

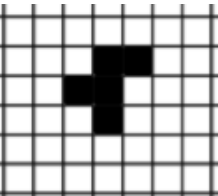
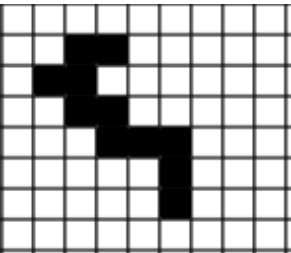
Date(s):	Unit 0 L 1	Conway's Game of Life
Prerequisite Knowledge: Booleans, Conditionals, Coordinates, Snap Custom Blocks/procedural abstraction		
Enduring Understanding: AAP-2 - The way statements are sequenced and combined in a program determines the computed result. Programs incorporate iteration and selection constructs to represent repetition and make decisions to handle varied input values.		
Learning Objective(s): LO AAP-2.B - Represent a step-by-step algorithmic process using sequential code statements LO AAP-2.G - Express an algorithm that uses selection without using a programming language LO AAP-2.M - Create algorithms LO AAP-2.I - Write nested conditional statements and determine the result of nested conditional statements		
Standard: 9-12.CT.8 - Develop a program that effectively uses control structures in order to create a computer program for practical intent, personal expression, or to address a societal issue.		
Differentiation: Pair Programming, Visual Diagramming from previous lab, Pseudocode, starter code		

AIM: How do we code an algorithm with sequential code statements using selection? SWBAT: Write a step-by-step algorithm using nested conditional statements		Notes/Q's/CFUs/CM
WARM UP (3 MINS): <u>DO NOW:</u> Discuss with your partner the CGOL rules. Write it as pseudocode in a format to help you directly translate it into code.		Q: Isn't the rules basically pseudocode? Why do I have to write it over again? A: In a way, yes. However, given your experience programming, rearranging the rules according to specific conditions you are checking would be very helpful when translating it to code. Challenge yourself to get one step closer to real code. Explain it to your group/partner.
LESSON (5 MINS) <u>Teacher</u> - Demo the CGOL program. Refresh the CGOL rules. Direct the code that needs to be completed. <u>Students</u> - Copy the CGOL file and watch the demo of the code and explanation of the helper blocks / starter code.		Demo notes: Demo the CGOL program. Direct which two custom blocks to complete. For Snap make sure to indicate which helper blocks they can use. For python give students code stubs, especially to animate it.
ACTIVITY (30 MINS) <u>Teacher</u> - Students code the countNeighbors and the getNextGenCell custom		

<p>blocks. For Python, students will code more methods but starter code will be provided.</p> <p><u>Students:</u> Pair programming the lab. Discussing and coding Conway's Game of Life algorithm.</p>	<p>CM: Students may not know how many neighbors to check. There are at most 8 neighbors for any specific cell.</p> <p>CM: Any cell outside the border are considered dead cells. Inform students they can use the isValidCell in their code.</p> <p>CM: Students will have to rearrange the rules slightly for the most common/standard solution. Can provide hint that they will need to check if the cell is alive first. Then check the number of neighbors.</p> <p>CM: Other alternative solution to ignore if cell is dead or alive and focus on number of neighbors is not as common and may not be feasible given starter code.</p>
<p>ASSESSMENT (5 MINS)</p> <p>MCQ: Which code below helps count the number of neighbors?</p> <p>Optional MCQ: Which code below helps avoid any errors and accounting for any out-of-bounds error in countNeighbors?</p>	<p>Assessment: Students can code an algorithm for Conway's Game of Life using nested conditionals.</p> <p>If lab takes two lab days, Give the first MCQ to help provide scaffolding to students to finish countNeighbors.</p>
<p>HOMEWORK:</p> <p>Students may finish the coding assignment at home if they need more time.</p>	

Day 2 (if needed)

<p>AIM: How do we code an algorithm with sequential code statements using selection?</p> <p>SWBAT: Write a step-by-step algorithm using nested conditional statements</p>	<p>Notes/Q's/CFUs/CM</p>
<p>WARM UP (3 MINS):</p> <p><u>DO NOW:</u></p> <p>Which helper block will you use to help avoid referencing cells out of bounds?</p> <p>Ans: isValidCell.</p>	<p>Q: What does isValidCell do exactly?</p> <p>A: isValidCell checks if the cell you want to check is within the board. So checks if it's within the 22x28 grid.</p>

<p>LESSON (5 MINS)</p> <p><u>Teacher</u> - Demo the CGOL program. Refresh the CGOL rules. Direct the code that needs to be completed. In order to check out of bounds, use the isValidCell block.</p> <p><u>Students</u> - Students will check over yesterday's work and see if their code uses the isValidCell correctly.</p>	<p>Demo notes: Demo some new interesting patterns.</p> <p>For Snap make sure to indicate which helper blocks they can use.</p> <p>For python give students code stubs, especially to animate it.</p>
<p>ACTIVITY (30 MINS)</p> <p><u>Teacher</u> - Students code the countNeighbors and the getNextGenCell custom blocks. For Python, students will code more methods but starter code will be provided.</p> <p><u>Students</u>: Pair programming the lab. Discussing and coding Conway's Game of Life algorithm.</p>	<p>CM: Students may not know how many neighbors to check. There are at most 8 neighbors for any specific cell.</p> <p>CM: Any cell outside the border are considered dead cells. Inform students they can use the isValidCell in their code.</p> <p>CM: Students will have to rearrange the rules slightly for the most common/standard solution. Can provide hint that they will need to check if the cell is alive first. Then check the number of neighbors.</p> <p>CM: Other alternative solution to ignore if cell is dead or alive and focus on number of neighbors is not as common and may not be feasible given starter code.</p>
<p>ASSESSMENT (5 MINS)</p> <p>MCQ: Given this pattern below, what is the result after 10 generations?</p>  <p>ANS:</p> 	<p>Assessment: Students can code an algorithm for Conway's Game of Life using nested conditionals.</p> <p>If lab takes two lab days, Give the first MCQ to help provide scaffolding to students to finish countNeighbors. Move onto generateNextGenCell on day 2.</p>

HOMEWORK: Students may finish the coding assignment at home if they need more time.	