Expressing Computer Science Concepts Through Kodu Game Lab

Kathryn T. Stolee Teale Fristoe

March 10, 2011

Challenge

Learning to program is hard

Challenge

Learning to program is hard

Teaching programming is hard, too

Motivation

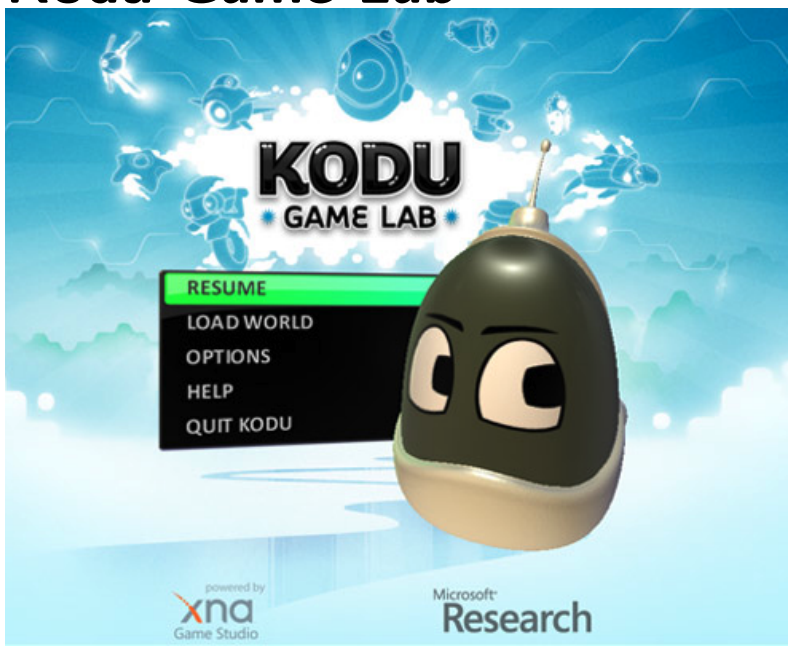
Educational Programming Languages:

- Useful to introduce novices to programming
- Commonly used in a classroom setting
- Used to create video games, simulations, animations, art ...
- Focus is on ease-of-use and attractiveness
- Many examples: Alice, Greenfoot, Scratch ...

About Kodu

- A video game for creating 3d video games
- Designed to compete with modern console games
- Available on Xbox or PC
- Uses an Xbox controller as the interface for playing and programming
- 110,000+ installs in 150+ countries

Kodu Game Lab



Microsoft FUSELABS

University of Nebraska Lincoln

University of California Santa Cruz

How Kodu is Different

- Integrates common gaming concepts (e.g., scoring, camera positioning, termination conditions)
- Programming model does not resemble syntax and abstraction level of mainstream languages

Kodu World

How Kodu is Different

- Integrates common gaming concepts (e.g., scoring, camera positioning, termination conditions)
- Programming model does not resemble syntax and abstraction level of mainstream languages

Kodu World

Little is known about how skills learned in Kodu may transfer to more traditional languages

Interacting with Kodu

Play Mode

- Play the game
- Test or simulate programming logic

Edit Mode

- Modify terrain and objects
- Program characters

Interacting with Kodu

Play Mode

- **Play the game**
- **Test or simulate programming logic**

Edit Mode

- Modify terrain and objects
- Program characters

Kodu World

Interacting with Kodu

Play Mode

- Play the game
- Test or simulate programming logic

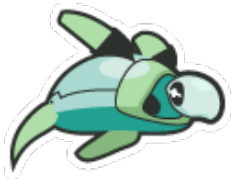
Edit Mode

- **Modify terrain and objects**
- **Program characters**

Kodu

Tree

Turtle



Apple

Programming: Kodu Language

- Language is a high-level, visual, and event-driven
- Statements take the form of 'When – Do' clauses

Programming: Kodu Language

- Language is a high-level, visual, and event-driven
- Statements take the form of 'When – Do' clauses

Rule $\rightarrow$ Condition Action

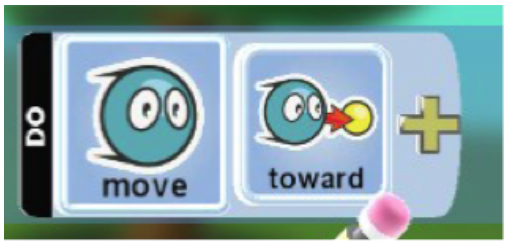
Programming: Kodu Language

- Language is a high-level, visual, and event-driven
- Statements take the form of 'When – Do' clauses

Rule → **Condition** Action

Programming: Kodu Language

- Language is a high-level, visual, and event-driven
- Statements take the form of 'When – Do' clauses



Rule $\rightarrow$ Condition **Action**

Programming: Kodu Language

- Language is a high-level, visual, and event-driven
- Statements take the form of 'When – Do' clauses

Rule $\rightarrow$ **Condition Action**

Programming: User Perspective

- Programming involves visual composition of tiles

Programming: Rule Prioritization

- Rules are ordered and organized into pages
- Conflicting rules resolve action using order

Conflicting Rules

Programming: Page Usage

- Characters can have up to 12 pages of programming
- Switching pages changes character behavior (e.g., a power-up)

Big Question

Big Question

Can traditional computer science concepts (e.g., boolean logic, objects, variables, iteration, control flow) be expressed in Kodu?

Research Questions

RQ1: Which computer science concepts can be expressed through the Kodu Language?

RQ2: How often does each computer science concept appear in the programs created by the Kodu community?

Research Questions

RQ1: Which computer science concepts can be expressed through the Kodu Language?
Kodu Language Analysis

RQ2: How often does each computer science concept appear in the programs created by the Kodu community?

Research Questions

RQ1: Which computer science concepts can be expressed through the Kodu Language?
Kodu Language Analysis

RQ2: How often does each computer science concept appear in the programs created by the Kodu community?
Analysis of ∼350 Kodu Programs

CS Concepts in Kodu

Obviously Supported:

- Objects: Encapsulation, Creation, Deletion
- Control Structures: if – then, iteration
- Variables: Global, Local, Random
- Boolean Logic: Negation

Subtly Supported:

- Objects: Cloning (class system)
- Boolean Logic: Conjunction, Disjunction
- Control Flow: Cycles, Fan-in, Fan-out

Yet to be Investigated:

- Function calls
- ...

CS Concepts in Kodu

Obviously Supported:

- Objects: Encapsulation, Creation, Deletion
- Control Structures: if – then, iteration
- Variables: Global, Local, Random
- **Boolean Logic: Negation**

Subtly Supported:

- Objects: Cloning (class system)
- **Boolean Logic: Conjunction, Disjunction**
- **Control Flow: Cycles, Fan-in, Fan-out**

Yet to be Investigated:

- Function calls
- ...

Boolean Logic: Negation

```
not A
```

Boolean Logic: Negation

`not A`

```
1 when A do B
2 when not A do C
```

Boolean Logic: Negation

`not A`

1 `when A do B`

2 `when` **`not A`** `do C`

Boolean Logic: Conjunction

- Condition
- Action

```
1 when A do B
    2 when C do D
    3 when always do E
```

Boolean Logic: Conjunction

- **Condition**

  $A \wedge C \Rightarrow D$

- Action

```
1 when A do B
    2 when C do D
    3 when always do E
```

Boolean Logic: Conjunction

- Condition

  $A \wedge C \Rightarrow D$

- **Action**

  $A \Rightarrow B \wedge E$

```
1 when A do B
    2 when C do D
    3 when always do E
```

Boolean Logic: Conjunction

- Condition

  $A \wedge C \Rightarrow D$

- Action

  $A \Rightarrow B \wedge E$

```
1 when A do B
    2 when C do D
    3 when always do E
```

Boolean Logic: Disjunction

$$A \vee C \Rightarrow B$$

Boolean Logic: Disjunction

$$\mathtt{A} \vee \mathtt{C} \Rightarrow \mathtt{B}$$

```
1 when A do B
2 when C do B
```

Boolean Logic: Disjunction

$\mathtt{A} \vee \mathtt{C} \Rightarrow \mathtt{B}$

1 when **A** do **B**

2 when **C** do **B**

Expressions of Control Flow

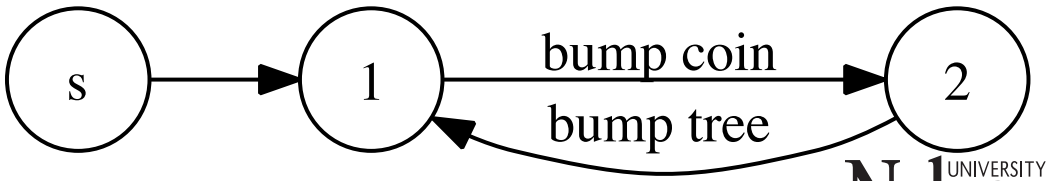
- Pages represent character state in terms of behavior
- Switching pages can create non-linear program flow

Expressions of Control Flow

- Pages represent character state in terms of behavior
- Switching pages can create non-linear program flow

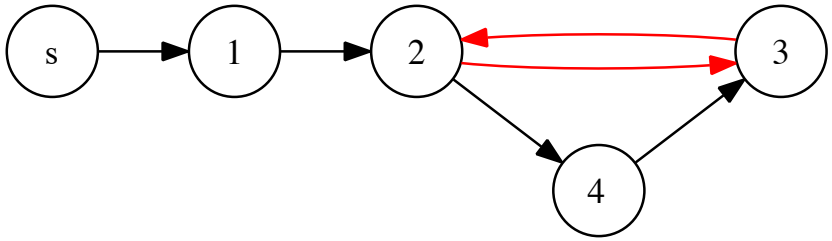
Expressions of Control Flow

- Pages represent character state in terms of behavior
- Switching pages can create non-linear program flow



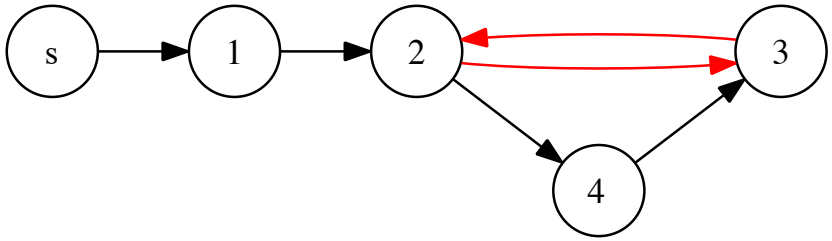
Control Flow Patterns

Cycles (iteration)

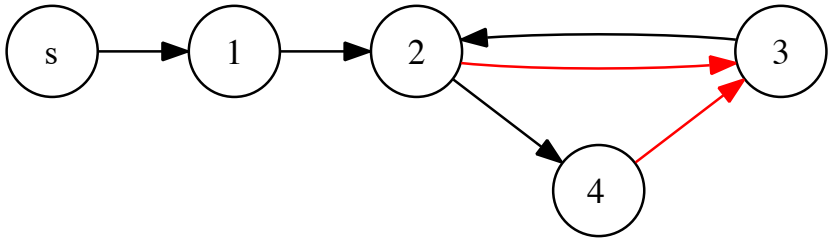


Control Flow Patterns

Cycles (iteration)

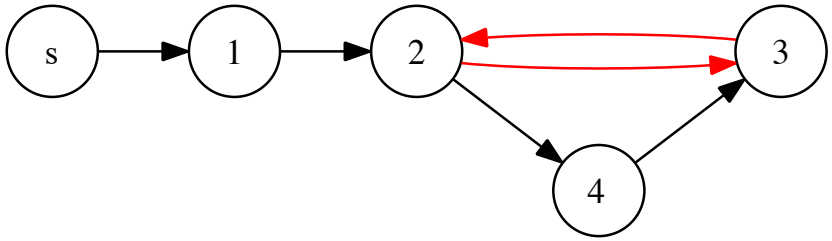


Fan-in (reuse)

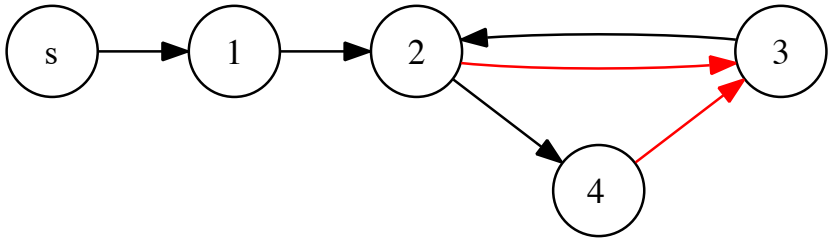


Control Flow Patterns

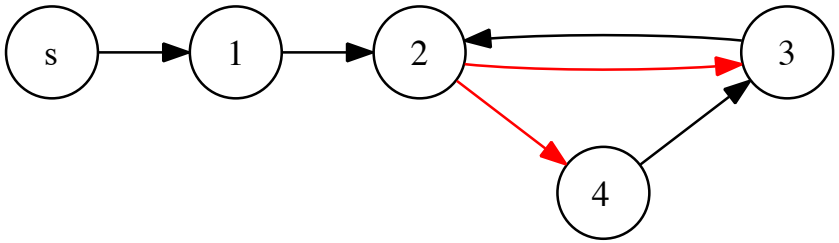
Cycles (iteration)



Fan-in (reuse)



Fan-out (conditional flow)



Research Questions

RQ1: Which computer science concepts can be expressed through the Kodu Language?
Kodu Language Analysis

RQ2: How often does each computer science concept appear in the programs created by the Kodu community?
Analysis of ∼350 Kodu Programs

Profile for Kodu Program

346 programs from the XBox community (13 months of data)

Property	**Average**	**Median**
Rules	109	54
Total Tiles	497	231
Programmed Characters	18	12

Boolean Logic in the Community

CS Concept	**# Games**	**% Games**
Logical *Not*	61	17.6%
If-Then-Else Statements	29	8.4%
Logical *And* Condition[1]	17	20.9%
Logical *And* Action[1]	13	16.0%
Logical *Or*	208	59.9%

[1]The indentation feature needed for the logical *and* was introduced on March 19, 2010. These values consider only the 81 (23.4%) games published after that date.

Boolean Logic in the Community

CS Concept	# Games	% Games
Logical *Not*	61	17.6%
If-Then-Else Statements	29	8.4%
Logical *And* Condition[1]	17	20.9%
Logical *And* Action[1]	13	16.0%
Logical *Or*	208	59.9%

[1]The indentation feature needed for the logical *and* was introduced on March 19, 2010. These values consider only the 81 (23.4%) games published after that date.

Control Flow in the Community

CS Concept	# Games	% Games
2+ States	202	58.4%
Cycles	166	47.9%
Fan-in > 1	174	50.3%
Fan-out > 1	123	35.5%

Control Flow in the Community

CS Concept	# Games	% Games
2+ States	202	58.4%
Cycles	166	47.9%
Fan-in > 1	174	50.3%
Fan-out > 1	123	35.5%

Control Flow in the Community

CS Concept	# Games	% Games
2+ States	202	58.4%
Cycles	166	47.9%
Fan-in > 1	174	50.3%
Fan-out > 1	123	35.5%

Threats to Validity

Internal Relationship between *expressing* concepts in code and *learning* those concepts is unverified.

External Programs were self-selected for uploading to Xbox community and may not be representative of the broader population.

Conclusion

What We Learned:

- Many fundamental computer science concepts *can* be expressed in Kodu
- Users make extensive use of the language and complex language constructs

Questions Left Unanswered:

- Are Kodu users internalizing the computer science concepts that appear in their games? (Requires evaluation)
- Can Kodu be used to teach computer science?

Expressing Computer Science Concepts Through Kodu Game Lab

Kathryn T. Stolee Teale Fristoe

kstolee@cse.unl.edu

teale@soe.ucsc.edu

March 10, 2011