RELAY RACE

RUN, JUMP, AND GET UNRULY!

GAME SETUP

2–4
PLAYERS

2
SPLATS

GAME SUMMARY

UNRULINESS: Running

GAME RULES: Jump on Splat 1 until it turns green, step on splat 2 to finish

This activity is an adaptation of the six Splat example game, Relay Race. Place two Splats running distance apart. Students in each group take turns running across the room to stomp on their Splat until it turns green. Once green, they race back to jump on the other Splat at the finish line to get a point and it is the next student's turn. A relay race! A stopwatch keeps track of the time it takes to get 10 points and students compete for the fastest time. Groups first work independently, then come together for a head-to-head competition.

NOTES

PLAYING TOGETHER

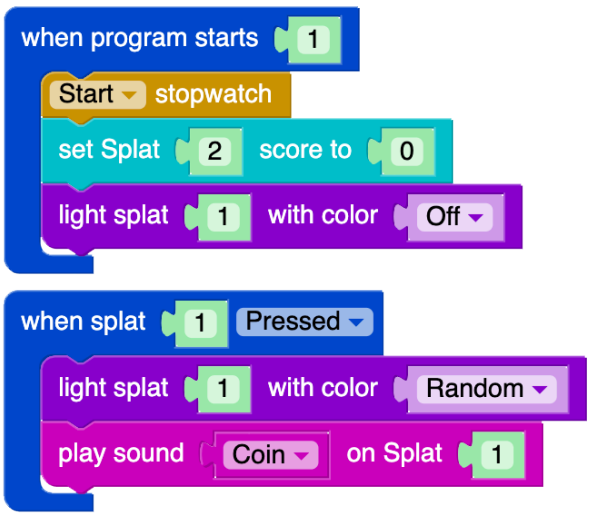
- Smaller groups take turns playing
- Use tape to mark where students can stand while waiting for their turn
- Create 'lanes' using tape or yardsticks

REMOTE PLAY

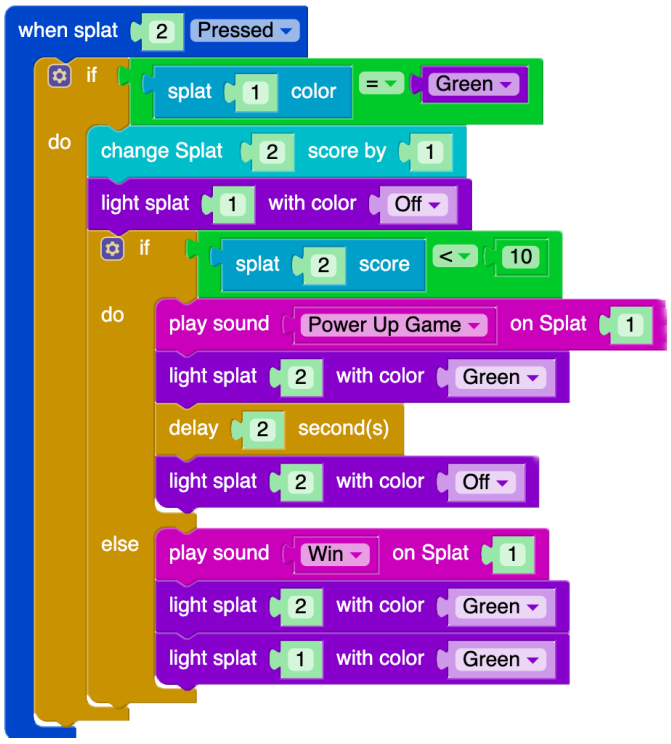
- Splats web-app
- Virtual breakout rooms
- Students click on the Splats instead of stomping

HOW IT WORKS

This program uses nested conditional statements. In a conditional statement, if the condition is true the program will do one thing, if false it will do another. Nested statements allow for complex logic, because any blocks inside of conditional statements only run if the conditions for the outer statement are met.



When this program begins, a stopwatch starts. Splat 1 lights up a random color when Splat 1 is pressed. Once the random color is green, players run to jump on Splat 2 to get a point. Conditional statements (**IF/DO** and **IF/DO/ELSE**) are used to add points and end the game when the group reaches 10 points. An **IF/DO** block creates a rule for the program that happens if the **IF** condition is true. In an **IF/DO/ELSE** block, an **ELSE** rule is made for the scenario where the condition is not true. The first conditional statement is, **IF** Splat 1 is pressed while green, change the Splat score by one. The second conditional statement is, **IF** Splat 2 is pressed **AND** the score is less than 10, light Splat 1green and the group keeps playing. **ELSE**, once the score is equal to 10, a sound is played and the game ends.



SUGGESTED OUTLINE

1 INTRODUCE OUTLINE & KEY CONCEPTS

Introduce the activity, explain how the game will be played, and demonstrate game play. Have groups brainstorm potential algorithms and the key elements for this program. Have groups write down an ordered list of rules for the game or create a flowchart.

2 WORK TIME: PART ONE

Lead a class brainstorm about how to start the program. Review the code for **WHEN PROGRAM STARTS** and when splat 1 pressed as a class! Give groups time to build their code for the first two starting blocks. Assist groups in working through the **WHEN SPLAT 2 PRESSED** code. Have groups break down every element that needs to be included. Talk about the role of nesting in programs.

3 WORK TIME: PART TWO

Support groups in rotating through different roles: coding, testing, debugging, documentation, etc. Give groups time to play with their code. Review code writing with the class.

4 STUDENT SHOWCASE!

Give groups time to present their code and describe key choices. If possible, have groups compete against each other in a full-class game, or play the biggest Relay Race you can!

GOING FURTHER

EXTENSION

This can be an independent activity. Encourage students to build the program in different ways.

SUPPORT

Provide students with the essential blocks and ask them to decide the order, in groups or as a class. Alternatively, provide the whole code to students and ask them to discuss what it does and how it works. If game play is too long, change the number of points needed to win from 10 to 3. You can also modify the example as a full class discussion.

CSTA STANDARDS

ALGORITHMS & PROGRAMMING

GRADES 3—5

1B-AP-08 ALGORITHMS	Compare and refine multiple algorithms for the same task and determine which is the most appropriate. **(P6.3, 3.3)**
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-16 DEVELOPMENT	Take on varying roles, with teacher guidance, when collaborating with peers during the design, implementation, and review stages of program development. **(P2.2)**
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-10 ALGORITHMS	Use flowcharts and/or pseudocode to address complex problems as algorithms. **(P4.4, 4.1)**
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)**
2-AP-19 DEVELOPMENT	Document programs in order to make them easier to follow, test, and debug. **(P7.2)**