import React, { useState, useEffect } from 'react'

import './Math3Game.css'

export const Match3Game2 = () => {

  const [grid, setGrid] = useState([]);

  const [revealedCells, setRevealedCells] = useState([]);

  const [win, setWin] = useState(false);

  // Define the image paths to place on the grid

  const initialImagePaths = [

    `${process.env.PUBLIC\_URL}/red\_color.png`,

    `${process.env.PUBLIC\_URL}/red\_color.png`,

    `${process.env.PUBLIC\_URL}/red\_color.png`,

    `${process.env.PUBLIC\_URL}/green\_color.png`,

    `${process.env.PUBLIC\_URL}/green\_color.png`,

    `${process.env.PUBLIC\_URL}/green\_color.png`,

    `${process.env.PUBLIC\_URL}/green\_color.png`,

    `${process.env.PUBLIC\_URL}/green\_color.png`,

    `${process.env.PUBLIC\_URL}/blue\_color.png`,

    `${process.env.PUBLIC\_URL}/blue\_color.png`,

    `${process.env.PUBLIC\_URL}/blue\_color.png`,

    `${process.env.PUBLIC\_URL}/blue\_color.png`,

    `${process.env.PUBLIC\_URL}/blue\_color.png`,

    `${process.env.PUBLIC\_URL}/blue\_color.png`,

    `${process.env.PUBLIC\_URL}/blue\_color.png`,

    `${process.env.PUBLIC\_URL}/blue\_color.png`,

    // Add more image paths here as needed

  ];

  // Function to shuffle an array

  function shuffleArray(array) {

    const shuffled = array.slice();

    for (let i = shuffled.length - 1; i > 0; i--) {

      const j = Math.floor(Math.random() \* (i + 1));

      [shuffled[i], shuffled[j]] = [shuffled[j], shuffled[i]];

    }

    return shuffled;

  }

  const resetGrid = () => {

    // Shuffle the imagePaths array

    const shuffledPaths = shuffleArray([...initialImagePaths]);

    // Create the grid by mapping the shuffled image paths

    const rows = 4;

    const cols = 4;

    const newGrid = [];

    for (let i = 0; i < rows; i++) {

      const row = [];

      for (let j = 0; j < cols; j++) {

        // Ensure there are image paths left to place

        if (shuffledPaths.length > 0) {

          const imagePath = shuffledPaths.pop();

          row.push(imagePath);

        } else {

          row.push(null); // Use null for cells with no more images

        }

      }

      newGrid.push(row);

    }

    setGrid(newGrid);

    setRevealedCells([]); // Reset revealed cells

    setWin(false); // Reset win state

  };

  const revealCell = (rowIndex, colIndex) => {

    if (!revealedCells.includes(`${rowIndex}-${colIndex}`)) {

      // Reveal the image in the clicked cell

      const updatedRevealedCells = [...revealedCells, `${rowIndex}-${colIndex}`];

      setRevealedCells(updatedRevealedCells);

      if (updatedRevealedCells.length === 3) {

        const images = updatedRevealedCells.map((cell) => {

          const [row, col] = cell.split('-');

          return grid[parseInt(row)][parseInt(col)];

        });

        if (images.every((image) => image === images[0])) {

          setWin(true);

        }

      }

    }

  };

  useEffect(() => {

    // Initial grid setup

    resetGrid();

  }, []);

  return (

    <div className="container">

      <h1 className="title">Bell Link Game Jump Up Casino</h1>

      <button className="reset" onClick={resetGrid}>

        Reset

      </button>

      <div className="board">

        {grid.map((row, rowIndex) => (

          <div key={rowIndex} className={`row${rowIndex + 1}`}>

            {row.map((imagePath, colIndex) => (

              <div

                key={colIndex}

                className={`boxes ${revealedCells.includes(`${rowIndex}-${colIndex}`) ? 'revealed' : ''}`}

                onClick={() => revealCell(rowIndex, colIndex)}

              >

                {revealedCells.includes(`${rowIndex}-${colIndex}`) && (

                  <img src={imagePath} alt="Colorful Bell" />

                )}

              </div>

            ))}

          </div>

        ))}

      </div>

      {win && <div className="winner">Congratulations! You have matched 3 images!</div>}

    </div>

  );

};

export default Match3Game2;