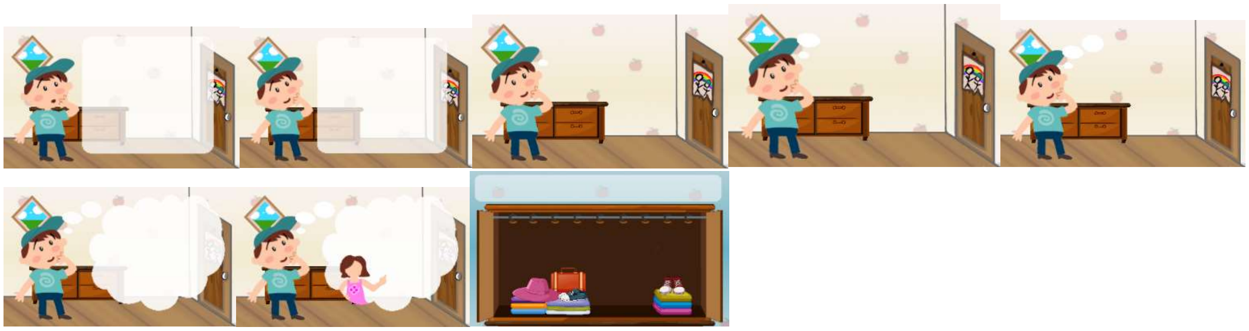
VARIABLES DIALOGUE

ACTIVE

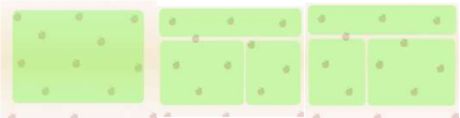
Top bgs:



**Note: The development of this is just cycling screens. If the game goes through an optimization, assets can be pulled out and bg screens condensed

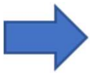
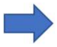
Ex: TOP.1=TOP.2, but *boy* asset would change. TOP.3=TOP.4=TOP.5=TOP.6, with a building “thought cloud” set of assets (x4). If this change is desired, screens will be appropriately modified ASAP.

Bottom bgs:



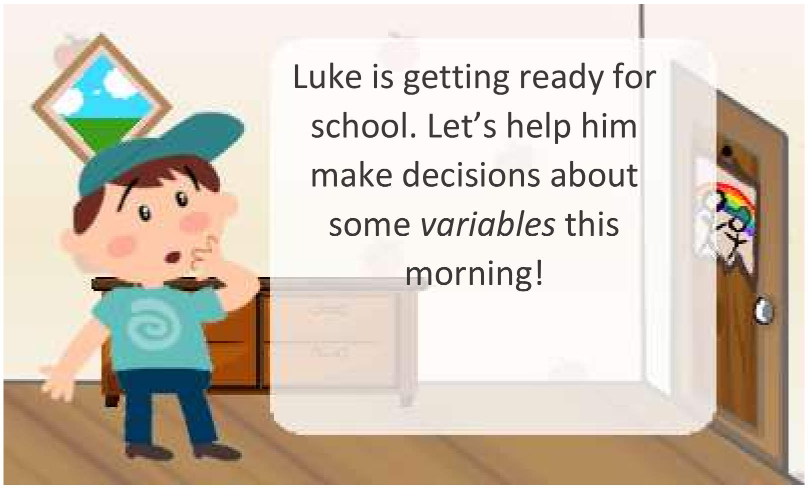
Assets:

Resized for TOP.8

 *resized:*  *and rotated:* ▷

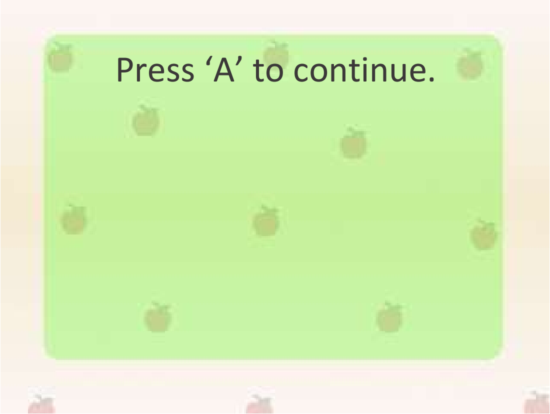
original size used

PANEL 1: TOP.1; BOTTOM.1



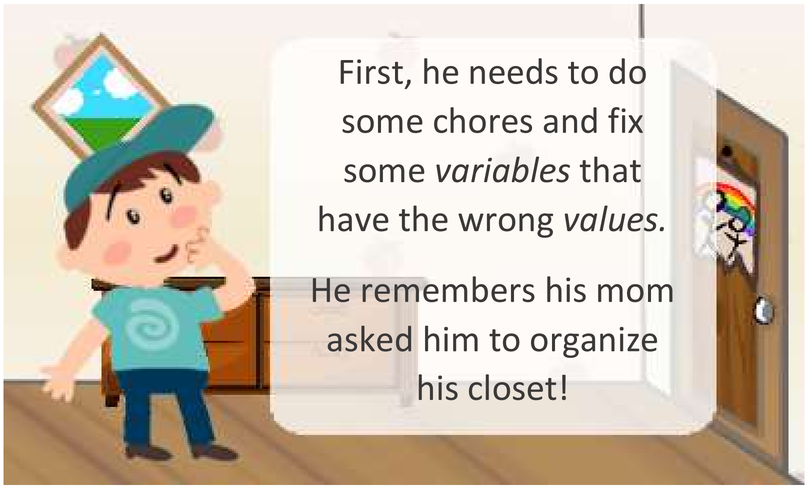
Luke is getting ready for school. Let’s help him make decisions about some *variables* this morning!

Immediate next screen from tutorial for now.



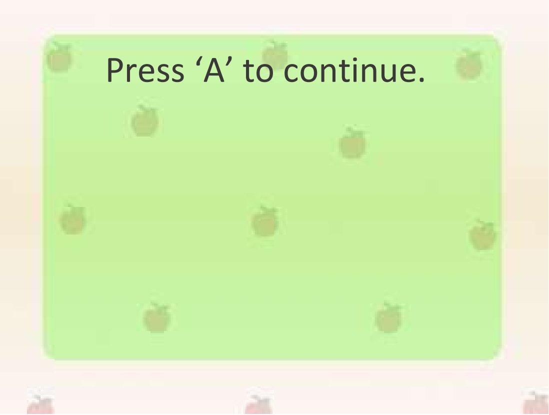
Press ‘A’ to continue.

PANEL 2: TOP.2; BOTTOM.1



First, he needs to do some chores and fix some *variables* that have the wrong *values.*

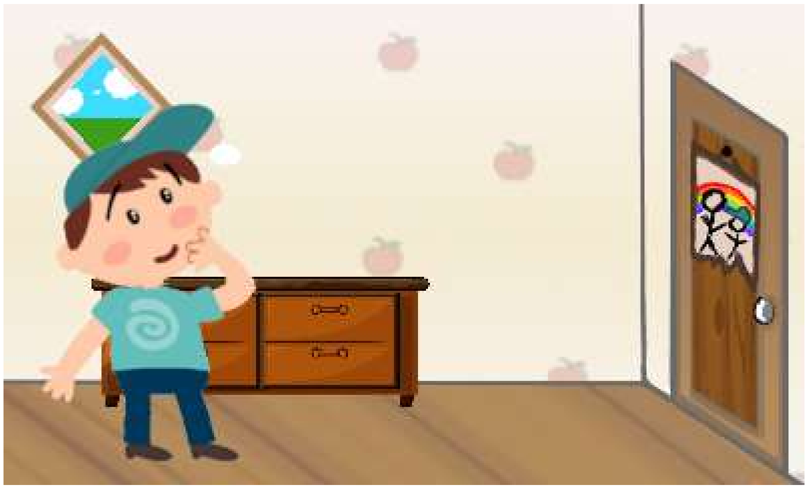
He remembers his mom asked him to organize his closet!

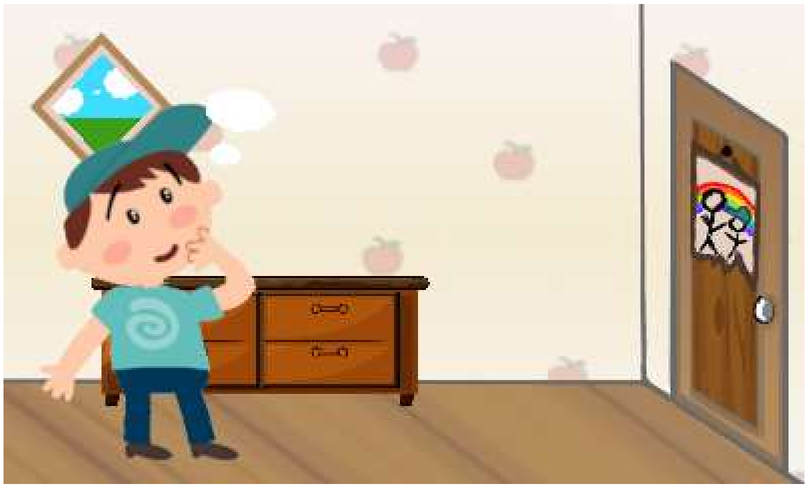


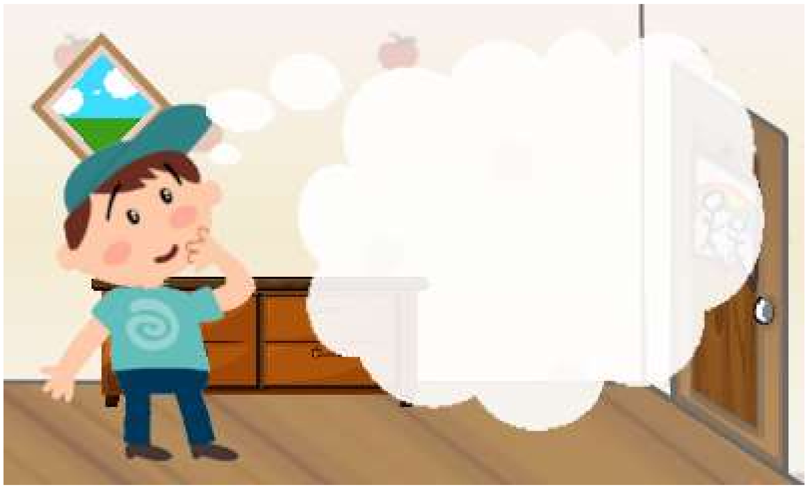
Press ‘A’ to continue.

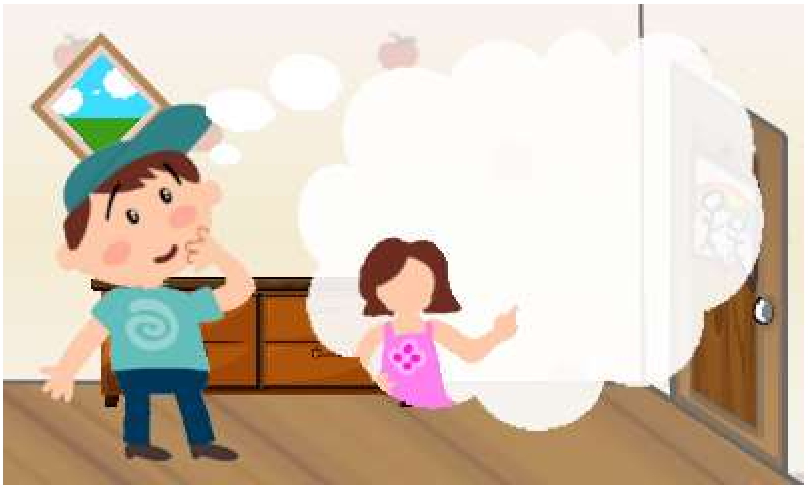
PANEL 3: *SEQUENCE* TOP.3->TOP.4->TOP.5->TOP.6->TOP.7; BOTTOM.1 *unchanged*

Note: as mentioned in asset intro, this may need to be consolidated to asset population. Ideally, this would be *timed and automatic* (simulating animation).

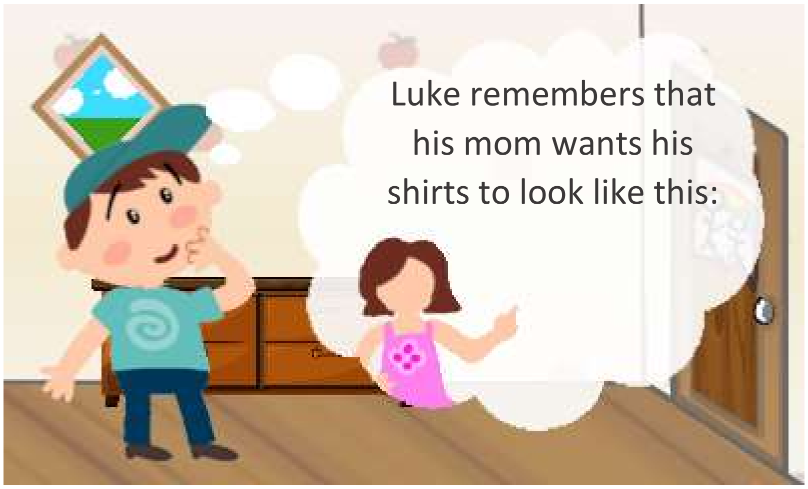




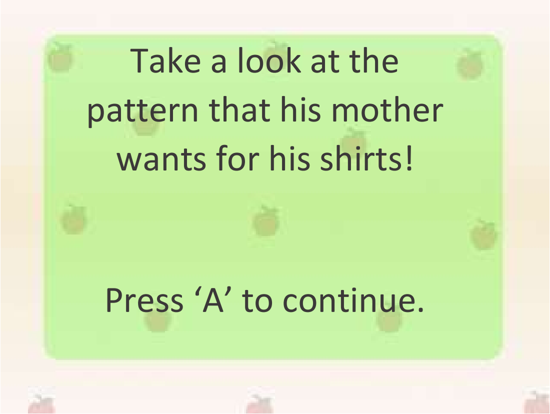




PANEL 4: TOP.7; BOTTOM.1



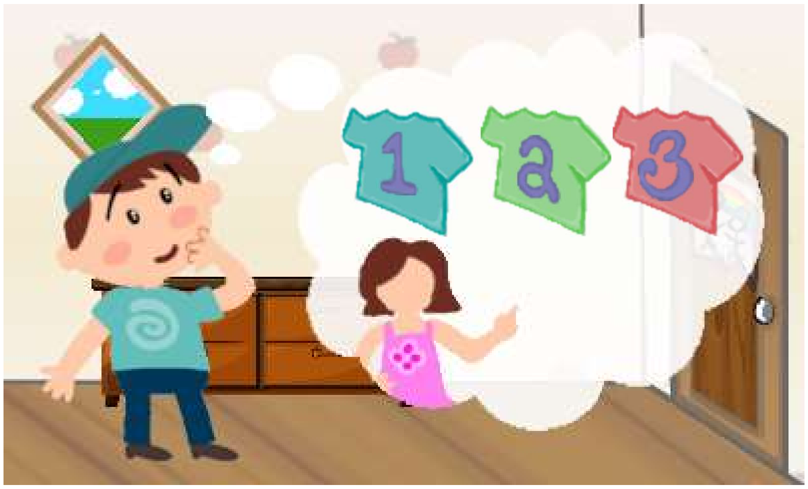
Luke remembers that his mom wants his shirts to look like this:



Take a look at the pattern that his mother wants for his shirts!

Press ‘A’ to continue.

PANEL 5: TOP.7; BOTTOM.1 *unchanged; text remains*

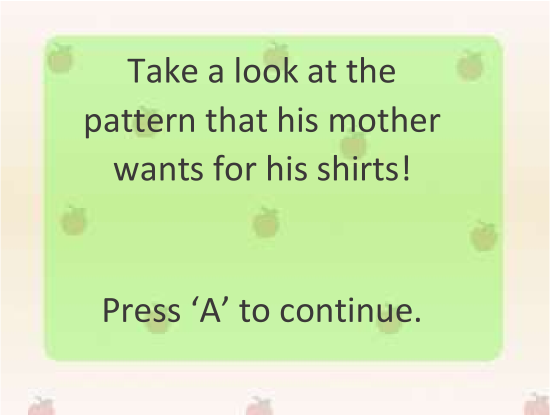


Generation of the pattern. Space seems to be limited to 3 shirts. We can work with multiple pattern types, but start with something simple:

All [color] + incrementing numbers

All [number] + alternating colors

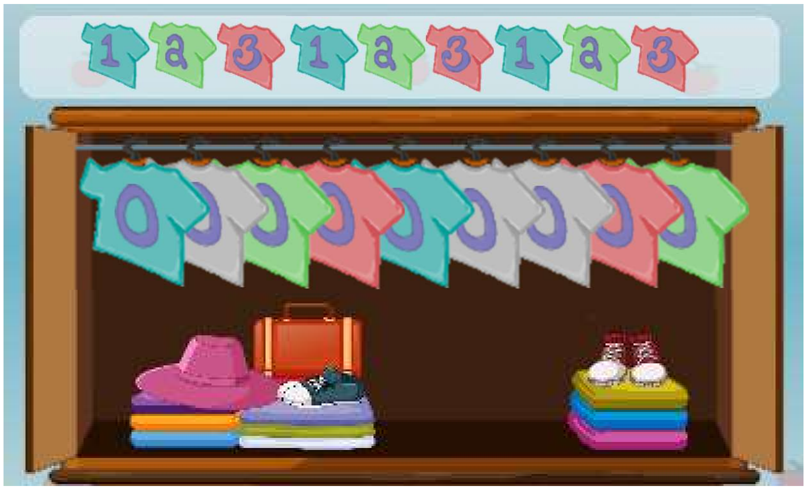
The example here is too complicated for the first iteration.



Take a look at the pattern that his mother wants for his shirts!

Press ‘A’ to continue.

PANEL 6: TOP.8; BOTTOM.1

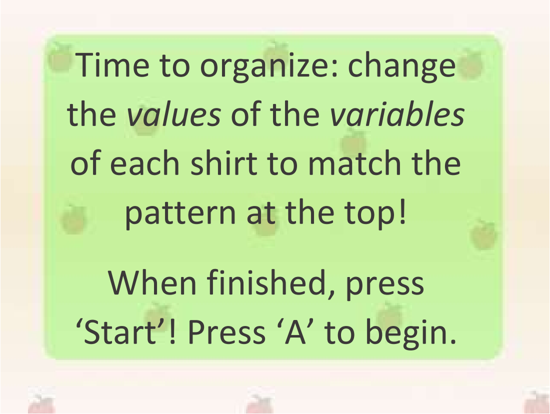


Asset population is going to need to be done in a specific order to reflect this look – shirts and numbers will need their own arrays and specific starting y values and x-offsets.

Assuming assets pile on top of one another:

Right → left generation

Place shirt; place number; place shirt; place number; etc



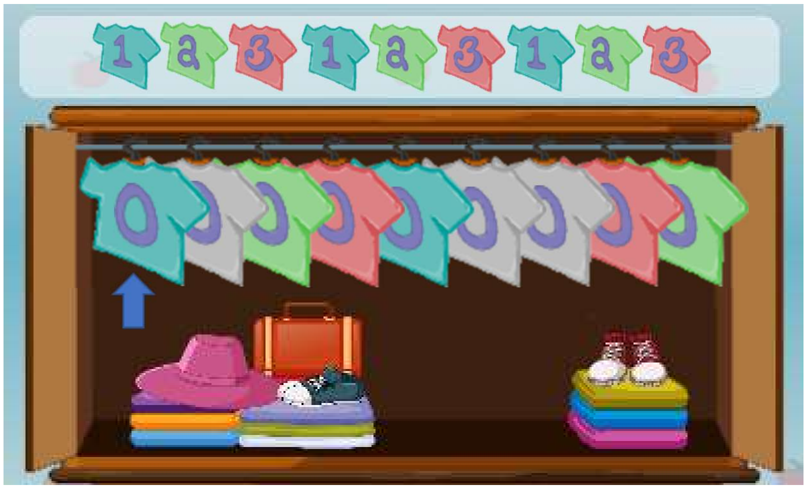
Time to organize: change the *values* of the *variables* of each shirt to match the pattern at the top!

When finished, press ‘Start’! Press ‘A’ to begin.

4 asset arrays: correct/win state shirts and numbers (can be smaller size) and random pattern references.

1) *TOP: Correct pattern array stored with 3 or 9 items: generate with same y value and consistent x-offset. Same goes for the numbers.*
2) *CLOSET: Traverse a new random shirt pattern array backwards (and number array) but interchanging. Y values the same, but x-offsets will be different than the top population, and might be different between each other.*

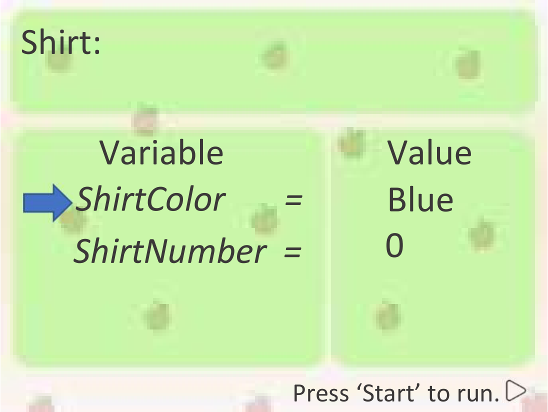
PANEL 7: TOP.8; BOTTOM.2



Begin game functionality. All instructional text is gone, and the game will run based on user input only.

Note that the "start"/run trigger is now available. Depending on how the win state is generated, they may or may not be able to "run" the sequence until its been filled in.

If the user input is literally overwriting the randomly generated sequence (that was drawn in), we just need to make sure that their changes are translated to the win-state sequence.



Shirt:

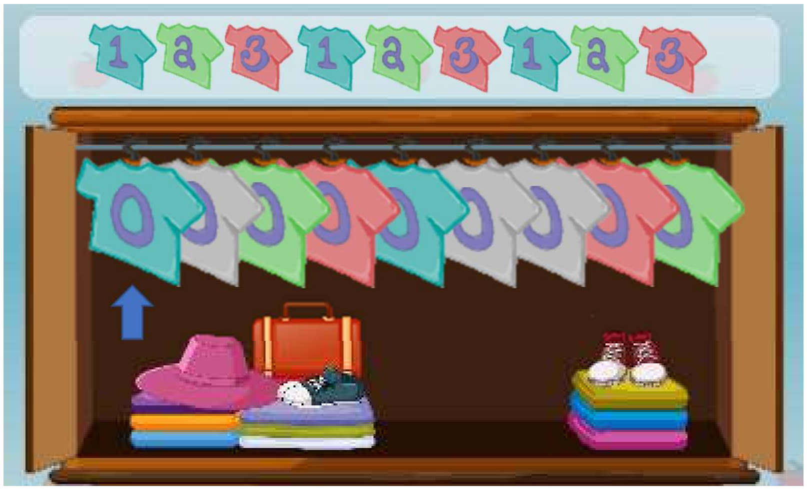
	Variable		Value
➡	*ShirtColor*	=	Blue
	ShirtNumber	=	0

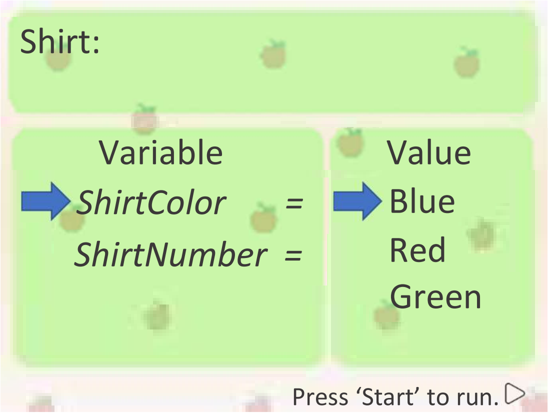
Press 'Start' to run. ▷

Similar to the tutorial: On the top screen, if a shirt is "highlighted", the bottom screen fills in the current available data. They should press 'A' to begin interacting with the data.

The bottom screen arrow indicator can either generate immediately, or wait for user to press 'A'.

PANEL 8: TOP.8; BOTTOM.2 *on press 'A'*

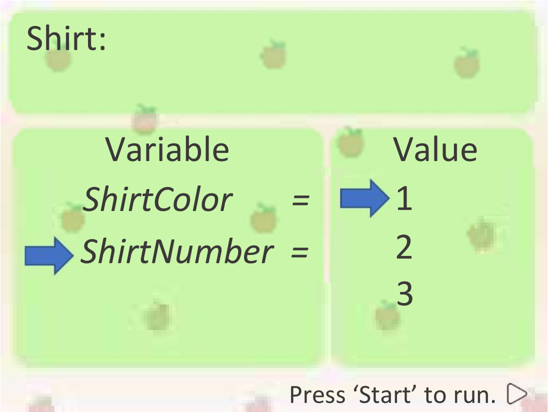




Shirt:

Variable		Value
➡ *ShirtColor*	=	➡ Blue
ShirtNumber	=	Red
		Green

Press ‘Start’ to run. ▷



Shirt:

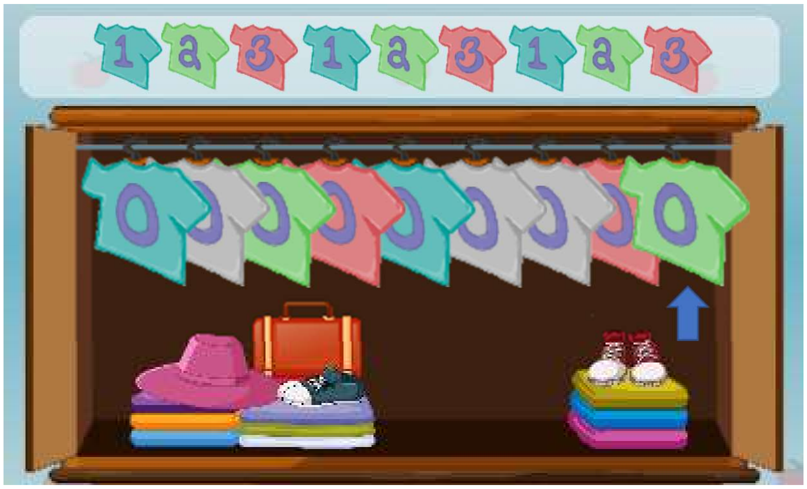
Variable		Value
ShirtColor	=	➡ 1
➡ *ShirtNumber*	=	2
		3

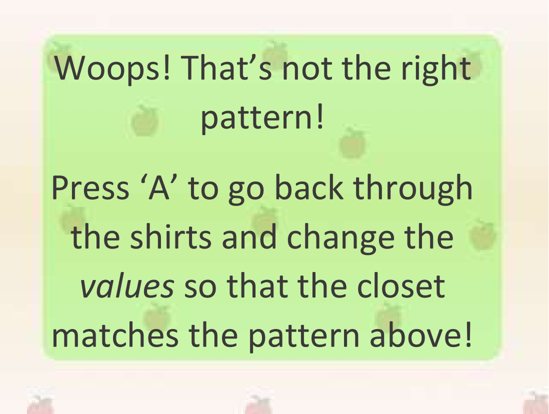
Press ‘Start’ to run. ▷

Goal: to have the shirts in the closet actively reflect the changes made by the user. I am worried that asset population will be an issue since the generation/layering is so specific.

Short term: layering updates will look bad until the scene completely re-draws the updated asset arrays.

PANEL 9: *ERROR INCORRECT SEQUENCE* TOP.8; BOTTOM.1



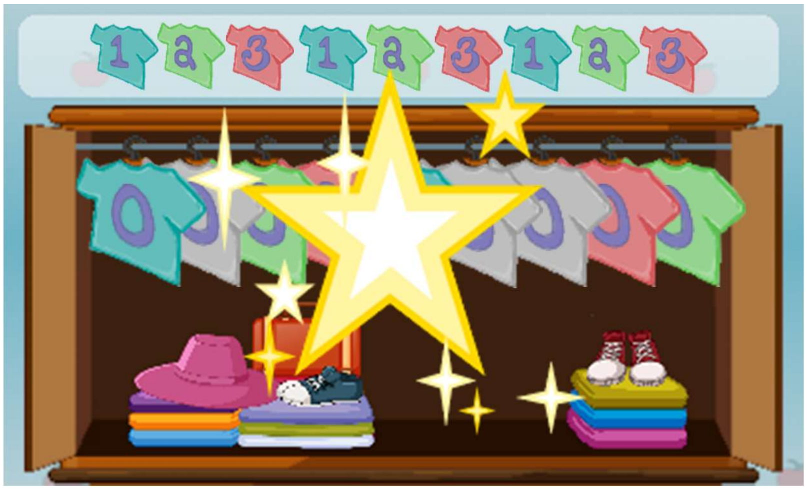


Woops! That’s not the right pattern!

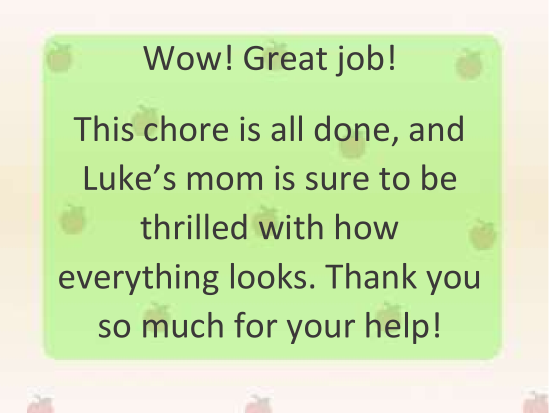
Press ‘A’ to go back through the shirts and change the *values* so that the closet matches the pattern above!

Once they press ‘A’, the bottom screen simply repopulates the BOTTOM.2 screen and generates the data based on the shirt still highlighted in the top screen (whatever they were on right before they hit run).

PANEL 10: *WIN* TOP.8; BOTTOM.1



Two assets oscillating: success_stars_topFitted_1 and success_stars_topFitted_2 (in objects folder) will simulate “sparkling”. These assets are sized exactly as the top screen, no positioning required.



Wow! Great job!

This chore is all done, and Luke’s mom is sure to be thrilled with how everything looks. Thank you so much for your help!

END GAME.