Debugging games
A new approach to computing education

Andrew J. Ko
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

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

Tutorials

Games

All require learners to struggle through the errors they create

Most provide little feedback about errors and no debugging guidance

And dozens of other competitive coding games

All coding oriented, not debugging oriented

Alice

touchdevelop

codecademy

scratch

Khan Academy

Code Hunt

Code Combat

Lightbot

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


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 every program he writes fails and he's not sure why.
A full IDE in the browser

Level 1. Let's get to the puppy!

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!
Each feature designed for learning

Level 20. Press the button, open the gate!

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

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!
Every level is a defective program that the player must repair.

Level 20. Press the button, open the gate!

code

goto /button/
say "Let's click the button to see its function
/button/:openFence()"
Programs control Gidget the robot

Written in a simple Pythonic language

goto /button/
say "Let's click the button to see if the robot opens the fence.
/button/::openFence()

function getPiglet()
  goto /piglet/
  set /piglet/::nickname to "weird pig"
  set /piglet/::age to 3
  grab /piglet/

  getBird()
  getThePiggy()

  goto /basket/
Programs operate on objects, each with properties and functions.

Let's figure out how to open the gate with the button, and give that piglet some new properties before
Passing the test cases means passing the level
Program can be executed one instruction at a time, showing player exactly how the program executes.
Gidget provides explanations about language semantics and goals after each instruction executes.
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.
In-context instructional hints on design patterns and debugging strategies

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/
```

world

```
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 parameter list within the parentheses.

   ```
   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!
   ```
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
Curriculum

Across 27 levels, players learn

- Variables, conditionals, loops, functions, object-orientation
- Reproduction, testing concepts, procedural algorithm design, debugging strategies
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.

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.

<table>
<thead>
<tr>
<th>Condition</th>
<th>Goal (Level 1)</th>
<th>Respective Game Images</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inanimate</td>
<td>block on bin</td>
<td></td>
</tr>
<tr>
<td>Invertebrate</td>
<td>beetle on jar</td>
<td></td>
</tr>
<tr>
<td>Vertebrate</td>
<td>kitten on basket</td>
<td></td>
</tr>
</tbody>
</table>

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

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
Is it fun?

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?

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

Laughter during a pre-test
Laughter while debugging
Laughter while writing
new levels
Learners strongly motivated by 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?
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.