Integration Guide: Enhanced Persistent Piece Reactions

Overview

This enhanced reaction system gives each piece a persistent emotional state that reflects their current situation in the game. Each piece will continuously display reactions based on their position, threats, and opportunities.

Key Features

🎭 Persistent Emotions

- **Home States**: Trapped, eager to start
- **Path States**: Confident, hunting, scared, vulnerable, safe
- **Special States**: Chasing, being chased, blocking enemies
- **Victory States**: Almost home, finished

🔄 Dynamic Updates

- Reactions update every 2 seconds
- Emojis rotate within each emotion category
- Temporary reactions override persistent ones

Integration Steps

1. Import the Enhanced Reaction System

javascript

```javascript
// Add to your computer.js imports
import {
    initializeAllPieceEmotions,
    updateAllPieceReactions,
    updateReactionsLoop,
    onPieceKill,
    onRollSix,
    onPieceCantMove,
    triggerTemporaryReaction,
    addReactionStyles
} from './enhanced_reactions.js';
```

2. Initialize on Game Start

Replace your existing `window.onload` function:

```javascript
window.onload = async function() {
    initializeBoard();
    createPieces();

    // Add these new lines
    addReactionStyles();
    initializeAllPieceEmotions();
    updateReactionsLoop(); // Start the reaction update loop

    onGameStart();
    await nextTurn();
    gameStarted = true;
};
```

3. Update Existing Functions

Replace `showPieceReaction` function:

```javascript
// Replace the existing simple function with:
function showPieceReaction(piece, emoji) {
    // This is now handled by the persistent system
    // You can trigger temporary reactions instead:
    triggerTemporaryReaction(piece, 'CONFIDENT', 2000);
}
```

Update `rollDice` function:

```javascript
async function rollDice() {
    // Your existing code...

    // Replace the existing emoji assignment with:
    if (diceValue === 6) {
        const currentPlayerPieces = players[currentTurn].pieces;
        onRollSix(currentPlayerPieces);
    }

    // Your existing playable pieces logic...

    if (playablePieces.length === 0) {
        // Instead of just ending turn, show reaction
        playablePieces.forEach(piece => {
            onPieceCantMove(piece);
        });
        dice.style.pointerEvents = 'auto';
        await nextTurn();
    }

    // Rest of your existing code...
}
```

Update `checkAndKillOpponent` function:

javascript

```javascript
async function checkAndKillOpponent(movedPiece) {
    const currentCell = movedPiece.parentNode;
    const movedPiecePlayer = movedPiece.dataset.player;

    if (currentCell.classList.contains('safe-cell')) {
        return;
    }

    const piecesOnCell = Array.from(currentCell.querySelectorAll('.piece'));

    await Promise.all(piecesOnCell.map(async piece => {
        if (piece !== movedPiece && piece.dataset.player !== movedPiecePlayer) {
            const opponentPlayer = players[piece.dataset.player];
            const pieceId = piece.dataset.pieceId;
            const pieceNumber = parseInt(pieceId.split('-')[1]);
            iskilledOtherPlayer = true;

            // Add this line to trigger reactions
            onPieceKill(movedPiece, piece);

            const homeCircle = document.getElementById(opponentPlayer.homeCircles[pieceNumber -
            if (homeCircle) {
                await animatePieceToCell(piece, homeCircle, 500);
                piece.dataset.position = 'home';
                piece.dataset.pathIndex = -1;
                piece.style.width = `${pieceSize}px`;
                piece.style.height = `${pieceSize}px`;
            }
        }
    }));

    arrangePiecesInCell(currentCell);
}
```

Update `nextTurn` function:

javascript

```javascript
async function nextTurn() {
    stopHeartbeat(currentTurn);
    await sleep(500);

    const currentIndex = playerColorsInGame.indexOf(currentTurn);
    currentTurn = playerColorsInGame[(currentIndex + 1) % playerColorsInGame.length];
    currentPlayerDisplay.textContent = currentTurn;
    currentPlayerDisplay.className = '';
    currentPlayerDisplay.classList.add(`${currentTurn}-turn`);

    startHeartbeat(currentTurn);
    changeDiceColor(currentTurn);

    // Add this line to update reactions when turn changes
    updateAllPieceReactions();

    if (computerPlayers[currentTurn]) {
        dice.style.pointerEvents = 'none';
        await diceRollAnimation();
    }
}
```

4. CSS Enhancements

The system automatically adds required CSS animations. Make sure your `.piece-emoji` class has proper positioning:

```css
.piece {
    position: relative; /* Ensure pieces are positioned relatively */
}

.piece-emoji {
    position: absolute;
    top: -8px;
    right: -5px;
    font-size: 12px;
    z-index: 10;
    pointer-events: none;
    display: block;
    line-height: 1;
    transition: all 0.3s ease;
}
```

Emotion Categories

🏠 Home Emotions

- **TRAPPED_AT_HOME**: 😤 😔 🙄 😑 😠 😮‍💨
- **EAGER_TO_START**: 😍 🤩 😋 🥺 👀 🔥

🛤️ Path Emotions

- **CONFIDENT**: 😎 😏 🕶️ 💪 😈 🔥
- **HUNTING**: 👹 👿 😈 👺 🦈 🎯
- **VULNERABLE**: 😰 😨 👀 🫣 😬 ⚠️
- **SCARED**: 😱 😭 🥺 😰 💀 ☠️
- **SAFE_AND_HAPPY**: 😊 😌 🛡️ 😁 🤗 💚

🎯 Special Emotions

- **CHASING_OPPONENT**: 🏃 💨 🎯 ⚡ 🔥 😤
- **BEING_CHASED**: 🏃 💨 😱 😳 😵 💀
- **ALMOST_HOME**: 🏁 🎯 ✨ 🚀 ⭐ 🏆
- **FINISHED**: 🏆 👑 🎉 🥇 🌟 ✨

⚡ Temporary Reactions

- **JUST_KILLED**: 😈 💀 🔥 😎 🎯 💥 (4 seconds)
- **JUST_GOT_KILLED**: 😭 💔 😵 ☠️ 😥 💀 (4 seconds)
- **ROLLED_SIX**: 🎲 🔥 😍 🎊 ⚡ 🎉 (2 seconds)
- **CANT_MOVE**: 😫 🤷 😩 😞 🚫 😤 (2 seconds)

Benefits

1. **Immersive Experience**: Each piece has personality and reacts to the game state
2. **Strategic Feedback**: Players can see which pieces are in danger or hunting
3. **Emotional Connection**: Players develop attachment to their pieces
4. **Visual Storytelling**: The game tells a story through piece reactions
5. **Enhanced Gameplay**: Reactions provide subtle hints about game strategy

Customization

You can easily customize reactions by:

- Adding new emotion categories to `PIECE_EMOTIONS`
- Modifying emoji sets for different themes
- Adjusting update frequency in `updateReactionsLoop`
- Adding new temporary reaction triggers

The system is designed to be extensible and can grow with your game's features!