Actions is used and have them remove the indentation. The rule's action will then execute continuously since the empty WHEN part is always taken to be true.

To test the rules shown in the diagram, use the Indent2 world and add a consume rule that eats the octopus and adds 1 to the yellow score. You'll see that the kodu pursues the first green octopus silently (Case 2), but once the score is above zero it pursues green octopuses noisily by playing the "good" sound (Case 3).