MEMORY CHALLENGE

TEST YOUR MEMORY!

GAME SETUP

2–4
PLAYERS
🟦🟦⬜⬜

6
SPLATS
🟦🟦🟦🟦🟦🟦

GAME SUMMARY

UNRULINESS: Sitting, Walking

GAME RULES: Use variables and functions to keep track of which Splats lit up green

Students will create a memory game using functions. Six Splats are placed on the ground or tabletop. Two will light up quickly, then disappear. Press on the two Splats you think you saw!

NOTES

PLAYING TOGETHER

- Small groups spaced apart can play simultaneously
- Students can wear gloves or use objects to press the Splats in order to avoid spreading germs
- This activity can also be played via the web-app

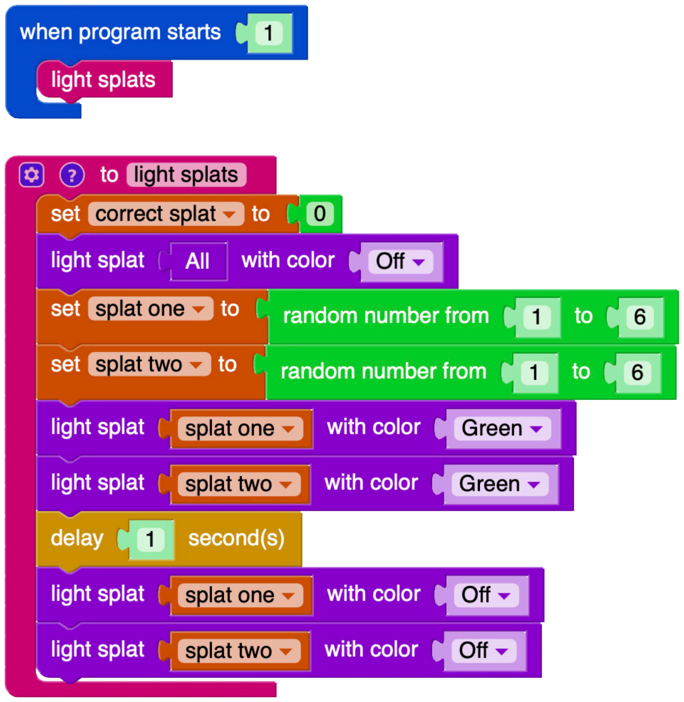
REMOTE PLAY

- Splats web-app
- Virtual breakout rooms

HOW IT WORKS

PART ONE

Functions are a way to package code that performs a specific task. In the first part of this program the function, **LIGHT SPLATS**, randomly lights up two out of six Splats green for 1 second and then turns them off. Functions are named by what they do. The purposes of using a function are: to simplify the number of rules a program has to process each time it runs, and to help make the program easier to debug, to read, and understand.



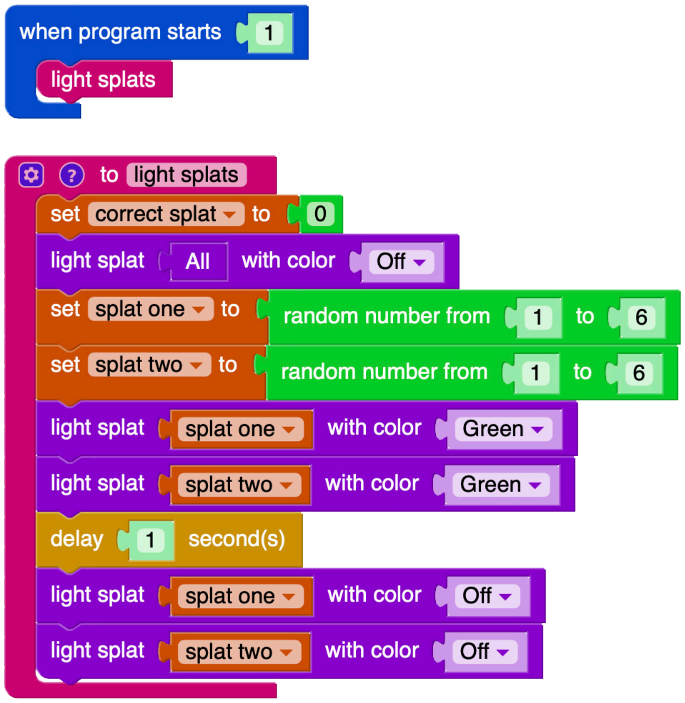
```
when program starts 1
    light splats

to light splats
    set correct splat to 0
    light splat All with color Off
    set splat one to random number from 1 to 6
    set splat two to random number from 1 to 6
    light splat splat one with color Green
    light splat splat two with color Green
    delay 1 second(s)
    light splat splat one with color Off
    light splat splat two with color Off
```

CODE IMAGE: PART 1

PART TWO

In the second part of this program, by using two conditional statements—**IF/DO** blocks—we create rules so that if the green Splats were correctly remembered and pressed, all six Splats light up green–and students know they beat the memory challenge.

Variables are used in order to know whether or not the Splats pressed were the correct green Splats. We use the variable **SPLAT ONE** and **SPLAT TWO** to keep track of which random 2 out of the 6 Splats turned green. The variable **CORRECT SPLAT** keeps track of whether both **SPLAT ONE** and **SPLAT TWO** were pressed. Note that in the **LIGHT SPLATS** function, **CORRECT SPLAT** is always reset to 0, to make the next round begin.



```
when program starts 1
    light splats

to light splats
    set correct splat to 0
    light splat All with color Off
    set splat one to random number from 1 to 6
    set splat two to random number from 1 to 6
    light splat splat one with color Green
    light splat splat two with color Green
    delay 1 second(s)
    light splat splat one with color Off
    light splat splat two with color Off
```

CODE IMAGE: PART 2

SUGGESTED OUTLINE

1 **INTRODUCE OUTLINE & KEY CONCEPTS**
Demonstrate how to play the game at the front of the room. Have students brainstorm the essential blocks, key program needs, and compare algorithms.

2 **WORK TIME: PART ONE**
Review the essential blocks and tie them directly to the game rules. Highlight the tole of functions and variables this program and support groups in breaking down the program development process.

3 **WORK TIME: PART TWO**
Support groups in building their programs and rotating through essential roles: coding, testing, debugging, documentation, etc. Encourage groups to test each others programs and iterate based on feedback.

4 **STUDENT SHOWCASE!**
Give groups time to play and present their games, describing key decisions and modifications.

GOING FURTHER

EXTENSION
Ask groups to come up with ways to keep the variables **SPLAT ONE** and **SPLAT TWO** from being assigned to the same Splat, ensuring that two different Splats always light up.

SUPPORT
The game can be slowed down by increasing the time on the **DELAY** blocks. Provide students with the structure of the full program. For additional help, have groups write out all the needed steps on paper first.

CSTA STANDARDS

ALGORITHMS & PROGRAMMING

GRADES 6—8

2-AP-10 ALGORITHMS	Use flowcharts and/or pseudocode to address complex problems as algorithms. **(P4.4, 4.1)**
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)**
2A-AP-14 MODULARITY	Create procedures with parameters to organize code and make it easier to reuse. **(P4.1.4.3)**
2A-AP-17 DEVELOPMENT	Systematically test and refine programs using a range of test cases. **(P6.1)**
1A-AP-19 DEVELOPMENT	Document programs in order to make them easier to follow, test, and debug. **(P7.2)**