Exercise: Create a Simple "Snake" Game in ReactJS

Prerequisites:

- Basic knowledge of JavaScript, HTML, and CSS.
- Familiarity with ReactJS basics (components, state, props).

Step 1: Setting Up the Project

1. **Install Node.js and npm**: Ensure Node.js and npm are installed on your computer. You can download them from nodejs.org.
2. **Create a New React Project**: Open your terminal and run the following command to create a new React project:

   ```
   npx create-react-app snake-game
   cd snake-game
   ```

3. **Start the Development Server**:

   ```
   npm start
   ```

Step 2: Create the Game Board

1. **Create a Game Component**: In the `src` directory, create a new file called `Game.js` and add the following code:

   ```
   import React from 'react';
   import './Game.css';

   const Game = () => {
       return (
           <div className="game-board">
               <h1>Snake Game</h1>
               <div className="board"></div>
           </div>
       );
   };

   export default Game;
   ```

2. **Update the App Component**: In `src/App.js`, import and render the `Game` component:

   ```
   import React from 'react';
   import Game from './Game';
   import './App.css';

   function App() {
       return (
           <div className="App">
               <Game />
           </div>
       );
   }

   export default App;
   ```

3. **Add CSS for the Game Board**: In `src/Game.css`, add the following styles:

   ```
   .game-board {
       text-align: center;
   }
   ```

```
.board {
    width: 400px;
    height: 400px;
    border: 1px solid #000;
    margin: 0 auto;
    position: relative;
}
```

Step 3: Create the Snake

1. **Define the Initial State**: Update `Game.js` to include state for the snake:

```
import React, { useState } from 'react';
import './Game.css';

const Game = () => {
    const [snake, setSnake] = useState([{ x: 10, y: 10 }]);

    return (
        <div className="game-board">
            <h1>Snake Game</h1>
            <div className="board">
                {snake.map((segment, index) => (
                    <div
                        key={index}
                        className="snake-segment"
                        style={{
                            left: `${segment.x * 20}px`,
                            top: `${segment.y * 20}px`,
                        }}
                    ></div>
                ))}
            </div>
        </div>
    );
};

export default Game;
```

2. **Add CSS for the Snake**: In `src/Game.css`, add the following styles:

```
.snake-segment {
    width: 20px;
    height: 20px;
    background-color: green;
    position: absolute;
}
```

Step 4: Add Movement

1. **Handle Key Presses**: Update `Game.js` to handle arrow key presses and move the snake:

```
import React, { useState, useEffect } from 'react';
import './Game.css';

const Game = () => {
    const [snake, setSnake] = useState([{ x: 10, y: 10 }]);
    const [direction, setDirection] = useState('RIGHT');

    useEffect(() => {
```

```
    const handleKeyDown = (event) => {
        switch (event.key) {
            case 'ArrowUp':
                setDirection('UP');
                break;
            case 'ArrowDown':
                setDirection('DOWN');
                break;
            case 'ArrowLeft':
                setDirection('LEFT');
                break;
            case 'ArrowRight':
                setDirection('RIGHT');
                break;
            default:
                break;
        }
    };

    document.addEventListener('keydown', handleKeyDown);

    return () => {
        document.removeEventListener('keydown', handleKeyDown);
    };
}, []);

useEffect(() => {
    const moveSnake = () => {
        setSnake((prevSnake) => {
            const newSnake = [...prevSnake];
            const head = { ...newSnake[0] };

            switch (direction) {
                case 'UP':
                    head.y -= 1;
                    break;
                case 'DOWN':
                    head.y += 1;
                    break;
                case 'LEFT':
                    head.x -= 1;
                    break;
                case 'RIGHT':
                    head.x += 1;
                    break;
                default:
                    break;
            }

            newSnake.unshift(head);
            newSnake.pop();
            return newSnake;
        });
    };

    const intervalId = setInterval(moveSnake, 200);
```

```
        return () => clearInterval(intervalId);
    }, [direction]);

    return (
        <div className="game-board">
            <h1>Snake Game</h1>
            <div className="board">
                {snake.map((segment, index) => (
                    <div
                        key={index}
                        className="snake-segment"
                        style={{
                            left: `${segment.x * 20}px`,
                            top: `${segment.y * 20}px`,
                        }}
                    ></div>
                ))}
            </div>
        </div>
    );
};

export default Game;
```

Step 5: Add Food and Collision Detection

1. **Add Food**: Update `Game.js` to include food and handle eating:

```
import React, { useState, useEffect } from 'react';
import './Game.css';

const getRandomPosition = () => ({
    x: Math.floor(Math.random() * 20),
    y: Math.floor(Math.random() * 20),
});

const Game = () => {
    const [snake, setSnake] = useState([{ x: 10, y: 10 }]);
    const [direction, setDirection] = useState('RIGHT');
    const [food, setFood] = useState(getRandomPosition());

    useEffect(() => {
        const handleKeyDown = (event) => {
            switch (event.key) {
                case 'ArrowUp':
                    setDirection('UP');
                    break;
                case 'ArrowDown':
                    setDirection('DOWN');
                    break;
                case 'ArrowLeft':
                    setDirection('LEFT');
                    break;
                case 'ArrowRight':
                    setDirection('RIGHT');
                    break;
                default:
                    break;
```

```
        }
    };

    document.addEventListener('keydown', handleKeyDown);

    return () => {
        document.removeEventListener('keydown', handleKeyDown);
    };
}, []);

useEffect(() => {
    const moveSnake = () => {
        setSnake((prevSnake) => {
            const newSnake = [...prevSnake];
            const head = { ...newSnake[0] };

            switch (direction) {
                case 'UP':
                    head.y -= 1;
                    break;
                case 'DOWN':
                    head.y += 1;
                    break;
                case 'LEFT':
                    head.x -= 1;
                    break;
                case 'RIGHT':
                    head.x += 1;
                    break;
                default:
                    break;
            }

            newSnake.unshift(head);

            // Check if snake eats food
            if (head.x === food.x && head.y === food.y) {
                setFood(getRandomPosition());
            } else {
                newSnake.pop();
            }

            return newSnake;
        });
    };

    const intervalId = setInterval(moveSnake, 200);

    return () => clearInterval(intervalId);
}, [direction, food]);

return (
    <div className="game-board">
        <h1>Snake Game</h1>
        <div className="board">
            {snake.map((segment, index) => (
                <div
```

```
                        key={index}
                        className="snake-segment"
                        style={{
                            left: `${segment.x * 20}px`,
                            top: `${segment.y * 20}px`,
                        }}
                    ></div>
                ))}
                <div
                    className="food"
                    style={{
                        left: `${food.x * 20}px`,
                        top: `${food.y * 20}px`,
                    }}
                ></div>
            </div>
        </div>
    );
};

export default Game;
```

2. **Add CSS for the Food**: In `src/Game.css`, add the following styles:

```
.food {
    width: 20px;
    height: 20px;
    background-color: red;
    position: absolute;
}
```

Step 6: Handle Game Over

1. **Detect Collision with Walls**: Update `Game.js` to check for collisions with walls and end the game:

```
import React, { useState, useEffect } from 'react';
import './Game.css';

const getRandomPosition = () => ({
    x: Math.floor(Math.random() * 20),
    y: Math.floor(Math.random() * 20),
});

const Game = () => {
    const [snake, setSnake] = useState([{ x: 10, y: 10 }]);
    const [direction, setDirection] = useState('RIGHT');
    const [food, setFood] = useState(getRandomPosition());
    const [gameOver, setGameOver] = useState(false);

    useEffect(() => {
        const handleKeyDown = (event) => {
            switch (event.key) {
                case 'ArrowUp':
                    setDirection('UP');
                    break;
                case 'ArrowDown':
                    setDirection('DOWN');
                    break;
                case 'ArrowLeft':
```

```
                setDirection('LEFT');
                break;
            case 'ArrowRight':
                setDirection('RIGHT');
                break;
            default:
                break;
        }
    };

    document.addEventListener('keydown', handleKeyDown);

    return () => {
        document.removeEventListener('keydown', handleKeyDown);
    };
}, []);

useEffect(() => {
    const moveSnake = () => {
        setSnake((prevSnake) => {
            const newSnake = [...prevSnake];
            const head = { ...newSnake[0] };

            switch (direction) {
                case 'UP':
                    head.y -= 1;
                    break;
                case 'DOWN':
                    head.y += 1;
                    break;
                case 'LEFT':
                    head.x -= 1;
                    break;
                case 'RIGHT':
                    head.x += 1;
                    break;
                default:
                    break;
            }

            // Check for collision with walls
            if (head.x < 0 || head.x >= 20 || head.y < 0 || head.y >= 20) {
                setGameOver(true);
                return prevSnake;
            }

            newSnake.unshift(head);

            // Check if snake eats food
            if (head.x === food.x && head.y === food.y) {
                setFood(getRandomPosition());
            } else {
                newSnake.pop();
            }

            return newSnake;
        });
```

```
        };

        if (!gameOver) {
            const intervalId = setInterval(moveSnake, 200);
            return () => clearInterval(intervalId);
        }
    }, [direction, food, gameOver]);

    return (
        <div className="game-board">
            <h1>Snake Game</h1>
            {gameOver && <h2>Game Over</h2>}
            <div className="board">
                {snake.map((segment, index) => (
                    <div
                        key={index}
                        className="snake-segment"
                        style={{
                            left: `${segment.x * 20}px`,
                            top: `${segment.y * 20}px`,
                        }}
                    ></div>
                ))}
                <div
                    className="food"
                    style={{
                        left: `${food.x * 20}px`,
                        top: `${food.y * 20}px`,
                    }}
                ></div>
            </div>
        </div>
    );
};

export default Game;
```

Step 7: Restart the Game

1. **Add Restart Functionality**: Update `Game.js` to allow restarting the game:

```
import React, { useState, useEffect } from 'react';
import './Game.css';

const getRandomPosition = () => ({
    x: Math.floor(Math.random() * 20),
    y: Math.floor(Math.random() * 20),
});

const Game = () => {
    const [snake, setSnake] = useState([{ x: 10, y: 10 }]);
    const [direction, setDirection] = useState('RIGHT');
    const [food, setFood] = useState(getRandomPosition());
    const [gameOver, setGameOver] = useState(false);

    const resetGame = () => {
        setSnake([{ x: 10, y: 10 }]);
        setDirection('RIGHT');
```

```
    setFood(getRandomPosition());
    setGameOver(false);
};

useEffect(() => {
    const handleKeyDown = (event) => {
        switch (event.key) {
            case 'ArrowUp':
                setDirection('UP');
                break;
            case 'ArrowDown':
                setDirection('DOWN');
                break;
            case 'ArrowLeft':
                setDirection('LEFT');
                break;
            case 'ArrowRight':
                setDirection('RIGHT');
                break;
            default:
                break;
        }
    };

    document.addEventListener('keydown', handleKeyDown);

    return () => {
        document.removeEventListener('keydown', handleKeyDown);
    };
}, []);

useEffect(() => {
    const moveSnake = () => {
        setSnake((prevSnake) => {
            const newSnake = [...prevSnake];
            const head = { ...newSnake[0] };

            switch (direction) {
                case 'UP':
                    head.y -= 1;
                    break;
                case 'DOWN':
                    head.y += 1;
                    break;
                case 'LEFT':
                    head.x -= 1;
                    break;
                case 'RIGHT':
                    head.x += 1;
                    break;
                default:
                    break;
            }

            // Check for collision with walls
            if (head.x < 0 || head.x >= 20 || head.y < 0 || head.y >= 20) {
                setGameOver(true);
```

```
                return prevSnake;
            }

            newSnake.unshift(head);

            // Check if snake eats food
            if (head.x === food.x && head.y === food.y) {
                setFood(getRandomPosition());
            } else {
                newSnake.pop();
            }

            return newSnake;
        });
    };

    if (!gameOver) {
        const intervalId = setInterval(moveSnake, 200);
        return () => clearInterval(intervalId);
    }
}, [direction, food, gameOver]);

return (
    <div className="game-board">
        <h1>Snake Game</h1>
        {gameOver && (
            <>
                <h2>Game Over</h2>
                <button onClick={resetGame}>Restart Game</button>
            </>
        )}
        <div className="board">
            {snake.map((segment, index) => (
                <div
                    key={index}
                    className="snake-segment"
                    style={{
                        left: `${segment.x * 20}px`,
                        top: `${segment.y * 20}px`,
                    }}
                ></div>
            ))}
            <div
                className="food"
                style={{
                    left: `${food.x * 20}px`,
                    top: `${food.y * 20}px`,
                }}
            ></div>
        </div>
    </div>
);
};

export default Game;
```

2. **Add CSS for the Restart Button**: In `src/Game.css`, add the following styles:

```css
.game-board button {
    margin-top: 20px;
    padding: 10px 20px;
    font-size: 16px;
    cursor: pointer;
}
```