Overview

In my first project for the General Assembly Software Engineering Flex course, I was given the task of building an arcade game. I decided to choose to create Tetris for two reasons. Firstly, I was keen to test myself and, secondly, I am a fan of the game.

Brief

- Render a game in the browser

- Use JavaScript, CSS and HTML

- Use JavaScript for DOM manipulation

- Deploy the game using GitHub pages

Game Requirements

- The game should stop if a Tetrimino fills the highest row of the game board

- The player should be able to rotate each Tetrimino about its own axis

- If a line is completed it should be removed and the pieces above should take its place

- The game should stop if a Tetrimino fills the highest row of the game board

- The player should be able to rotate each Tetrimino about its own axis

- If a line is completed it should be removed and the pieces above should take its place

Technologies Used

- HTML

- CSS

- JavaScript (ES6)

- Git

- GitHub Pages

Method Undertaken

- When a start button is pressed a 10 x 20 grid is created

- I created 200 divs (variable name is cell) and added a class to them to create my grid. Each cell is then added to a array called cells.

- I also created a score, line cleared count, top score using localstorage and a display showing what level you wre currently on

- I created my Tetriminos by setting the axis point of the Tetrimino as the current location and then created the other three cells from that point. The axis was always the second entry in the array.

Text

Description automatically generated

Before the game starts, I need to create the first shape for the main grid and the next shape

I used a setTimeout to move the Tetrimino down the grid, the speed of the intervals between movements vary depending on the level you are on

The Tetrimino is added to the grid by assigning the specific class to that shape, for example, ‘jshape’

Everytime the setTimeout method completes a cycle the shape is removed from the grid (class removed from those cells) and re-added one row below (class added to the new cells)

If arrowleft, arrowright or arrowdown are pressed the Tetrimino will move in the direction pressed on the keyboard, using the same method as stated above.

If arrowUp is pressed one will be added to the current rotation of the shape and the next entry in shapes array will be used to add the class to the cells

Everytime a shape is removed, there are checks to see whether the future position of the Tetrimino is allowed and whether the current position but one row down is allowed. If both of these checks find any cells with the full class we mark the current Tetriminos as full and create a new shape. If the shape can‘t move right or left or be rotated but can be moved down it continues its descent down the grid.

Text

Description automatically generated

When the shape reaches the bottom of the grid or can’t move down past another Tetrimno the class ‘full’ is added to the cell and a new shape created.

Line Completion

I split the cells array into chunks of 10 which signifies a line in my grid, then a for loop iterates through each row of the chunks of 10 cells, if all the cells contain the class ‘full’ the row of cells are removed from the array and the line of divs are removed from the grid. I then used unshift to add the cells back to the array and prepend to add the divs back to the grid.

Text

Description automatically generated

Score/Level/Top Score/LinesCleared

Everytime a line is removed the score, and lines cleared is updated. If you get the top score this is updated as soon as you surpass it. The level is linked to how many rows you have cleared.

Screenshots Final Product

A picture containing diagram

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

Diagram

Description automatically generated with medium confidence