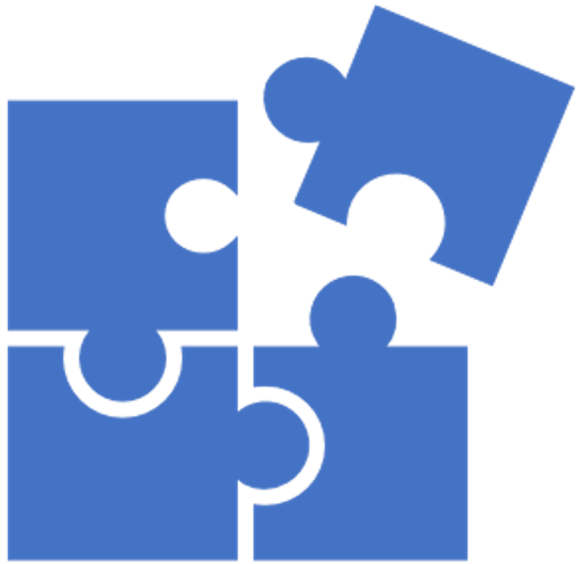


TILESLIDE MASERMIND:

FIFTEEN & EIGHT PUZZLE

BY:

India Jackson, Kiki Ajibade, Asma Syeda, Natania Tedla

Problem

FIFTEEN PUZZLE

The problem for this project is to design ‘Fifteen puzzle game”. This a web-based sliding puzzle game with a 4x4 grid with each puzzle piece numbered from 1-15.

The game is played by rearranging the puzzle pieces from a shuffled state until all puzzle pieces are arranged in ascending order.

Therefore our goal for the game is to design an efficient working game system that will ensure that each puzzle piece is number from 1 - 15, shuffle the puzzle, allow users to play the game by moving pieces around and check if the game has been solved.

Requirements

- Background Image
- Solvable state
- Shuffle logic
- Moving pieces logic

Languages Used:

- HTML
- CSS
- JAVASCRIPT

Extra Features

- end-of-game notification
- extra animation
- game time with music file
- multiple backgrounds
- different size puzzles
- Slide Animation to tiles when clicked

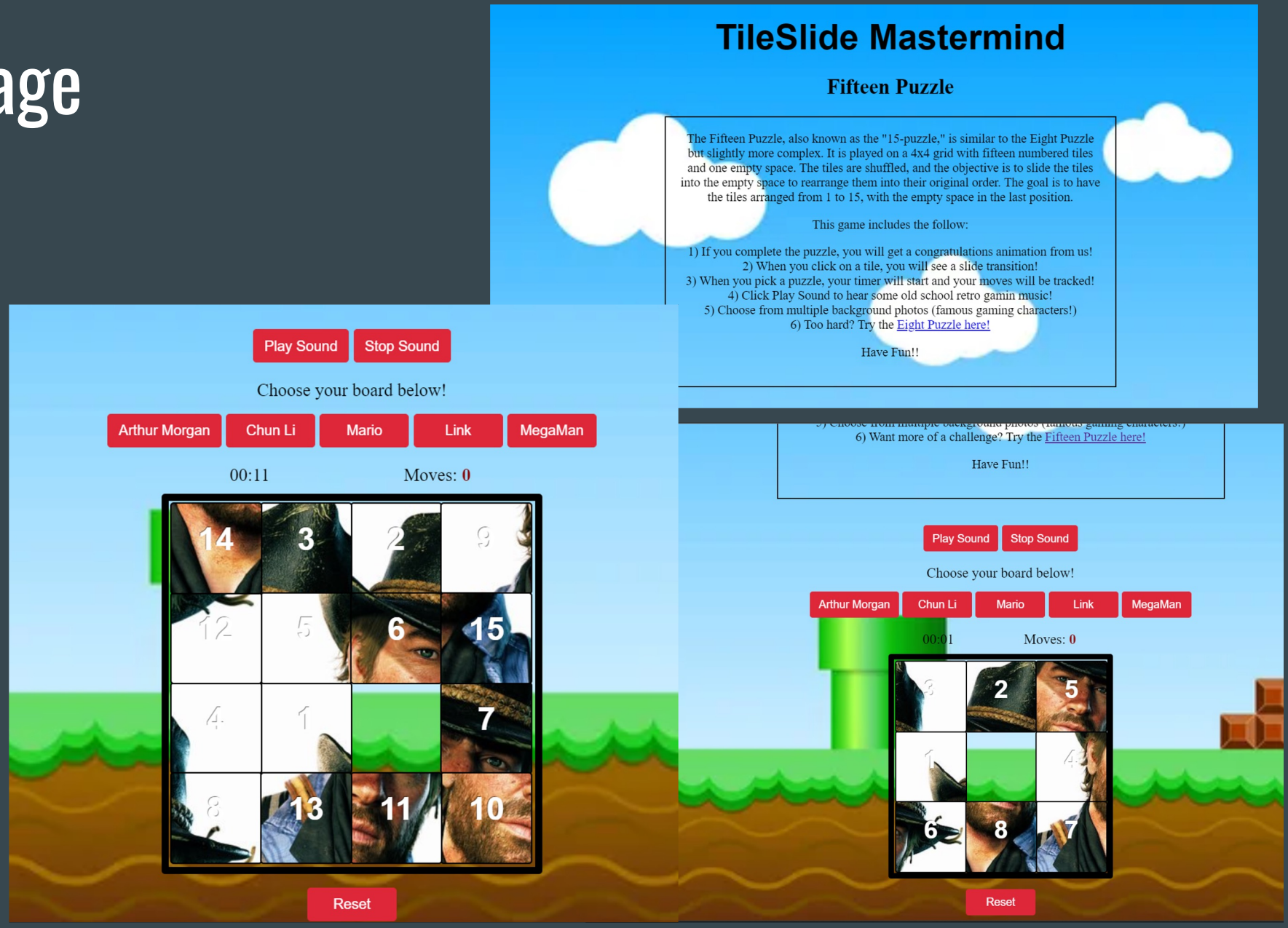
Scrum Framework

Increased Productivity: We were able to break the project down into smaller chunks which allowed us to delegate tasks more efficiently, and kept us focused on our individual tasks as a result increased our overall productivity.

Better communication: we had meeting and discussions everyday and we were able to give proper feedback to each other and discuss current and upcoming tasks.

Increase Motivation: Allowed us to stay motivated on our task because we knew what we each had to do and how our parts contributed to the main project.

Game Page


TileSlide Mastermind
Fifteen Puzzle
The Fifteen Puzzle, also known as the "15-puzzle," is similar to the Eight Puzzle but slightly more complex. It is played on a 4x4 grid with fifteen numbered tiles and one empty space. The tiles are shuffled, and the objective is to slide the tiles into the empty space to rearrange them into their original order. The goal is to have the tiles arranged from 1 to 15, with the empty space in the last position.
This game includes the follow:
1) If you complete the puzzle, you will get a congratulations animation from us!
2) When you click on a tile, you will see a slide transition!
3) When you pick a puzzle, your timer will start and your moves will be tracked!
4) Click Play Sound to hear some old school retro gamin music!
5) Choose from multiple background photos (famous gaming characters!)
6) Too hard? Try the Eight Puzzle here!
Have Fun!!
Play Sound
Stop Sound
Choose your board below!
Arthur Morgan
Chun Li
Mario
Link
MegaMan
00:11
Moves: 0
14
3
2
12
5
6
15
4
1
7
8
13
11
10
Reset
6) Want more of a challenge? Try the Fifteen Puzzle here!
Have Fun!!
Play Sound
Stop Sound
Choose your board below!
Arthur Morgan
Chun Li
Mario
Link
MegaMan
00:01
Moves: 0
2
5
6
8
7
Reset

Puzzles function

```
// Create Solved Puzzle..
function puzzleOne() {

  if (state == 0) {
    return;
  }
  game.innerHTML = ""; // Clear the game area
  let n = 1;
  for (let i = 0; i < 4; i++) {
    for (let j = 0; j < 4; j++) {
      let cell = document.createElement("span"); // create one cell
      cell.id = `cell-${i}-${j}`; //                'cell-' + i + '-' + j; give id to the
      cell
      cell.style.left = i * 100 + 1 * i + 1 + "px"; // position of the cell from left
      cell.style.top = j * 100 + 1 * j + 1 + "px"; // position of the cell from the top
      if (n <= 15) {
        // numberCell
        cell.classList.add("number");
        cell.classList.add(`piece_1-${n}`);
        cell.innerHTML = (n++).toString();

        /*
        // Set the background image for the puzzle piece based on the puzzleIndex
        const imageUrl = `image_${puzzleIndex}/bitfuul-images/image${i + 1}x${j + 1}.jpg`;
        cell.style.backgroundImage = `url('${imageUrl}')`;
        */

      } else {
        // emptyCell
        cell.className = "empty";
      }
      game.appendChild(cell);
    }
  }
  // resetNoOfMoves();
  setTimeout(shuffle, 2000);
}
 // Create Solved Puzzle..
 function puzzleTwo() {

  if (state == 0) {
    return;
  }
```

```
  // Create Solved Puzzle..
function puzzleFive() {

  if (state == 0) {
    return;
  }
  game.innerHTML = ""; // Clear the game area
  let n = 1;
  for (let i = 0; i < 4; i++) {
    for (let j = 0; j < 4; j++) {
      let cell = document.createElement("span"); // create one cell
      cell.id = `cell-${i}-${j}`; //                 'cell-' + i + '-' + j; give id to the
      cell
      cell.style.left = i * 100 + 1 * i + 1 + "px"; // position of the cell from left
      cell.style.top = j * 100 + 1 * j + 1 + "px"; // position of the cell from the top
      if (n <= 15) {
        // numberCell
        cell.classList.add("number");
        cell.classList.add(`piece_5-${n}`);
        cell.innerHTML = (n++).toString();

      } else {
        // emptyCell
        cell.className = "empty";
      }
      game.appendChild(cell);
    }
  }
  // resetNoOfMoves();
  setTimeout(shuffle, 2000);
}
```

Shuffle Implementation

```
// shuffle the game
function shuffle() {
  if (state == 0) {
    return;
  }
  game.removeAttribute("class"); // remove animation..
  state = 0;
  let previousCell;
  let i = 1;
  let interval = setInterval(function() {
    if (i <= 150) {
      let adjacent = getAdjacentCells(getEmptyCell());
      if (previousCell) {
        for (let j = adjacent.length - 1; j >= 0; j--) {
          if (adjacent[j].innerHTML == previousCell.innerHTML) {
            adjacent.splice(j, 1);
          }
        }
      }
      // Gets random adjacent cell and memorizes it for the next iteration
      previousCell = adjacent[getRandomNumberBetween(0, adjacent.length - 1)];
      moveNumberCellToEmpty(previousCell, 0);
      i++;
    } else {
      clearInterval(interval);
      state = 1;
    }
  }, 5);
  // resetNoOfMoves();
}
```

Moving number cell to empty

```
function moveToEmptyCell(cell, playingOrShuffling) {
  // Checks if selected cell has number
  if (cell.className != "empty") {
    // Tries to get empty adjacent cell
    let emptyCell = grabEmptyAdjacentCell(cell);

    if (emptyCell) {
      if (playingOrShuffling === 1) {
          noOfMoves++;
          document.getElementById("counter").innerText = noOfMoves;
      }
      // There is empty adjacent cell..
      // styling and id of the number cell
      let tempCell = { style: cell.style.cssText, id: cell.id };
      // Exchanges id and style values
      cell.style.cssText = emptyCell.style.cssText;
      cell.id = emptyCell.id;
      emptyCell.style.cssText = tempCell.style;
      emptyCell.id = tempCell.id;

      if (state == 1) {
        // Checks the order of numbers
        solvedState();
      }
    }
  }
}
```

Solved state

```
// Checks if the order of numbers is correct and we get solved-state..
function solvedState() {
  // Checks if the empty cell is in correct position
  if (grabCell(3, 3).className != "empty") {
    return;
  }

  let n = 1;
  // Goes through all cells and checks numbers
  for (let i = 0; i < 4; i++) {
    for (let j = 0; j < 4; j++) {
      if (n <= 15 && grabCell(i, j).innerHTML != n.toString()) {
        // Order is not correct
        return;
      }
      n++;
    }
  }
  // Puzzle is solved, offers to shuffle it
  startCongrats(), shuffle();
}
```

DEMO

THANK YOU