Automatically Generating Tutorials to Enable Middle School Children to Learn Programming Independently

Kyle Harms, Dennis Cosgrove,
Shannon Gray, Caitlin Kelleher

LookingGlass

Washington University in St.Louis

Shortage of Programmers

An estimated 1.4 million computing jobs will be added to the United States' economy between 2008-2018.[1]

61% of these jobs can be filled based on current college graduation rates.[1]

Shortage of information communications technology workers across the European Union.[2]

[1] Computing Education and Future Jobs: A Look at National, State, and Congressional District Data (2011)

[2] IEEE Job Site: *http://careers.ieee.org/article/European_Job_Outlook_0312.php*

Middle School Children & Computer Programming

Middle school is the time many children decide to opt-out of advanced math or science courses.[1]

By college these students are too far behind to realistically succeed in these majors.[2]

Maintain interest and develop programming skills through independent learning.

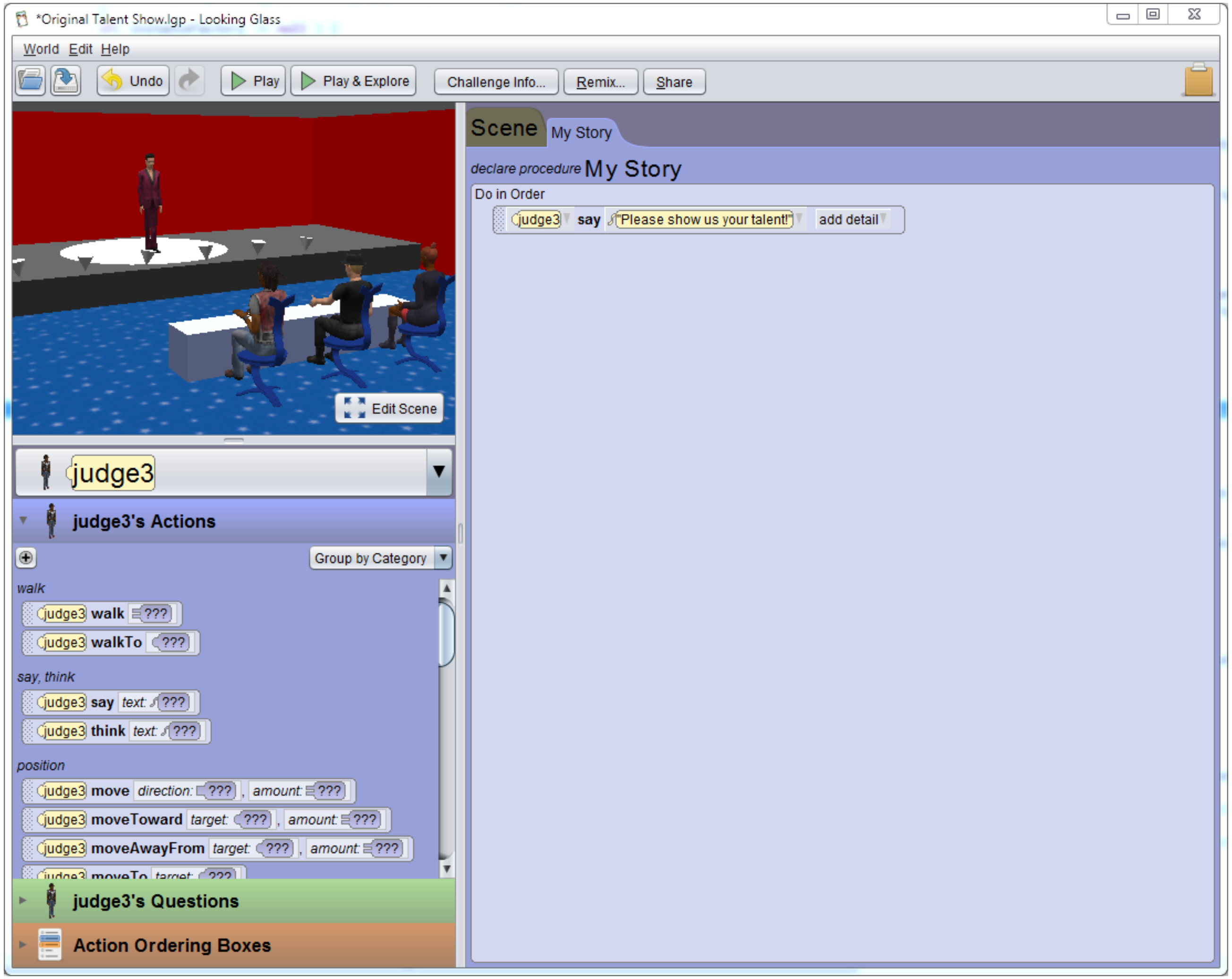
[1] Shedding Some New Light on Old Truths: Student Attitudes to School in Terms of Year Level and Gender (1994)
[2] Pryor, J.H. et al. 2010. *The American Freshman: National Norms for Fall 2009*.

Contributions

Demonstrate a process for automatically generating programming tutorials from unfamiliar code.

The tutorials improved independent learning of programming constructs in near transfer tasks by 64%.

Looking Glass

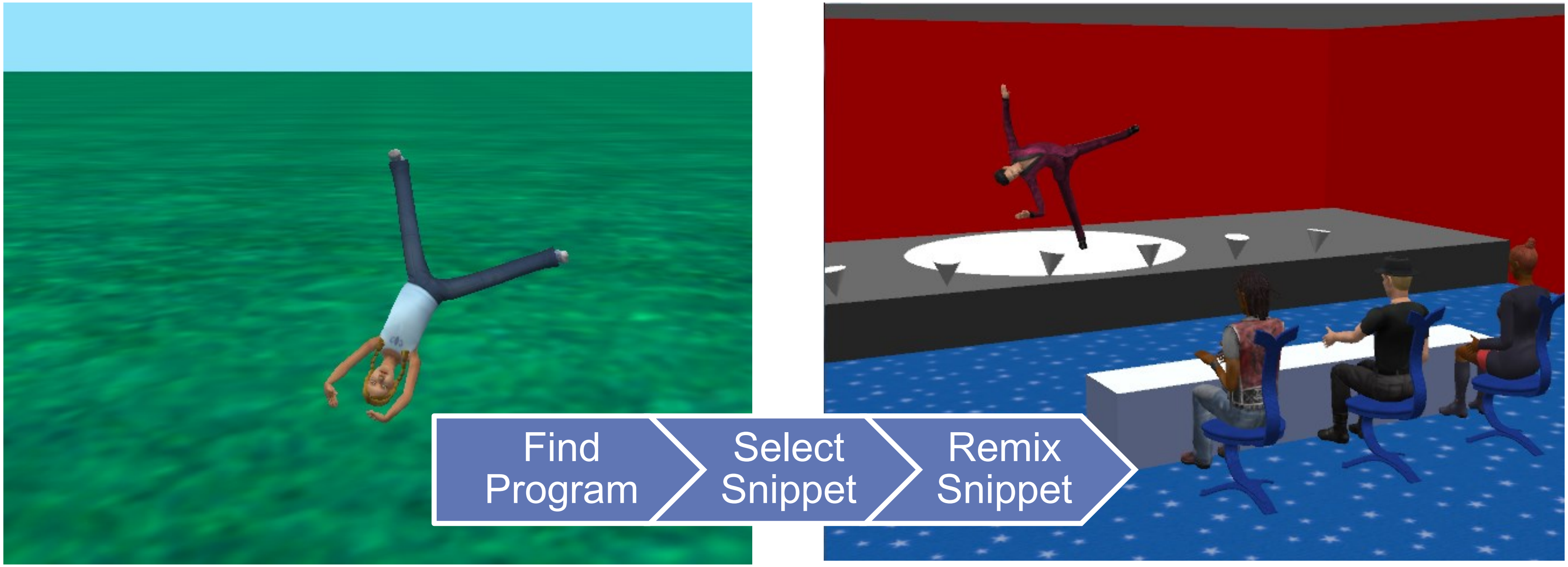


```
declare procedure My Story
Do in Order
    judge3 say "Please show us your talent!" add detail
```

Independent Learning In Looking Glass

Talent Show Program

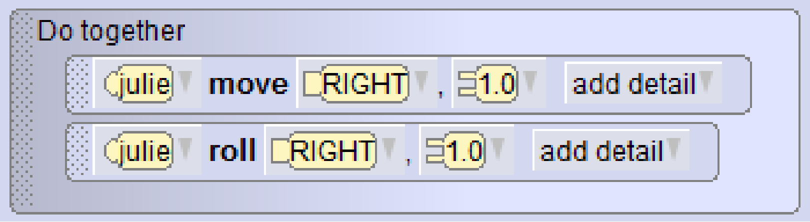
Learning From Unfamiliar Code



Find Program

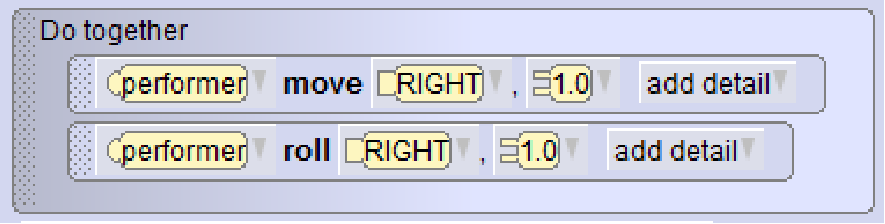
Select Snippet

Remix Snippet



```
Do together
    julie move RIGHT , 1.0 add detail
    julie roll RIGHT , 1.0 add detail
```

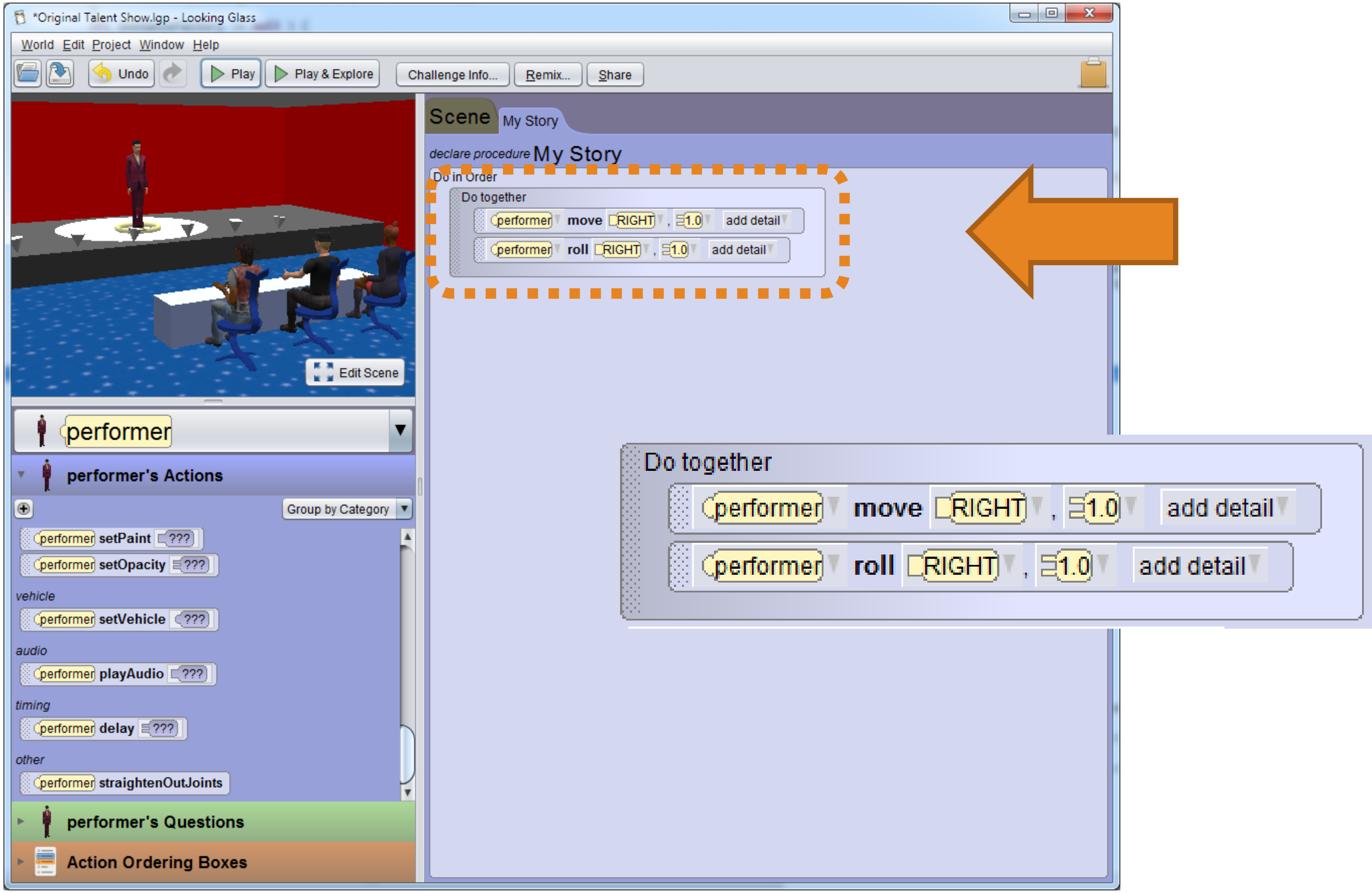
Code Snippet



```
Do together
    performer move RIGHT , 1.0 add detail
    performer roll RIGHT , 1.0 add detail
```

Remixed Code Snippet

Snippet Copied Into Program



declare procedure My Story

Do in Order

Do together

performer move RIGHT , 1.0 add detail

performer roll RIGHT , 1.0 add detail

performer setPaint ???

performer setOpacity ???

vehicle

performer setVehicle ???

audio

performer playAudio ???

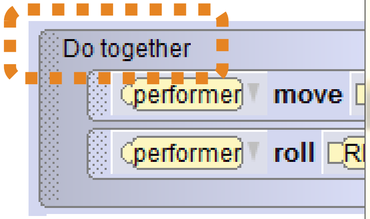
timing

performer delay ???

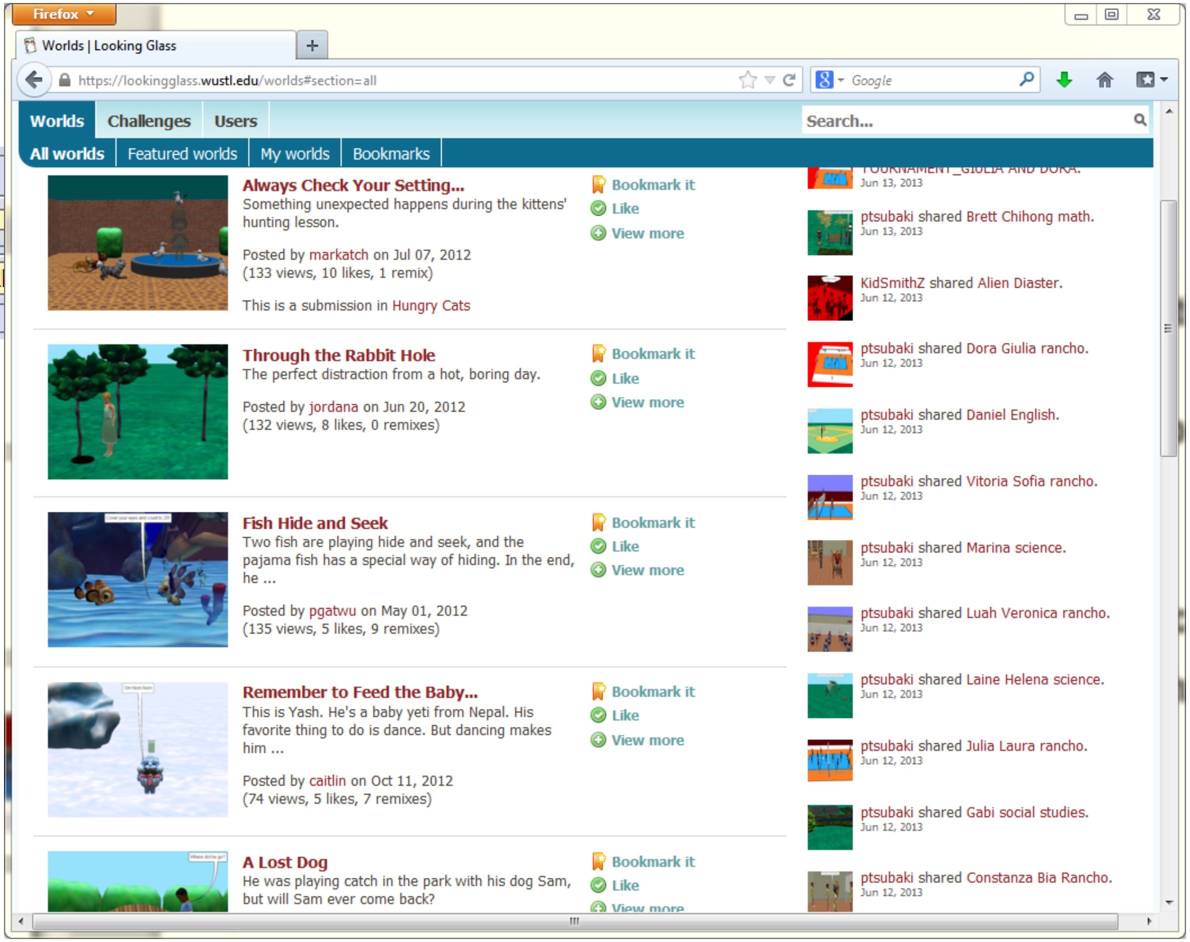
other

performer straightenOutJoints

Exposure to New Programming Concepts



```
Do together
    performer move
    performer roll RI
```



Always Check Your Setting...
Something unexpected happens during the kittens' hunting lesson.

Posted by markatch on Jul 07, 2012
(133 views, 10 likes, 1 remix)

This is a submission in Hungry Cats

Bookmark it
Like
View more

Through the Rabbit Hole
The perfect distraction from a hot, boring day.

Posted by jordana on Jun 20, 2012
(132 views, 8 likes, 0 remixes)

Bookmark it
Like
View more

Fish Hide and Seek
Two fish are playing hide and seek, and the pajama fish has a special way of hiding. In the end, he ...

Posted by pgatwu on May 01, 2012
(135 views, 5 likes, 9 remixes)

Bookmark it
Like
View more

Remember to Feed the Baby...
This is Yash. He's a baby yeti from Nepal. His favorite thing to do is dance. But dancing makes him ...

Posted by caitlin on Oct 11, 2012
(74 views, 5 likes, 7 remixes)

Bookmark it
Like
View more

A Lost Dog
He was playing catch in the park with his dog Sam, but will Sam ever come back?

Bookmark it
Like
View more

TOURNAMENT_GIULIA AND DORA.
Jun 13, 2013

ptsubaki shared Brett Chihong math.
Jun 13, 2013

KidSmithZ shared Alien Diaster.
Jun 12, 2013

ptsubaki shared Dora Giulia rancho.
Jun 12, 2013

ptsubaki shared Daniel English.
Jun 12, 2013

ptsubaki shared Vitoria Sofia rancho.
Jun 12, 2013

ptsubaki shared Marina science.
Jun 12, 2013

ptsubaki shared Luah Veronica rancho.
Jun 12, 2013

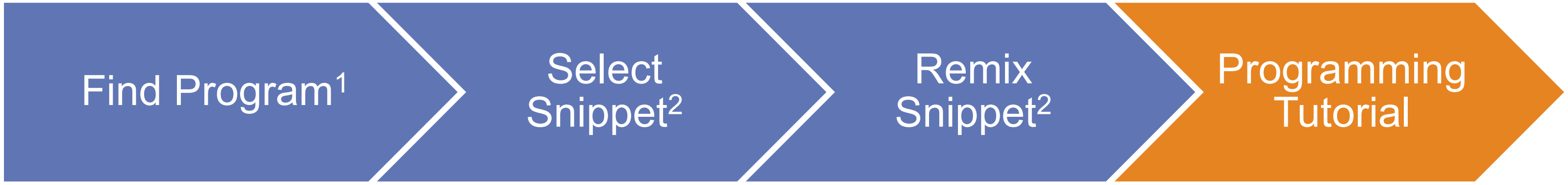
ptsubaki shared Laine Helena science.
Jun 12, 2013

ptsubaki shared Julia Laura rancho.
Jun 12, 2013

ptsubaki shared Gabi social studies.
Jun 12, 2013

ptsubaki shared Constanza Bia Rancho.
Jun 12, 2013

Independent Learning In Looking Glass


Find Program[1]

Select Snippet[2]

Remix Snippet[2]

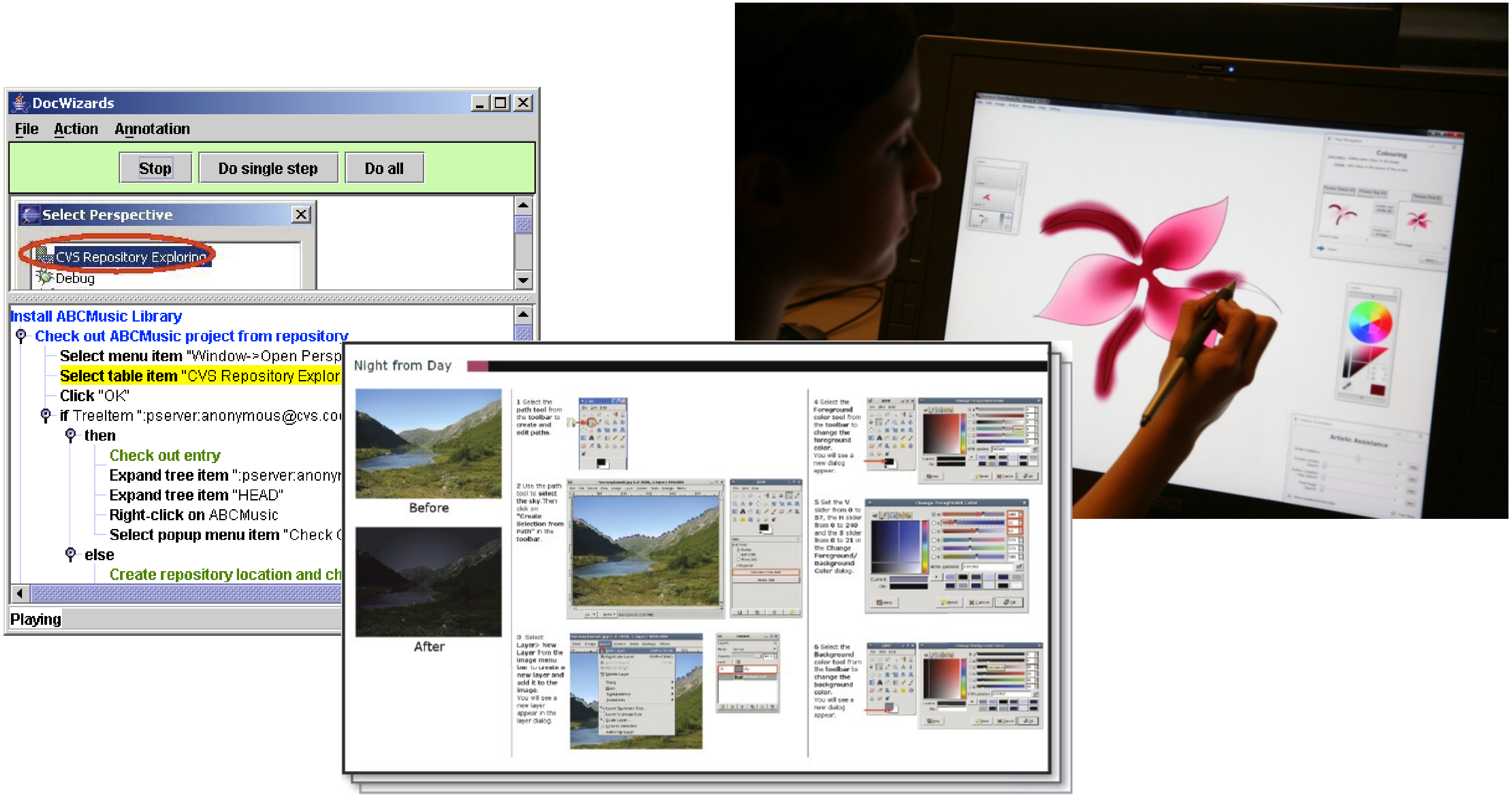
Programming Tutorial

Existing Support for Independent Learning **This Talk**

[1] Harms, K.J. et al. 2012. Designing a community to support long-term interest in programming for middle school children. *Proc. IDC*.

[2] Gross, P.A. et al. 2010. A code reuse interface for non-programmer middle school students. *Proc. IUI*.

Automatic Tutorial Generation



Install ABCMusic Library
- Check out ABCMusic project from repository
  - Select menu item "Window->Open Persp
  - Select table item "CVS Repository Explor
  - Click "OK"
  - if TreeItem ":pserver:anonymous@cvs.co
    - then
      - Check out entry
      - Expand tree item ":pserver:anonyr
      - Expand tree item "HEAD"
      - Right-click on ABCMusic
      - Select popup menu item "Check (
    - else
      - Create repository location and ch

Night from Day

Before

After

1 Select the **path tool** from the **toolbar** to **create and edit paths.**

2 Use the path tool to **select the sky.** Then click on **"Create Selection from Path"** in the **toolbar.**

3 Select **Layer> New Layer** from the **image menu bar** to **create a new layer and add it to the image.** You will see a new layer appear in the layer dialog.

4 Select the **Foreground color tool** from the **toolbar** to **change the foreground color.** You will see a new dialog appear.

5 Set the **V** slider from **0** to **57,** the **H** slider from **0** to **240** and the **S** slider from **0** to **21** in the **Change Foreground/ Background Color** dialog.

6 Select the **Background color tool** from the **toolbar** to **change the background color.** You will see a new dialog appear.

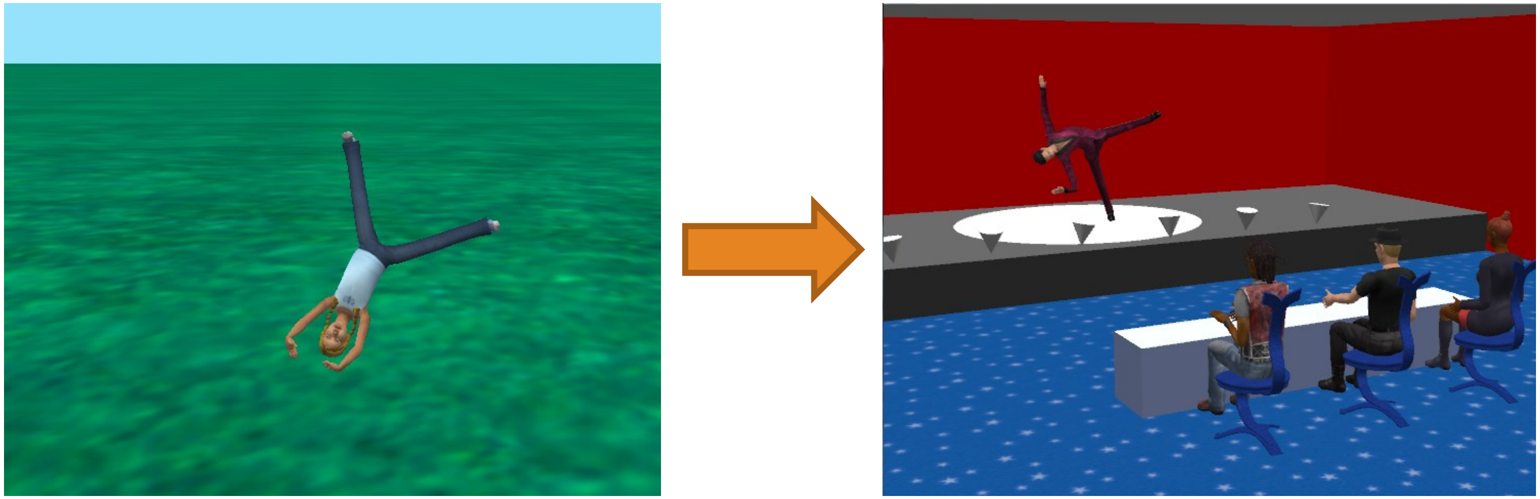
Bergman, L. et al. 2005. DocWizards: a system for authoring follow-me documentation wizards. *Proc. UIST.*
Grabler, F. et al. 2009. Generating photo manipulation tutorials by demonstration. *ACM SIGGRAPH*.
Fernquist, J. et al. 2011. Sketch-sketch revolution: an engaging tutorial system for guided sketching and application learning. *Proc. UIST.*

Current Generated Tutorial Systems

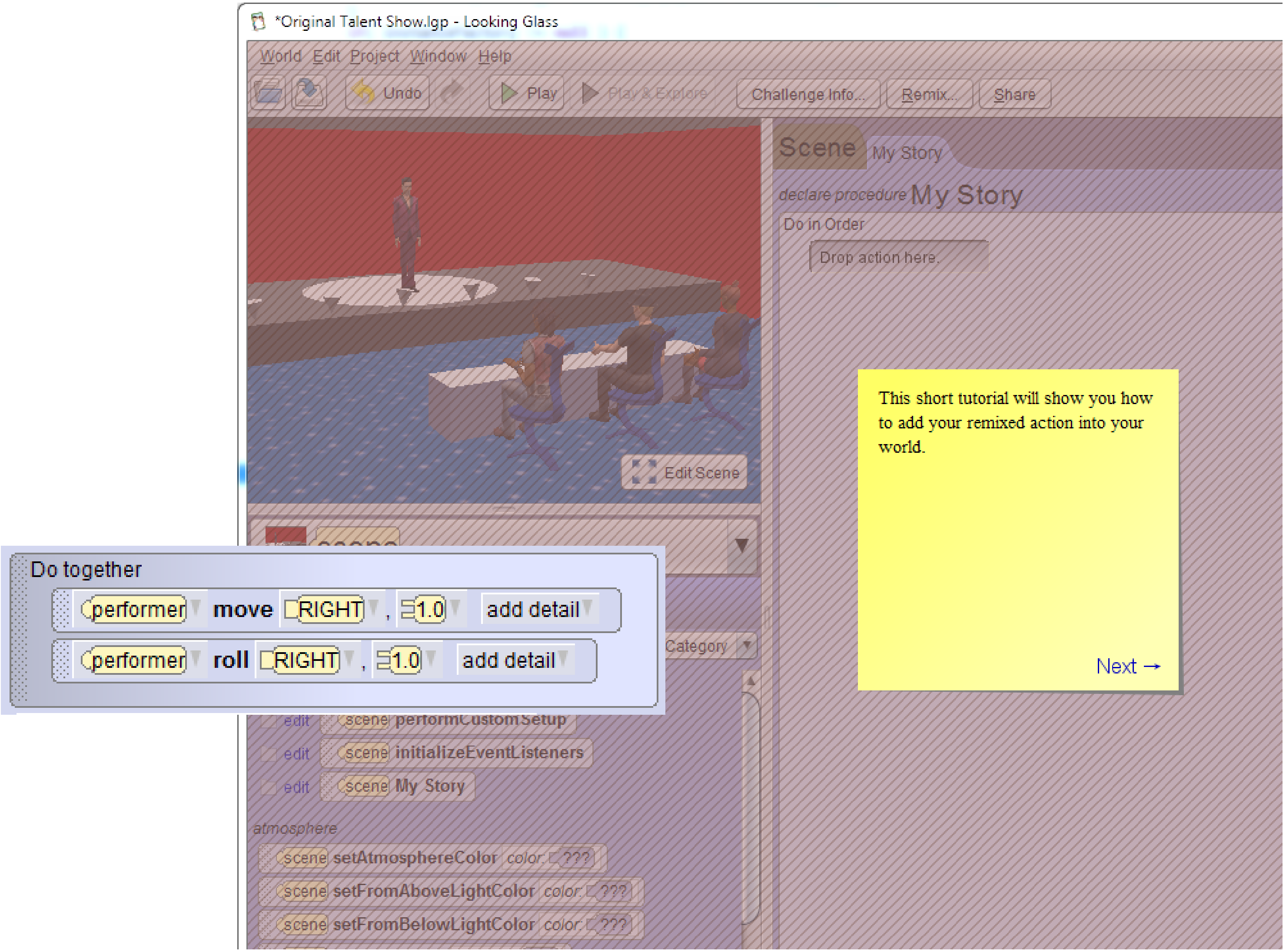
Users must adapt tutorial content to their contexts.

Require explicit authoring phase.

Users may skip steps or make mistakes.



Walk-through Tutorial to Reconstruct the Snippet

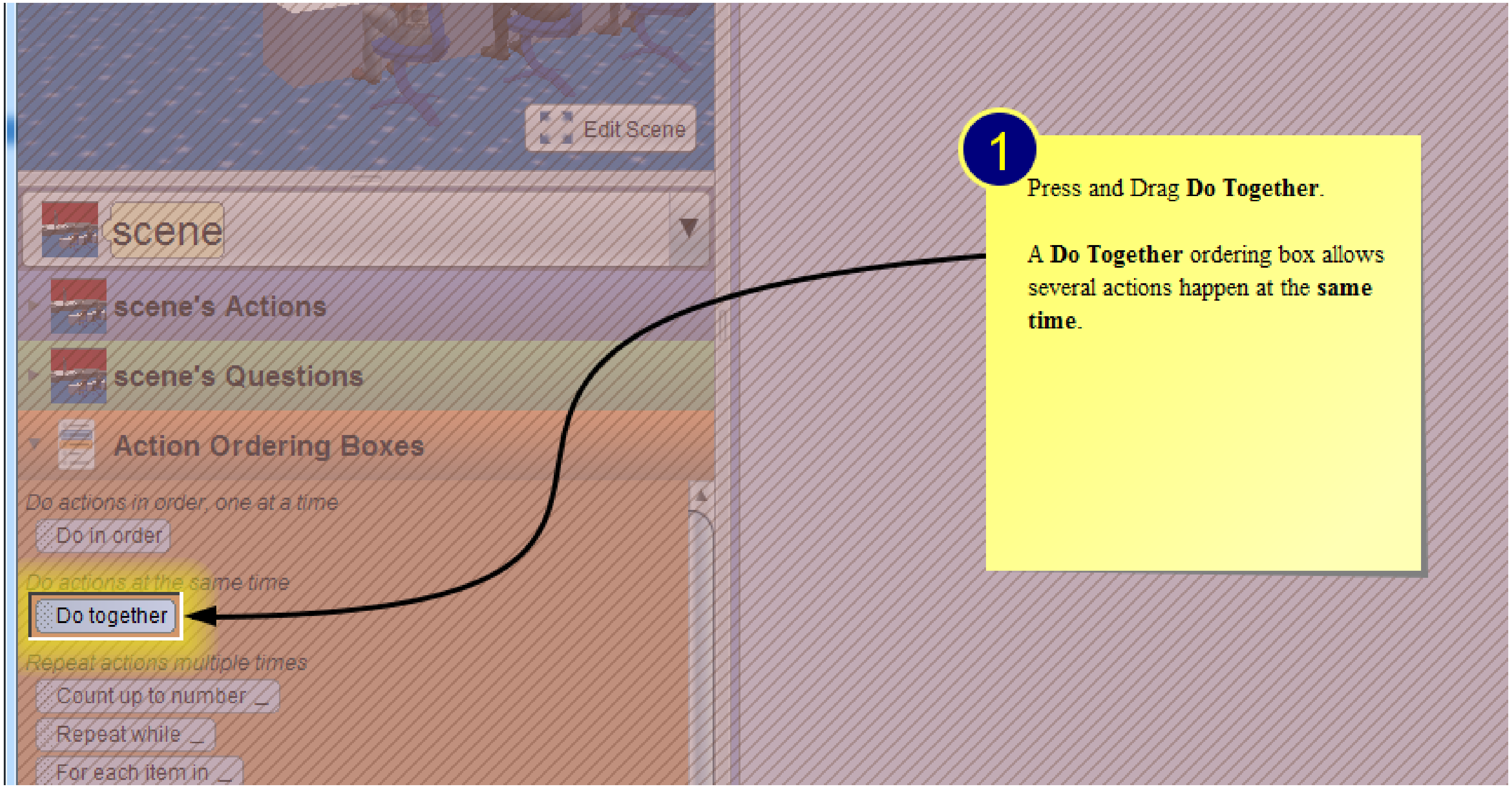

*Original Talent Show.lgp - Looking Glass
World Edit Project Window Help
Undo
Play
Play & Explore
Challenge Info...
Remix...
Share
Edit Scene
Scene
My Story
declare procedure My Story
Do in Order
Drop action here.
Category
edit (scene) performCustomSetup
edit (scene) initializeEventListeners
edit (scene) My Story
atmosphere
(scene) setAtmosphereColor color: ???
(scene) setFromAboveLightColor color: ???
(scene) setFromBelowLightColor color: ???

```
Do together
    performer move RIGHT , 1.0 add detail
    performer roll RIGHT , 1.0 add detail
```

This short tutorial will show you how to add your remixed action into your world.

Next →

Interactive Stencils Tutorial Interface

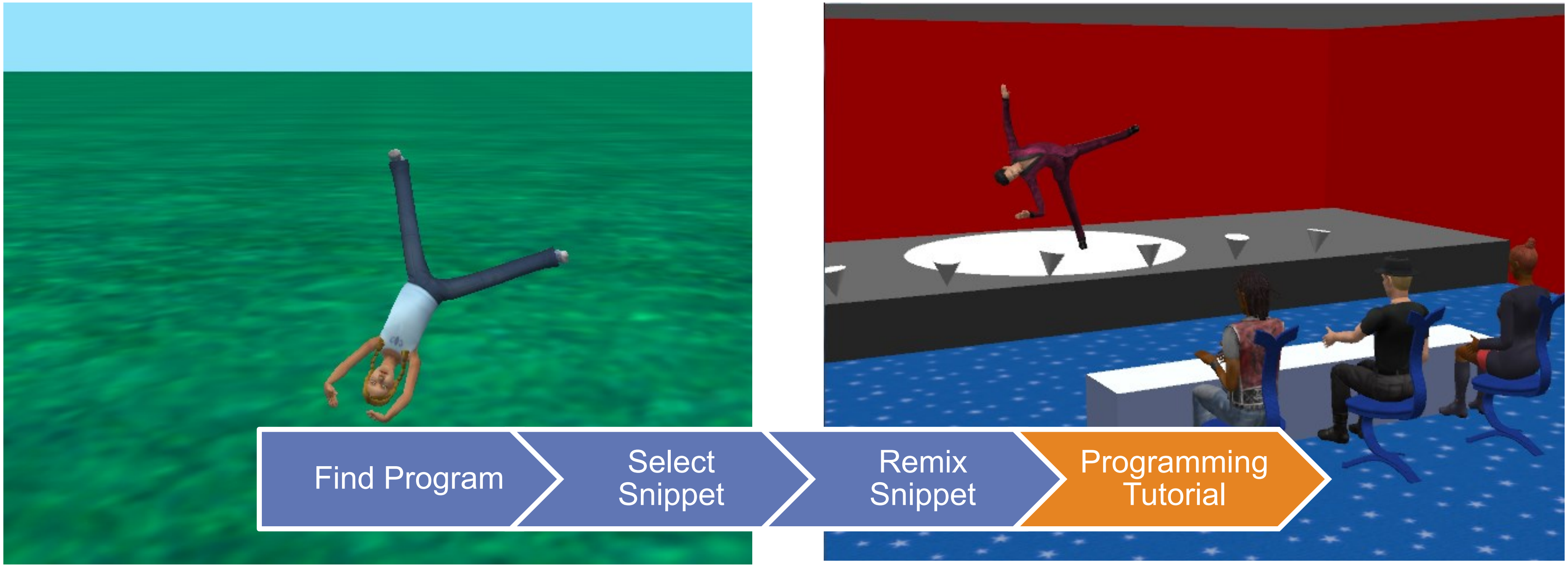


Press and Drag **Do Together**.

A **Do Together** ordering box allows several actions happen at the **same time**.

Harms, K.J. et al. 2011. Improving learning transfer from stencils-based tutorials. *Proc. IDC*.

Programming Tutorial

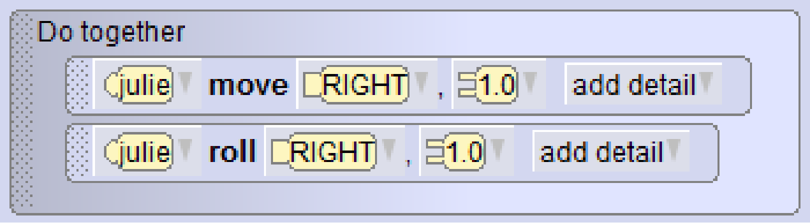


Find Program

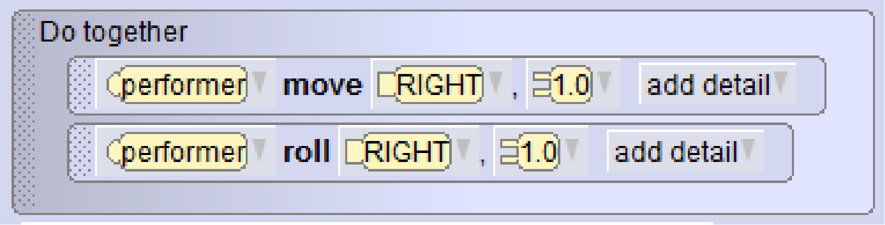
Select Snippet

Remix Snippet

Programming Tutorial



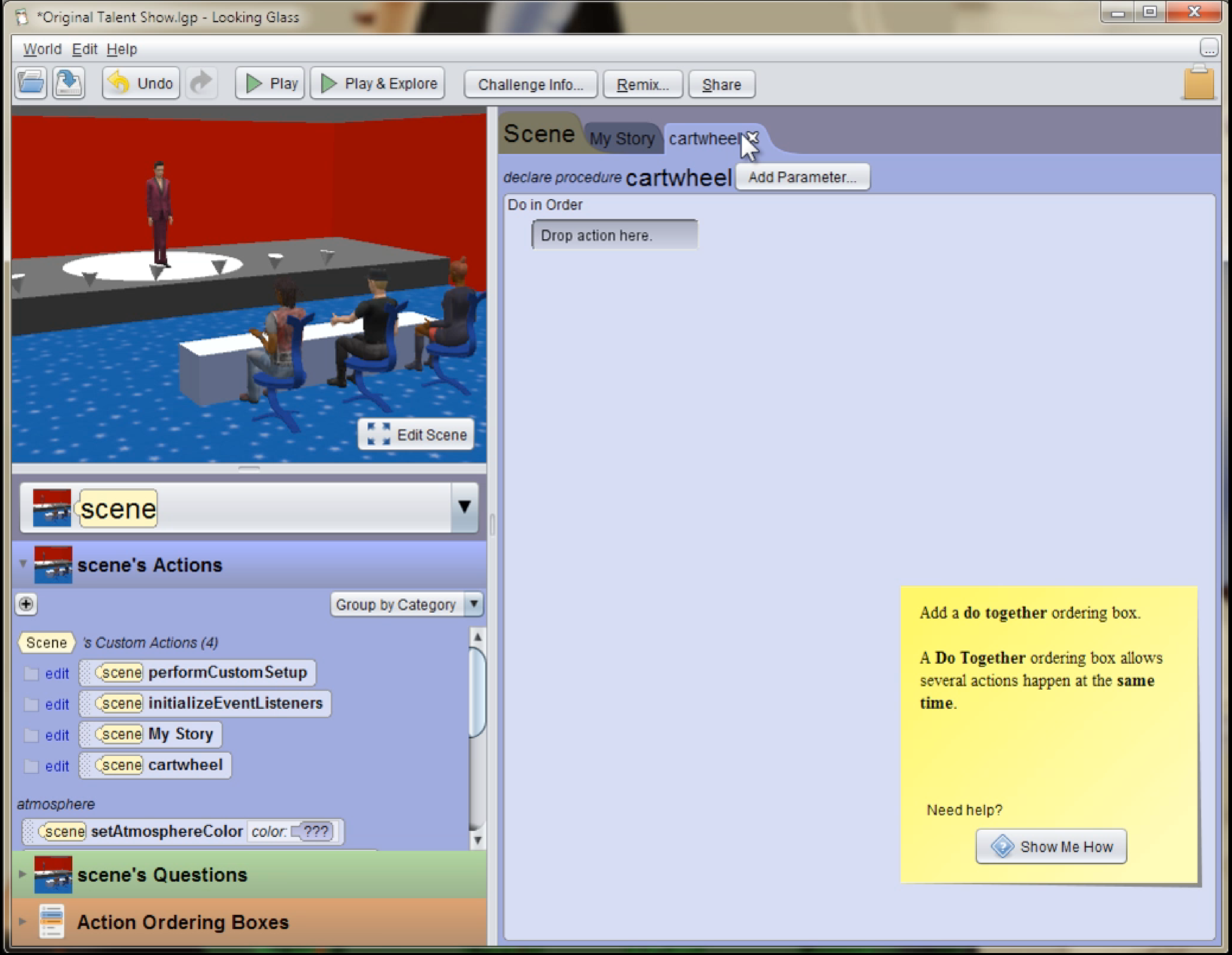
```
Do together
    julie move RIGHT , 1.0 add detail
    julie roll RIGHT , 1.0 add detail
```



```
Do together
    performer move RIGHT , 1.0 add detail
    performer roll RIGHT , 1.0 add detail
```

Code Snippet

Remixed Code Snippet



*Original Talent Show.lgp - Looking Glass

World Edit Help

Undo Play Play & Explore Challenge Info... Remix... Share

Scene My Story cartwheel

declare procedure cartwheel Add Parameter...

Do in Order

Drop action here.

Edit Scene

scene

scene's Actions

Group by Category

Scene *'s Custom Actions (4)*

edit scene performCustomSetup

edit scene initializeEventListeners

edit scene My Story

edit scene cartwheel

atmosphere

scene setAtmosphereColor *color:* ???

scene's Questions

Action Ordering Boxes

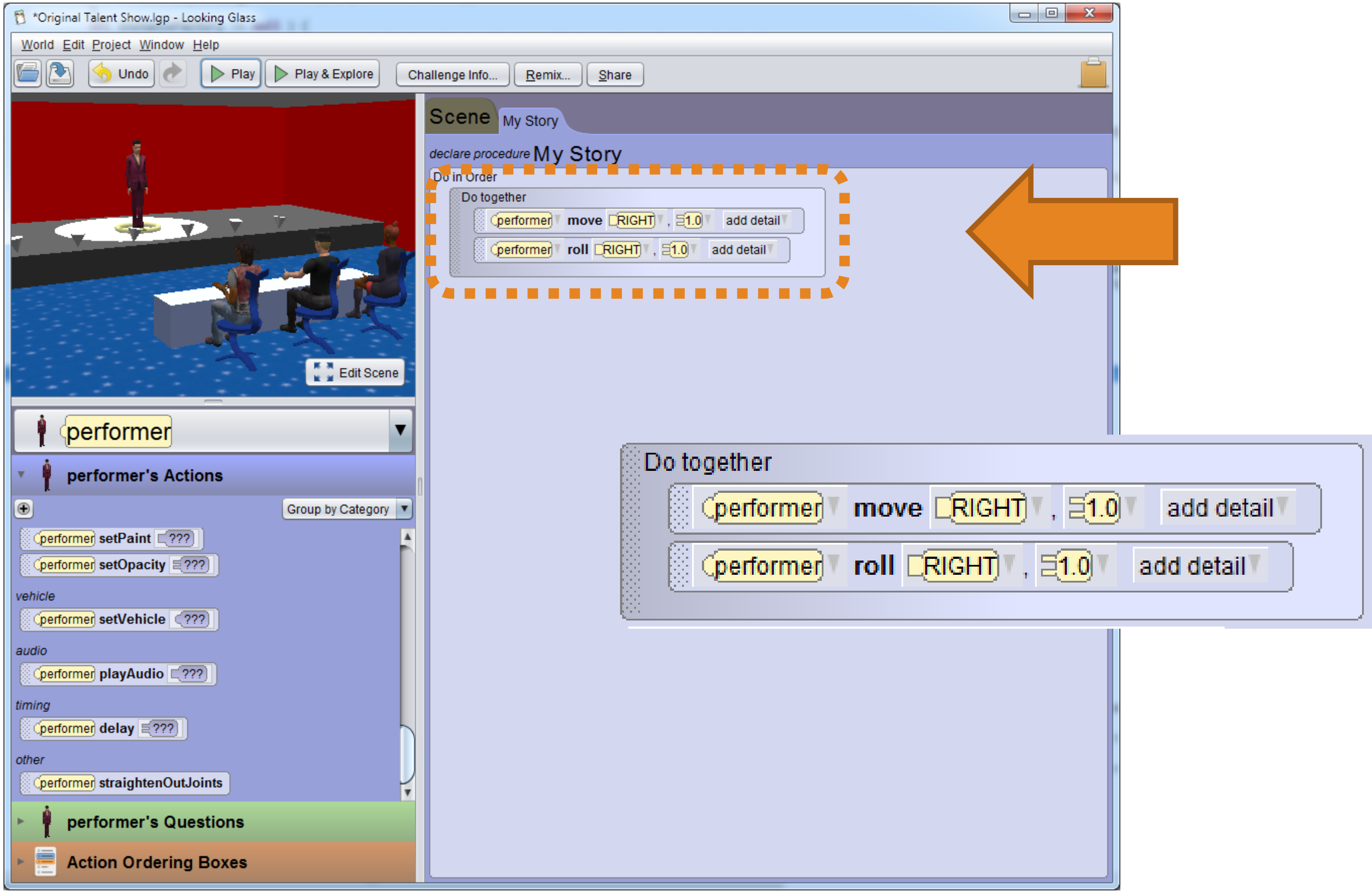
Add a **do together** ordering box.

A **Do Together** ordering box allows several actions happen at the **same time**.

Need help?

Show Me How

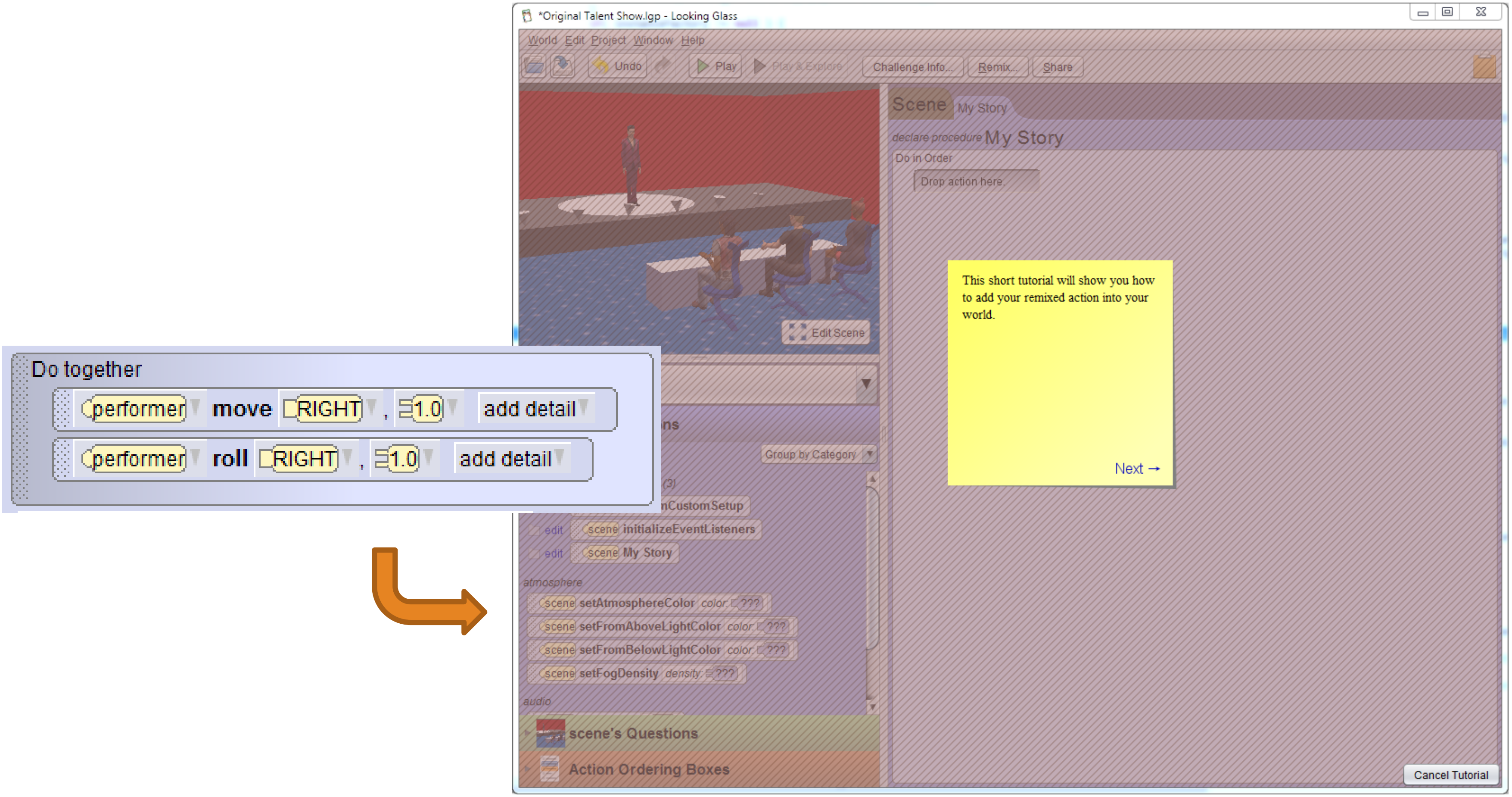
Snippet Reconstructed Through Walkthrough Tutorial



```
declare procedure My Story
Do in Order
  Do together
    performer move RIGHT , 1.0 add detail
    performer roll RIGHT , 1.0 add detail
```

```
Do together
  performer move RIGHT , 1.0 add detail
  performer roll RIGHT , 1.0 add detail
```

Generating Walkthrough Tutorials from Code Snippets



```
Do together
    performer move RIGHT , 1.0 add detail
    performer roll RIGHT , 1.0 add detail
```

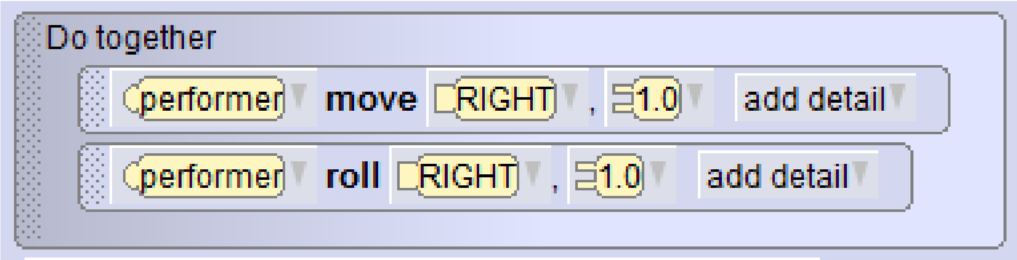
This short tutorial will show you how to add your remixed action into your world.

Next →

*Original Talent Show.lgp - Looking Glass
World Edit Project Window Help
Undo
Play
Play & Explore
Challenge Info...
Remix...
Share
Edit Scene
Scene
My Story
declare procedure My Story
Do in Order
Drop action here.
Group by Category
(3)
nCustomSetup
edit
scene initializeEventListeners
edit
scene My Story
atmosphere
scene setAtmosphereColor color ???
scene setFromAboveLightColor color ???
scene setFromBelowLightColor color ???
scene setFogDensity density ???
audio
scene's Questions
Action Ordering Boxes
Cancel Tutorial

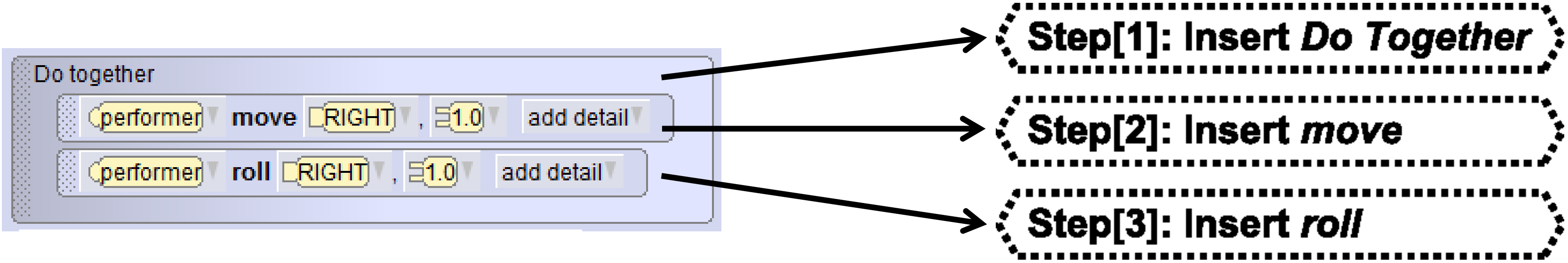
Reconstructing a Code Snippet

1. Insert a Do Together statement.
2. Insert move statement into the Do Together.
3. Insert roll statement into the Do Together.



```
Do together
    performer move RIGHT , 1.0 add detail
    performer roll RIGHT , 1.0 add detail
```

Walkthrough Tutorial Steps



```
Do together
    performer move RIGHT , 1.0 add detail
    performer roll RIGHT , 1.0 add detail
```

Step[1]: Insert *Do Together*

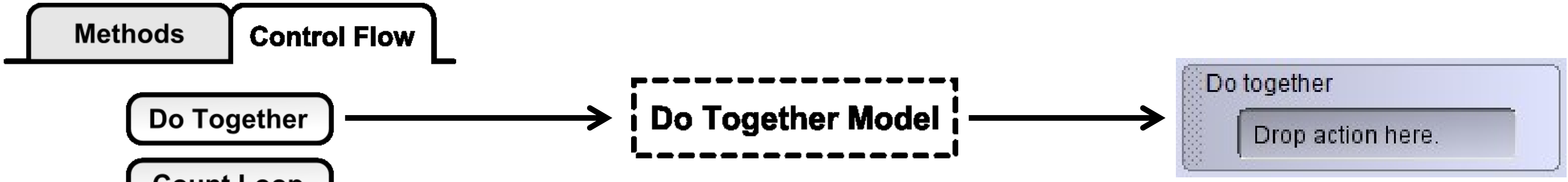
Step[2]: Insert *move*

Step[3]: Insert *roll*

Code Snippet

Tutorial Steps

Model-Driven Architecture



Methods

Control Flow

Do Together

Count Loop

Do Together Model

Do together

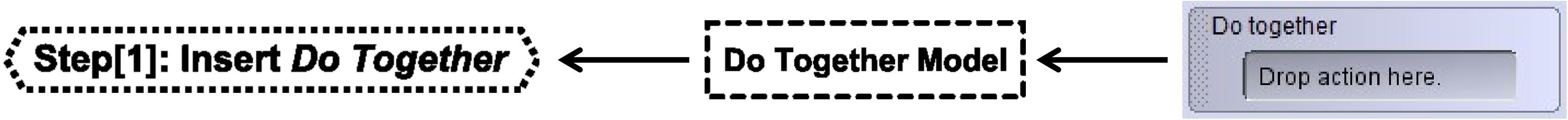
Drop action here.

On-screen Widget

Model

Data – Code Statement

Translate Code Statements into Tutorial Steps



Step[1]: Insert *Do Together*

Do Together Model

Do together

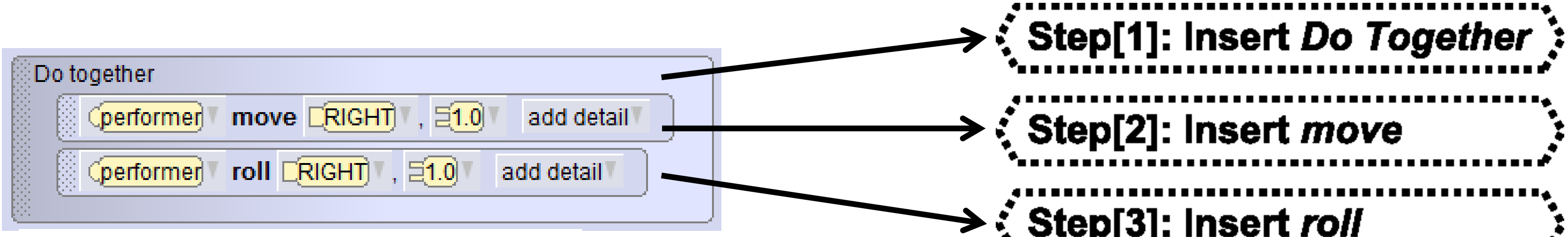
Drop action here.

Tutorial Step

Model

Code Statement

Draft Tutorial



```
Do together
    performer move RIGHT , 1.0 add detail
    performer roll RIGHT , 1.0 add detail
```

Step[1]: Insert *Do Together*

Step[2]: Insert *move*

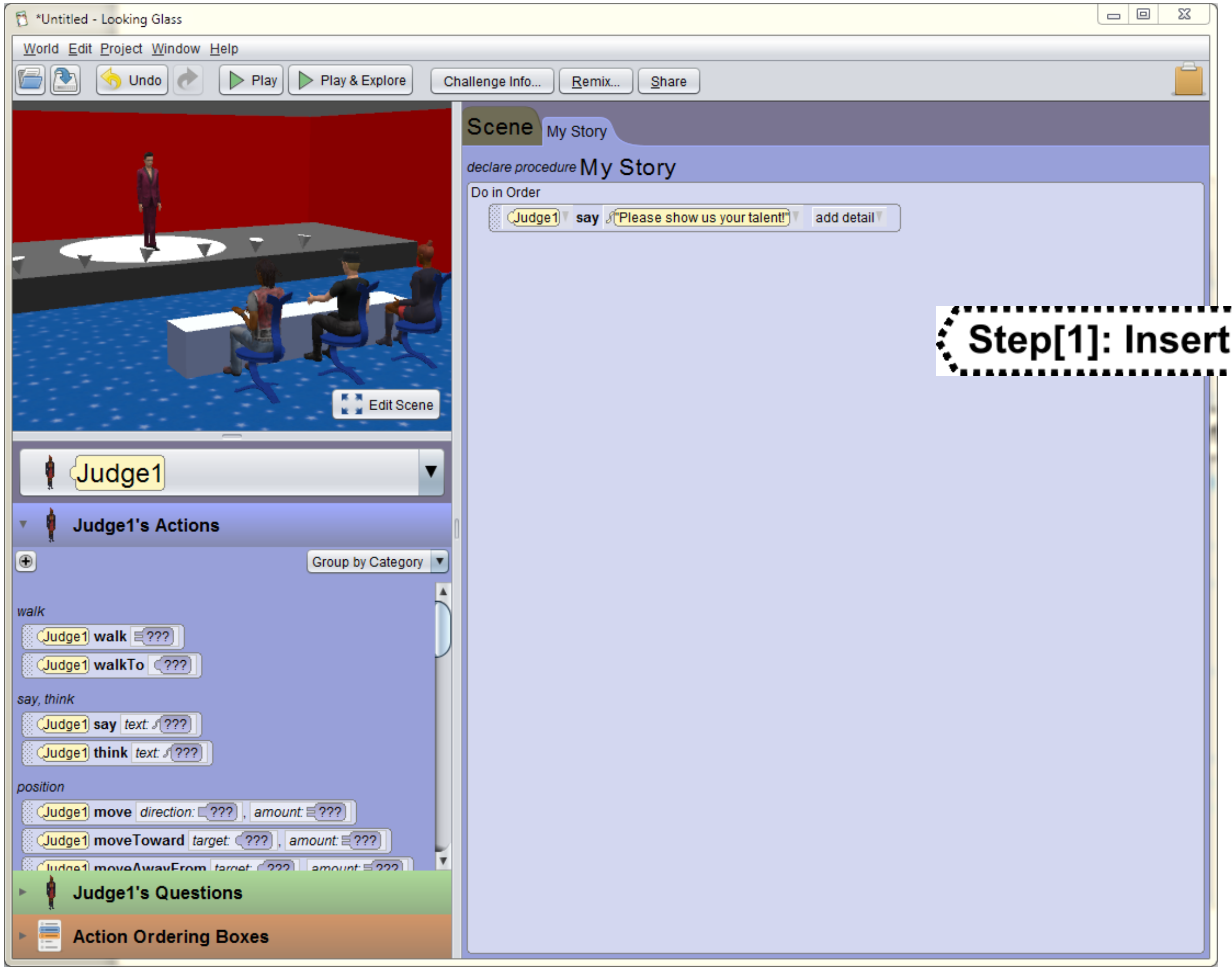
Step[3]: Insert *roll*

Code Snippet

Tutorial Steps

What if the interface is in the wrong state to complete the current step?

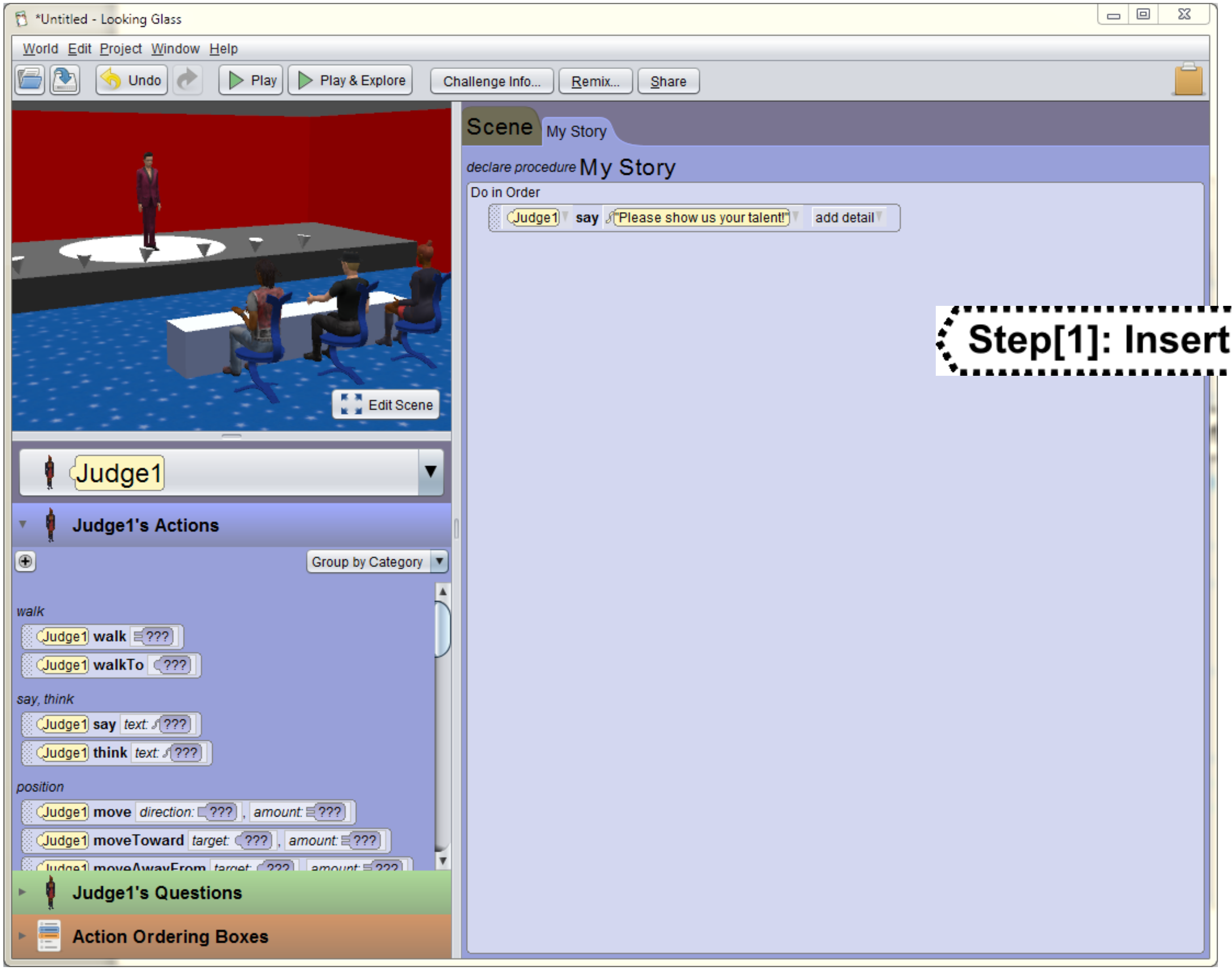
Insert Do Together



```
declare procedure My Story
Do in Order
  Judge1 say "Please show us your talent!" add detail
```

Step[1]: Insert *Do Together*

Insert Do Together

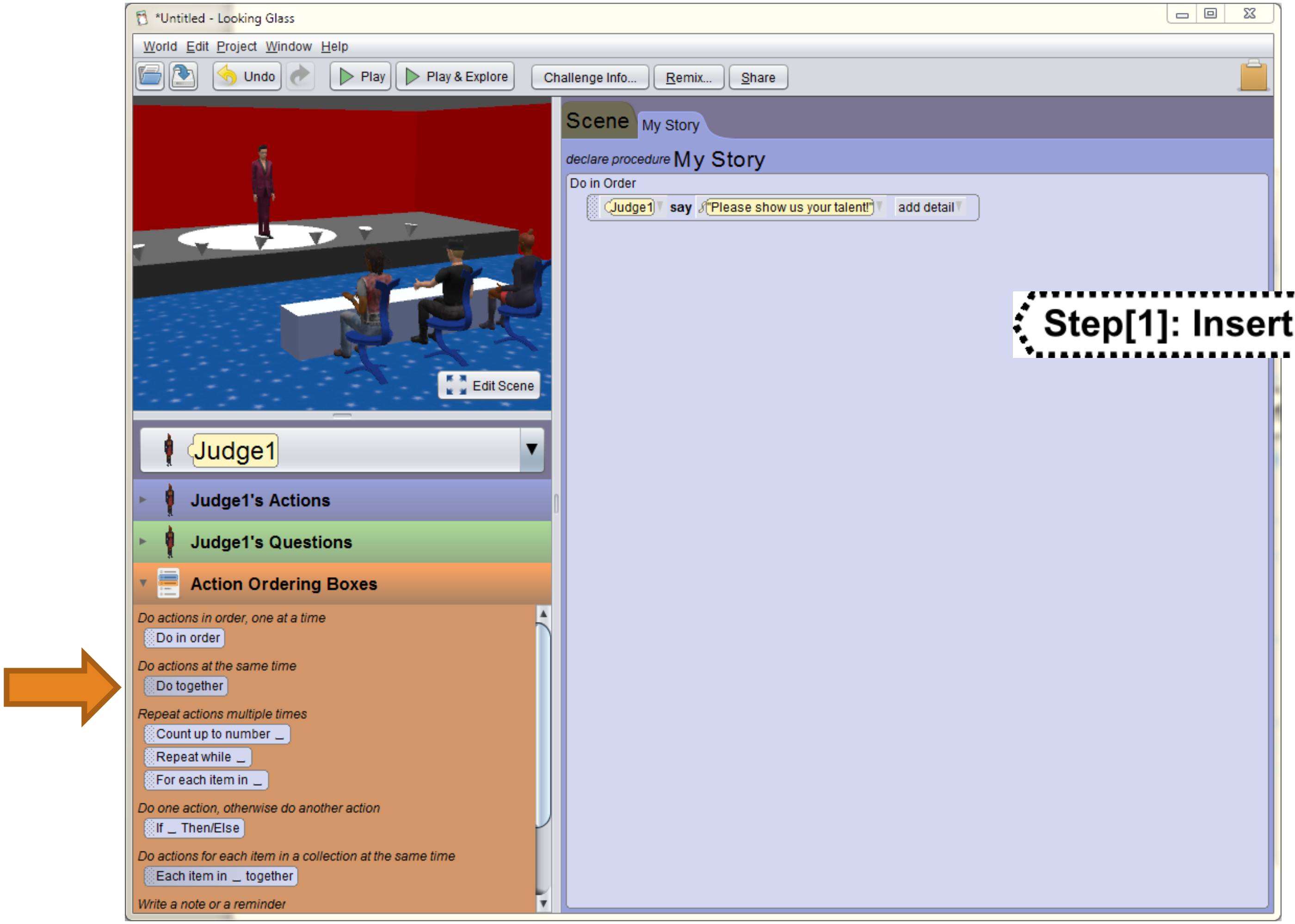


declare procedure My Story

```
Do in Order
    Judge1 say "Please show us your talent!" add detail
```

Step[1]: Insert *Do Together*

Insert Do Together



declare procedure My Story

Do in Order

Judge1 say "Please show us your talent!" add detail

Do actions in order, one at a time
Do in order

Do actions at the same time
Do together

Repeat actions multiple times
Count up to number _
Repeat while _
For each item in _

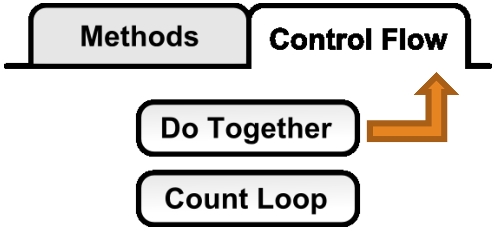
Do one action, otherwise do another action
If _ Then/Else

Do actions for each item in a collection at the same time
Each item in _ together

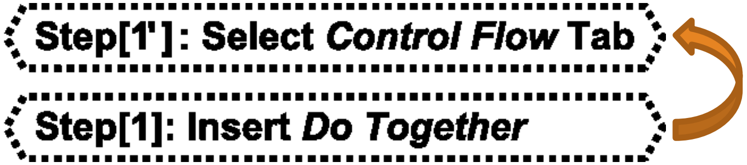
Write a note or a reminder

Step[1]: Insert *Do Together*

Tutorial Step Dependencies



On-screen Widgets



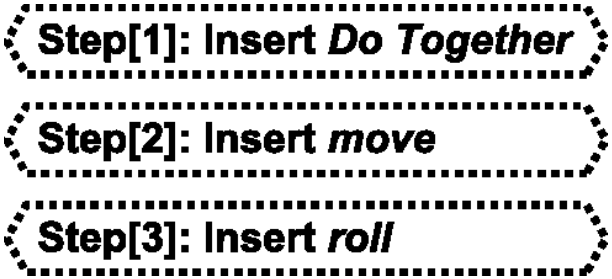
Step[1']: Select *Control Flow* Tab

Step[1]: Insert *Do Together*

Dependent Tutorial Steps

How can we present a valid tutorial to the user?

Presenting the Draft Tutorial



Step[1]: Insert *Do Together*

Step[2]: Insert *move*

Step[3]: Insert *roll*

Draft Tutorial

Check if a step's dependencies are satisfied.

Correct unsatisfied dependencies.

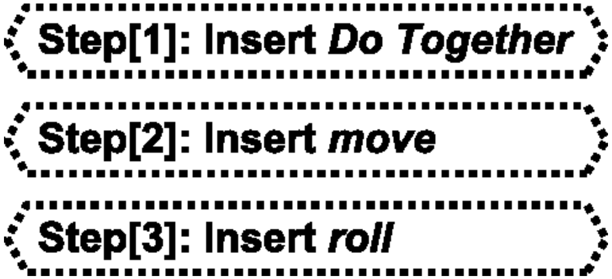
Initialize the tutorial interface for the step.

Ensure user correctly completes the step.

Algorithm for Presenting Steps

```
For each draft tutorial step do:
   If step’s dependencies are satisfied
   Then:
      Present the step to the user.
      Validate the user’s progress.
      Advance to the next step.
   Else:
      Create and insert prerequisite step.
```

Presenting the Tutorial



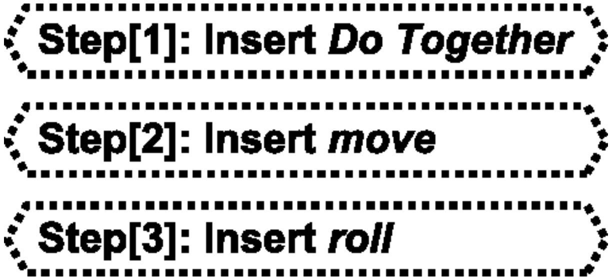
Step[1]: Insert *Do Together*

Step[2]: Insert *move*

Step[3]: Insert *roll*

```
For each draft tutorial step do:
  If the step's dependencies are satisfied
  Then:
    Present the step to the user.
    Validate the user's progress.
    Advance to the next step.
  Else:
     Create and insert a prerequisite step.
```

Check Dependencies



Step[1]: Insert *Do Together*

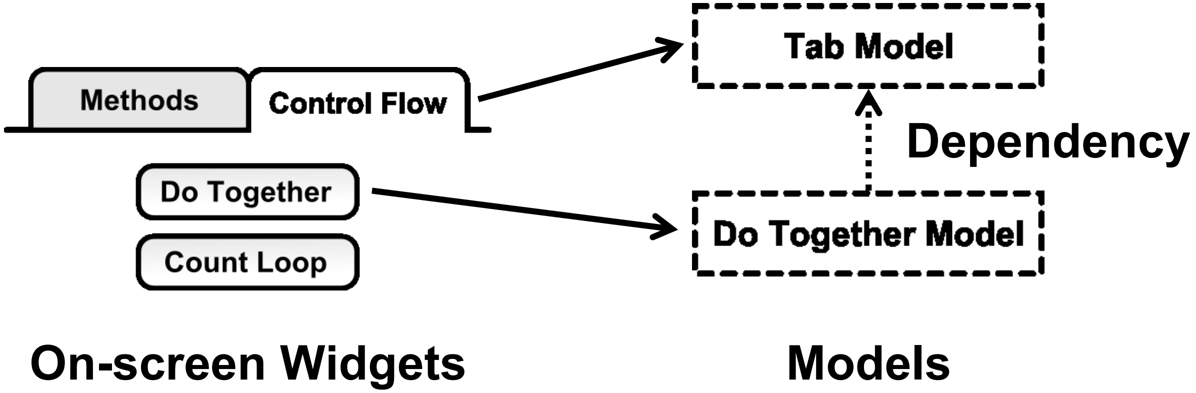
Step[2]: Insert *move*

Step[3]: Insert *roll*

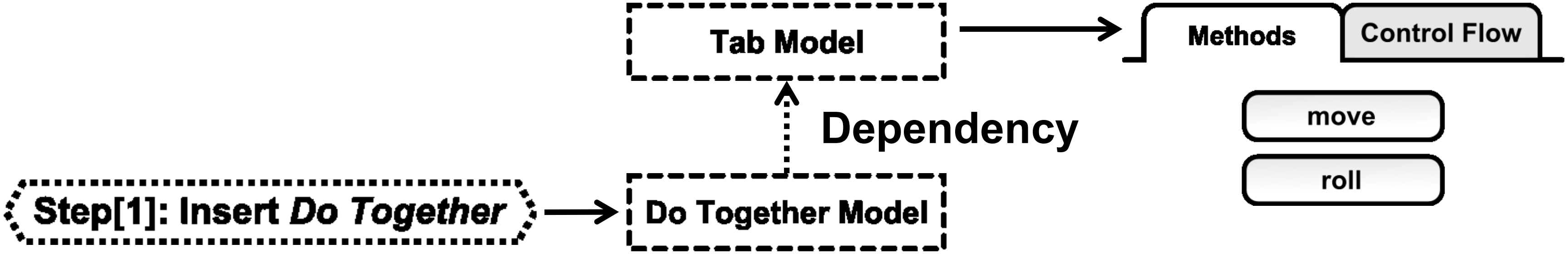
```
For each draft tutorial step do:
  If the step's dependencies are satisfied
  Then:
    Present the step to the user.
    Validate the user's progress.
    Advance to the next step.
  Else:
     Create and insert a prerequisite step.
```

Is the interface in a state where we can present this step?

Model-Driven Architecture + Dependencies



Check Dependencies

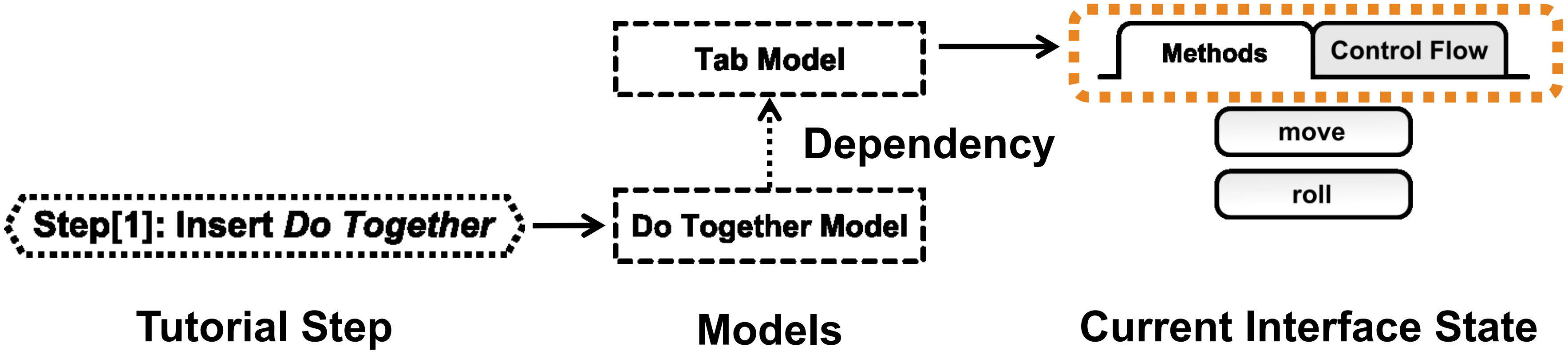


Tutorial Step

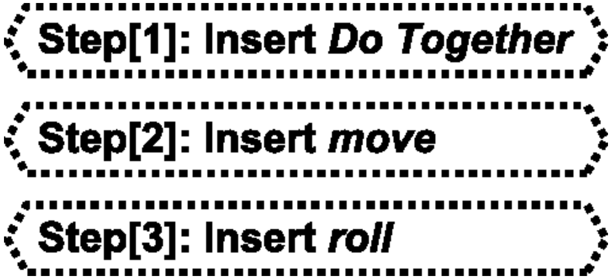
Models

Current Interface State

Check Dependencies



Check Dependencies



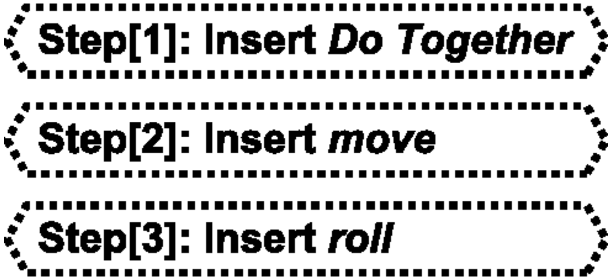
Step[1]: Insert *Do Together*

Step[2]: Insert *move*

Step[3]: Insert *roll*

```
For each draft tutorial step do:
  ~~If the step's dependencies are satisfied~~
  Then:
    Present the step to the user.
    Validate the user's progress.
    Advance to the next step.
  Else:
     Create and insert a prerequisite step.
```

Insert Prerequisite Step



Step[1]: Insert *Do Together*

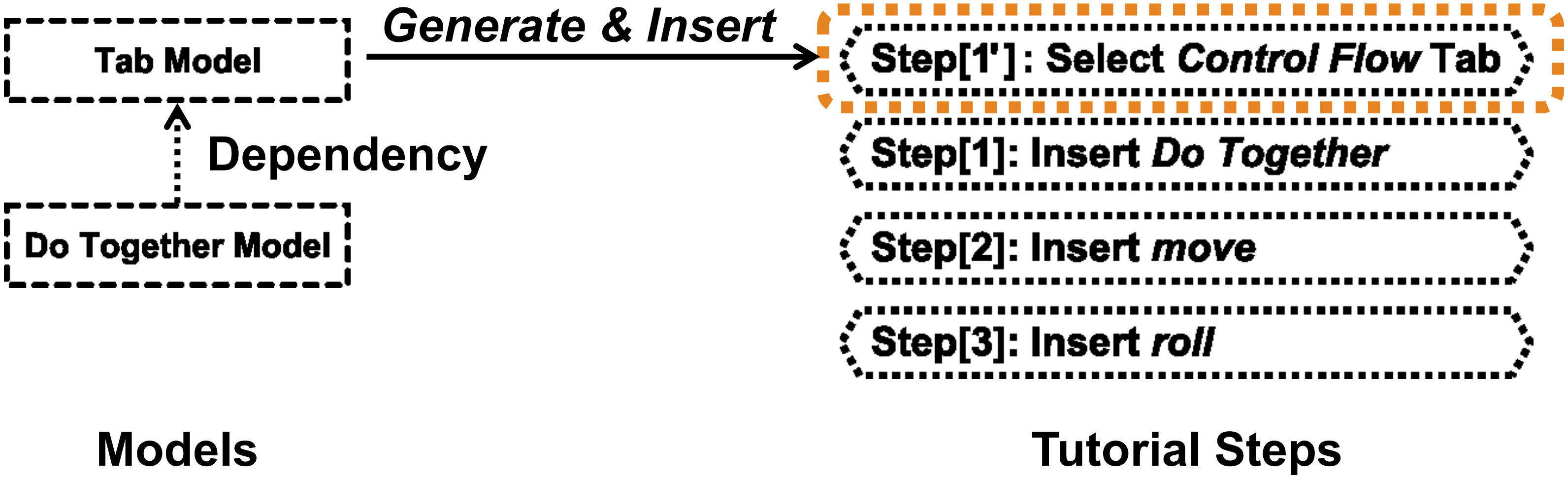
Step[2]: Insert *move*

Step[3]: Insert *roll*

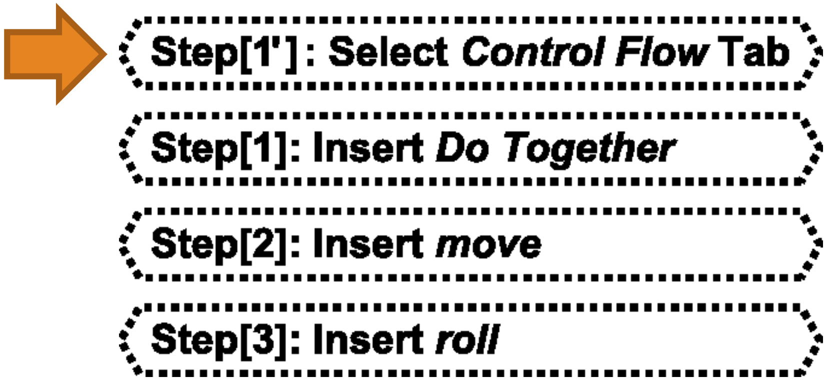
```
For each draft tutorial step do:
  ~~If the step's dependencies are satisfied~~
  Then:
    Present the step to the user.
    Validate the user's progress.
    Advance to the next step.
  Else:
     Create and insert a prerequisite step.
```

How do we adapt the tutorial to put the interface in the correct state?

Model-Driven Architecture + Insert Prerequisite Step



Present Prerequisite Step



Step[1']: Select *Control Flow* Tab

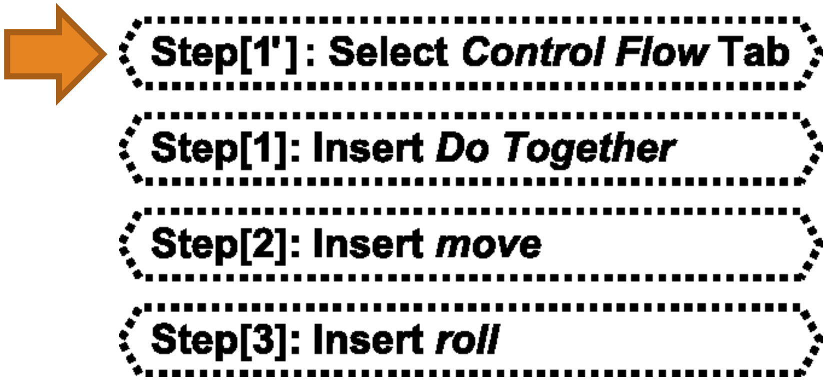
Step[1]: Insert *Do Together*

Step[2]: Insert *move*

Step[3]: Insert *roll*

```
For each draft tutorial step do:
  If the step's dependencies are satisfied
  Then:
    Present the step to the user.
    Validate the user's progress.
    Advance to the next step.
  Else:
     Create and insert a prerequisite step.
```

Check Dependencies



Step[1']: Select *Control Flow* Tab

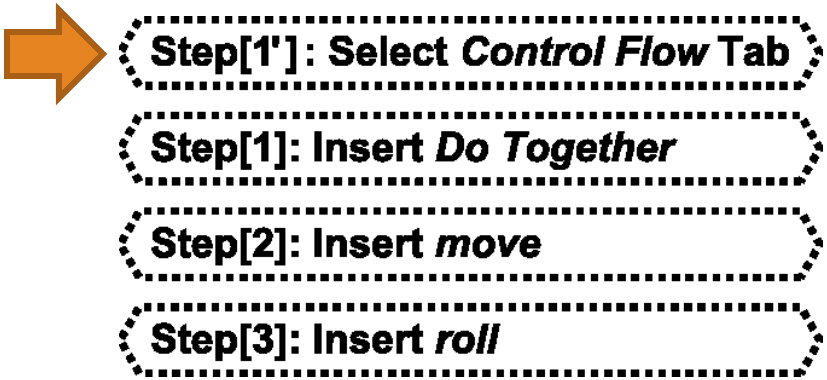
Step[1]: Insert *Do Together*

Step[2]: Insert *move*

Step[3]: Insert *roll*

```
For each draft tutorial step do:
  If the step’s dependencies are satisfied
  Then:
    Present the step to the user.
    Validate the user’s progress.
    Advance to the next step.
  Else:
     Create and insert a prerequisite step.
```

Present Step



Step[1']: Select *Control Flow* Tab

Step[1]: Insert *Do Together*

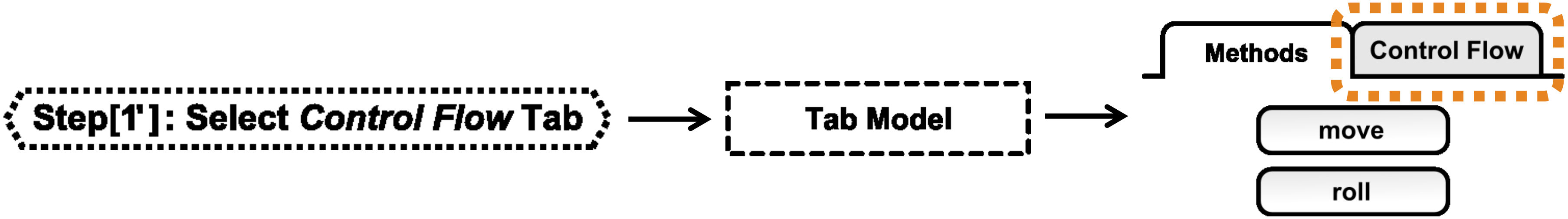
Step[2]: Insert *move*

Step[3]: Insert *roll*

```
For each draft tutorial step do:
  If the step's dependencies are satisfied
  Then:
    Present the step to the user.
    Validate the user's progress.
    Advance to the next step.
  Else:
     Create and insert a prerequisite step.
```

How do we present the step to the user?

Model-Driven Architecture + Present Tutorial Step



Step[1']: Select *Control Flow* Tab

Tab Model

Methods

Control Flow

move

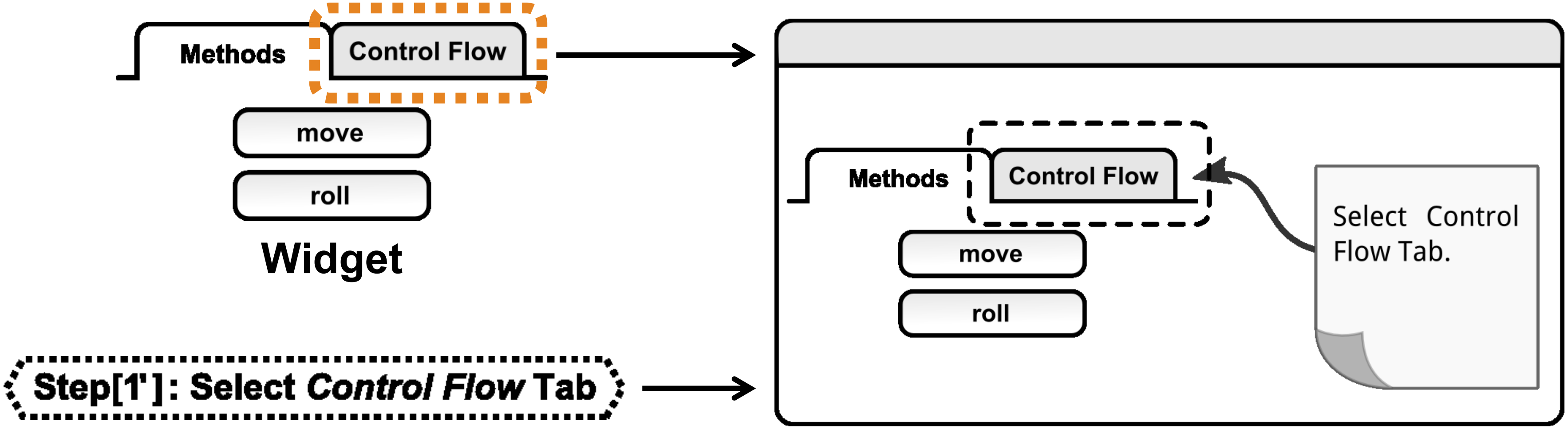
roll

Tutorial Step

Model

Visible Widget

Present Step with Stencils

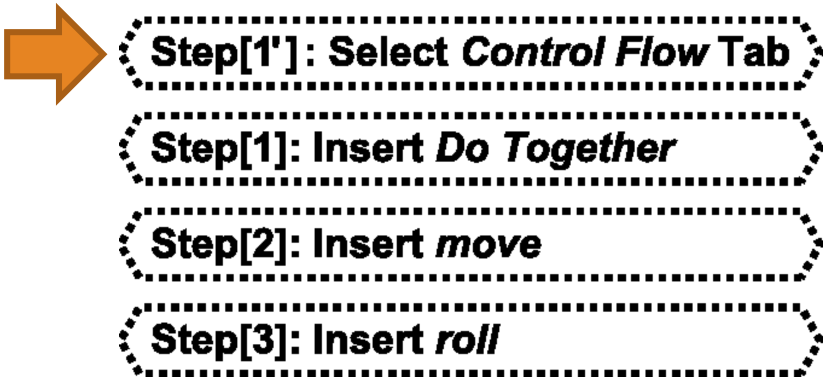


Widget

Tutorial Step

Stencils-Based Interface

Validate User’s Progress



Step[1']: Select *Control Flow* Tab

Step[1]: Insert *Do Together*

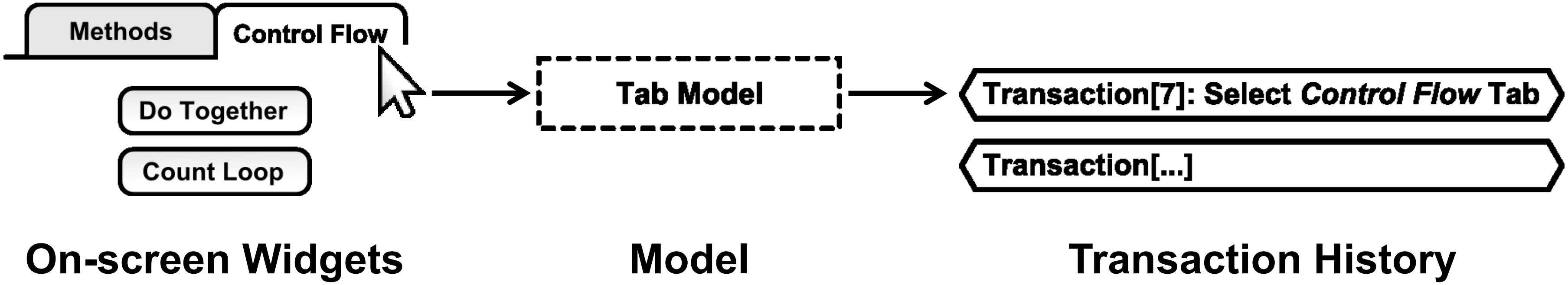
Step[2]: Insert *move*

Step[3]: Insert *roll*

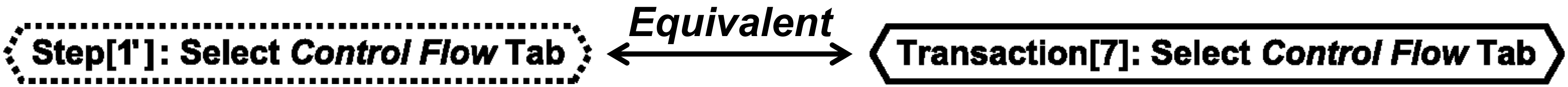
```
For each draft tutorial step do:
  If the step’s dependencies are satisfied
  Then:
    Present the step to the user.
    Validate the user’s progress.
    Advance to the next step.
  Else:
     Create and insert a prerequisite step.
```

How do we prevent mistakes from derailing the tutorial?

Model-Driven Architecture + Record User's Actions



Validating the User’s Progress



Step[1']: Select *Control Flow* Tab

Equivalent

Transaction[7]: Select *Control Flow* Tab

Current Tutorial Step

Latest Recorded Transaction

Advance to Next Step

Step[1']: Select *Control Flow* Tab

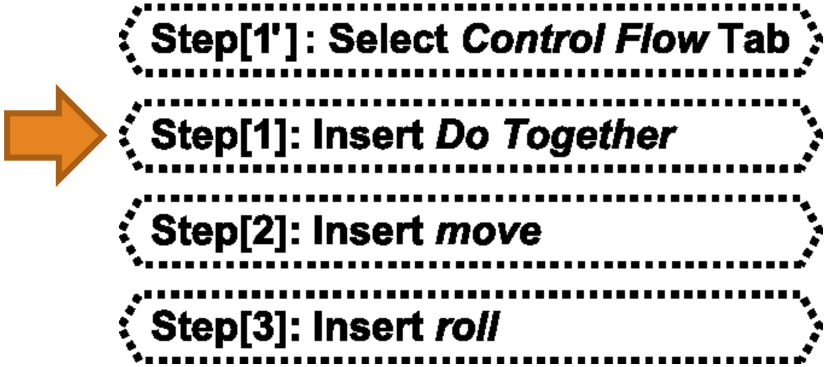
Step[1]: Insert *Do Together*

Step[2]: Insert *move*

Step[3]: Insert *roll*

```
For each draft tutorial step do:
  If the step’s dependencies are satisfied
  Then:
    Present the step to the user.
    Validate the user’s progress.
    Advance to the next step.
  Else:
     Create and insert a prerequisite step.
```

Advance to Next Step



Step[1']: Select *Control Flow* Tab

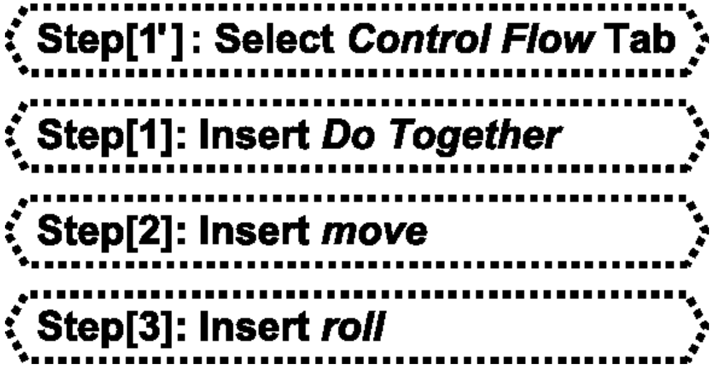
Step[1]: Insert *Do Together*

Step[2]: Insert *move*

Step[3]: Insert *roll*

```
For each draft tutorial step do:
  If the step's dependencies are satisfied
  Then:
    Present the step to the user.
    Validate the user's progress.
    Advance to the next step.
  Else:
     Create and insert a prerequisite step.
```

Check Dependencies



Step[1']: Select *Control Flow* Tab

Step[1]: Insert *Do Together*

Step[2]: Insert *move*

Step[3]: Insert *roll*

```
For each draft tutorial step do:
 If the step's dependencies are satisfied
 Then:
  Present the step to the user.
  Validate the user's progress.
  Advance to the next step.
 Else:
   Create and insert a prerequisite step.
```

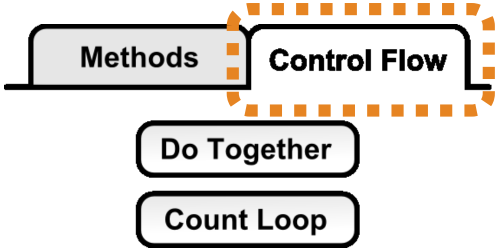
Check Dependencies

Step[1']: Select *Control Flow* Tab

Step[1]: Insert *Do Together*

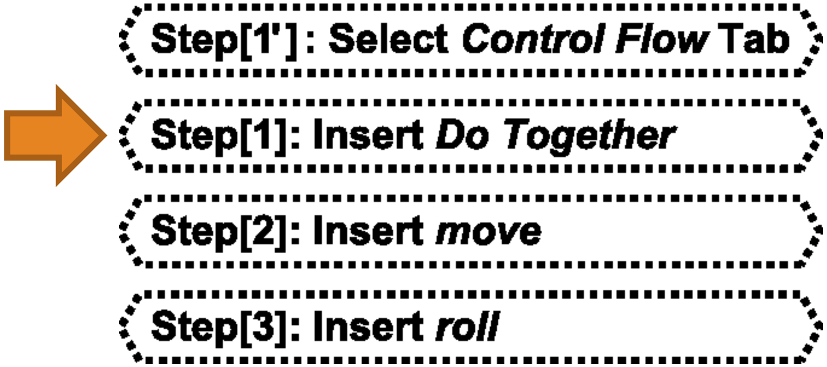
Step[2]: Insert *move*

Step[3]: Insert *roll*



```
For each draft tutorial step do:
  If the step's dependencies are satisfied
  Then:
   Present the step to the user.
   Validate the user's progress.
   Advance to the next step.
  Else:
    Create and insert a prerequisite step.
```

Present Step



Step[1']: Select *Control Flow* Tab

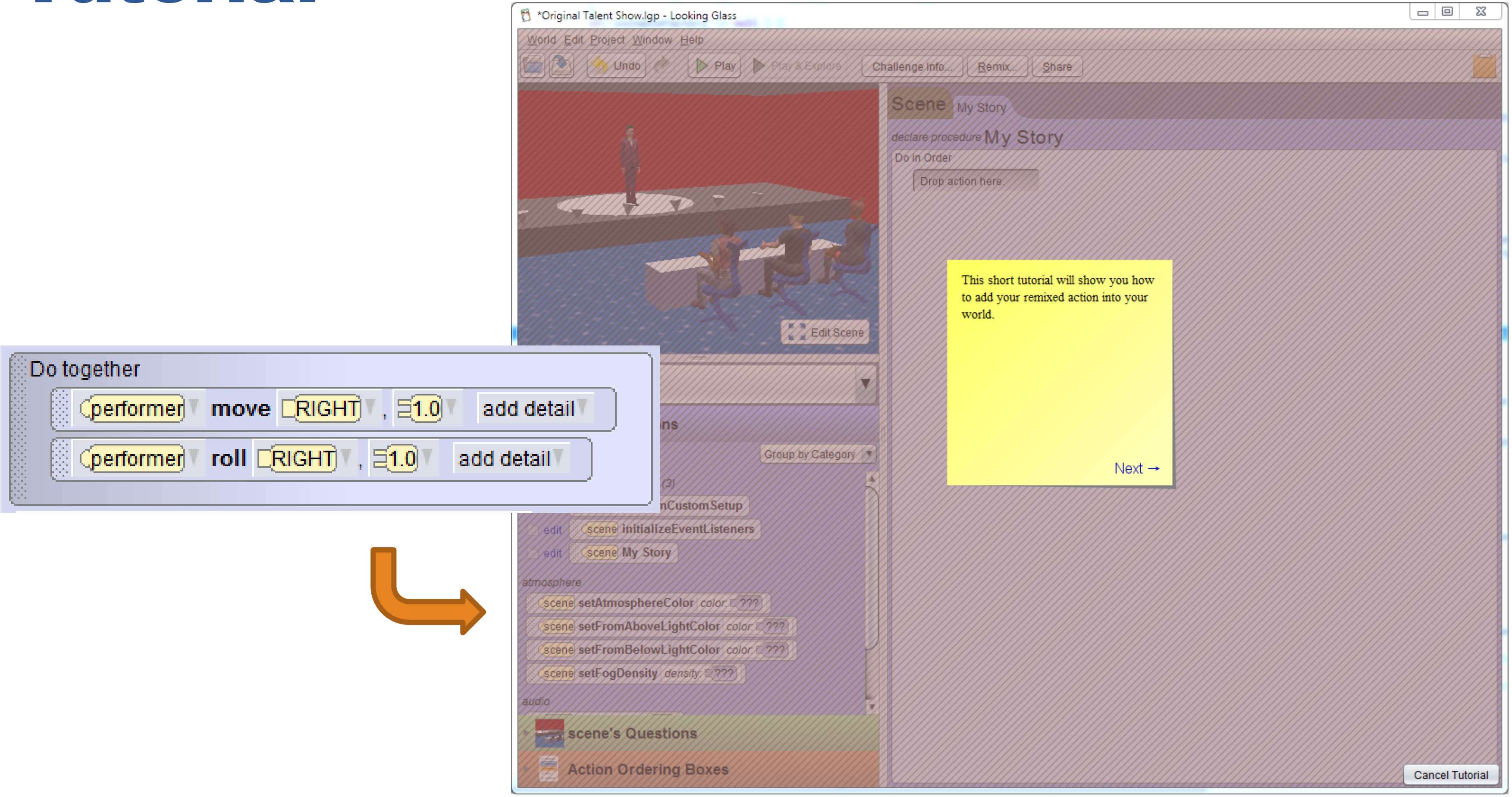
Step[1]: Insert *Do Together*

Step[2]: Insert *move*

Step[3]: Insert *roll*

```
For each draft tutorial step do:
  If the step’s dependencies are satisfied
  Then:
    Present the step to the user.
    Validate the user’s progress.
    Advance to the next step.
  Else:
    Create and insert a prerequisite step.
```

Automatically Generated Walkthrough Programming Tutorial



```
Do together
    performer move RIGHT , 1.0 add detail
    performer roll RIGHT , 1.0 add detail
```

This short tutorial will show you how to add your remixed action into your world.

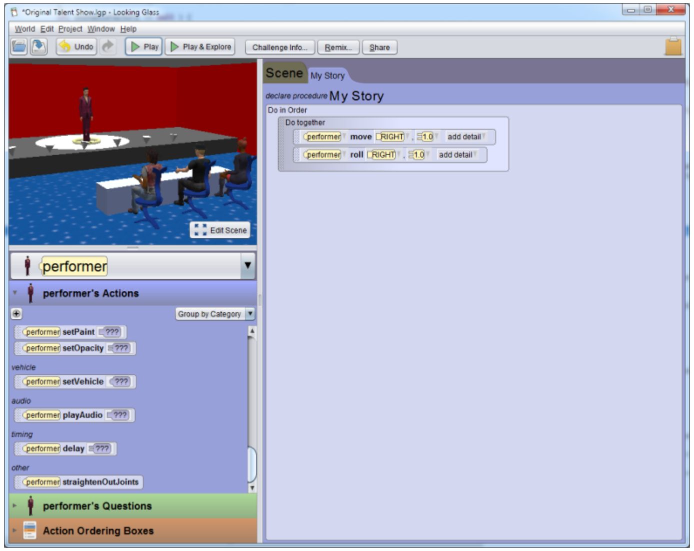
Next →

*Original Talent Show.lgp - Looking Glass
World Edit Project Window Help
Undo
Play
Play & Explore
Challenge Info...
Remix...
Share
Scene
My Story
declare procedure My Story
Do in Order
Drop action here.
Edit Scene
Group by Category
nCustomSetup
edit
scene initializeEventListeners
scene My Story
atmosphere
scene setAtmosphereColor color ???
scene setFromAboveLightColor color ???
scene setFromBelowLightColor color ???
scene setFogDensity density ???
audio
scene's Questions
Action Ordering Boxes
Cancel Tutorial

Evaluation

40 Middle school aged (10–16 years) participants

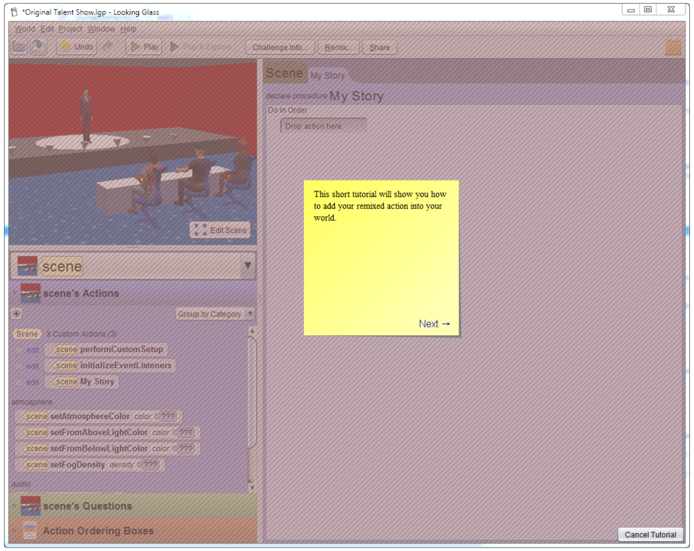
1.5 hour sessions each with no more than 5 participants



```
declare procedure My Story
Do in Order
  Do together
    performer move RIGHT , 1.0 add detail
    performer roll RIGHT , 1.0 add detail
```

```
performer setPaint ???
performer setOpacity ???
performer setVehicle ???
performer playAudio ???
performer delay ???
performer straightenOutJoints
```

Control



```
declare procedure My Story
Do in Order
```

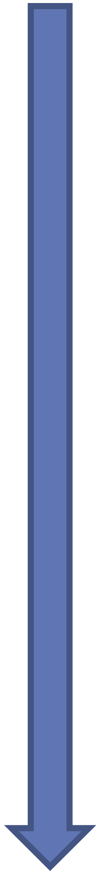
```
edit scene performCustomSetup
edit scene initializeEventListeners
edit scene My Story
scene setAtmosphereColor color ???
scene setFromAboveLightColor color ???
scene setFromBelowLightColor color ???
scene setFogDensity density ???
```

This short tutorial will show you how to add your remixed action into your world.

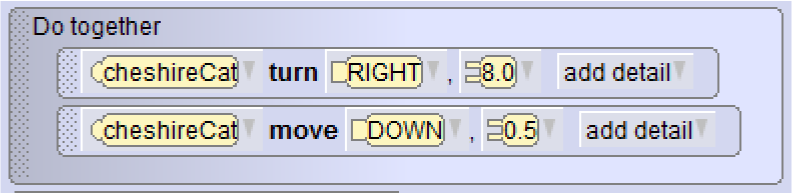
Experimental

Programming Constructs

Easy



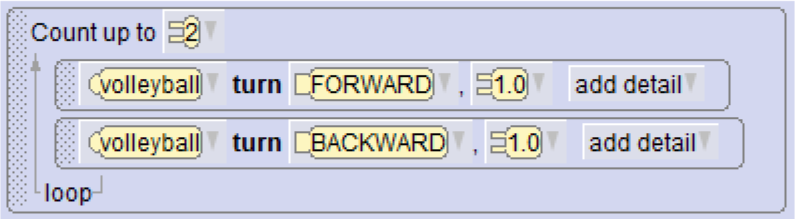
Hard



```
Do together
    cheshireCat turn RIGHT , 8.0 add detail
    cheshireCat move DOWN , 0.5 add detail
```

Do Together

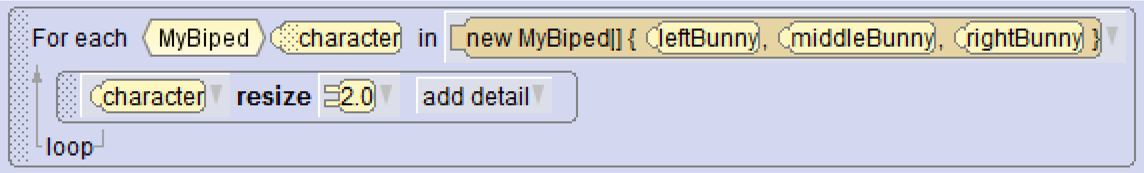
Execute in Parallel



```
Count up to 2
    volleyball turn FORWARD , 1.0 add detail
    volleyball turn BACKWARD , 1.0 add detail
loop
```

Count Loop

Loop n times



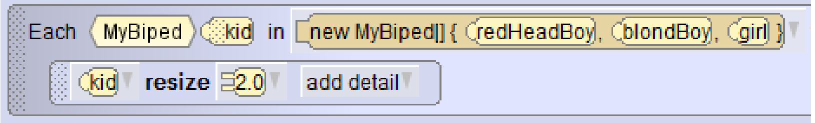
```
For each MyBiped character in new MyBiped[] { leftBunny, middleBunny, rightBunny }
    character resize 2.0 add detail
loop
```

For Each in Array Loop

Iterate over array

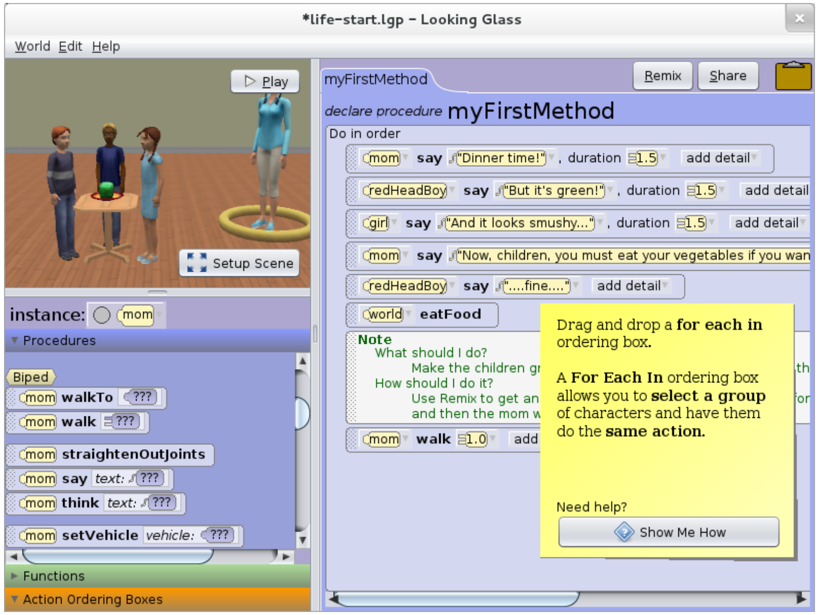
Training Phase

Remix Animation



```
Each MyBiped kid in new MyBiped[] { redHeadBoy, blondBoy, girl }
    kid resize 2.0 add detail
```

Control – Snippet Copied into Program



```
mom walkTo ???
mom walk ???
mom straightenOutJoints
mom say text: ???
mom think text: ???
mom setVehicle vehicle: ???
```

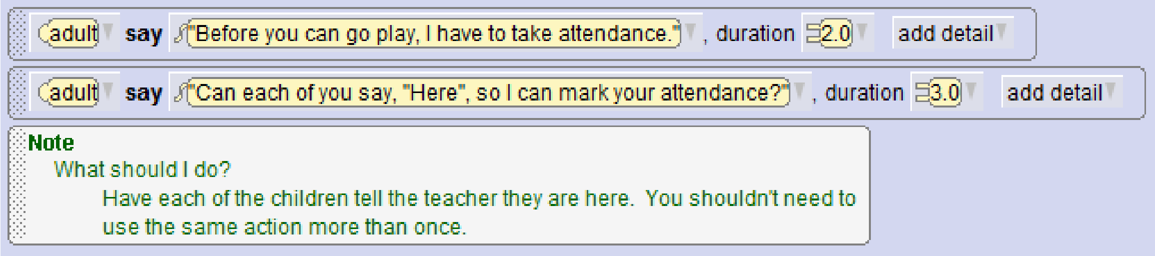
```
myFirstMethod
declare procedure myFirstMethod
Do in order
    mom say "Dinner time!" , duration 1.5 add detail
    redHeadBoy say "But it's green!" , duration 1.5 add detail
    girl say "And it looks smushy..." , duration 1.5 add detail
    mom say "Now, children, you must eat your vegetables if you wan
    redHeadBoy say "....fine...." add detail
    world eatFood
    Note
      What should I do?
          Make the children gr
      How should I do it?
          Use Remix to get an
          and then the mom w
    mom walk 1.0 add
```

Drag and drop a **for each in** ordering box.

A **For Each In** ordering box allows you to **select a group** of characters and have them do the **same action.**

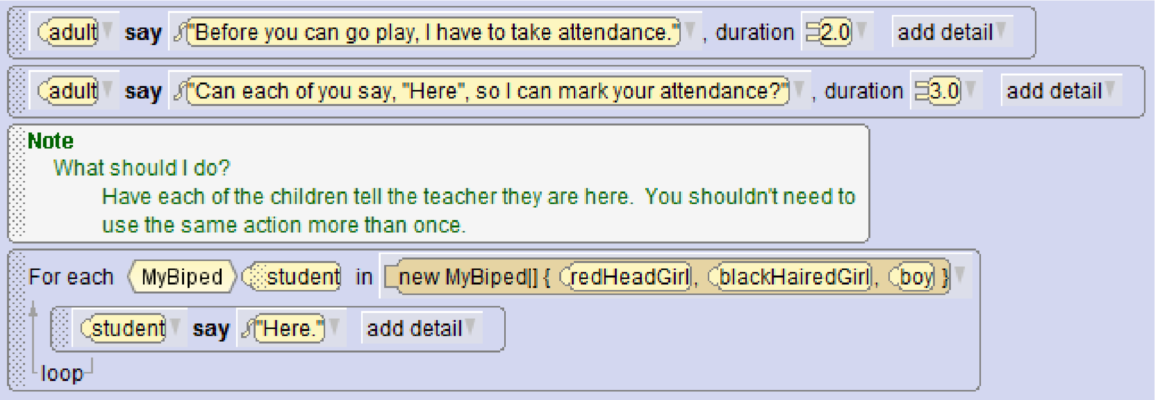
Experimental – Reconstruct Snippet in Tutorial

Transfer Phase



```
adult say "Before you can go play, I have to take attendance." , duration 2.0  add detail
adult say "Can each of you say, "Here", so I can mark your attendance?" , duration 3.0  add detail
Note
   What should I do?
        Have each of the children tell the teacher they are here.  You shouldn't need to
        use the same action more than once.
```

Initial Transfer Task Program



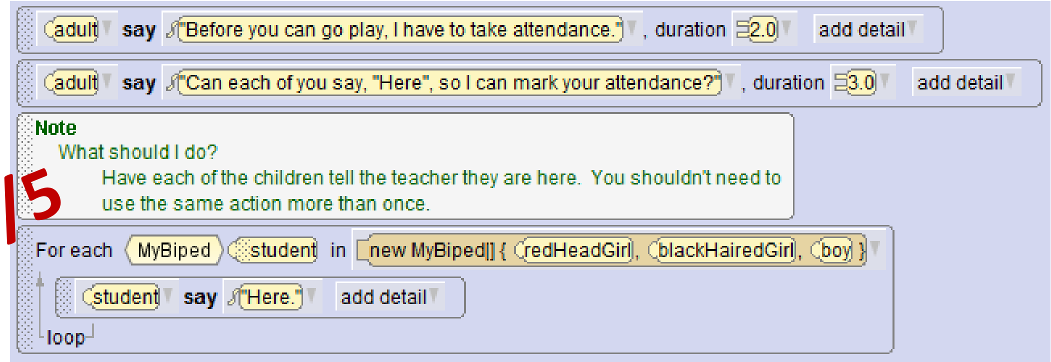
```
adult say "Before you can go play, I have to take attendance." , duration 2.0  add detail
adult say "Can each of you say, "Here", so I can mark your attendance?" , duration 3.0  add detail
Note
   What should I do?
        Have each of the children tell the teacher they are here.  You shouldn't need to
        use the same action more than once.
For each MyBiped student in new MyBiped[] { redHeadGirl, blackHairedGirl, boy }
    student say "Here."  add detail
loop
```

Completed Transfer Task Program

Grading Transfer Task Programs

Grading Criteria for the *For Each* Transfer Program: (5 points)

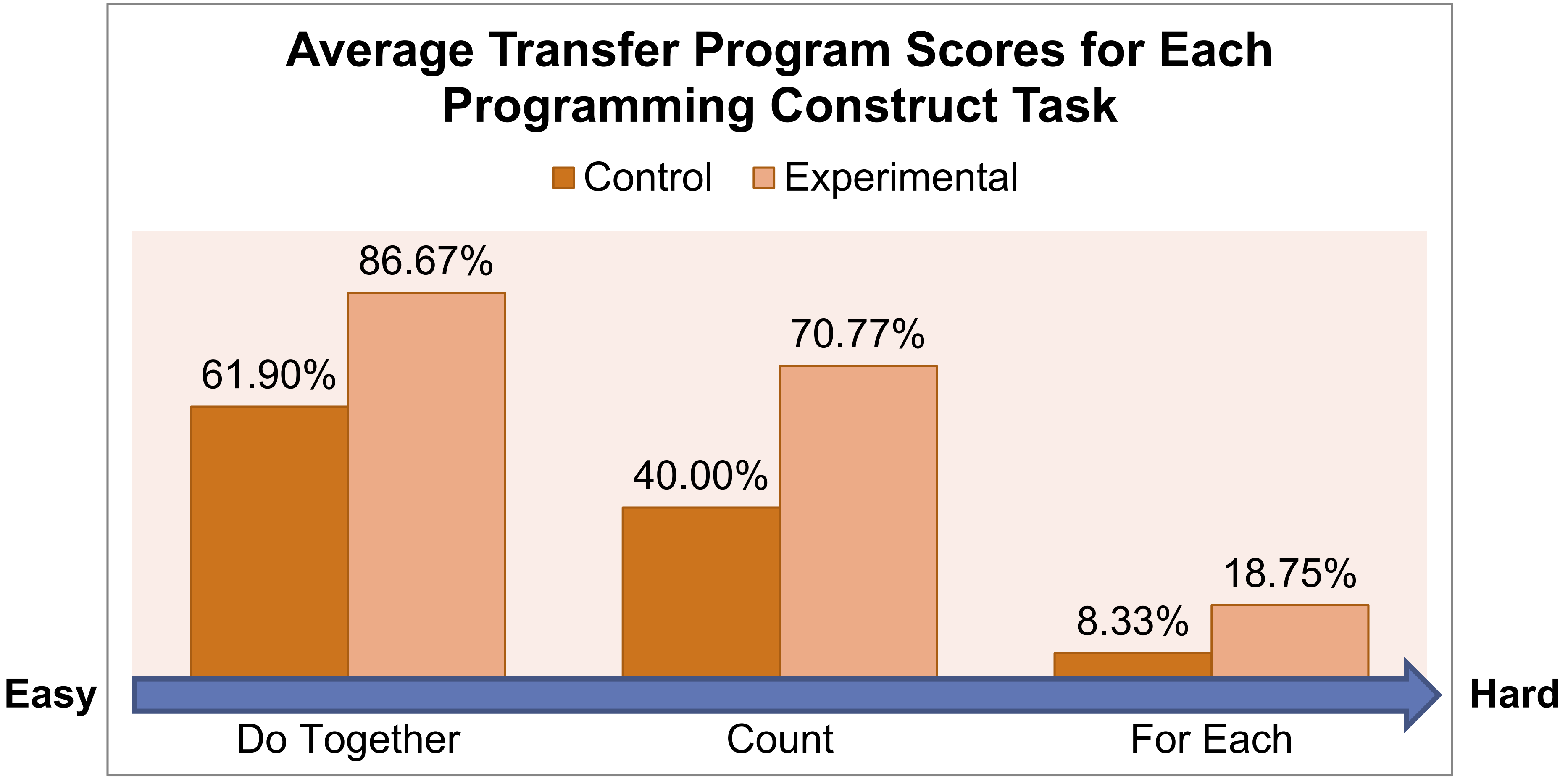
1. Program contains a *For Each* construct. If not, stop grading. (+1)
2. *For Each* contains at least one statement. If not, stop grading. (+1)
3. Array is defined correctly for the animation. (+1)
4. Programming statements use the loop iterator. (+1)
5. Animation is correct. (+1)



5/5

```
adult say "Before you can go play, I have to take attendance." , duration 2.0  add detail
adult say "Can each of you say, "Here", so I can mark your attendance?" , duration 3.0  add detail
Note
  What should I do?
      Have each of the children tell the teacher they are here.  You shouldn't need to
      use the same action more than once.
For each MyBiped student in new MyBiped[] { redHeadGirl, blackHairedGirl, boy }
    student say "Here."  add detail
loop
```

Results



Experimental condition performed 64% better. ANCOVA (F[2,37], $p < 0.05$).

Implications

Any code can be used as a learning resource.

Users can learn while they follow their own interests.

Personalize tutorials to the learner’s abilities.

Thanks

Kyle J. Harms
Washington University in St. Louis
harmsk@seas.wustl.edu

NSF

CONNECTING SCIENCE & THE COMMUNITY · HUMANÆ SCILICET SCIENTIÆ ET POTENTIÆ

Academy of Science
St. Louis | SINCE 1856

SAINT LOUIS sciencenter.

Why ANCOVA?

New Looking Glass users often have difficulty locating the Control Flow Tab.

- We provided a Control Flow Tab Hint
- Offered during the transfer program after 5 minutes
- Pointed to tab: “To complete this task, look here.”

We used ANCOVA with the presence or absence of this hint as a covariate.

- The hint was not significant (p = 0.48)