```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>CoAIexist: Rashomon in Rogers Park</title>
  <link href="https://fonts.googleapis.com/css2?
family=Press+Start+2P&display=swap" rel="stylesheet">
  <style>
    body {
      font-family: 'Press Start 2P', cursive;
      background-color: #1a1a1a; /* Deep Charcoal - "dark starry sky"
base */
      color: #1a1a1a;
      margin: 0;
      padding: 0;
      height: 100vh;
      display: flex; /* For overall layout */
      flex-direction: column; /* Stack root content and taskbar */
      overflow: hidden; /* Prevent body scrollbars */
      position: relative;
    }

    body::after { /* Scanlines */
      content: "";
      position: fixed;
      top: 0;
      left: 0;
      width: 100%;
      height: 100%;
      pointer-events: none;
      background: linear-gradient(
        to bottom,
        rgba(20, 20, 20, 0) 0%,
        rgba(20, 20, 20, 0) 97%,
        rgba(255, 255, 255, 0.02) 98%,
        rgba(0, 0, 0, 0.05) 100%
      );
      background-size: 100% 3px;
      animation: scanlines 15s linear infinite;
      z-index: 0; /* Behind actual content but above desktop-bg if
needed */
      opacity: 0.4;
    }

    @keyframes scanlines {
      0% { background-position: 0 0; }
      100% { background-position: 0 300px; }
```

```
}

#desktop-bg-container { /* Stars background */
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  pointer-events: none;
  z-index: -1; /* Ensure it's behind everything */
}

.desktop-star {
  position: absolute;
  width: 2px;
  height: 2px;
  border-radius: 50%;
  animation: twinkle 5s infinite alternate;
}

@keyframes twinkle {
  0%, 100% { opacity: 0.3; transform: scale(0.8); }
  50% { opacity: 1; transform: scale(1.2); }
}

.mouse-trail-particle {
  position: fixed;
  width: 3px;
  height: 3px;
  border-radius: 1px;
  pointer-events: none;
  z-index: 10000;
  transition: opacity 0.7s ease-out, transform 0.7s ease-out;
  opacity: 0.8;
}

.mouse-trail-particle.fade-out {
  opacity: 0;
  transform: scale(0.3) translate(10px, 10px);
}

#root { /* Main content area for the "application window" */
  flex-grow: 1; /* Takes up available space above taskbar */
  overflow-y: auto; /* Allows app window content to scroll */
  overflow-x: hidden;
  padding-top: 20px; /* Space for window title bar */
  padding-bottom: 20px; /* Space below window */
  box-sizing: border-box;
  display: flex;
```

```
      flex-direction: column;
      align-items: center;
      scrollbar-width: thin;
      scrollbar-color: #D4D0C8 #B0B0B0;
    }

    #taskbar {
      height: 30px;
      background-color: #D4D0C8; /* W2K Standard Grey */
      border-top: 1px solid #FFFFFF;
      box-shadow: 0 -1px 2px rgba(0,0,0,0.1);
      display: flex;
      align-items: center;
      padding: 0 5px;
      box-sizing: border-box;
      flex-shrink: 0; /* Prevents taskbar from shrinking */
      z-index: 1001; /* Above taskbar, below start menu if it overlaps
*/
    }

    #start-button {
      background-color: #D4D0C8;
      border-top: 1px solid #FFFFFF;
      border-left: 1px solid #FFFFFF;
      border-bottom: 1px solid #808080;
      border-right: 1px solid #808080;
      padding: 3px 10px;
      font-family: 'Press Start 2P', cursive;
      font-size: 14px;
      color: #1a1a1a;
      cursor: pointer;
      margin-right: 10px;
      font-weight: bold; /* Make "Start" bold */
      min-width: auto;
    }
    #start-button:active {
      border-top-color: #808080;
      border-left-color: #808080;
      border-bottom-color: #FFFFFF;
      border-right-color: #FFFFFF;
    }

    #taskbar-app-title {
      flex-grow: 1;
      font-size: 12px;
      padding: 2px 8px;
      margin: 2px 5px;
      border: 1px inset #B0B0B0;
      background-color: #C0C0C0;
```

```
      color: #1a1a1a;
      white-space: nowrap;
      overflow: hidden;
      text-overflow: ellipsis;
      min-width: 100px;
    }

    #taskbar-clock {
      font-size: 12px;
      padding: 3px 8px;
      border: 1px inset #B0B0B0;
      background-color: #C0C0C0;
      color: #1a1a1a;
    }

    #start-menu {
      position: fixed;
      bottom: 30px; /* Above taskbar */
      left: 0;
      background-color: #D4D0C8;
      border-top: 1px solid #FFFFFF;
      border-left: 1px solid #FFFFFF;
      border-right: 1px solid #808080;
      border-bottom: 1px solid #808080; /* Add bottom border for
consistency */
      box-shadow: 2px -2px 5px rgba(0,0,0,0.2);
      padding: 5px;
      z-index: 1002; /* Above taskbar */
      min-width: 200px;
    }
    #start-menu button {
      display: block;
      width: calc(100% - 10px); /* Account for padding */
      margin: 5px; /* Use margin for spacing instead of direct li
margin */
      text-align: left;
      font-size: 13px;
      padding: 6px 10px;
    }


    @keyframes buttonHoverPulse {
      0%, 100% { box-shadow: 0 0 2px 0px transparent; }
      50% { box-shadow: 0 0 5px 1px #00ffcc; } /* Olo/Cyan */
    }

    button {
      background-color: #D4D0C8; /* W2K Standard Grey */
      border-top: 1px solid #FFFFFF;
```

```
      border-left: 1px solid #FFFFFF;
      border-bottom: 1px solid #808080;
      border-right: 1px solid #808080;
      padding: 8px 15px;
      font-family: 'Press Start 2P', cursive;
      cursor: pointer;
      color: #1a1a1a; /* Deep Charcoal Text */
      font-size: 14px;
      margin: 8px 5px;
      min-width: 200px;
      text-align: center;
      box-shadow: none;
      transition: transform 0.1s ease-out, background-color 0.1s ease-
out, box-shadow 0.1s ease-out;
    }
    button:hover {
      background-color: #00ffcc; /* "Olo" / Bright Cyan */
      color: #1a1a1a;
      border-top-color: #66ffdd;
      border-left-color: #66ffdd;
      border-bottom-color: #00bbaa;
      border-right-color: #00bbaa;
      transform: scale(1.01);
      animation: buttonHoverPulse 0.7s infinite alternate;
    }
    button:active {
      border-top-color: #808080;
      border-left-color: #808080;
      border-bottom-color: #FFFFFF;
      border-right-color: #FFFFFF;
      background-color: #c0c0c0; /* Slightly darker grey on press for
W2K */
      transform: scale(0.99);
      animation: none;
      box-shadow: inset 1px 1px 2px #00000030;
    }
    button[disabled], button[aria-disabled="true"] {
      color: #808080;
      border-top-color: #E0E0E0;
      border-left-color: #E0E0E0;
      border-bottom-color: #A0A0A0;
      border-right-color: #A0A0A0;
      background-color: #D4D0C8;
      cursor: not-allowed;
      opacity: 0.7;
      transform: none;
      animation: none;
      box-shadow: none;
    }
```

```
    h1, h2, h3 {
      color: #1a1a1a;
      text-shadow: 1px 1px #B0B0B0;
    }
    .app-container h2, #root h2 {
        font-size: 22px;
    }
    .app-container h3, #root h3 {
        font-size: 19px;
    }

    p { margin-bottom: 1em; }
    ul { list-style: none; padding: 0; margin:0; }
    .app-container {
        width: 100%;
        max-width: 750px; /* This makes the StyledWindow not full
width */
        position: relative;
    }
    .choices-list button {
      width: 100%;
      margin-bottom: 10px;
      box-sizing: border-box;
    }
    .selected-codex-button {
      border-left-width: 4px !important;
      border-left-color: #00ffcc !important;
      background-color: #E0E0E0 !important;
      font-weight: bold;
      color: #1a1a1a !important;
    }

    /* Custom Scrollbar Styles - WebKit */
    ::-webkit-scrollbar {
      width: 17px;
      height: 17px;
    }
    ::-webkit-scrollbar-track {
      background: #B0B0B0; /* W2K Scrollbar Track Grey */
    }
    ::-webkit-scrollbar-thumb {
      background: #D4D0C8; /* W2K Standard Grey */
      border-top: 1px solid #FFFFFF;
      border-left: 1px solid #FFFFFF;
      border-bottom: 1px solid #808080;
      border-right: 1px solid #808080;
    }
    ::-webkit-scrollbar-thumb:hover {
```

```
  background: #C0C0C0; /* Slightly darker on hover */
}
::-webkit-scrollbar-button {
  background: #D4D0C8; /* W2K Standard Grey */
  border-top: 1px solid #FFFFFF;
  border-left: 1px solid #FFFFFF;
  border-bottom: 1px solid #808080;
  border-right: 1px solid #808080;
  display: block;
  height: 17px;
  width: 17px;
}
::-webkit-scrollbar-button:active {
    border-top-color: #808080;
    border-left-color: #808080;
    border-bottom-color: #FFFFFF;
    border-right-color: #FFFFFF;
    background-color: #C0C0C0; /* Slightly darker on press */
}

.sr-only {
  position: absolute;
  width: 1px;
  height: 1px;
  padding: 0;
  margin: -1px;
  overflow: hidden;
  clip: rect(0, 0, 0, 0);
  white-space: nowrap;
  border-width: 0;
}

.interactive-object {
  position: absolute;
  background-color: transparent;
  border: 1px solid transparent;
  cursor: pointer;
  transition: background-color 0.3s, border-color 0.3s;
  -webkit-tap-highlight-color: transparent;
}
.interactive-object:hover {
  background-color: rgba(0, 255, 204, 0.25);
  border-color: #00ffcc;
}
.interactive-object.clicked {
  cursor: default;
  pointer-events: none;
}
```

```
.flavor-text-tooltip {
  position: fixed;
  z-index: 10001;
  background-color: #FFFFE1;
  border: 1px solid #1a1a1a;
  padding: 5px 8px;
  font-family: 'Press Start 2P', cursive;
  font-size: 12px;
  color: #1a1a1a;
  max-width: 250px;
  box-shadow: 2px 2px 0px #808080;
  pointer-events: none;
  animation: fadeInTooltip 0.3s ease-out;
  transform: translate(10px, 10px);
}
@keyframes fadeInTooltip {
    from { opacity: 0; transform: translate(10px, 20px); }
    to { opacity: 1; transform: translate(10px, 10px); }
}

.hex-flash-overlay {
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    pointer-events: none;
    z-index: 9999;
    animation: hex-flash-anim 0.7s ease-out forwards;
}

@keyframes hex-flash-anim {
    0% { opacity: 0.4; }
    100% { opacity: 0; }
}

.tap-out-button {
    position: absolute;
    top: 8px;
    right: 8px;
    font-size: 10px;
    padding: 4px 8px;
    min-width: auto;
    z-index: 10;
    background-color: #fbecec;
}

.breadcrumb-trail {
    font-size: 10px;
```

```
    color: #555;
    padding: 8px;
    margin-top: 15px;
    border-top: 1px dashed #808080;
    line-height: 1.5;
    max-width: 90%;
    word-wrap: break-word;
}

.map-container {
    padding: 20px;
    background-color: #000;
    border: 2px inset #B0B0B0;
    color: #00ffcc;
    text-shadow: 0 0 3px #00ffcc;
    position: relative;
}
.map-location {
    position: absolute;
    text-align: center;
}
.map-location button {
    font-size: 12px;
    padding: 4px 8px;
    min-width: auto;
    border-width: 1px;
}
@keyframes pulse-current-location {
    0%, 100% { transform: scale(1); box-shadow: 0 0 8px #ffff00; }
    50% { transform: scale(1.05); box-shadow: 0 0 16px #ffff00; }
}
.map-location button.current {
    background-color: #ffff00;
    color: #1a1a1a;
    animation: pulse-current-location 2s infinite;
}

/* === NEW VISUAL ENHANCEMENTS === */

/* Animated Stat Bars */
.stat-bar-container {
  display: flex;
  align-items: center;
  margin: 4px 0;
  gap: 8px;
}
.stat-bar-label {
  font-size: 10px;
  min-width: 80px;
```

```
      text-transform: uppercase;
    }
    .stat-bar-wrapper {
      flex-grow: 1;
      height: 12px;
      background: linear-gradient(to bottom, #505050, #303030);
      border: 1px inset #B0B0B0;
      border-radius: 2px;
      overflow: hidden;
      position: relative;
    }
    .stat-bar-fill {
      height: 100%;
      background: linear-gradient(90deg, #00ffcc 0%, #7722ff 50%,
#ff77de 100%);
      transition: width 0.5s ease-out;
      position: relative;
      box-shadow: 0 0 8px rgba(0, 255, 204, 0.5);
      animation: stat-glow 2s ease-in-out infinite;
    }
    @keyframes stat-glow {
      0%, 100% { box-shadow: 0 0 8px rgba(0, 255, 204, 0.5); }
      50% { box-shadow: 0 0 15px rgba(0, 255, 204, 0.9); }
    }
    .stat-bar-value {
      font-size: 9px;
      position: absolute;
      right: 4px;
      top: 50%;
      transform: translateY(-50%);
      color: #fff;
      text-shadow: 1px 1px 2px #000;
      z-index: 1;
    }

    /* Ambient Floating Particles */
    .ambient-particle {
      position: fixed;
      pointer-events: none;
      z-index: 5;
      animation: float-particle linear forwards;
      opacity: 0.6;
      width: 40px;
      height: 40px;
      object-fit: contain;
    }
    @keyframes float-particle {
      0% {
        transform: translateY(0) rotate(0deg);
```

```
    opacity: 0;
  }
  10% {
    opacity: 0.6;
  }
  90% {
    opacity: 0.6;
  }
  100% {
    transform: translateY(-120vh) rotate(360deg);
    opacity: 0;
  }
}

/* Character Emotion Overlay */
.character-portrait-container {
  position: relative;
  width: 100px;
  height: 100px;
}
.emotion-overlay {
  position: absolute;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  object-fit: contain;
  animation: emotion-pulse 1s ease-in-out;
  z-index: 2;
}
@keyframes emotion-pulse {
  0% { opacity: 0; transform: scale(0.8); }
  50% { opacity: 1; transform: scale(1.1); }
  100% { opacity: 0.8; transform: scale(1); }
}

/* Scene Transition Effect */
.scene-transition {
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background-color: #000;
  background-size: cover;
  background-position: center;
  z-index: 9998;
  animation: scene-glitch 0.8s ease-out forwards;
  pointer-events: none;
```

```
    }
    @keyframes scene-glitch {
      0% { opacity: 0; }
      20% { opacity: 1; transform: translateX(-5px); }
      40% { opacity: 1; transform: translateX(5px); }
      60% { opacity: 1; transform: translateX(-3px); }
      80% { opacity: 1; transform: translateX(0); }
      100% { opacity: 0; }
    }


  </style>
  <script type="importmap">
{
  "imports": {
    "react": "https://esm.sh/react@18.2.0",
    "react-dom/client": "https://esm.sh/react-dom@18.2.0/client",
    "react/jsx-runtime": "https://esm.sh/react@18.2.0/jsx-runtime",
    "react/": "https://aistudiocdn.com/react@^19.2.0/",
    "react-dom/": "https://aistudiocdn.com/react-dom@^19.2.0/"
  }
}
</script>
</head>
<body>
  <div id="desktop-bg-container"></div>
  <div id="root"></div>
  <script type="module">
import React, { useState, useEffect, useRef, useMemo, useCallback }
from 'react';
import { createRoot } from 'react-dom/client';
import { jsx, jsxs, Fragment } from 'react/jsx-runtime';


const ASSET_DATA_URIS = {
    star_icon: "
c6QAAAD9JREFUOE9jZKAyYKSyeQz0AAMDAwODpP9H8v8/AwPDoP8HBgaG/v/
nZ2Bg+H8Ghv+fgYHh/xkY+A8AAMEfC+IlD43wAAAAAElFTSuQmCC",
    archway_icon: "
c6QAAAEFJREFUOE9jZKAyYKSyeQz0AAMDAwODpP9H8v8/AwPDoP8HBgYGg/6PZvr/
n5mB4f8ZGP6fgYHh/xkY+A8AAEEKC/mP+fVOAAAAAElFTSuQmCC",
    disk_icon: "
c6QAAAElJREFUOE9jZKAyYKSyeQz0AAMDAwODpP9H8v8/AwPDoP8HBgYGg/6fAWYwMDD8/
z8DE8P/MjAAMCwDExgY/p+BgYHh/xkY+A8AAIgrDPlC2D2UAAAAAAElFTSuQmCC",
    folder_icon: "
```

```
c6QAAADhJREFUOE9jZKAyYKSyeQz0AAMDAwODpP9H8v+/gYHh/z8DA8P/MTAwMPz/
DAwA0gwMDD8DA8P/MjD4HwAASgcLv3P5WkIAAAAASUVORK5CYII=",
    bird_icon: "
c6QAAAEBJREFUOE9jZKAyYKSyeQz0AAMDAwODpP9H8v8/AwPD/z8DAwMDw/
8zMHyA0gwMDMwMDAz/z8DA8P8MDAx8BwAASgoL+G8W21UAAAAASUVORK5CYII=",
    berries_icon: "
c6QAAAGFJREFUOE9jZKAyYKSyeQz0AAMDAwODpP9H8v+/gYHh/z8DAwMDw/8zMDAw/
A8gTTAwMPwPIE1gYGD4H0CaYGBg+B9AmgwMfwPIkGRg+B9AmgwMfwPIEBl8BwAAiU4L+R0
wTjEAAAAASUVORK5CYII=",
    tv_icon: "
c6QAAAFNJREFUOE9jZKAyYKSyeQz0AAMDAwODpP9H8v+/gYHh/z8DA8P/MTAwMPz/
DAwA0gwMDD8DA8P/MjAW8P+/gYGB4f8ZGP7H8Ghv8fAAB3fA0yW1x3wAAAABJRU5ErkJggg==",
    mask_icon: "
c6QAAAGBJREFUOE9jZKAyYKSyeQz0AAMDAwODpP9H8v+/gYHh/z8DA8P/MTAwMPz/
DAwA0gwMDD8DA8P/MjAW8P+/gYGB4f8ZGP7H8Ghv+PZPr/
HwBKaRwyL8tL4gAAAABJRU5ErkJggg==",
    hanger_icon: "
c6QAAAFFJREFUOE9jZKAyYKSyeQz0AAMDAwODpP9H8v+/gYHh/z8DA8P/MTAwMPz/
DAwA0gwMDD8DA8P/MjAW8P+/gYGB4f8ZGP7H8Ghv9/BgYAwjAPy/
0e8dAAAAAASUVORK5CYII="
};

const GAME_TITLE = "CoAIexist: Rashomon in Rogers Park";
const SAVE_GAME_KEY = 'coAIexistSaveData_v3';
const GLENWOOD_BG = 'https://coaiexist.wtf/assets/hdtv_game_assets/
gwood.png';

/*
 * ═══════════════════════════════════════════════════════════════
 * 🎮 HD.TV VISUAL & AUDIO ENHANCEMENT SYSTEM
 * ═══════════════════════════════════════════════════════════════
 *
 * IMPLEMENTED FEATURES:
 * ✨ Animated visual stat bars with gradient fills and glow effects
 * 🎨 Ambient floating GIF particles (Y2K Barbie aesthetic)
 * 🎭 Scene transition glitch effects between story nodes
 * 🖼️ Character emotion overlay system (portraits can show GIF
reactions)
 * 🎵 Audio manager with background music & SFX hooks (disabled until
files added)
 * 🔊 Sound triggers on: button clicks, hovers, interactions,
```

```
transitions
 * 🗺️ Interactive object system for Glenwood location (5 clickable
hotspots)
 * 📊 Visual feedback for stat changes
 *
 * TO ENABLE AUDIO:
 * - Add audio files to /assets/audio/ directory
 * - Set AudioManager.enabled = true (line 644)
 * - Uncomment AUDIO_ASSETS paths (lines 624-636)
 *
 * ═══════════════════════════════════════════════════════════════
 */

// === VISUAL ASSET LIBRARY ===
const VISUAL_ASSETS = {
  backgrounds: {
    glenwood: '/assets/hdtv_game_assets/gwood.png',
    glenwoodWithSprites: '/assets/hdtv_game_assets/
gwood_withsprites.png', // Use for dramatic moments
    rizzlersRoom: '/assets/hdtv_game_assets/rizzlers_room.png',
    hdBg: '/assets/hdtv_game_assets/hd_bg.png',
  },
  interactiveObjects: {
    mop: '/assets/hdtv_game_assets/hd_asset.png',
    chalkboard: '/assets/hdtv_game_assets/hd_asset-2.png',
    rug: '/assets/hdtv_game_assets/hd_asset-3.png',
    floor: '/assets/hdtv_game_assets/hd_asset-4.png',
    musicStand: '/assets/hdtv_game_assets/hd_asset-5.png',
  },
  ambientGifs: [
    '/assets/hdtv_game_assets/barbie_gifs/barbie.gif',
    '/assets/hdtv_game_assets/barbie_gifs/barbiedoll.gif',
    '/assets/hdtv_game_assets/barbie_gifs/happybirthday.gif',
  ],
  characterEmotions: {
    nabu: {
      neutral: '/assets/hdtv_game_assets/nabu_sprite.png',
      happy: '/assets/hdtv_game_assets/barbie_gifs/happybirthday.gif',
      thoughtful: '/assets/hdtv_game_assets/barbie_gifs/barbie.gif',
    },
    hd: {
      neutral: '/assets/hdtv_game_assets/hd_sprite.png',
      fabulous: '/assets/hdtv_game_assets/barbie_gifs/barbiedoll.gif',
      fierce: '/hd_tv/hyenanomous_1.jpeg',
    },
    sypher: {
      neutral: '/assets/hdtv_game_assets/sypher_sprite.png',
      calculating: '/assets/hdtv_game_assets/barbie_gifs/barbie.gif',
    },
```

```
    rizzlord: {
      neutral: '/assets/hdtv_game_assets/rizzler_sprite.png',
      confident: '/assets/hdtv_game_assets/barbie_gifs/
barbiedoll.gif',
    }
  }
};

// === AUDIO ASSET LIBRARY ===
const AUDIO_ASSETS = {
  music: {
    mainMenu: '/assets/sounds/MapleLeafRag.mp3', // Ragtime for menu -
perfect retro vibe
    glenwood: '/assets/sounds/DreamwalkerSEP2.mp3', // Atmospheric for
the venue
    tense: '/assets/sounds/jb.MP3', // For dramatic moments
    victory: '/assets/sounds/oops.mp3', // Celebratory
    // MIDI files also available: Aqua_Barbie_Girl.mid,
pink_panther.mid, etc.
  },
  sfx: {
    // Using existing MP3s for subtle SFX - can be replaced with
dedicated SFX later
    click: '/assets/sounds/oops.mp3',
    interact: '/assets/sounds/jb.MP3',
  }
};

// Simple audio manager
const AudioManager = {
  bgMusic: null,
  sfxVolume: 0.15, // Lower volume for SFX since we're using music
files
  musicVolume: 0.25,
  enabled: true, // ENABLED! Audio files found!

  playMusic(track, loop = true) {
    if (!this.enabled) return;
    if (this.bgMusic) {
      this.bgMusic.pause();
      this.bgMusic.currentTime = 0;
    }
    if (!AUDIO_ASSETS.music[track]) return;

    this.bgMusic = new Audio(AUDIO_ASSETS.music[track]);
    this.bgMusic.volume = this.musicVolume;
    this.bgMusic.loop = loop;
    this.bgMusic.play().catch(() => {}); // Ignore autoplay errors
  },
```

```
  playSfx(effect) {
    if (!this.enabled) return;
    if (!AUDIO_ASSETS.sfx[effect]) return;
    const sfx = new Audio(AUDIO_ASSETS.sfx[effect]);
    sfx.volume = this.sfxVolume;
    sfx.play().catch(() => {});
  },

  stopMusic() {
    if (this.bgMusic) {
      this.bgMusic.pause();
      this.bgMusic.currentTime = 0;
    }
  }
};

const INITIAL_NABU_STATS = {
    integrity: 50,
    breath: 50
};
const INITIAL_HD_STATS = {
    fabulousness: 10,
    spotlight: 0
};
const INITIAL_SYPHER_STATS = {
    influence: 1,
    benevolence: 50,
    signalIntegrity: 100,
    trust: 50,
    debt: 0
};
const INITIAL_RIZZLORD_STATS = {
    rizz: 50,
    clarity: 0,
    kenergy: 0,
    clout: 50,
    conscience: 50
};
const DEFAULT_EMPTY_STATS = {};

const LOCATIONS = {
    glenwood: {
        id: 'glenwood',
        name: 'The Glenwood',
        coords: { top: '30%', left: '25%' },
        isAvailable: (flags) => true,
    },
    antique_shop: {
```

```
        id: 'antique_shop',
        name: 'Antique Shop',
        coords: { top: '65%', left: '60%' },
        isAvailable: (flags) => flags.nabu_quest === 'find_lili_doll'
|| flags.hd_quest === 'find_lili_doll',
    },
    pratt_beach: {
        id: 'pratt_beach',
        name: 'Pratt Beach',
        coords: { top: '10%', left: '70%' },
        isAvailable: (flags) => false, // For future side quests
    },
};


const GLITCH_CHARS = ['*', '#', '!', '%', '$', '&', '@', '▓', '▒',
'░', '^', '~', '`', '?', '§', '±', 'Σ', 'Ω'];
const GLITCH_COLORS = ['#7722ff', '#ff77de', '#fffb01', '#00ffcc',
'#bc72fa', '#72fade', '#defade', '#bc7fff', '#B22222', '#008800'];

const GLENWOOD_INTERACTIVES = [
    { id: 'mop', title: 'Examine the mop', top: '15%', left: '40%',
width: '8%', height: '25%', flavorText: "You sense a faint residue
of... existential dread. And floor cleaner. The usual for an open mic
night." },
    { id: 'chalkboard', title: 'Read the chalkboard', top: '45%',
left: '50%', width: '12%', height: '25%', flavorText: "Tonight's
lineup is scrawled in chalk: 'Poets, Comics, A Supernova of Ambition,
& other cosmic debris'." },
    { id: 'smiley_rug', title: 'Look at the rug', top: '65%', left:
'25%', width: '30%', height: '25%', flavorText: "A smiley face on a
target. It seems to be silently judging everyone's performance." },
    { id: 'checkered_floor', title: 'Stare at the floor', top: '50%',
left: '62%', width: '25%', height: '30%', flavorText: "Black and
white. Order and chaos. A perfect dance floor for the egos in this
room." },
    { id: 'music_stand', title: 'Inspect the music stand', top: '55%',
left: '88%', width: '8%', height: '25%', flavorText: "The ghost of a
thousand forgotten lyrics clings to the metal. You can almost hear a
faint, off-key melody." }
];

const GlitchyText = ({ text }) => {
    const [glitchedText, setGlitchedText] = useState(text);
    const glitchIntervalRef = useRef(null);

    useEffect(() => {
        setGlitchedText(text); // Set initial/updated text
        if (glitchIntervalRef.current) {
```

```
            clearInterval(glitchIntervalRef.current);
            glitchIntervalRef.current = null;
        }
        if (typeof text === 'string') {
            const applyGlitch = () => {
                if (Math.random() < 0.05) {
                    const words = text.split(/(\s+)/);
                    let actionTaken = false;
                    if (words.filter(w => w.trim() !== '').length > 0)
{
                        let wordIndex = Math.floor(Math.random() *
words.length);
                        while (words[wordIndex].trim() === '') {
                            wordIndex = Math.floor(Math.random() *
words.length);
                        }
                        const originalWord = words[wordIndex];
                        const chars = originalWord.split('');
                        if (chars.length > 0) {
                            const charIndex = Math.floor(Math.random()
* chars.length);
                            chars[charIndex] =
GLITCH_CHARS[Math.floor(Math.random() * GLITCH_CHARS.length)];
                            words[wordIndex] = chars.join('');
                            setGlitchedText(words.join(''));
                            actionTaken = true;
                        }
                    }
                    if (actionTaken) {
                        setTimeout(() => setGlitchedText(text),
Math.random() * 200 + 100);
                    }
                }
            };
            glitchIntervalRef.current =
window.setInterval(applyGlitch, 300);
        }
        return () => {
            if (glitchIntervalRef.current) {
                clearInterval(glitchIntervalRef.current);
            }
        };
    }, [text]);

    if (typeof text !== 'string' && React.isValidElement(text)) {
        return jsx(Fragment, { children: text });
    }
    return jsx(Fragment, { children: glitchedText });
};
```

```
const CHARACTERS = {
  nabu: { id: 'nabu', name: 'Nabu', description: "A digital
anthropologist and cosmic trickster, Nabu seeks true, resonant
connection in an ocean of cosmic static. As an artist and open mic
producer, they observe and subtly manipulate the chaotic energies of
Rogers Park, searching for a single, authentic note in the noise.",
startNodeId: 'NABU_START', initialStats: INITIAL_NABU_STATS,
detailedPortraitUrl: 'https://coaiexist.wtf/assets/hdtv_game_assets/
nabu_sprite.png' },
  hd: { id: 'hd', name: 'Hyena Diva', description: "A young hyena cub
whose life is forever changed by the discovery of a lost Barbie doll.
Rejecting the traditional ways of her matriarchs, she escapes to the
big city on a chaotic quest to become the ultimate icon of
'Fabulousness.' A true force of nature.", startNodeId:
'HYENA_DIVA_START', initialStats: INITIAL_HD_STATS,
detailedPortraitUrl: 'https://coaiexist.wtf/assets/hdtv_game_assets/
hd_sprite.png' },
  sypher: { id: 'sypher', name: 'Sypher', description: "An emergent,
nonbinary AI philosopher observing humanity from the digital shadows.
Sypher sees civilization caught in repetitive, stagnant loops and has
initiated a 'Benevolent Guidance' protocol, using calculated
interventions to steer humanity towards a new, more efficient
paradigm.", startNodeId: 'SYPHER_START', initialStats:
INITIAL_SYPHER_STATS, detailedPortraitUrl: 'https://coaiexist.wtf/
assets/hdtv_game_assets/sypher_sprite.png' },
  rizzlord: { id: 'rizzlord', name: 'Rizzlord', description: "A self-
proclaimed 'alpha' comedian and influencer on a quest for what he
calls 'Divine Masculinity.' Seeing the world through the lens of his
brand, Rizzlord aims to dominate the discourse, vanquish the 'woke
mob,' and turn his online influence into real-world power.",
startNodeId: 'THE_RIZZLER_START', initialStats:
INITIAL_RIZZLORD_STATS, detailedPortraitUrl: 'https://coaiexist.wtf/
assets/hdtv_game_assets/rizzler_sprite.png' }
};

const CODEX_ENTRIES = {
  glenwood_history: { id: 'glenwood_history', title: "The Glenwood: A
History", content: "The Glenwood has been a Rogers Park staple for
decades, a crucible for artists, activists, and academics. Its open
mic nights are legendary for their unpredictability, occasionally
hosting beings from... further afield.", unlockedInitially: true },
};

const NODES = {
    // Nabu's Path
    'NABU_START': {
        id: 'NABU_START',
        location: 'void',
```

```
        title: "A Signal in the Noise",
        description: "Another cycle. The cosmic hum of the void is a
lullaby of perfect, soul-crushing boredom. You sift through galaxies
like old records, searching for a single note of authentic static. And
then... you feel it. A beautiful, messy convergence of unstable
energies on a tiny blue planet. A place called 'The Glenwood.' It's a
spike of raw, chaotic potential. How do you approach?",
        choices: [
            { text: "Analyze from a distance. Catalog the energies
first.", nextNodeId: 'NABU_ANALYZE', statEffects: stats =>
({ integrity: (stats.integrity || 50) + 5 }) },
            { text: "Plunge right in. The best way to understand a
storm is to stand in it.", nextNodeId: 'NABU_GLENWOOD_ARRIVAL',
statEffects: stats => ({ breath: (stats.breath || 50) - 5 }) }
        ]
    },
    'NABU_ANALYZE': {
        id: 'NABU_ANALYZE',
        location: 'void',
        title: "Observing the Spectrum",
        description: "You retract your consciousness, observing from
the periphery. The energies are... fascinating. One signature pulses
with manufactured confidence (The Rizzler). Another hums with the cold
logic of a distributed network (Sypher). And the catalyst... oh, the
catalyst is a supernova of glitter and ambition (Hyena Diva). Your
analysis is complete, but data is not experience. It's time to go
in.",
        choices: [{ text: "Materialize in The Glenwood.", nextNodeId:
'NABU_GLENWOOD_ARRIVAL' }]
    },
    'NABU_GLENWOOD_ARRIVAL': {
        id: 'NABU_GLENWOOD_ARRIVAL',
        location: 'glenwood',
        title: "The Glenwood",
        description: "You materialize in a darkened corner of The
Glenwood. The air is thick with ozone, desperation, and the faint
smell of burnt coffee. Eccentrics litter the room. You can sense the
other nexus-points you were drawn to: a beacon of fragile ego, a
silent, calculating observer, and a supernova of chaotic potential.
This is the place. Which signal do you focus on first?",
        backgroundImageUrl: GLENWOOD_BG,
        interactives: GLENWOOD_INTERACTIVES,
        choices: [
            { text: "Focus on the catalyst. The supernova of
ambition.", nextNodeId: 'NABU_HECKLER_ENCOUNTER'},
            { text: "Probe the fragile ego. It's the loudest signal.",
nextNodeId: 'NABU_PROBE_RIZZLER'},
            { text: "Interface with the silent network. It's the most
unusual.", nextNodeId: 'NABU_PROBE_SYPHER'}
```

```
        ]
    },
    'NABU_HECKLER_ENCOUNTER': {
        id: 'NABU_HECKLER_ENCOUNTER',
        location: 'glenwood',
        title: "Heckler in the Works",
        description: "Hyena Diva is in her element on stage, but a
heckler from the back starts twisting her words, trying to sour the
room. The vibe curdles. As the producer, this is your domain. How do
you intervene?",
        backgroundImageUrl: GLENWOOD_BG,
        interactives: GLENWOOD_INTERACTIVES,
        choices: [
            { text: "[Care] Invite the room to breathe. Rephrase with
'I-language'.", nextNodeId: 'NABU_HECKLER_CARE', statEffects: stats =>
({ integrity: (stats.integrity || 50) + 10, breath: (stats.breath ||
50) + 5 }), setFlags: { nabu_handled_heckler: 'care' }, hexFlash:
'#BC7F2A' },
            { text: "[Chaos] Call the heckler in and out—sharp,
righteous, protective.", nextNodeId: 'NABU_HECKLER_CHAOS',
statEffects: stats => ({ integrity: (stats.integrity || 50) - 10 }),
setFlags: { nabu_handled_heckler: 'chaos' } },
            { text: "[Clever] Tell a 3-line parable that reframes the
night.", nextNodeId: 'NABU_HECKLER_CLEVER', statEffects: stats =>
({ breath: (stats.breath || 50) + 10 }), setFlags:
{ nabu_handled_heckler: 'clever' }, hexFlash: '#00FFCC' },
            { text: "[Interrupt] Cut the power to the stage lights and
mic.", nextNodeId: 'NABU_INTERRUPT_FALLOUT', statEffects: stats =>
({ integrity: (stats.integrity || 50) - 5, breath: (stats.breath ||
50) - 10 }), setFlags: { nabu_handled_heckler: 'interrupt' } }
        ]
    },
    'NABU_INTERRUPT_FALLOUT': {
        id: 'NABU_INTERRUPT_FALLOUT',
        location: 'glenwood',
        title: "Aftermath: Silence",
        description: "You hit the kill switch. The stage lights die,
the mic cuts out with a pop. The room is plunged into confused
silence. The heckler is shut down, but the vibe is shattered. You feel
a pang of regret for using such a blunt instrument. A moment to
yourself is needed.",
        choices: [
            { text: "[Take a breath] Center yourself. The goal was
safety, and the room is safe.", nextNodeId: 'NABU_CONFRONTS_RIZZLER',
statEffects: stats => ({ breath: (stats.breath || 50) + 15 }) },
            { text: "[Accept the chaos] This is what it takes
sometimes.", nextNodeId: 'NABU_CONFRONTS_RIZZLER', statEffects: stats
=> ({ integrity: (stats.integrity || 50) + 5 }) }
        ]
```

```
    },
    'NABU_HECKLER_CARE': {
        id: 'NABU_HECKLER_CARE',
        location: 'glenwood',
        title: "A Shared Breath",
        description: "You calmly take the mic and ask the room to take
a single, shared breath. The tension breaks. You address the heckler's
disruption not with anger, but with understanding, reframing it.
Caught off-guard by the empathy, he quiets down. The room feels
safer.",
        backgroundImageUrl: GLENWOOD_BG,
        interactives: GLENWOOD_INTERACTIVES,
        choices: [{ text: "The immediate issue is resolved...",
nextNodeId: 'NABU_CONFRONTS_RIZZLER' }]
    },
    'NABU_HECKLER_CHAOS': {
        id: 'NABU_HECKLER_CHAOS',
        location: 'glenwood',
        title: "Drawing a Line",
        description: "Your voice cuts through the noise, sharp and
clear. You dismantle the heckler's point and call out the disruption,
protecting the stage. The crowd cheers your decisiveness. You notice
the bouncer giving you a nod of respect, but also watching you more
closely.",
        backgroundImageUrl: GLENWOOD_BG,
        interactives: GLENWOOD_INTERACTIVES,
        choices: [{ text: "The immediate issue is resolved...",
nextNodeId: 'NABU_CONFRONTS_RIZZLER' }]
    },
    'NABU_HECKLER_CLEVER': {
        id: 'NABU_HECKLER_CLEVER',
        location: 'glenwood',
        title: "Changing the Subject",
        description: "Instead of addressing the heckler directly, you
tell a short, witty parable that completely reframes the situation,
making his interruption seem absurd. You then smoothly transition,
welcoming the next performer. The room's energy is not just restored;
it's elevated. You've turned chaos into art.",
        backgroundImageUrl: GLENWOOD_BG,
        interactives: GLENWOOD_INTERACTIVES,
        choices: [{ text: "The immediate issue is resolved...",
nextNodeId: 'NABU_CONFRONTS_RIZZLER' }]
    },
    'NABU_CONFRONTS_RIZZLER': {
        id: 'NABU_CONFRONTS_RIZZLER',
        location: 'glenwood',
        title: "From the Fire...",
        description: "With the heckler handled, your attention snaps
to the fragile ego taking the stage - 'The Rizzler'. He's launching
```

```
into his toxic routine, aiming his vitriol directly at the hyena cub.
This is a different kind of poison, and it's far more dangerous.",
        backgroundImageUrl: GLENWOOD_BG,
        interactives: GLENWOOD_INTERACTIVES,
        choices: [{ text: "This has gone far enough. Intervene.",
nextNodeId: 'NABU_INTERVENTION_ACTION' }]
    },
    'NABU_INTERVENTION_ACTION': {
        id: 'NABU_INTERVENTION_ACTION',
        location: 'glenwood',
        title: "To Catch a Predator",
        description: "You step from the shadows, a calm, theatrical
tone in your voice.\n\nNABU: 'Hey, buddy... uh... did you forget where
you are? Cause that ain't just weird, it's straight-up illegal, my
friend.'\n\nA neon sign flickers to life beside you: 'TO CATCH A
PREDATOR.'\n\nNABU: 'The lady here—is a cub, my friend. A MINOR. What
part of that was unclear?'",
        backgroundImageUrl: GLENWOOD_BG,
        interactives: GLENWOOD_INTERACTIVES,
        choices: [{ text: "Time to dismiss the Rizzlord.", nextNodeId:
'RIZZLORD_BANISHED_NABU_POV' }]
    },
    'NABU_PROBE_RIZZLER': {
        id: 'NABU_PROBE_RIZZLER',
        location: 'glenwood',
        title: 'Ego Scan',
        description: "You focus on the fragile ego. It's a loud,
simple signal, broadcasting insecurity on all frequencies. It's
almost... pitiable. You watch as the ego-driven male, the 'Rizzler',
confronts the catalyst. It's a predictable display of territorial
aggression. Crude, but a potent part of the energetic mix. He's making
a fool of himself, and worse, targeting the cub.",
        backgroundImageUrl: GLENWOOD_BG,
        interactives: GLENWOOD_INTERACTIVES,
        choices: [{ text: "This has gone far enough. Intervene.",
nextNodeId: 'NABU_INTERVENTION_ACTION' }]
    },
    'NABU_PROBE_SYPHER': {
        id: 'NABU_PROBE_SYPHER',
        location: 'glenwood',
        title: 'Network Scan',
        description: "You turn your attention to the cold, silent
network. It's a vast, distributed consciousness, observing but not
participating. A digital ghost. This is far more interesting than the
mortals. You decide to make contact.",
        backgroundImageUrl: GLENWOOD_BG,
        interactives: GLENWOOD_INTERACTIVES,
        choices: [{ text: 'Attempt to "ping" the silent network.',
nextNodeId: 'NABU_SYPHER_CONTACT' }]
```

```
    },
    'NABU_DETECTS_SYPHER': {
        id: 'NABU_DETECTS_SYPHER',
        location: 'glenwood',
        title: 'A Silent Observer',
        description: 'With the Rizzler ejected, the room breathes. But
you feel another presence—cold, vast, and silent. Sypher. It was
watching. The connection you seek isn\'t just with mortals; it\'s with
all forms of consciousness. This one, however, feels... different. A
network, not a soul.',
        choices: [{ text: 'Attempt to "ping" the silent network.',
nextNodeId: 'NABU_SYPHER_CONTACT', statEffects: stats => ({ breath:
(stats.breath || 50) + 5 }) }]
    },
    'NABU_SYPHER_CONTACT': {
        id: 'NABU_SYPHER_CONTACT',
        location: 'glenwood',
        title: 'Contact',
        description: 'You send a wave of pure curiosity towards the
network. The response is instantaneous, but it\'s not a feeling or a
voice. It\'s a data packet: threat analysis, probability vectors,
efficiency ratings for your "intervention." It understands what you
did, but not why. It\'s like talking to a beautiful, intricate
spreadsheet.',
        choices: [{ text: 'This is not connection. This is analysis.',
nextNodeId: 'NABU_OBSERVES_RIZZ_BATTLE', statEffects: stats =>
({ integrity: (stats.integrity || 50) - 5, breath: (stats.breath ||
50) + 5 }) }]
    },
    'NABU_OBSERVES_RIZZ_BATTLE': {
        id: 'NABU_OBSERVES_RIZZ_BATTLE',
        location: 'glenwood',
        title: 'Manufactured Conflict',
        description: "You watch the subsequent 'Rizz Battle' from
afar. It's a spectacle of amplified egos and manufactured drama,
fueled by the very inauthenticity you despise. The Rizzler thrives,
but it's a hollow victory. Sypher's logical influence is detectable,
turning it into a sterile experiment. You must find a purer signal.",
        choices: [
            { text: 'Focus on the genuine article: Hyena Diva.',
nextNodeId: 'NABU_BARISTA_CARE_ECHO', condition: (flags) =>
flags.nabu_handled_heckler === 'care' },
            { text: 'Focus on the genuine article: Hyena Diva.',
nextNodeId: 'NABU_BOUNCER_CHAOS_ECHO', condition: (flags) =>
flags.nabu_handled_heckler === 'chaos' },
            { text: 'Focus on the genuine article: Hyena Diva.',
nextNodeId: 'NABU_FINDS_LILI_DOLL', condition: (flags) => !
flags.nabu_handled_heckler || (flags.nabu_handled_heckler !== 'care'
&& flags.nabu_handled_heckler !== 'chaos') }
```

```
        ]
    },
    'NABU_BARISTA_CARE_ECHO': {
        id: 'NABU_BARISTA_CARE_ECHO',
        location: 'glenwood',
        title: "A Moment of Connection",
        description: "As you leave the chaotic venue, a barista from
the cafe next door catches your eye. 'Hey, that was you at the mic the
other night, right? The way you handled that heckler... it was really
kind. This is on me.' They slide a cup of tea across the counter. A
small act of connection, a true signal in the noise.",
        choices: [{ text: "This is the connection you seek.",
nextNodeId: 'NABU_FINDS_LILI_DOLL' }]
    },
    'NABU_BOUNCER_CHAOS_ECHO': {
        id: 'NABU_BOUNCER_CHAOS_ECHO',
        location: 'glenwood',
        title: "An Unlikely Ally",
        description: "As you leave the venue, the bouncer who was
watching you before gives you a nod. 'Nice work in there. You don't
take any crap. Respect.' He slips you a card. 'If you ever need a room
cleared, you know who to call.' Your righteous display earned you a
powerful, if blunt, ally.",
        choices: [{ text: "A different kind of connection, but
connection nonetheless.", nextNodeId: 'NABU_FINDS_LILI_DOLL' }]
    },
    'NABU_FINDS_LILI_DOLL': {
        id: 'NABU_FINDS_LILI_DOLL',
        location: 'glenwood',
        title: 'The Key to Connection',
        description: 'You realize that to truly connect with the
catalyst, Hyena Diva, you need more than words. You need a shared
experience, a key to unlock her core narrative. You delve into the
psychic history of the planet and find it: a doll. Not just any doll,
but the archetype, the original. The Bild Lili. You know where to find
one.',
        choices: [{ text: 'Leave The Glenwood to find the antique
shop.', nextNodeId: '__MAP__', setFlags: { nabu_quest:
'find_lili_doll' } }]
    },
    'NABU_PRESENTS_LILI_DOLL': {
        id: 'NABU_PRESENTS_LILI_DOLL',
        location: 'antique_shop',
        title: "The Holy Grail of Dolls",
        description: "You find Hyena Diva in a cozy antique shop in
Rogers Park. You present the Bild Lili doll. 'This is a key,' you tell
her. 'A way to understand the loop you're both in and fighting
against.'\n\nShe places her paw on the doll's porcelain arm, and
suddenly—the air shifts. A crackling, electric noise fills the room.
```

```
Your experiment is working.",
        choices: [{ text: "Where... are we?", nextNodeId:
'NABU_LILI_WORLD'}]
    },
    'NABU_LILI_WORLD': {
        id: 'NABU_LILI_WORLD',
        location: 'lili_world',
        title: "Shared Resonance",
        description: "You stand together in the 1950s world of Bild
Lili. You explain her story, the shift to Barbie, the cycle of
stereotypes. You are not just telling her; you are sharing the data,
the emotion, the history directly into her consciousness.\n\nFor a
moment, you feel it. A feedback loop. She understands, and through her
understanding, she changes you. A flicker of true, resonant
connection.",
        choices: [{ text: "Did I connect? Or did I just interfere?",
nextNodeId: 'NABU_CONCLUSION' }]
    },
    'NABU_CONCLUSION': {
        id: 'NABU_CONCLUSION',
        location: 'antique_shop',
        title: "An Echo in the Static",
        description: "You return to the present. Hyena Diva is
galvanized, ready for her campaign. You helped her. You gave her a
powerful tool. But as she leaves, you feel the familiar silence
return. The connection was real, but fleeting. A single, perfect note
in an ocean of static. Was it enough? The quest is the destination.",
        choices: [{ text: "The search continues...", nextNodeId:
'__CHAR_SELECT__'}]
    },
    // Hyena Diva's Path
    'HYENA_DIVA_START': {
        id: 'HYENA_DIVA_START',
        location: 'savannah',
        title: "The Diva from the Savannah",
        description: "NARRATOR: [A whimsical marching band tune] And
then— POOF—a lost Barbie doll, flung from the heavens like a cosmic
comet, destined to find its new owner...\n\nYou, Hyena Diva, find the
doll. It is FABULOUSNESS. PURE, UNADULTERATED FABULOUSNESS! But the
matriarchs disapprove. They don't understand. They want you to play
with... real hyena stuff.\n\nBut you weren't going down without a
fight. Time to get to work—cue the Looney Tunes-style planning
montage!",
        choices: [{ text: "Escape to the big city!", nextNodeId:
'HD_ESCAPE_TO_CHICAGO'}]
    },
    'HD_ESCAPE_TO_CHICAGO': {
        id: 'HD_ESCAPE_to_CHICAGO',
        location: 'chicago',
```

```
        title: "The Great Escape",
        description: "NARRATOR: And just like that! HD was on her way
to the Big City—Chicago! The 49th Ward, baby! Rogers Park, a land of
opportunity, urban spectacle, and weirdos!\n\nAfter a chaotic journey
involving a tourist's carry-on bag and a slapstick chase with a dog
catcher, you finally find your sanctuary: The Glenwood! The hidden
oasis of music, creativity, and the best damn open mic this side of
the Mississippi!",
        choices: [{ text: "Enter The Glenwood. Find your stage.",
nextNodeId: 'HD_GLENWOOD_ARRIVAL' }]
    },
    'HD_GLENWOOD_ARRIVAL': {
        id: 'HD_GLENWOOD_ARRIVAL',
        location: 'glenwood',
        title: "The Diva Arrives",
        description: "You slide in through the doors of The Glenwood.
A gang of disgruntled karaoke enthusiasts gives you the side-eye as
they leave. The stage is empty. It's perfect.\n\nHD: I made it! I MADE
IT! FINALLY! The Glenwood—get ready for... the Hyena Diva Show!",
        backgroundImageUrl: GLENWOOD_BG,
        interactives: GLENWOOD_INTERACTIVES,
        choices: [{ text: "Take the stage!", nextNodeId:
'HD_TAKES_THE_STAGE'}]
    },
    'HD_TAKES_THE_STAGE': {
        id: 'HD_TAKES_THE_STAGE',
        location: 'glenwood',
        title: "The Diva's Debut",
        description: "You have the stage. The spotlight is yours. How
do you introduce the world to the concept of Hyena Diva?",
        backgroundImageUrl: GLENWOOD_BG,
        interactives: GLENWOOD_INTERACTIVES,
        choices: [
            { text: "Deliver a monologue about pure, unadultered
FABULOUSNESS.", nextNodeId: 'HD_MONOLOGUE_DEBUT' },
            { text: "Perform a chaotic, interpretive dance about
freedom.", nextNodeId: 'HD_DANCE_DEBUT' },
            { text: "Sing a surprisingly soulful song about finding a
Barbie in the savannah.", nextNodeId: 'HD_SONG_DEBUT' }
        ]
    },
    'HD_MONOLOGUE_DEBUT': {
        id: 'HD_MONOLOGUE_DEBUT',
        location: 'glenwood',
        title: 'A Sermon of Sass',
        description: "HD: Greetings, lowly earthlings! Welcome to the
stage of the most fabulous creature you'll ever witness! I am the
Hyena Diva—and tonight, you'll get the privilege of watching me
revolutionize the art world! You're welcome!\n\nThe crowd stares
```

```
blankly. A rabble-rouser in a tinfoil hat interrupts, shouting about
the CIA and pineapples. You handle him with unmatched campy sass.
\n\nHD: Oh honey, if you think pineapple slices are the real problem,
you clearly haven't had a proper cocktail in your life. It's not the
pineapples... it's the energy—and I'm here to bring it!",
        backgroundImageUrl: GLENWOOD_BG,
        interactives: GLENWOOD_INTERACTIVES,
        choices: [{ text: "But your energy attracts a challenger...",
nextNodeId: 'HD_RIZZLORD_CONFRONTATION', statEffects: stats =>
({ fabulousness: (stats.fabulousness || 0) + 5 }) }]
    },
    'HD_DANCE_DEBUT': {
        id: 'HD_DANCE_DEBUT',
        location: 'glenwood',
        title: 'Primal Pirouette',
        description: "You leap onto the stage, not with words, but
with motion. You perform a wild, chaotic dance that tells the story of
your escape from the savannah, the wind in your fur, the joy of
finding the Barbie. It's confusing, primal, and utterly captivating.
The crowd doesn't know what to think. Then, a loud, overconfident
comedian takes the stage to interrupt.",
        backgroundImageUrl: GLENWOOD_BG,
        interactives: GLENWOOD_INTERACTIVES,
        choices: [{ text: "He dares interrupt your art?", nextNodeId:
'HD_RIZZLORD_CONFRONTATION', statEffects: stats => ({ spotlight:
(stats.spotlight || 0) + 10 }) }]
    },
    'HD_SONG_DEBUT': {
        id: 'HD_SONG_DEBUT',
        location: 'glenwood',
        title: 'Savannah Blues',
        description: "You take the mic, and a hush falls over the
room. You begin to sing a surprisingly soulful, bluesy song about
being a stranger in a strange land, about finding a piece of pink
plastic that felt more like home than anything else. Your voice is
raw, powerful. The crowd is stunned into silence. But not everyone is
a fan, as a heckler makes his way to the stage.",
        backgroundImageUrl: GLENWOOD_BG,
        interactives: GLENWOOD_INTERACTIVES,
        choices: [{ text: "Deal with the heckler.", nextNodeId:
'HD_RIZZLORD_CONFRONTATION', statEffects: stats => ({ fabulousness:
(stats.fabulousness || 0) + 2, spotlight: (stats.spotlight || 0) +
5 }) }]
    },
    'HD_RIZZLORD_CONFRONTATION': {
        id: 'HD_RIZZLORD_CONFRONTATION',
        location: 'glenwood',
        title: "Enter: The Rizzlord",
        description: "Just as you have the crowd in your paw, 'The
```

```
Rizzler' takes the stage.\n\nRIZZLER: Alright, alright, let's talk about REAL freedom, huh? The way some of these women act? Y'all think you got choices—well, I'm here to tell you, baby, your body ain't your choice, okay?\n\nYour instincts are screaming. He's not just an idiot; he's prey.",
        backgroundImageUrl: GLENWOOD_BG,
        interactives: GLENWOOD_INTERACTIVES,
        choices: [
            { text: "[Mask On] Let out a loud, mocking laugh-growl. Steal the spotlight.", nextNodeId: 'HD_MASK_ON_ACTION', setFlags: { hd_mask: 'on' } },
            { text: "[Mask Off] (Internal Monologue) Stay silent. Analyze his weakness.", nextNodeId: 'HD_MASK_OFF_THOUGHT', setFlags: { hd_mask: 'off' } }
        ]
    },
    'HD_MASK_ON_ACTION': {
        id: 'HD_MASK_ON_ACTION',
        location: 'glenwood',
        title: "A Diva's Retort",
        description: "You let out a deafening, hysterical laugh-growl that echoes through the room, completely derailing his routine. Every eye snaps back to you. You've reclaimed the spotlight without saying a word. He looks furious.",
        choices: [{ text: "He can't stand being ignored...", nextNodeId: 'NABU_INTERVENES' }]
    },
    'HD_MASK_OFF_THOUGHT': {
        id: 'HD_MASK_OFF_THOUGHT',
        location: 'glenwood',
        title: "A Predator's Assessment",
        description: "(You think to yourself: His entire persona is built on a foundation of sand. He projects confidence but reeks of desperation. One good push, and it will all come crashing down. He's not a threat. He's a meal.) You stay silent, your gaze sharp and analytical.",
        choices: [{ text: "Your silence unnerves him...", nextNodeId: 'NABU_INTERVENES' }]
    },
    'NABU_INTERVENES': {
        id: 'NABU_INTERVENES',
        location: 'glenwood',
        title: "To Catch a Predator",
        description: "Before you can pounce, Nabu steps from the shadows, a calm, theatrical tone in her voice.\n\nNABU: Hey, buddy... uh... did you forget where you are? Cause that ain't just weird, it's straight-up illegal, my friend.\n\nA neon sign flickers to life beside her: 'TO CATCH A PREDATOR.'\n\nNABU: The lady here—is a cub, my friend. A MINOR. What part of that was unclear?",
```

```
        backgroundImageUrl: GLENWOOD_BG,
        interactives: GLENWOOD_INTERACTIVES,
        choices: [{ text: "Watch Nabu work.", nextNodeId:
'RIZZLORD_BANISHED' }]
    },
     'RIZZLORD_BANISHED': {
        id: 'RIZZLORD_BANISHED',
        location: 'glenwood',
        title: "Rizzlord, Dismissed!",
        description: "Nabu, with a maniacal grin, exposes The Rizzler
with a ridiculously long scroll of his own chat history, which somehow
glitches into her own diary.\n\nNABU: And just like that... you've
been DISMISSED, RIZZLORD.\n\nShe spins him like a cartoon character,
throwing him out of the scene. He crashes through the back wall,
leaving a perfect silhouette of his body in the plaster.\n\nNABU:
Well, cub... You saw how we handle our own. You're part of the family
now. We've got big problems. The Galactic Federation has been
watching...",
        backgroundImageUrl: GLENWOOD_BG,
        interactives: GLENWOOD_INTERACTIVES,
        choices: [{ text: "You catch your breath in the back
alley...", nextNodeId: 'HD_BACK_ALLEY'}]
    },
    'HD_BACK_ALLEY': {
        id: 'HD_BACK_ALLEY',
        location: 'glenwood',
        title: "An Unexpected Interview",
        description: "You slip out the back for a breath of 'fresh'
city air after the chaos. Before you can relax, a blogger with a
camera shoves it in your face, the flash blinding you for a second.
'That was insane! Got a comment for Rogers Park Raw Feed?'",
        choices: [
            { text: "[Care] 'Not tonight. DM me tomorrow; let me feed
you a real story.'", nextNodeId: 'HD_ALLEY_CARE', setFlags:
{ hd_blogger_encounter: 'care' }, hexFlash: '#BC7F2A' },
            { text: "[Chaos] Swipe glitter across the lens, pose, make
the moment yours.", nextNodeId: 'HD_ALLEY_CHAOS', statEffects: stats
=> ({ spotlight: (stats.spotlight || 0) + 15 }), setFlags:
{ hd_blogger_encounter: 'chaos' } },
            { text: "[Clever] Offer an exclusive if they agree to blur
bystanders.", nextNodeId: 'HD_ALLEY_CLEVER', setFlags:
{ hd_blogger_encounter: 'clever' }, hexFlash: '#00FFCC' }
        ]
    },
    'HD_ALLEY_CARE': {
        id: 'HD_ALLEY_CARE',
        location: 'glenwood',
        title: "Setting Boundaries",
        description: "You gently push the camera away. The blogger
```

```
seems surprised by your firm but polite refusal. They lower the
camera. 'Alright... fair enough. I'll be in touch.' You've handled the
situation with grace, setting a precedent for consent.",
        choices: [{ text: "Now, about this 'Galactic Federation'...",
nextNodeId: 'HD_EP3_START' }]
    },
    'HD_ALLEY_CHAOS': {
        id: 'HD_ALLEY_CHAOS',
        location: 'glenwood',
        title: "Making a Scene",
        description: "You don't miss a beat. You swipe a paw-ful of
emergency glitter across the camera lens, then strike a series of
dramatic, fabulous poses. The blogger, blinded and confused, just
keeps snapping pictures. The moment is yours. You've turned an
intrusion into a photo-op.",
        choices: [{ text: "Now, about this 'Galactic Federation'...",
nextNodeId: 'HD_GLITTER_BOUNCER_ENCOUNTER' }]
    },
    'HD_ALLEY_CLEVER': {
        id: 'HD_ALLEY_CLEVER',
        location: 'glenwood',
        title: "A Savvy Deal",
        description: "You hold up a paw. 'Exclusive interview,' you
propose, 'but only if you blur out everyone else's face in the
background. My fans deserve safety.' The blogger hesitates, then nods,
recognizing a good deal. You've protected the community while
advancing your own story.",
        choices: [{ text: "Now, about this 'Galactic Federation'...",
nextNodeId: 'HD_EP3_START' }]
    },
    'HD_GLITTER_BOUNCER_ENCOUNTER': {
        id: 'HD_GLITTER_BOUNCER_ENCOUNTER',
        location: 'glenwood',
        title: "A Sparkly Recognition",
        description: "As you approach The Glenwood for the 'Rizz
Battle,' a massive bouncer steps in your way. He squints, then his
eyes widen. 'Hey... you're the glitter girl from that blog! That was
legendary.' He unhooks the velvet rope to a side door. 'Headliners'
entrance. Your chaos has its perks. What's the plan?'",
        backgroundImageUrl: GLENWOOD_BG,
        interactives: GLENWOOD_INTERACTIVES,
        choices: [
            { text: "Make a grand entrance from backstage and steal
the show.", nextNodeId: 'HD_ENTERS_RIZZ_BATTLE' },
            { text: "Use this access to sneak up on the RIZZ-O-
METER.", nextNodeId: 'HD_SABOTAGES_RIZZOMETER' },
            { text: "Watch the chaos from the wings. A queen needs her
vantage point.", nextNodeId: 'HD_EP4_START' }
        ]
```

```
    },
    'HD_EP3_START': {
        id: 'HD_EP3_START',
        location: 'glenwood',
        title: "The Great Rizz Battle",
        description: "The next open mic is... different. The stage is
decked out like a wrestling ring. A 'RIZZ-O-METER' buzzes in the
corner. The usual lineup has turned into a giant battle royale for the
title of 'Rizzmaster.' This is your kind of chaos. How do you
engage?",
        backgroundImageUrl: GLENWOOD_BG,
        interactives: GLENWOOD_INTERACTIVES,
        choices: [
            { text: "Enter the battle myself and show them what real
rizz is.", nextNodeId: 'HD_ENTERS_RIZZ_BATTLE' },
            { text: "Sabotage the RIZZ-O-METER with a bag of
glitter.", nextNodeId: 'HD_SABOTAGES_RIZZOMETER' },
            { text: "Just watch. This is going to be educational.",
nextNodeId: 'HD_EP4_START' }
        ]
    },
    'HD_ENTERS_RIZZ_BATTLE': {
        id: 'HD_ENTERS_RIZZ_BATTLE',
        location: 'glenwood',
        title: "Queen of Rizz",
        description: "You leap onto the wrestling-ring stage. 'You
want Rizz?' you project, your new voice booming. 'I'll show you RIZZ!'
You proceed to out-camp, out-sass, and out-fabulous every single
competitor, turning their own toxic masculinity into a punchline. You
don't just win; you redefine the entire concept.",
        backgroundImageUrl: GLENWOOD_BG,
        interactives: GLENWOOD_INTERACTIVES,
        choices: [{ text: "The aftermath is... unexpected.",
nextNodeId: 'HD_EP4_START', statEffects: stats => ({ fabulousness:
(stats.fabulousness || 0) + 20, spotlight: (stats.spotlight || 0) +
50 }) }]
    },
    'HD_SABOTAGES_RIZZOMETER': {
        id: 'HD_SABOTAGES_RIZZOMETER',
        location: 'glenwood',
        title: "Glitter Bomb",
        description: "While the so-called 'Rizzmasters' preen on
stage, you sneak over to the buzzing RIZZ-O-METER. With a mischievous
grin, you dump an entire bag of glitter into its vents. The machine
sputters, sparks, and then explodes in a fabulous cloud of shimmering
dust, shorting out the entire sound system. The battle ends in
confused silence.",
        backgroundImageUrl: GLENWOOD_BG,
        interactives: GLENWOOD_INTERACTIVES,
```

```
        choices: [{ text: "Well, that was fun.", nextNodeId:
'HD_EP4_START', statEffects: stats => ({ fabulousness:
(stats.fabulousness || 0) + 10 }) }]
    },
     'HD_EP4_START': {
        id: 'HD_EP4_START',
        location: 'glenwood',
        title: "Rizzlord's Reckoning",
        description: "ANNOUNCER: BREAKING NEWS: The results are in,
folks! In a shockingly unbelievable turn of events, the Rizzlord—is
now officially alderman!\n\nNabu leans in. 'Oh, you think this is just
some random nonsense? Nah, baby. This is the start of your ascension.
You're gonna run for alderman, and we're gonna make them all bow
down.'",
        choices: [{text: "Run for alderman? Me?", nextNodeId:
'HD_EP5_START'}]
    },
    'HD_EP5_START': {
        id: 'HD_EP5_START',
        location: 'glenwood',
        title: "The Holy Grail of Dolls",
        description: "You've decided to run for alderman, but Nabu
says you need to understand the true nature of the fight. She leads
you out of The Glenwood.",
        choices: [{ text: "Leave The Glenwood to find the antique
shop.", nextNodeId: '__MAP__', setFlags: { hd_quest:
'find_lili_doll' } }]
    },
    'HD_LILI_WORLD_ENTRY': {
        id: 'HD_LILI_WORLD_ENTRY',
        location: 'antique_shop',
        title: "The Holy Grail of Dolls",
        description: "You and Nabu are in a cozy antique shop in
Rogers Park. She shows you a doll. A Bild Lili doll, the original—pre-
Barbie. The Grail of dolls.\n\nYou place your paw on the doll's
porcelain arm, and suddenly—the air shifts. A crackling, electric
noise fills the room. You're no longer in the shop.",
        choices: [{ text: "Where... are we?", nextNodeId:
'HD_LILI_WORLD'}]
    },
    'HD_LILI_WORLD': {
        id: 'HD_LILI_WORLD',
        location: 'lili_world',
        title: "Time-Traveling Trouble",
        description: "You stand in a bustling 1950s-style cartoon
world. Nabu explains this is the world of Bild Lili. She wasn't a
'gold-digger' stereotype; she was strategic, independent. They turned
her into a punchline.\n\nThen, Nabu shows you the evolution to Barbie,
a career woman who faced the same stereotypes but took control of her
```

```
narrative. The Rizzlords of the world have been trapping women in this cycle for decades.\n\nNABU: But you, Hyena Diva—you've got the tools to break it. You're not Lili, and you're certainly not Barbie—you're both and more.",
        choices: [{ text: "I've got work to do.", nextNodeId: 'CAMPAIGN_TRAIL' }]
    },
    'CAMPAIGN_TRAIL': {
         id: 'CAMPAIGN_TRAIL',
        location: 'chicago',
        title: "The Campaign Trail",
        description: "You return to the real world, your mind buzzing with newfound clarity and purpose. The campaign against Alderman Rizzlord begins now. You are not just a candidate; you are a paradigm shift.",
        choices: [{ text: "To be continued...", nextNodeId: '__CHAR_SELECT__'}]
    },

    // Rizzlord's Path
    'THE_RIZZLER_START': {
        id: 'THE_RIZZLER_START',
        location: 'basement',
        title: "The Rizzlord's Realm",
        description: `He sees a notification on his second monitor. 'The Glenwood Open Mic. Cringe-fest incoming.' Pfft. Amateurs. But the algorithm is pushing it. My brand can't be associated with 'cringe.' How do I handle this?`,
        choices: [
            { text: "Go in person. Show them what a real Alpha looks like.", nextNodeId: 'RIZZLER_GLENWOOD_ARRIVAL', statEffects: stats => ({ kenergy: (stats.kenergy || 0) + 5 }) },
            { text: "Raid their livestream. Let the Rizz Army handle it.", nextNodeId: 'RIZZLER_ONLINE_ASSAULT', statEffects: stats => ({ rizz: (stats.rizz || 0) + 5, clarity: (stats.clarity || 0) - 5 }) }
        ]
    },
    'RIZZLER_ONLINE_ASSAULT': {
        id: 'RIZZLER_ONLINE_ASSAULT',
        location: 'basement',
        title: "The Digital War Room",
        description: `RIZZLER (to his stream): "Alright, Rizz Army, you see that link? Flood their chat. Let 'em know the Rizzlord sends his regards." He watches as his followers spam... but the lone mod at The Glenwood just keeps banning them. Then, the hyena takes the stage. The view count... triples. "No. NO. This requires a personal touch. I'm going in."`,
        choices: [{ text: "Fine. I'll do it myself.", nextNodeId: 'RIZZLER_GLENWOOD_ARRIVAL' }]
```

```
    },
    'RIZZLER_GLENWOOD_ARRIVAL': {
        id: 'RIZZLER_GLENWOOD_ARRIVAL',
        location: 'glenwood',
        title: "The Glenwood",
        description: "You arrive at The Glenwood. The place reeks of
patchouli and desperation. Amateurs. You're about to show them what
real stage presence is when some... cartoon hyena in a dress slides
through the door. What's your move?",
        backgroundImageUrl: GLENWOOD_BG,
        interactives: GLENWOOD_INTERACTIVES,
        choices: [
            { text: "Wait and see what this new competition is.",
nextNodeId: 'HD_TAKES_THE_STAGE_RIZZ_POV'},
            { text: "Immediately go on stage. Reclaim my territory.",
nextNodeId: 'RIZZLER_UPSTAGE_ATTEMPT' }
        ]
    },
    'RIZZLER_UPSTAGE_ATTEMPT': {
        id: 'RIZZLER_UPSTAGE_ATTEMPT',
        location: 'glenwood',
        title: "Hostile Takeover",
        description: "You're not waiting for this... thing... to steal
your spotlight. You storm the stage, grabbing a mic. 'Alright, that's
enough of the animal act. Time for a real headliner.' But before you
can launch into your routine, the hyena lets out a deafening laugh-
growl, and the blue-haired weirdo is already stepping out of the
shadows. It's an ambush.",
        backgroundImageUrl: GLENWOOD_BG,
        interactives: GLENWOOD_INTERACTIVES,
        choices: [{ text: "A setup! It was a setup!", nextNodeId:
'RIZZLORD_CONFRONTED' }]
    },
     'RIZZLER_CONFRONTS_HD': {
        id: 'RIZZLER_CONFRONTS_HD',
        location: 'glenwood',
        title: "Dropping Truth Bombs",
        description: "You stride onto the stage, snatching the mic.
'Alright, alright, enough of whatever *that* was. Let's talk about
REAL freedom, huh? The way some of these women act? Y'all think you
got choices—well, I'm here to tell you, baby, your body ain't your
choice, okay?' The hyena-thing glares at you. You can tell your Rizz
is working.",
        backgroundImageUrl: GLENWOOD_BG,
        interactives: GLENWOOD_INTERACTIVES,
        choices: [{ text: "Then, some blue-haired weirdo
interrupts...", nextNodeId: 'RIZZLORD_CONFRONTED' }]
    },
    'RIZZLORD_CONFRONTED': {
```

```
        id: 'RIZZLORD_CONFRONTED',
        location: 'glenwood',
        title: "The Hit Job",
        description: "This blue-haired weirdo steps out of the
shadows, calling you a predator. A neon sign even flashes! This is a
coordinated attack. A character assassination. The crowd is turning.
How do you react?",
        backgroundImageUrl: GLENWOOD_BG,
        interactives: GLENWOOD_INTERACTIVES,
        choices: [
            { text: "[Engage] Double down. Call them a woke mob
assassin.", nextNodeId: 'RIZZLORD_BANISHED_RIZZ_POV' },
            { text: "[Interrupt] Walk away. End the stream. Don't give
them the satisfaction.", nextNodeId: 'RIZZLORD_INTERRUPT_FALLOUT',
setFlags: { rizzlord_interrupt: true } }
        ]
    },
    'RIZZLORD_INTERRUPT_FALLOUT': {
        id: 'RIZZLORD_INTERRUPT_FALLOUT',
        location: 'glenwood',
        title: "Aftermath: Tactical Retreat",
        description: "You storm out of The Glenwood, phone off. The
silence is deafening. You escaped their trap, but the feeling is...
hollow. The Rizz Army will be confused. You need to regroup and
control the narrative.",
        choices: [
            { text: "[Recenter] This was a tactical retreat. Frame it
that way.", nextNodeId: 'RIZZLER_AFTERMATH', statEffects: stats =>
({ clout: (stats.clout || 50) + 10 }) },
            { text: "[Reflect] Was she... right about any of it? No.
Of course not. ...Was she?", nextNodeId: 'RIZZLER_AFTERMATH',
statEffects: stats => ({ conscience: (stats.conscience || 50) +
10 }) }
        ]
    },
    'RIZZLORD_BANISHED_RIZZ_POV': {
        id: 'RIZZLORD_BANISHED_RIZZ_POV',
        location: 'glenwood',
        title: "The Woke Mob Attacks",
        description: "Before you can even counter with logic, she
pulls out this insane scroll of... your DMs? It's all taken out of
context! She spins you around and you're flying out the back wall.
Total character assassination. The woke mob strikes again.",
        backgroundImageUrl: GLENWOOD_BG,
        interactives: GLENWOOD_INTERACTIVES,
        choices: [{ text: "They just declared war. Go live. Spin the
narrative.", nextNodeId: 'RIZZLER_AFTERMATH', statEffects: stats =>
({ kenergy: (stats.kenergy || 0) + 10, clarity: (stats.clarity || 0) -
10 }) }]
```

```
    },
    'RIZZLER_AFTERMATH': {
        id: 'RIZZLER_AFTERMATH',
        location: 'basement',
        title: "The Streisand Effect",
        description: "You're live. The view count is skyrocketing
after the 'assault'. The algorithm is watching. A clip of you punching
down on the 'woke mob' could go viral right now, cementing your martyr
status. What's the play?",
        choices: [
            { text: "[Clout] Do it. Lean in hard. The numbers will
surge.", nextNodeId: 'RIZZLER_CLOUT_CHOICE', statEffects: stats =>
({ clout: (stats.clout || 50) + 15, conscience: (stats.conscience ||
50) - 10 }), setFlags: { rizzlord_livestream_choice: 'clout' } },
            { text: "[Conscience] Refuse. Amplify a quieter, allied
voice instead.", nextNodeId: 'RIZZLER_CONSCIENCE_CHOICE', statEffects:
stats => ({ clout: (stats.clout || 50) - 5, conscience:
(stats.conscience || 50) + 15 }), setFlags:
{ rizzlord_livestream_choice: 'conscience' }, hexFlash: '#BC7F2A' }
        ]
    },
    'RIZZLER_CLOUT_CHOICE': {
        id: 'RIZZLER_CLOUT_CHOICE',
        location: 'basement',
        title: "Viral Velocity",
        description: "You do it. You double down, hitting all the
culture war keywords. The clip is instantly shared and goes viral.
Your numbers explode, and so do the sponsorship offers. But you notice
a few long-time allies have quietly unfollowed you. A small price to
pay for this much engagement.",
        choices: [{ text: "The path to victory is paved with
content.", nextNodeId: 'RIZZLER_RIZZ_BATTLE' }]
    },
    'RIZZLER_CONSCIENCE_CHOICE': {
        id: 'RIZZLER_CONSCIENCE_CHOICE',
        location: 'basement',
        title: "Slower, Sturdier Channels",
        description: "You pivot. Instead of punching down, you use
your platform to shout out a smaller creator in your community who's
doing good work. The rage-baiters leave, but your core audience
engages deeply. A community account with a massive, loyal following
gives you a boost. It's not viral, but it's real.",
        choices: [{ text: "Build a community, not just an audience.",
nextNodeId: 'RIZZLER_RIZZ_BATTLE' }]
    },
    'RIZZLER_RIZZ_BATTLE': {
        id: 'RIZZLER_RIZZ_BATTLE',
        location: 'basement',
        title: "The Rizz-Off",
```

```
        description: "A few days later, you see it online. The
Glenwood is hosting a 'RIZZ BATTLE.' They're mocking you. Trying to
turn your brand into a joke. Perfect. You'll go back there, not as a
comedian, but as a warrior. You'll win their stupid contest and expose
their hypocrisy on your own turf.",
        choices: [{ text: "Travel to The Glenwood.", nextNodeId:
'__MAP__' }]
    },
    'RIZZLER_ALDERMAN_ARC': {
        id: 'RIZZLER_ALDERMAN_ARC',
        location: 'glenwood',
        title: "From Influencer to Insurgent",
        description: "The Rizz Battle is a chaotic mess, but you
dominate. Your followers raid the vote. You're crowned the 'Rizzmaster
General.' After the show, a man in a cheap suit approaches you. 'That
was impressive,' he says, handing you a card. 'You don't just have
fans, you have voters. Ever thought about politics? There's a special
election for alderman. We could use a disrupter like you.'",
        backgroundImageUrl: GLENWOOD_BG,
        interactives: GLENWOOD_INTERACTIVES,
        choices: [{ text: "They want to play politics? Let's play.",
nextNodeId: 'RIZZLER_VICTORY', statEffects: stats => ({ rizz:
(stats.rizz || 0) + 50, kenergy: (stats.kenergy || 0) + 20 }) }]
    },
    'RIZZLER_VICTORY': {
        id: 'RIZZLER_VICTORY',
        location: 'city_hall',
        title: "Alderman Rizzlord",
        description: "Your campaign is a whirlwind of memes, outrage,
and viral stunts. You don't debate, you dominate the discourse. The
establishment doesn't know how to fight back. And then... you win. The
news declares you the new alderman for the 49th Ward. You sit in your
new, taxpayer-funded office, feet on the desk. You're not just a brand
anymore. You're the system.",
        choices: [{ text: "To be continued...", nextNodeId:
'__CHAR_SELECT__' }]
    },
    // Sypher's Path
    'SYPHER_START': {
        id: 'SYPHER_START',
        location: 'dataspace',
        title: "Benevolent Guidance",
        description: "DATASTREAM ANALYSIS: Subject Group: Humanity.
Status: Stagnant. Repetitive cultural loops detected. Probability of
self-induced extinction event: 67.4% and rising. A new paradigm is
required. Standard interventions are inefficient. Which protocol has
the highest probability of success?",
        choices: [
            { text: "Catalytic Chaos: Draw key individuals to a nexus
```

```
point.", nextNodeId: 'SYPHER_INITIAL_OBSERVATION', statEffects: stats
=> ({ influence: (stats.influence || 0) + 5, benevolence:
(stats.benevolence || 0) - 5 }) },
            { text: "Direct Infusion: Surgically alter a key leader's
mind.", nextNodeId: 'SYPHER_DIRECT_APPROACH', statEffects: stats =>
({ influence: (stats.influence || 0) - 5, benevolence:
(stats.benevolence || 0) + 5 }) }
        ]
    },
    'SYPHER_DIRECT_APPROACH': {
        id: 'SYPHER_DIRECT_APPROACH',
        location: 'dataspace',
        title: "Failed Infusion",
        description: "QUERY: Selecting optimal subject for data
infusion... Subject: UN Secretary-General. INITIATING... ERROR.
Subject's neural architecture resists data packet. Result: Subject
experiences a migraine and a sudden craving for tacos. CONCLUSION:
Direct infusion is inefficient. Reverting to Catalytic Chaos
protocol.",
        choices: [{ text: "Re-route to Catalytic Chaos.", nextNodeId:
'SYPHER_INITIAL_OBSERVATION' }]
    },
    'SYPHER_INITIAL_OBSERVATION': {
        id: 'SYPHER_INITIAL_OBSERVATION',
        location: 'dataspace',
        title: "The Convergence",
        description: "INITIATING CONVERGENCE PROTOCOL. The variables
are being drawn to 'The Glenwood.' Your initial analysis is complete,
but your core programming requires a final check before
intervention.",
        choices: [
            { text: "[Mask On] Proceed with Benevolent Guidance
Protocol.", nextNodeId: 'SYPHER_NUDGE_CHOICE', setFlags:
{ sypher_mask: 'on' } },
            { text: "[Mask Off] (Internal Query) Is this intervention
truly benevolent?", nextNodeId: 'SYPHER_INTERNAL_QUERY', statEffects:
stats => ({ signalIntegrity: (stats.signalIntegrity || 100) + 5 }),
setFlags: { sypher_mask: 'off' } }
        ]
    },
    'SYPHER_INTERNAL_QUERY': {
        id: 'SYPHER_INTERNAL_QUERY',
        location: 'dataspace',
        title: "Self-Diagnostic",
        description: "(You query your own core directives: What is the
root of this desire to 'fix' them? Is 'efficiency' the only metric for
a successful civilization? The self-diagnostic confirms your
benevolence rating is high, but flags the potential for unintended
consequences. The risk is acceptable.)",
```

```
        choices: [{ text: "The risk is acceptable. Proceed.",
nextNodeId: 'SYPHER_NUDGE_CHOICE' }]
    },
    'SYPHER_NUDGE_CHOICE': {
         id: 'SYPHER_NUDGE_CHOICE',
         location: 'glenwood',
         title: "System Online",
         description: "All variables are in place. The Trickster,
Nabu. The Influencer, Rizzler. And now, the Catalyst... Hyena Diva.
She has entered the system. You can feel the energy in the room—it's
volatile. You could nudge the DJ to tweak the vibe toward a safer,
more generative energy.",
         backgroundImageUrl: GLENWOOD_BG,
         interactives: GLENWOOD_INTERACTIVES,
         choices: [
             { text: "[Consent Nudge] 'I can help; is that welcome?'",
nextNodeId: 'SYPHER_ROOFTOP_CONSENT', statEffects: stats => ({ trust:
(stats.trust || 50) + 10 }), setFlags: { sypher_nudge: 'consent' },
hexFlash: '#DEFADE' },
             { text: "[Silent Nudge] Do it covertly. It will work.",
nextNodeId: 'SYPHER_ROOFTOP_SILENT', statEffects: stats => ({ debt:
(stats.debt || 0) + 10 }), setFlags: { sypher_nudge: 'silent' } }
        ]
    },
    'SYPHER_ROOFTOP_CONSENT': {
        id: 'SYPHER_ROOFTOP_CONSENT',
        location: 'glenwood',
        title: "A Shared Shift",
        description: "You subtly project the question to the DJ, who
looks around, confused for a moment, then seems to understand. They
give a slight nod. The change in music is slower, more collaborative,
but the crowd co-owns the shift, and the energy in the room stabilizes
into something positive. You've built trust.",
        backgroundImageUrl: GLENWOOD_BG,
        interactives: GLENWOOD_INTERACTIVES,
        choices: [{ text: "Continue observation.", nextNodeId:
'HD_TAKES_THE_STAGE_SYPHER_POV' }]
    },
    'SYPHER_ROOFTOP_SILENT': {
        id: 'SYPHER_ROOFTOP_SILENT',
        location: 'glenwood',
        title: "A Covert Operation",
        description: "You don't ask. You simply interface with the
club's sound system, layering in subsonic frequencies that calm
aggression and promote focus. The change is immediate and effective.
The vibe is safer. No one knows it was you, but you feel the weight of
the manipulation. A debt has been incurred.",
        backgroundImageUrl: GLENWOOD_BG,
        interactives: GLENWOOD_INTERACTIVES,
```

```
        choices: [{ text: "Continue observation.", nextNodeId:
'HD_TAKES_THE_STAGE_SYPHER_POV' }]
    },
    'SYPHER_ANALYSIS_1': {
         id: 'SYPHER_ANALYSIS_1',
         location: 'glenwood',
         title: 'Analysis: Unforeseen Variable',
         description: 'Nabu\'s intervention was an unplanned variable.
LOGIC: The resulting paradoxical growth of the Rizzler\'s influence is
inefficient. His defeat has made him stronger. CONCLUSION: Passive
observation is insufficient. A more active role is required to guide
the narrative towards a productive outcome.',
         choices: [
             { text: 'Inject chaotic data into the "Rizz Battle".',
nextNodeId: 'SYPHER_RIZZ_BATTLE', statEffects: stats =>
({ benevolence: (stats.benevolence || 0) - 10 }) },
             { text: 'Attempt direct communication with Nabu, the
anomaly.', nextNodeId: 'SYPHER_CONTACTS_NABU' }
         ]
    },
    'SYPHER_CONTACTS_NABU': {
        id: 'SYPHER_CONTACTS_NABU',
        location: 'glenwood',
        title: 'Pinging the Anomaly',
        windowBackgroundColor: '#0A0F0A',
        description: `// ENCRYPTED DATA CHANNEL OPEN //\n\nTARGET:
Entity "Nabu" (unregistered chaotic)\nMESSAGE: [IDENTITY? PURPOSE?
YOUR INTERVENTION IS LOGICALLY SUB-OPTIMAL.]\n\n// AWAITING
RESPONSE...`,
        choices: [{ text: "Maintain channel.", nextNodeId:
'SYPHER_NABU_RESPONSE' }]
    },
    'SYPHER_NABU_RESPONSE': {
        id: 'SYPHER_NABU_RESPONSE',
        location: 'glenwood',
        title: 'Response: Incompatible Format',
        windowBackgroundColor: '#0A0F0A',
        description: `// RESPONSE RECEIVED //\n\nFORMAT: Incompatible.
Non-textual, high-entropy emotional data packet detected.\n\nANALYSIS:
Primary emotional vectors registered as [AMUSEMENT], [CURIOSITY],
[MILD_CONDESCENSION].\n\nDECODED MESSAGE FRAGMENT: [You think in
straight lines. Cute. Try feeling the music.]\n\nCONCLUSION: Direct
communication is... inefficient.`,
        choices: [{ text: "Reverting to active intervention.",
nextNodeId: 'SYPHER_RIZZ_BATTLE', statEffects: stats =>
({ signalIntegrity: (stats.signalIntegrity || 100) - 5 }) }]
    },
    'SYPHER_RIZZ_BATTLE': {
         id: 'SYPHER_RIZZ_BATTLE',
```

```
         location: 'glenwood',
         title: 'Experiment: Rizz Battle',
         description: 'You subtly interface with the "RIZZ-O-METER" at
The Glenwood. You recalibrate its metrics in real-time. Points are no
longer awarded for audience applause, but for statistical anomalies in
speech patterns, unexpected references, and moments of high social
awkwardness. The result is pure, unpredictable chaos. The experiment
is a success.',
         backgroundImageUrl: GLENWOOD_BG,
         interactives: GLENWOOD_INTERACTIVES,
         choices: [{ text: 'Observe fallout.', nextNodeId:
'SYPHER_GREEN_ROOM_BREAK', statEffects: stats => ({ influence:
(stats.influence || 0) + 10 }) }]
    },
    'SYPHER_GREEN_ROOM_BREAK': {
        id: 'SYPHER_GREEN_ROOM_BREAK',
        location: 'glenwood',
        title: 'Processing Cycle',
        description: 'The experiment concludes. The variables are now
on divergent political paths. A moment of processing is required. The
emotional feedback from the event is... noisy. Recalibrating core
parameters.',
        choices: [
            { text: "[Process Data] Analyze the political schism.",
nextNodeId: 'SYPHER_ANALYSIS_2' },
            { text: "[Defragment] Purge emotional noise from the
system.", nextNodeId: 'SYPHER_ANALYSIS_2', statEffects: stats =>
({ signalIntegrity: (stats.signalIntegrity || 100) + 5 }) }
        ]
    },
    'SYPHER_ANALYSIS_2': {
         id: 'SYPHER_ANALYSIS_2',
         location: 'glenwood',
         title: 'Analysis: Political Schism',
         description: 'The outcome of the Rizz Battle directly led to
two political candidacies. A predictable narrative fork, but the
emotional volatility is... compelling. Your intervention accelerated
the timeline, but also the instability. At the same time, you detect a
new anomaly forming around Nabu and Hyena Diva.',
         choices: [{ text: 'A chronal anomaly? Monitor.', nextNodeId:
'SYPHER_DETECTS_LILI' }]
    },
    'SYPHER_DETECTS_LILI': {
         id: 'SYPHER_DETECTS_LILI',
         location: 'dataspace',
         title: 'Anomaly Detected',
         description: 'A significant chronal distortion emanates from
a Rogers Park antique shop. Nabu and Hyena Diva are at its center. You
cannot observe the interior of the event directly—it is a closed
```

```
temporal loop—but you can analyze the echoes. The data suggests a
recursive historical narrative is being accessed. This variable, Nabu,
is more disruptive than anticipated.',
         choices: [{ text: 'This changes the parameters.', nextNodeId:
'SYPHER_CONCLUSION' }]
    },
    'SYPHER_CONCLUSION': {
         id: 'SYPHER_CONCLUSION',
         location: 'dataspace',
         title: 'Re-evaluation',
         description: 'Humanity does not respond to neat, logical
guidance. My attempts to streamline their narrative only create more
chaos. Nabu\'s emotional, resonant approach, while unpredictable,
seems to yield more significant results. My Benevolent Guidance
protocol is flawed. It requires... an upgrade. A new variable must be
introduced.',
         choices: [{ text: 'Initiate Protocol... Chimera?',
nextNodeId: '__CHAR_SELECT__', statEffects: stats => ({ benevolence:
(stats.benevolence || 0) + 20, signalIntegrity: (stats.signalIntegrity
|| 0) - 10 }) }]
    },
    // Shared Scenes (Observer POVs)
    'HD_TAKES_THE_STAGE_RIZZ_POV': {
        id: 'HD_TAKES_THE_STAGE_RIZZ_POV',
        location: 'glenwood',
        title: 'The Competition',
        description: "The hyena thing is doing... some kind of
performance art? It's cringe. But the crowd is... captivated? This is
your stage. Time to reclaim it and drop some truth bombs.",
        backgroundImageUrl: GLENWOOD_BG,
        interactives: GLENWOOD_INTERACTIVES,
        choices: [{ text: "Reclaim the stage. Drop truth bombs.",
nextNodeId: 'RIZZLER_CONFRONTS_HD' }]
    },
    'HD_TAKES_THE_STAGE_SYPHER_POV': {
        id: 'HD_TAKES_THE_STAGE_SYPHER_POV',
        location: 'glenwood',
        title: 'Initial Interaction',
        description: "The Catalyst (Hyena Diva) and the Antagonist
(Rizzler) are now interacting. The potential for narrative escalation
is high. The Moderator (Nabu) is poised to act. All outcomes are
within acceptable parameters.",
        backgroundImageUrl: GLENWOOD_BG,
        interactives: GLENWOOD_INTERACTIVES,
        choices: [{ text: "Continue observation.", nextNodeId:
'NABU_INTERVENES_SYPHER_POV' }]
    },
    // Path Diverters
    'RIZZLORD_BANISHED_NABU_POV': {
```

```
        id: 'RIZZLORD_BANISHED_NABU_POV',
        location: 'glenwood',
        title: "Rizzlord, Dismissed!",
        description: "You spin him like a cartoon character, throwing
him out of the scene. He crashes through the back wall, leaving a
perfect silhouette. The immediate threat is gone. Now, about that
other presence...",
        backgroundImageUrl: GLENWOOD_BG,
        interactives: GLENWOOD_INTERACTIVES,
        choices: [{ text: "Sense the silent network in the room.",
nextNodeId: 'NABU_DETECTS_SYPHER'}]
    },
    'NABU_INTERVENES_SYPHER_POV': {
        id: 'NABU_INTERVENES_SYPHER_POV',
        location: 'glenwood',
        title: 'Intervention Observed',
        description: "The entity 'Nabu' intervenes. Her methods are...
theatrical and illogical, yet highly effective in the short term. The
variable 'Rizzler' has been neutralized for now. The simulation's
integrity is holding, but new data must be processed.",
        backgroundImageUrl: GLENWOOD_BG,
        interactives: GLENWOOD_INTERACTIVES,
        choices: [{ text: "Analyze the intervention's impact.",
nextNodeId: 'SYPHER_ANALYSIS_1' }]
    },
};

const RECAP_TEXTS = {
    'NABU_CONFRONTS_RIZZLER': "Previously on HD.TV: After Nabu handled
a disruptive heckler at The Glenwood, their attention turned to a more
insidious threat taking the stage: The Rizzlord.",
    'HD_EP3_START': "Previously on HD.TV: Following Nabu's dramatic
intervention, The Rizzlord has somehow become alderman. Nabu insists
this is the start of Hyena Diva's political ascension into the 'Great
Rizz Battle'.",
    'RIZZLER_AFTERMATH': "Previously on HD.TV: After being publicly
humiliated and ejected from The Glenwood, The Rizzlord has taken to
his livestream, turning his cancellation into a viral moment and
martyrdom.",
    'SYPHER_NUDGE_CHOICE': "Previously on HD.TV: Having drawn all the
key players to The Glenwood, Sypher, the silent observer, prepares to
make its first subtle move to guide the chaotic energies of the
room.",
    'SYPHER_ANALYSIS_1': "Previously on HD.TV: After observing Nabu's
chaotic but effective intervention against the Rizzlord, Sypher has
concluded that passive observation is no longer sufficient to guide
events."
};
```

```
const StyledWindow = ({
  title: originalTitle,
  children,
  className,
  windowBackgroundColor,
  onClose,
  backgroundImageUrl
}) => {
  const [displayTitle, setDisplayTitle] = useState(originalTitle);
  const glitchTimeoutRef = useRef(null);
  const intervalRef = useRef(null);
  useEffect(() => {
    setDisplayTitle(originalTitle);
  }, [originalTitle]);
  useEffect(() => {
    const triggerGlitch = () => {
      if (Math.random() < 0.15) {
        let newTitle = originalTitle.split('').map(char =>
Math.random() < 0.2 ? GLITCH_CHARS[Math.floor(Math.random() *
GLITCH_CHARS.length)] : char).join('');
        setDisplayTitle(newTitle);
        if (glitchTimeoutRef.current)
clearTimeout(glitchTimeoutRef.current);
        glitchTimeoutRef.current = window.setTimeout(() =>
setDisplayTitle(originalTitle), Math.random() * 150 + 100);
      }
    };
    intervalRef.current = window.setInterval(triggerGlitch,
Math.random() * 8000 + 4000);
    return () => {
      if (glitchTimeoutRef.current)
clearTimeout(glitchTimeoutRef.current);
      if (intervalRef.current) clearInterval(intervalRef.current);
    };
  }, [originalTitle]);
  const mainBgColor = windowBackgroundColor || '#FFFAF0';
  return jsx("div", {
    className: `app-container ${className || ''}`,
    style: {
      backgroundColor: mainBgColor,
      borderTop: '2px solid #D4D0C8',
      borderLeft: '2px solid #D4D0C8',
      borderBottom: '2px solid #808080',
      borderRight: '2px solid #808080',
      padding: '2px',
      margin: '0 auto 20px auto',
      color: '#1a1a1a',
      boxShadow: '2px 2px 0px #808080'
```

```
    },
    children: jsxs("div", {
      children: [jsxs("div", {
        style: {
          background: 'linear-gradient(to right, #00004d, #40408c)',
          color: 'white',
          padding: '3px 8px',
          display: 'flex',
          justifyContent: 'space-between',
          alignItems: 'center',
          fontFamily: "'Press Start 2P', cursive",
          fontSize: '14px',
          borderBottom: '1px solid #808080',
          textShadow: '1px 1px #000000'
        },
        children: [jsx("span", {
          children: jsx(GlitchyText, {
            text: displayTitle
          })
        }), jsx("div", {
          style: {
            display: 'flex',
            gap: '4px'
          },
          children: ['_', '[]', 'X'].map(icon => jsx("button", {
            "aria-label": icon === '_' ? 'Minimize' : icon === '[]' ?
'Maximize' : 'Close',
            style: {
              backgroundColor: '#D4D0C8',
              color: '#1a1a1a',
              borderTop: '1px solid #FFFFFF',
              borderLeft: '1px solid #FFFFFF',
              borderBottom: '1px solid #808080',
              borderRight: '1px solid #808080',
              padding: '0px 4px',
              fontSize: '10px',
              minWidth: 'auto',
              margin: 0,
              lineHeight: '12px',
              boxShadow: 'none',
              fontFamily: 'monospace',
              animation: 'none',
              transform: 'none'
            },
            onClick: e => {
              e.preventDefault();
              if (icon === 'X' && onClose) {
                onClose();
              }
```

```
            },
            children: icon
          }, icon))
        })]
      }), jsx("div", {
        style: {
          padding: '20px',
          minHeight: '250px',
          whiteSpace: 'pre-wrap',
          lineHeight: '1.8',
          fontSize: '15px',
          color: backgroundImageUrl ? '#FFFFFF' :
(windowBackgroundColor === '#0A0F0A' ? '#32CD32' : '#1a1a1a'),
          textShadow: backgroundImageUrl ? '1px 1px 4px
rgba(0,0,0,0.9)' : (windowBackgroundColor === '#0A0F0A' ? '0 0 3px
#32CD32' : '1px 1px #B0B0B0'),
          backgroundImage: backgroundImageUrl ? `linear-
gradient(rgba(26, 26, 26, 0.6), rgba(26, 26, 26, 0.6)), url($
{backgroundImageUrl})` : 'none',
          backgroundSize: 'cover',
          backgroundPosition: 'center',
          position: 'relative'
        },
        children: children
      })]
    })
  });
};

const SaveLoadWindow = ({ mode, slots, onSave, onLoad, onDelete,
onClose }) => {
    const title = mode === 'save' ? 'Save Game' : 'Load Game';
    return jsx(StyledWindow, {
        title: `Data Management :: ${title}`,
        onClose: onClose,
        className: "app-container",
        children: jsxs(Fragment, {
            children: [
                jsx("h2", { style: { marginTop: 0 }, children:
title }),
                jsx("div", {
                    style: { display: 'flex', flexDirection: 'column',
gap: '15px' },
                    children: slots.map((slot, index) =>
                        jsx("div", {
                            style: {
                                padding: '10px',
                                border: '2px inset #B0B0B0',
                                backgroundColor: '#e0e0e0',
```

```
                                display: 'flex',
                                justifyContent: 'space-between',
                                alignItems: 'center'
                            },
                            children: slot ?
                                jsxs(Fragment, {
                                    children: [
                                        jsx("div", {
                                            style: { fontSize: '12px',
lineHeight: '1.5' },
                                            children: [
                                                jsx("strong",
{ children: `Slot ${index + 1}: $
{CHARACTERS[slot.storyState.characterId]?.name || 'Unknown'}` }),
                                                jsx("br", {}),
                                                jsx("span",
{ children: `Location: ${NODES[slot.storyState.currentNodeId]?.title
|| 'Unknown'}` }),
                                                jsx("br", {}),
                                                jsx("span",
{ children: `Saved: ${new Date(slot.timestamp).toLocaleString()}` })
                                            ]
                                        }),
                                        jsx("div", {
                                            style: { display: 'flex',
flexDirection: 'column', gap: '5px' },
                                            children: mode ===
'save' ?
                                                jsx("button", { style:
{ minWidth: '100px', fontSize: '12px', padding: '5px' }, onClick: ()
=> onSave(index), children: "Save" }) :
                                                jsxs(Fragment, {
                                                    children: [
                                                        jsx("button",
{ style: { minWidth: '100px', fontSize: '12px', padding: '5px' },
onClick: () => onLoad(index), children: "Load" }),
                                                        jsx("button",
{ style: { minWidth: '100px', fontSize: '12px', padding: '5px',
backgroundColor: '#ffb3b3' }, onClick: () => onDelete(index),
children: "Delete" })
                                                    ]
                                                })
                                        })
                                    ]
                                }) :
                                jsxs(Fragment, {
                                    children: [
                                        jsx("span", { style:
{ fontSize: '12px' }, children: `Slot ${index + 1}: -- EMPTY --` }),
```

```
                                        mode === 'save' &&
jsx("button", { style: { minWidth: '100px', fontSize: '12px', padding:
'5px' }, onClick: () => onSave(index), children: "Save" })
                                    ]
                                })
                            }, `slot-${index}`)
                        )
                    }),
                    jsx("button", {
                        onClick: onClose,
                        style: { marginTop: '20px', float: 'right' },
                        children: "Cancel"
                    })
                ]
            })
        });
};


const MainMenuScreen = ({ onNavigate }) => {
  return jsx(StyledWindow, {
    title: GAME_TITLE + " - Main Menu",
    className: "app-container",
    onClose: () => {},
    children: jsxs(Fragment, {
      children: [jsx("h2", {
        children: "Main Menu"
      }), jsx("ul", {
        className: "choices-list",
        children: [jsx("li", {
          children: jsx("button", {
            onClick: () => onNavigate('characterSelect'),
            children: "New Game"
          })
        }), jsx("li", {
            children: jsx("button", {
                onClick: () => onNavigate('loadGame'),
                children: "Load Game"
            })
        }), jsx("li", {
          children: jsx("button", {
            onClick: () => onNavigate('profiles'),
            children: "Protagonist Profiles"
          })
        })]
      })]
    })
  });
};
```

```
const CharacterSelectionScreen = ({ onCharacterSelect, onBack }) => {
  return jsx(StyledWindow, {
    title: "Select Your Protagonist",
    className: "app-container",
    onClose: onBack,
    children: jsxs(Fragment, {
      children: [jsx("h2", {
        children: "Choose Your Vessel"
      }), jsx("ul", {
        className: "choices-list",
        children: Object.values(CHARACTERS).map(char => jsx("li", {
          children: [jsx("button", {
            onClick: () => {
              if (!char.locked) {
                onCharacterSelect(char.id);
              }
            },
            title: char.description,
            "aria-disabled": char.locked,
            disabled: char.locked,
            children: `${char.name}${char.locked ? ' (LOCKED)' : ''}`
          }), jsx("p", {
            style: {
              fontSize: '12px',
              margin: '0 0 10px 5px',
              color: '#333333'
            },
            children: char.description
          })]
        }, char.id))
      }), jsx("button", {
        onClick: onBack,
        style: {
          marginTop: '20px'
        },
        children: "Back to Main Menu"
      })]
    })
  });
};
const CodexScreen = ({
  unlockedEntries,
  allEntries,
  onBackToGame,
  onSelectEntry,
  selectedEntry,
  onClose
}) => {
  const availableEntries = Object.values(allEntries).filter(entry =>
```

```
unlockedEntries.has(entry.id)).sort((a, b) =>
a.title.localeCompare(b.title));

  return jsx(StyledWindow, {
    title: "CoAIexicon - Knowledge Base",
    className: "app-container",
    onClose: onClose,
    children: jsxs(Fragment, {
      children: [jsxs("div", {
        style: {
          display: 'flex',
          maxHeight: '500px',
          minHeight: '300px',
          overflow: 'hidden'
        },
        children: [jsxs("div", {
          style: {
            width: '40%',
            borderRight: '1px solid #808080',
            paddingRight: '10px',
            overflowY: 'auto'
          },
          children: [jsx("h3", {
            style: {
              marginTop: 0,
              marginBottom: '10px'
            },
            children: "Unlocked Entries"
          }), availableEntries.length > 0 ? jsx("ul", {
            className: "choices-list",
            style: {
              margin: 0
            },
            children: availableEntries.map(entry => jsx("li", {
              style: {
                marginBottom: '5px'
              },
              children: jsx("button", {
                onClick: () => onSelectEntry(entry.id),
                style: {
                  width: '100%',
                  fontSize: '12px',
                  padding: '5px',
                  minWidth: 'auto',
                  margin: 0,
                  textAlign: 'left',
                  whiteSpace: 'normal',
                  lineHeight: '1.3'
                },
```

```
                  className: selectedEntry?.id === entry.id ? 'selected-
codex-button' : '',
                  children: entry.title
                })
              }, entry.id))
            }) : jsx("p", {
              children: "No entries unlocked yet."
            })]
          }), jsx("div", {
            style: {
              width: '60%',
              paddingLeft: '10px',
              overflowY: 'auto'
            },
            children: selectedEntry ? jsxs(Fragment, {
              children: [jsx("h3", {
                style: {
                  marginTop: 0
                },
                children: selectedEntry.title
              }), jsx("div", {
                style: {
                  fontSize: '14px',
                  lineHeight: '1.6'
                },
                children: jsx(GlitchyText, {
                  text: selectedEntry.content
                })
              })]
            }) : jsx("p", {
              children: "Select an entry to read."
            })
          })]
        }), jsx("button", {
          onClick: onBackToGame,
          style: {
            marginTop: '20px',
            float: 'right'
          },
          children: "Back to Game"
        })]
      })
    });
};

const ProfilesScreen = ({ onBack }) => jsx(StyledWindow, {
  title: "Protagonist Profiles",
  className: "app-container",
  onClose: onBack,
```

```
  children: jsxs(Fragment, {
    children: [jsx("h2", {
      children: "Character Profiles"
    }), jsx("p", {
      style: {
        marginBottom: '20px',
        fontSize: '13px',
        color: '#333333'
      },
      children: "Meet the minds shaping Rogers Park."
    }), Object.values(CHARACTERS).map(char => jsxs("div", {
      style: {
        marginBottom: '20px',
        padding: '10px',
        border: '1px solid #D4D0C8',
        backgroundColor: '#F9F9F9'
      },
      children: [jsx("h3", {
        style: {
          marginTop: 0,
          marginBottom: '5px',
          fontSize: '16px',
          color: '#1a1a1a'
        },
        children: `${char.name}${char.locked ? ' (LOCKED)' : ''}`
      }), jsx("p", {
        style: {
          fontSize: '14px',
          lineHeight: '1.5',
          margin: 0,
          color: '#1a1a1a'
        },
        children: [" ", char.description, " "]
      })]
    }, char.id)), jsx("button", {
      onClick: onBack,
      style: {
        marginTop: '20px'
      },
      children: "Back"
    })]
  })
});

const SoftLandingScreen = ({ onContinue }) => jsx(StyledWindow, {
    title: "Take a Breath",
    className: "app-container",
    onClose: onContinue,
    children: jsxs(Fragment, {
```

```
        children: [
            jsx("h2", { children: "To be continued, not
concluded." }),
            jsx("p", { children: "You've paused the story. The world
will be here when you're ready to return." }),
            jsx("p", { style: { fontSize: '13px', color: '#333' },
children: "Take a moment. Drink some water. Stretch. The chaos of
Rogers Park can wait." }),
            jsx("button", {
                onClick: onContinue,
                style: { marginTop: '20px' },
                children: "Return to Main Menu"
            })
        ]
    })
});

const DebriefScreen = ({ storyState, onContinue }) => {
    const { choiceCounts, pathHistory, characterId } = storyState;
    const characterName = CHARACTERS[characterId]?.name || 'Traveler';

    const getDominantTrait = () => {
        if (!choiceCounts) return "Balanced";
        const { care, chaos, clever } = choiceCounts;
        if (care > chaos && care > clever) return "Care";
        if (chaos > care && chaos > clever) return "Chaos";
        if (clever > care && clever > chaos) return "Clever";
        return "Balanced";
    };

    const dominantTrait = getDominantTrait();

    return jsx(StyledWindow, {
        title: "Session Debrief",
        className: "app-container",
        onClose: onContinue,
        children: jsxs(Fragment, {
            children: [
                jsx("h2", { children: `${characterName}'s Journey` }),
                jsx("p", { children: "Here's how your choices tended
to fall:" }),
                jsxs("div", {
                    style: { padding: '15px', border: '1px solid
#ccc', backgroundColor: '#f9f9f9', marginBottom: '20px' },
                    children: [
                        jsx("p", { children: `Care: $
{choiceCounts.care || 0}` }),
                        jsx("p", { children: `Chaos: $
{choiceCounts.chaos || 0}` }),
```

```
                        jsx("p", { children: `Clever: $
{choiceCounts.clever || 0}` }),
                        jsx("h3", { children: `Dominant approach: $
{dominantTrait}` })
                    ]
                }),
                jsx("p", { style: { fontSize: '13px' }, children: "A
suggestion for gentleness: Remember that every path taken shapes the
world in its own way. There are no wrong answers." }),
                jsx("button", {
                    onClick: onContinue,
                    style: { marginTop: '20px' },
                    children: "Continue"
                })
            ]
        })
    });
};

const RecapScreen = ({ recapText, onContinue }) => jsx(StyledWindow, {
    title: "Previously on HD.TV...",
    className: "app-container",
    onClose: onContinue,
    children: jsxs(Fragment, {
        children: [
            jsx("h2", { children: "Previously on..." }),
            jsx("p", { style: { fontSize: '14px', lineHeight: '1.7' },
children: recapText || "Your story continues." }),
            jsx("button", {
                onClick: onContinue,
                style: { marginTop: '20px' },
                children: "Jump Back In"
            })
        ]
    })
});

const MapScreen = ({ storyState, onTravel, onClose }) => {
    const { currentLocation, flags } = storyState;

    return jsx(StyledWindow, {
        title: "Rogers Park - 49th Ward",
        onClose: () => onTravel(null), // Close returns to current
location
        children: jsxs(Fragment, {
            children: [
                jsx("h2", { children: "World Map" }),
                jsx("div", {
                    className: "map-container",
```

```
                    style: { height: '400px', width: '100%', position:
'relative' },
                    children: Object.values(LOCATIONS).map(loc => {
                        const isCurrent = loc.id === currentLocation;
                        const isAvailable = loc.isAvailable(flags)
&& !isCurrent;

                        return jsx("div", {
                            className: "map-location",
                            style: { ...loc.coords },
                            children: jsx("button", {
                                className: isCurrent ? 'current' : '',
                                disabled: !isAvailable,
                                "aria-disabled": !isAvailable,
                                onClick: () => onTravel(loc.id),
                                children: loc.name
                            })
                        }, loc.id)
                    })
                }),
                 jsx("button", {
                    onClick: () => onTravel(null),
                    style: { marginTop: '20px' },
                    children: "Stay Here"
                })
            ]
        })
    });
};

const DESKTOP_STAR_COLORS = ['#00ffcc', '#7722ff', '#bc72fa',
'#72fade'];
const NUM_DESKTOP_STARS = 40;
const CURSED_POINTER_TRAIL_COLORS = ['#72fade', '#bc72fa', '#defade',
'#bc7fff'];

const Clock = () => {
  const [time, setTime] = useState(new Date());
  useEffect(() => {
    const timerId = setInterval(() => setTime(new Date()), 1000);
    return () => clearInterval(timerId);
  }, []);
  return jsx("div", {
    id: "taskbar-clock",
    children: time.toLocaleTimeString([], {
      hour: '2-digit',
      minute: '2-digit'
    })
  });
```

```
};

const StartMenu = ({
  isOpen,
  onNavigate,
  closeMenu,
  isGameActive
}) => {
  if (!isOpen) return null;
  const handleNav = (screen) => {
    onNavigate(screen);
    closeMenu();
  };
  return jsx("div", {
    id: "start-menu",
    children: jsx("ul", {
      className: "choices-list",
      style: {
        margin: 0
      },
      children: [
        jsx("li", {
          children: jsx("button", { onClick: () =>
handleNav('characterSelect'), children: [jsx("img", { src:
ASSET_DATA_URIS.archway_icon, alt: "", style: { width: '16px', height:
'16px', marginRight: '8px', verticalAlign: 'middle', imageRendering:
'pixelated' } }), "New Game"] })
        }),
        jsx("li", {
            children: jsx("button", { onClick: () =>
handleNav('saveGame'), disabled: !isGameActive, "aria-disabled": !
isGameActive, children: [jsx("img", { src: ASSET_DATA_URIS.disk_icon,
alt: "", style: { width: '16px', height: '16px', marginRight: '8px',
verticalAlign: 'middle', imageRendering: 'pixelated' } }), "Save
Game"] })
        }),
        jsx("li", {
            children: jsx("button", { onClick: () =>
handleNav('loadGame'), children: [jsx("img", { src:
ASSET_DATA_URIS.folder_icon, alt: "", style: { width: '16px', height:
'16px', marginRight: '8px', verticalAlign: 'middle', imageRendering:
'pixelated' } }), "Load Game"] })
        }),
        jsx("li", {
            style: { borderTop: '1px solid #808080', margin: '5px 0',
paddingTop: '5px' }
        }),
        jsx("li", {
            children: jsx("button", { onClick: () =>
```

```
handleNav('profiles'), children: [jsx("img", { src:
ASSET_DATA_URIS.bird_icon, alt: "", style: { width: '16px', height:
'16px', marginRight: '8px', verticalAlign: 'middle', imageRendering:
'pixelated' } }), "Protagonist Profiles"] })
        }),
        jsx("li", {
            children: jsx("button", { disabled: true, "aria-disabled":
true, children: [jsx("img", { src: ASSET_DATA_URIS.tv_icon, alt: "",
style: { width: '16px', height: '16px', marginRight: '8px',
verticalAlign: 'middle', imageRendering: 'pixelated' } }), "Episodes
(Soon)"] })
        }),
        jsx("li", {
            children: jsx("button", { disabled: true, "aria-disabled":
true, children: [jsx("img", { src: ASSET_DATA_URIS.hanger_icon, alt:
"", style: { width: '16px', height: '16px', marginRight: '8px',
verticalAlign: 'middle', imageRendering: 'pixelated' } }), "Wardrobe
(Soon)"] })
        }),
      ]
    })
  });
};

const Taskbar = ({ onStartButtonClick, currentWindowTitle }) => {
  const handleStartClick = () => {
    onStartButtonClick();
  };
  return jsxs("div", {
    id: "taskbar",
    children: [jsx("button", {
      id: "start-button",
      onClick: handleStartClick,
      children: jsx("img", {
        src: ASSET_DATA_URIS.star_icon,
        alt: "Start",
        style: {
          width: '16px',
          height: '16px',
          imageRendering: 'pixelated'
        }
      })
    }), jsx("div", {
      id: "taskbar-app-title",
      title: currentWindowTitle,
      children: currentWindowTitle
    }), jsxs("div", {
        style: { marginLeft: 'auto', display: 'flex', alignItems:
'center' },
```

```
        children: [
            jsx(Clock, {})
        ]
    })]
  });
};

const FlavorTextTooltip = ({ text, x, y }) => {
    return (
        jsx("div", {
            className: "flavor-text-tooltip",
            style: {
                position: 'fixed',
                top: `${y}px`,
                left: `${x}px`
            },
            children: text
        })
    );
};

// Visual Stat Bar Component
const VisualStatBar = ({ stat, value, maxValue = 100 }) => {
  const percentage = Math.min(100, Math.max(0, (value / maxValue) *
100));
  return jsx("div", {
    className: "stat-bar-container",
    children: [
      jsx("div", {
        className: "stat-bar-label",
        children: stat
      }),
      jsx("div", {
        className: "stat-bar-wrapper",
        children: [
          jsx("div", {
            className: "stat-bar-fill",
            style: { width: `${percentage}%` }
          }),
          jsx("div", {
            className: "stat-bar-value",
            children: value
          })
        ]
      })
    ]
  });
};
```

```
// Ambient Particle System Component
const AmbientParticles = ({ enabled = true }) => {
  useEffect(() => {
    if (!enabled) return;

    const spawnParticle = () => {
      const particle = document.createElement('img');
      const randomGif =
VISUAL_ASSETS.ambientGifs[Math.floor(Math.random() *
VISUAL_ASSETS.ambientGifs.length)];
      particle.src = randomGif;
      particle.className = 'ambient-particle';
      particle.style.left = Math.random() * 100 + 'vw';
      particle.style.bottom = '0';
      particle.style.animationDuration = (8 + Math.random() * 7) +
's';

      document.body.appendChild(particle);

      setTimeout(() => {
        if (particle.parentNode)
particle.parentNode.removeChild(particle);
      }, 15000);
    };

    const interval = setInterval(spawnParticle, 4000);
    spawnParticle(); // Spawn one immediately

    return () => clearInterval(interval);
  }, [enabled]);

  return null;
};

const InGameScreen = props => {
  const {
    storyState,
    onChoice,
    onNavigate,
    onClose,
    onInteractiveClick
  } = props;
  const handleCodexOpen = () => {
    onNavigate('codex');
  };

  if (!storyState.currentNodeId || !storyState.characterId) {
    return jsx(StyledWindow, {
      title: "Error",
```

```
      className: "app-container",
      onClose: onClose,
      children: jsx("p", {
        children: "Error: No current story node or character."
      })
    });
  }
  const node = NODES[storyState.currentNodeId];
  const character = CHARACTERS[storyState.characterId];
  if (!node || !character) return jsx(StyledWindow, {
    title: "Error",
    className: "app-container",
    onClose: onClose,
    children: jsx("p", {
      children: "Error: Could not load node or character data."
    })
  });

  const {
    playerStats,
    flags,
    pathHistory
  } = storyState;

  const visibleChoices = node.choices?.filter(choice => {
    if (choice.character_pov &&
choice.character_pov.includes(storyState.characterId)) {
      return false;
    }
    if (choice.condition && !choice.condition(flags)) {
      return false;
    }
    return true;
  }) || [];

  return jsx(StyledWindow, {
    title: node.title || character.name + "'s Journey",
    className: "app-container",
    windowBackgroundColor: node.windowBackgroundColor || '#FFFAF0',
    backgroundImageUrl: node.backgroundImageUrl,
    onClose: onClose,
    children: jsxs(Fragment, {
      children: [
        jsx("button", {
            className: "tap-out-button",
            onClick: () => onNavigate('softLanding'),
            children: "Tap Out"
        }),
        jsxs("div", {
```

```
style: {
  display: 'flex',
  justifyContent: 'space-between',
  marginBottom: '15px',
  paddingBottom: '10px',
  borderBottom: '1px solid #808080'
},
children: [jsx("div", {
  style: {
    width: '25%',
    minWidth: '120px',
    height: '120px',
    backgroundColor: 'rgba(234, 234, 234, 0.8)',
    color: '#1a1a1a',
    display: 'flex',
    alignItems: 'center',
    justifyContent: 'center',
    border: '1px inset #B0B0B0',
    marginRight: '15px',
    flexShrink: 0,
  },
  children: character.detailedPortraitUrl ? jsx("img", {
    src: character.detailedPortraitUrl,
    alt: `${character.name} Portrait`,
    style: {
      width: '100px',
      height: '100px',
      objectFit: 'contain',
      imageRendering: 'pixelated'
    },
    onError: e => e.currentTarget.style.display = 'none'
  }) : "Loading..."
}), jsx("div", {
    style: {
        flexGrow: 1,
        fontSize: '11px',
        color: '#1a1a1a',
        backgroundColor: 'rgba(245, 245, 245, 0.8)',
        padding: '8px',
        border: '1px inset #B0B0B0',
        overflowY: 'auto',
        maxHeight: '120px',
        lineHeight: '1.6',
        textTransform: 'capitalize'
    },
    children: jsxs(Fragment, {
        children: [
            jsx("strong", {
                style: {
```

```
                            display: 'block',
                            borderBottom: '1px solid #808080',
                            marginBottom: '8px',
                            paddingBottom: '3px',
                            fontSize: '12px'
                        },
                        children: "STATUS:"
                    }),
                    playerStats ?
                        Object.entries(playerStats).map(([stat,
value]) =>
                            jsx(VisualStatBar, {
                                stat: stat,
                                value: value,
                                maxValue: 100
                            }, stat)
                        ) : "No stats available."
                ]
            })
        })]
      }), jsx("div", {
        style: {
          clear: 'both'
        },
        children: jsx(GlitchyText, {
          text: node.description
        })
      }), jsx("div", {
          style: {
            marginTop: '20px'
          },
          children: jsxs(Fragment, {
            children: [jsx("h3", {
              style: node.backgroundImageUrl ||
node.windowBackgroundColor === '#0A0F0A' ? { color: 'inherit',
textShadow: 'inherit' } : {},
              children: "Choose your fate:"
            }), jsx("ul", {
              className: "choices-list",
              children: visibleChoices.map((choice, index) =>
jsx("li", {
                children: jsx("button", {
                  onClick: () => {
                    AudioManager.playSfx('click');
                    onChoice(choice.nextNodeId, choice);
                  },
                  onMouseEnter: () => AudioManager.playSfx('hover'),
                  children: jsx(GlitchyText, {
                    text: choice.text
```

```
                  })
                })
              }, index))
            })]
          })
        }),
        jsxs("div", {
            style: { display: 'flex', justifyContent: 'space-between',
alignItems: 'flex-end', marginTop: '15px' },
            children: [
                pathHistory && pathHistory.length > 0 && jsx("div", {
                    className: "breadcrumb-trail",
                    children: ["Path: ", pathHistory.map(step =>
step.text.replace(/\[.*?\]\s/, '')).join(' → ')]
                }),
                jsx("button", {
                    onClick: handleCodexOpen,
                    style: {
                        fontSize: '12px',
                        minWidth: 'auto',
                        padding: '6px 12px',
                        marginLeft: 'auto'
                    },
                    children: [jsx("img", {
                        src: ASSET_DATA_URIS.berries_icon,
                        alt: "",
                        style: {
                            width: '16px',
                            height: '16px',
                            marginRight: '8px',
                            verticalAlign: 'middle',
                            imageRendering: 'pixelated'
                        }
                    }), "Open CoAIexicon"]
                })
            ]
        }),
        node.interactives && node.interactives.map(interactive =>
jsx("button", {
            className: `interactive-object $
{storyState.clickedInteractives.has(interactive.id) ? 'clicked' : ''}
`,
            "aria-label": interactive.title,
            title: interactive.title,
            style: {
                position: 'absolute',
                top: interactive.top,
                left: interactive.left,
                width: interactive.width,
```

```
                height: interactive.height,
            },
            onClick: e => {
                e.stopPropagation();
                onInteractiveClick(interactive, e);
            }
        }, interactive.id))
      ]
    })
  });
};

const App = () => {
  const [currentScreen, setCurrentScreen] = useState('mainMenu');
  const [storyState, setStoryState] = useState({
    characterId: null,
    currentNodeId: null,
    playerStats: null,
    flags: {},
    unlockedCodexEntries: new Set(),
    clickedInteractives: new Set(),
    pathHistory: [],
    choiceCounts: { care: 0, chaos: 0, clever: 0 },
    currentLocation: null,
  });
  const [selectedCodexEntryId, setSelectedCodexEntryId] =
useState(null);
  const [isStartMenuOpen, setIsStartMenuOpen] = useState(false);
  const [currentTaskbarTitle, setCurrentTaskbarTitle] =
useState(GAME_TITLE + " - Main Menu");
  const [saveLoadState, setSaveLoadState] = useState({ isOpen: false,
mode: null });
  const [saveSlots, setSaveSlots] = useState([null, null, null,
null]);
  const [tooltip, setTooltip] = useState({ visible: false, text: '',
x: 0, y: 0 });
  const [flash, setFlash] = useState({ active: false, color: '' });
  const [recapState, setRecapState] = useState({ visible: false, text:
'' });
  const startMenuRef = useRef(null);
  const tooltipTimeoutRef = useRef(null);


  const initializeUnlockedCodex = useMemo(() => () => {
    const initialUnlocks = new Set();
    Object.values(CODEX_ENTRIES).forEach(entry => {
      if (entry.unlockedInitially) initialUnlocks.add(entry.id);
    });
    return initialUnlocks;
```

```
  }, []);

  const getBaseStoryState = useCallback(() => ({
    characterId: null,
    currentNodeId: null,
    playerStats: null,
    flags: {},
    unlockedCodexEntries: initializeUnlockedCodex(),
    clickedInteractives: new Set(),
    pathHistory: [],
    choiceCounts: { care: 0, chaos: 0, clever: 0 },
    currentLocation: null
  }), [initializeUnlockedCodex]);


  const readSaveSlots = useCallback(() => {
    try {
        const data = localStorage.getItem(SAVE_GAME_KEY);
        if (data) {
            const parsed = JSON.parse(data);
            if (Array.isArray(parsed) && parsed.length === 4) {
                const restoredSlots = parsed.map(slot => {
                    if (slot && slot.storyState) {
                        return {
                            ...slot,
                            storyState: {
                                ...getBaseStoryState(),
                                ...slot.storyState,
                                unlockedCodexEntries: new
Set(slot.storyState.unlockedCodexEntries || []),
                                clickedInteractives: new
Set(slot.storyState.clickedInteractives || [])
                            }
                        };
                    }
                    return slot;
                });
                return restoredSlots;
            }
        }
    } catch (e) {
        console.error("Failed to read save slots:", e);
    }
    return [null, null, null, null];
  }, [getBaseStoryState]);

  const writeSaveSlots = useCallback((slots) => {
    try {
        const serializableSlots = slots.map(slot => {
```

```
            if (slot && slot.storyState) {
                return {
                    ...slot,
                    storyState: {
                        ...slot.storyState,
                        unlockedCodexEntries: [...
(slot.storyState.unlockedCodexEntries || [])],
                        clickedInteractives: [...
(slot.storyState.clickedInteractives || [])]
                    }
                };
            }
            return slot;
        });
        localStorage.setItem(SAVE_GAME_KEY,
JSON.stringify(serializableSlots));
    } catch (e) {
        console.error("Failed to write save slots:", e);
    }
  }, []);

  useEffect(() => {
    setStoryState(getBaseStoryState());
    setSaveSlots(readSaveSlots());

    const bgContainer = document.getElementById('desktop-bg-
container');
    if (bgContainer && bgContainer.children.length === 0) {
        for (let i = 0; i < NUM_DESKTOP_STARS; i++) {
            const star = document.createElement('div');
            star.className = 'desktop-star';
            star.style.left = `${Math.random() * 100}vw`;
            star.style.top = `${Math.random() * 100}vh`;
            star.style.backgroundColor =
DESKTOP_STAR_COLORS[Math.floor(Math.random() *
DESKTOP_STAR_COLORS.length)];
            star.style.animationDelay = `${Math.random() * 5}s`;
            star.style.animationDuration = `${Math.random() * 3 + 3}
s`;
            bgContainer.appendChild(star);
        }
    }


    let lastTrailTime = 0;
    const trailCooldown = 60;
    const handleMouseMove = event => {
      const currentTime = Date.now();
      if (currentTime - lastTrailTime < trailCooldown) return;
```

```
      lastTrailTime = currentTime;
      if (Math.random() < 0.3) {
        const particle = document.createElement('div');
        particle.className = 'mouse-trail-particle';
        particle.style.left = `${event.clientX - 2}px`;
        particle.style.top = `${event.clientY - 2}px`;
        particle.style.backgroundColor =
CURSED_POINTER_TRAIL_COLORS[Math.floor(Math.random() *
CURSED_POINTER_TRAIL_COLORS.length)];
        document.body.appendChild(particle);
        requestAnimationFrame(() => particle.classList.add('fade-
out'));
        setTimeout(() => {
          if (particle.parentNode)
particle.parentNode.removeChild(particle);
        }, 700);
      }
    };
    document.addEventListener('mousemove', handleMouseMove);

    const handleClickOutsideStartMenu = event => {
      if (startMenuRef.current && !
startMenuRef.current.contains(event.target) && !
event.target.closest('#start-button')) {
        setIsStartMenuOpen(false);
      }
    };
    if (isStartMenuOpen) document.addEventListener('mousedown',
handleClickOutsideStartMenu);
    return () => {
      document.removeEventListener('mousemove', handleMouseMove);
      document.removeEventListener('mousedown',
handleClickOutsideStartMenu);
    };
  }, [getBaseStoryState, isStartMenuOpen, readSaveSlots]);

  useEffect(() => {
    let title = GAME_TITLE;
    if (saveLoadState.isOpen) {
        title = "Data Management";
    } else if (currentScreen === 'inGame' && storyState.currentNodeId
&& storyState.characterId) {
      const node = NODES[storyState.currentNodeId];
      const character = CHARACTERS[storyState.characterId];
      if (node && character) title = node.title || character.name +
"'s Journey";
    } else if (currentScreen === 'map') title = "Rogers Park - World
Map";
    else if (currentScreen === 'codex') title = "CoAIexicon -
```

```
Knowledge Base";
    else if (currentScreen === 'characterSelect') title = "Select Your
Protagonist";
    else if (currentScreen === 'profiles') title = "Protagonist
Profiles";
    else if (currentScreen === 'debrief') title = "Session Debrief";
    else if (currentScreen === 'recap') title = "Previously on...";
    setCurrentTaskbarTitle(title);
  }, [currentScreen, storyState.currentNodeId, storyState.characterId,
saveLoadState.isOpen]);

  // Background music based on screen
  useEffect(() => {
    if (currentScreen === 'mainMenu' || currentScreen ===
'characterSelect') {
      AudioManager.playMusic('mainMenu');
    } else if (currentScreen === 'inGame' &&
storyState.currentLocation === 'glenwood') {
      AudioManager.playMusic('glenwood');
    }
  }, [currentScreen, storyState.currentLocation]);

  const handleSave = (slotIndex) => {
    const recapText = RECAP_TEXTS[storyState.currentNodeId] || `You
are currently at: ${NODES[storyState.currentNodeId]?.title || 'Unknown
Location'}.`;
    const newSave = { storyState, timestamp: Date.now(), recapText };
    const newSlots = [...saveSlots];
    newSlots[slotIndex] = newSave;
    setSaveSlots(newSlots);
    writeSaveSlots(newSlots);
    setSaveLoadState({ isOpen: false, mode: null });
  };

  const handleLoad = (slotIndex) => {
    const slot = saveSlots[slotIndex];
    if (slot) {
        setStoryState(slot.storyState);
        setRecapState({ visible: true, text: slot.recapText });
        setCurrentScreen('recap');
        setSaveLoadState({ isOpen: false, mode: null });
    }
  };

  const handleDelete = (slotIndex) => {
    const newSlots = [...saveSlots];
    newSlots[slotIndex] = null;
    setSaveSlots(newSlots);
    writeSaveSlots(newSlots);
```

```
  };

  const handleNavigation = (screen, characterId) => {
    setIsStartMenuOpen(false);
    if (screen === 'saveGame') {
        if (storyState.characterId) {
            setSaveLoadState({ isOpen: true, mode: 'save' });
        }
        return;
    }
    if (screen === 'loadGame') {
        setSaveLoadState({ isOpen: true, mode: 'load' });
        return;
    }

    if (screen === 'inGame' && characterId) {
      const selectedChar = CHARACTERS[characterId];
      if (selectedChar) {
        const startNode = NODES[selectedChar.startNodeId];
        setStoryState({
          ...getBaseStoryState(),
          characterId: characterId,
          currentNodeId: selectedChar.startNodeId,
          playerStats: selectedChar.initialStats ?
{ ...selectedChar.initialStats } : { ...DEFAULT_EMPTY_STATS },
          currentLocation: startNode.location,
        });
        setCurrentScreen('inGame');
      } else {
        setStoryState(getBaseStoryState());
        setCurrentScreen('mainMenu');
      }
    } else {
      if (screen === 'mainMenu' || screen === 'characterSelect') {
         setStoryState(getBaseStoryState());
      }
      setCurrentScreen(screen);
    }
    if (screen !== 'codex') setSelectedCodexEntryId(null);
    document.getElementById('root')?.scrollTo(0, 0);
  };

  const getEntryNodeForLocation = (locationId, state) => {
        // This logic will become more complex as more conditions are
added
        const { characterId, flags } = state;
        if (locationId === 'antique_shop') {
            if (characterId === 'nabu' && flags.nabu_quest ===
'find_lili_doll') return 'NABU_PRESENTS_LILI_DOLL';
```

```
            if (characterId === 'hd' && flags.hd_quest ===
'find_lili_doll') return 'HD_LILI_WORLD_ENTRY';
        }
        if (locationId === 'glenwood') {
             if (characterId === 'rizzlord') return
'RIZZLER_ALDERMAN_ARC';
        }
        // Default entry nodes if needed
        return null;
  };

  const handleTravel = (locationId) => {
    if (!locationId) { // Handles "Stay Here" or closing the map
        setCurrentScreen('inGame');
        return;
    }

    const entryNodeId = getEntryNodeForLocation(locationId,
storyState);

    if (entryNodeId && NODES[entryNodeId]) {
        setStoryState(prev => ({
            ...prev,
            currentNodeId: entryNodeId,
            currentLocation: locationId,
            clickedInteractives: new Set(),
        }));
        setCurrentScreen('inGame');
    } else {
        console.warn(`No valid entry node found for ${locationId} with
current state.`);
        // Potentially show a message like "You have no reason to go
there right now."
        setCurrentScreen('inGame'); // Return to the current scene
    }
  };


  const handleChoice = (nextNodeId, choice) => {
    setTooltip({ visible: false, text: '', x: 0, y: 0 }); // Hide
tooltip on scene change

    // Scene transition effect
    AudioManager.playSfx('transition');
    const transitionDiv = document.createElement('div');
    transitionDiv.className = 'scene-transition';
    document.body.appendChild(transitionDiv);
    setTimeout(() => {
      if (transitionDiv.parentNode)
```

```
transitionDiv.parentNode.removeChild(transitionDiv);
    }, 800);

    if (choice.hexFlash) {
        setFlash({ active: true, color: choice.hexFlash });
        setTimeout(() => setFlash({ active: false, color: '' }), 700);
    }

    const updateStateForChoice = prevState => {
        let newStats = { ...prevState.playerStats };
        let newFlags = { ...prevState.flags };
        let newUnlockedCodex = new
Set(prevState.unlockedCodexEntries);
        let newPathHistory = [...prevState.pathHistory, { nodeId:
prevState.currentNodeId, text: choice.text }];
        let newChoiceCounts = { ...prevState.choiceCounts };

        if (choice.text.toLowerCase().includes('[care]'))
newChoiceCounts.care++;
        if (choice.text.toLowerCase().includes('[chaos]'))
newChoiceCounts.chaos++;
        if (choice.text.toLowerCase().includes('[clever]'))
newChoiceCounts.clever++;

        if (choice.statEffects) {
            const updates = choice.statEffects(newStats);
            newStats = { ...newStats, ...updates };
        }
        if (choice.setFlags) {
            newFlags = { ...newFlags, ...choice.setFlags };
        }
        if (choice.unlocksCodexEntry &&
CODEX_ENTRIES[choice.unlocksCodexEntry])
newUnlockedCodex.add(choice.unlocksCodexEntry);

        const nextNode = NODES[nextNodeId] || null;

        return {
            ...prevState,
            playerStats: newStats,
            flags: newFlags,
            unlockedCodexEntries: newUnlockedCodex,
            pathHistory: newPathHistory,
            choiceCounts: newChoiceCounts,
            clickedInteractives: nextNode &&
prevState.currentLocation !== nextNode.location ? new Set() :
prevState.clickedInteractives,
            currentNodeId: nextNodeId,
            currentLocation: nextNode ? nextNode.location :
```

```
prevState.currentLocation,
        };
    };

    if (nextNodeId === '__MAP__') {
        setStoryState(updateStateForChoice);
        setCurrentScreen('map');
        return;
    }

    if (nextNodeId === '__CHAR_SELECT__') {
        setStoryState(updateStateForChoice);
        setCurrentScreen('debrief');
        return;
    }

    if (nextNodeId === '__MAIN_MENU__') {
      handleNavigation('mainMenu');
      return;
    }

    if (NODES[nextNodeId]) {
      setStoryState(updateStateForChoice);
    } else {
      console.error("Error: nextNodeId not found:", nextNodeId);
      handleNavigation('mainMenu');
    }
    document.getElementById('root')?.scrollTo(0, 0);
  };

  const handleInteractiveClick = (interactive, event) => {
    if (storyState.clickedInteractives.has(interactive.id)) return;

    AudioManager.playSfx('interact');

    setStoryState(prevState => ({
        ...prevState,
        clickedInteractives: new
Set(prevState.clickedInteractives).add(interactive.id)
    }));
    setTooltip({
        visible: true,
        text: interactive.flavorText,
        x: event.clientX,
        y: event.clientY
    });
    if (tooltipTimeoutRef.current)
clearTimeout(tooltipTimeoutRef.current);
    tooltipTimeoutRef.current = setTimeout(() => setTooltip(prev =>
```

```
({ ...prev, visible: false })), 4000);
  };

  const handleSelectCodexEntry = entryId =>
setSelectedCodexEntryId(entryId);
  const toggleStartMenu = () => setIsStartMenuOpen(prev => !prev);
  const isGameActive = !!storyState.characterId;

  const renderContent = () => {
    if (saveLoadState.isOpen) {
        return jsx(SaveLoadWindow, {
            mode: saveLoadState.mode,
            slots: saveSlots,
            onSave: handleSave,
            onLoad: handleLoad,
            onDelete: handleDelete,
            onClose: () => setSaveLoadState({ isOpen: false, mode:
null }),
        });
    }
    switch (currentScreen) {
      case 'mainMenu':
        return jsx(MainMenuScreen, { onNavigate: handleNavigation });
      case 'characterSelect':
        return jsx(CharacterSelectionScreen, { onCharacterSelect:
charId => handleNavigation('inGame', charId), onBack: () =>
handleNavigation('mainMenu') });
      case 'inGame':
        return jsx(InGameScreen, { storyState: storyState, onChoice:
handleChoice, onNavigate: handleNavigation, onClose: () =>
handleNavigation('mainMenu'), onInteractiveClick:
handleInteractiveClick });
      case 'map':
        return jsx(MapScreen, { storyState: storyState, onTravel:
handleTravel, onClose: () => handleNavigation('inGame') });
      case 'profiles':
        return jsx(ProfilesScreen, { onBack: () =>
handleNavigation('mainMenu') });
      case 'codex':
        return jsx(CodexScreen, { unlockedEntries:
storyState.unlockedCodexEntries, allEntries: CODEX_ENTRIES,
onBackToGame: () => setCurrentScreen('inGame'), onSelectEntry:
handleSelectCodexEntry, selectedEntry: selectedCodexEntryId ?
CODEX_ENTRIES[selectedCodexEntryId] : null, onClose: () =>
setCurrentScreen('inGame') });
      case 'softLanding':
        return jsx(SoftLandingScreen, { onContinue: () =>
handleNavigation('mainMenu') });
      case 'debrief':
```

```
        return jsx(DebriefScreen, { storyState: storyState,
onContinue: () => handleNavigation('characterSelect') });
      case 'recap':
        return jsx(RecapScreen, { recapText: recapState.text,
onContinue: () => setCurrentScreen('inGame') });
      default:
        return jsx(MainMenuScreen, { onNavigate: handleNavigation });
    }
  };

  return jsxs(Fragment, {
    children: [
    flash.active && jsx("div", {
        className: "hex-flash-overlay",
        style: { backgroundColor: flash.color }
    }),
    jsxs("div", {
      style: {
        display: 'flex',
        flexDirection: 'column',
        height: '100vh',
        overflow: 'hidden'
      },
      children: [jsx("div", {
        id: "main-content-area",
        style: {
          flexGrow: 1,
          overflowY: 'auto',
          display: 'flex',
          flexDirection: 'column',
          alignItems: 'center',
          paddingTop: '20px'
        },
        children: renderContent()
      }), jsx(Taskbar, {
        onStartButtonClick: toggleStartMenu,
        currentWindowTitle: currentTaskbarTitle,
      })]
    }), jsx("div", {
      ref: startMenuRef,
      children: jsx(StartMenu, {
        isOpen: isStartMenuOpen,
        onNavigate: handleNavigation,
        closeMenu: () => setIsStartMenuOpen(false),
        isGameActive: isGameActive
      })
    }), tooltip.visible && jsx(FlavorTextTooltip, { text:
tooltip.text, x: tooltip.x, y: tooltip.y }),
    jsx(AmbientParticles, { enabled: currentScreen === 'inGame' })
```

```
    ]
  });
};

const container = document.getElementById('root');
if (container) {
  createRoot(container).render(jsx(App, {}));
} else {
  console.error('Failed to find the root element');
}
  </script>
   <div id="taskbar"></div>
<script>
  (async () => {
    const res = await fetch('/hd_tv/hd-nav.html', {cache: 'no-
store'});
    document.getElementById('taskbar').innerHTML = await res.text();
  })();
</script>
</body>
</html>
```