

Lab 5.3 - Fewer Balls

Big Ideas

Products can be designed for life cycle

Personal design interests require the evaluation and refinement of skills

Tools and technologies can be adapted for specific purposes

In this lab you will get a chance to reuse and repurpose previously created code. You will also get a chance to evaluate and refine your skills and to see how previous code can be adapted for specific purposes. As you complete this lab, think about these Big Ideas and how they've been a part of all of the labs and activities in the course.


Have you ever completed a program and thought the following?

- I wish I had more time, I could remake this into...
- I could use this part of my program to start a program that...
- If I altered this a bit I could make a certain game that...
- If I could learn how to ... then I could probably create a program that...

These ideas are consistent with the Big Ideas in the course. The important thing to remember is that in designing and creating computer programs, you can always see how ideas and code can be reused, repurposed and improved upon.

In this lab, you will build on what you created in lab 5.2 to enable better management of the number of sprites in the program.

Part 1 - Getting Out of Hand

1. Open up your SNAP program from [Lab 5.2](#). Modify your program so that, when the 'd' key is pressed, all bouncing balls are deleted. DO NOT delete the prototype-- you should be able to create new bouncing balls after you have removed the old clones. Use the  block and a message.
2. What if you wanted to remove only a few clones? Or only specific clones? What would be needed in order to accomplish that?

Part 2 - Better Control

1. Add a **global** variable to your program called `g_nextID` and give it a value of 1 when the green flag is clicked.
2. In your master ball sprite, create a **sprite** variable called `s_ID`. Modify your program so that each time a new clone is created, the clone's `s_ID` variable gets the value currently in `g_nextID` and `g_nextID` is incremented by 1.
3. Change your program so that when the 'd' key is pressed, the *newest* bouncing ball gets deleted. Think about the right way to use the variables you created in the previous steps to know which sprite to delete.
 - a. *(Hint: in order for this to work right, you should reuse old IDs once the clones are deleted. So, for example, if the most recently created clone was number 6, and you hit 'd', clone number 6 should be deleted. Then, if a new clone is created, it should be a new clone number 6.)*
 - b. Try to do this without requiring a lot of special cases in your code-- every clone should operate in the same way to determine if it should be deleted.
4. BONUS: Add code so that if the 'x' key is pressed the program asks for an ID number and deletes that numbered clone. All clones with higher numbers should be renumbered so that ID numbers remain contiguous. (For instance, if clone number 5 is deleted, then clone numbers 6, 7, and 8 should be renumbered as clone numbers 5, 6, and 7 respectively. Then, the next clone created should be a new clone number 8.) This is tricky and will require you to think very carefully about how to use the variables.



Lab 5.3 Criteria	
1.2 Delete all clones	0.5 points
2.1 Assign unique s_ID to each clone	0.5 points
2.2 Delete newest clone	0.5 points
2.3 Delete newest and add combined	0.5 points
2.4 Bonus: Delete specific clone	0.5 points
PROJECT TOTAL	2.5 points