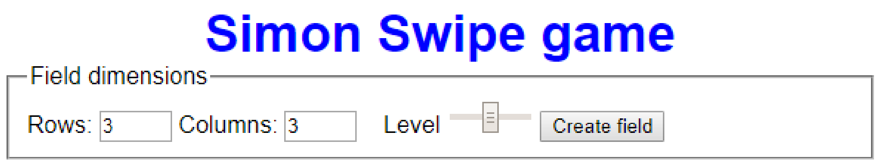
SIMON SWIPE GAME

REQUIREMENTS

The source code of the following web page is provided:

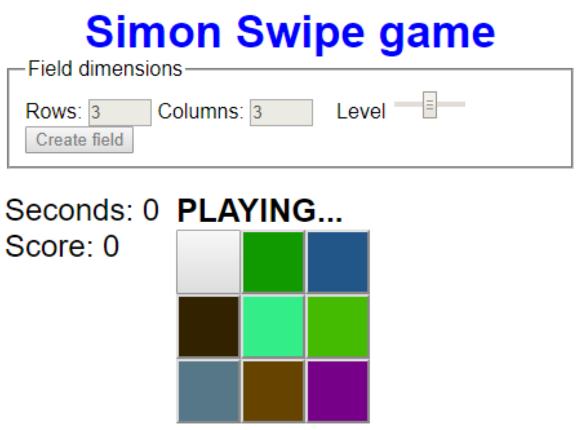


The user can choose the number of rows and number of columns for the game between a minimum value of (2) and a maximum of (5).

The difficulty of the game is selected with the slider. There should be three levels of difficulty: 1, 2, and 3. It controls the speed the sequence is shown and the number of elements of the sequence.

Once the size of the field and the level of difficulty of the game are chosen, the player clicks the button “Create field” The result of clicking this button will be:

- The selection controls for number of rows, columns, level and the “Create field” button will be disabled.
- A stopwatch, counting the number of seconds the game lasts until the player wins or loses, will be displayed in the div *crono*
- A score counter must appear in the div *score*
- The message “Playing... “ must be displayed in the div *message*
- The field will be automatically generated according to the options selected by the player.
- The first sequence will be played on the board, showing the first element of the sequence

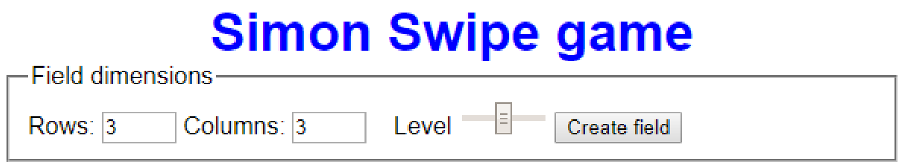


The game consists of repeating the sequence that has been played, by clicking in order on the cells that have been red lighted:

- If the sequence is repeated successfully, the score is incremented by one and the sequence is played again adding a new element at the end.
- If the sequence is not successfully repeated, an error message will appear on the screen and the game is over. The message "Ohhh!!! You lost" written in the colour red will be displayed in the div *message,* the stopwatch stops and all the events are disabled.

STEP ONE:

Build the initial form webpage with HTML and CSS



Simon Swipe game

Field dimensions

Rows: 3 Columns: 3 Level

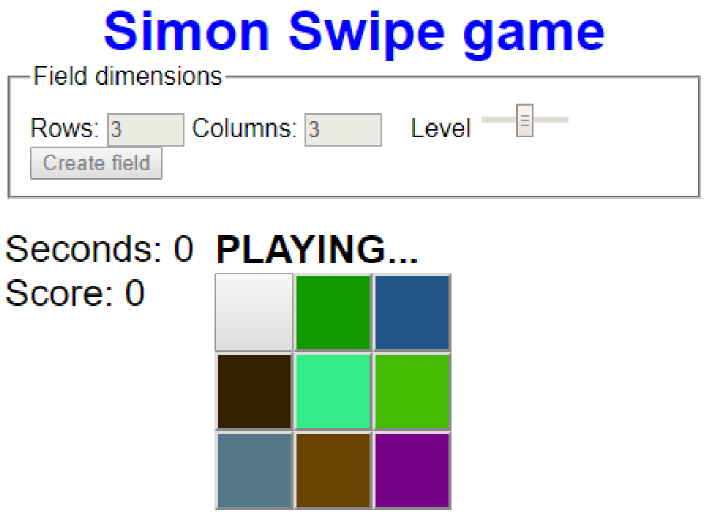
- Use Number-type input controls for the *Rows* and *Columns* fields. Minimum value is 2, the maximum value is 5, and the default value is 3
- Use Range-type input control for the *Level* slider. Min value is 1, max value is 3, and the default value is 2. The step size will be 1
- The *Create field* button's type is Button. Add an eventListener for the click event of the button, that will call the function startGame()
- Create a *message* div, a *counters* div on the left, with a *crono* and a *score* div inside, and a *field* div to store the board

Create the startGame() function:

- Disable fieldset controls (Rows, Columns, Level and the button)
- Show a message PLAYING… in message div
- Show the Seconds and the score in the counters div
- Draw the field with the size chosen:
  - Use a function drawField(width, height)
  - Use a double *for* looping to draw the grid
  - Use button-type input buttons with a size of 50px by 50px. Give each button an identifier with a value containing the coordinates (x,y) of where it is located in the field. Ex.

    <input type="button" id="1_2" style="width:50px; height:50px> is the button located in the second row and third column. (The coordinates begin in (0,0))

- Give a different background color to each of the buttons. You can do this with an array of colors or by adding a constant to the previous color



- Test the game so far:
  - Use a function playGame(width, height)
  - Assign to each button representing a square the event “onmousedown”:
    - Use a function enableEventsField for adding the event listeners to the buttons of the field
    - Use getElementsByName to obtain the array of all buttons of the field (Assign to all of them the same value for the name property)
    - Go over the array generated with a loop and assign the event onMouseDown to each button. You could also assign the event to the container div and use the target property

```
buttons[i].addEventListener('mousedown', check)
```

  - Use a function disableEventsField for disabling the event listeners to the buttons of the field

- Create the function called by the event mousedown (check()):

  - In order to test the app so far, every time a button is clicked, a message “The button (x, y) has been clicked will be shown. This procedure will be expanded in step 2:

    - Every event produces an argument with information about the event. Use the target id to get the id of the button clicked. Use the function split to get the two coordinates (Ex. If the id is 1_2)

```
var point = e.target.id.split('_');
```