Bitesize

GCSE Computer Science

Creating Battleship: using arrays to create a game of Battleship

In the following example, we will look at how a two-dimensional array can be used to represent Battleship, the board game by Hasbro.

Battleship is a game which is played on a two-dimensional grid. Each point on the grid represents a space on a map of the sea where a ship might be. Two players plot their ships on a grid in secret and then take it in turns to guess where their opponent has plotted their ships.

This activity includes pseudocode, followed by explanations about how to code each step in JavaScript, followed by a complete coded solution in JavaScript and Python 3.

NOTE: in our example we will only show how one player can place a ship and how another can try to sink it.

Pseudocode

We are going to create a game which follows this pseudocode:

1. *create a 9 by 9 array called 'game'*
2. *ask the user to input the column where their boat starts – store this value as 'x'*
3. *ask the user to input the row where their boat starts – store this value as 'y'*
4. *ask the user to say if they want the boat to be horizontal (h) or vertical (v)*
5. *convert all inputs from character strings to numbers*
6. *the boat that is created is called '4'*
7. *set the length of the boat to be 4*
8. *if h is selected:*
    - *counting from x+0 to x+3 using 'c', put a '4' on the game grid at [y][c]*
9. *if v is selected:*
    - *counting from y+0 to y+3 using 'c', put a '4' on the game grid at [c][x]*
10. *display the result*
11. *the second user attempts to locate the boat by entering 'x' and 'y' coordinates*
    a. *if a '.' Is found at the selected coordinates then a 'miss' is output*
    b. *otherwise if a '*' is found then a 'boat already hit' is output*
    c. *otherwise if a '4' is found then a 'hit' is output and the cell value is changed to '*'*
12. *check to see if the whole boat has been sunk:*
    a. *if yes then game over*
    b. *otherwise continue*

The solution

A. Creating the grid

The following code examples should work if copy and pasted into an html document and executed in a web browser:

To declare this array in JavaScript, we could use the code:

```
char game [9][9];
```

We can declare *and* initialise an array as follows- by giving it some values when we create it. For example:

```
char game [9][9] = {
{'.','.','.','.','.','.','.','.','.'},
{'.','.','.','.','.','.','.','.','.'},
{'.','.','.','.','.','.','.','.','.'},
{'.','.','.','.','.','.','.','.','.'},
{'.','.','.','.','.','.','.','.','.'},
{'.','.','.','.','.','.','.','.','.'},
{'.','.','.','.','.','.','.','.','.'},
{'.','.','.','.','.','.','.','.','.'},
{'.','.','.','.','.','.','.','.','.'}
};
```

This would fill every row and every column with a full stop.

B. Adding ships

The above code establishes the grid but what if we want to add some ships? The following example shows the grid with a ship called 1 and another ship called 5.

	0	1	2	3	4	5	6	7	8
0	1	.	.	.	.	.	.	.	.
1	.	.	.	.	.	.	.	.	.
2	.	.	.	.	5	5	5	5	5
3	.	.	.	.	.	.	.	.	.
4	.	.	.	.	.	.	.	.	.
5	.	.	.	.	.	.	.	.	.
6	.	.	.	.	.	.	.	.	.
7	.	.	.	.	.	.	.	.	.
8	.	.	.	.	.	.	.	.	.

The one in the top left of the grid represents a ship which is just the size of one grid space. It would be created with the following line of code:

```
game [0][0] = '1';
```


This saves a '1' in the **array** at the position referenced by the index [0][0].

The 5s represent a bigger ship. Bigger ships (several characters in one go) can be created by manually assigning each character or by using a FOR loop (FOR loops are explained in more detail in the Algorithms and control flow Study guide). The following code creates a ship called '5' which is five data points long.

```
for (c = 4; c < 9; c++)
        {game[2][c] = '5';}
```

It will be positioned along row number 2 and will have a '5' character under each column from 4 to 8. The column value will be referred to as c.

The above code is equivalent to this:

```
game[2][4]  = '5';
game[2][5]  = '5';
game[2][6]  = '5';
game[2][7]  = '5';
game[2][8]  = '5';
```

By using a loop, we save ourselves the need to create a line of code for each value. As you can see from the equivalent code, 'c' is used to count from 4 to 8. Using a loop is a good skill, and what is meant will be clear to other programmers who see the code. The nine in the loop control (c<9) could be changed to create shorter ships.

C. Finding the enemy ships

When the time comes to try to find the other player's ships, inputs can be used again to specify x and y coordinates.

We can then use an IF statement to see if we have 'hit' a ship. (For more about Boolean data and IF statements, see the Boolean logic Study guide.)

For example:

```
if (game[y][x] != '.')
        { printf("Kaboom! You hit a ship."); }
else
        { printf("Missed")}
```

The full JavaScript code

The following code will execute the game in a web browser if saved as an html document.

```
<!doctype html>
<html>
<head>
        <meta http-equiv="X-UA-Compatible" content="IE=9" />
</head>
<body>
<h1> Battleship </h1>
</body>
<style>
  body { background-color: #eff; }
</style>
<script>

var game = [
        ["·","·","·","·","·","·","·","·","·","·"],
        ["·","·","·","·","·","·","·","·","·","·"],
        ["·","·","·","·","·","·","·","·","·","·"],
        ["·","·","·","·","·","·","·","·","·","·"],
        ["·","·","·","·","·","·","·","·","·","·"],
        ["·","·","·","·","·","·","·","·","·","·"],
        ["·","·","·","·","·","·","·","·","·","·"],
        ["·","·","·","·","·","·","·","·","·","·"],
        ["·","·","·","·","·","·","·","·","·","·"],
        ["·","·","·","·","·","·","·","·","·","·"],
      ];

var board = document.createElement("PRE");  // Preparing the HTML <pre> element to display the board on the page
document.body.appendChild(board);

var button=document.createElement("BUTTON");        //        Preparing the "Fire! button to allow the player to fire at the ship
button.onclick = fire;                                                                        //        Clicking the button calls the fire() function
var t=document.createTextNode("Fire!");
document.body.appendChild(button);
button.appendChild(t);


//        Function to return a HTML code string to display the board within the <pre> element defined above
function drawBoard() {
        var boardContents = "";
        var i;
        var j;
```



```
	for (i=0; i<9; i++) {
		for (j=0; j<9; j++) {
			boardContents = boardContents + game[i][j]+" "; // Append array contents
for each board square
		}
		boardContents = boardContents + "<br>"; // Append a line break at the end of each
horizontal line
	}
	return boardContents; // Return string representing board in HTML

}

board.innerHTML = drawBoard(); // Display the board on the page using the above function

var x=prompt("Where would you like to place your cruiser? Enter an X coordinate: (0-8)");
var y=prompt("Where would you like to place your cruiser? Enter a Y coordinate: (0-8)");
var direction=prompt("Place (h)orizontally, (v)ertically");

x = Number(x);  // Convert the string returned by "prompt" into a number
y = Number(y);  // Convert the string returned by "prompt" into a number

//	Draw cruiser horizontally
if (direction[0] == "h") {
  var c;
  for (c = x; c < (x + 4); c++) {
     game[y][c] = '4';
  }
}

//	Draw cruiser vertically
if (direction[0] == "v") {
  var c;
  for (c = y; c < (y + 4); c++) {
     game[c][x] = '4';
  }
}

board.innerHTML = drawBoard(); // Redraw board with cruiser added

//	Function for firing a shot when the "Fire! button is pressed
function fire() {
	var fireX=prompt("Where would you like to fire? Enter an X coordinate: (0-8)");
	var fireY=prompt("Where would you like to fire? Enter a Y coordinate: (0-8)");

	fireX = Number(fireX);	//	Convert the string returned by "prompt" into a number
	fireY = Number(fireY);	//	Convert the string returned by "prompt" into a number

	if (game[fireY][fireX] == ".")	{	//	Check if the specified coordinate is occupied
by the cruiser
```



```
                alert("Missed.");
        } else if (game[fireY][fireX] == "*") {
                alert("You already hit the ship there.");
        } else {
                alert("Kaboom! You hit a ship");
                game[fireY][fireX] = "*";
                board.innerHTML = drawBoard();          //      Redraw board with hit marker at
specified coordinate
        }

        var shipfound;
        var i;
        var j;
        //      Check if there are any ships remaining on the board
        for (i=0; i<9; i++) {
                for (j=0; j<9; j++) {
                        if (game[i][j] != "." && game[i][j] != "*") {
                                shipfound = true;       //      Set to true if a ship is found
                        }
                }
        }

        if (!shipfound) { //     If no ships are found end the game
                alert("All ships have been sunk. Well done Captain! Game over");
                document.body.removeChild(button);  //      Remove the fire button from the
page after game over
        }
}

</script>
</html>
```

Note:

- The board.inner HTML = draw Board() section of the code is used to display the result of the program. This calls the drawBoard function that runs a FOR loop to count through each point of the array and displays the results that have been produced by the code before.
- The column of the array is represented by i and the row is represented by j. i and j are both used as counters from 0 to 8. This gives us a pair of numbers which start off as (0,0) and finish off as (8,8). These pairs represent co-ordinates on the grid.
- x and y are used for placing a battleship.
- i and j are solely used for working through the grid for display.

The full Python code

Be careful to follow the indentation if copying this code into a Python IDE.

```
#!/usr/bin/python
#
# Make the game board, fill each place with '.'
#



def getInt(prompt):
    while True:
        try:
            result = int(input(prompt +" >>> "))
            return result
        except Exception: # error, don't care which - just report
            print ("No bad data, try again : ")

def getString(prompt):
    while True:
        try:
            result = str(input(prompt +" >>> "))
        except Exception: # error, don't care which - just report
            print ("No bad data, try again : ")
        else:
            if result =='v' or result == 'h':
                return result

game= [["." for x in range(9)] for y in range(9)]

# place a ship
print ("Where would you like to place your cruiser? (length = 4)")

x = getInt("Enter an X coordinate: ")
y = getInt("Enter a Y coordinate: ")

direction = getString("Place (h)orizontally, (v)ertically?")

if direction.startswith('h'):
    for a in range (x, x + 4):
        game[y][a] = '4'
elif direction.startswith('v'):
    for a in range (y, y + 4):
        game[a][x] = '4'

#Play the game - keep looping until all ships in the board have been
hit
gameover = False
while not gameover:
    print ("Where would you like to fire?")
    print ()
```



```
    firex = getInt("Enter an X coordinate: ")
    firey = getInt("Enter a Y coordinate: ")

    if game[firey][firex] == '.':
        print ("Missed.")
    elif game[firey][firex] == '*':
        print ("You already hit, the ship fired  there.")
    else:
        game[firey][firex] = '*'
        print ("Kaboom! You hit a ship")

    shipfound = False
    for plane in game:
        for row in plane:
            for char in row:
                if not (char == "." or char == "*"):
                    shipfound = True

    if not shipfound:
        gameover = True

print ("All ships have been sunk. Well done Captain! Game over")
```

Extension

The code above only allows one player to place a ship, and another player to try to sink it. Extend the code to allow two players to place a ship and take turns sinking each other's ships.