

# Debugging games

A new approach to computing education

Andrew J. Ko

**dub**



Information School  
UNIVERSITY of WASHINGTON



**W**

# Debugging is one of the most difficult parts of software development

Developers must master...

Reproduction

Input minimization

Tools for inspecting runtime behavior

Strategies for localizing defects

# Debugging is even more difficult for **novices**

In the classroom...

We (usually) don't teach them how to do it

We (usually) require them to do it alone

Novices inject a lot of defects

We require them to learn it while also learning algorithms, data structures, and programming languages

# Debugging is even *more* difficult for **teen girls**

Many already believe they're not good at "computers", so they're less likely to persist when they encounter failures

Social experiences are key to engaging girls, but debugging be quite solitary

Goode, J., Estrella, R., & Margolis, J. (2006). Lost in translation: Gender and high school computer science. In J. M. Cohoon & W. Aspray (Eds.) *Women and Information Technology Research on Underrepresentation*, 89- 114.

# If we were to teach debugging...

What would we teach?

How would we teach it?

How would we teach it in a way that better engages and teaches learners who lack self-efficacy, such as girls?

# Existing learning tech teaches coding, not debugging

Creativity + tinkering



Alice

SCRATCH

(and 100's of others in the past 50 years)

All require learners to struggle through the errors they create

Tutorials



Most provide little feedback about errors and no debugging guidance

Games



And dozens of other competitive coding games

All coding oriented, not debugging oriented

We've been exploring a new kind of learning technology we call a **debugging game**

Led by my Ph.D. student Michael Lee, and several others:

Margaret Burnett (Oregon State)

1 postdoc, 3 other Ph.D. students, and 12 undergraduates, and 2 high school students



# Why a game?

91% of U.S. kids aged 2-17 play video games

Social games are particularly popular among teen girls in the U.S.

Teen girls particularly enjoy both cooperative and competitive puzzle and strategy games

NPD (2011). Kids & gaming 2011 report. NPD Group.

Ito, M., Baumer, S., Bittanti, M., boyd, d., Cody, R., Herr B., Horst, H.A., Lange, P.G., Mahendran, D., Martinez, K., Pascoe, C.J., Perkel, D., Robinson, L., Sims, C., and Tripp, L. (2009). *Hanging Out, Messing Around, Geeking Out: Living and Learning with New Media*. Cambridge: MIT Press.



# Participatory design

In 2011, Mike Lee and I gathered together a group of 10-12 year old girls and ideated possible game designs

We arrived at a design in which a player *helps* a computer that's struggling to write correct programs

This way, debugging was about cooperating with a computer to solve a problem rather than fighting with it to make progress

Gidget is sent to clean up a chemical spill

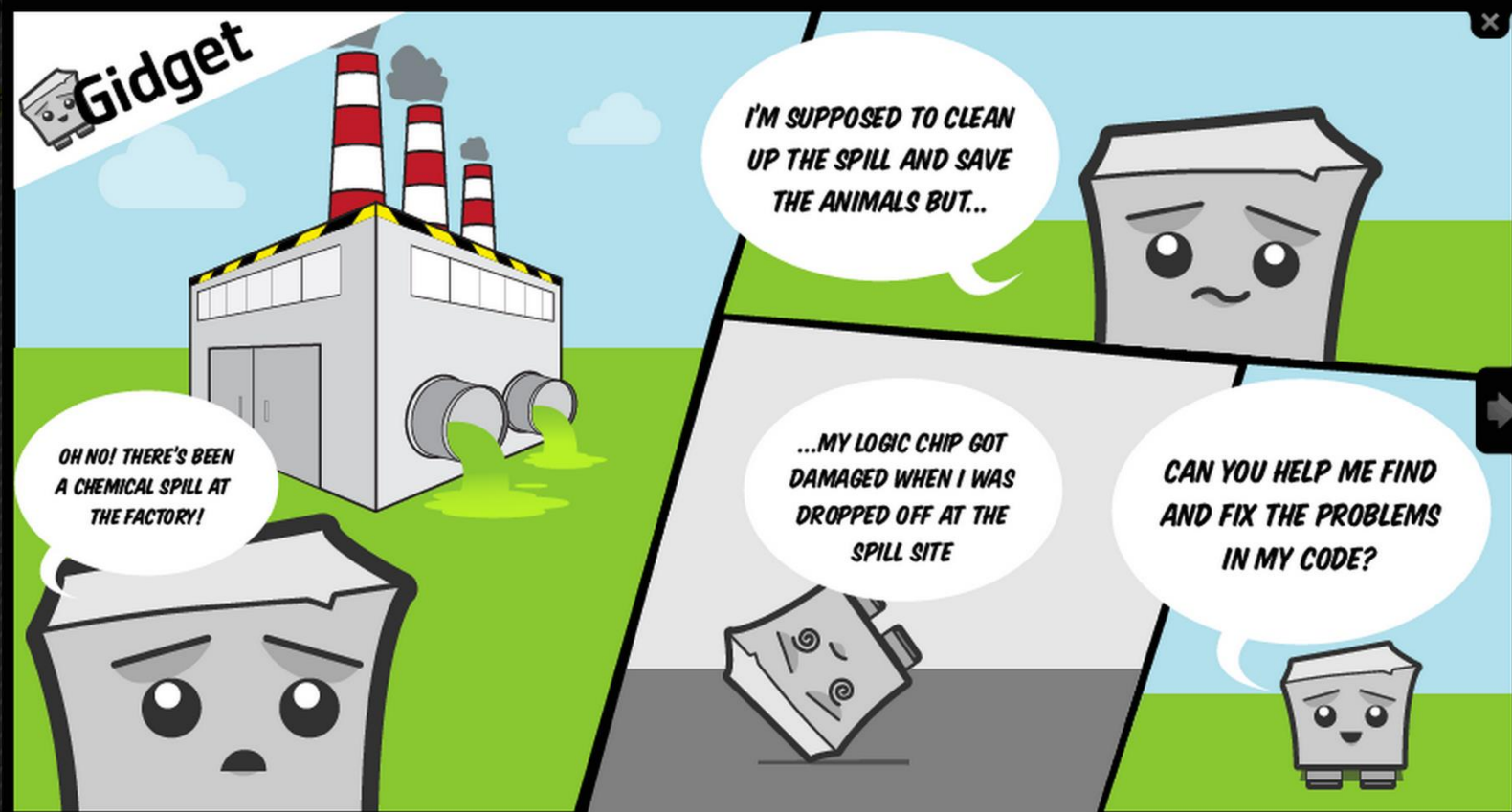
He confides in the player that he every program he writes fails and he's not sure why

Level 1. Let's get to the puppy!

demo  
Logout

Welcome to Gidget!

slide 1 of 9



# A full IDE in the browser

Level 1. Let's get to the puppy!



player1  
Logout

code

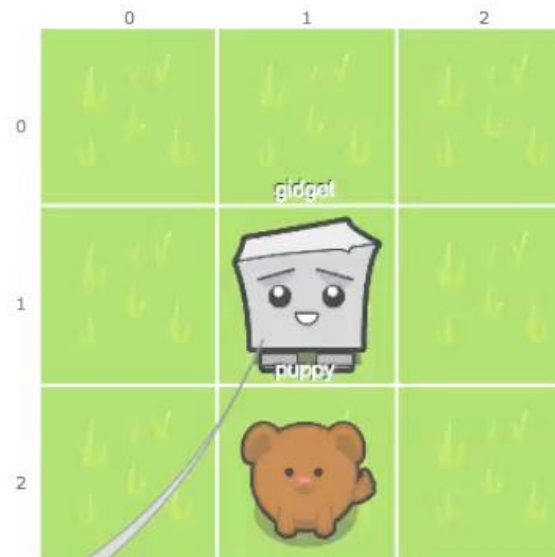
Original Code Clear Code

left

```
ensure /gidget/:position = /puppy/:position
```

One step One line To end Stop!

world



It looks like the goal of this level is to move myself to the **puppy**! Use the buttons on the left to see what my code does, and click on the white area (on the left) to start editing!

← Prev Next →

gidget



energy	100
grabbed	[ ]
image	"default"
labeled	true
layer	1
name	"gidget"
position	[ 1, 1 ]
rotation	0
scale	1
transparency	1

# Each feature designed for learning

Level 20. Press the button, open the gate!



demo  
Logout

## code

Original Code Clear Code

```
goto /button/  
say "Let's click the button to see its function  
/button/:openFence()  
function getPiglet() ?  
  goto /piglet/  
  set /piglet/:nickname to "wilbur"  
  set /piglet/:age to 3  
  grab /piglet/  
getBird() ?  
getThePiggy() ?  
goto /basket/
```

```
ensure /piglet/:nickname = "babe"  
ensure /piglet/:age = 3  
ensure # /piglet/ on /basket/ = 1
```

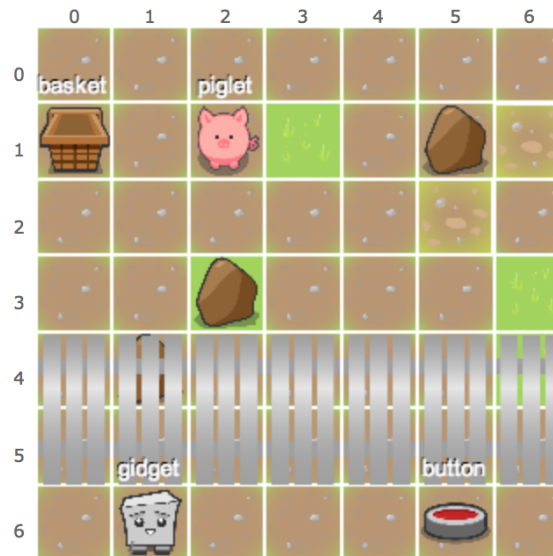
One step

One line

To end

Stop!

## world



Let's figure out how to open the **gate** with the **button**, and give that **piglet** some new properties before we put it in the **basket**! Try running my code first to see what happens!

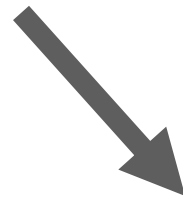
← Prev Next →

## gidget



energy	100
grabbed	[ ]
image	"default"
labeled	true
layer	1
name	"gidget"
position	[ 6, 1 ]
rotation	0
scale	1
transparency	1

Every level is a defective program that the player must repair



**Level 20. Press the button, open the gate!**



**code**

Original Code

Clear Code

```
goto /button/  
say "Let's click the button to see its function  
/button/:openFence()
```

**word**

0

0

basket

1





Programs  
control  
Gidget the  
robot

Written in a  
simple  
Pythonic  
language

## code

Original Code

```
goto /button/  
say "Let's click the button to s  
/button/:openFence()  
function getPiglet()   
    goto /piglet/  
    set /piglet/:nickname to "w  
    set /piglet/:age to 3  
    grab /piglet/  
getBird()   
getThePiggy()   
goto /basket/
```

Programs  
operate on  
objects,  
each with  
properties  
and  
functions

## world



Let's figure out how to open the **gate**  
with the **button**, and give that  
**piglet** some new properties before

```
getThePiggy() ?
```

```
goto /basket/
```

```
ensure /piglet/:nickname = "babe"
```

```
ensure /piglet/:age = 3
```

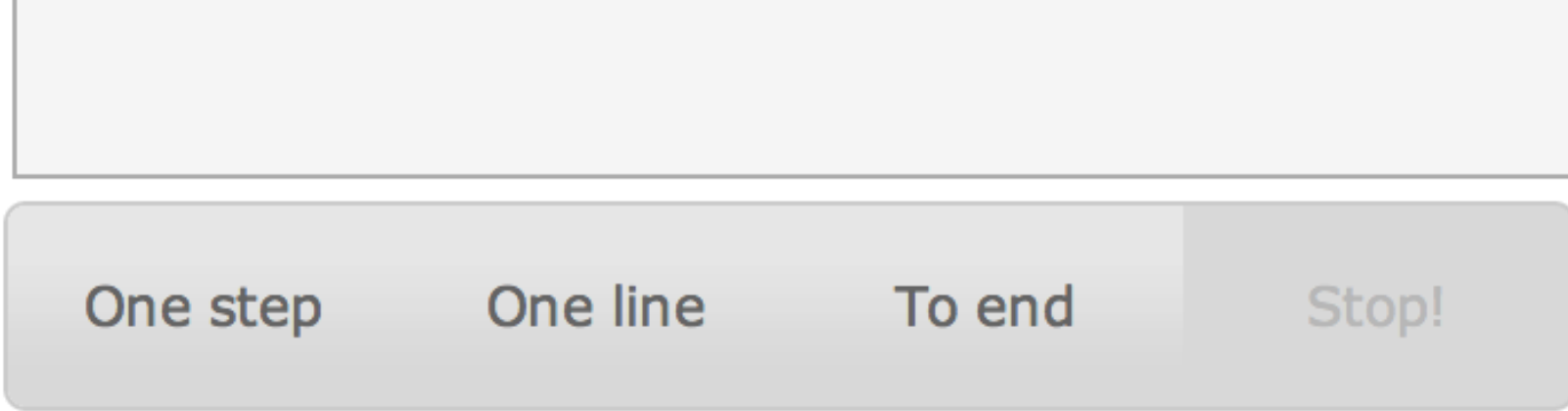
```
ensure # /piglet/ on /basket/ = 1
```

Passing the test cases  
means passing the level



Level  
with  
**pig**  
we  
my





— Program can be executed one instruction at a time, showing player exactly how the program executes



Let's figure out how to open the **gate** with the **button**, and give that **piglet** some new properties before we put it in the **basket**! Try running my code first to see what happens!

← **Prev**      **Next** →

Gidget provides explanations about language semantics and goals after each instruction executes

# gidget



energy	100
grabbed	[ ]
image	"default"
labeled	true
layer	1
name	"gidget"
position	[6, 1]
rotation	0
scale	1
transparency	1

Object state  
and call stack  
are fully  
inspectable

Each  
instruction's  
operations are  
highlighted and  
explained

# Context-sensitive documentation on language syntax and semantics avoid the need for tools like Stack Overflow

The screenshot shows a programming environment with a green background and a grass pattern. At the top left, it says "Level 1. Let's get to the puppy!". On the right, there are icons for a book, a play button, and a gear, along with the text "demo Logout".

The main area is divided into three sections:

- code**: A text area with the word "left" highlighted in yellow. Below it, there's a line of code: "ensure /gidget/:position =".
- Dictionary**: A scrollable list of keywords including "=", "#", "addition", "and", "create", "done", "down", "drop", "else", "energy", "ensure", "eol", "expression", "first", "for", "function", "goto", "grab", and "grabbed".
- function**: A detailed documentation panel for the "function" keyword. It includes a definition: "A block of code that can be defined then called upon to run whenever. Functions can have parameters which can be filled in with values when called. Functions may also be tied to objects in the world. Example(s): This code will make Gidget go to the first puppy." Below this is a code block showing a function definition: "function sortThis(anAnimal, aPlace) !This line starts the definition of a function with two parameters called anAnimal and aPlace goto anAnimal !Gidget will go to the value that is passed into anAnimal grab anAnimal !Gidget will grab the value that is passed into anAnimal goto aPlace !Gidget will go to the value that is passed into aPlace drop anAnimal !Gidget will drop the object that is passed into anAnimal sortThis(/puppy/,/basket/) !Gidget calls the function, passing /puppy/ and /basket/ into anAnimal and aPlace, respectively." At the bottom of the panel is a question mark icon and the text "hover over the icon to see more!".

On the right side, there's a vertical list of values: "100", "[ ]", "default", "true", "1", "gidget", "[1, 1]", "0", "1", "1".

At the bottom, there are navigation buttons: "One step", "One line", "To end", "Prev", and "Next".

# In-context instructional hints on design patterns and debugging strategies

## code

Original Code Clear Code

```
goto /button/  
say "Let's click the button to see its function  
/button/:openFence()  
function getPiglet() ?  
  goto /piglet/  
  set /piglet/:nickname  
  set /piglet/:age to 3  
  grab /piglet/  
getBird() ?  
getThePiggy() ?  
goto /basket/
```

```
ensure /piglet/:nickname  
ensure /piglet/:age = 3  
ensure # /piglet/ on /bask
```

## world



### drag to reposition

#### Here's a hint...

Are you trying to write and use functions? Try something like this:

```
function reset(myobject) ! These lines are  
  goto myobject ! for step 1  
  set myobject:scale to 1  
reset(/goop/) ! the function can be used for the goop,  
reset(/bird/) ! then reused for the bird!
```

Functions allow you to write code once and then use it multiple times by referring to its name.









1. First, write the function definition.
  - 1.1 Type the keyword "function".
  - 1.2 On the same line as "function", type the name you want to give to the function, followed by parentheses.
  - 1.3 If you want to use parameters, type the

gate  
piglet

put it  
code

Code Clear Code

## world

	0	1	2	3	4	5
0	bucket		bird			kitten
1					gidget	
2	piglet					
3			goop		basket	
4				puppy		
5						

Can you help me determine where the **goop** will end up after running the

## gidget



Where will the **goop** be after I execute the current code (assuming I have unlimited energy)? Please click on the grid, then press the button below to check!

Can you tell me how you arrived at your answer? It will help me with my logic chip repairs!

0 words written.

In-game assessments provide positive feedback on learning, testing ability to mental simulate program execution and language semantics



# Playing the game is equivalent to debugging

Understand the tests

Execute the program

Reproduce the problem

Localize the defects(s)

Write a patch that passes  
the tests

## code

```
goto /button/  
say "Let's click  
/button/:openF  
function getPigl  
  goto /piglet/  
  set /piglet/:r  
  set /piglet/:a  
  grab /piglet/  
getBird() - ?  
getThePiggy() -  
goto /basket/
```

```
ensure /piglet/:ni  
ensure /piglet/:ag  
ensure # /piglet/
```



# Curriculum

Across 27 levels, players learn

Variables, conditionals,  
loops, functions, object-  
orientation

Reproduction, testing  
concepts, procedural algorithm  
design, debugging strategies

## code

```
goto /button/  
say "Let's click  
/button/:openF  
function getPigl  
  goto /piglet/  
  set /piglet/:r  
  set /piglet/:a  
  grab /piglet/  
getBird() - ?  
getThePiggy() -  
goto /basket/
```

```
ensure /piglet/:ni  
ensure /piglet/:ag  
ensure # /piglet/
```



# Do players learn?

Yes!


Pre-post test into the game to measure CS1 learning gains

**Preliminary results.** Just 5 hours of game play produces comparable learning outcomes to weeks of CS 1 instruction


*Study in progress*

# Why is it effective?

The game redirects player's attention to contextually appropriate instruction by framing Gidget as a collaborator

 Looked for **bucket** to **scan** and detected a **bucket**. Added it to the results list.

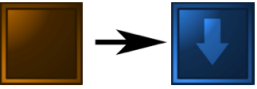

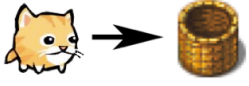
I looked for **bucket** to **scan** and detected a **bucket**. I'm going to add it to my results list!



When Gidget has a face and uses personal pronouns, players play twice as long and repair defects twice as fast

# Why is it effective?

The game leverages recent work on preferential attention to focus player attention on the right data

Condition	Goal (Level 1)	Respective Game Images
Inanimate	<i>block on bin</i>	
Invertebrate	<i>beetle on jar</i>	
Vertebrate	<i>kitten on basket</i>	

Players who manipulate animate objects play twice as long and fast as players who manipulated inanimate objects

Lee, M.J. and Ko, A.J. (2012). Investigating the Role of Purposeful Goals on Novices' Engagement in a Programming Game. IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), 163-166.

# Why is it effective?

Contextual instruction on debugging and problem solving:

Used to immediately attend to & fix anti pattern and potential errors

Used to create a mental to-do list

Preparing for publication



The screenshot shows a code editor with the following code:

```
goto /button/  
say "Let's click the button to see its function"  
/button/:openFence()  
function getPiglet() ?  
  goto /piglet/  
  set /piglet/:nickname  
  set /piglet/:age to 3  
  grab /piglet/  
getBird() ?  
getThePiggy() ?  
goto /basket/
```

Below the code, there is a section with the following code:

```
ensure /piglet/:nickname  
ensure /piglet/:age = 3  
ensure # /piglet/ on /basket/
```

At the bottom, there are two buttons: "One step" and "One line".

On the right side, there is a hint box with a green header "drag to reposition". The text inside the hint box reads:

**Here's a hint...**  
Are you trying to write and use a function like this:

```
function reset(myobject) ! T  
  goto myobject ! f  
  set myobject:scale to 1  
reset(/goop/) ! the functio  
reset(/bird/) ! then reuse
```

Below the hint box, there is a section with the following text:

Functions allow you to write code multiple times by referring to a function.

1. First, write the function definition.
  - 1.1 Type the keyword "function".
  - 1.2 On the same line as "function", type the name of the function, followed by parentheses.
  - 1.3 If you want to use parameters, type the parameter(s) in the parentheses.
  - 1.4 Indent the commands that this function is called.
2. Then, call the function.
  - 2.1 Type the name of the function, followed by the values you want to pass.

# Is it fun?

Lee, M.J. and Ko, A.J. (2011). Personifying Programming Tool Feedback Improves Novice Programmers' Learning. International Computing Education Research Workshop (ICER), 109-116.

Over 500 rank novice programmers have played on Mechanical Turk for an average of ~60 minutes (paid \$0.05/level)

*"It did not even seem like I was learning programming. It truly felt like I was just playing a game. I tend to become frustrated easily yet this held my attention and made it so I didn't want to give up."*

# Is it fun?

Charters, P., Lee, M.J., Ko, A.J., Loksa, D. (2014). Challenging Stereotypes and Changing Attitudes: The Effect of a Brief Programming Encounter on Adults' Attitudes toward Programming (2014) ACM Symposium on Computer Science Education,

In pre/post surveys from 200 players:

Attitudes prior to the game were negative

*“Programming is complicated and boring.”*

Attitudes toward programming improved significantly, becoming positive

*“I now know that programming can be fun and easy, also anyone can do it.”*

Change occurred regardless of gender, population density, or level of education.

# 4 week long summer camps

Two co-ed in Corvallis, OR

Two girls only in Seattle, WA

72 teens age 12-17

50 girls signed up

Played the game in pairs

3 days to  
complete game

2 days to design  
new levels to  
challenge friends  
and family

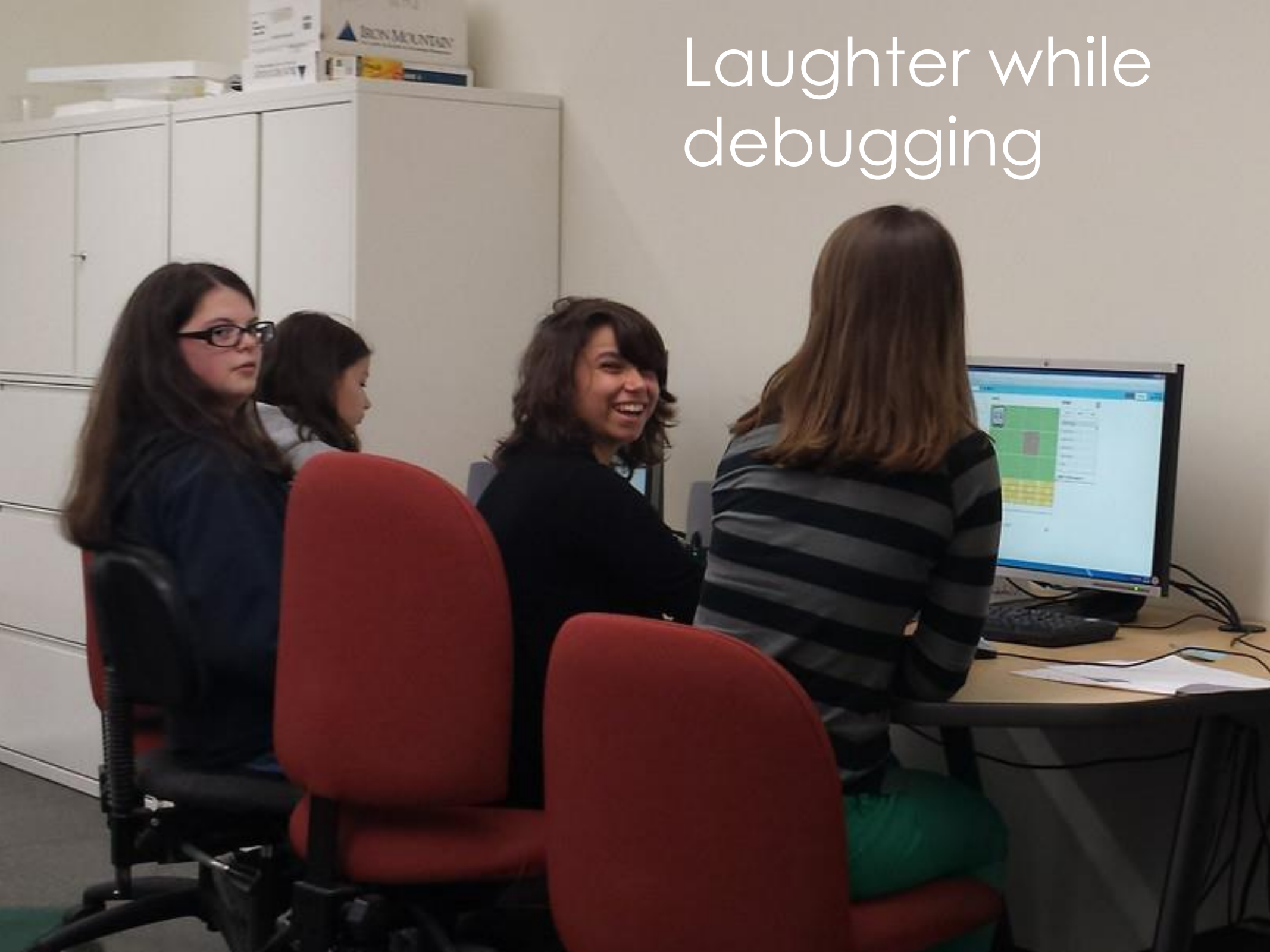
Lee, M.J., Bahmani, F., Kwan, I., LaFerte, J., Charters, P., Horvath, A., Luor, F., Cao, J., Law, C., Beswetherick, M., Long, S., Burnett, M.M., Ko, A.J. (2014). Principles of a Debugging-First Puzzle Game for Computing Education. IEEE Symposium on Visual Languages and Human-Centric Computing, to appear.



Laughter  
during a  
pre-test



Laughter while  
debugging



Laughter while writing  
new levels





Learners strongly motivated by to challenging their parents

# Future work

Can debugging games teach advanced programming languages and skills?

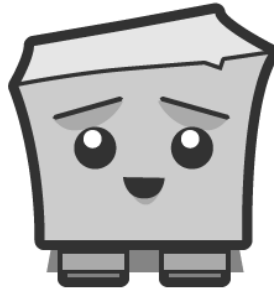
Can we generate puzzles that optimize learning *and* engagement?

Can we improve learning through pair debugging?

Can we engage player's social networks through viral learning?

*Public release this Fall*

**helpgidget.org**



Hi! I'm Gidget.

I'm going on a mission soon, but sometimes I make silly mistakes and can't fix them by myself. Will you come back and help me when I'm ready to go?

# Questions?

This work was generously supported by Microsoft Research and the National Science Foundation (NSF) under Grants CNS- 1240786, CNS-1240957, CNS-1339131, CCF-0952733, CCF- 1339131, IIS-1314356, IIS-1314384, and OISE-1210205.

Any opinions, findings, conclusions or recommendations are those of the authors and do not necessarily reflect the views of NSF.