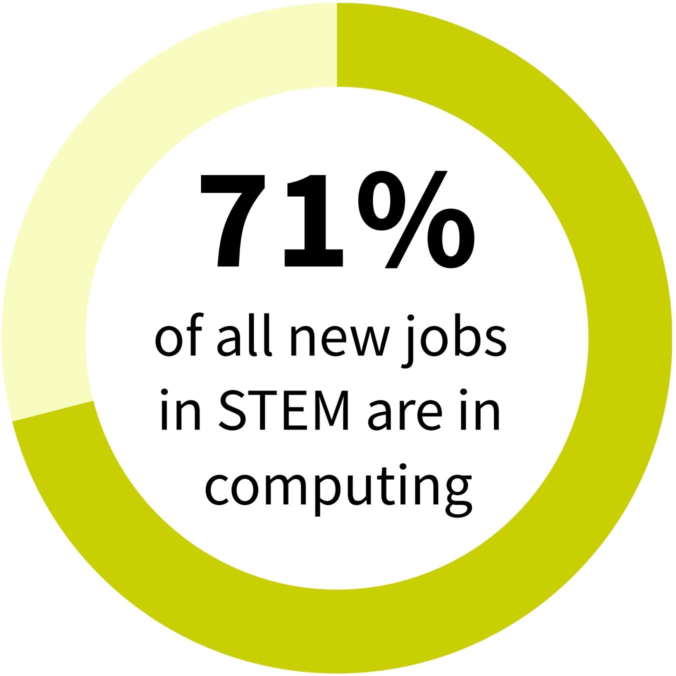
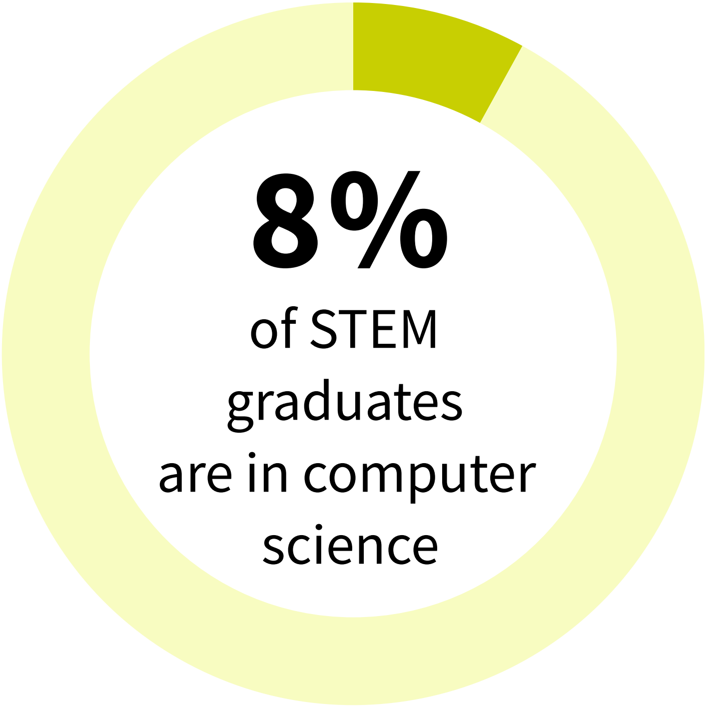
Code Puzzle Completion Problems in Support of Learning Programming Independently

Kyle J. Harms

Washington University in St. Louis

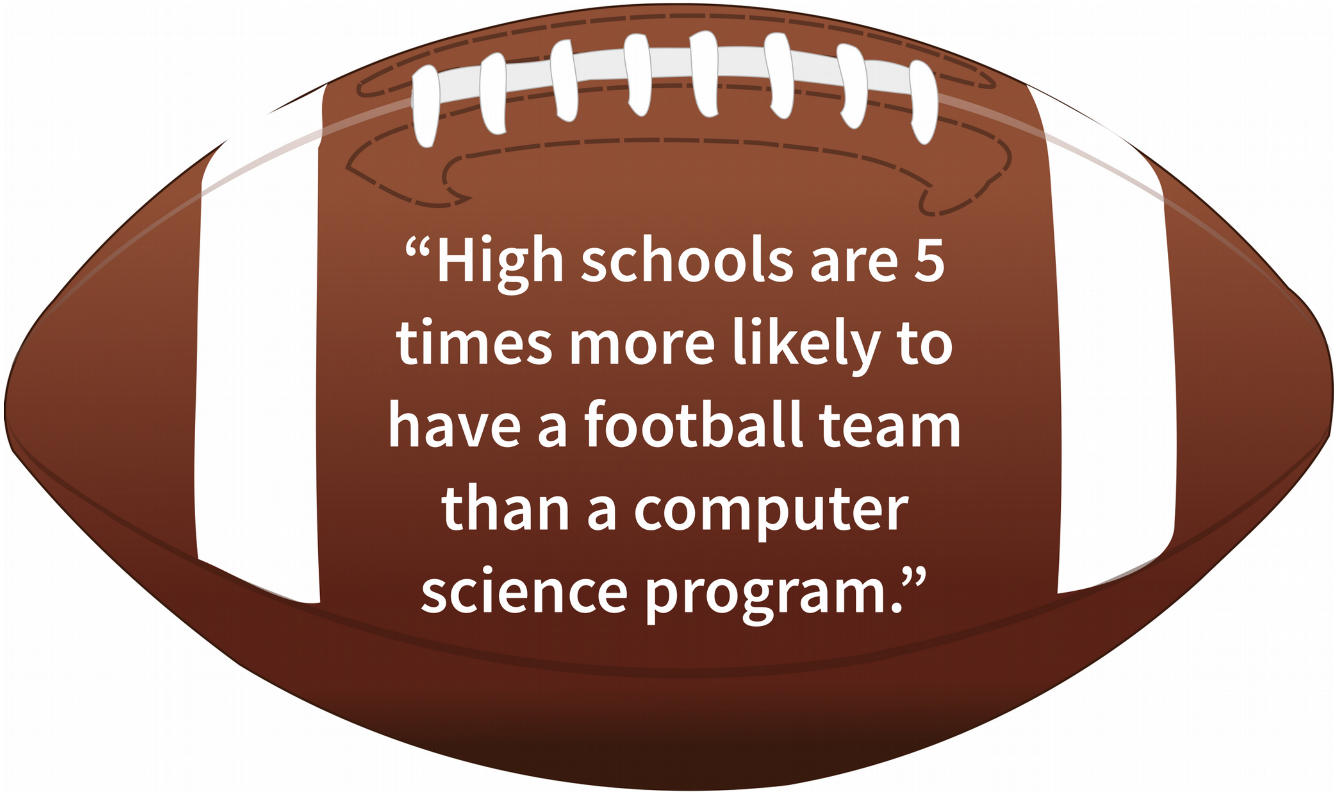
Lots of Jobs... Too Few Computer Science Graduates





"Why computer science in K-12," Code.org, https://code.org/promote, 2016.

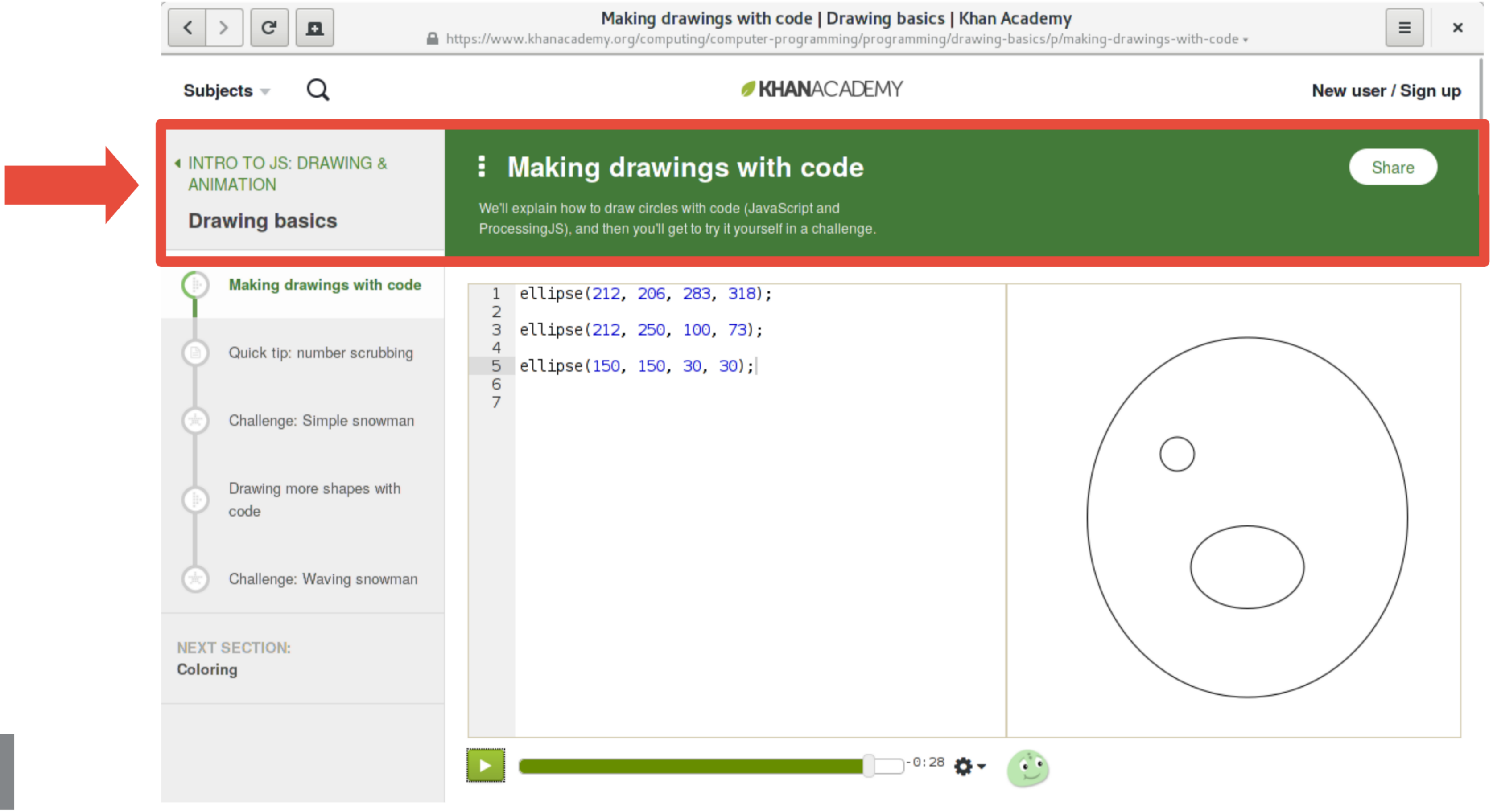
K-12: Lack of Access to Learn Programming



Kurose, Jim. "An Expanding and Expansive View of Computing". 2016. Presentation. https://engineering.wustl.edu/Events/Pages/CSE-Colloquia-Series-Jim-Kurose.aspx

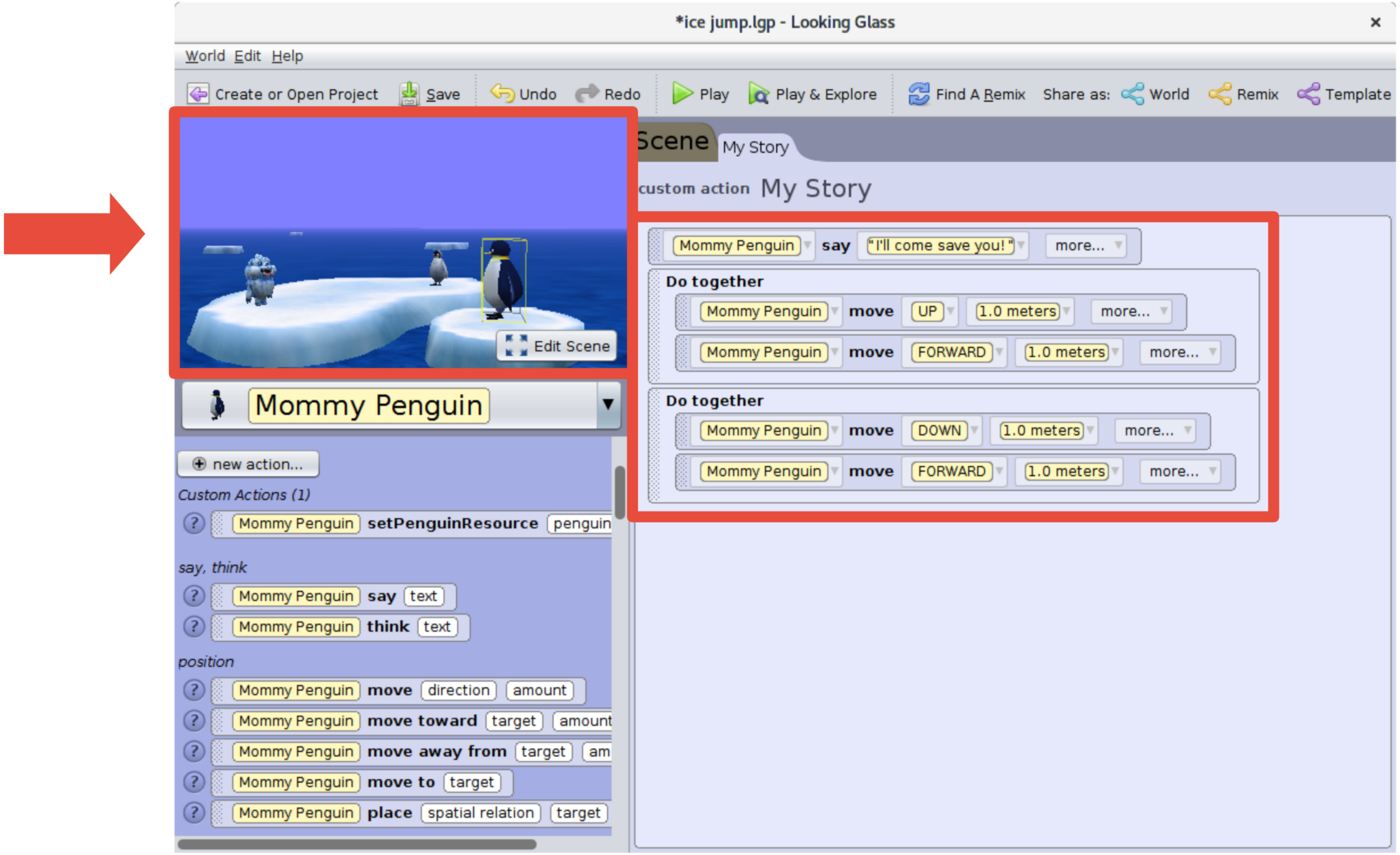
How can we help children learn programming independently?

Massive Open Online Courses



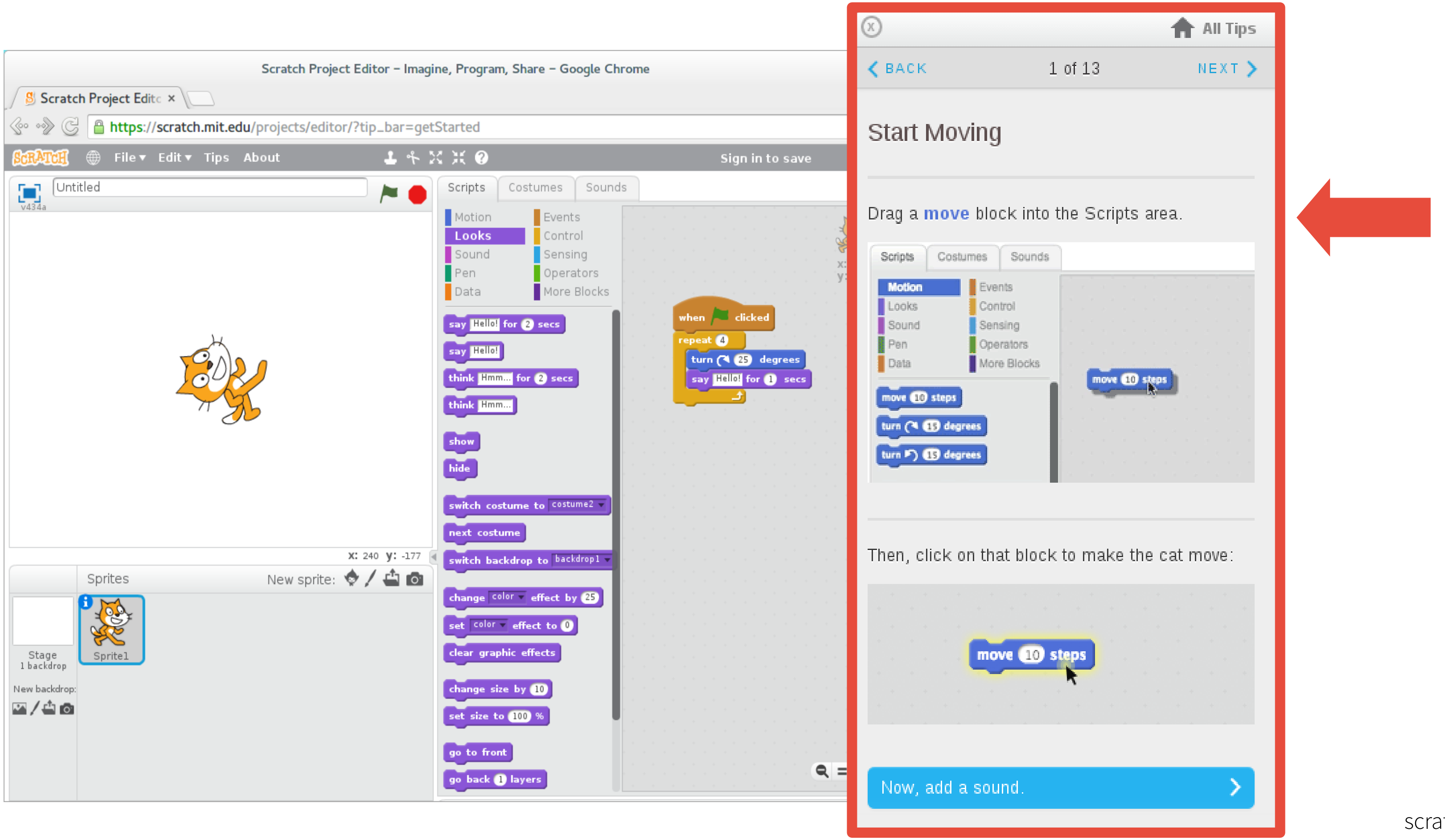
www.khanacademy.org

Motivating Novice Programming Environments



lookingglass.wustl.edu

Programming Tutorials



scratch.mit.edu

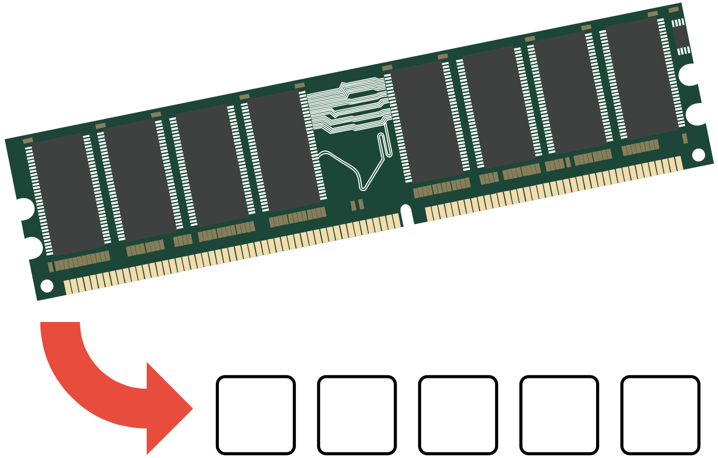
Code Puzzles

code.org

Cognitive Load Theory

J. Sweller and P. Chandler, “Why Some Material Is Difficult to Learn,” Cognition and Instruction, vol. 12, no. 3, pp. 185–233, 1994.
Sweller, P. Ayres, and S. Kalyuga, Cognitive Load Theory. Springer, 2011.

Human Cognitive Architecture

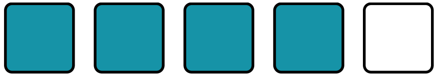


Working Memory **Long Term Memory**

G. A. Miller, “The magical number seven, plus or minus two: some limits on our capacity for processing information,” Psychological Review, vol. 63, no. 2, pp. 81–97, 1956.

Cognitive Load

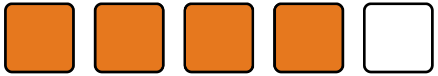
Beneficial



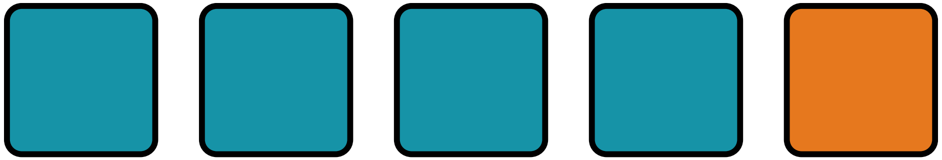
Intrinsic Cognitive Load

Germane Cognitive Load

Detrimental



Extraneous Cognitive Load



Working Memory

Sweller, P. Ayres, and S. Kalyuga, Cognitive Load Theory. Springer, 2011.

Beneficial Cognitive Load

Write a program that sorts an array of integers.

```
public static void bubbleSort( int[] elements ) {
    int tmp;
    for( int i = 0; i < ( elements.length - 1 ); i++ ) {
        for( int j = 1; j < ( elements.length - i ); j++ ) {
            if( elements[ j - 1 ] > elements[ j ] ) {
                tmp = elements[ j - 1 ];
                elements[ j - 1 ] = elements[ j ];
                elements[ j ] = tmp;
            }
        }
    }
}
```

Low Beneficial Cognitive Load

Write a program that sorts an array of integers. **When you are finished, comment your code!**

```
public static void bubbleSort( int[] elements ) {
    int tmp; // swap variable

    // check all elements in the array
    for( int i = 0; i < ( elements.length - 1 ); i++ ) {

        // move the next largest element to the end
        for( int j = 1; j < ( elements.length - i ); j++ ) {

            // compare adjacent elements; swap elements if
            // they are in reverse order
            if( elements[ j - 1 ] > elements[ j ] ) {

                // swap the elements
                tmp = elements[ j - 1 ];
                elements[ j - 1 ] = elements[ j ];
                elements[ j ] = tmp;
            }
        }
    }
}
```

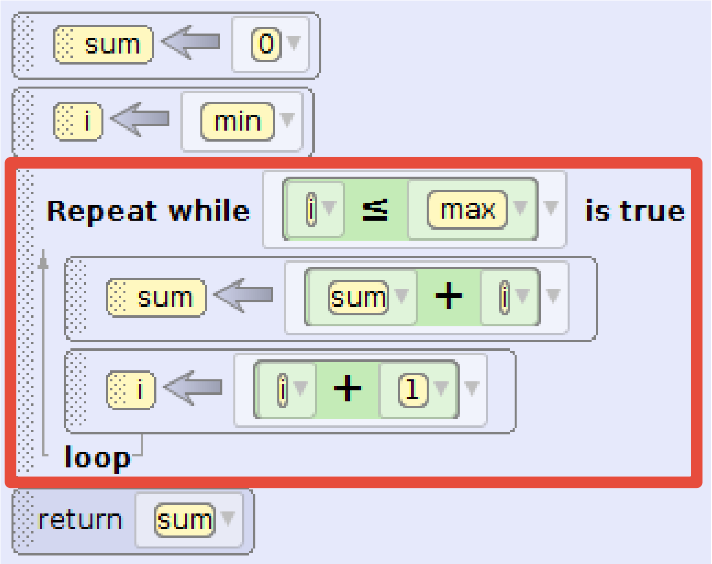
High Beneficial Cognitive Load

Detrimental Cognitive Load

Programming Syntax

```
public static int sum( int min, int max ) {
    assert max > min;

    int sum = 0;
    for( int i = min; i <= max; i++ ) {
        sum = sum + i;
    }
    return sum;
}
```

Blocks-Based Programming

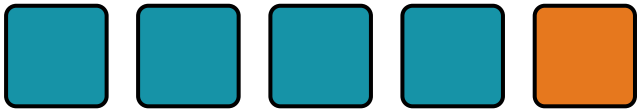


High Detrimental Cognitive Load

Low Detrimental Cognitive Load

Completion Problem

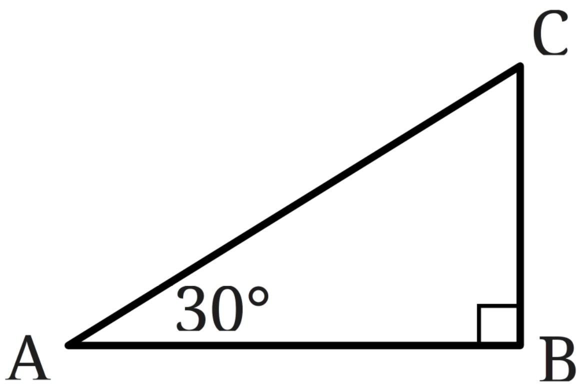
“A partial worked example where the learner has to complete some key solution steps.”



J. J. G. Van Merrienboer and M. B. M. De Croock, “Strategies for Computer-Based Programming Instruction: Program Completion Vs. Program Generation,” Journal of Educational Computing Research, vol. 8, no. 3, pp. 365–394, Jan. 1992.
Sweller, P. Ayres, and S. Kalyuga, Cognitive Load Theory. Springer, 2011.

Example Problem

What is $\angle BCA$?

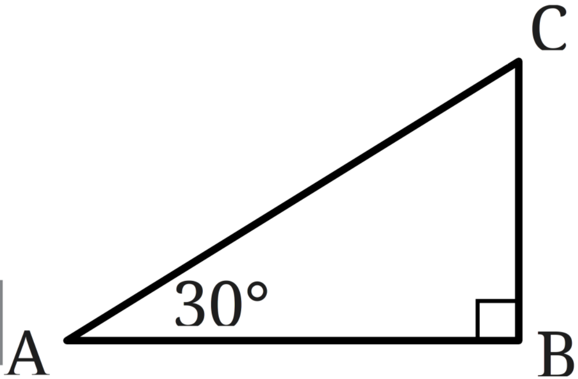


Conventional vs. Completion Problems

Conventional Problem

What is $\angle BCA$?

$$\angle BCA = \_\_\_\_$$

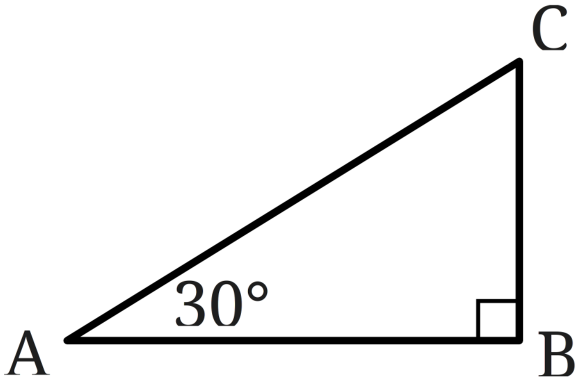


Completion Problem

What is $\angle BCA$?

$$\angle BAC + \angle CBA + \angle BCA = 180°$$

$$30° + \_\_\_\_ + \angle BCA = 180°$$



Completion Problems: Beneficial Cognitive Load

Conventional Problem

What is ∠BCA?

$\angle BCA = \underline{60°}$

Completion Problem

What is ∠BCA?

$\angle BAC + \angle CBA + \angle BCA = 180°$

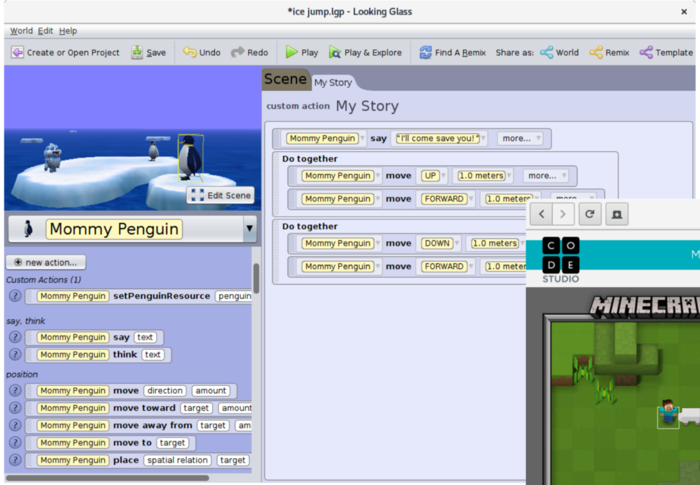
$30° + \underline{90°} + \angle BCA = 180°$

$\angle BCA = 180° - 30° - 90°$

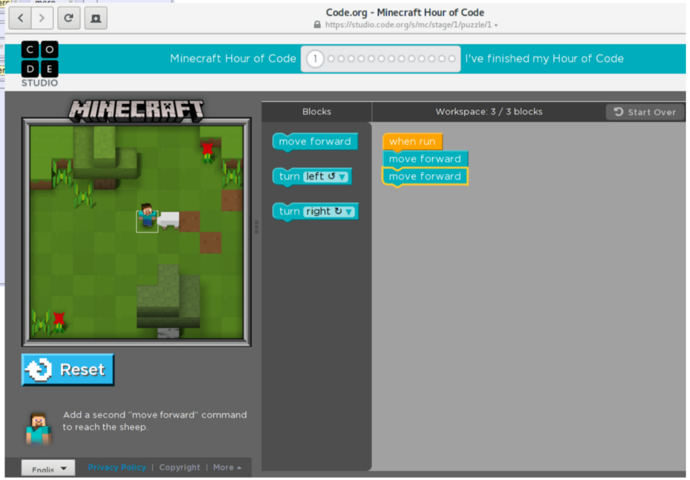
$\angle BCA = 60°$

Idea...

Motivating Programming Environment



+ Code Puzzle Activity



+ Completion Problems

What is $\angle BCA$?

$$\angle BAC + \angle CBA + \angle BCA = 180°$$

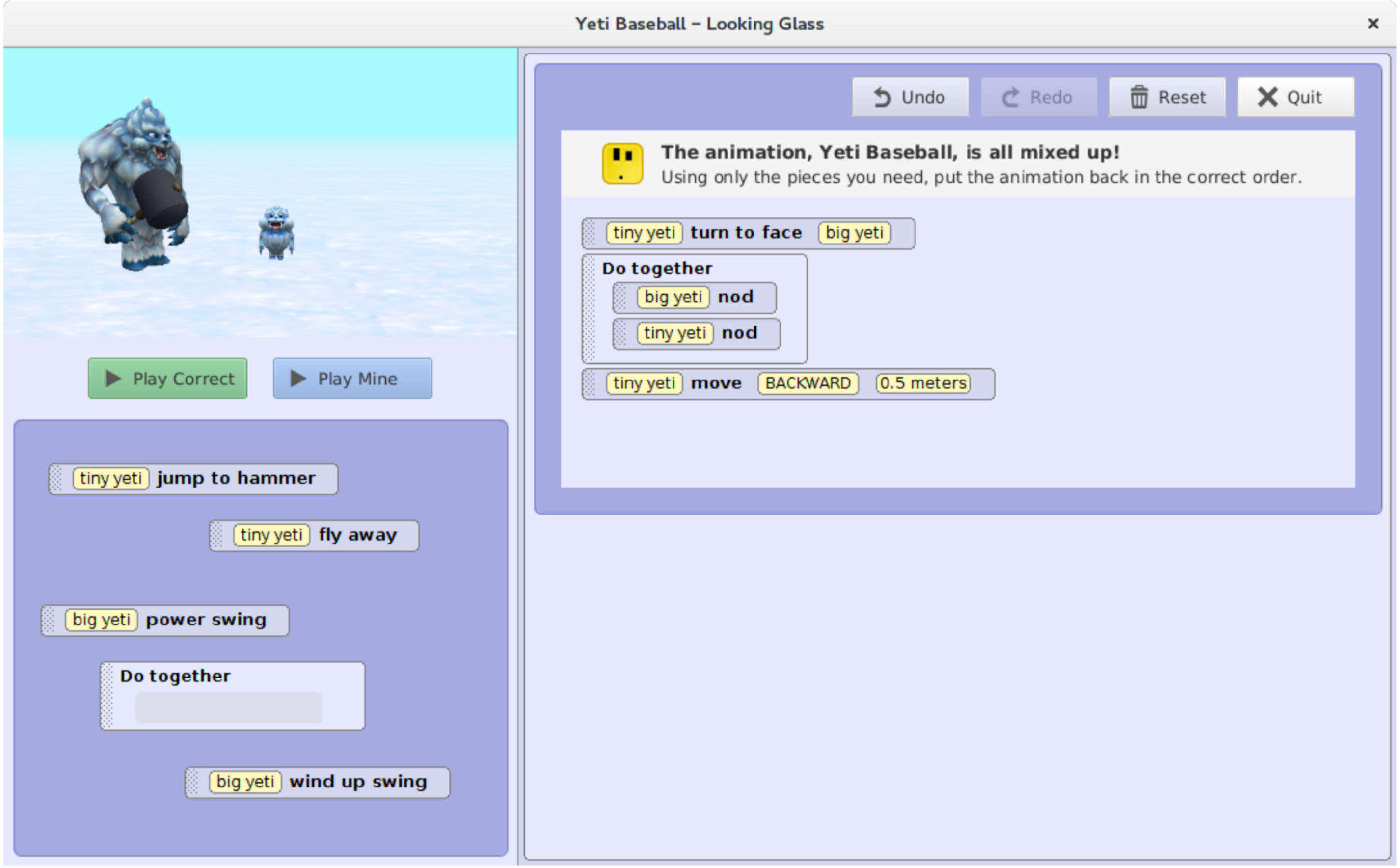
$$30° + \underline{90°} + \angle BCA = 180°$$

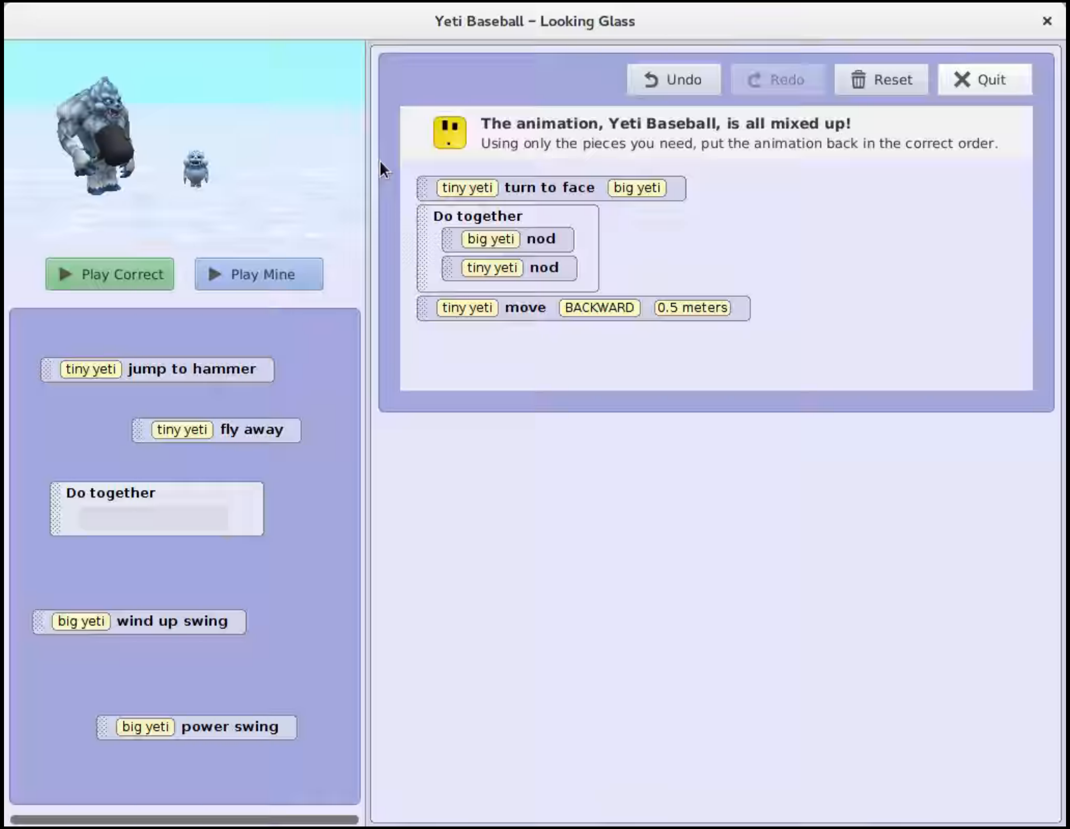
$$\angle BCA = 180° - 30° - 90°$$

$$\angle BCA = 60°$$

How might we combine a motivating programming environment, code puzzles, and completion problems?

Code Puzzle Completion Problems




Yeti Baseball – Looking Glass
Play Correct
Play Mine
tiny yeti jump to hammer
tiny yeti fly away
Do together
big yeti wind up swing
big yeti power swing
Undo
Redo
Reset
Quit
The animation, Yeti Baseball, is all mixed up!
Using only the pieces you need, put the animation back in the correct order.
tiny yeti turn to face big yeti
Do together
big yeti nod
tiny yeti nod
tiny yeti move BACKWARD 0.5 meters

Do code puzzle completion problems support children in using programming constructs in near transfer situations independently?

Hypotheses

Hypothesis I Middle school children will demonstrate greater learning efficiency of programming constructs by finishing training materials in less time and successfully completing more near transfer tasks, when training independently using code puzzle completion problems compared to current best practice, tutorials.

Hypothesis II Errors in code puzzles, commonly called distractors, will decrease learning efficiency by increasing middle school children's time to complete code puzzle completion problems while also decreasing their ability to successfully utilize programming constructs in near transfer situations.

Hypothesis III When given the freedom to choose their own informal learning resources, middle school children will perceive value in using code puzzle completion problems by expressing more interest, enjoyment, and preference towards using them compared to tutorials.

Formative Evaluations: Developing Code Puzzle Completion Problems & an Introductory Curriculum

K. J. Harms, N. Rowlett, and C. Kelleher. 2015. Enabling independent learning of programming concepts through programming completion puzzles. In 2015 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), 271–279. https://doi.org/10.1109/VLHCC.2015.7357226

Formative Evaluations

I. Completion Problem → Puzzle Interface & Mechanics

10 iterations

23 participants - St. Louis Science Center

30 minutes

II. Introductory Curriculum

8 iterations

21 participants - St. Louis Academy of Science

90 minutes

Things we tried:

Scrambling the code.

Fun & unpredictable animations.

Feedback only after solving the puzzle.

Do other activities while watching program execution.

Overlapping puzzle statements.

Demonstrating construct behavior/example-based puzzles.

Detailed code and animations.

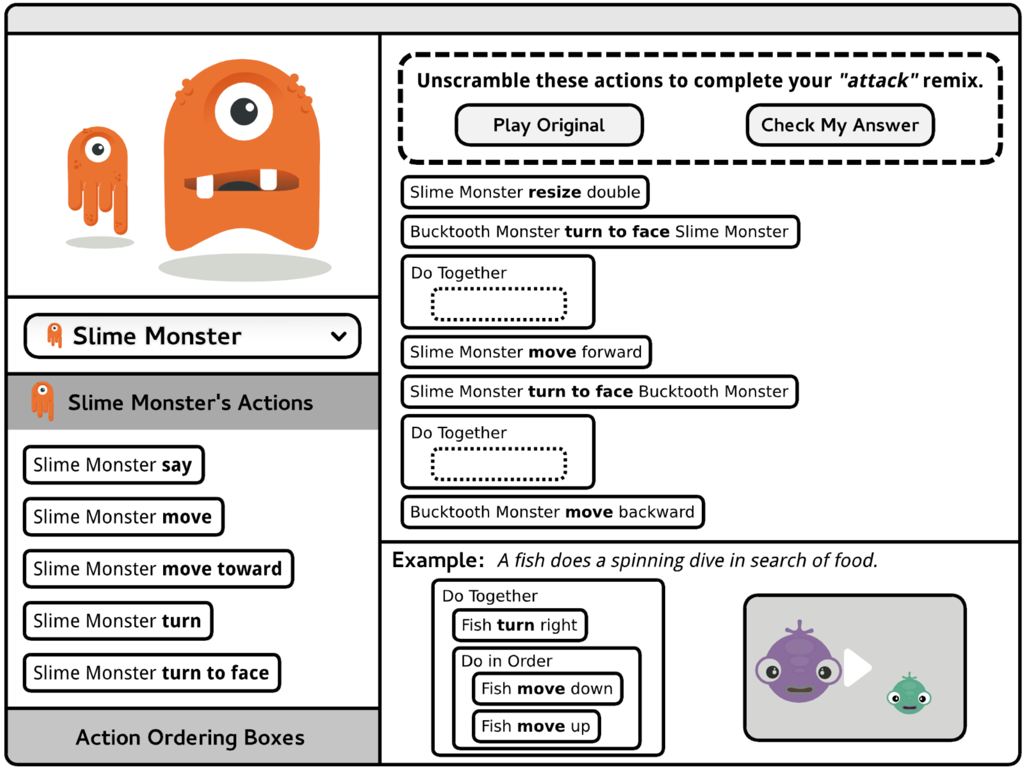
Scrambling the code.

What is a Programming Completion Problem?

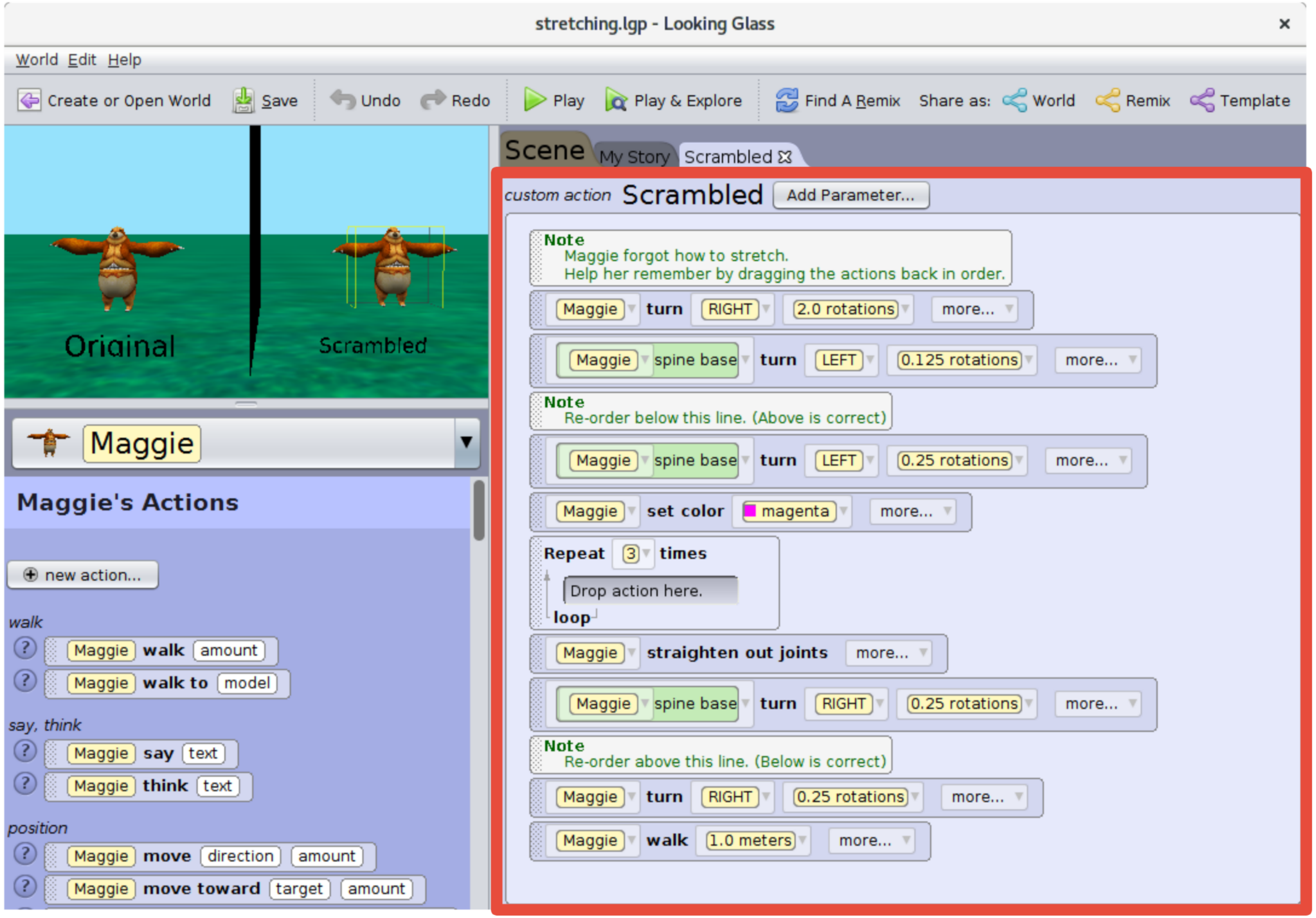
Give the first few lines?

Leave blanks throughout the code?

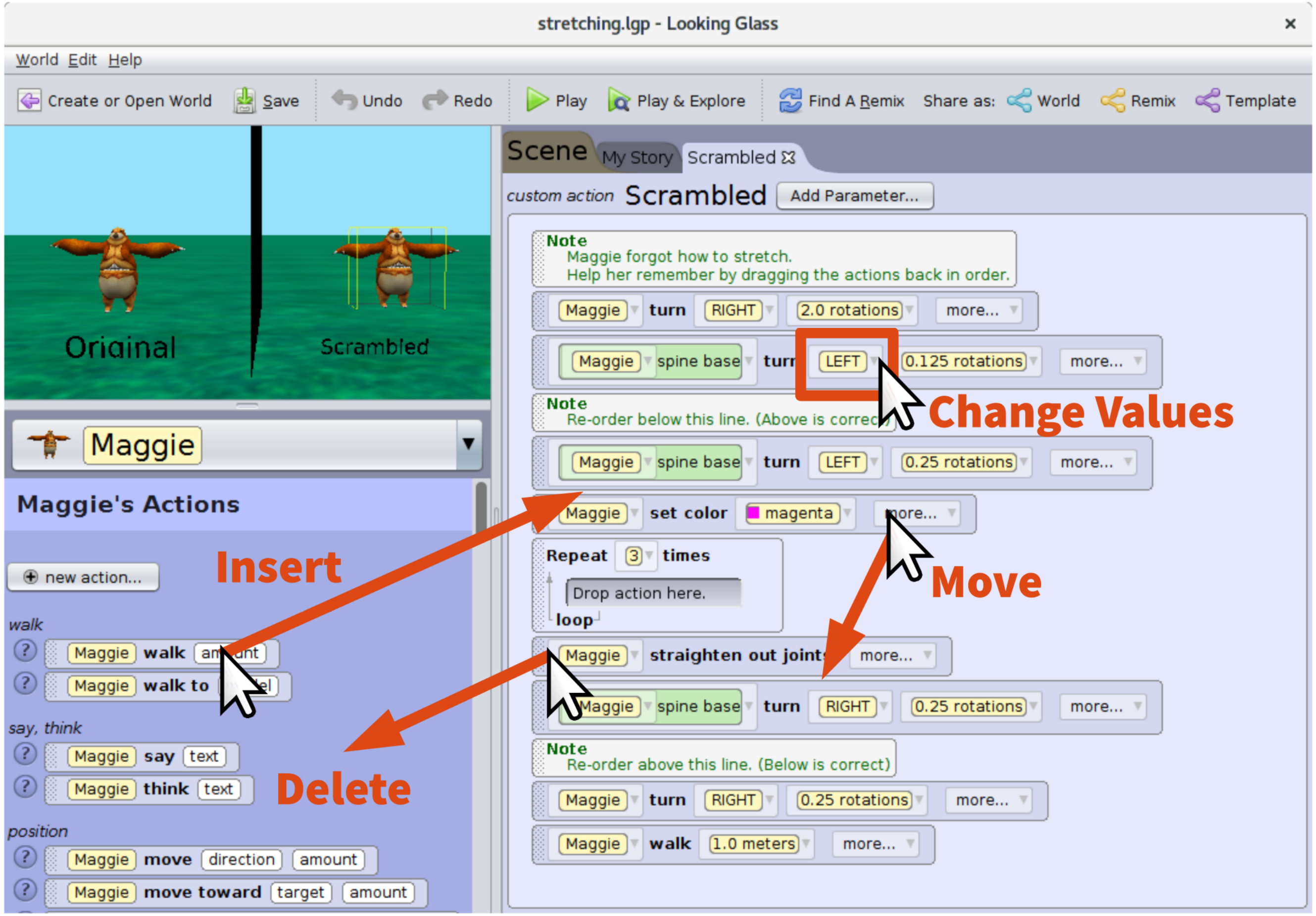
Scramble the order of the statements?



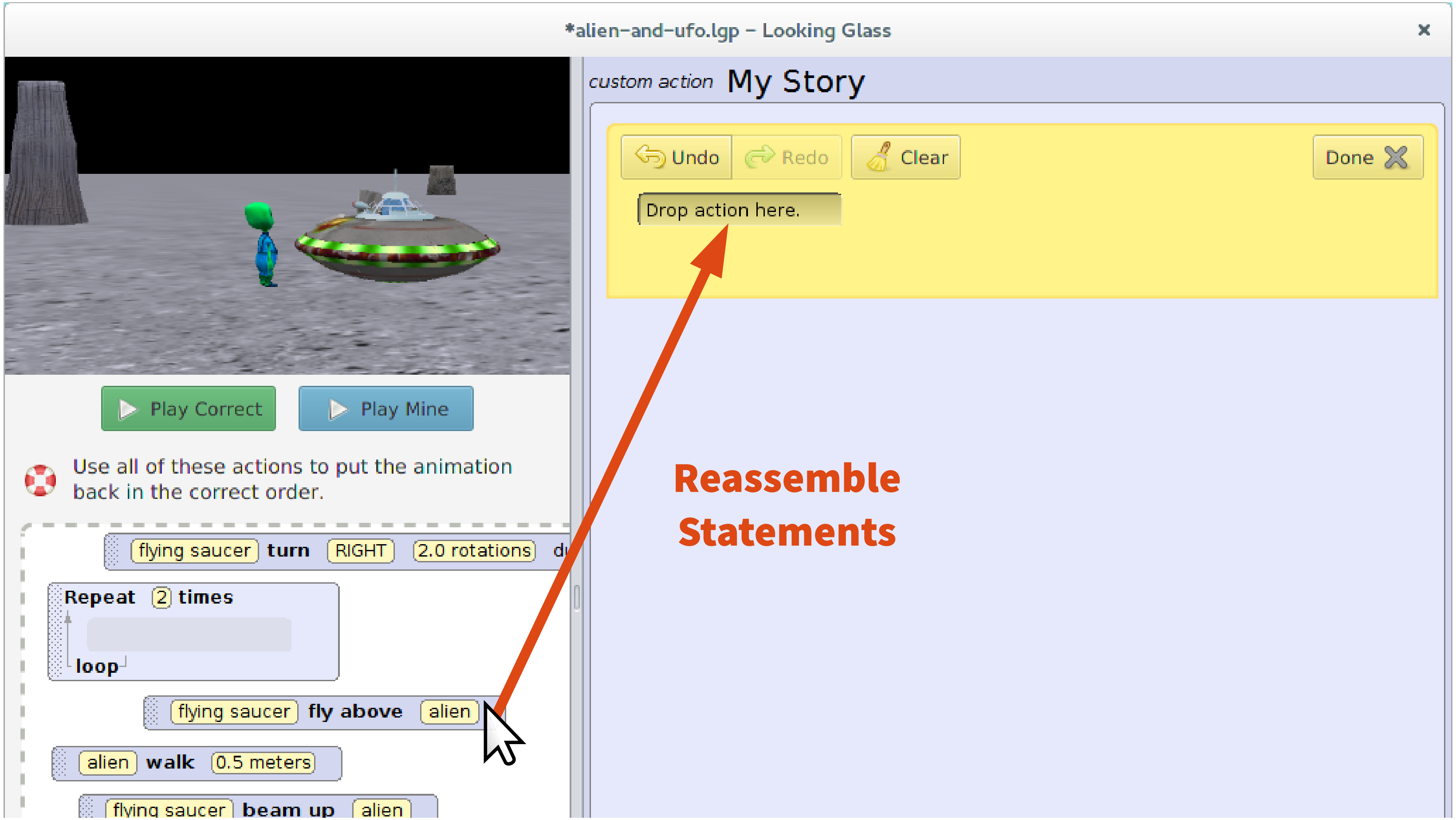
Code Scramble



Code Scramble: Many Editable Dimensions



Limit the Editable Dimensions

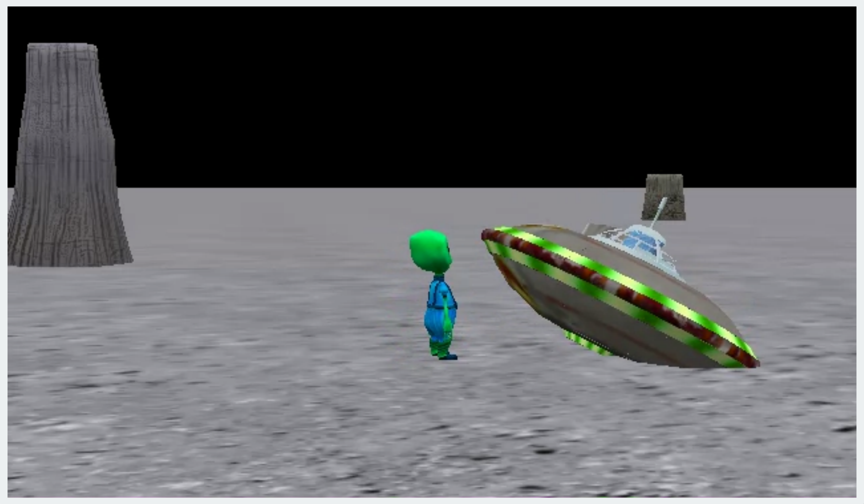


Fun & unpredictable animations.

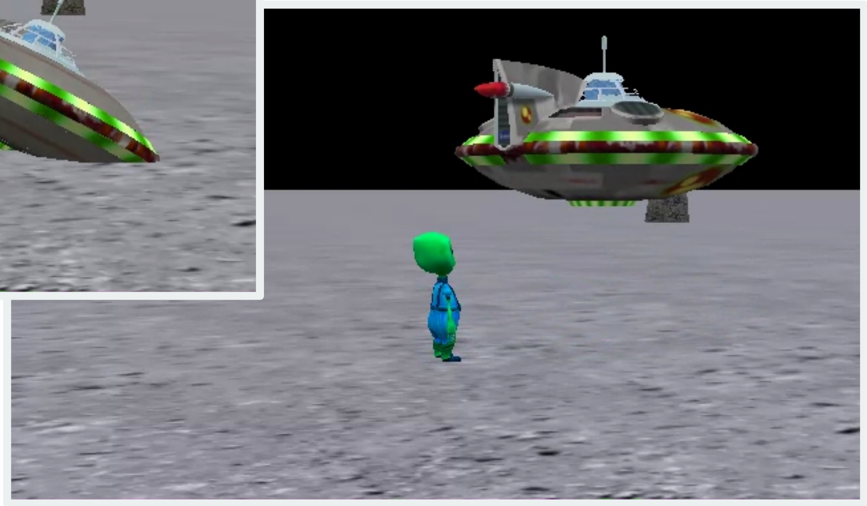
Fun & Unpredictable: Hard to Remember

```
Brown Monkey turn to face Grey Monkey
Grey Monkey turn to face Brown Monkey
Repeat 2 times
    Brown Monkey talk
    Grey Monkey talk
loop
Brown Monkey move UP 0.5 meters duration 0.5
Brown Monkey move DOWN 0.5 meters duration
Repeat 3 times
    Grey Monkey say "EEK EEK"
    Brown Monkey turn RIGHT 4.0 rotations
    Grey Monkey hop side to side
loop
Grey Monkey lay down
```

Memorable Segments

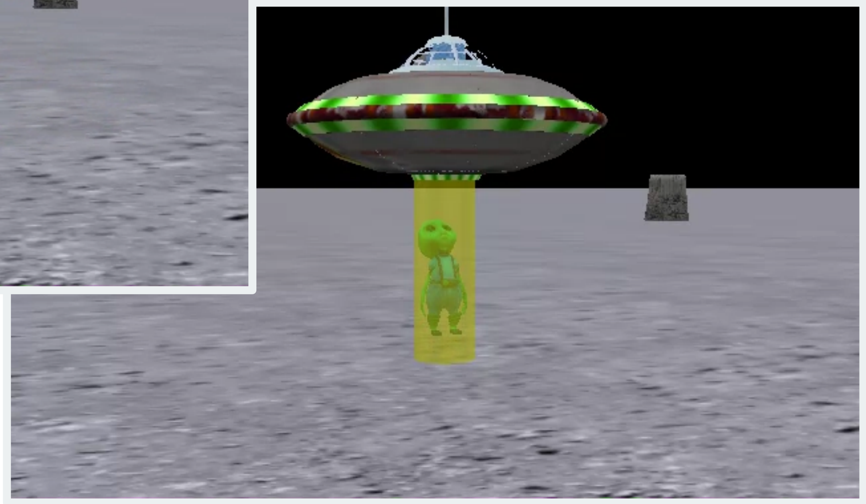


Segment I – alien repairs flying saucer



Segment II – flying saucer starts up

Segment III - alien leaves in flying saucer



Memorable Segments

Segment I

```
alien walk 0.5 meters
Repeat 2 times
    alien kick
    flying saucer shake
loop
```

Segment II

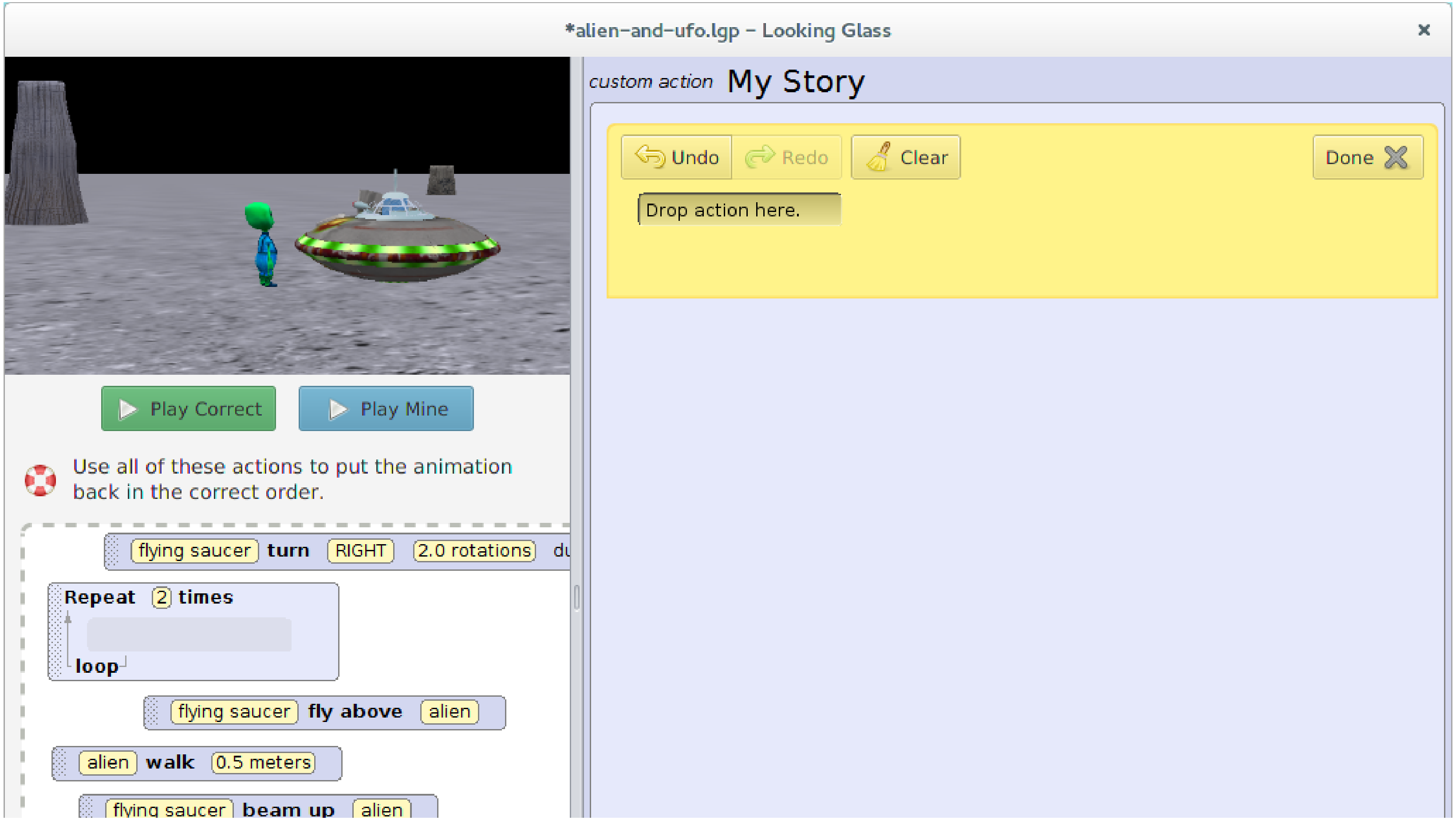
```
Do together
    flying saucer fly above alien
    flying saucer turn RIGHT 2.0 rotations duration 2.0 seconds
```

Segment III

```
flying saucer beam up alien
flying saucer fly away
```

Motivating Programming Environment + Code Puzzle Activity + Completion Problems

Code Puzzle Completion Problems

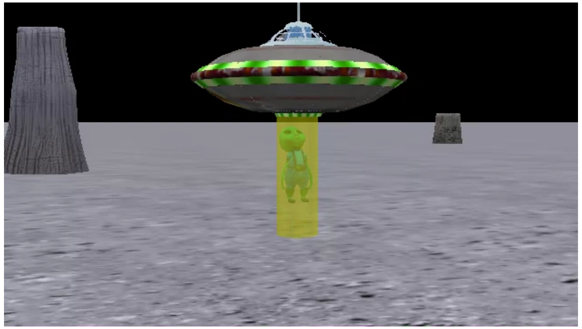


Introductory Curriculum

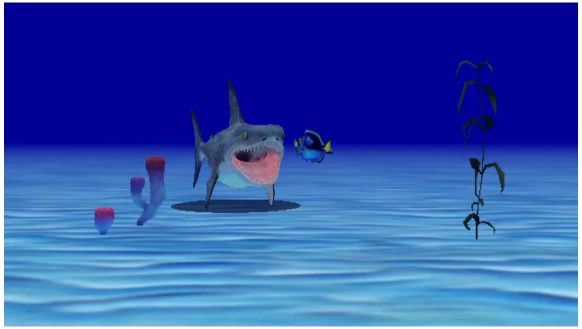
1. Sequential

2. Repeated

3. Parallel



4. Repeated & Parallel



5. Parallel { Repeated }

6. Repeated { Parallel }

Hypotheses

Hypothesis I Middle school children will demonstrate greater learning efficiency of programming constructs by finishing training materials in less time and successfully completing more near transfer tasks, when training independently using code puzzle completion problems compared to current best practice, tutorials.

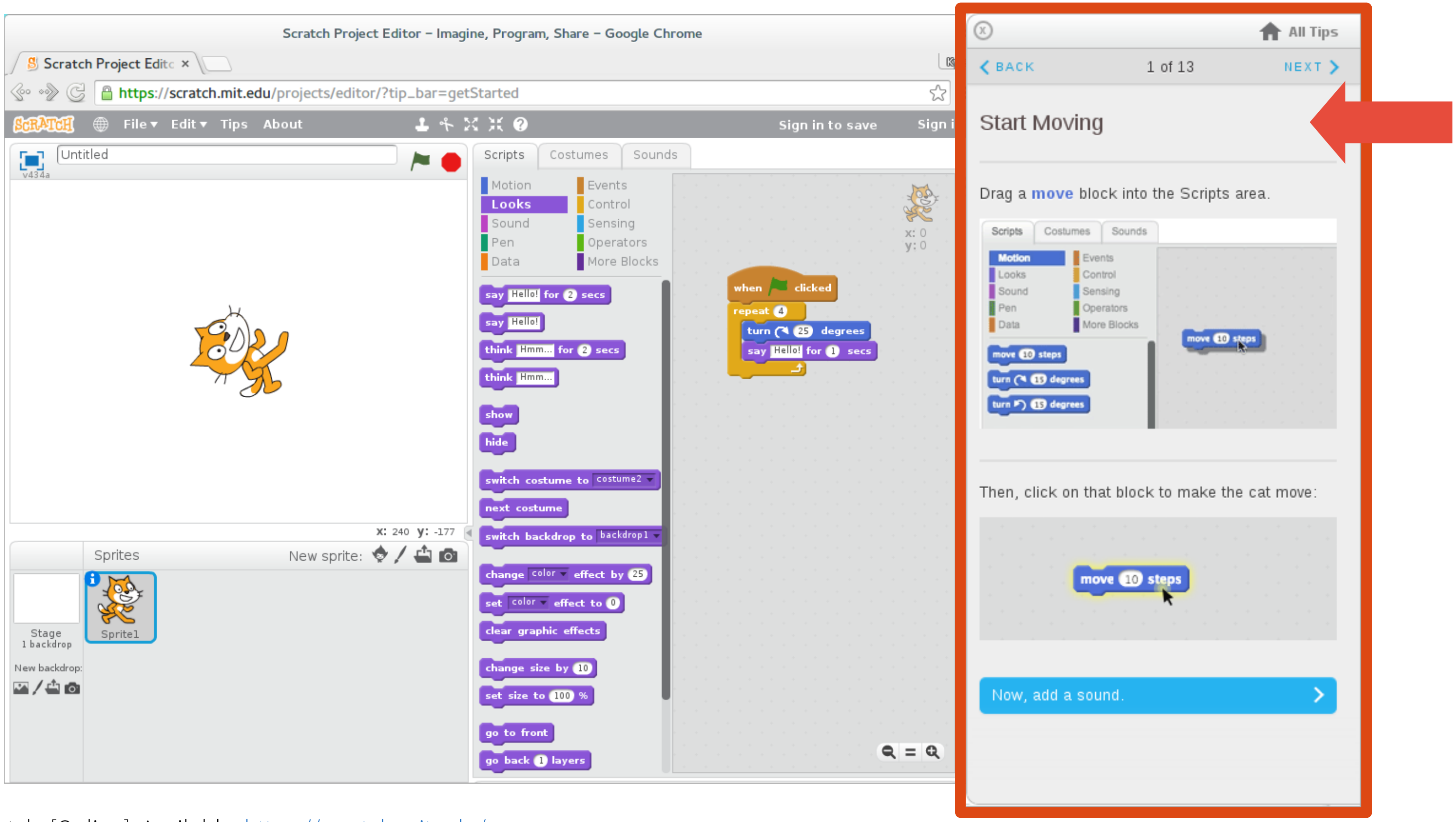
Hypothesis II Errors in code puzzles, commonly called distractors, will decrease learning efficiency by increasing middle school children's time to complete code puzzle completion problems while also decreasing their ability to successfully utilize programming constructs in near transfer situations.

Hypothesis III When given the freedom to choose their own informal learning resources, middle school children will perceive value in using code puzzle completion problems by expressing more interest, enjoyment, and preference towards using them compared to tutorials.

Summative Evaluation:
Hypothesis I – Are code puzzle completion problems effective?

K. J. Harms, N. Rowlett, and C. Kelleher. 2015. Enabling independent learning of programming concepts through programming completion puzzles. In 2015 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), 271–279. https://doi.org/10.1109/VLHCC.2015.7357226

Literary Search: Tutorials

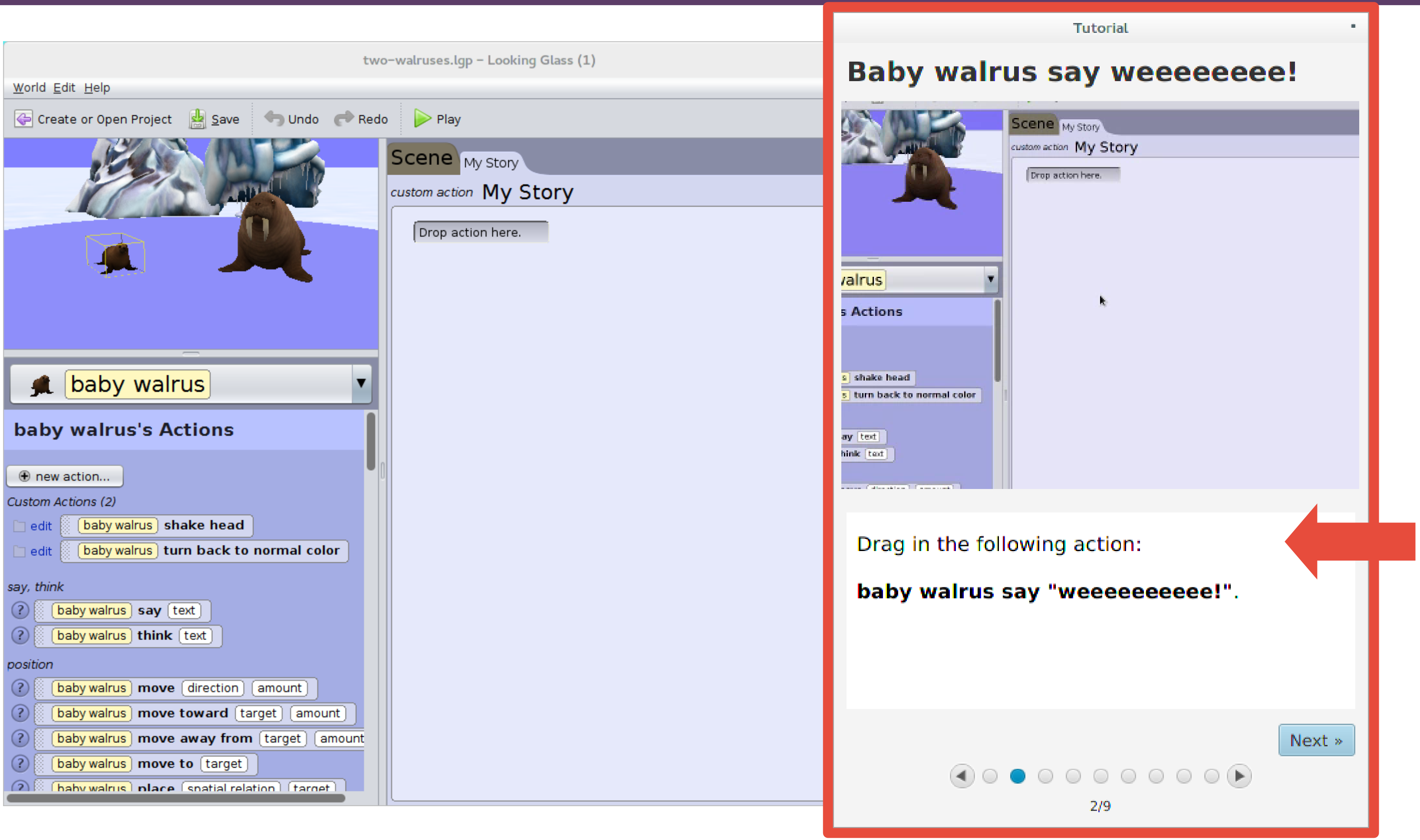


"Scratch," Scratch. [Online]. Available: https://scratch.mit.edu/.
K. J. Harms, D. Cosgrove, S. Gray, and C. Kelleher, "Automatically Generating Tutorials to Enable Middle School Children to Learn Programming Independently," 2013.

Hypothesis I

Middle school children will demonstrate greater learning efficiency of programming constructs by finishing training materials in less time and successfully completing more near transfer tasks, when training independently using code puzzle completion problems compared to current best practice, tutorials.

Tutorials


two-walruses.lgp – Looking Glass (1)
World Edit Help
Create or Open Project
Save
Undo
Redo
Play
baby walrus
baby walrus's Actions
new action...
Custom Actions (2)
edit
baby walrus shake head
edit
baby walrus turn back to normal color
say, think
baby walrus say text
baby walrus think text
position
baby walrus move direction amount
baby walrus move toward target amount
baby walrus move away from target amount
baby walrus move to target
Scene
My Story
custom action My Story
Drop action here.
Tutorial
Baby walrus say weeeeeeee!
Drag in the following action:
baby walrus say "weeeeeeeee!".
Next »
2/9

Summative Evaluation

Between-Subjects Study

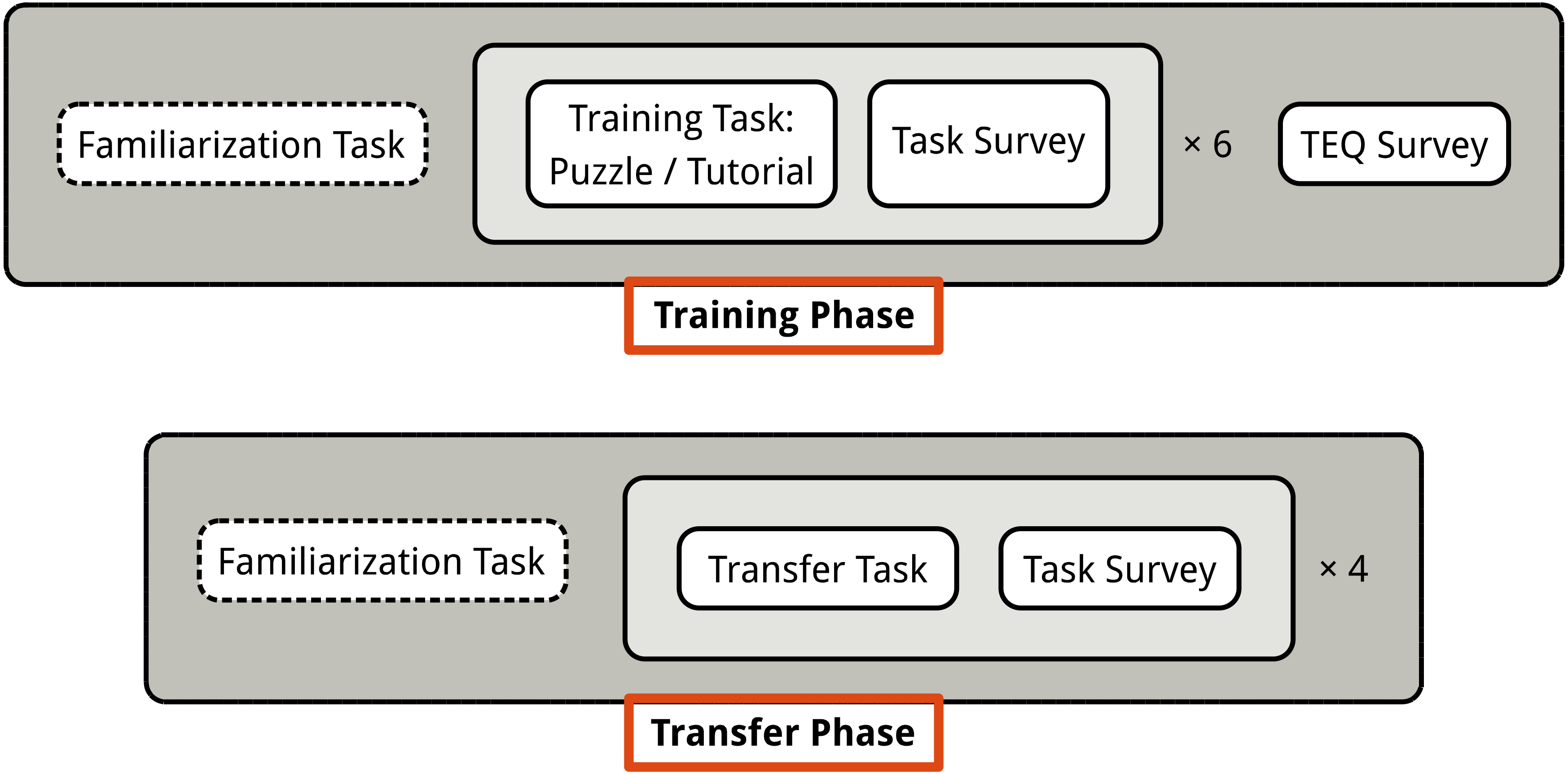
- Control: Tutorials
- Experimental: Puzzles

27 participants (10 to 15 years)

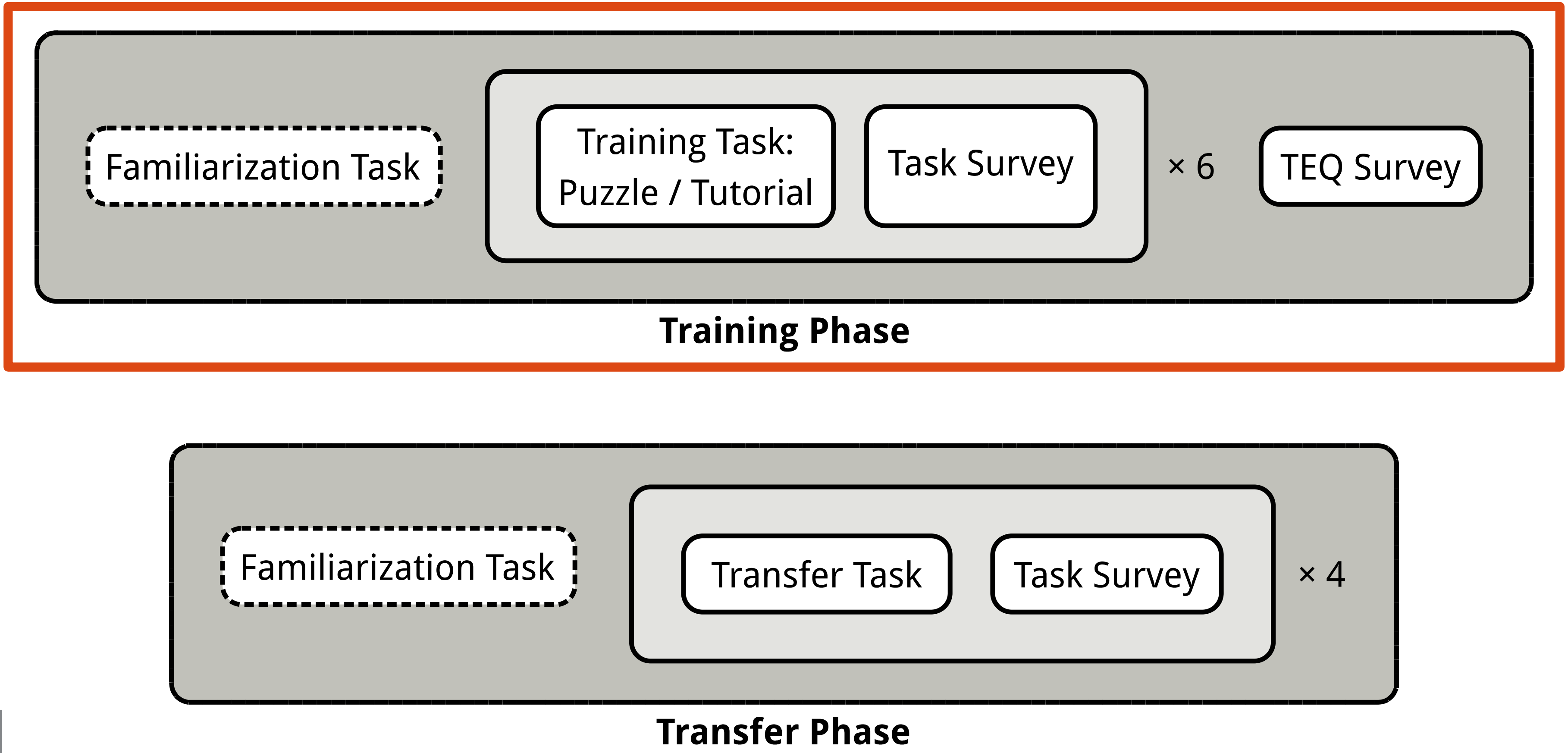
- 12 Female, 15 Male
- Average Age: 11.59
- Minimal Programming Experience (< 3 hours)

2 hours

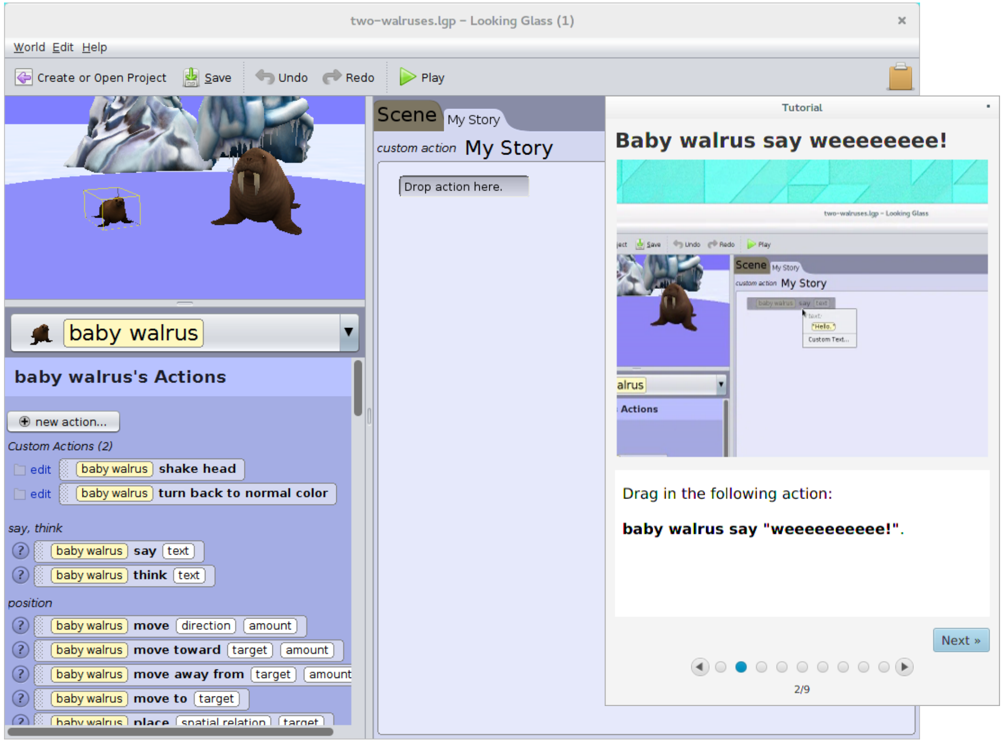
Study Design



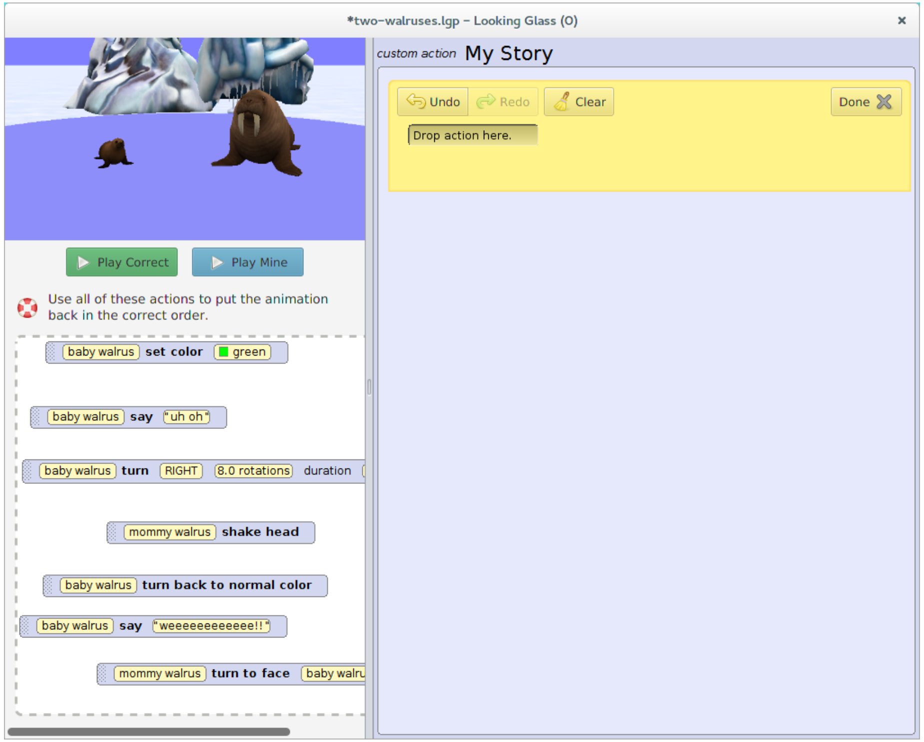
Study Design



Training Tasks × 6

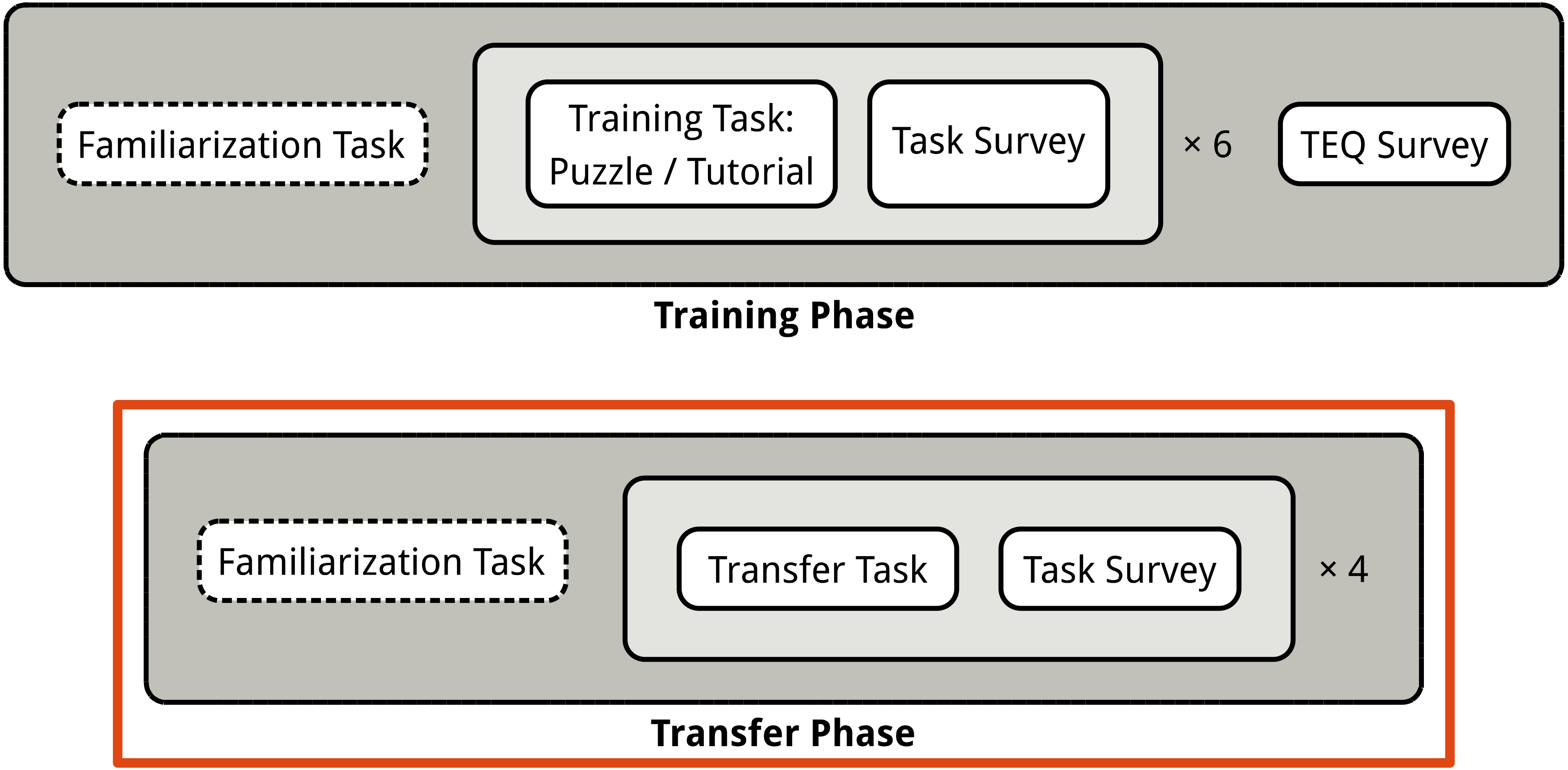


Control: Tutorials

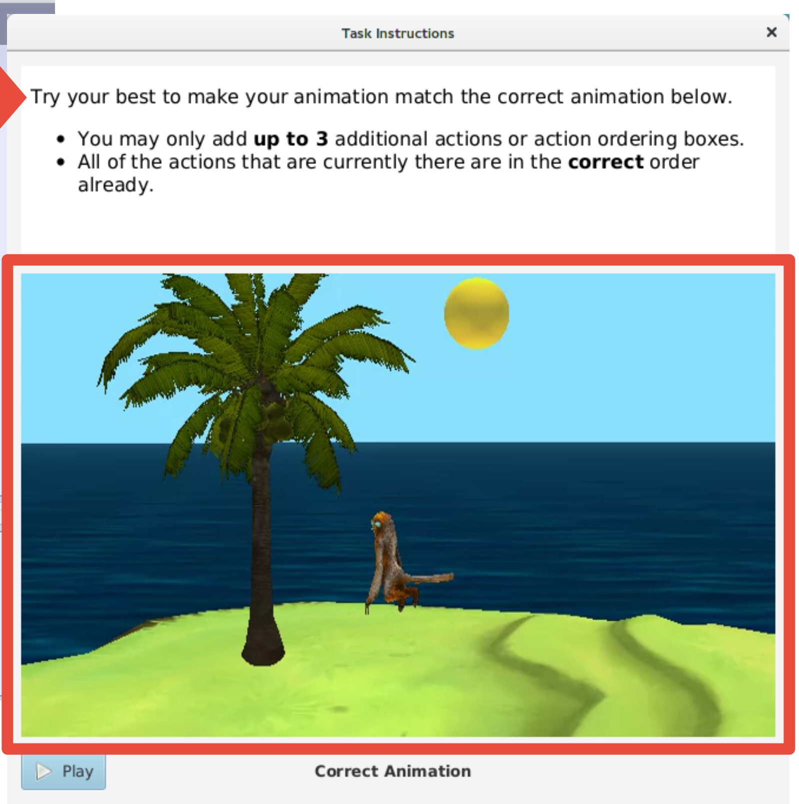


Experimental: Puzzles

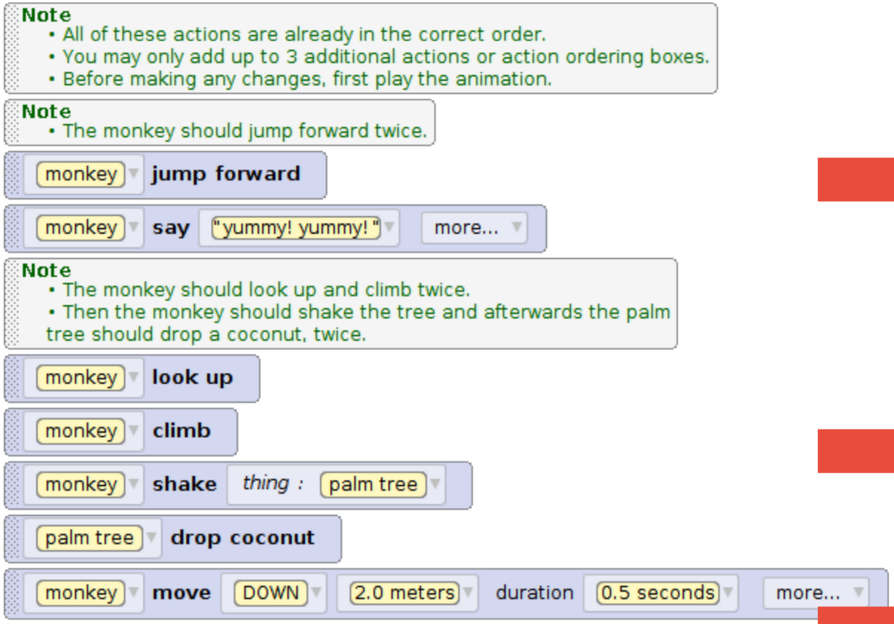
Study Design



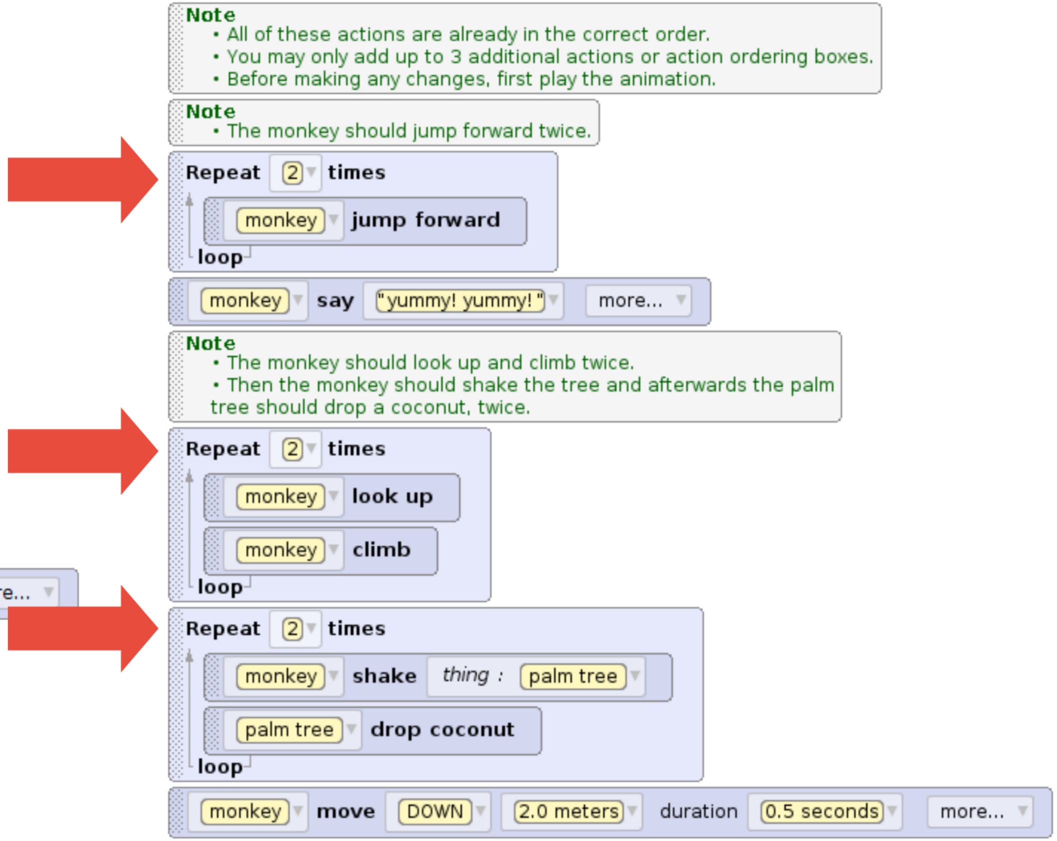
Transfer Tasks × 4



Completed Transfer Task



Initial State



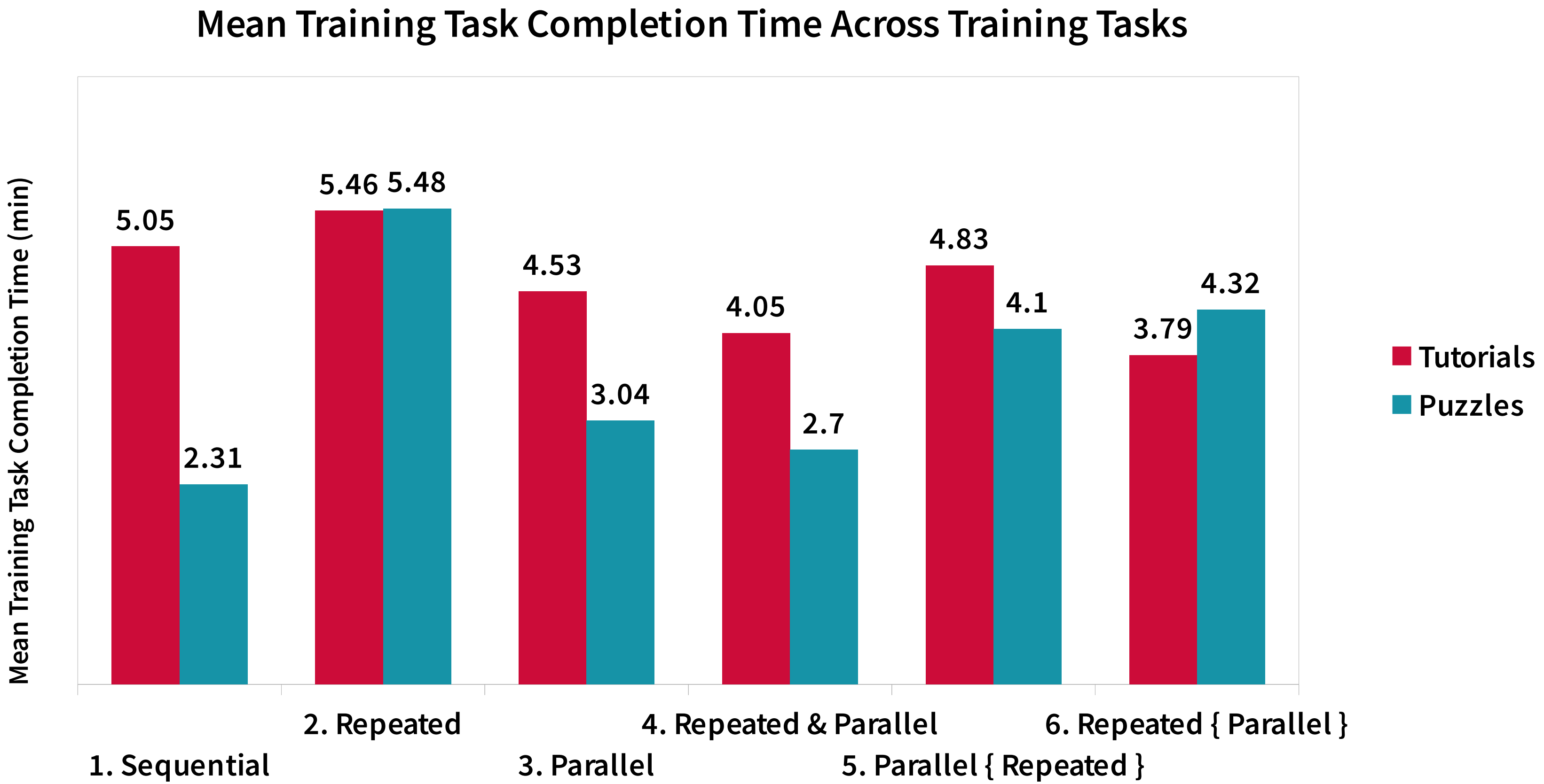
Completed Task

Hypothesis I

Middle school children will **demonstrate greater learning efficiency** of programming constructs by finishing training materials in less time and **successfully completing more near transfer tasks**, when training independently using code puzzle completion problems compared to current best practice, tutorials.

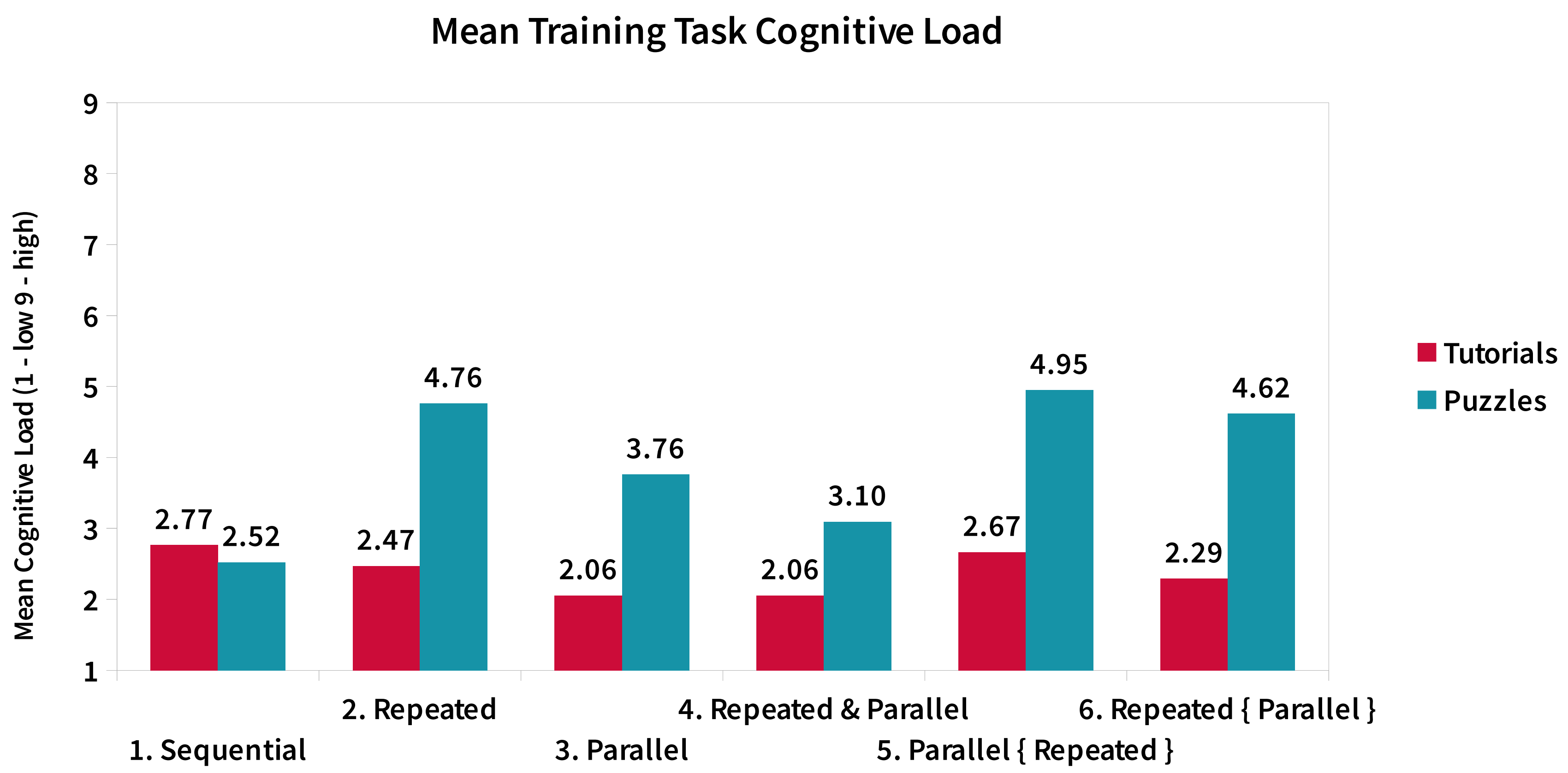
Training Efficiency

Tutorials Take Longer To Complete Than Puzzles



$V = .75$, $F(6, 29) = 14.12$, $p < .001$, $\eta^2 = .75$

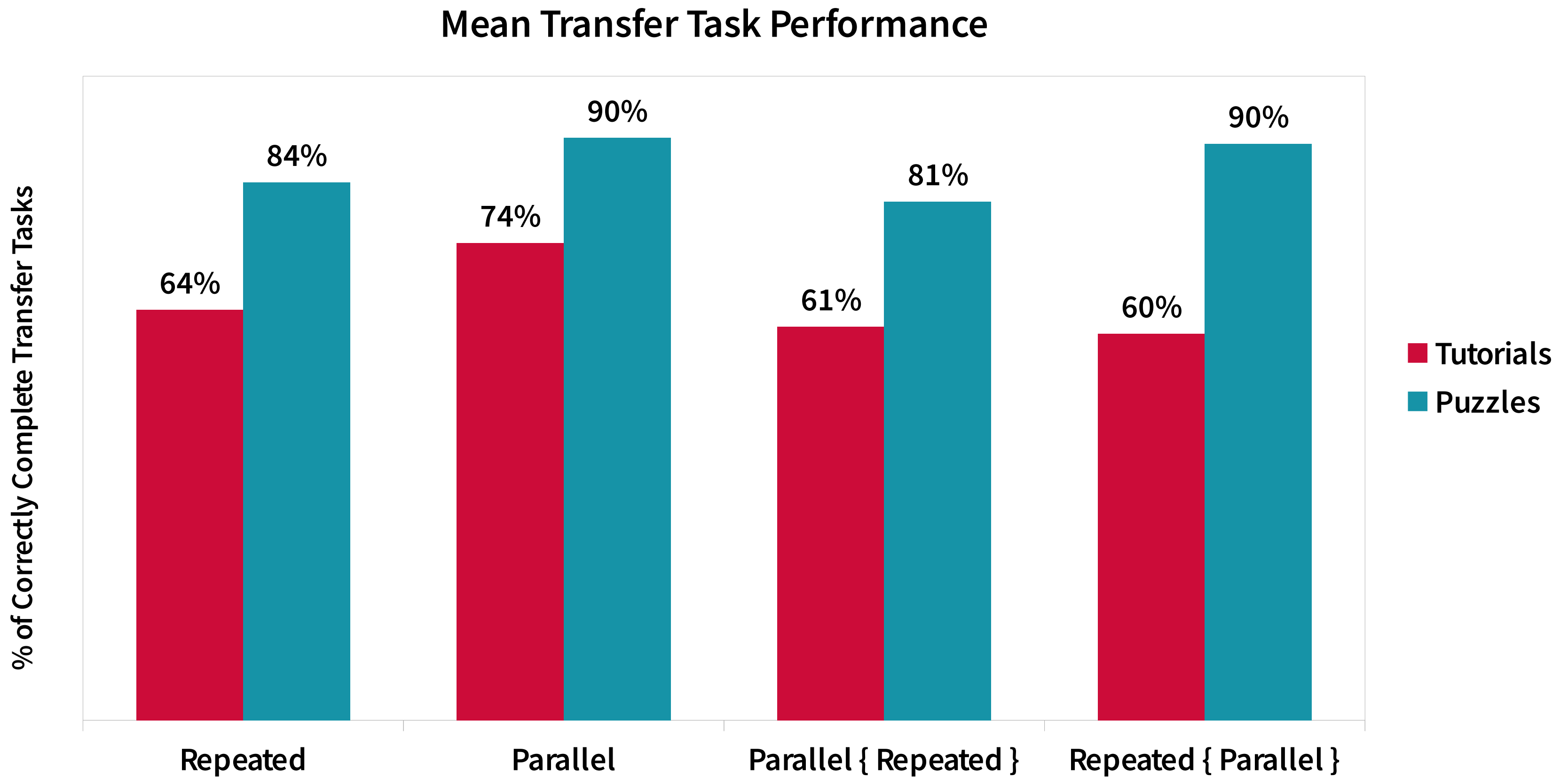
Puzzles Impose More Cognitive Load Than Tutorials



$V = .51$, $F(6, 29) = 4.45$, $p = .003$, $\eta^2 = .51$

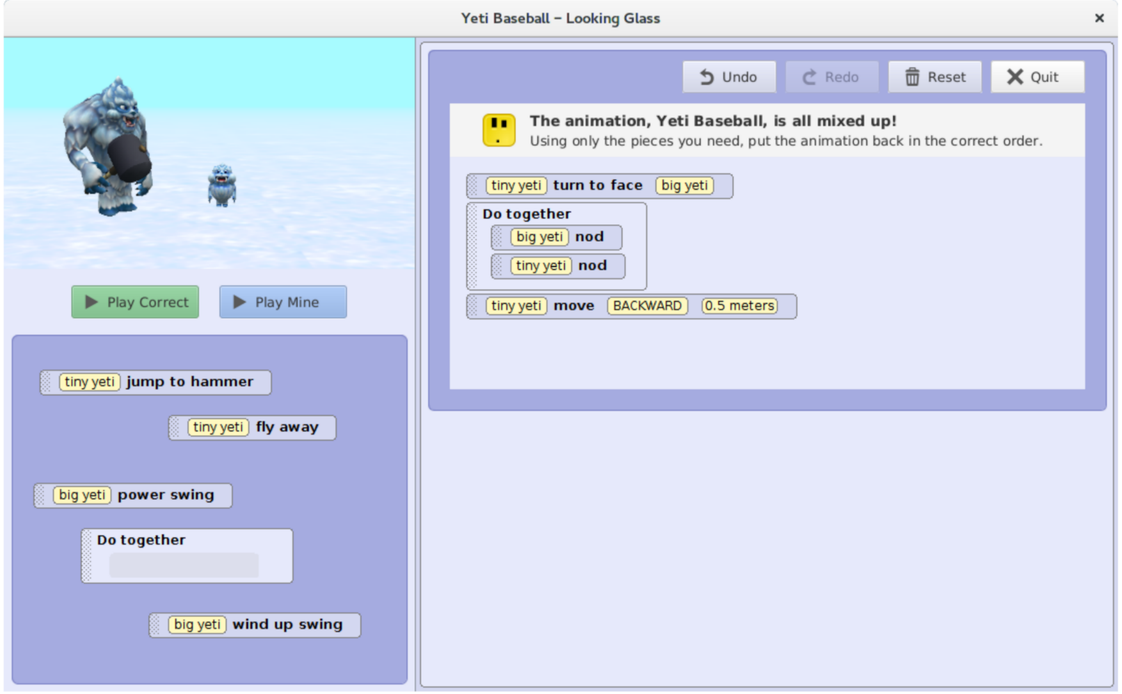
Transfer Task Performance

Puzzle Users Performed Better on Near Transfer Tasks



$V = .27$, $F(4, 31) = 2.80$, $p = .043$, $\eta^2 = .27$

Hypothesis I: Accept



Code puzzle completion problem participants performed 33% better on transfer tasks while requiring 21% less training time compared to tutorial participants.

Summative Evaluation:
Hypothesis II – Do errors in code puzzles decrease learning?

Kyle James Harms, Jason Chen, and Caitlin L. Kelleher. 2016. Distractors in Parsons Problems Decrease Learning Efficiency for Young Novice Programmers. In Proceedings of the 2016 ACM Conference on International Computing Education Research (ICER ’16), 241–250. https://doi.org/10.1145/2960310.2960314

Parsons Problems & Distractors

csp-8-3-4: The following is the correct code for printing the even numbers from 0 to 10, **but it also includes some extra code that you won't need**. Drag the needed blocks from the left and put them in the correct order on the right. Don't forget to indent blocks in the body of the loop. Just drag the block to the further right to indent. Click the *Check Me* button to check your solution.

Drag from here

```
counter = counter + 1
```

Drop blocks here

```
counter = 0
while counter <= 10:
    print(counter)
    counter = counter + 2
```

Check Me Reset

Perfect!

Parsons, D. and Haden, P. 2006. Parson's Programming Puzzles: A Fun and Effective Learning Tool for First Programming Courses.
Ericson, B.J. et al. 2015. Analysis of Interactive Features Designed to Enhance Learning in an Ebook.

Errors and Learning

Learning with Errors: Teach Common Misconceptions

Print TRUE if string1 and string2 contain the same value.

```java
String string1 = new String("abc");
String string2 = new String("abc");

if (string1 == string2) {
  System.out.println("TRUE");
} else {
  System.out.println("FALSE");
}
```

```java
String string1 = new String("abc");
String string2 = new String("abc");

if (string1.equals(string2)) {
  System.out.println("TRUE");
} else {
  System.out.println("FALSE");
}
```


Muller, D. a. et al. 2008. Saying the wrong thing: improving learning with multimedia by including misconceptions.
Große, C.S. and Renkl, A. 2007. Finding and fixing errors in worked examples: Can this foster learning outcomes?
Wang, M. et al. 2015. Using Feedback to Improve Learning: Differentiating between Correct and Erroneous Examples.

Distractors in Multiple Choice Tests

1. Consider the following code segment.

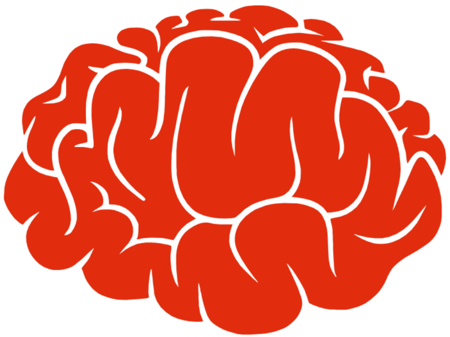
```
String string1 = new String("abc");
String string2 = new String("abc");

if (string1 == string2) {
  System.out.println("TRUE");
} else {
  System.out.println("FALSE");
}
```

What is printed as a result of executing the code segment?

(a) TRUE   (b) FALSE   (c) TRUE
FALSE

Richland, L.E. et al. 2009. The pretesting effect: Do unsuccessful retrieval attempts enhance learning?
Little, J.L. and Bjork, E.L. 2012. The persisting benefits of using multiple-choice tests as learning events.
Roediger III, H.L. and Marsh, E.J. 2005. The Positive and Negative Consequences of Multiple-Choice Testing.

Testing & Information Retrieval



1. Consider the following code segment.

```java
String string1 = new String("abc");
String string2 = new String("abc");

if (string1 == string2) {
  System.out.println("TRUE");
} else {
  System.out.println("FALSE");
}
```

What is printed as a result of executing the code segment?

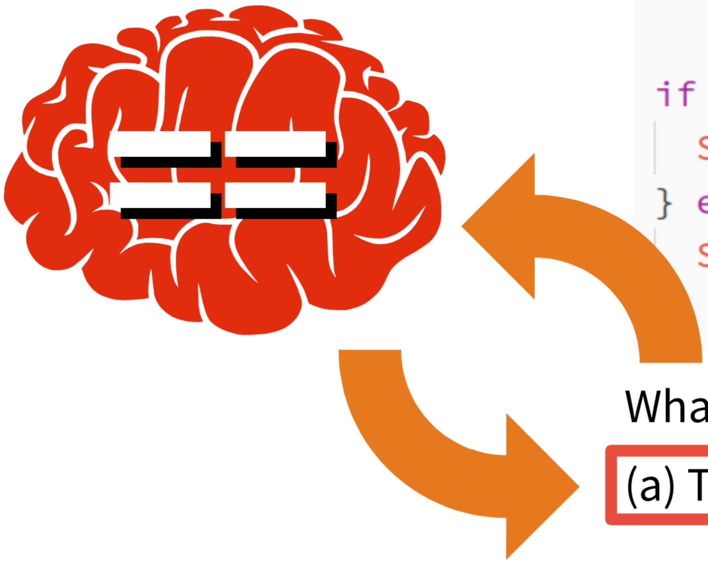
(a) TRUE

(b) FALSE

(c) TRUE
FALSE

Richland, L.E. et al. 2009. The pretesting effect: Do unsuccessful retrieval attempts enhance learning?
Little, J.L. and Bjork, E.L. 2012. The persisting benefits of using multiple-choice tests as learning events.
Roediger III, H.L. and Marsh, E.J. 2005. The Positive and Negative Consequences of Multiple-Choice Testing.

The Testing Effect



1. Consider the following code segment.

```java
String string1 = new String("abc");
String string2 = new String("abc");

if (string1 == string2) {
  System.out.println("TRUE");
} else {
  System.out.println("FALSE");
```

What is printed as a result of executing the code segment?

(a) TRUE

(b) FALSE

(c) TRUE
FALSE

Bridger, E.K. and Mecklinger, A. 2014. Errorful and errorless learning: The impact of cue–target constraint in learning from errors.
Roediger, H.L. and Karpicke, J.D. 2006. Test-Enhanced Learning Taking Memory Tests Improves Long-Term Retention.
Karpicke, J.D. 2012. Retrieval-Based Learning Active Retrieval Promotes Meaningful Learning.

A Distractor is an *Error*...

csp-8-3-4: The following is the correct code for printing the even numbers from 0 to 10, **but it also includes some extra code that you won't need**. Drag the needed blocks from the left and put them in the correct order on the right. Don't forget to indent blocks in the body of the loop. Just drag the [illegible] to the further right to indent. Click the *Check Me* button to check your solution.

Drag from here

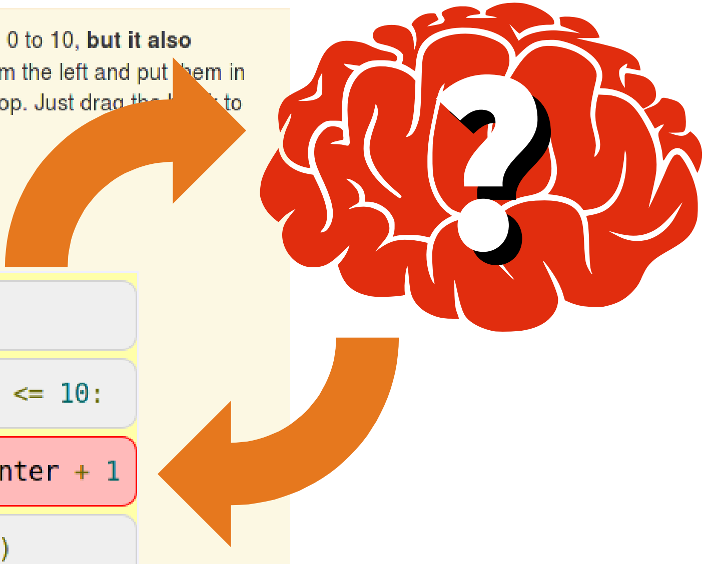
```
counter = counter + 2
```

Drop blocks here

```
counter = 0
while counter <= 10:
counter = counter + 1
print(counter)
```

Check Me | Reset

Code fragments in your program are wrong, or in wrong order. This can be fixed by moving, removing, or replacing highlighted fragments.



Hypothesis II

Errors in code puzzles, commonly called distractors, will decrease learning efficiency by increasing middle school children’s time to complete code puzzle completion problems while also decreasing their ability to successfully utilize programming constructs in near transfer situations.

Formative Evaluation: What is a Distractor in a Code Puzzle?

“Unnecessary Code”

“Extra Fragments”

“Erroneous Code”

Distractors can be used to...

- “illustrate a particular point”
- “highlight programming principles the student may ignore”

Parsons, D. and Haden, P. 2006. Parson’s Programming Puzzles: A Fun and Effective Learning Tool for First Programming Courses.

Summative Evaluation

Between-Subjects Study

Control: No Distractors

Experimental: Distractors

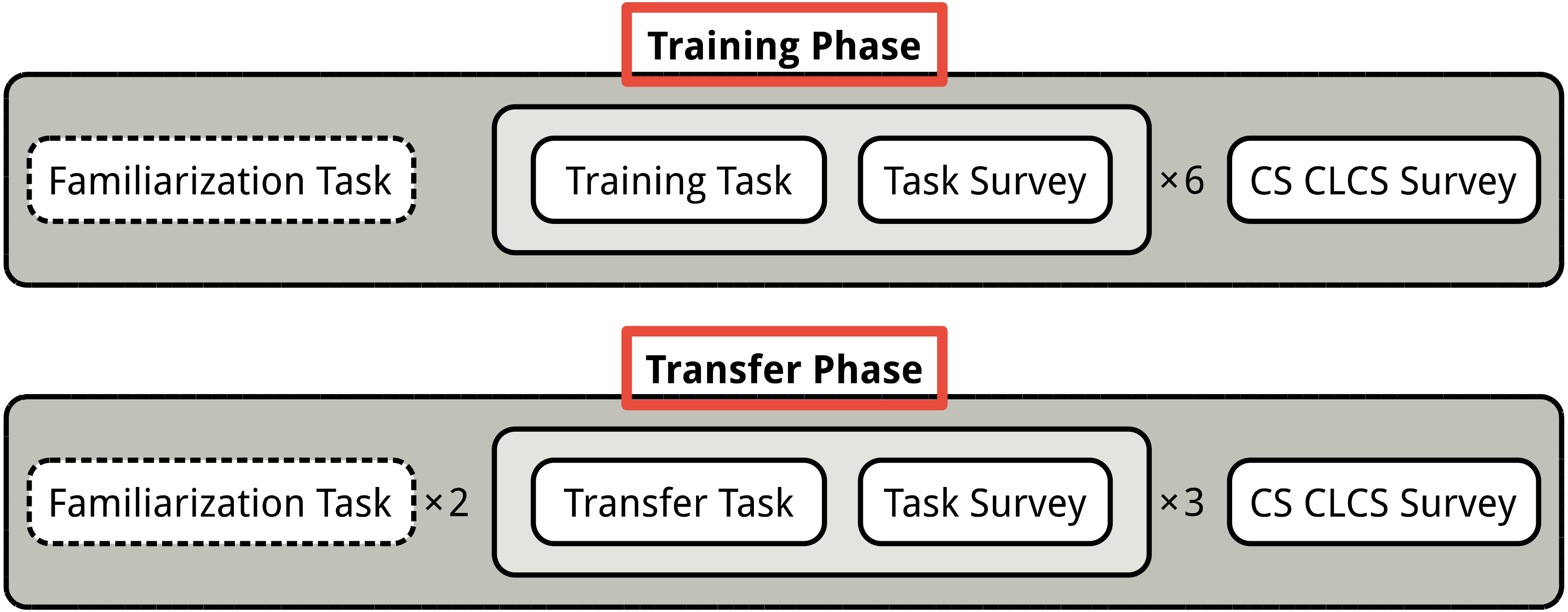
92 participants (10 to 15 years)

32 female, 60 male

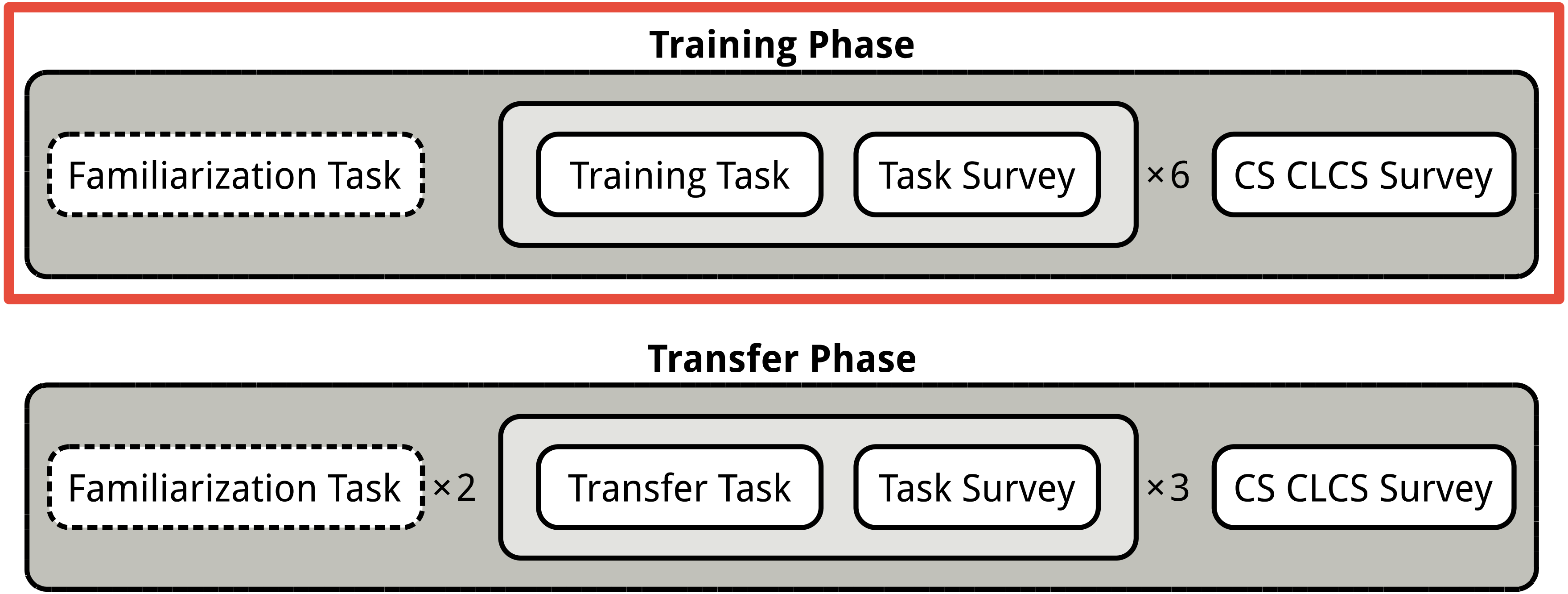
Average age: 12.9 years

2 hours

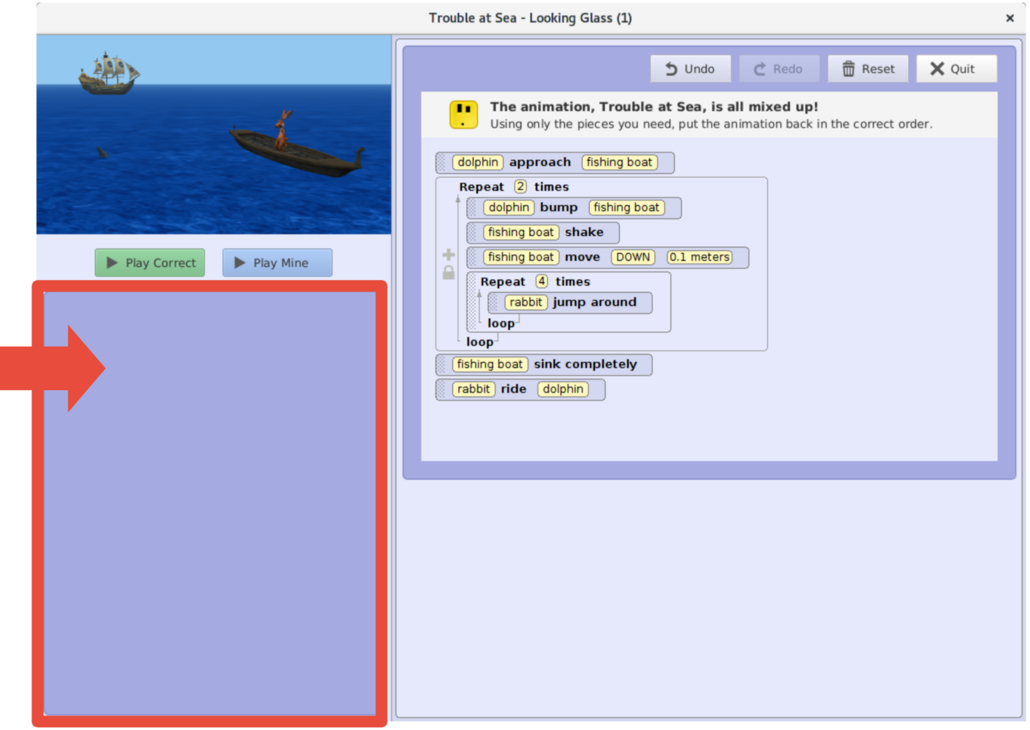
Study Design: Training Phase



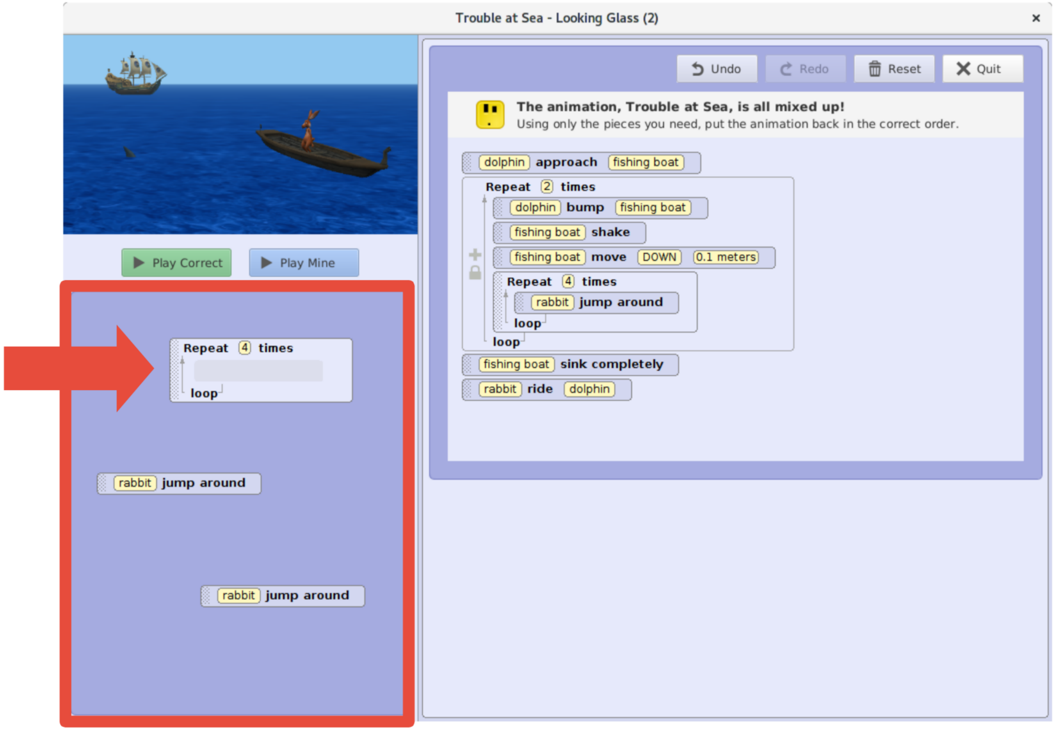
Study Design: Training Phase



Training Tasks × 6

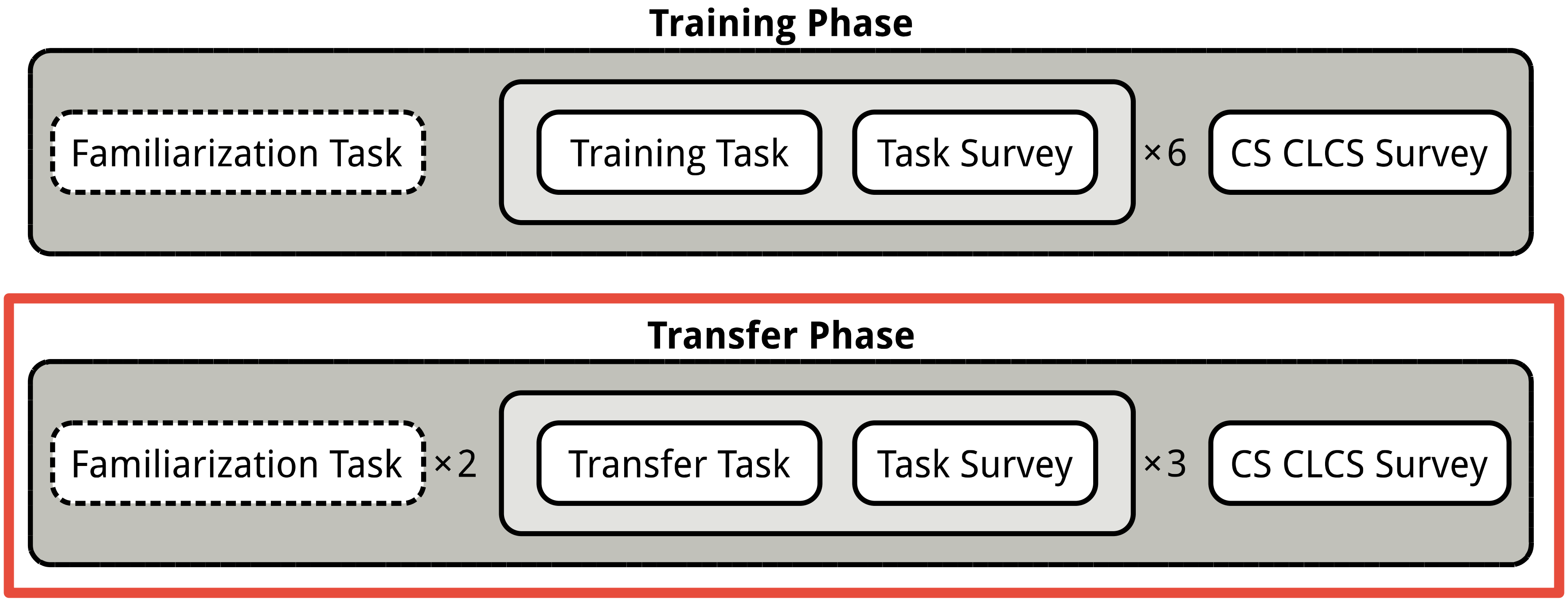


Control: No Distractors

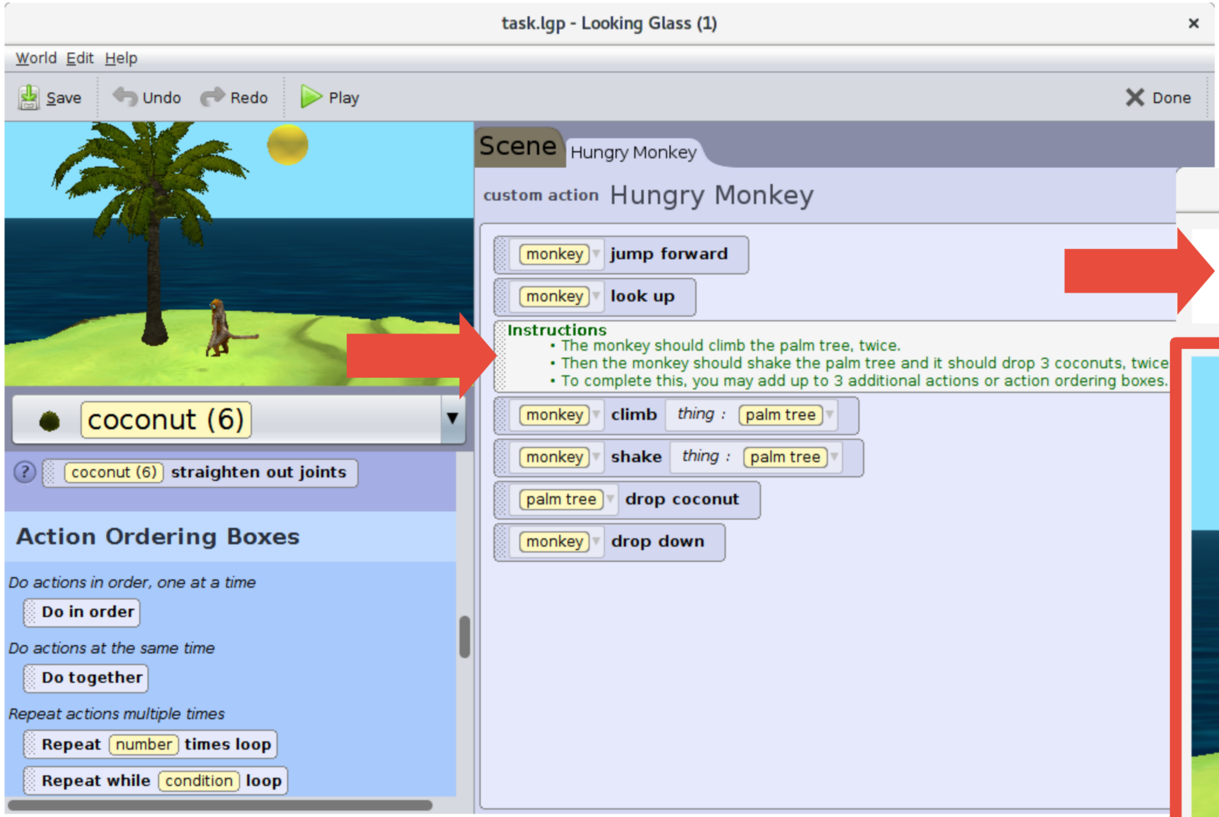


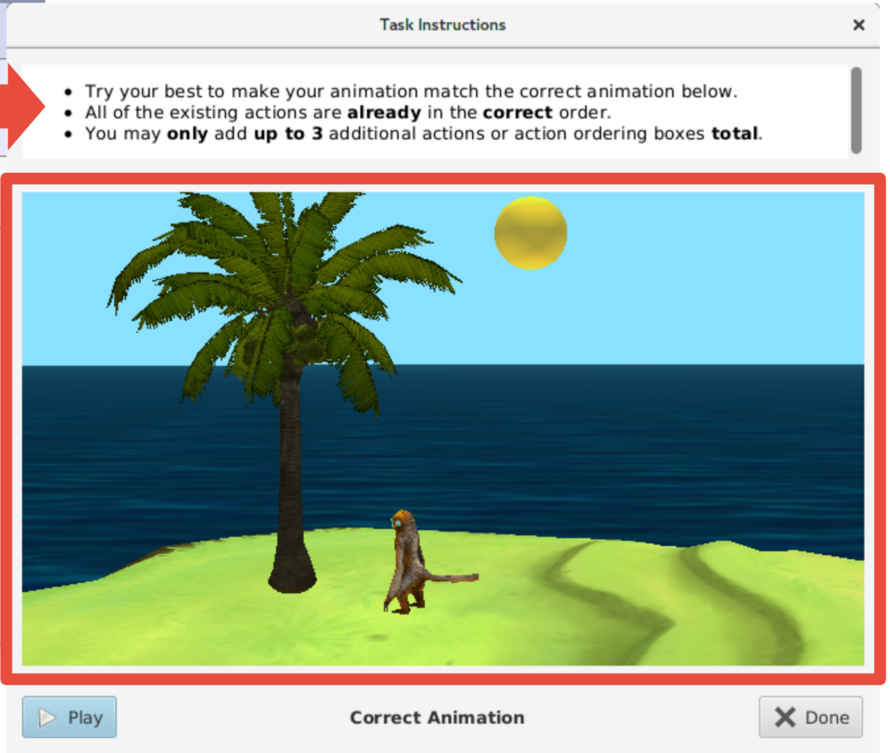
Experimental: Distractors

Study Design: Transfer Phase



Transfer Tasks × 3



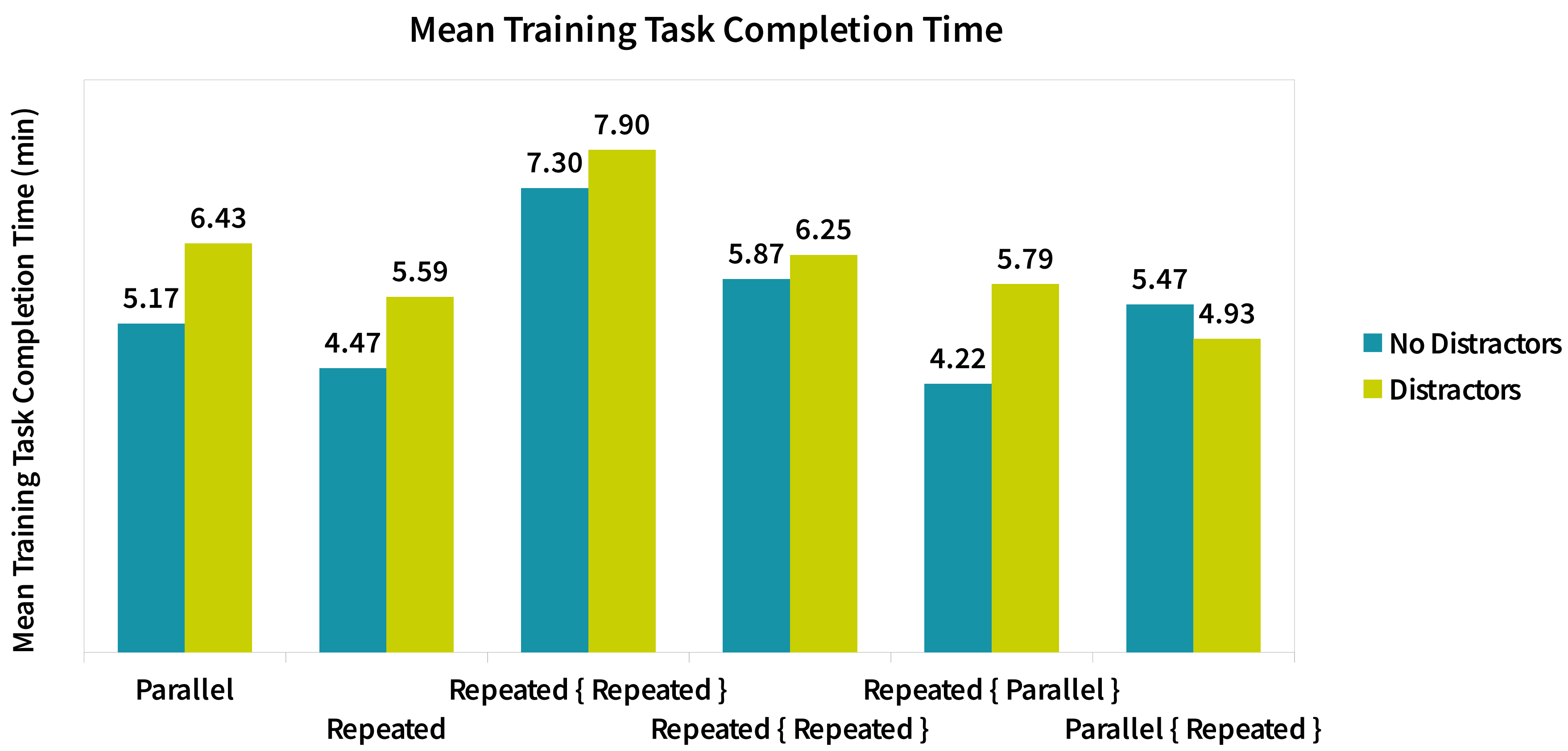


Hypothesis II

Errors in code puzzles, commonly called distractors, will **decrease learning efficiency** by increasing middle school children's time to complete code puzzle completion problems while also **decreasing their ability to successfully utilize programming constructs in near transfer situations**.

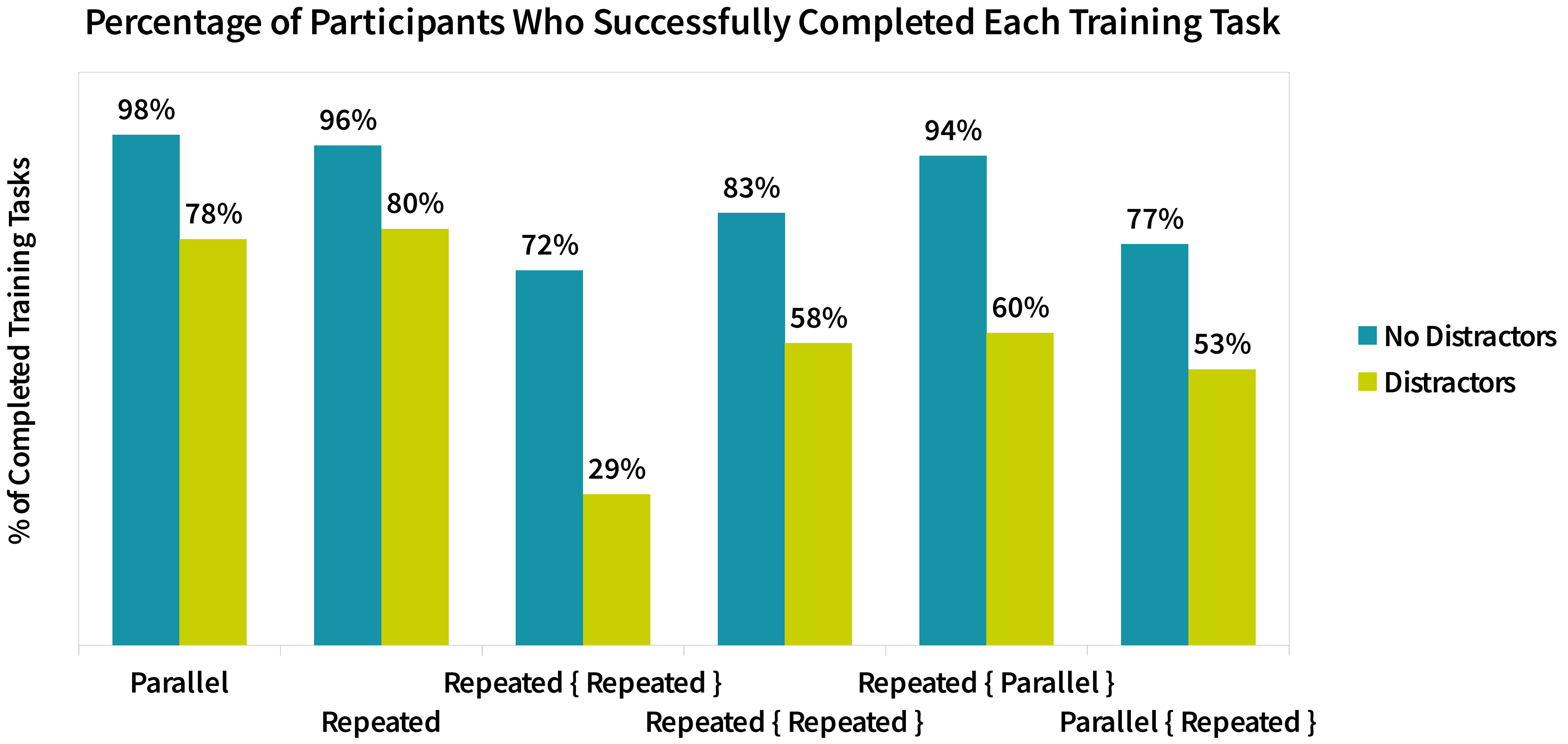
Training Efficiency

Distractors Increased Puzzle Completion Time



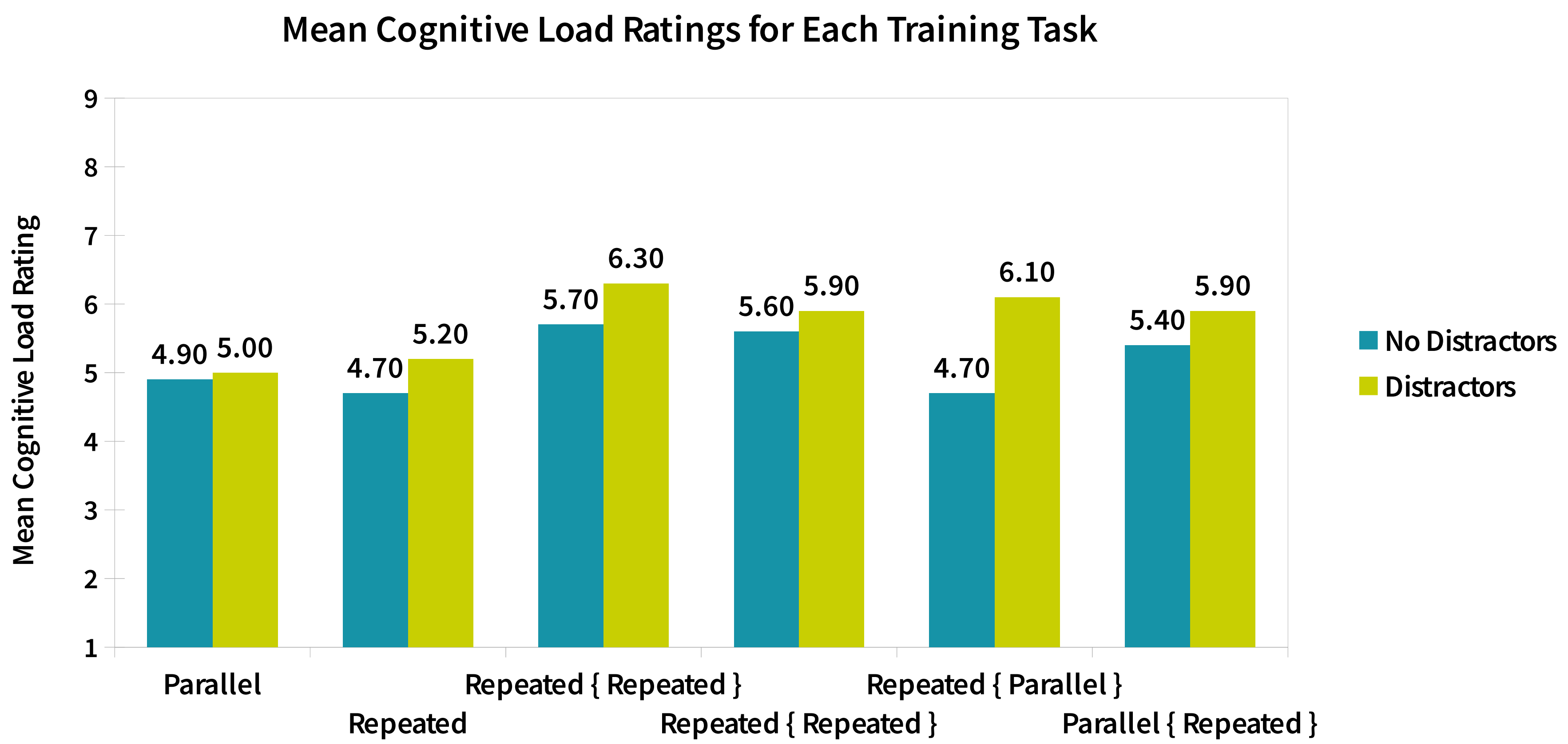
$V = .17$, $F(6, 81) = 2.86$, $p = .014$, $\eta^2 = .17$

Distractor Participants Completed Fewer Puzzles Successfully



$V = .30, F(6, 81) = 5.65, p < .001, \eta^2 = .30$

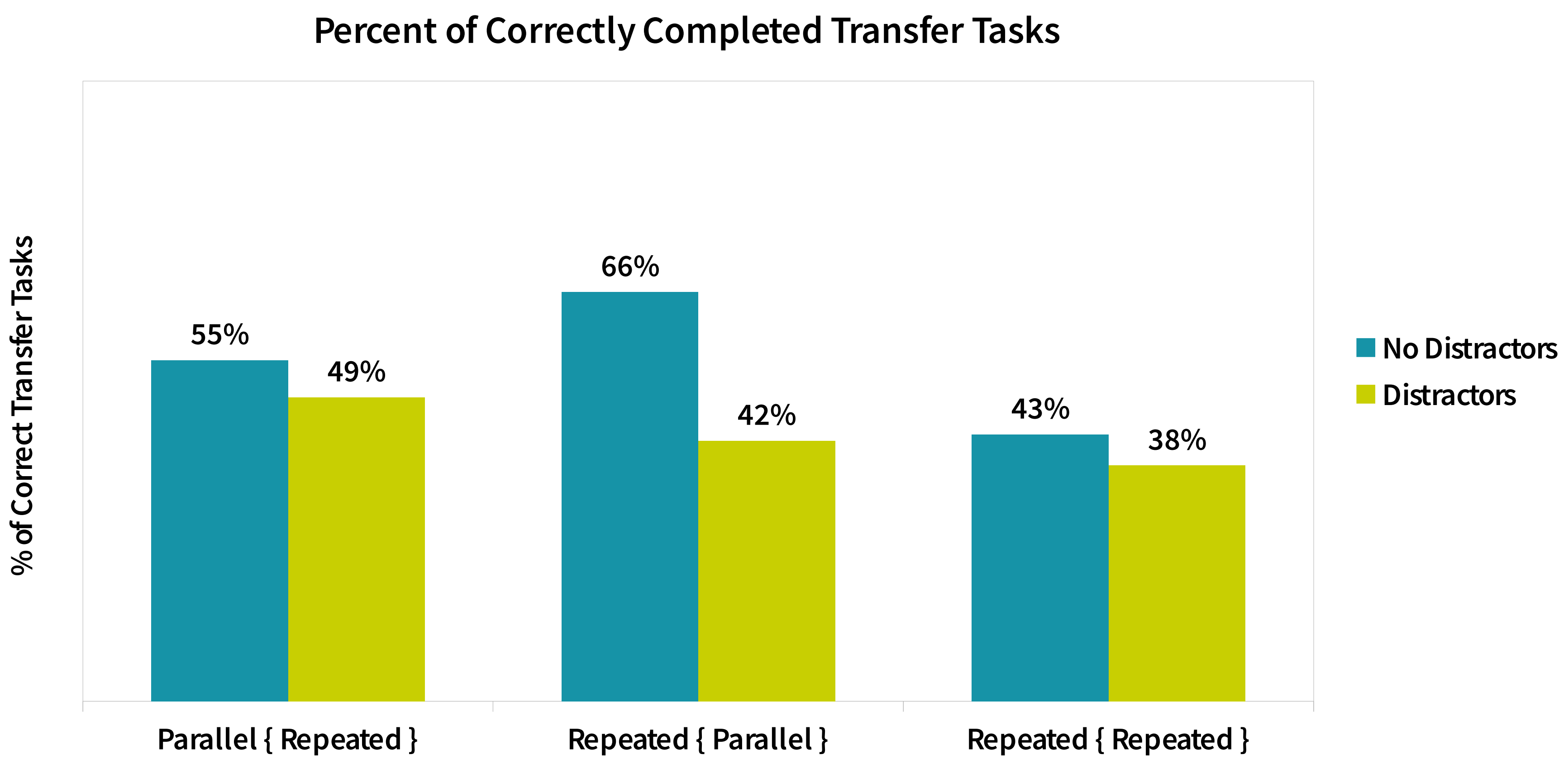
Distractors Increased Cognitive Load (Extraneous) $F(1, 86) = 6.03$, $p = .016$, $\omega^2 = .05$



$V = .21$, $F(6, 81) = 3.55$, $p = .004$, $\eta^2 = .21$

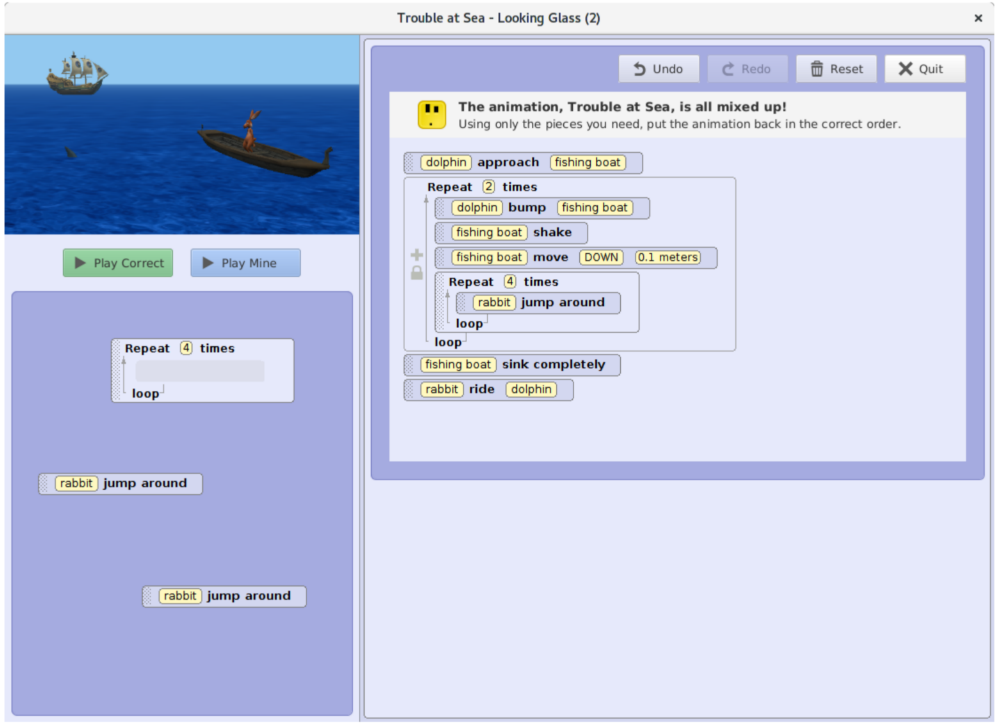
Transfer Task Performance

No Difference in Transfer Task Performance



$V = .06$, $F(3, 84) = 1.78$, $p = .16$, $\eta^2 = .06$

Hypothesis II: Partially Accept



Distractors increased learners' cognitive load, increased their time on task by 14% and decreased their success at completing code puzzles by 26%.

However, distractors did not significantly effect transfer task performance.

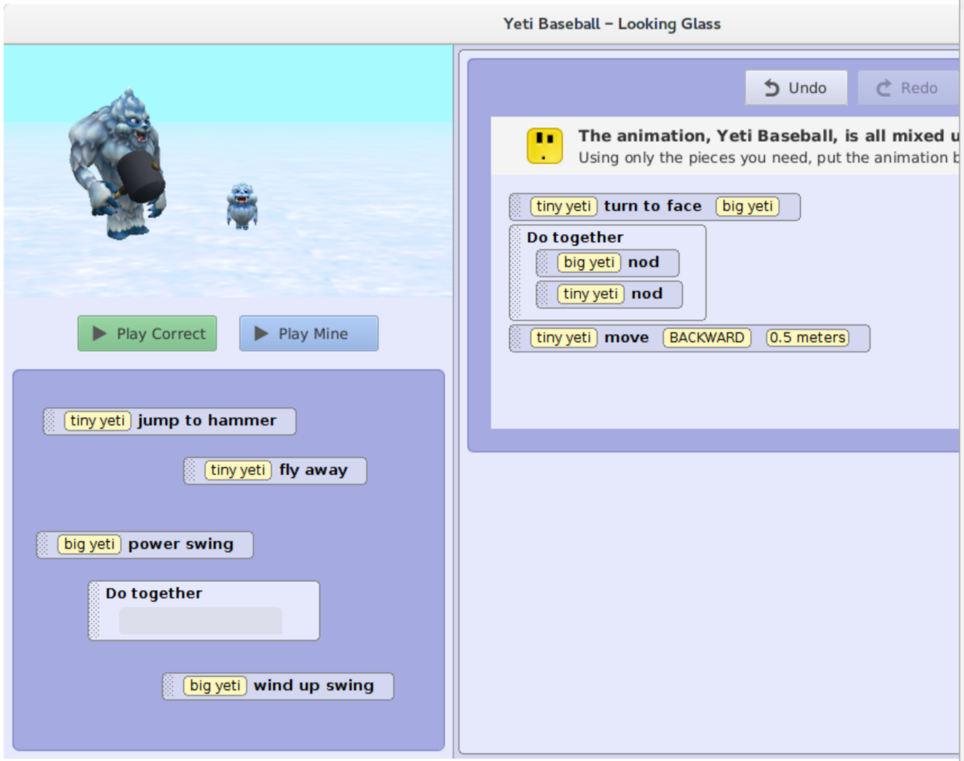
Summative Evaluation:
Hypothesis III: Will learners be motivated to choose to use puzzles?

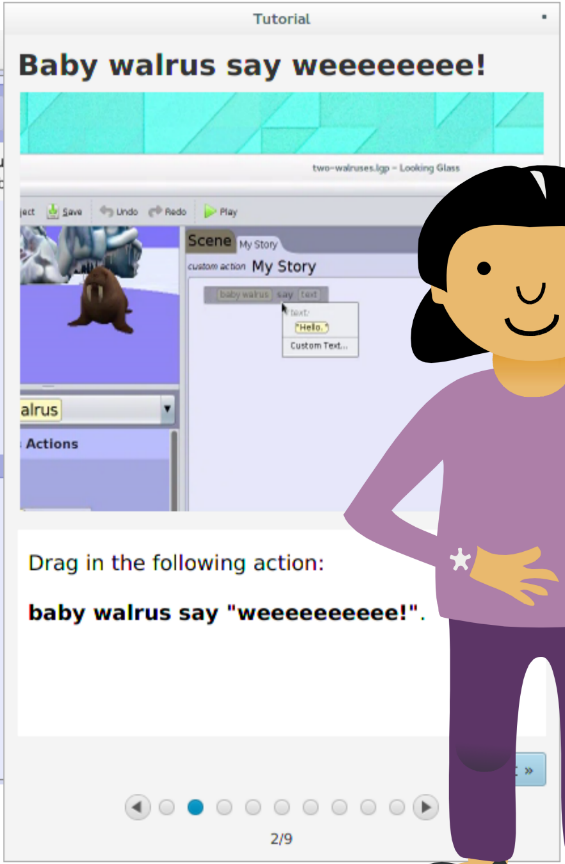
K. J. Harms, E. Balzuweit, J. Chen, and C. Kelleher. 2016. Learning programming from tutorials and code puzzles: Children's perceptions of value. In 2016 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), 59–67. https://doi.org/10.1109/VLHCC.2016.7739665

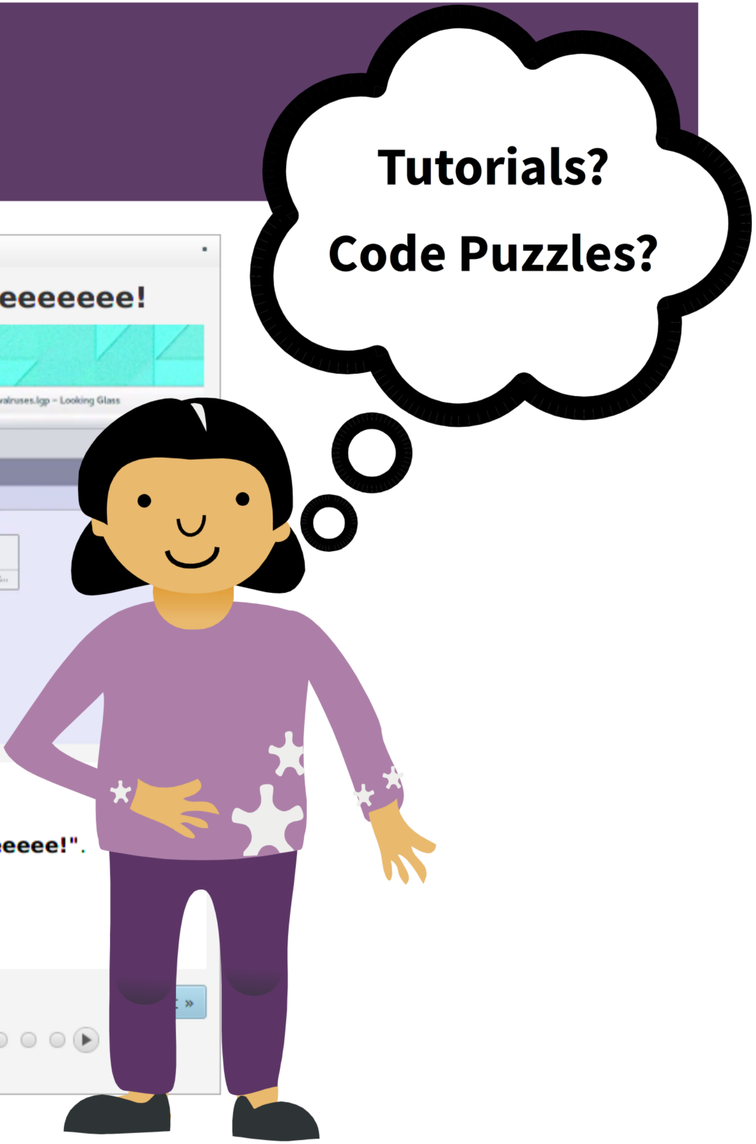
Goal-Oriented Learners

Independent Learning Support in Novice Programming Environments

What do Goal-Oriented Users Choose to Use?







Hypothesis III

When given the freedom to choose their own informal learning resources, middle school children will perceive value in using code puzzle completion problems by expressing more interest, enjoyment, and preference towards using them compared to tutorials.

Summative Evaluation

30 participants (10 to 15 years)

- 14 female, 16 male
- Average age: 11.2 years

2 hours

Instructional Tasks

Dizzy Walrus

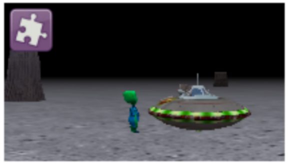
Hammer Hazard

Monkey Business

Yeti Baseball

The Snow Dance

Messed Up Magic



Space Mechanic

Trouble at Sea

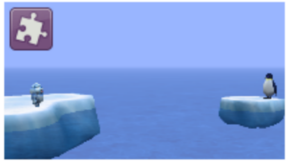
Whack-a-Yeti

Crazy Cauldron

Shark Snack

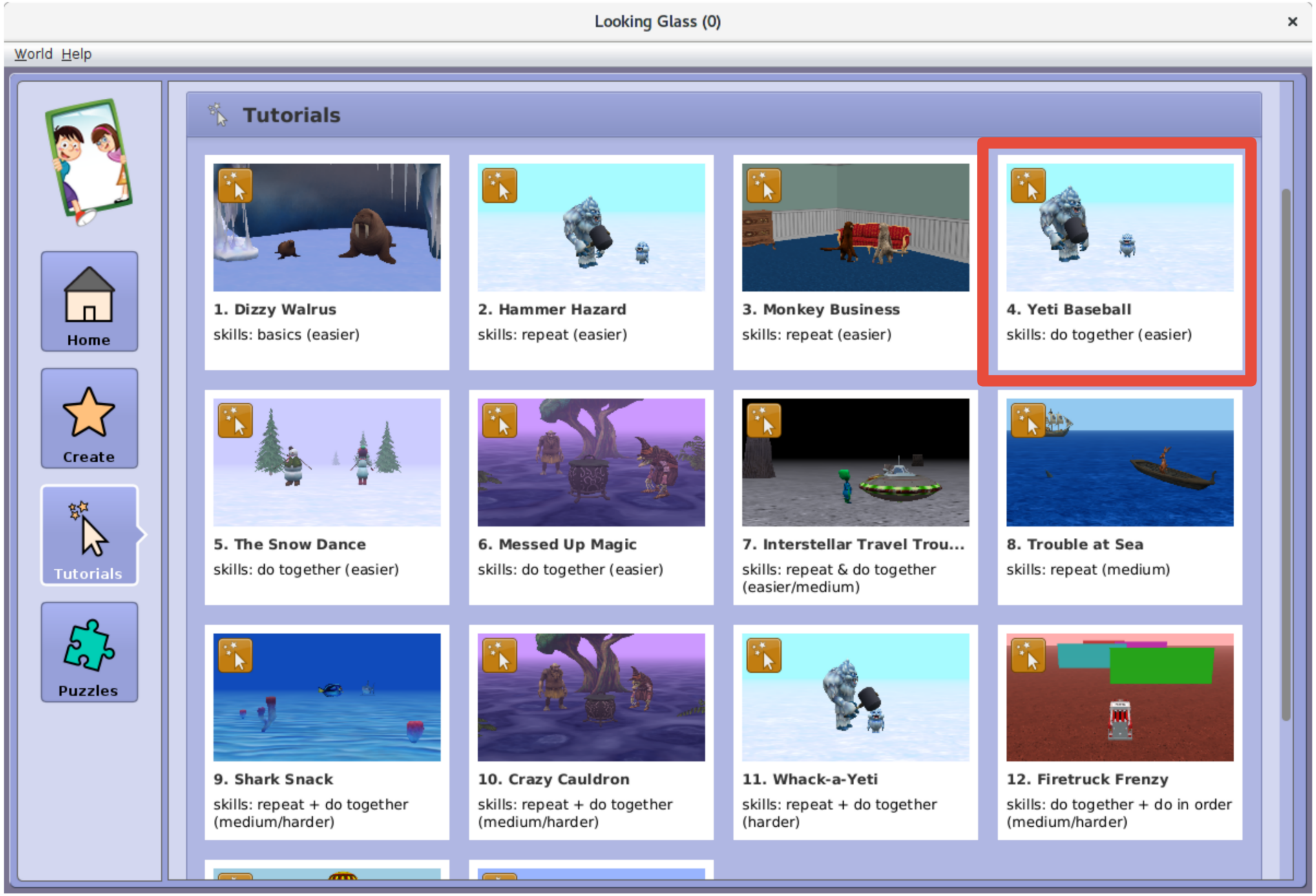
Firetruck Frenzy

Air Traffic

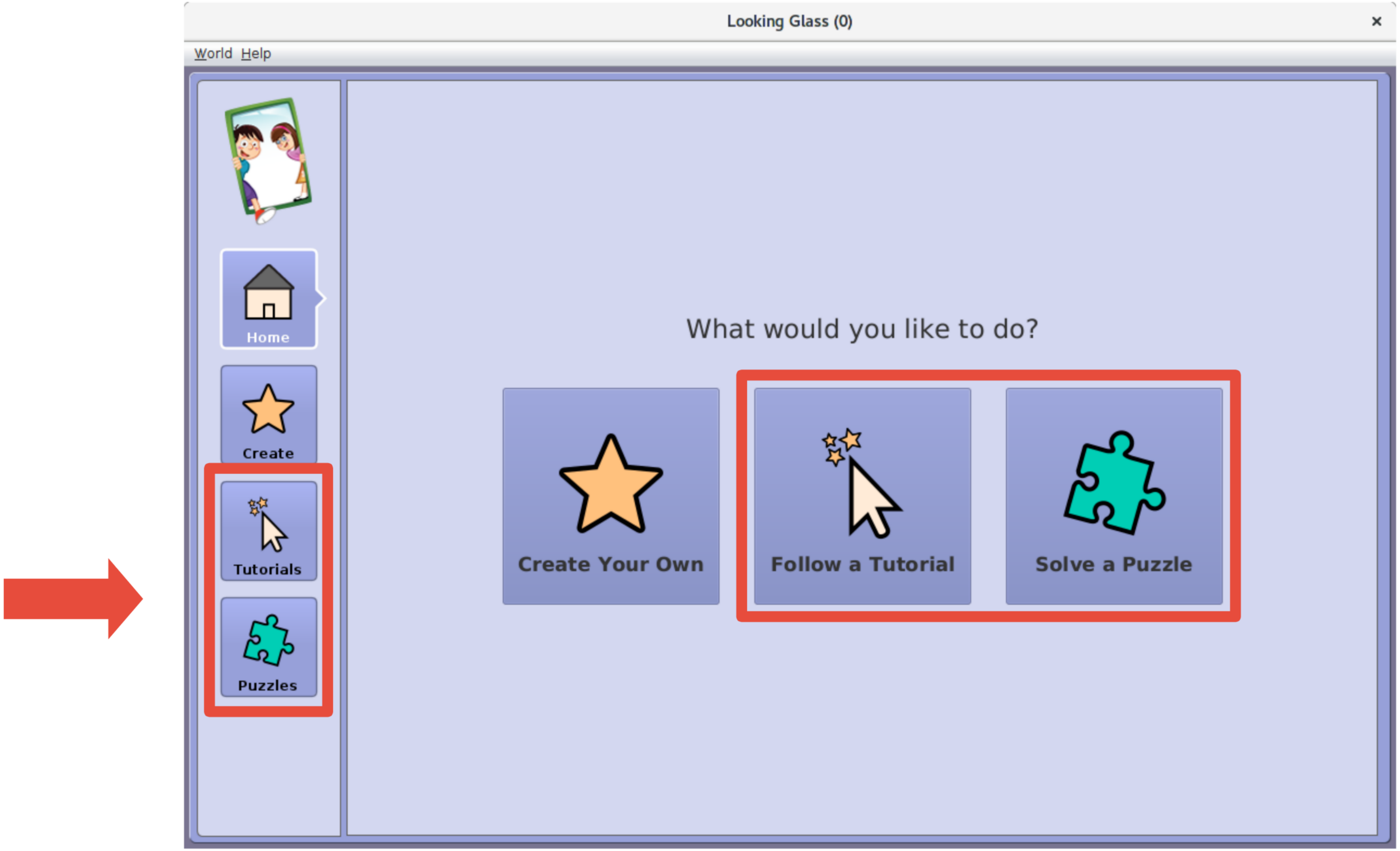


Polar Surprise

Choose Instructional Task × 6



As Tutorial or Puzzle...



Semi-Structured Interviews

Pre-Study Interview

“How would you rate your current programming or coding expertise? why?”

...

Post-Task Interviews × 6

“Was the experience of completing the tutorial/puzzle valuable to you in any way? why?”

“Did you learn anything new or did you acquire any new skills while doing the tutorial/puzzle?”

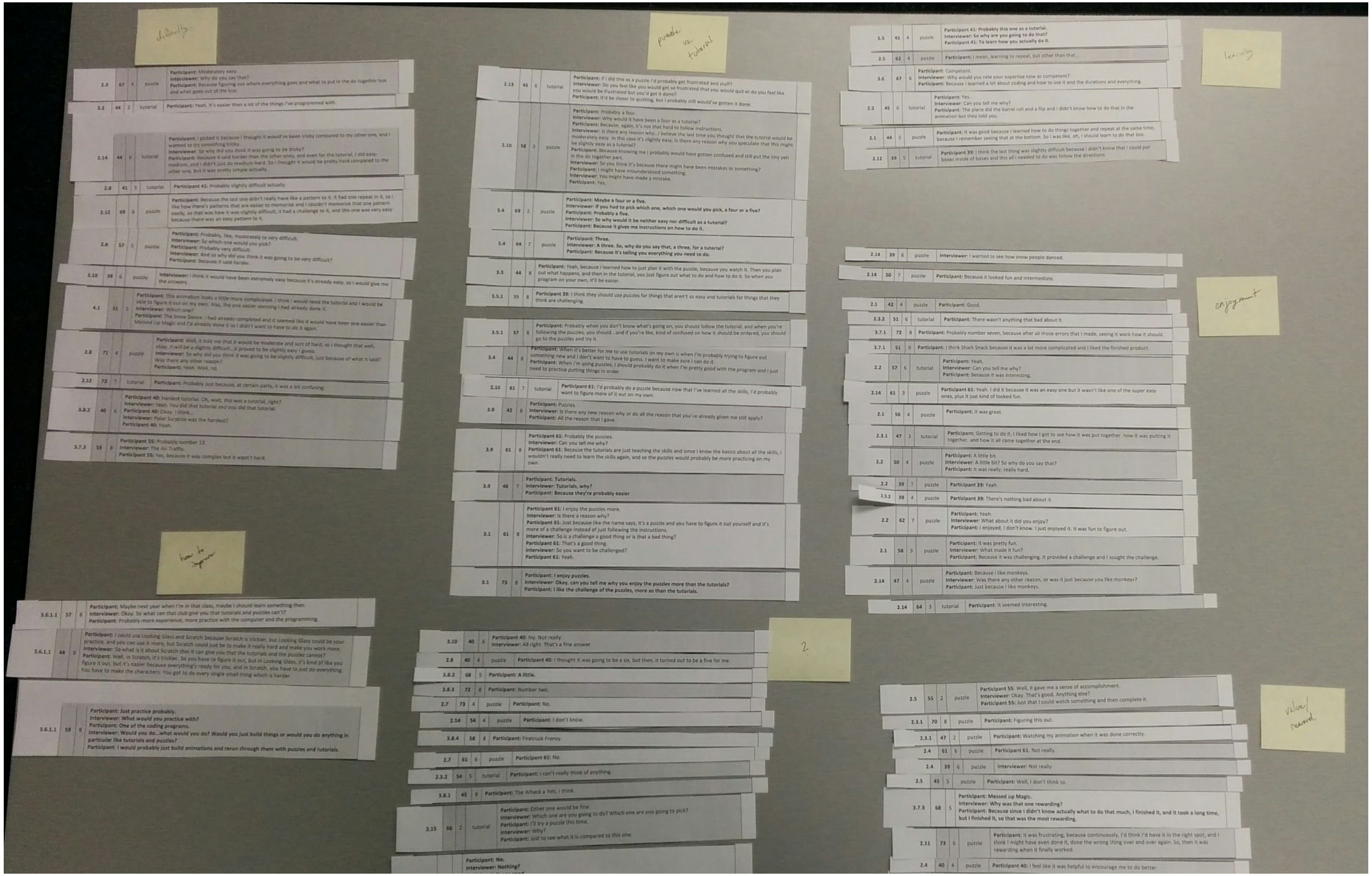
...

Post-Study Interview

“When is it better to to use tutorials/puzzles on your own? why?”

...

Interview Response Analysis: Categories



High-Level Category Themes – Interrater Agreement

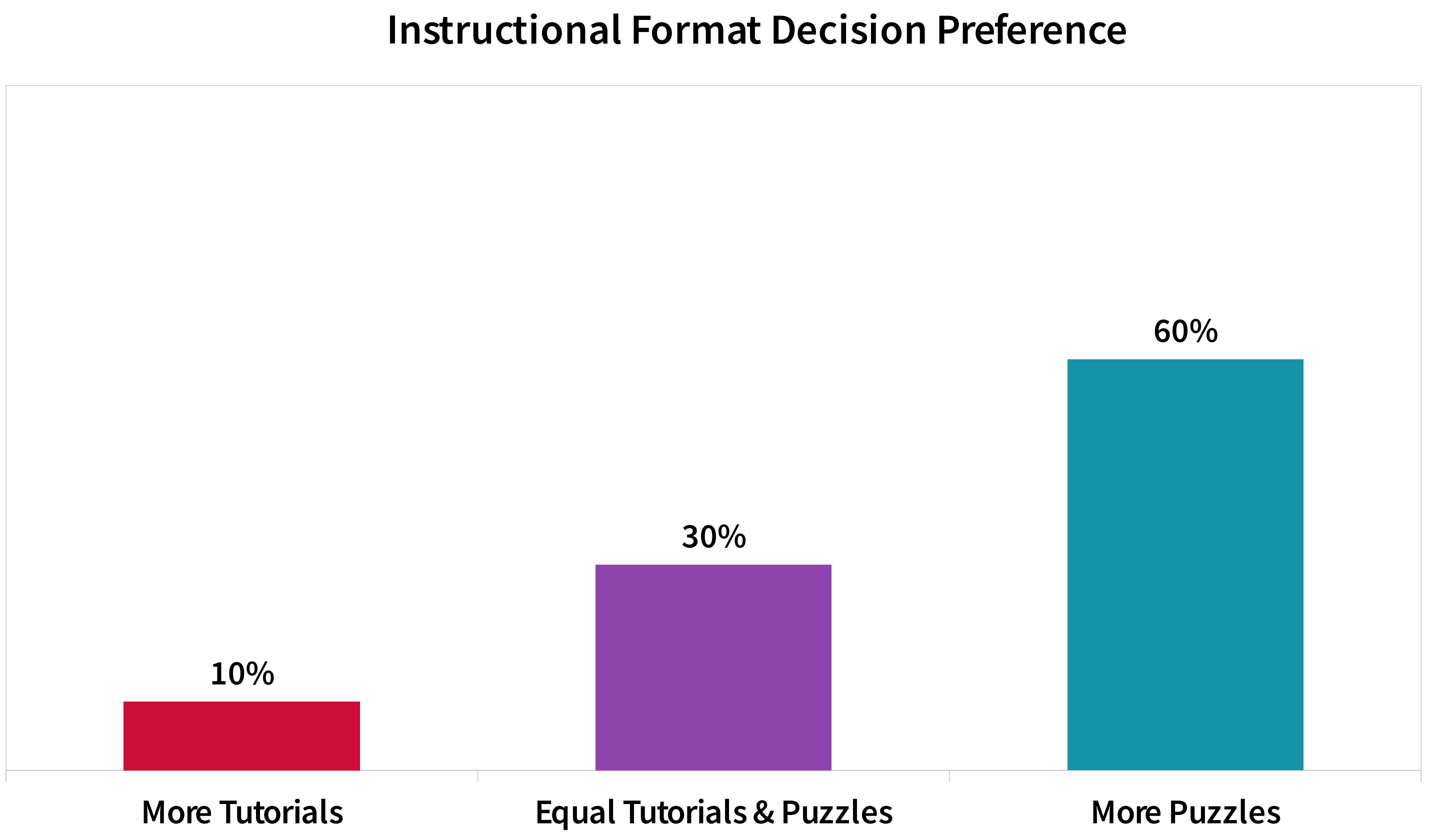
	Cohen's κ	*p*
High-Level Categories	.95	p < .001
Decision Rationales	.89	p < .001
Experience Outcomes	.86	p < .001
Expected Task Difficulty	.85	p < .001
Sources of Ease & Difficulty	.88	p < .001

Hypothesis III

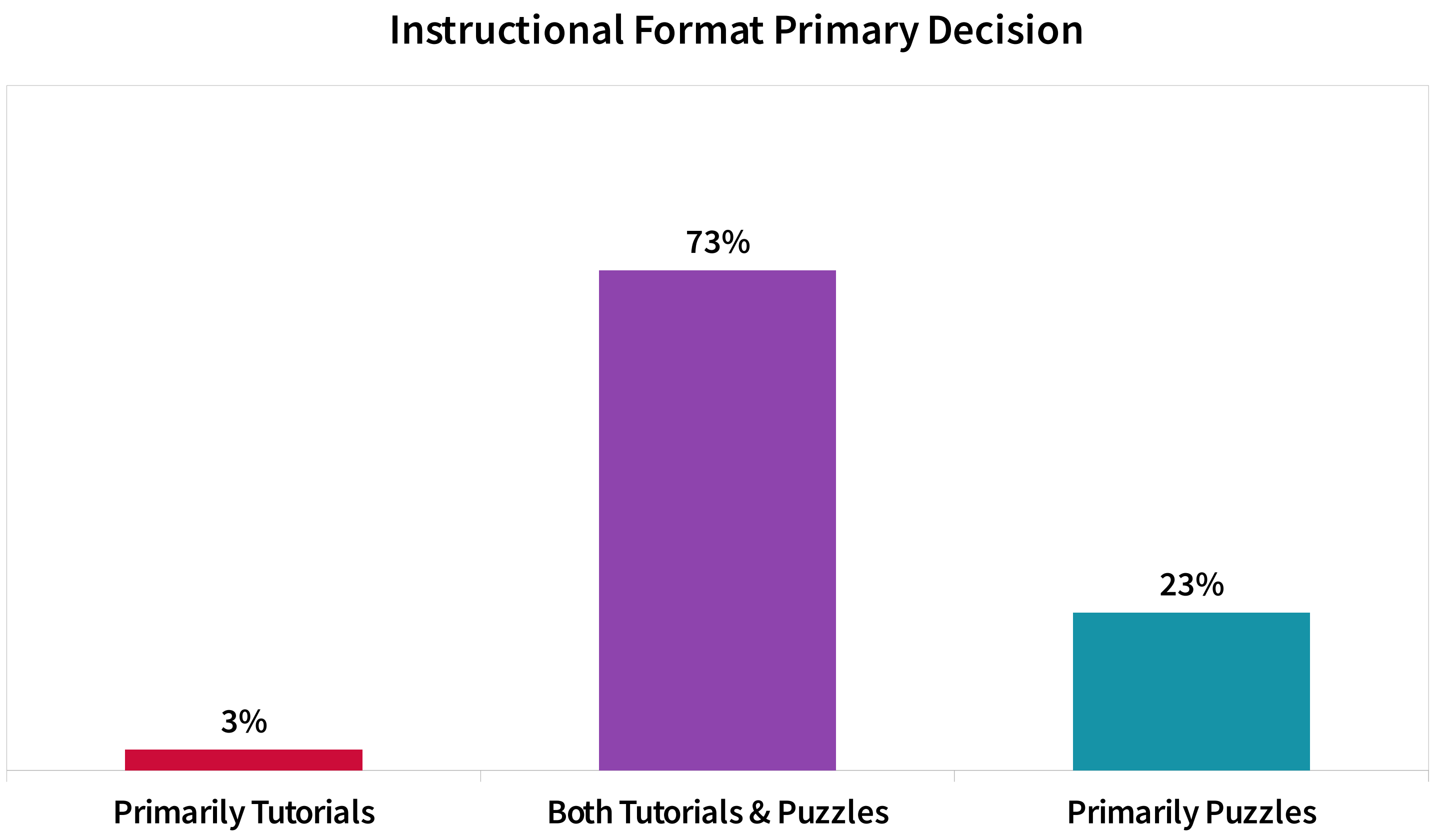
When given the freedom to choose their own informal learning resources, middle school children will perceive value in using code puzzle completion problems by expressing **more interest, enjoyment, and preference towards using them compared to tutorials**.

Decisions

Decisions: Preferred *X*

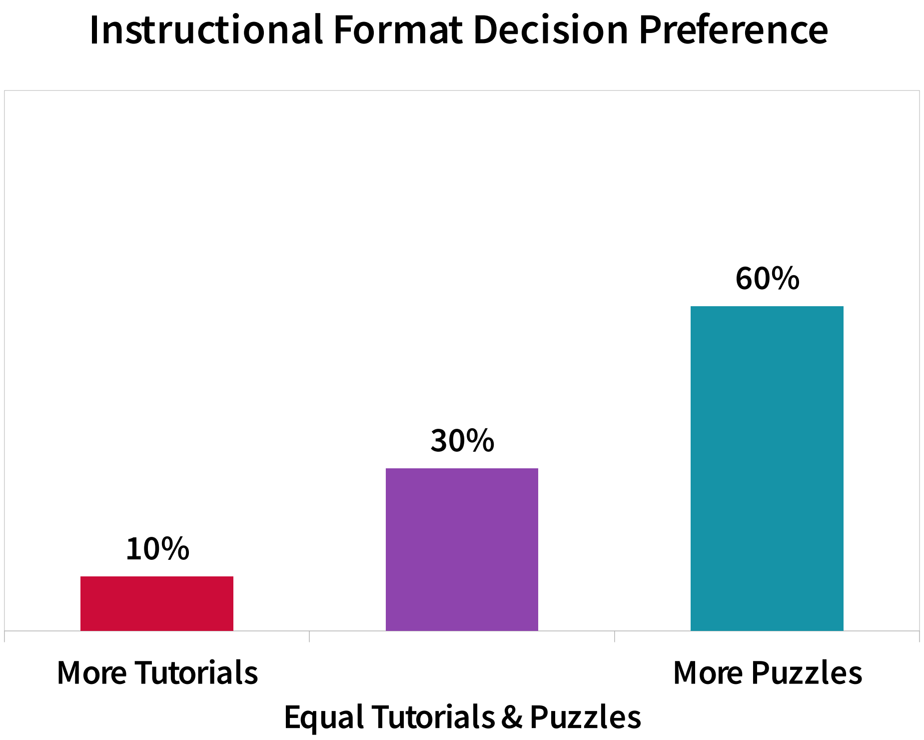


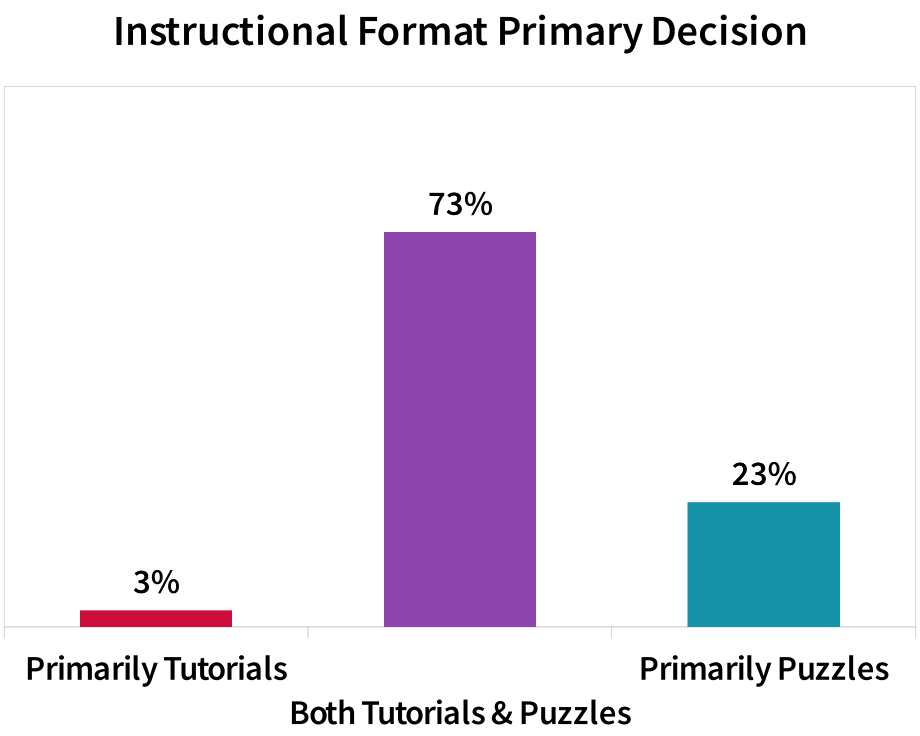
Decisions: Primarily* *X*



*($\geq n - 1$)

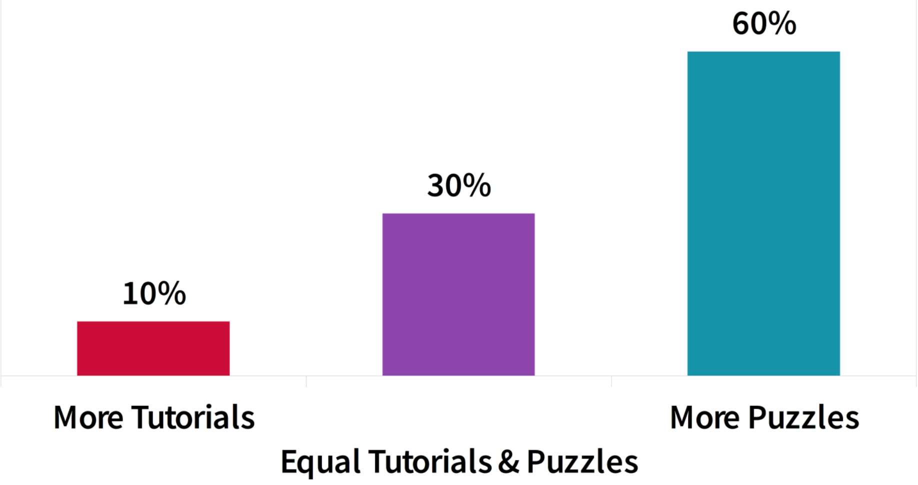
Preference vs. Primary Use



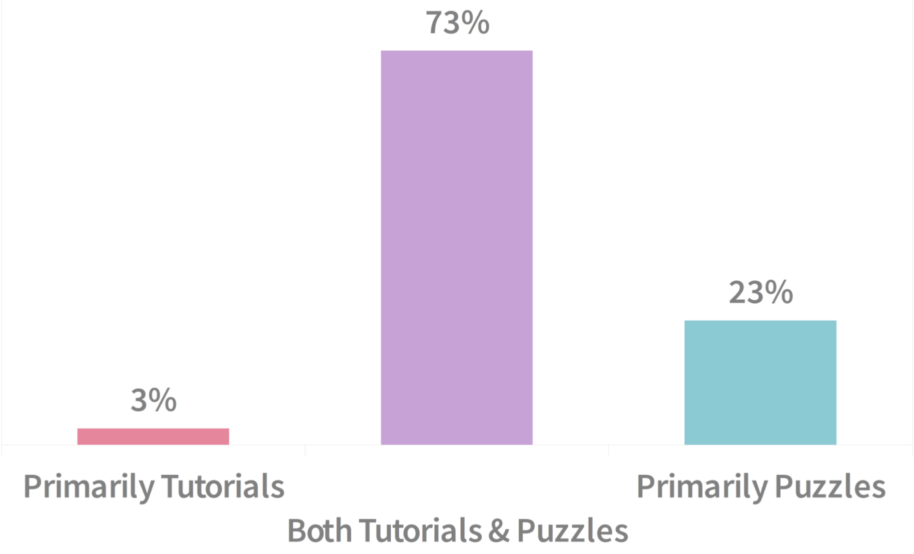


Puzzle Preference

Instructional Format Decision Preference



Instructional Format Primary Decision



High-Level Categories

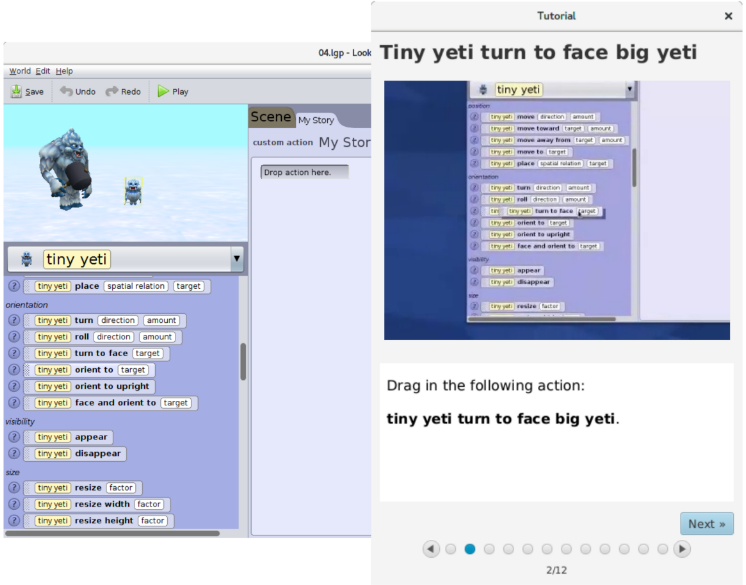
	Cohen's κ	*p*
High-Level Categories	.95	$p < .001$
Decision Rationales	.89	$p < .001$
Experience Outcomes	**.86**	**$p < .001$**
Expected		$p < .001$
Sources of Ease & Difficulty	.88	$p < .001$

Enjoyment

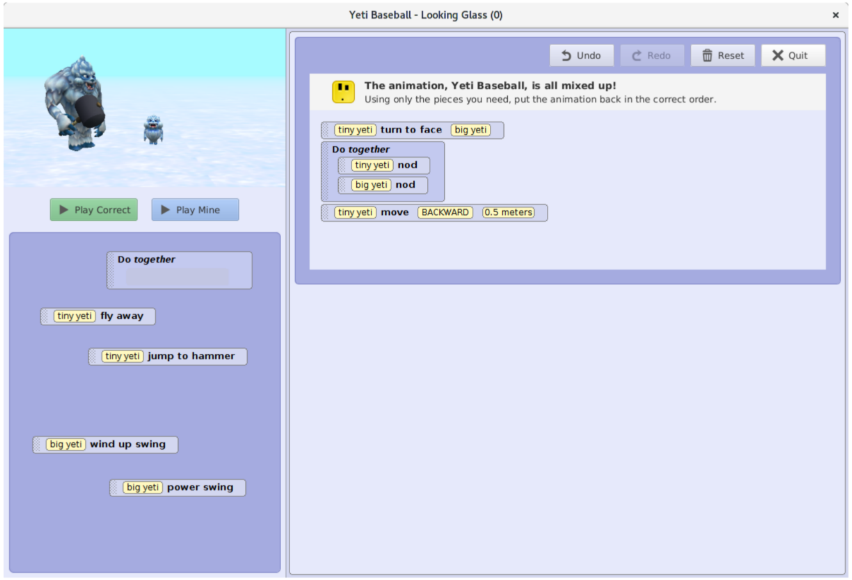
Improve Programming Skills

Experience Outcomes: Liked Instructional Format

"I like the puzzles. They've all been fun."

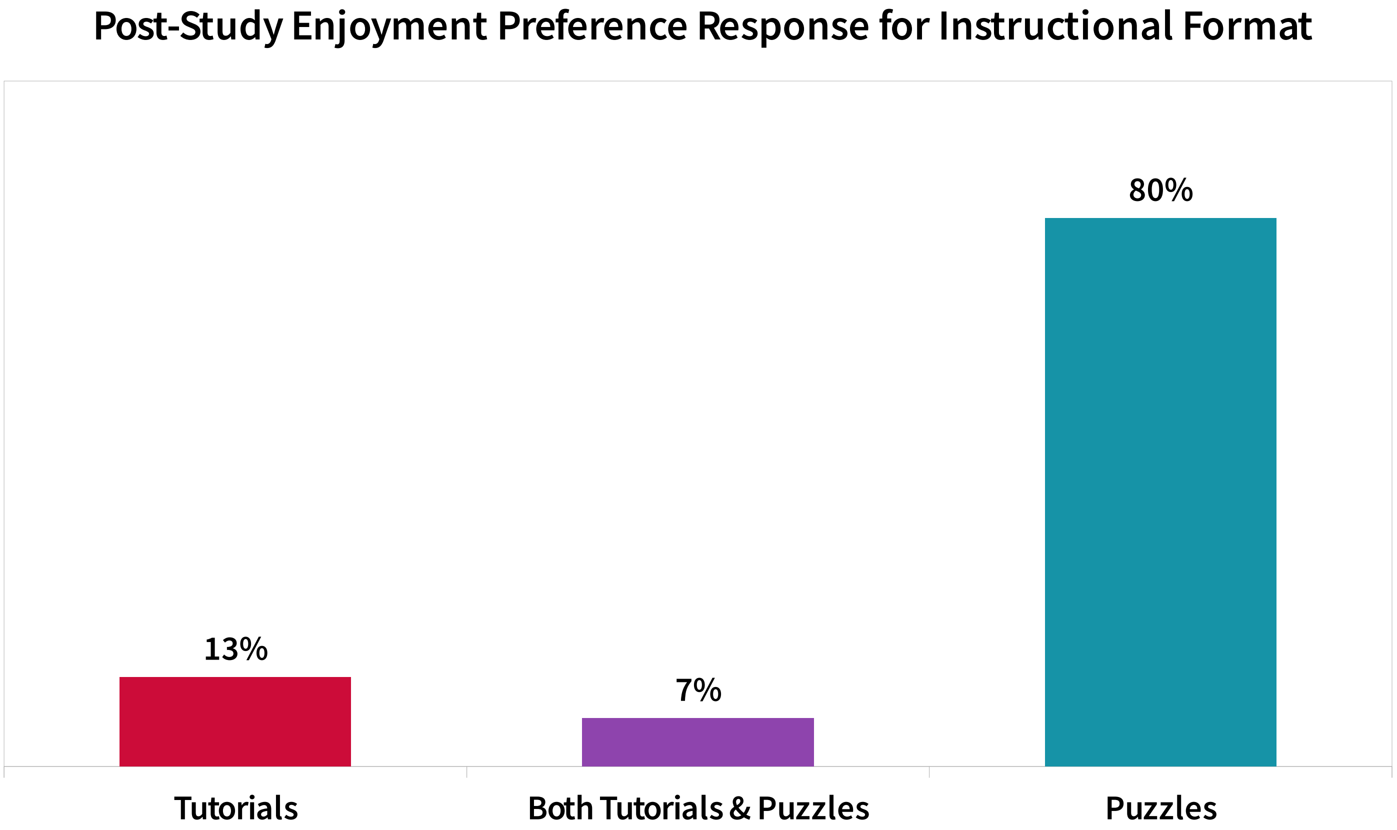


Liked Tutorial
17% of participants

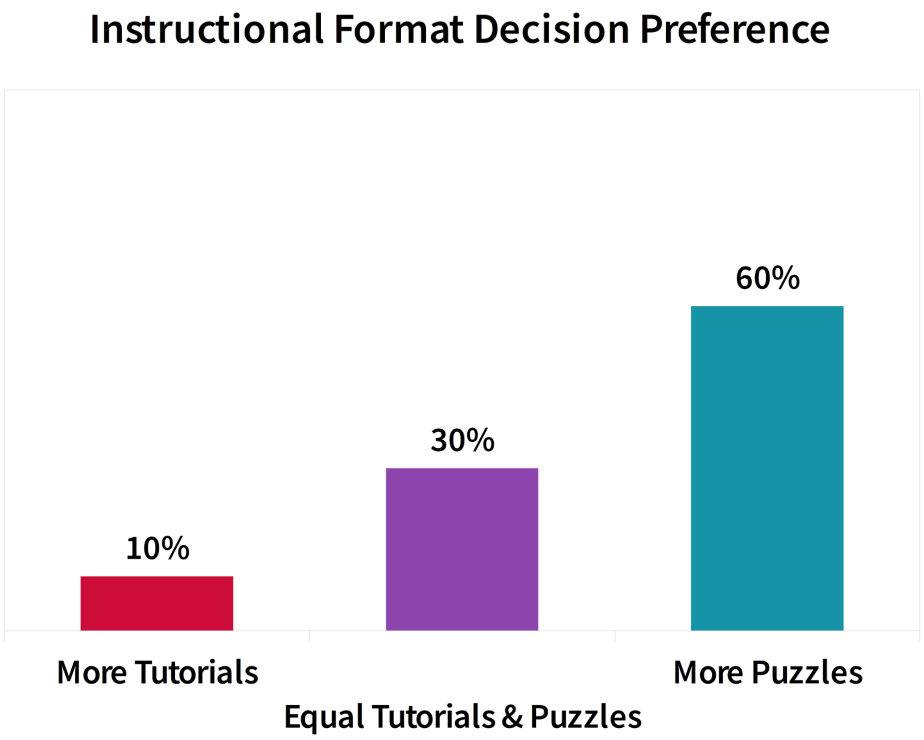


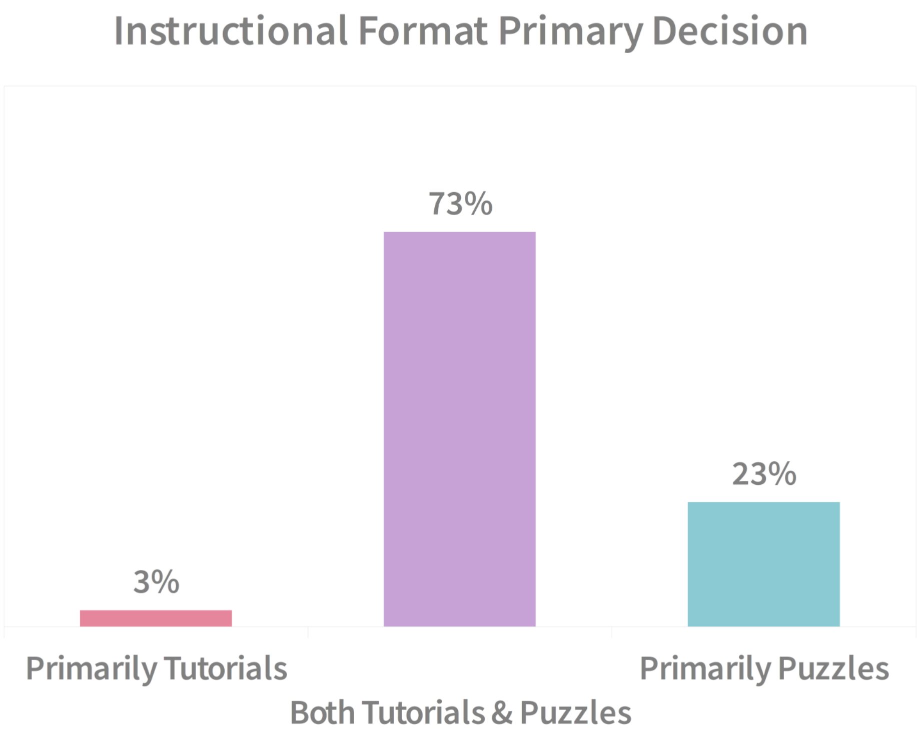
Liked Puzzle
43% of participants

Post-Study Interview: Which do you enjoy more?

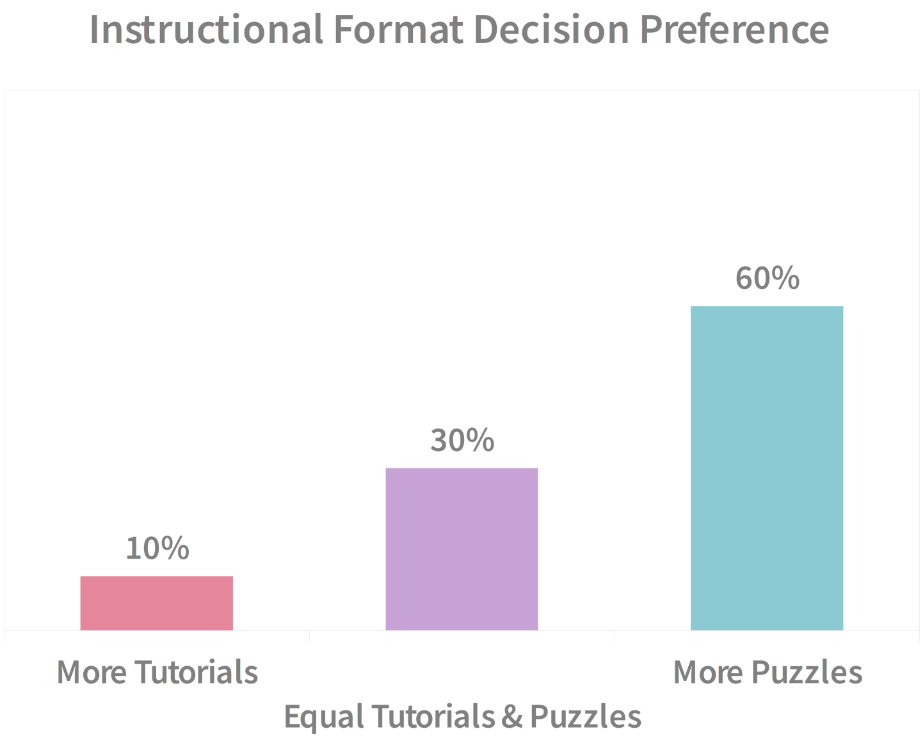


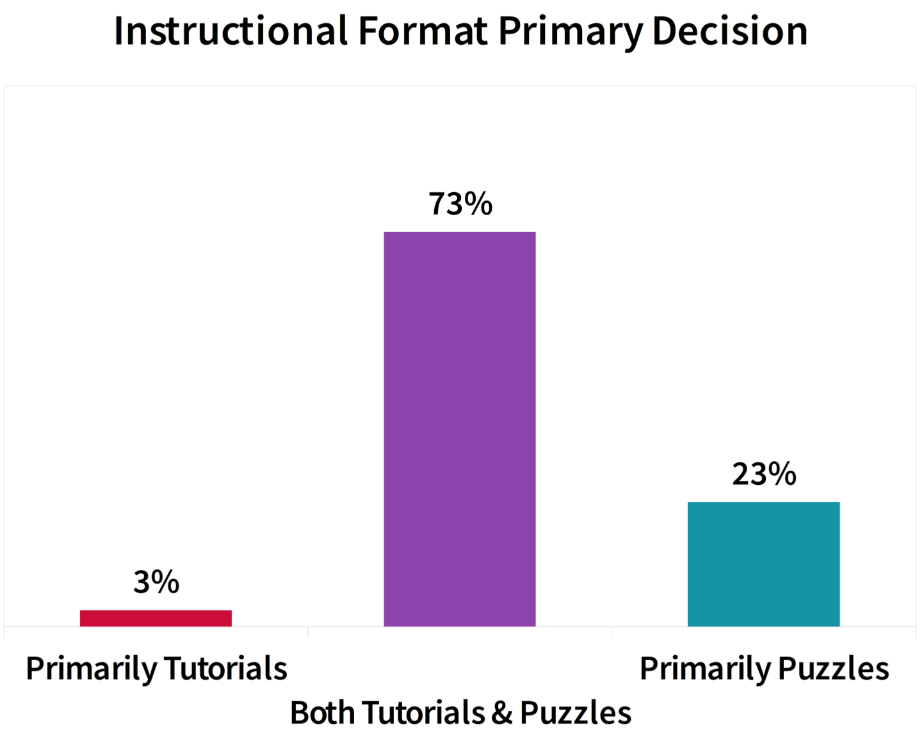
Puzzle Preference





Using Tutorial and Puzzles





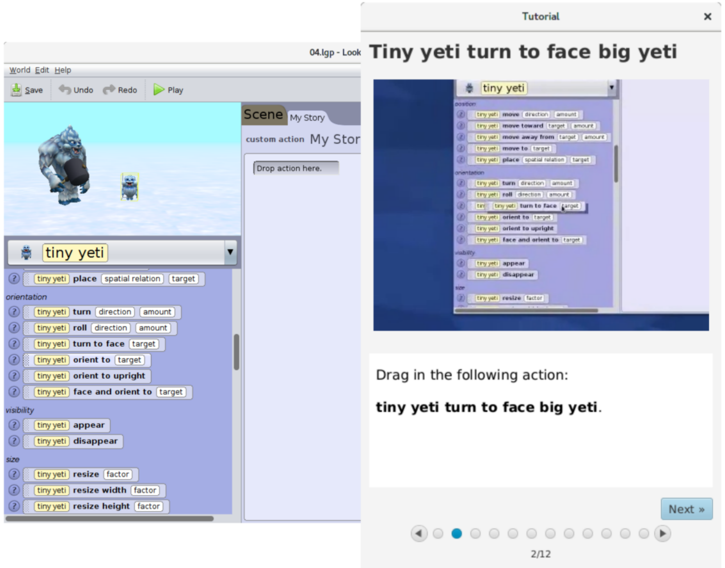
High-Level Categories

	Cohen's κ	*p*
High-Level Categories	.95	p < .001
Decision Rationales	.89	p < .001
Experience Outcomes	**.86**	**p < .001**
Expected		p < .001
Sources of Ease & Difficulty	.88	p < .001

Enjoyment

Improve Programming Skills

Experience Outcomes: Improve Programming Skills

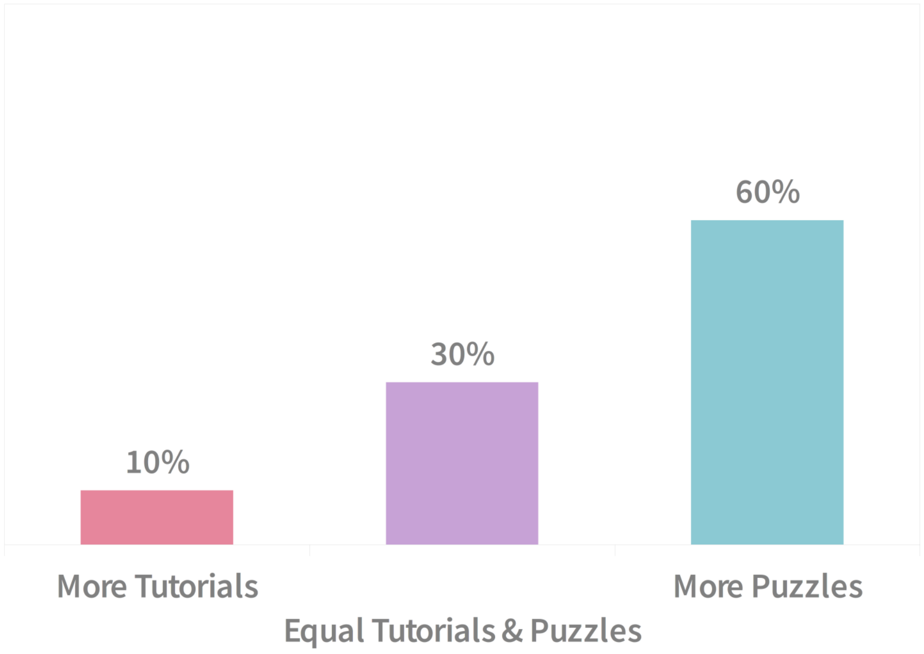


Tutorial Improved Programming Skills
60% of Participants

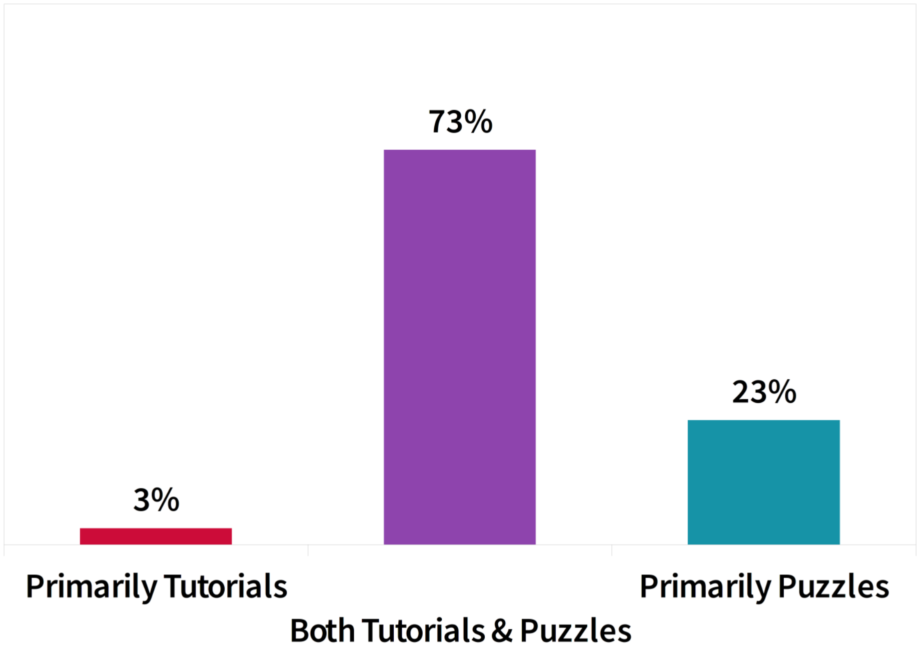
Puzzle Improved Programming Skills
70% of Participants

Using Tutorial and Puzzles

Instructional Format Decision Preference



Instructional Format Primary Decision



Decision Rationale Themes

	Cohen's κ	*p*
High-Level Categories	.95	$p < .001$
Decision Rationales	**.89**	**$p < .001$**
Experie		$p < .001$
Expected		$p < .001$
Sources of Ease & Difficulty	.88	$p < .001$

Personal Preference

Improve Programming Skills

Challenge

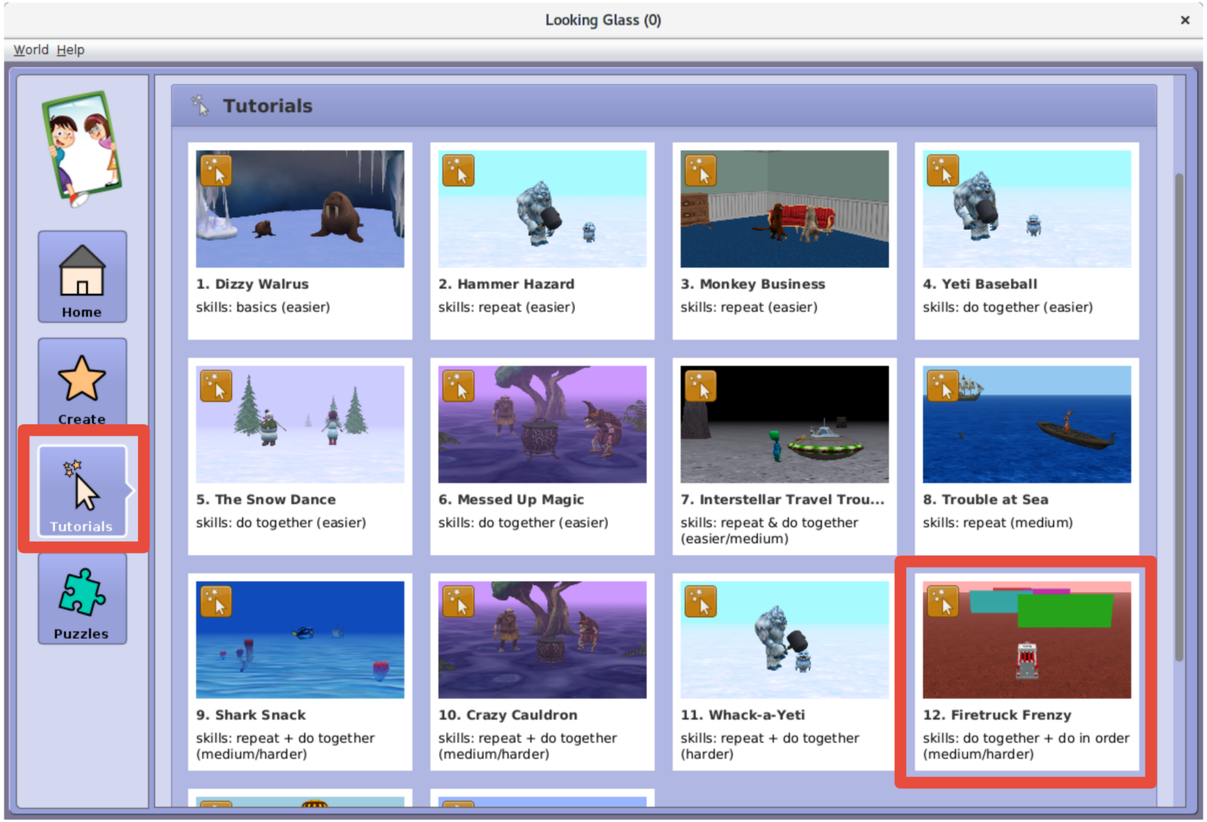
#1 Decision Rationale: Enjoyed Animation

"Probably just because it looked like a fun animation to do."

83% of participants made at least one decision because they enjoyed the animation.

Story-Based Animations are Motivating!

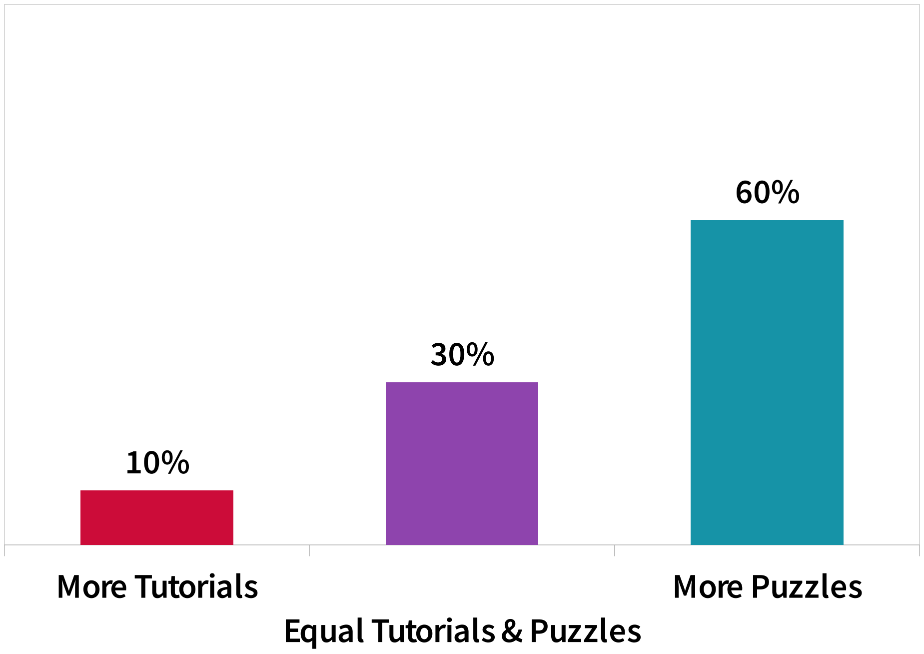
"I picked it as a tutorial because it looked like it had lots more complexity than the other ones and I didn't want to just jump right in without knowing what I was doing."



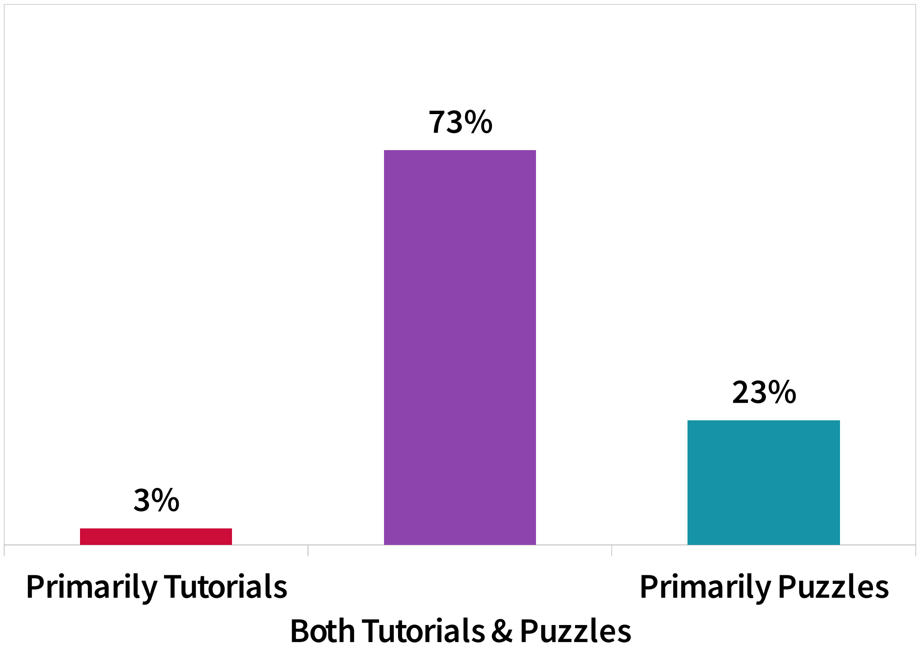
Caitlin Kelleher, Randy Pausch, and Sara Kiesler. 2007. Storytelling Alice Motivates Middle School Girls to Learn Computer Programming. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '07), 1455–1464. https://doi.org/10.1145/1240624.1240844

Learners Like the Puzzles More, But Care More About the Animation

Instructional Format Decision Preference



Instructional Format Primary Decision



Caitlin Kelleher, Randy Pausch, and Sara Kiesler. 2007. Storytelling Alice Motivates Middle School Girls to Learn Computer Programming. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '07), 1455–1464. https://doi.org/10.1145/1240624.1240844

Hypothesis III: Accept

Participants preferred to choose puzzles over tutorials because they found the puzzles as more interesting, enjoyable, and challenging.

However, participants often used both because they wanted to work on a specific animation.

Participants also found both tutorials and puzzles useful for improving their programming skills.

Contributions

Formative Evaluations I & II:

Lessons learned from the development of code puzzle completion problems and curriculum.

Summative Evaluation: Hypothesis I

Code puzzle completion problems are efficient and effective in supporting independent learning.

Summative Evaluation: Hypothesis II

Errors in code puzzle completion problems decrease learning efficiency.

Summative Evaluation: Hypothesis III

Independent learners will choose to use code puzzles because they find them motivating and useful for improving their programming skills.

Future Work

Future Work

Improving the Efficiency & Effectiveness of Code Puzzle Completion Problems

- Using Worked Examples
- Promoting Self-Explanation
- Errors for Experienced Programmers

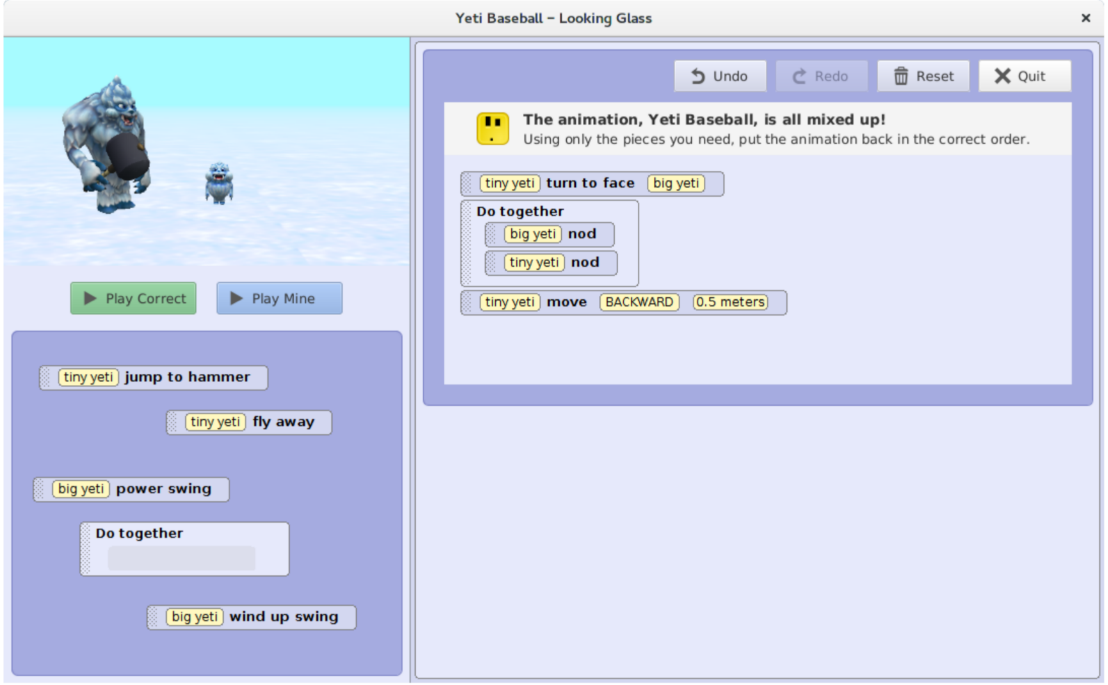
Extending Code Puzzle Completion Problems

- Alternative Contexts
- Curricular Content
- Alternative Completion Approaches

Supporting Goal-Oriented Learners

- Choices for Instructional Formats
- Self-Directed Learning
- Self-Assessment & Scaffolding

Completion Problems + Worked Examples

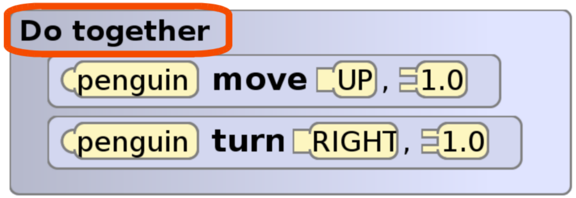


Problem Completion *Effect*

+

Example:

The 'Do together' block makes these actions play at the same time.



Worked Example *Effect*

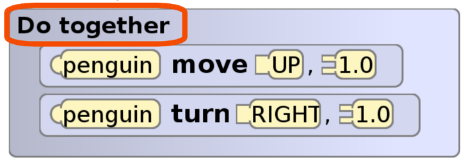
John Sweller, Paul Ayres, and Slava Kalyuga. 2011. Cognitive Load Theory. Springer.
Michelle Ichinco, Kyle J. Harms, and Caitlin Kelleher. "Towards Understanding Successful Novice Example Use in Blocks-Based Programming". In: Journal of Visual Languages and Sentient Systems: Special Issue on Blocks Programming (2017). Forthcoming.

Programming Worked Examples

“A worked example provides a step-by-step solution to a problem.”

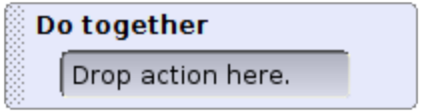
Example:

The 'Do together' block makes these actions play at the same time.

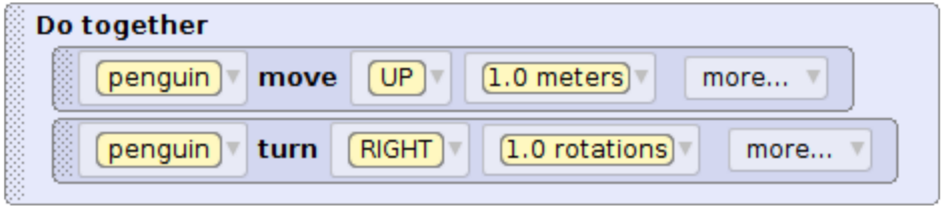


Annotated Example?

Step 1:



Step 2:



Worked Example?

John Sweller, Paul Ayres, and Slava Kalyuga. 2011. Cognitive Load Theory. Springer.

Questions?

http://research.engineering.wustl.edu/~harmsk/

https://lookingglass.wustl.edu/