Level Two: Explorations

NATIONAL PEER REVIEWED

23002DD

Discovering Computer Science & Programming through Scratch

Updated for Scratch 3.0

About this Guide

This book of explorations in Scratch is the second book in a collection of three. While you are advised to go through the first book before using this one, this is not a requirement, though we do assume you have basic facility with the Scratch programming environment, and can write scripts using repeat and forever loops, conditional statements (if-then, if-then-else), and can use variables both in simple expressions and in conditional tests. This book delves deeper into important principles of computer science such as generalization and modularity, and introduces some additional features of Scratch such as clones and lists. Going through this book should strengthen your programming skills, hopefully help you write better programs, and give you some interesting ideas for further exploration. We hope you enjoy it.

Authors:

Lenny Pitt, Professor Emeritus of Computer Science, University of Illinois

Judy Rocke, Curriculum Development Specialist, Office for Mathematics, Science, and Technology Education (MSTE), University of Illinois

Jana Sebestik, Assistant Director STEM Curriculum Design, MSTE, University of Illinois

Support for this guide is provided by the 4-H Computing Connections (CS4H) project funded by the University of Illinois Extension and Outreach Initiative and also by the Department of Energy and the Department of Homeland Security under Award Number DE-OE0000780.

Layout and Design:

Christina Tran, Graphic Designer, MSTE, University of Illinois, iamchristinatran.com

Scratch is a project of the Lifelong Kindergarten Group at the MIT Media Lab (**scratch.mit.edu**). Images of the Scratch cat are used with permission. All other screenshots and images used in this guide are licensed under the Creative Commons Attribution-ShareAlike License.

Table of Contents

For the Facilitator

This is the second computer science manual in a collection of three (so far). It uses the Scratch programming environment. The activities in this book assume that the user has experience with loops, conditional statements, and variables. This book delves deeper into important principles of computer science such as generalization and modularity, and introduces some additional features of Scratch such as clones and lists.

This curriculum provides youth with a series of tutorials and challenges within the Scratch environment. Some of the activities are short and may take only thirty minutes, but others are more complex and offer opportunities to explore. Young people can work on the activities individually, with partners, or in a guided instructional setting. If students are working together, it is important to make sure that each student has equal time at the keyboard. It will also be helpful if each youth has his/her own guidebook.

As a facilitator of this project, encourage youth to talk about what they learn as they try new scripts and find new blocks. Youth will learn faster and more, when they discuss their projects with others. The Scratch community encourages users to share their projects on the Scratch website and to remix others' projects. Just be sure to give credit to the original project creator. There are nearly 100,000,000 registered Scratch users sharing projects. Join the fun!

This curriculum was written for youth in Grades 5-12, but may be used and adapted for younger and older audiences, based on experience.

All three levels of Discovering Computer Science & Programming through Scratch were written using both the CSTA (Computer Science Teachers Association) K-12 Computer Science Standards and the ISTE (International Society for Technology in Education) Standards for Students as guidance. In addition to developing computer programming skills and providing computing practice, these learning materials offer opportunities for collaboration and stimulate computational thinking. Activities have been designed to promote creative design and encourage empowered learners. The global Scratch community emphasizes positive digital citizenship and responsible collaboration.

Additionally, Using Mathematics and Computational Thinking is one of the Next Generation Science Standards Scientific and Engineering Practices, while these lessons explicitly address the following Common Core State Standards for Mathematical Practice:

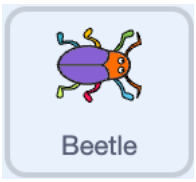
MP1: Make sense of problems and persevere in solving them.
MP2: Reason abstractly and quantitatively
MP6: Attend to precision
MP8: Look for and express regularity in repeated reasoning.

Clones

▶ Introduction to Clones

In Scratch a clone is a copy of a sprite. You can create and delete clones while a program is running or as a result of interaction with the user.

This lesson helps you understand clones and how they work.

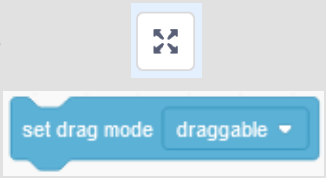


1. Create a beetle sprite and delete the cat sprite. Shrink the beetle so it has ample room to move around the stage.
2. Create these scripts for the beetle.

3. Press the "c" key. You do not see anything, but there is a clone. It is behind the beetle so you cannot see it. This clone starts in the same position as the original beetle.
4. Press the space bar to move the beetle. You still cannot see the clone. Why not? The clone's script is the same as the original beetle's script, so it moves in exactly the same way. It is still behind the original beetle.

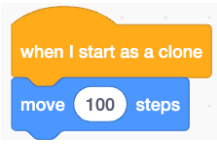
5. Drag the beetle somewhere else on the stage. This reveals the clone. Now press the space bar. Notice that both beetles move.

Note: Click this box shown above the stage. Now you are in the full screen, or player, mode. To drag the beetle around on this screen, click this "set drag mode (draggable)" block found in the **SENSING** category. You can use this block in the beetles scripts, but you do not have to. Click it once and try dragging the beetle around the screen in full screen mode.



A clone is not quite a separate sprite. It does not appear in the sprite area just below the stage. To find out which of the beetles on the stage is the original, click the beetle sprite shown below the stage. This causes the sprite on the stage to flash. Which is the original beetle?

6. Click the green flag. The clone is gone and only the original beetle remains. Next create this script for the beetle.



7. Use the mouse to move the beetle to the left side of the stage. Then press the "c" key. The clone is now 100 steps ahead of the original beetle because the new script moves the clone 100 steps when it starts.
8. Click the green flag and then press the "c" key once. How many beetles are there? Two, the original and the clone. Press the space bar to move both beetles forward 50 steps.

9. Press the “c” key again. How many beetles do you see? Are you sure? Use your mouse to drag all the beetles around. How many do you see now? Press the space bar to see all of them move forward 50 steps. Can you explain why there are more than three beetles?

10. Click the green flag. Then press the “c” key three times. How many beetles do you see? Drag them around to make sure. Are you correct?

> If the beetle or the clone travels off the stage, use the mouse to drag it back onto the stage.

11. Complete this table:

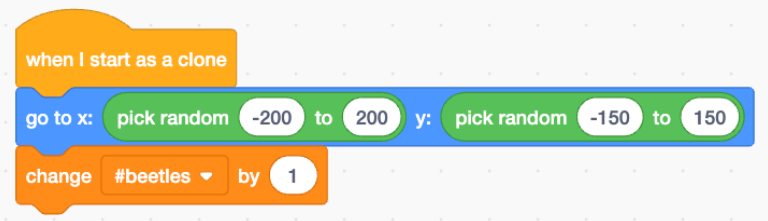
Number of times the “c” key is pressed	Number of beetles
0	1
1	2
2	4
3	8
4	
5	
6	

12. When the “c” key is pressed, each clone makes a clone of itself, so the number of beetles doubles each time. Instead of counting beetles, you can write a script that will count them for you. First create a variable called **#beetles**.

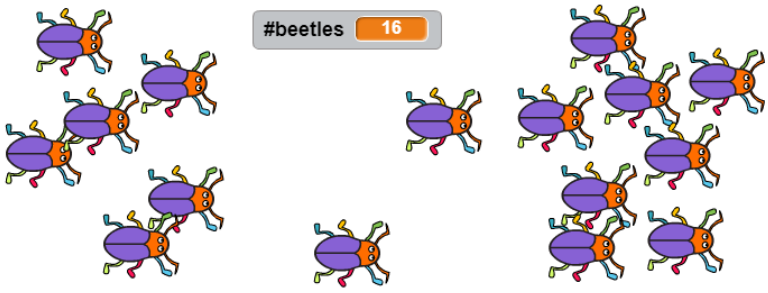
13. Create this script: **When green flag clicked**, **set #beetles to 1**.

14. Add the block **change #beetles by 1** to the **When I start as a clone** script. Now when a clone is created the variable **#beetles** increases by one.

15. In order to make it easier to see the number of beetles on the stage, change the **When I start as a clone** script so that it looks like this. Now the clones appear randomly on the stage instead of on top of one another.



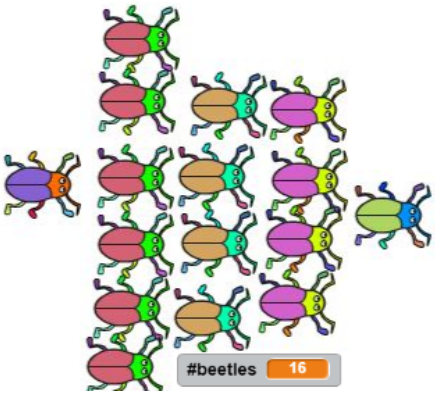
16. Click the green flag. Then press the “c” key repeatedly to see how **#beetles** changes.



17 While each clone is identical and has the same scripts, their behavior and experiences create differences between them. Create this script: **When this sprite clicked**, **change color effect by 25**.

18 Click the green flag. Then press the "c" key so that you have two beetles, the original and a clone. Click on one of them. Notice that a sprite and its clone do not always have to look alike.

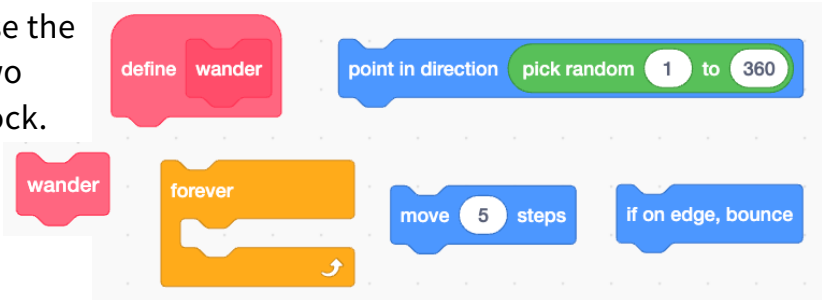
19 Press the "c" key again. Notice the new clones look exactly like the beetles from which they were cloned.



> Must all clones do the same thing as the original beetle? Not quite. They do use the same scripts, but by using randomness, you can have them do different things.

All the beetles and clones point towards the right. To change this, create a script that makes the beetles wander in different directions around the stage:

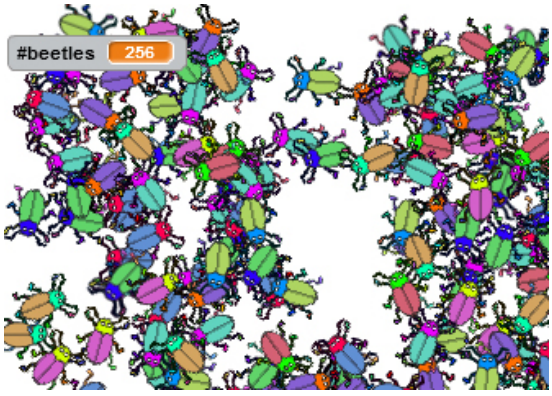
20 Create a new block. Call it **wander**. Use the blocks to the right to define it. Only two of the blocks go inside the **forever** block. Add this new block, **wander**, to the **When I start as a clone** script. Click the green flag. Press the "c" key.



21 Notice the original beetle does not wander around the stage. Find a way to make the original beetle also wander around the stage. What did you do?

22 Try adding the **change color effect by 10** block to the **wander** block definition. What happens?

23 Now click the green flag. You have one wandering beetle. Press the "c" key two times. You have four wandering beetles. Press the "c" key many times. There is a maximum number of clones that Scratch will let you create, but you can have a stage covered with creeping, crawling beetles. Press the stop sign to get rid of the beetles.



24 Clones can be deleted as a result of events that occur while the program is running. Get this new sprite called Frog. Add these blocks to the definition of the **wander** block: **if touching Frog then**, **change #beetles by –1**, **delete this clone**. Does this have to be inside the **forever** block?

25 Test your script. Click the green flag. Then press the "c" key four times. The **#beetles** shows 16 beetles. Watch as all the clones are deleted as they touch the frog. Only the original is left. Watch until the original beetle touches the frog. How many beetles are shown in the **#beetles**? What happened? Why does it not show just 1 beetle?

26 To find out, click the green flag, but do not create any clones. Do not press the "c" key. Watch the original beetle and the **#beetles** as the original beetle touches the frog. Notice the entire time they are touching the **#beetles** decreases. You end with a negative number of beetles. This does not happen when a clone touches a frog. Why? The difference is that the clones delete as soon as they touch the frog, so the **#beetles** only decreases by 1, but when the original beetle touches the frog it is not deleted, and it remains touching for a while. This causes the **#beetles** to decrease repeatedly. You cannot delete the original beetle, but you can hide it so that it stops touching the frog immediately after it first touches it.

27 Use this **hide** block from the **Looks** category in the **if touching Frog then** block. Does the **#beetles** variable count correctly now?

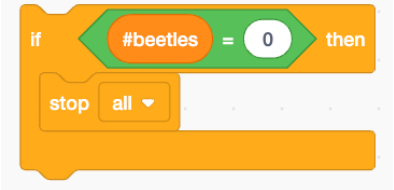
28 Now the original beetle hides when it touches the frog, but its clones are hidden too. To see the clones add a **show** block to the **when I start as a clone script**. Also, **when the green flag is clicked**, the original beetle is not visible on the stage. It is still hidden. Add a block to make it **show** when the green flag is clicked.

29 Click the green flag and press the "c" key 6 times. Does the script work as you expect?

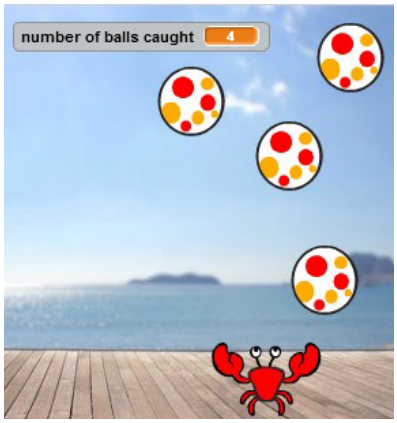
30 The frog stays in one place on the stage. There is more than one way to make it move around the stage. Try using these blocks with random numbers for x and y to make the frog move. Which do you like better? Why?

31 Try adding sound so that a "pop" sound is heard whenever the frog touches the beetle.

32 Notice the frog continues to move around the stage after all the beetles are gone. Use these blocks to make the frog stop moving when there are no beetles left on the screen. Where will you put it in the beetles script? Does it work? Does the frog stop moving when all the beetles are gone?



▶ The Crab and the Beach Balls



When programmers are asked to solve a problem or to write programs for a given project, they are often given task specifications. These might involve an overall description of the project, what the expected inputs and outputs need to be. Programmers are not usually told how to program it. Often their programs must interact with already existing code, so they may be told exactly what the existing code expects as input, what output it produces, etc. Being able to understand specifications and create programs that match the specification is a valuable skill.

Below are your programming specifications for a clone project called ***The Crab and the Beach Balls***. Read all the specifications before you begin working.

Clone Project: *The Crab and the Beach Balls*

Overall Idea:

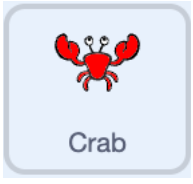
Create a game in which a crab moves back and forth trying to catch beach balls that appear randomly and then fall to the ground.

Game Specifications:

- Choose an appropriate backdrop for your project.
- When the green flag is clicked, the score is reset, and the scripts for the beach ball and the crab are started.
- Your project should have a score box. 

    - A score box shows the crab's score, which starts at 0 at the beginning of the game.
    - A point is scored when the crab touches a beach ball before it falls to the bottom of the stage.
- Your project should have two sprites: A crab and a beach ball.

Crab



1. When the green flag is clicked, the crab's location is near the bottom, center of the stage.
2. Create a way to make the crab move left and right across the bottom of the stage in response to the user's input. It never moves up or down.

I pledge my head to clearer thinking,

my heart to greater loyalty,

my hands to larger service, and

my health to better living,

for my club, my community,
my country and my world.