Using Kodu to Teach Reasoning About Programs

David S. Touretzky
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA

Joint work with:
Christina Gardner-McCune and Ashish Aggarwal
University of Florida

October 19, 2016

Funded by a gift from Microsoft Research.

WHY TEACH REASONING ABOUT PROGRAMS?

- Teaching kids to program is not the main goal.
  - They can copy code without understanding it.
  - Programming by trial and error ≠ understanding.
- We teach programming as part of a broader effort to teach computational thinking.
  - Reasoning about programs is part of CT.
- If kids can reason about programs, then they will also be able to write programs.

WHAT CAN A COMPETENT REASONER DO?

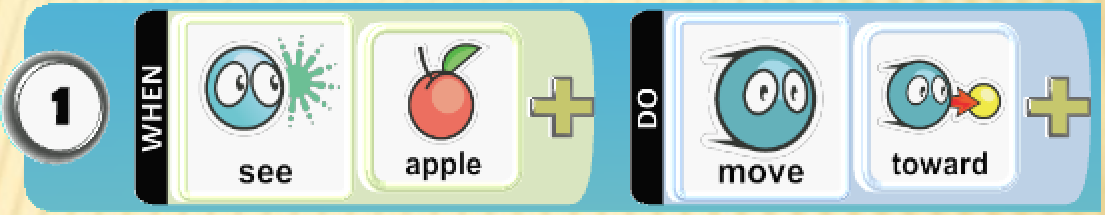
- Explain observed program behavior in terms of the code and the “laws” of computation.

- Predict program behavior from the code. How?
  - Mental simulation: execute the code in one’s head.
  - Recognize patterns in the code that provide insight and eliminate the need to explicitly simulate.

- Construct programs by applying design patterns and computational principles.

1. WHY IS KODU DIFFERENT?

- More powerful primitives than other languages designed for children.



- The WHEN part does pattern matching.
- The DO part uses object-centered actions, not screen coordinates.
- Every rule is a conditional.
- Implicit looping: rules run all the time.

WHY IS KODU DIFFERENT? (2)

- Kodu is a robot language, not a graphics language.
  - Characters are semi-autonomous.
  - Capable of complex perception & goal-directed action.
  - They fidget when they have nothing else to do.
- Kodu worlds are truly three-dimensional.
  - Scratch is 2D; Alice is pseudo-3D.
- Built in physics (gravity, collisions, inertia, wind, …)
- Built in sound effects.

WHY IS KODU DIFFERENT? (3)

- Teaching young kids to reason about programs requires that the programs be short.

- But programs also need to be interesting!

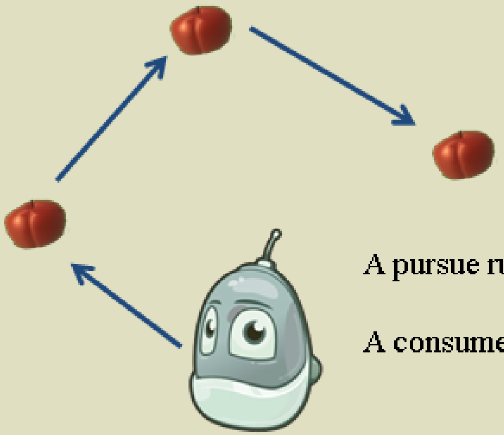
- Because the Kodu language and worlds are so rich, one can write interesting 2-3 line Kodu programs.

- Kodu's idioms and laws provide a good framework for teaching kids to reason about these programs.

2. KODU IDIOMS (DESIGN PATTERNS)

Pursue and Consume

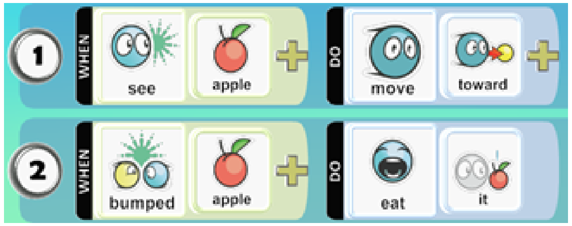
Make the Kodu go to objects and eat them.



A pursue rule involves *motion*.

A consume rule *uses up* the object.

Pursue and Consume



1 WHEN see apple + DO move toward +

2 WHEN bumped apple + DO eat it

Pursue rule

Consume rule

General Form:

WHEN see *thing* DO move toward

WHEN bumped *thing* DO *consume it*

"Consume" can be "eat", "grab", "vanish", or something else.

Filter by color:

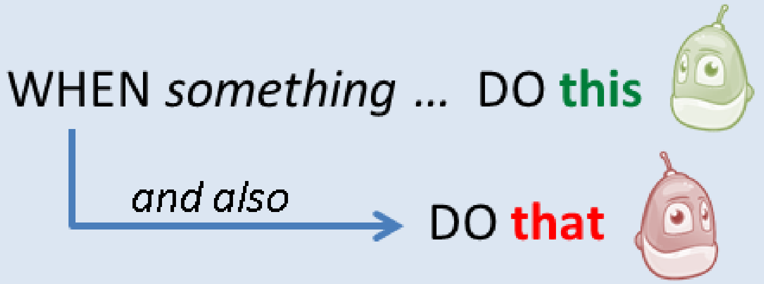
WHEN see *color thing* DO move toward

WHEN bumped *color thing* DO *consume* it

“DO TWO THINGS” IDIOM

Do Two Things

Make the Kodu take two actions with one rule.



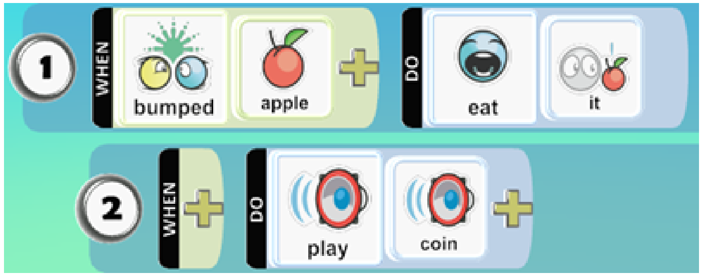
WHEN *something* … DO **this**

and also

DO **that**

Do Two Things

When you’ve bumped an apple, eat it *and also* play the coin sound.



General Form:

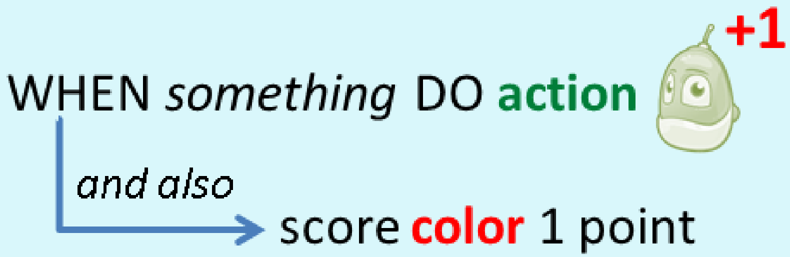
```
    WHEN something DO action1
      ↳WHEN DO action2
```

Indenting the second rule makes it dependent on the WHEN part of the rule above.

“COUNT ACTIONS” IDIOM

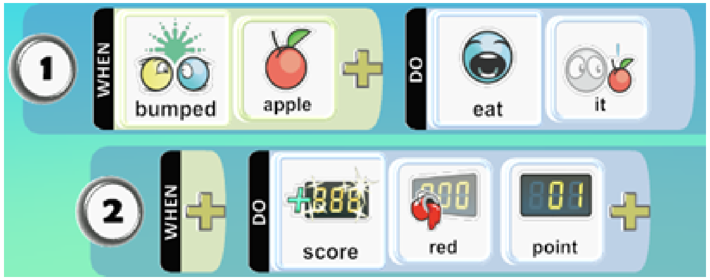
Count Actions

Make the Kodu keep a count of an action it takes.
This is a special case of Do Two Things.



Count Actions

When you eat an apple, add one to the red score.



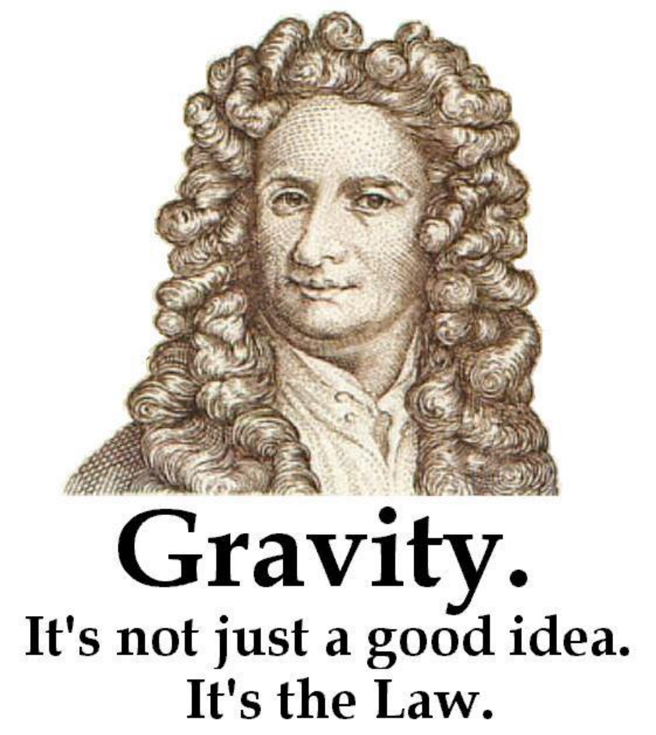
General Form:

WHEN *something* DO ***action***
↳WHEN DO score ***color*** 1 point

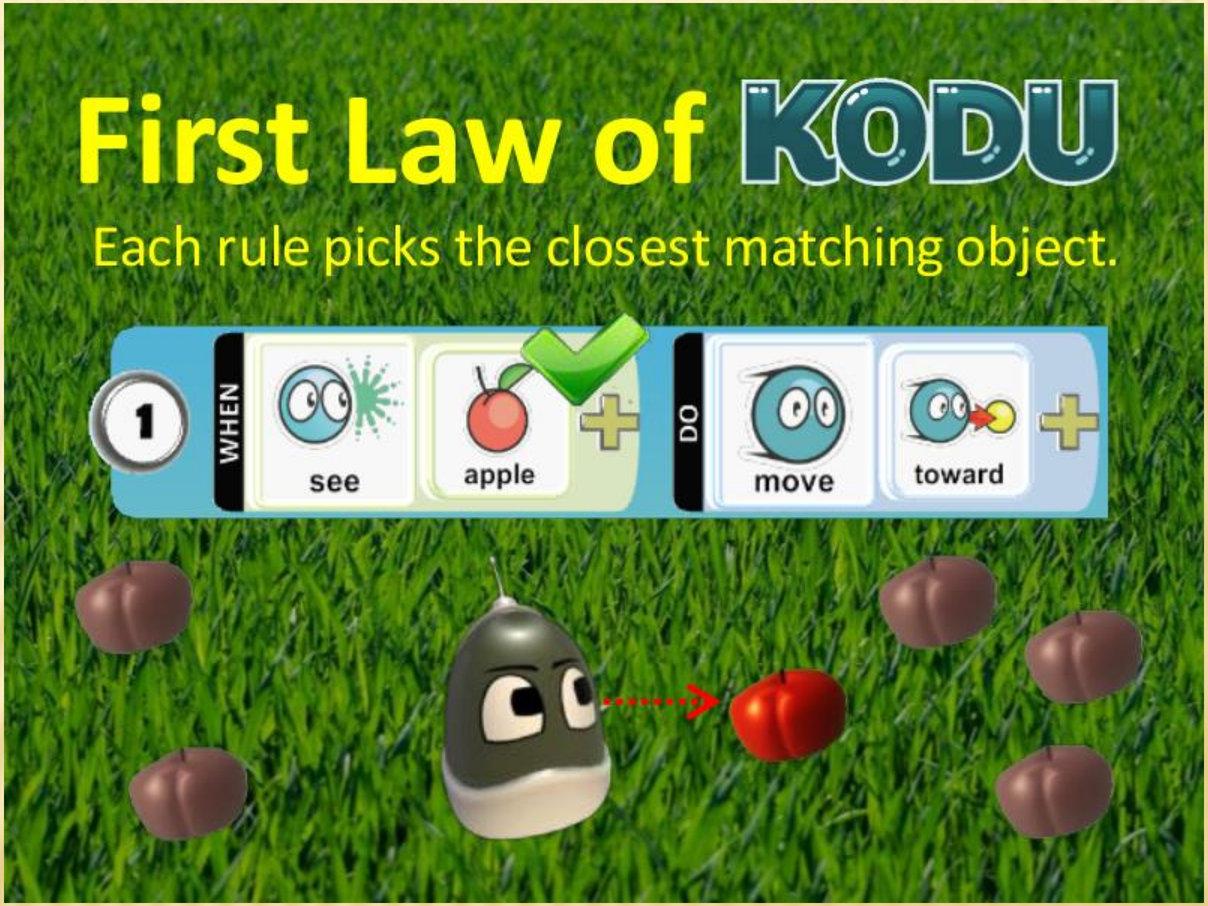
Scores named by colors, such as “red”, are displayed automatically.
Scores named by letters, like “A”, are kept but not displayed.

3. LAWFULNESS

- Not "obedience to authority"!
- "Lawful" in the scientific sense:
  - Every action has a cause.
  - The causes are knowable.
  - So behavior is predictable.
- As in Newton's laws.



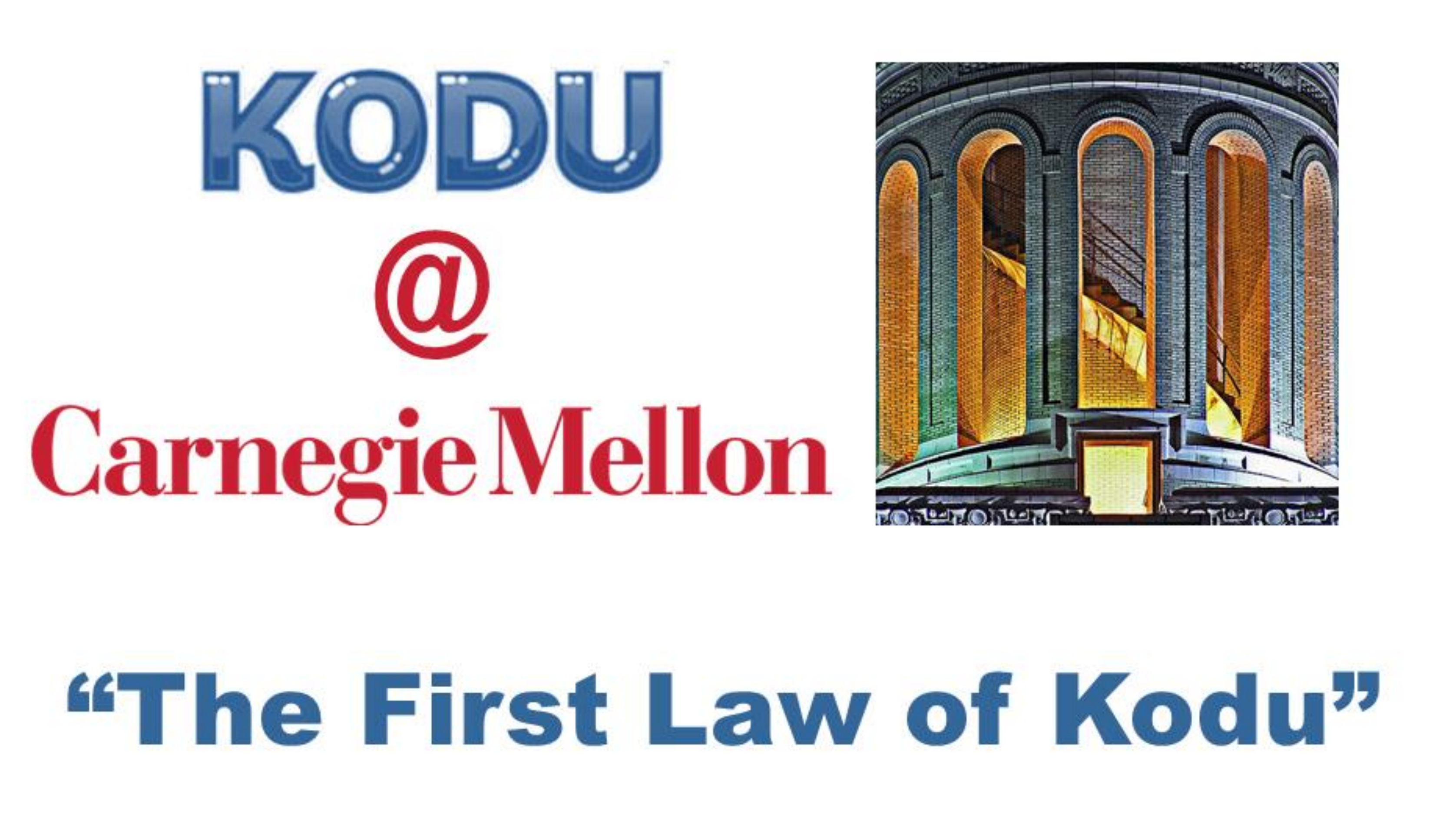
LAWS OF KODU (1): VARIABLE BINDING



First Law of KODU

Each rule picks the closest matching object.

VIDEO: THE FIRST LAW OF KODU



KODU

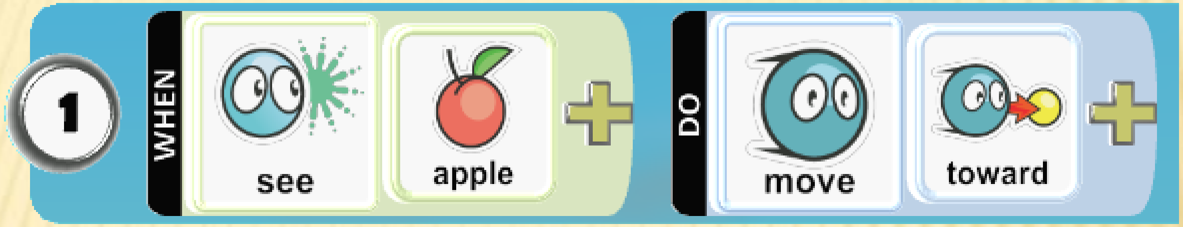
@

Carnegie Mellon

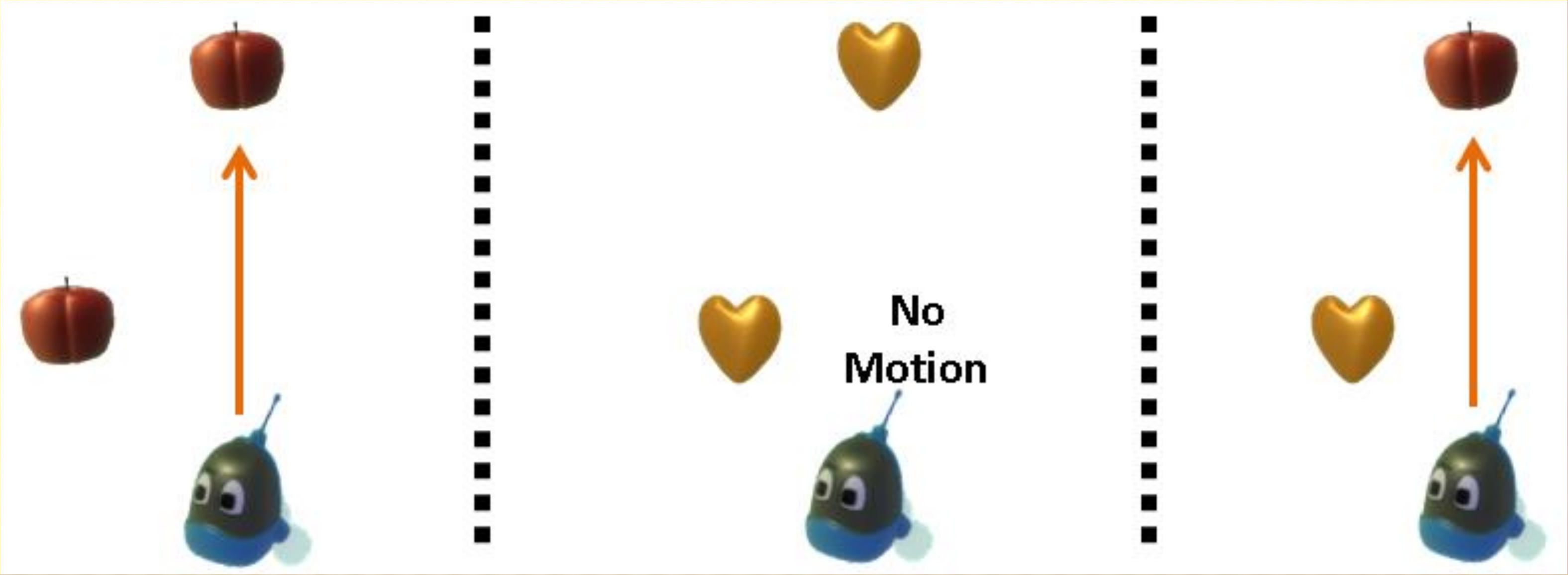
“The First Law of Kodu”

https://www.youtube.com/watch?v=xK_tUcsyNuQ

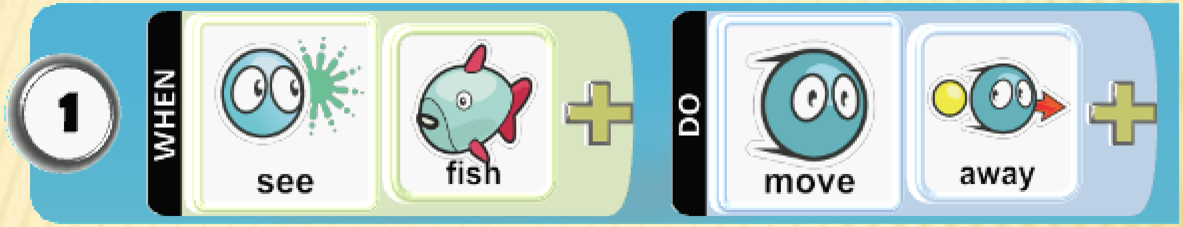
WHICH SCENARIO VIOLATES THE FIRST LAW?

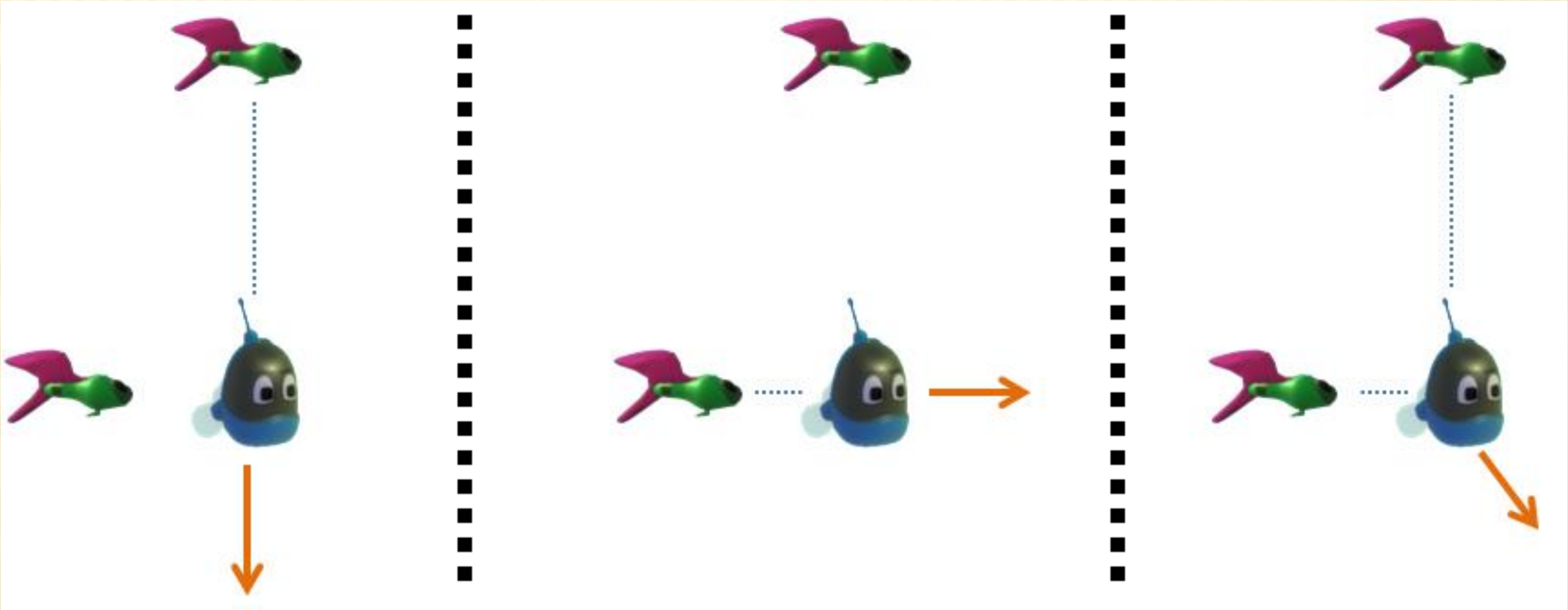


1 WHEN see apple + DO move toward +

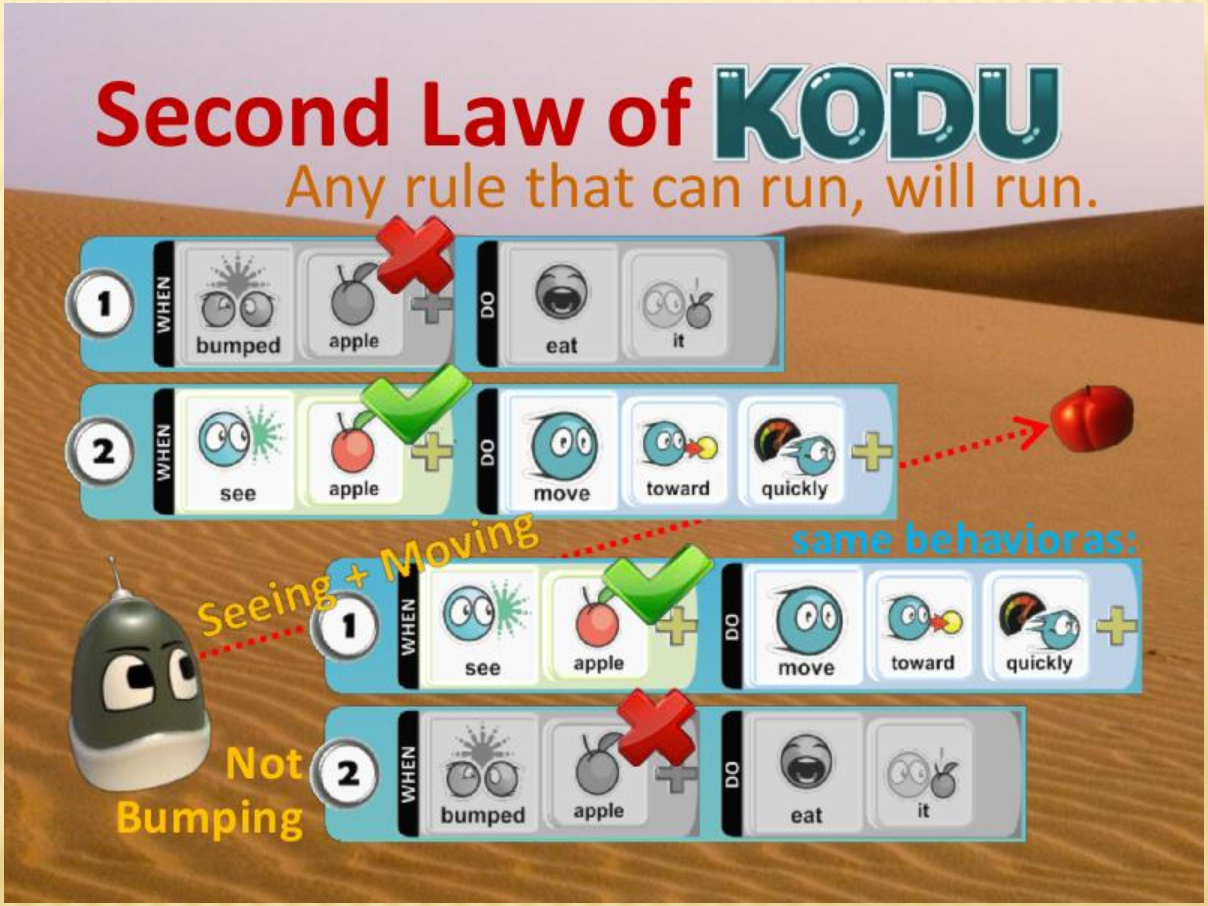


WHICH SCENARIO OBEYS THE FIRST LAW?





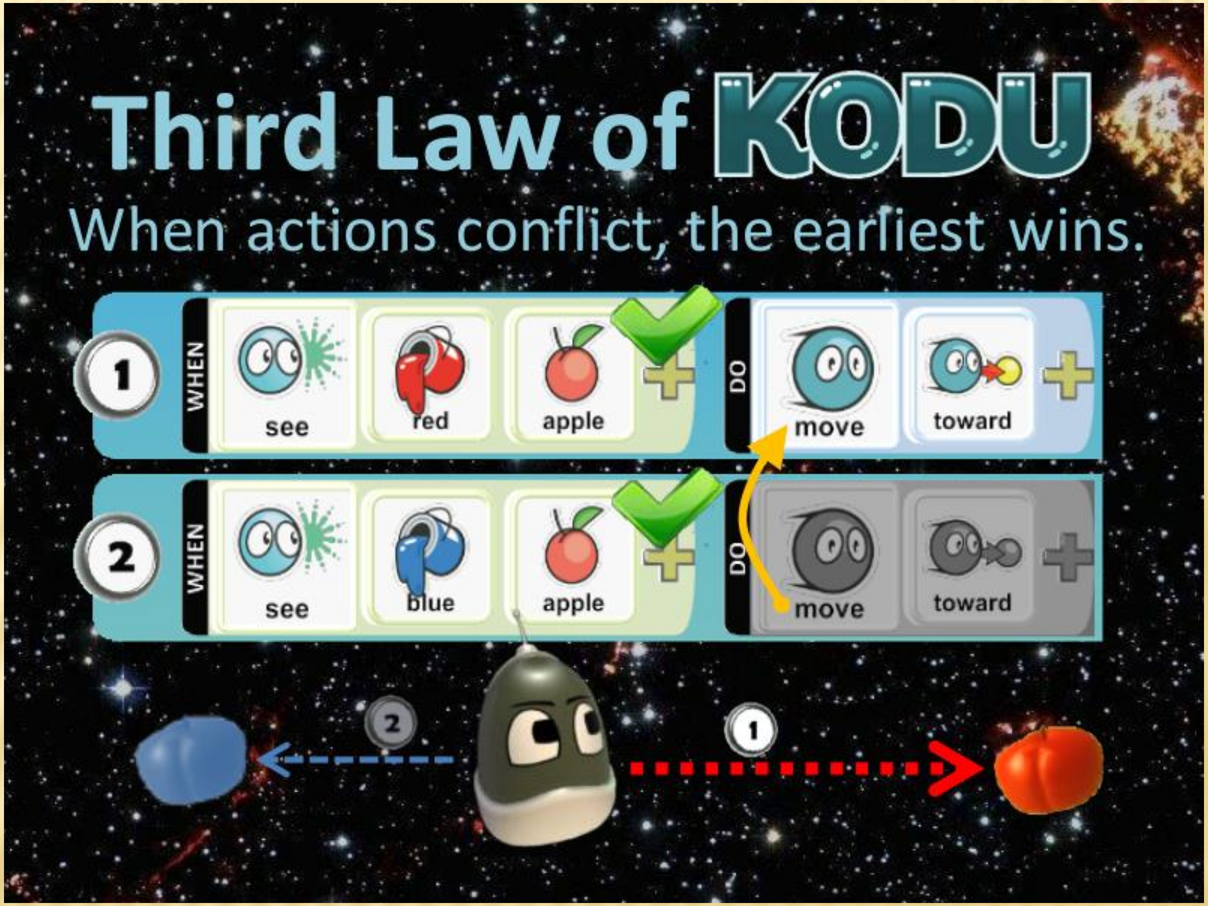
LAWS OF KODU (2): RULE EXECUTION


Second Law of KODU
Any rule that can run, will run.
1 WHEN bumped apple DO eat it
2 WHEN see apple DO move toward quickly
same behavior as:
Seeing + Moving
1 WHEN see apple DO move toward quickly
Not Bumping
2 WHEN bumped apple DO eat it

VIDEO: THE SECOND LAW OF KODU

https://www.youtube.com/watch?v=eEdgnUz6Kac

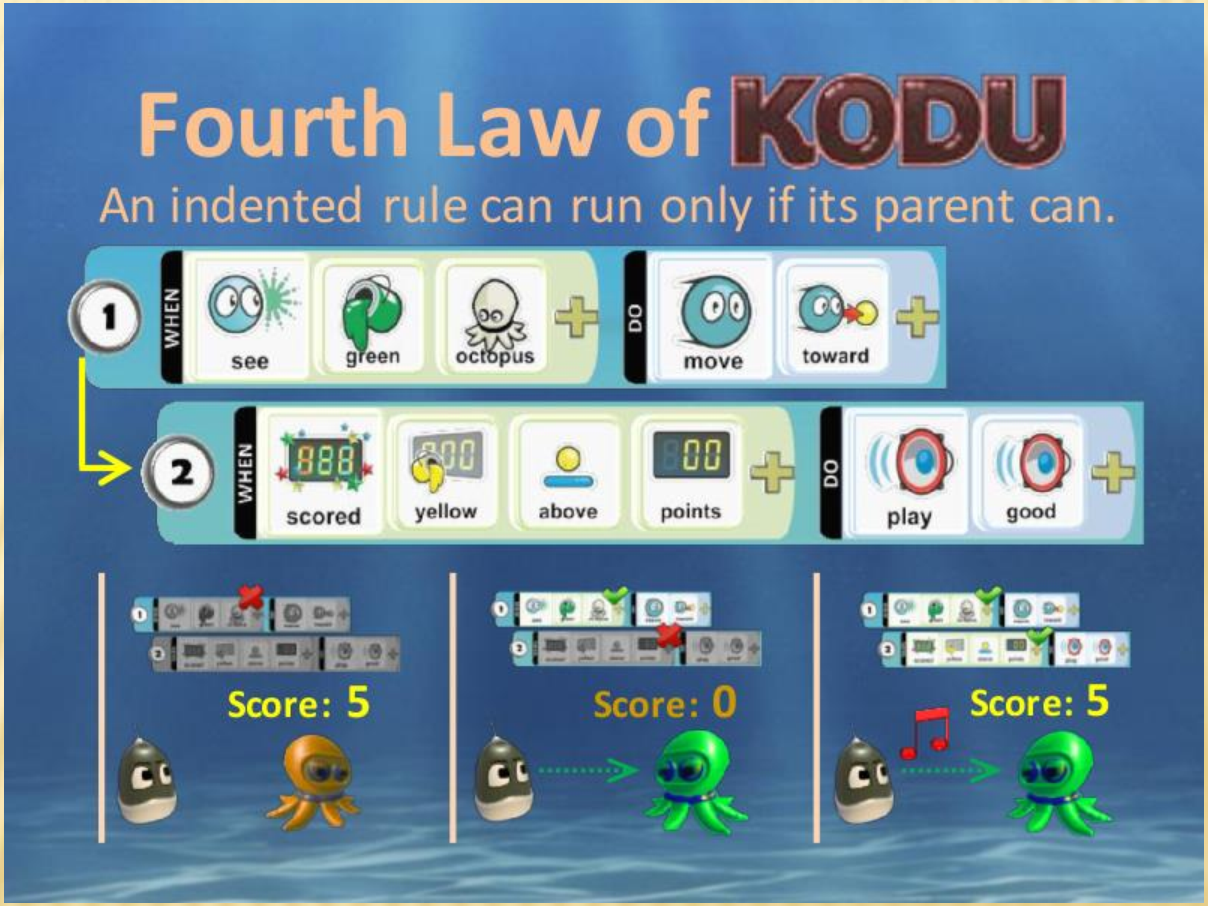
LAWS OF KODU (3): CONFLICT RESOLUTION



Third Law of KODU

When actions conflict, the earliest wins.

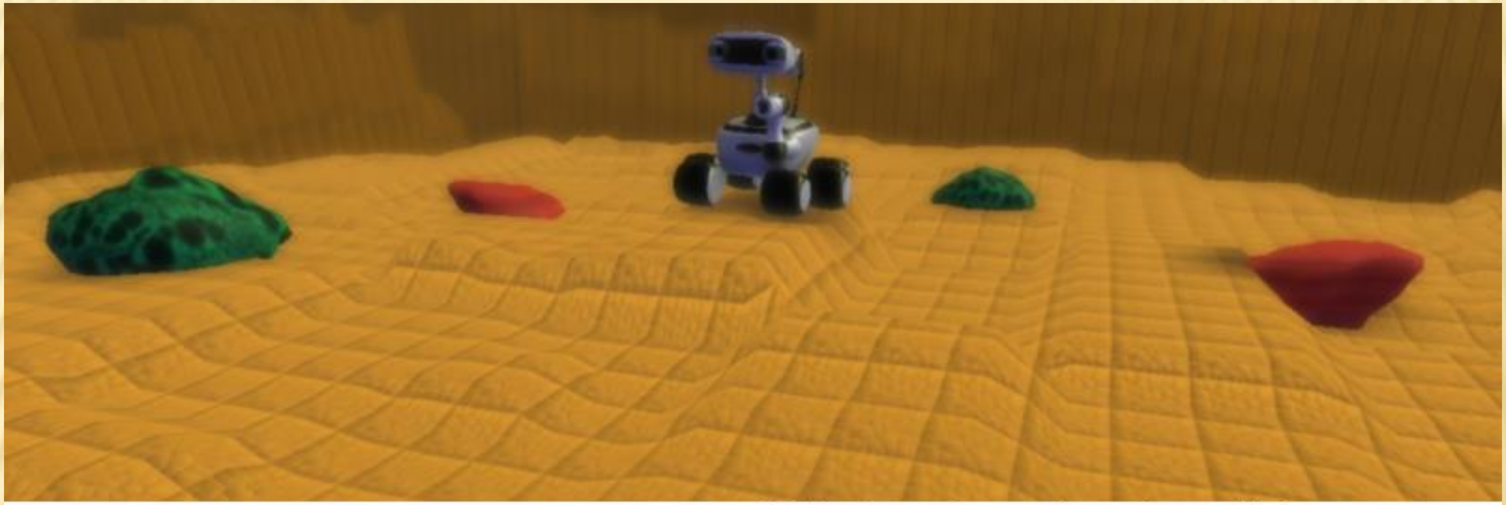
LAWS OF KODU (4): DEPENDENCY

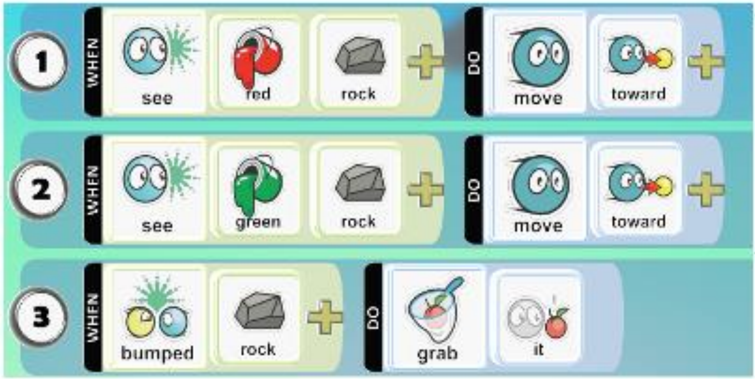


Fourth Law of KODU

An indented rule can run only if its parent can.

EXAMPLES OF REASONING PROBLEMS





With these three rules, what will the rover grab first? Circle your answer.

- **a.** A red rock
- **b.** A green rock
- **c.** It will grab any rock at random.
- **d.** The closest rock no matter what color.

When will the rover grab its first green rock?

- **a.** When the red rocks are gone.
- **b.** Right after it grabs a red rock.
- **c.** It will never grab a green rock; it will keep looking for red rocks.
- **d.** It will only grab a green rock if it bumps into one by accident.

4. Common Fallacies

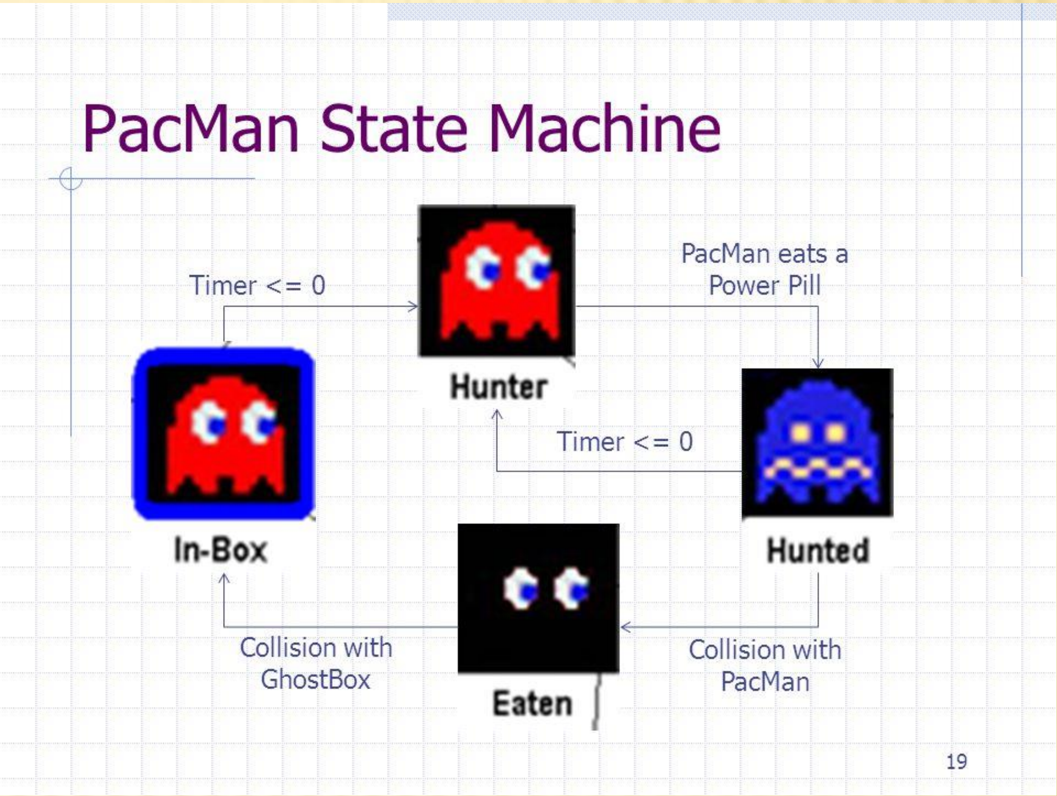
- The sequential procedure fallacy:
  - Students think rules run in the order they're written.
  - This would be true in Scratch or Python.
  - In Kodu, rules can run in any order (Second Law).

- The collective decision fallacy:
  - Students think the rules pick one "closest" object.
  - Actually, each rule makes its own choice (First Law).
  - Rule ordering (Third Law) determines which object is acted upon if the actions conflict.

5. STATE MACHINES

- State machines are found in every area of computer science.
  - Automata theory, digital logic design, network protocols, game design, parsing, robot programming, etc.
- Important tool for describing and reasoning about behavior.
- Most K-12 teachers have never heard of them!

STATE MACHINE FOR GHOSTS IN PACMAN



PacMan State Machine

Timer <= 0

Hunter

PacMan eats a Power Pill

In-Box

Timer <= 0

Hunted

Collision with GhostBox

Eaten

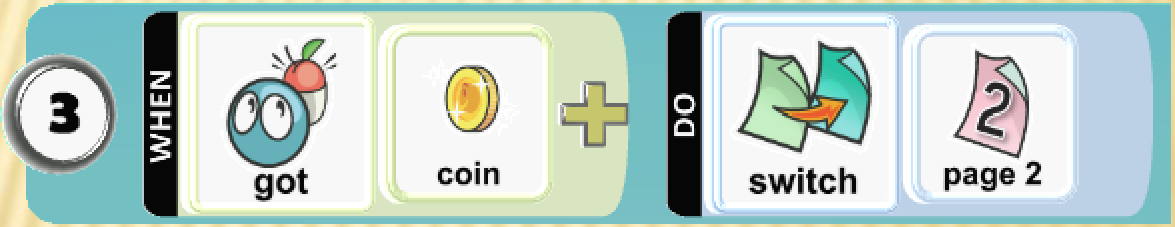
Collision with PacMan

19

Image from http://images.slideplayer.com/11/3228338/slides/slide_19.jpg

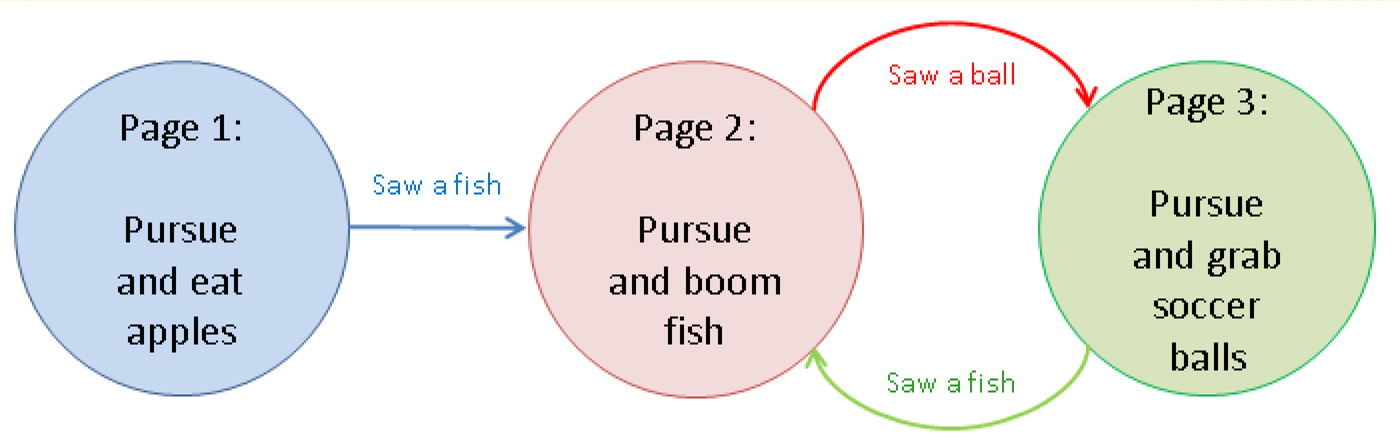
STATE MACHINES IN KODU

- A Kodu program is a set of pages.
- Each page can contain multiple rules.
- The “switch to page” action transfers control from one page to another.



- **Pages are the states.**
- **“Switch to page” rules are the transitions.**

REASONING ABOUT STATE MACHINES



1. After the kodu grabs a soccer ball, will it ever eat another apple?

2. If there are no fish, can the kodu ever grab a soccer ball?

CONCLUSIONS

- Kids should learn to reason about programs:
  - Recognize common design patterns.
  - Know the “laws” of their computational framework.
  - Be able to mentally simulate a program to predict its behavior.
- Kodu is a good framework for teaching this kind of reasoning because:
  - Its idioms and laws are accessible to kids.
  - Kodu programs can be both short and interesting.

FOR MORE INFORMATION

- Microsoft’s Kodu site:

  http://www.kodugamelab.com

- My “Kodu Resources for Teachers” site:

  http://www.cs.cmu.edu/~dst/Kodu

QUESTIONS?