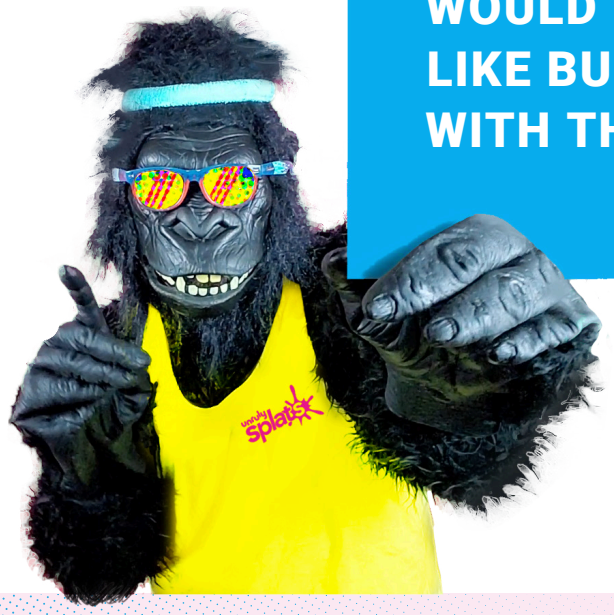
SPLATS GOALIE



GAME SETUP

2–4
PLAYERS

4
SPLATS

GAME SUMMARY

UNRULINESS: Jumping

GAME RULES: Step on the Splat when it turns green to win point

Goalie is a game similar to “whack-a-mole”, but instead of whacking moles you block soccer balls! A ‘ball’, or lit up Splat, will appear randomly on one of four Splats—arranged in a line on the floor. When the green light appears, step on the Splat to block the ball! Students will work together to build and debug the included code.

NOTES

PLAYING TOGETHER

- Smaller groups take turns playing
- Have a designated ‘goalie’ press the Splats to avoid crowding
- Individually play using the web-app

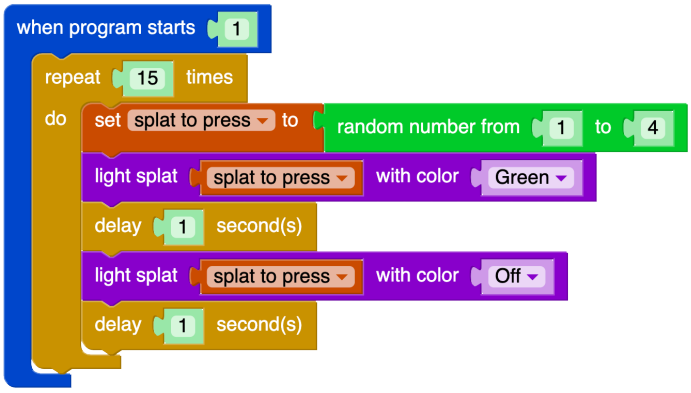
REMOTE PLAY

- Splats web-app
- Virtual breakout rooms
- Have a designated ‘goalie’ click on the Splats that light up green

HOW IT WORKS

This challenge is an opportunity to work with a more complex program and learn to debug code. The code for this activity has bugs built in! Debugging this code will be required to come up with a functioning program, an example of which is included below. Important concepts to understand are nesting, **IF/DO** statements, **REPEAT** loops and **VARIABLES**. The program has two sections—one for making the soccer ball appear and one for scoring.

BASE CODE



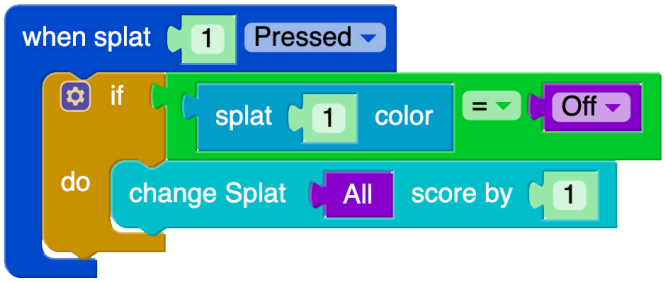
CODE IMAGE: A

To fix the buggy version of the scoring code, first identify the correct way to use a variable to keep track of when the green Splat is pressed. Debugging is a process of documentation, tracking (working backwards to find the source), and reproduction (re-creating the bug to confirm).

This program lights a random Splat green fifteen times by assigning a random number to the variable **SPLAT TO PRESS** in a repeat loop. The corrected code will add a point to the score if the Splat is pressed while green. The **ALL** block changes the total score on all Splats by one.

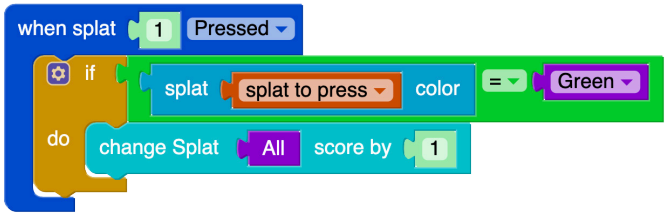
Once you have a **WORKING WHEN SPLAT PRESSED** block for Splat **1**, duplicate it three times by right clicking, or long pressing on the top block. Once copied, change **WHEN SPLAT PRESSED** to **2**, **3**, and **4** to cover scoring for all four Splats!

SCORING WITH BUGS



CODE IMAGE: B

SCORING WITHOUT BUGS



CODE IMAGE: C

SUGGESTED OUTLINE

1 INTRODUCE OUTLINE & KEY CONCEPTS

Introduce the activity. Explain that students will work to debug a program. Introduce the essential blocks and key concepts, go over the first section of code as a class! (Code Image A)

2 WORK TIME: PART ONE

Give time for students to brainstorm how to use the **RANDOM NUMBER** block in the program to generate their soccer 'ball'. Ask groups to build this first section. Project or print out the code for the starting program. Support students as they work together to finish this section. Have groups identify the key goals of this code and the sequence of events. (Code Image B)

3 WORK TIME: PART TWO

Provide groups with the buggy scoring code, included in the Code Key. Walk students through the steps of debugging: documentation, tracking (working backwards to find the source), and reproduction (re-creating the bug to confirm). Have groups work together to debug the code. Review the debugged code as a class and have groups test out their debugged code. Support groups in modifying their final game code to add additional elements. (Code Image C)

4 STUDENT SHOWCASE!

Give groups time to present their games. Have groups explain the key choices they needed to make and their debugging process.

GOING FURTHER

EXTENSION

Encourage groups to modify their game speed, add a second ball to their game, create a variable for the number of Splats, and explore the **IF/DO** block.

SUPPORT

Part 1: Provide clues for block placement. Walk groups through the creation and use of variables.
Part 2: Only locate bugs, ask groups to find the bugs, not fix them. Or, only fix bugs and give groups bug location and ask them to come up with solutions.

CSTA STANDARDS

ALGORITHMS & PROGRAMMING

GRADES 3—5

1B-AP-09 VARIABLES	Create programs that use variables to store and modify data. **(P5.2)**
1B-AP-10 CONTROL	Create programs that include sequences, events, loops, and conditionals. **(P5.2)**
1B-AP-11 MODULARITY	Decompose (break down) problems into smaller, manageable subproblems to facilitate the program development process. **(P3.2)**
1B-AP-12 MODULARITY	Modify, remix, or incorporate portions of an existing program into one's own work, to develop something new or add more advanced features. **(P5.3)**
1B-AP-15 DEVELOPMENT	Test and debug (identify and fix errors) a program or algorithm to ensure it runs as intended. **(P.6.1, 6.2)**
1B-AP-17 DEVELOPMENT	Describe choices made during program development using code comments, presentations, and demonstrations. **(P.7.2)**

CSTA STANDARDS

ALGORITHMS & PROGRAMMING

GRADES 6—8

2-AP-11 VARIABLES	Create clearly named variables that represent different data types and perform operations on their values. **(P5.1, 5.2)**
2-AP-12 CONTROL	Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals. **(P5.1, 5.2)**
2-AP-13 MODULARITY	Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. **(P3.2)**
2-AP-17 DEVELOPMENT	Systematically test and refine programs using a range of test cases. **(P6.1)**