Lesson 4 - Variables - Teacher's Guide

http://www.stencyl.com/teach/act4/

Objective

Introduce students to (local) variables and promote principles of good coding. Students will learn the following concepts.

Outcome

Students will take an existing behavior and modify it, so that it can be more easily reused across the game, thereby promoting good coding habits.

Lesson Plan (1 - 2 hours)

Discussion
20 minutes

Cover the topics under Discussion Notes (Page 2)
Present the topics. Pose questions at appropriate points and encourage students to participate in the discussion.

Activity
40 minutes

Extend an existing game
Students will apply what they've just learned to add functionality to an existing game.

Activity
60 minutes

Work on extra activities
Students will work on a more challenging set of activities in order to demonstrate mastery of the concepts they've learned.

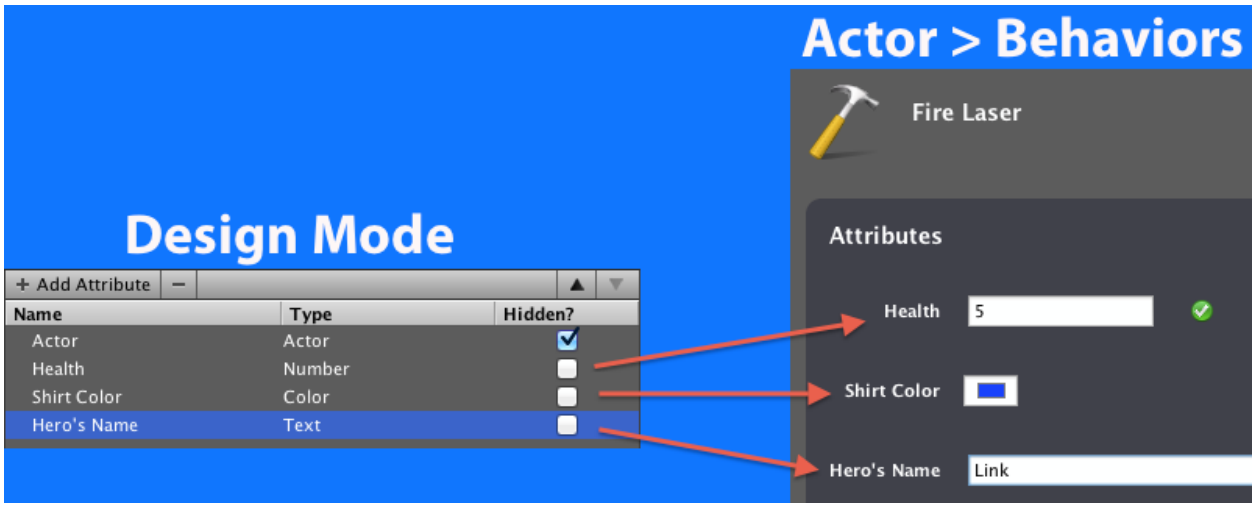
> **Note**: Extra activities are optional but recommended.

Discussion Notes

Topic 1: Attributes

In the prior lesson, we worked with Global Variables (Game Attributes), variables that can be referenced and modified anywhere within a game.

In contrast, Local Variables (Attributes) only apply to specific Actors and Scenes.



In the example above, the attributes include Health, Shirt Color and Hero's Name. *(Ignore Actor, which is an internal reference and best left out of this discussion)*

Discussion: What's the benefit of a local variable?

Answer: Suppose that local variables did not exist. If we only had global variables, how would we be able to have variables corresponding to specific actors? If we don't know ahead of time what actors will participate, it becomes virtually impossible (or at best, inconvenient) to have enough global variables to account for this.

Topic 2: Demonstrate how to work with Attributes

We recommend briefly showing students how to create an attribute (number or text), using the blocks corresponding to that attribute and showing how the attributes show up as configurable fields when a behavior is attached to an Actor or Scene.

TODO: Flesh this out into an explicit demo.

Topic 3: Good Coding Habits

Code Reuse

Discussion: What happens if you have a feature that you want every enemy in a game to share? Should you remake it for every enemy, or is there a better way?

Answer: That's what behaviors are for.

Behaviors let you reuse functionality across an entire game (and by extension, across multiple projects). That complex Jump behavior only has to be written once. This ability to reuse a feature is the principle of **Code Reuse**.

Writing things once is a good thing. If you discover a bug, you just have to fix it one place, versus several.

No Magic Numbers

Magic Numbers are unexplained numbers used in code. Generally speaking, if the number isn't 0, 1 or 2, it's a Magic Number. It's almost certainly a Magic Number if you use that same number multiple times in the same behavior.

Magic Numbers are tempting to use because they are easy to type in and forget. But over time, they can build up and make your code more difficult to maintain.

Attributes are the solution to Magic Numbers. They solve the problem in two ways.

- Attributes force you to come up with a name for the value, so the use of the value becomes self-documenting.
- If you decide to change the value, for whatever reason, you change it one place, versus several. Forgetting to change things that happen in multiple places is one of the most common sources for bugs.