ADVENTURE SPLATS

GAME SETUP

2–4

PLAYERS

■■□□

5–6

SPLATS

■■■■■□

RESCUE THE LOST SPLAT!

GAME SUMMARY

UNRULINESS: Walking

GAME RULES: Each player, grab and hold a button of Splat one to begin the game. Together, pick up Splats 2-4 by pressing and holding their buttons. Without dropping any buttons, reach the finish Splat.

Two to four students, each holding a button of Splat one, have to work as a team to pick up and rescue the Splats scattered around the room. Players to pick up each ‘Lost Splat’ one at a time, holding each very carefully, until they make it to the finish Splat!

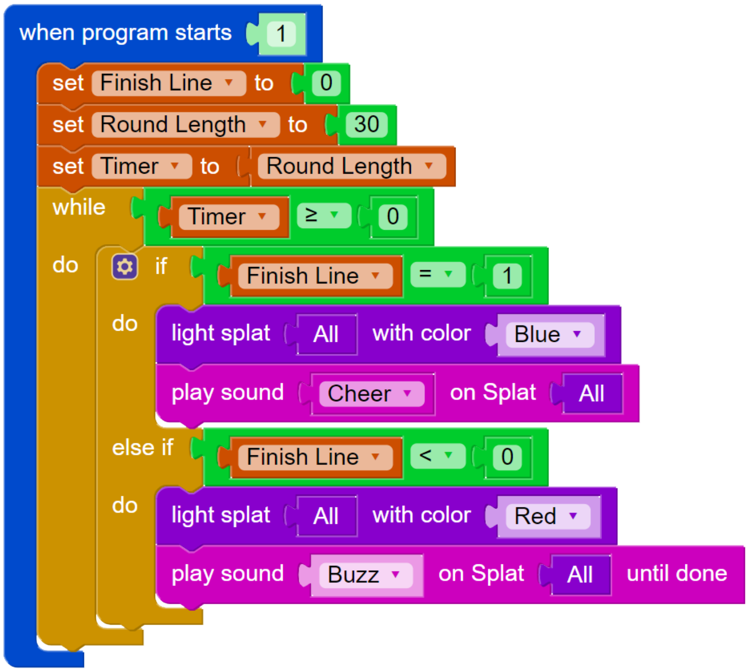
HOW IT WORKS

PART ONE

This program features a **WHILE/DO** conditional block, as well as an audible timer using the **TIMER** variable, used later inside of a **SAY** block. When the program starts, the game ending variable **FINISH LINE** is set to zero to start the game, the **ROUND LENGTH** is set to some number of seconds, and the **TIMER** variable is set to match it.

Once the variables are set up, the **WHILE/DO** block is used to monitor that the **TIMER** variable hasn't reached zero, ending the game if it does.

While the game is running, the **FINISH LINE** variable is tracked. If the finish Splat is pressed, and the **FINISH LINE** variable is set to one, the game will end with a cheer! If any of the Splat buttons are released, and **FINISH LINE** becomes a negative number, the game ends with a buzz.



```
when program starts 1
  set Finish Line to 0
  set Round Length to 30
  set Timer to Round Length
  while Timer ≥ 0
  do
    if Finish Line = 1
    do
      light splat All with color Blue
      play sound Cheer on Splat All
    else if Finish Line < 0
    do
      light splat All with color Red
      play sound Buzz on Splat All until done
```

CODE IMAGE: PART 1

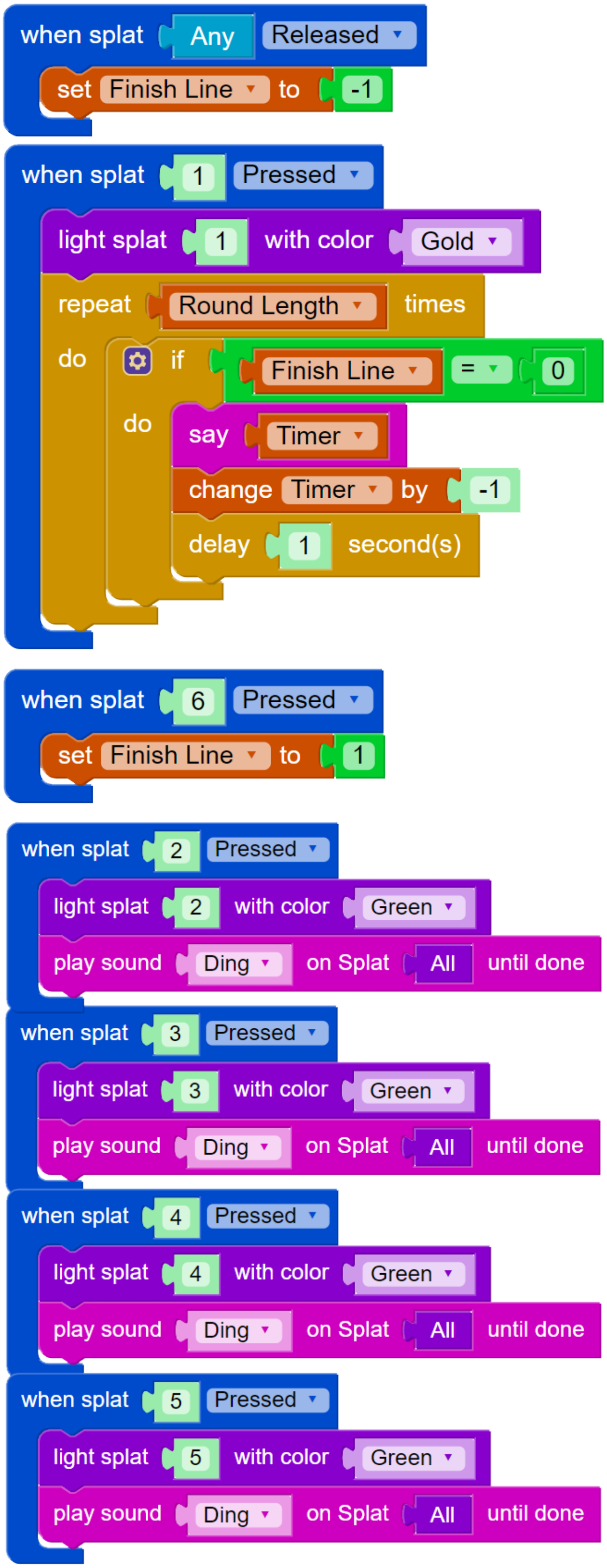
PART TWO

Connecting the code to the game rules: the **WHEN SPLAT RELEASED** block is triggered if any Splat is dropped, setting **FINISH LINE** to a negative number and triggering the buzz sound from part one while stopping the timer.

The round timer begins counting when Splat 1 is pressed. While the game hasn't finished, the **REPEAT** block announces the time remaining by using the **SAY** block. Each repeat takes a second off of the **TIMER** variable, and continues counting down for however long the **ROUND LENGTH** variable was set to in the when program **STARTS** block.

Splats 2 through 5 only need to show the players that they have been picked up, by lighting up and making a sound.

Splat 6 acts as the finish line. When pressed, it sets the **FINISH LINE** variable to one, ending the round with a win!



```
when splat Any Released
    set Finish Line to -1

when splat 1 Pressed
    light splat 1 with color Gold
    repeat Round Length times
    do
        if Finish Line = 0
        do
            say Timer
            change Timer by -1
            delay 1 second(s)

when splat 6 Pressed
    set Finish Line to 1

when splat 2 Pressed
    light splat 2 with color Green
    play sound Ding on Splat All until done

when splat 3 Pressed
    light splat 3 with color Green
    play sound Ding on Splat All until done

when splat 4 Pressed
    light splat 4 with color Green
    play sound Ding on Splat All until done

when splat 5 Pressed
    light splat 5 with color Green
    play sound Ding on Splat All until done
```

CODE IMAGE: PART 2

SUGGESTED OUTLINE

1 **INTRODUCE EXERCISE**
Explain the game rules and demonstrate how to play the game. Lead groups in identifying the objectives of their program and planning first steps. Have students share their sub-tasks and objective lists with the class.

2 **GUIDED WORK TIME**
Introduce the essential blocks and tie them directly to the game rules. Give groups time to brainstorm different ways to track game elements like round length, when a Splat is picked up or dropped, and when the game is over.

3 **GROUP WORK TIME**
Support groups as they build their programs. Ensure group members are sharing note taking, testing, and coding roles.

4 **STUDENT SHOWCASE!**
Let the games begin! Have each group connect to Splats one at a time and complete the adventure!

GOING FURTHER

EXTENSION
Students can add additional challenges to this adventure, such as time restrictions and obstacles between each Splat.

SUPPORT
Have students draw a diagram of how the game will be played, identifying the role each Splat will play in the game.

COMPUTER SCIENCE TEACHERS ASSOCIATION STANDARDS

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-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-13 **DEVELOPMENT**	Use an iterative process to plan the development of a program by including others’ perspectives and considering user preferences. **(P.1.1, 5.1)**
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-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. **(P3.2)**
2-AP-19 **DEVELOPMENT**	Document programs in order to make them easier to follow, test, and debug. **(P7.2)**