American History: Seeing the Past

Developed by Anh Trinh, Ademola Olayinka, and Carolyn Sun, under the supervision of Dr. Susan Rodger at Duke University. Special thanks to Mr. Gary Horton for ideas on the premise and gameplay.

Premise: One of Benjamin Franklin's unknown inventions was his Time Traveler glasses. Charged by lightning rods, the glasses take one to the past. They were discovered in a tin box buried in the cornerstone column at the Founders Hall in Philadelphia. An unpublished copy of *Poor Richard's Almanac*, which was found alongside it, explains how the glasses work. The glasses allow one to travel back to different time periods of American history and witness American history first hand.

Educational Objective:

- Incorporate quizzes into an interactive game (We recommend a multiple choice quiz).
- Game design includes objects relating to the class content, American history.

Game Play:

- There are 7 questions in a set order. If you answer a question correctly, you will be allowed to select an object .
- Selecting an object gives you points. Different objects give you different amounts of points. Thus, the objective is to answer questions correctly, discover what objects give you the most points, and earn the highest score possible.
- To explore the world and look for different objects, you press different keys to move the camera.
- Your score is updated on the top left of the screen.

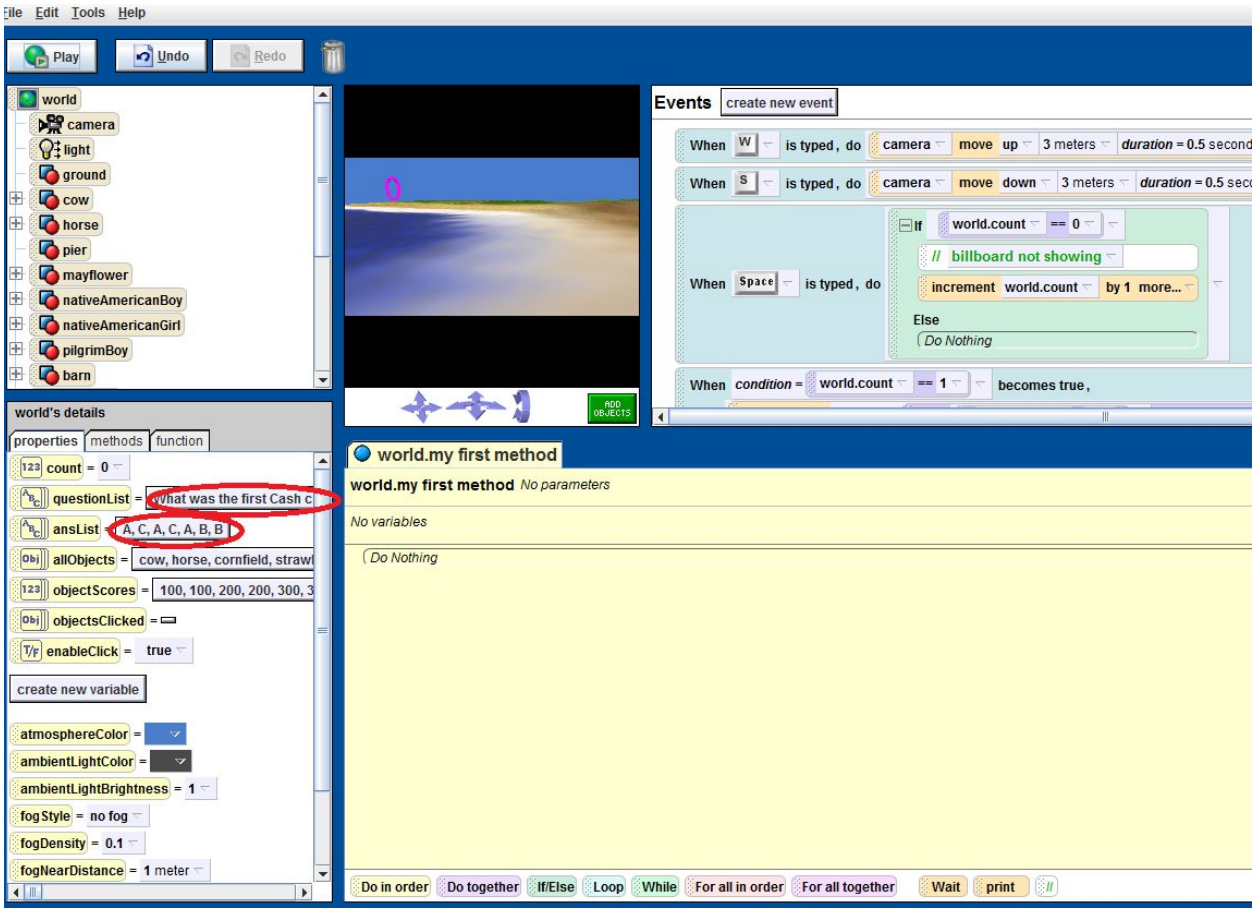
Commands:

- Press the Spacebar to start the game.
- Use the arrow keys to move the camera around.
- Press W and S to move the camera up and down respectively.
- Mouse Click on objects to earn points.

This write-up will teach you how to:

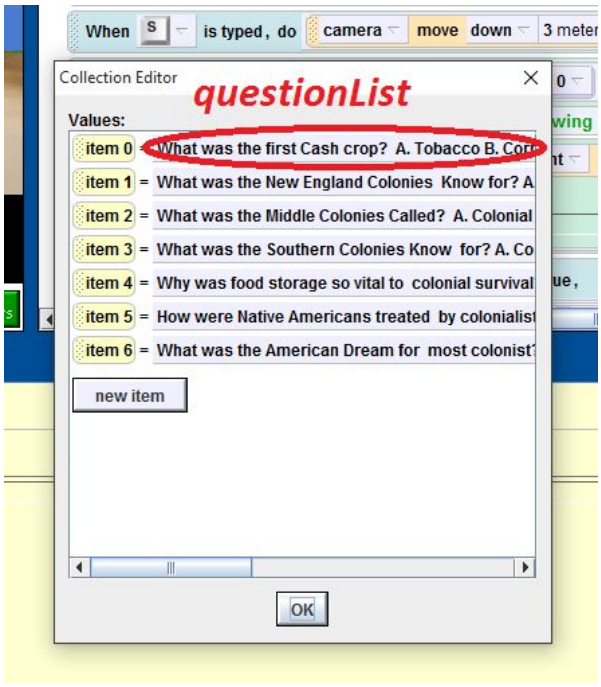
How to modify the set of questions and answers

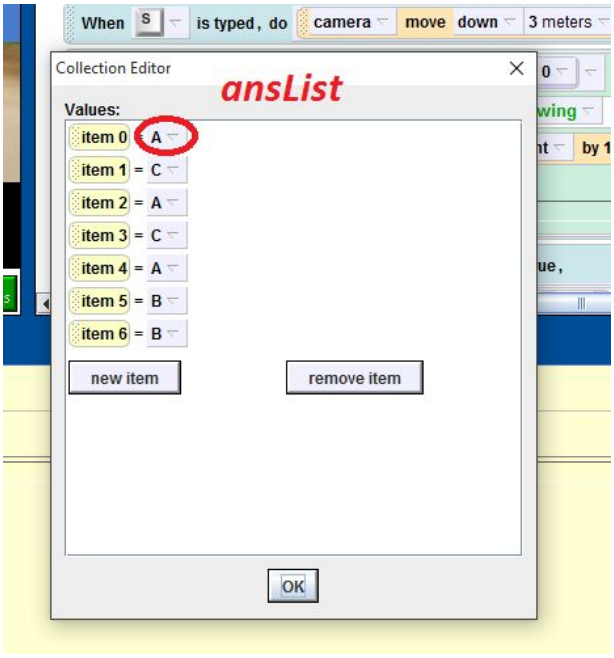
Step 1: In world's properties panel, there are a list of questions and a list of answers. They are named *questionList* and *ansList* respectively. To edit these lists, click on the grey blocks next to the names.



Step 2: Click on the grey boxes to edit the questions and answers.The order of the questions in *questionList* correspond to that of their answers in *ansList*. For example, the first question is item 0 in *questionList* (recall that list indexes start at 0). The answer to the first question is also item 0 in *ansList*.

For a more specific example:
In *questionList*, item 0 is “What was the first Cash crop? A. Tobacco B. Corn C. Wheat D. Maize”.
The answer to that question is found in *ansList*. Item 0 of *ansList* is A.

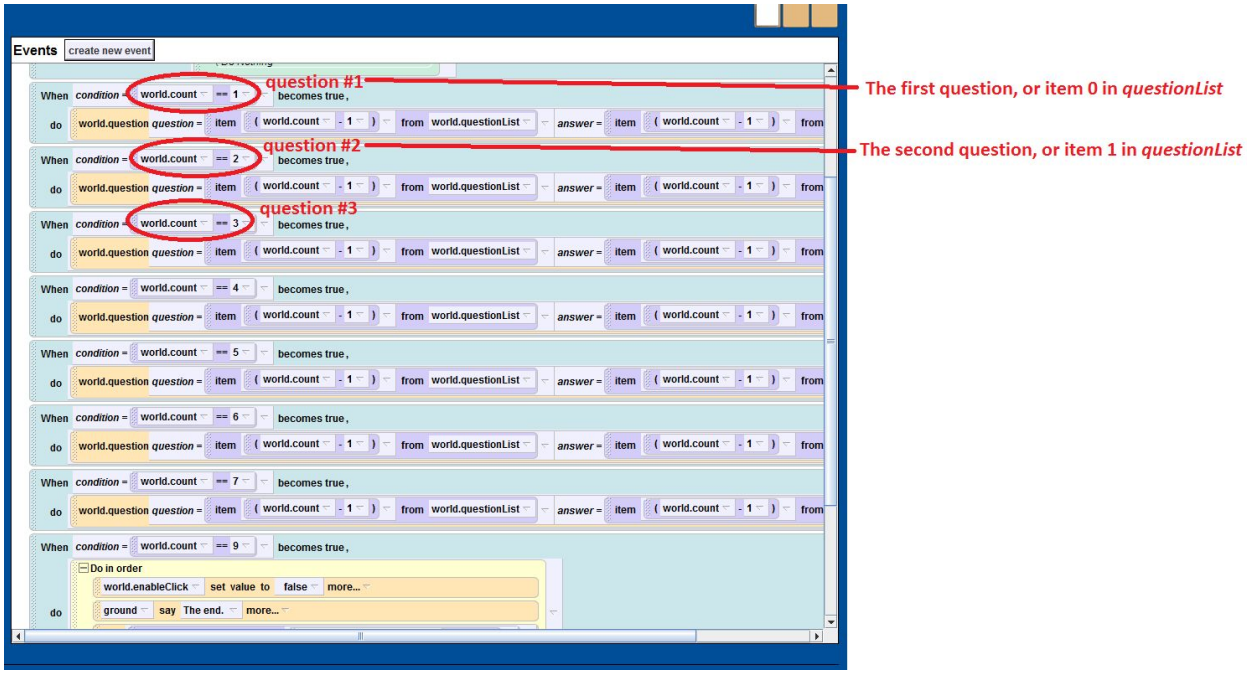




Be aware that the students’ answers (string inputs) must match exactly with the answers you put in *ansList*. Thus, if you are using multiple choice, be sure to tell your students whether the letter should be lowercase or capitalized. Or, if an answer is “Both A and C”, inform your students that they must type their answers exactly like the choices given in the question.

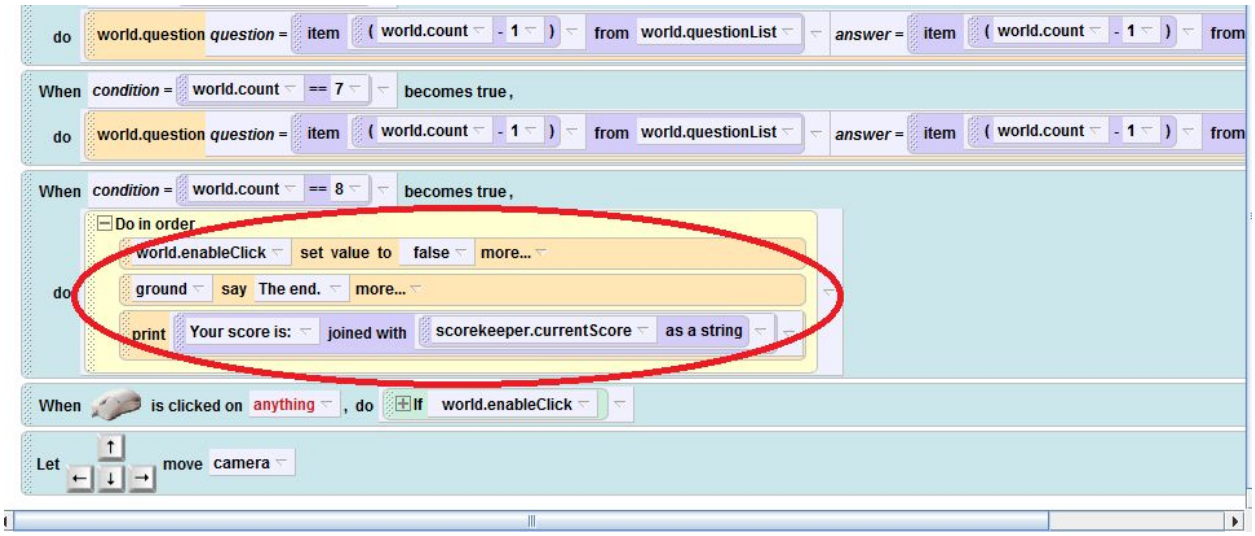
How do add/delete questions

Adding or deleting the number of questions is a bit trickier. Here is what the code in the Events panel looks like right now:



If you go to the world's properties panel, there is a number variable called *count*. This *world.count* variable is keeping track of how many questions the user has answered. *world.count* starts off as 0, and once you press *Space* to start the game, *world.count* becomes 1. If you look at the Events panel, when *world.count* becomes 1, the user is asked the first question, or item 0 in *questionList.* Once the user answers the first question, *world.count* becomes 2. When *world.count* becomes 2, the user is asked the second question, or item 1 in *questionList.*

If you scroll to the bottom of the Events panel, you will see an Event that executes when the user has answered all 7 questions:



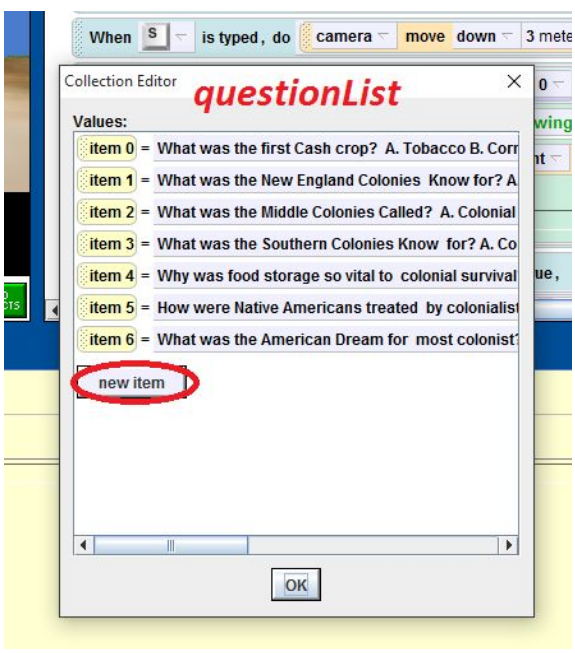
This event is the “end” event; the ground says “The end.” and the program prints your score. This “end” event executes when it reaches x + 1, where x is the number of questions in *questionList*. In this case, there are 7 questions, so the “end” event executes when *world.count* reaches 8.

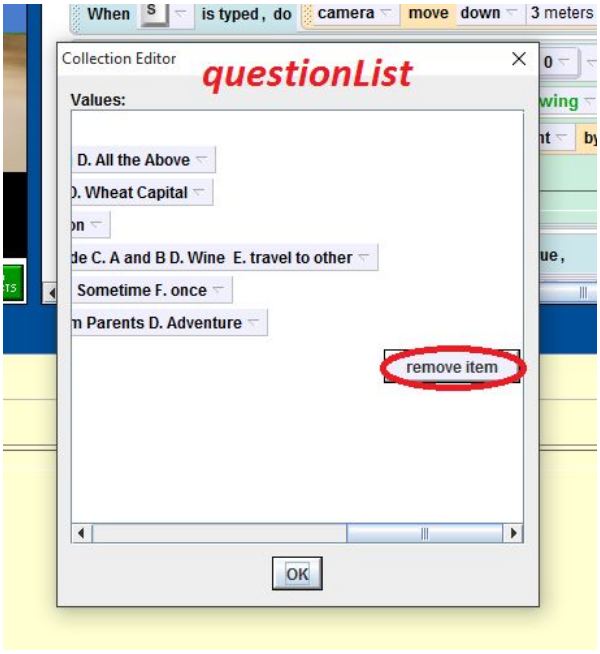
So to add/delete the number of questions, you must make sure the “end” event executes at the right time (when *world.count* reaches x + 1).

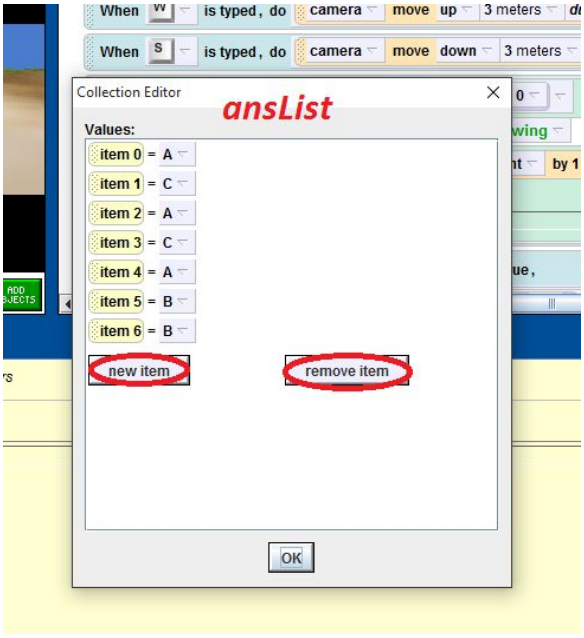
Step 1: First, you must add questions/answers to the lists. To do this, go to the world’s properties panel and find the lists *questionList* and *ansList*. Click on the grey boxes next to the names.

Step 2: Use the buttons *new item* and *remove item* to add/delete questions and answers. Remember that the order of the questions in *questionList* correspond to that of their answers in *ansList*.

Note that for *questionList*, you may have to scroll to the right to see the button *remove item*.

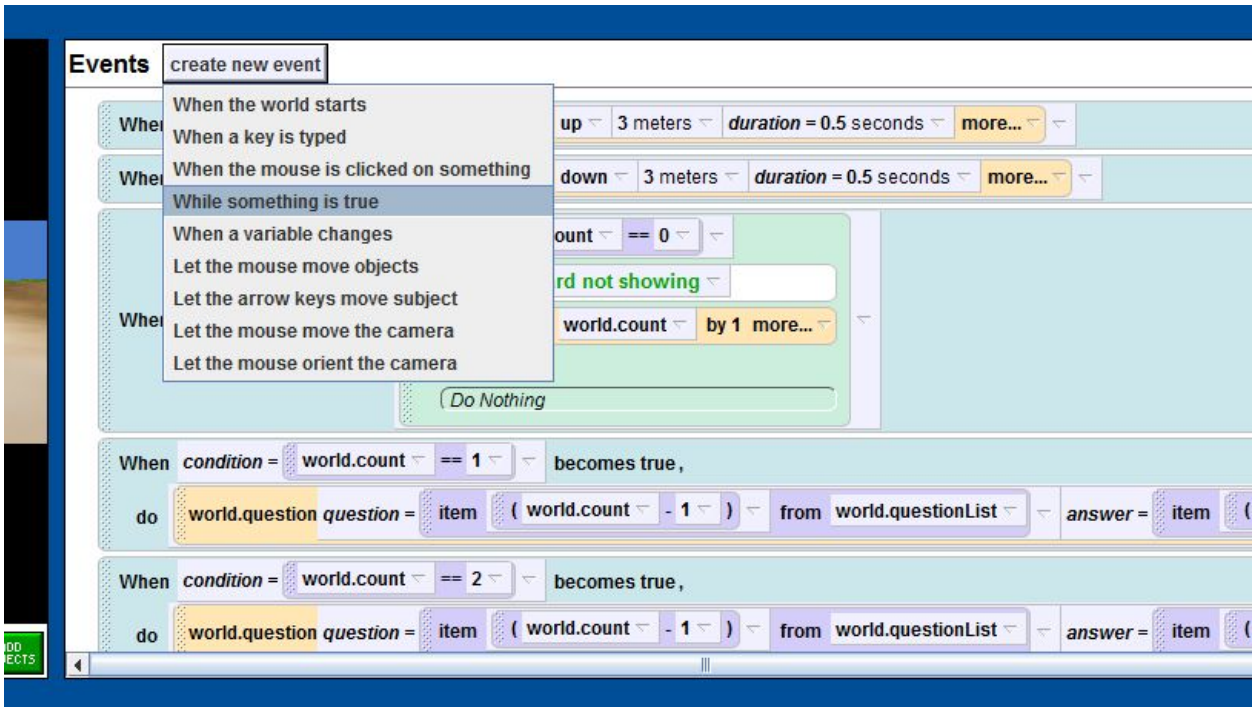


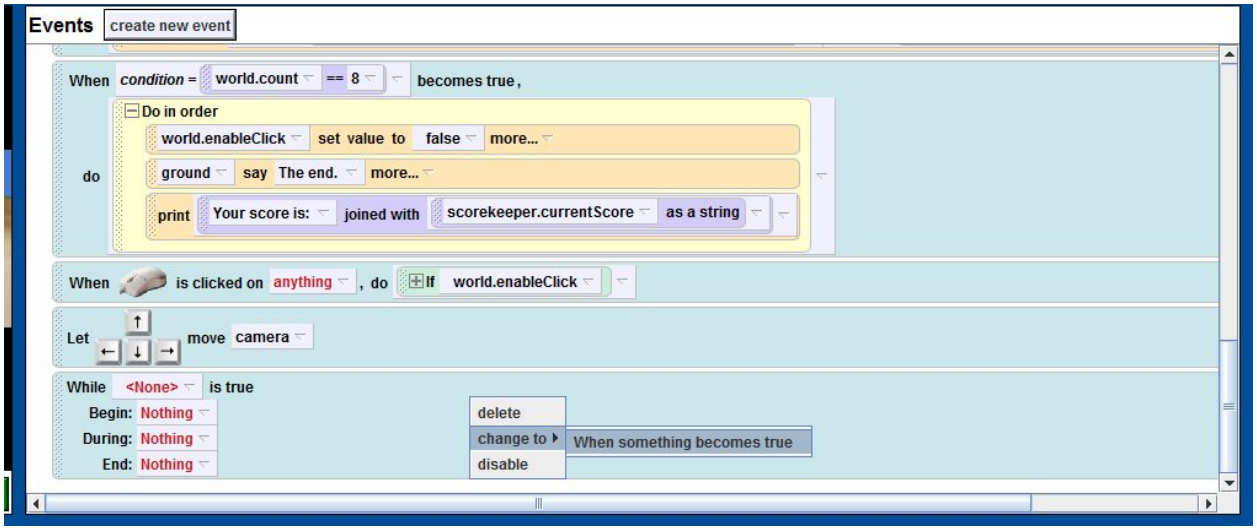




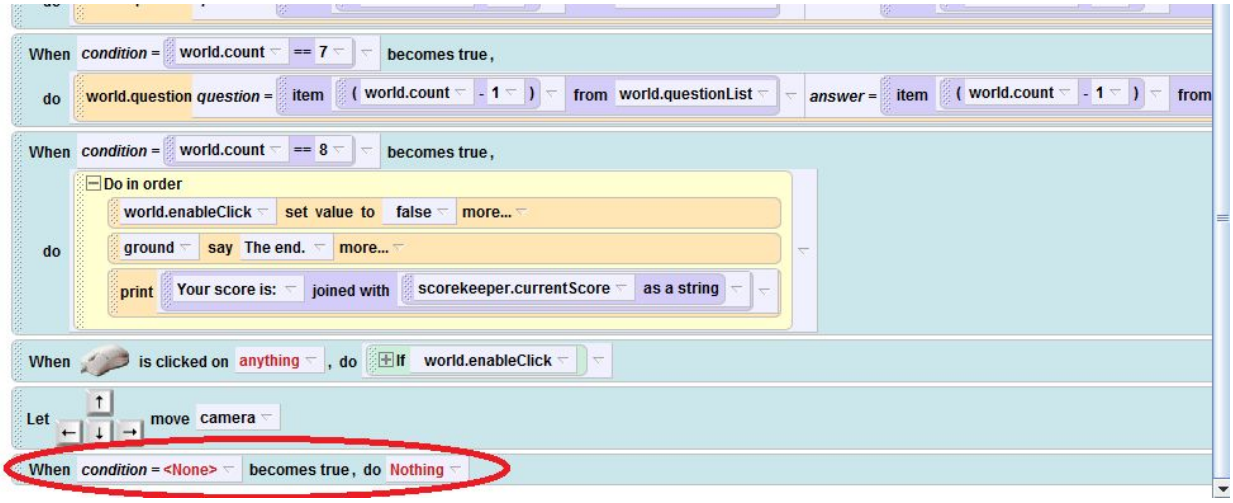
Step 3: Once you have updated *questionList* and *ansList* to your liking, you must add/delete Events to match your lists. In other words, the “end” event must execute when *world.count* reaches x + 1.

Consider this example in which you want to increase the number of questions from 7 to 9. First, you must create an Event that asks the user the 8th question when *world.count* equals 8. To do this, create a new event called “While something is true.” It will appear at the bottom of the Events panel (scroll down). Right click on it → change to → when something becomes true.

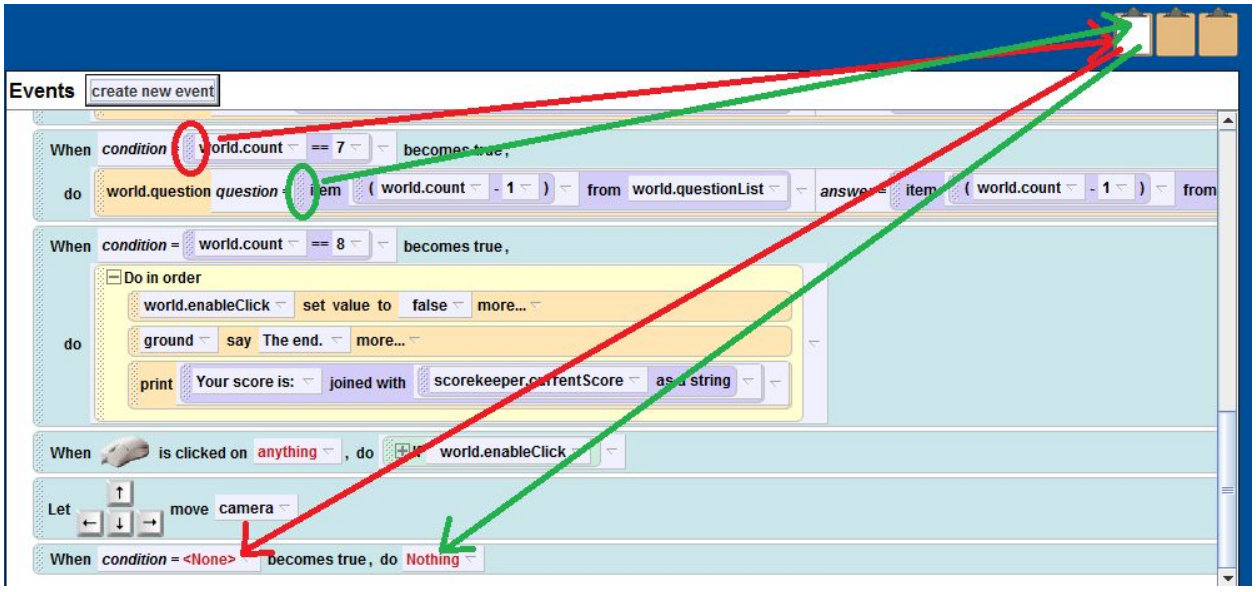




You should end up with this:



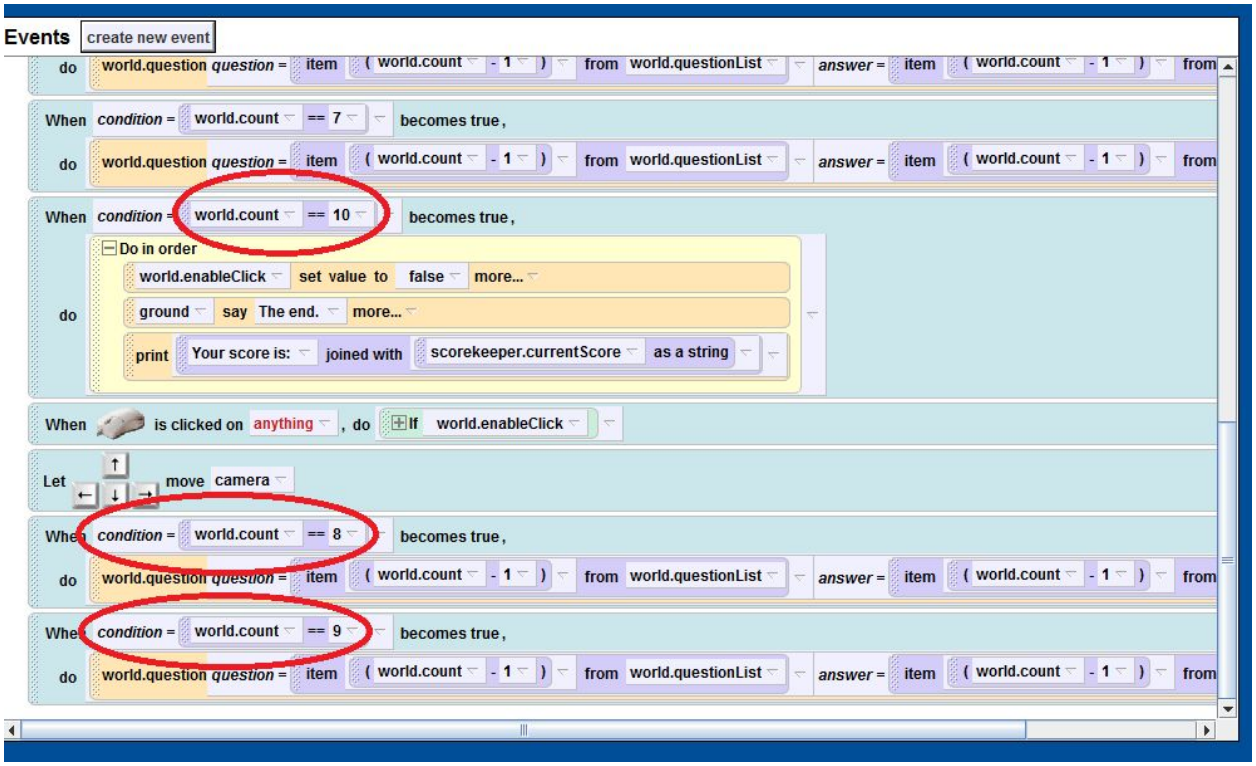
Step 4: Then, you must update the condition and method of the new Event (where it says <None> and Nothing right now). To do this, make copies of the condition and method of an Event we made (such as the Event that says *When world.count == 7)*. Make copies by clicking and dragging a statement to the clipboard. Make sure you drag the circled areas (see image below) of each statement, to ensure that you have copied the entire statement. The clipboard should turn white when you have dropped a statement in there. Then, click and drag from the clipboard to the new Event.



Repeat Steps 3 and 4 for however many more questions you want to add. For this example, we did it a total of two times to add questions 8 and 9. Then, we updated the new Events to “When *world.count* == 8” and “When *world.count* == 9”, so questions 8 and 9 can execute. Lastly, we updated the “end” event to “When *world.count* == 10”. Again, the “end” event executes when

world.count equals x + 1, where x is the number of questions. There are 9 questions, so the "end" event executes when *world.count* equals 9 + 1 = 10.

Our screen looked like this:



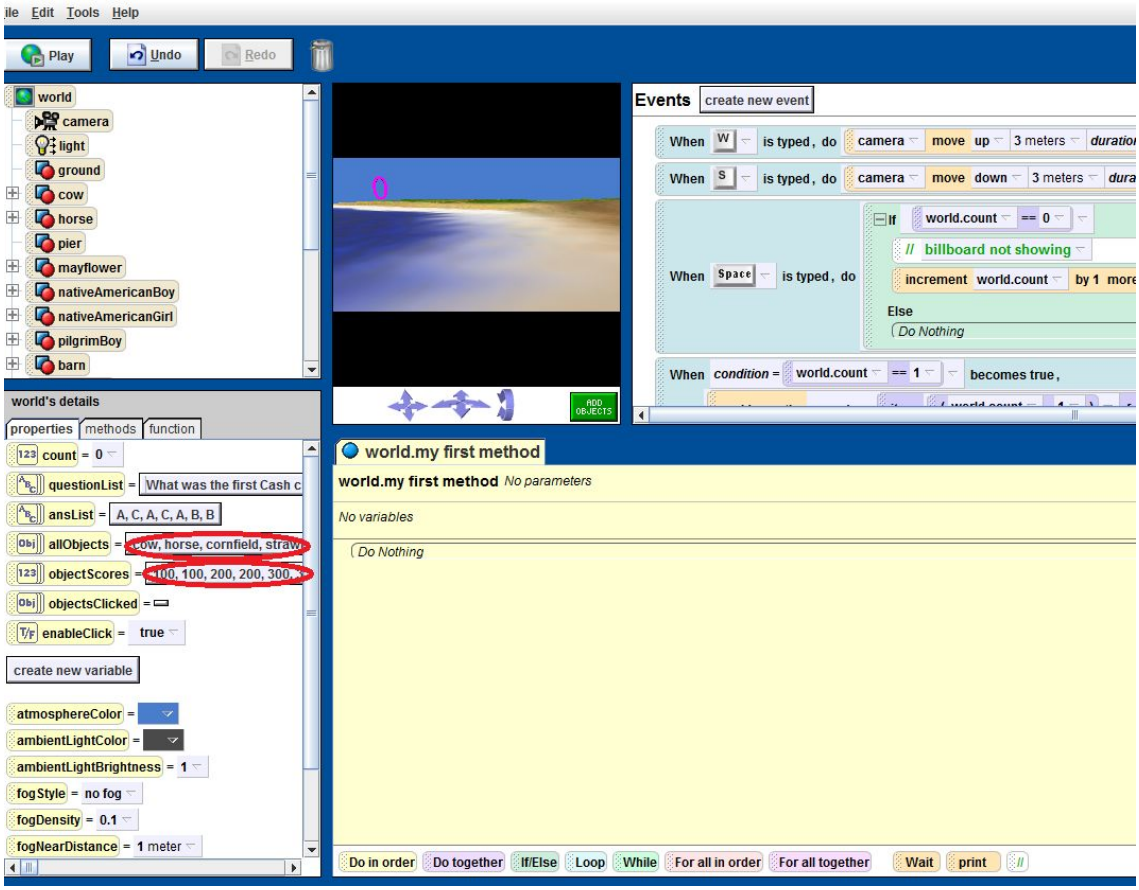
Now our game will play with 9 questions!

To decrease the number questions, remove items from *questionAns* and *ansList*. Then, delete the appropriate amount of Events (right-click on the Event → delete). Lastly, update the condition of the "end" Event to execute when *world.count* == x + 1, where x is the number of questions.

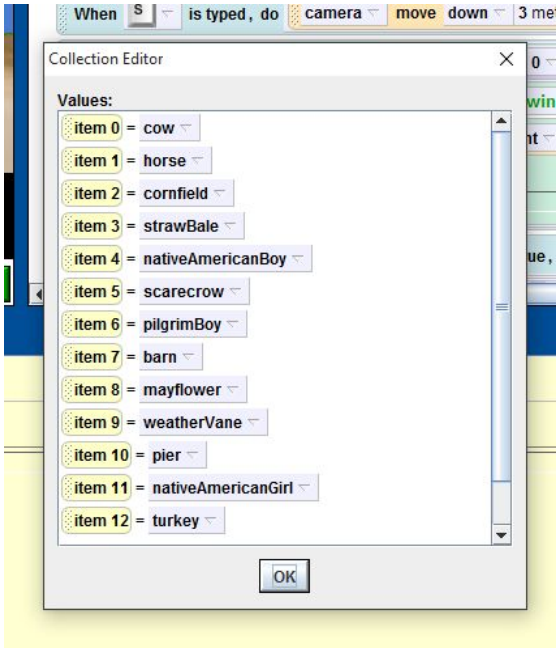
How to modify/add objects and object scores

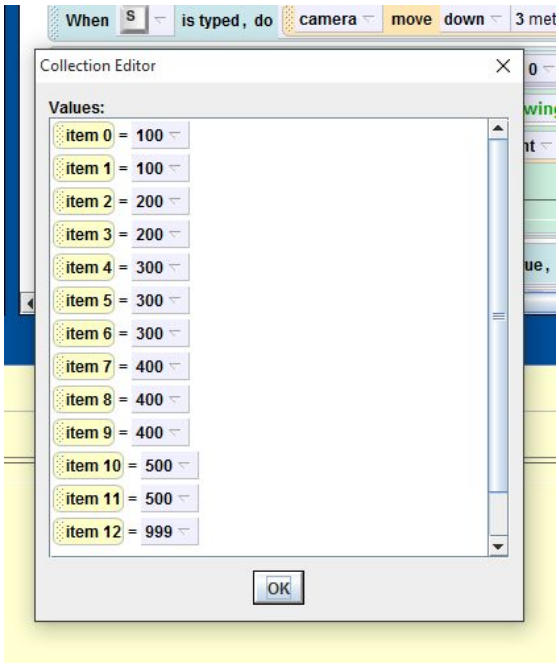
Step 1: When you modify the questions to suit a topic or class unit of your choice, you may also want to modify the objects and setting. To do this, first save a copy of the downloaded game using *File → Save World As* in the top left corner of the screen. Then, you can use the green *Add Objects* button to make any changes to the objects and/or world. When you are done adding objects, be sure to change the camera back to *startView*. Also, note that there must be more objects than questions.

Step 2: Now you must make simple changes to the code. In world's properties panel, there are a list of objects and a list of their respective scores. They are named *allObjects* and *objectScores*. To edit these lists, click on the grey blocks next to the names.



Step 3: Use the buttons *new item* and *remove item* to add/delete objects and their respective scores. You will have to scroll down to see the *new item* and *remove item* buttons. Again, the order of the objects in *allObjects* correspond to that of their scores in *objectScores*. For example, the first object is item 0 in *allObjects*, and its score is item 0 in *objectScores*.





In the images above, the score for "cow" is 100, the score for "horse" is 100, the score for "cornfield" is 200, and so on.

We recommend that you assign higher scores for the objects that are more difficult to find.