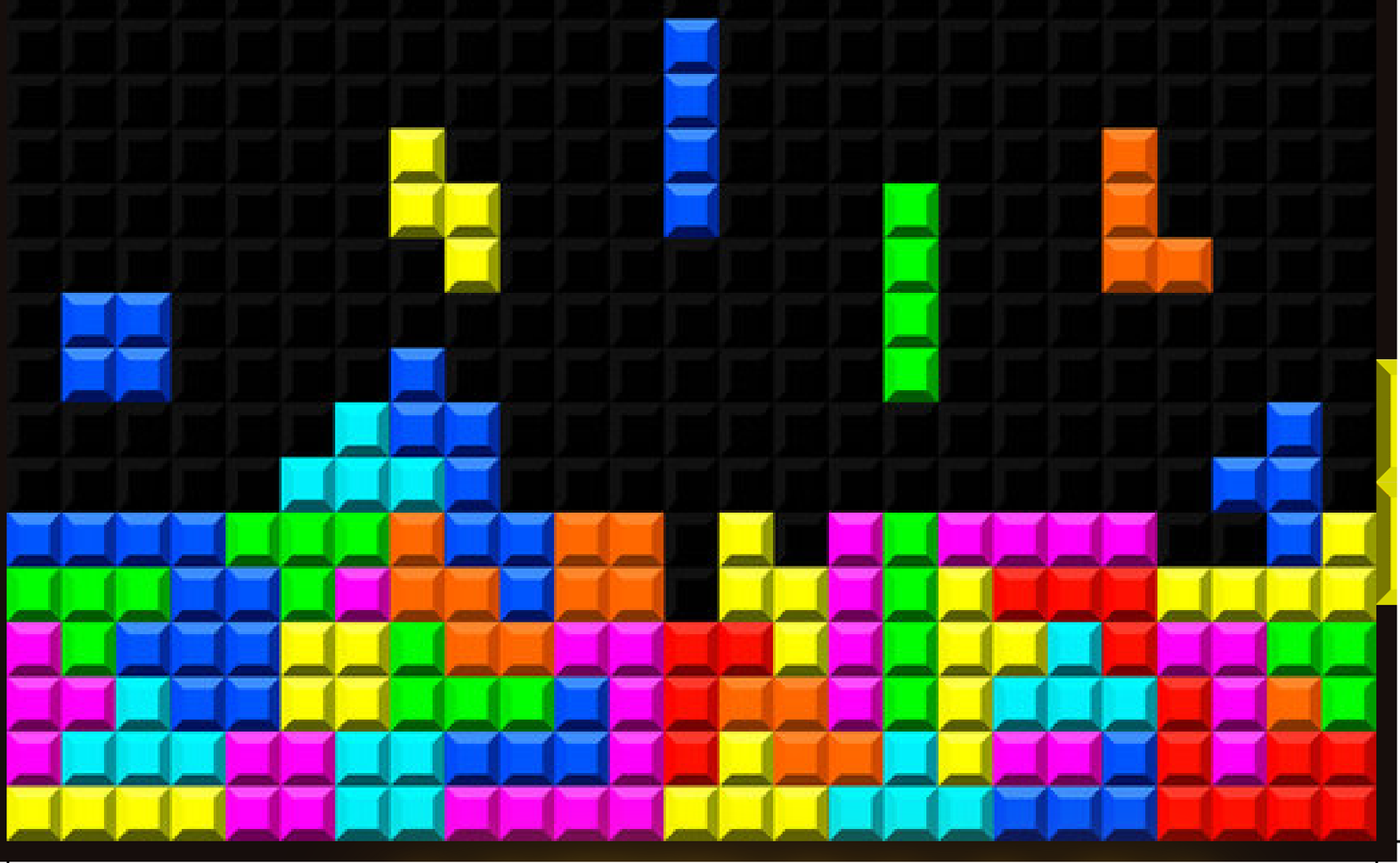
18/03/2025

TEWELY

UN PUZZLE DE CIBERSEGURIDAD

ALUMNA: HERNANDEZ MARTINEZ HELLEN ALINE
PROFE: MORENO VILLALBA LEONARDO MIGUEL
MATERIA: DESARROLLO DE APLICACIONES WEB



Introducción.

1.Introducción: ¿Por qué Tewerly?.....pag3

2.objetivo del proyecto: pag3

3.Justificación de la elección del juego.....pag3

4.Especificaciones generales del juego (género, mecánicas, reglas principales). .pag3

5.Bocetos o wireframes de la interfaz gráfica.....pag5

6.Lista de recursos necesarios (imágenes, sonidos, fuentes, etc.). pag9

7.Explicación de la estructura de carpetas y archivos del proyecto. pag11

8.Desarrollo del Juego pag12

9.Pruebas y depuración pag31

10.Conclusiones y Mejoras Futura pag37

1.INTRODUCCIÓN: ¿POR QUÉ TEWERLY?

Tewerly: Un Viaje Interactivo hacia la Ciberseguridad

En un mundo cada vez más digitalizado, la ciberseguridad ha pasado de ser un tema de especialistas a una necesidad cotidiana. Tewerly nace como respuesta a esta realidad, combinando la mecánica adictiva de los juegos de puzzle con un propósito educativo fundamental: familiarizar a personas de todas las edades con los conceptos básicos de la ciberseguridad de manera entretenida y accesible.

A diferencia de los métodos tradicionales de aprendizaje sobre ciberseguridad, que suelen ser técnicos y poco atractivos para el público general, Tewerly transforma este conocimiento vital en una experiencia lúdica donde los jugadores absorben conceptos fundamentales mientras se divierten.

2.OBJETIVO DEL PROYECTO:

Tewerly tiene varios objetivos importantes. Primero, quiere enseñar sobre ciberseguridad, como hacer contraseñas seguras y detectar correos de phishing, de una manera sencilla para todos. También busca que las personas entiendan por qué es tan importante proteger su información en línea y los riesgos que esto conlleva.

El juego hace que aprender sobre estos temas complicados sea divertido, ayudando a que la información se quede en la memoria. Además, fomenta habilidades como el pensamiento lógico y la toma de decisiones rápidas.

A través de una historia, Tewerly conecta estos conceptos con situaciones reales, lo que hace que aprender sea más significativo. No es solo un juego; es una herramienta educativa que se siente como diversión. Con niveles y cuestionarios, los jugadores no solo se divierten, sino que también refuerzan lo que han aprendido.

3.JUSTIFICACIÓN DE LA ELECCIÓN DEL JUEGO

Elegí este juego porque tengo buenos recuerdos de los videojuegos en una tele LG, y uno en particular era mi favorito: "Perfect Puzzle: Magic Jewelry". Aunque era parecido a Tetris, su diseño y mecánica eran diferentes y más atractivos. Esa experiencia me inspiró a crear Tewerly, un juego que mezcla la diversión de los rompecabezas con la educación en ciberseguridad.

El nombre "Tewerly" combina "jewelry" (joyería) y "tower" (torre), simbolizando cómo organizamos nuestras defensas digitales. La temática de la joyería me parece divertida e interesante para cualquier persona, sin importar su experiencia previa.

Tewerly busca hacer que aprender sobre ciberseguridad sea entretenido y fácil. A través del juego, los jugadores aprenderán conceptos importantes como crear contraseñas seguras y reconocer correos de phishing, mientras desarrollan habilidades como el pensamiento crítico y la planificación. Además, la historia del juego conectará estos conceptos con situaciones cotidianas, mostrando la importancia de cuidar nuestra información en un mundo digital.

4.ESPECIFICACIONES GENERALES DEL JUEGO (GÉNERO, MECÁNICAS, REGLAS PRINCIPALES).

Género

- Puzzle/Arcade: Basado en la mecánica clásica de Tetris
- Educativo: Integra conocimientos de ciberseguridad
- Narrativo: Incluye modo historia con una trama ambientada en el futuro

Temática

- Ciberseguridad y defensa digital
- Ambientado en 2035, en un mundo hiperconectado
- El jugador asume el rol de un agente de la Agencia de Ciberseguridad Global (ACG)

Mecánicas principales

Gameplay básico (estilo Tetris)

- Piezas: 5 tipos diferentes de piezas geométricas que caen desde la parte superior
- Movimiento: El jugador puede mover las piezas horizontalmente y rotarlas
- Objetivo principal: Completar líneas horizontales para eliminarlas del tablero
- Caída de piezas: Velocidad incremental según nivel y tiempo restante
- Pieza fantasma: Muestra dónde caerá la pieza actual para facilitar la planificación

Progresión

- 5 niveles con dificultad progresiva (velocidad de caída más rápida)
- Sistema de tiempo: 3 minutos (180 segundos) por nivel
- Sistema de puntuación: Basado en líneas completadas y multiplicador por nivel

Modo Historia

- 5 episodios narrativos que introducen conceptos de ciberseguridad
- Imágenes de fondo temáticas para cada episodio
- Narrativa que avanza con la progresión del jugador

Sistema Educativo

- Cuestionarios de ciberseguridad entre niveles
- 3 preguntas por cuestionario con opciones múltiples
- Se requieren 2 de 3 respuestas correctas para avanzar

Reglas principales

Reglas de juego

1. Las piezas caen automáticamente desde la parte superior del tablero
2. El jugador puede mover las piezas lateralmente, rotarlas y acelerar su caída
3. Al completar una línea horizontal, esta se elimina y aumenta la puntuación
4. El juego termina si las piezas alcanzan la parte superior del tablero
5. Para completar un nivel, el jugador debe sobrevivir durante 3 minutos

Reglas de progresión

1. Después de cada nivel, el jugador debe responder un cuestionario de ciberseguridad
2. Se necesitan al menos 2 respuestas correctas para avanzar al siguiente nivel
3. La dificultad aumenta en cada nivel:
    - Nivel 1: Velocidad base
    - Nivel 2: 15% más rápido que el nivel 1
    - Nivel 3: 30% más rápido que el nivel 1
    - Nivel 4: 45% más rápido que el nivel 1
    - Nivel 5: 60% más rápido que el nivel 1

Sistema de guardado

- Guarda automáticamente el progreso del jugador (nivel actual)
- Almacena el nombre de usuario
- Registra las 10 mejores puntuaciones en una tabla de clasificación

Características adicionales

- Efectos de sonido para movimientos, rotaciones, eliminación de líneas y eventos del juego
- Soporte para modo claro/oscuro (adaptable a las preferencias del sistema)
- Diseño responsivo para diferentes tamaños de pantalla
- Interfaz retro inspiradora con efectos visuales de pantalla CRT

Este juego combina hábilmente la mecánica adictiva de Tetris con aprendizaje educativo sobre ciberseguridad, todo envuelto en una narrativa futurista que motiva al jugador a seguir avanzando mientras refuerza conceptos importantes de seguridad digital.

5.BOCETOS O WIREFRAMES DE LA INTERFAZ GRÁFICA.

Versión 1.0: Fundamentos del Juego

La primera versión de Tewerly estableció los cimientos del juego y su mecánica principal inspirada en los puzzles de bloques tradicionales. Esta versión incluía:

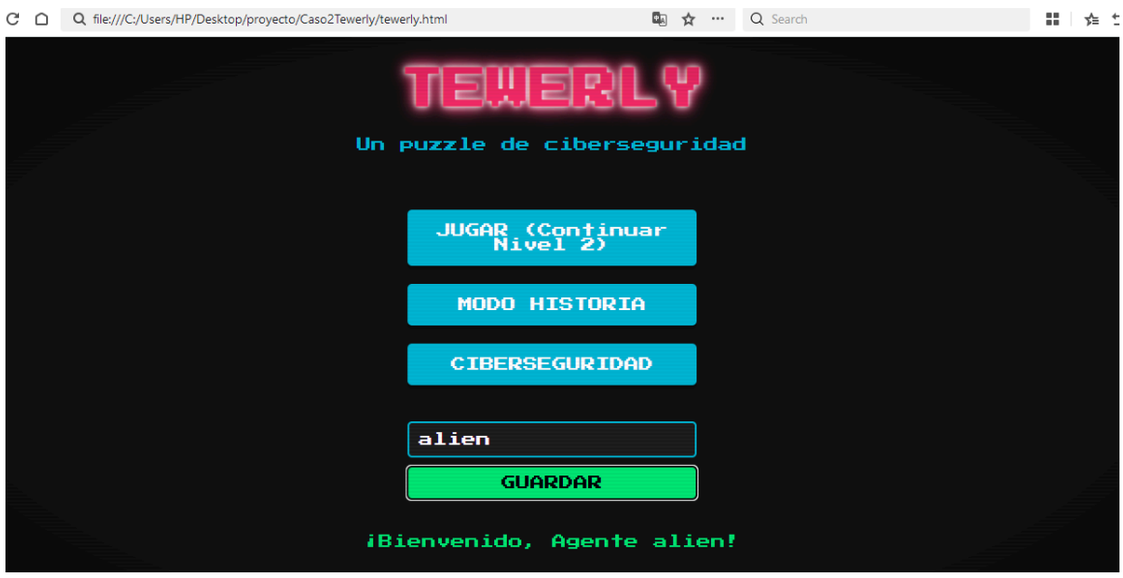
- Mecánica básica de juego: Sistema de caída de piezas y eliminación de líneas completas
- 3 niveles iniciales: Cada uno con su propio conjunto de desafíos
- Sistema de puntuación: Basado en la cantidad de líneas eliminadas y nivel actual
- Cuestionarios de ciberseguridad: Al finalizar cada nivel para reforzar el aprendizaje
- Interfaz retro: Diseño con efectos de escáner y CRT para una estética digital nostálgica
- Sección informativa: Con conceptos básicos sobre ciberseguridad

Esta versión sentó las bases de la experiencia de juego, combinando entretenimiento con educación en ciberseguridad de manera accesible.

Versión 2.0: Narrativa Expandida

La segunda versión expandió significativamente el juego, añadiendo profundidad narrativa y más contenido:

- Modo Historia: Narrativa inmersiva sobre una agencia de ciberseguridad enfrentando amenazas digitales
- 5 niveles completos: Ampliación a 5 episodios con dificultad progresiva
- Personalización del jugador: Integración del nombre del jugador en la narrativa como "Agente"
- Nuevas preguntas de ciberseguridad: Ampliación del contenido educativo con temas avanzados
- Dificultad progresiva: Ajuste automático de la velocidad y complejidad según el nivel
- Mensajes narrativos entre niveles: Transiciones que enriquecen la historia mientras se avanza

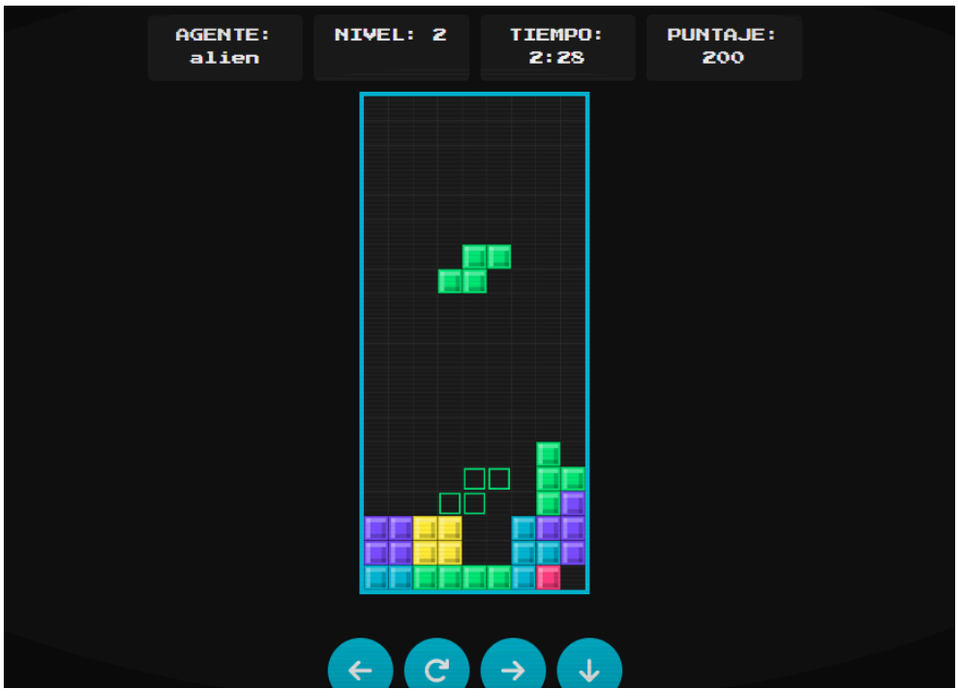


Esta actualización transformó Tewerly de un simple juego de puzzle a una experiencia narrativa completa con un arco argumental sobre amenazas cibernéticas modernas.

Versión 3.0: Experiencia Inmersiva Mejorada

La tercera versión se centró en mejorar la experiencia sensorial y añadir elementos competitivos:

- Tabla de récords: Sistema para guardar y mostrar las mejores puntuaciones
- Efectos visuales mejorados: Animaciones para eventos importantes del juego
- Pantalla de juego estática: Mejora de la jugabilidad evitando desplazamientos indeseados
- Fondos temáticos: Imágenes de fondo relacionadas con cada episodio de la historia
- Optimización para móviles: Mejor experiencia en dispositivos táctiles
- Indicador visual mejorado: "Pieza fantasma" que muestra dónde caerá la pieza actual



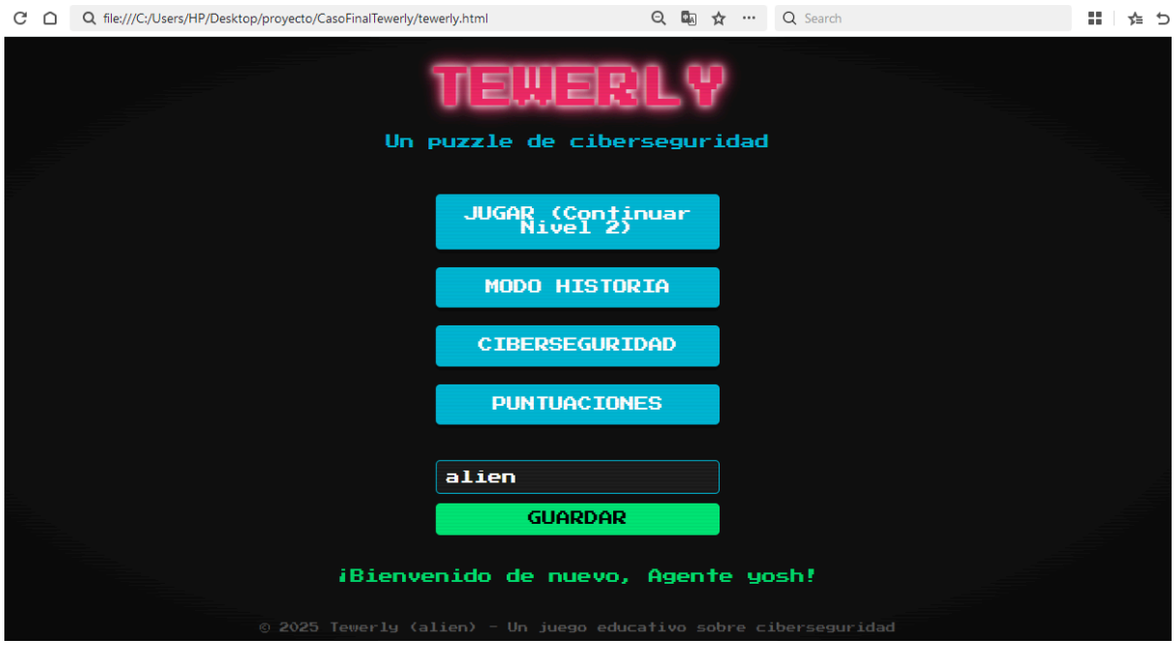
Esta versión elevó Tewerly a una experiencia más inmersiva, mientras mantenía su enfoque educativo original.

Versión 3.5: Refinamiento y Experiencia Sensorial Completa (Actual)

La versión actual (que te proporcioné) representa un refinamiento significativo de la experiencia de usuario con características clave:

- Sistema completo de efectos de sonido:
  - Sonidos distintos para movimientos, rotaciones y caídas de piezas
  - Efectos especiales para eliminación de líneas
  - Sonidos de celebración al completar niveles
  - Efecto sonoro de game over
  - Volumen optimizado a 0.3 para una experiencia equilibrada
- Imágenes de fondo para el modo historia:
  - Cada uno de los 5 episodios tiene una imagen única de Unsplash
  - Imágenes precargadas para rendimiento óptimo
  - Overlay semitransparente para mejorar legibilidad del texto
- Sistema avanzado de tabla de puntuaciones:
  - Almacenamiento local de las 10 mejores puntuaciones
  - Visualización de nombre, puntuación, nivel alcanzado y fecha
  - Ordenamiento automático por puntuación más alta
  - Diseño visual coherente con la estética del juego

- Dificultad progresiva mejorada:
  - Factor de incremento de velocidad aumentado a 0.15 entre niveles
  - Reducción del intervalo de caída de 50ms a 70ms por nivel
  - Aceleración gradual dentro de cada nivel basada en tiempo restante
- Interfaz perfeccionada:
  - Botón dedicado para acceder a la tabla de puntuaciones
  - Mejor contraste y legibilidad de texto en todas las pantallas
  - Feedback visual y auditivo para todas las acciones
  - Modo oscuro/claro automático según preferencias del sistema
- Optimización técnica:
  - Mejor gestión de memoria con precarga de recursos
  - Manejo de errores mejorado para la reproducción de audio
  - Compatibilidad con diversos navegadores y dispositivos
  - Código modular y bien estructurado para futuras expansiones



6.LISTA DE RECURSOS NECESARIOS (IMÁGENES, SONIDOS, FUENTES, ETC.).

Fuentes Tipográficas

- Press Start 2P - Fuente principal del juego
  - Origen: Google Fonts
  - URL: https://fonts.googleapis.com/css2?family=Press+Start+2P&display=swap
  - Uso: Todo el texto del juego, títulos, menús y contenido
  -

Imágenes de Fondo para Episodios

1. Episodio 1: El Despertar
   - URL: https://images.unsplash.com/photo-1550751827-4bd374c3f58b
   - Descripción: Imagen tecnológica con código/circuitos para representar el inicio de la crisis
2. Episodio 2: Siguiendo el Rastro
   - URL: https://images.unsplash.com/photo-1558494949-ef010cbdcc31
   - Descripción: Visualización de redes/conexiones para representar el rastreo digital
3. Episodio 3: La Infiltración
   - URL: https://images.unsplash.com/photo-1526374965328-7f61d4dc18c5
   - Descripción: Imagen de código binario/matriz para ilustrar la infiltración digital
4. Episodio 4: La Revelación
   - URL: https://images.unsplash.com/photo-1544197150-b99a580bb7a8
   - Descripción: Imagen relacionada con inteligencia artificial/tecnología avanzada
5. Episodio 5: El Enfrentamiento Final
   - URL: https://images.unsplash.com/photo-1537498425277-c283d32ef9db
   - Descripción: Imagen dramática de tecnología/circuitos para el clímax

Iconos

- Font Awesome 6.4.0 - Paquete de iconos
  - URL: https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min.css
  - Iconos específicos utilizados:
    - fa-arrow-left - Botones de retorno y movimiento izquierdo
    - fa-arrow-right - Movimiento derecho
    - fa-arrow-down - Movimiento hacia abajo
    - fa-redo - Botón de rotación

Efectos de Sonido

1. Sonido de Movimiento
   - URL: https://assets.mixkit.co/active_storage/sfx/212/212-preview.mp3
   - Descripción: Efecto corto que suena al mover piezas lateralmente
   - ID: move-sound
2. Sonido de Rotación
   - URL: https://assets.mixkit.co/active_storage/sfx/270/270-preview.mp3
   - Descripción: Efecto que suena al rotar piezas
   - ID: rotate-sound
3. Sonido de Caída Rápida
   - URL: https://assets.mixkit.co/active_storage/sfx/146/146-preview.mp3
   - Descripción: Efecto que suena al hacer caer rápidamente una pieza
   - ID: drop-sound

1. Sonido de Línea Completada
    - URL: https://assets.mixkit.co/active_storage/sfx/888/888-preview.mp3
    - Descripción: Efecto que suena cuando se completa y elimina una línea
    - ID: line-clear-sound
2. Sonido de Nivel Completado
    - URL: https://assets.mixkit.co/active_storage/sfx/1010/1010-preview.mp3
    - Descripción: Efecto de celebración que suena al completar un nivel
    - ID: level-complete-sound
3. Sonido de Game Over
    - URL: https://assets.mixkit.co/active_storage/sfx/220/220-preview.mp3
    - Descripción: Efecto dramático que suena cuando termina el juego
    - ID: game-over-sound

Paleta de Colores

Colores de Sistema

- Color primario: #00b8d4 - Azul cian (botones, bordes, elementos interactivos)
- Color secundario: #ff5722 - Naranja (botones secundarios, acentos)
- Color de texto: #ffffff - Blanco (texto general)
- Color de fondo: #121212 - Negro muy oscuro (fondo principal)
- Color de rejilla: #1e1e1e - Gris muy oscuro (fondo del tablero de juego)
- Color de borde: #333333 - Gris oscuro (bordes de celdas)
- Color de encabezado: #ff2c6d - Rosa (título del juego, encabezados)
- Color de éxito: #00e676 - Verde brillante (indicadores positivos)
- Color de advertencia: #ffeb3b - Amarillo (advertencias)
- Color de peligro: #ff3d00 - Rojo (errores, game over)

Colores de Piezas

1. #FF4081 - Rosa
2. #7C4DFF - Púrpura
3. #00E676 - Verde
4. #FFEB3B - Amarillo
5. #FF5722 - Naranja
6. #00B8D4 - Azul cian

Archivos JavaScript

- tewerly.js - Archivo principal de JavaScript que contiene toda la lógica del juego

Archivos CSS

- tewerly.css - Archivo principal de CSS que contiene todos los estilos del juego

Recursos de Almacenamiento Local

- tewerly_username - Nombre de usuario del jugador
- tewerly_level - Nivel actual del jugador
- tewerly_leaderboard - Datos de la tabla de puntuaciones (JSON)

Recursos Adicionales de Interfaz

- Efectos visuales CSS:
    - Efecto escáner (scanlines)
    - Efecto CRT (monitor antiguo)
    - Efectos de animación (glow, pulse, shake, typing)
    - Responsive design para diferentes dispositivos

7.EXPLICACIÓN DE LA ESTRUCTURA DE CARPETAS Y ARCHIVOS DEL PROYECTO.

Descripción de los Archivos

1. index.html
Este archivo contiene la estructura completa de la interfaz del juego, incluyendo:

- Metatags y configuración básica
- Enlaces a recursos externos (CSS, fuentes, iconos)
- Estructura de las diferentes pantallas del juego:
  - Menú principal
  - Pantalla de juego
  - Pantalla de historia
  - Pantalla de información
  - Pantalla de cuestionario
  - Pantalla de nivel completado
  - Pantalla de juego terminado
  - Tabla de puntuaciones
- Elementos precargados (imágenes y audios)
- Enlaces a scripts

2. tewerly.css
Contiene todos los estilos del juego, incluyendo:

- Importación de la fuente "Press Start 2P"
- Variables CSS para colores y tema
- Soporte para modo oscuro
- Estilos responsivos
- Animaciones y efectos visuales
- Estilos para cada pantalla del juego
- Media queries para adaptación a diferentes dispositivos

3. tewerly.js
Contiene toda la lógica del juego, incluyendo:

- Constantes y variables globales
- Inicialización del juego
- Lógica de piezas (creación, movimiento, rotación)
- Control de juego (teclado, botones)
- Renderizado del tablero y piezas
- Sistema de puntuación
- Gestión de niveles y cuestionarios
- Almacenamiento local para puntuaciones y progreso
- Gestión de pantallas y navegación entre ellas
- Sistema de audio para efectos de sonido

Recursos Externos
El juego utiliza varios recursos externos que se cargan directamente desde CDNs:

1. Fuentes:
   - Google Fonts: "Press Start 2P" (fuente con estilo pixelado/retro)

1. Iconos:
   - Font Awesome 6.4.0 para íconos de interfaz

1. Imágenes de fondo (cargadas desde Unsplash):
   - Episodio 1: photo-1550751827-4bd374c3f58b
   - Episodio 2: photo-1558494949-ef010cbdcc31
   - Episodio 3: photo-1526374965328-7f61d4dc18c5
   - Episodio 4: photo-1544197150-b99a580bb7a8
   - Episodio 5: photo-1537498425277-c283d32ef9db

1. Archivos de audio (para efectos de sonido):
   - Movimiento: assets.mixkit.co/.../212-preview.mp3
   - Rotación: assets.mixkit.co/.../270-preview.mp3
   - Caída: assets.mixkit.co/.../146-preview.mp3
   - Limpieza de línea: assets.mixkit.co/.../888-preview.mp3
   - Nivel completado: assets.mixkit.co/.../1010-preview.mp3
   - Juego terminado: assets.mixkit.co/.../220-preview.mp3

Esta estructura simple facilita la implementación y el mantenimiento del juego, ya que todos los componentes esenciales están en archivos individuales claramente separados por función (estructura, estilo y comportamiento).

8.DESARROLLO DEL JUEGO

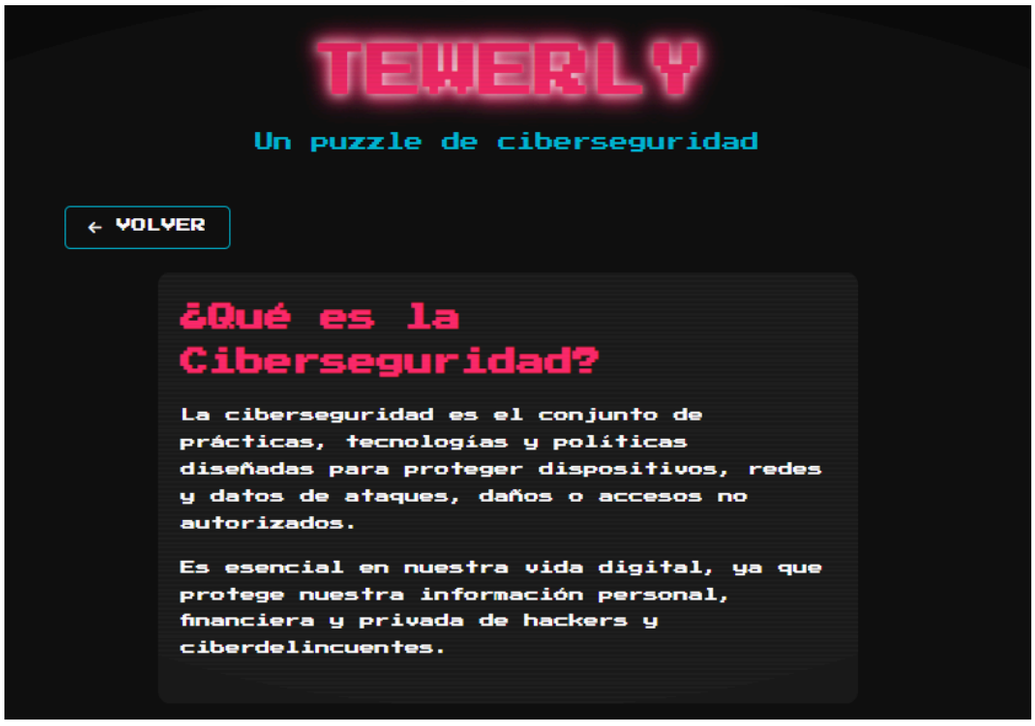
El HTML de Tewerly está organizado como una aplicación de una sola página (SPA) con múltiples "pantallas" que se muestran u ocultan según la interacción del usuario. A continuación, analizo sus componentes principales:

1. Estructura Base y Metadatos

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Tewerly - Puzzle de Ciberseguridad</title>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.4.0/css/all.min.css">
    <link rel="stylesheet" href="tewerly.css">
</head>
```

Pantalla del modo historia de Tewerly

Pantalla de datos sobre la Ciberseguridad



- Documento HTML5 con idioma español
- Metaetiquetas para codificación y configuración responsive
- Enlace a Font Awesome para iconos
- Enlace a la hoja de estilos personalizada

2. Efectos Visuales Retro

Estos divs vacíos se utilizan para crear efectos visuales retro (líneas de escaneo y efecto CRT) como capas superpuestas en toda la aplicación.

```
<div class="scanlines"></div>
<div class="crt-effect"></div>
```

3. Estructura de Pantallas Múltiples

El HTML implementa un patrón de "pantallas virtuales" dentro del contenedor principal: Cada representa una "pantalla" diferente que se muestra u oculta mediante JavaScript según el estado del juego.

```
<div class="container">
    <header class="game-header">...</header>

    <div id="main-menu" class="main-menu">...</div>
    <div id="game-screen" class="game-screen">...</div>
    <div id="info-screen" class="info-screen">...</div>
    <div id="story-screen" class="story-screen">...</div>
    <div id="story-message-screen" class="story-message-screen">...</div>
    <div id="quiz-screen" class="quiz-screen">...</div>
    <div id="level-complete" class="level-complete">...</div>
    <div id="game-over" class="game-over">...</div>
    <div id="leaderboard-screen" class="leaderboard-screen">...</div>

    <footer class="game-footer">...</footer>
</div>
```

4. Menú Principal

- Botones principales de navegación
- Formulario para ingresar nombre de usuario
- Elemento para mostrar mensaje de bienvenida

```
<div id="main-menu" class="main-menu">
    <button id="start-btn" class="menu-btn">JUGAR <span id="continue-text"></span></button>
    <button id="story-mode-btn" class="menu-btn">MODO HISTORIA</button>
    <button id="info-btn" class="menu-btn">CIBERSEGURIDAD</button>
    <button id="leaderboard-btn" class="menu-btn">PUNTUACIONES</button>
    <div class="username-form">
        <input type="text" id="username-input" class="username-input" placeholder="Ingresa tu
nombre" maxlength="15">
        <button id="save-username-btn" class="save-btn">GUARDAR</button>
    </div>
    <div id="player-welcome" class="player-welcome"></div>
</div>
```

5. Pantalla de Juego

- Panel de información del juego (nombre, nivel, tiempo, puntaje)
- Elemento <canvas> donde se renderiza el juego
- Controles táctiles para dispositivos móviles

```
<div id="game-screen" class="game-screen">
    <button id="game-back-btn" class="back-btn">
        <i class="fas fa-arrow-left"></i> VOLVER
    </button>

    <div class="game-info">
        <div class="game-stat">AGENTE: <span id="player-name-display">Anónimo</span></div>
        <div class="game-stat">NIVEL: <span id="level-display">1</span></div>
        <div class="game-stat">TIEMPO: <span id="timer-display">3:00</span></div>
        <div class="game-stat">PUNTAJE: <span id="score-display">0</span></div>
    </div>

    <canvas id="game-canvas"></canvas>

    <div class="game-controls">
        <button id="left-btn" class="control-btn"><i class="fas fa-arrow-left"></i></button>
        <button id="rotate-btn" class="control-btn"><i class="fas fa-redo"></i></button>
        <button id="right-btn" class="control-btn"><i class="fas fa-arrow-right"></i></button>
        <button id="down-btn" class="control-btn"><i class="fas fa-arrow-down"></i></button>
    </div>
</div>
```

6. Otras Pantallas Especializadas

El HTML incluye varias pantallas adicionales, cada una con propósito específico:

- Pantalla de Modo Historia: Muestra episodios y permite iniciar niveles desde un contexto narrativo
- Pantalla de Información: Proporciona datos educativos sobre ciberseguridad
- Pantalla de Mensajes de Historia: Muestra narrativa entre niveles
- Pantalla de Cuestionario: Presenta preguntas sobre ciberseguridad
- Pantalla de Nivel Completado: Felicita al jugador y muestra progreso
- Pantalla de Juego Terminado: Muestra puntaje final y opciones para reiniciar
- Pantalla de Tabla de Puntuaciones: Muestra los mejores puntajes

7. Recursos Precargados

- Imágenes precargadas para episodios de la historia (ocultas con CSS)
- Elementos de audio para efectos de sonido

```
<div style="display: none">
    <img id="bg-episode1" src="https://images.unsplash.com/photo-1550751827-4bd374c3f58b"
    alt="Episodio 1">
    <!-- Más imágenes... -->
</div>

<audio id="move-sound" src="https://assets.mixkit.co/active_storage/sfx/212/212-preview.mp3"
preload="auto"></audio>
<!-- Más audios... -->
```

Explicación del diseño aplicado con CSS

1. Configuración Base y Variables CSS

```css
@import url('https://fonts.googleapis.com/css2?family=Press+Start+2P&display=swap');

:root {
    --primary-color: #00b8d4;
    --secondary-color: #ff5722;
    --text-color: #ffffff;
    --bg-color: #121212;
    /* Más variables... */
}

.dark {
    /* Variables para modo oscuro... */
}
```

- Importación de fuente pixelada "Press Start 2P" para estética retro
- Sistema de variables CSS para tema coherente y fácil modificación
- Clase .dark para modo oscuro

2. Reseteo y Estilos Generales

- Reseteo básico para consistencia entre navegadores
- Configuración del body como un contenedor flex vertical
- Contenedor principal centrado con ancho máximo

```css
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: 'Press Start 2P', cursive;
}

body {
    background-color: var(--bg-color);
    color: var(--text-color);
    line-height: 1.6;
    overflow-x: hidden;
    min-height: 100vh;
    display: flex;
    flex-direction: column;
}

.container {
    width: 100%;
    max-width: 800px;
    margin: 0 auto;
    padding: 20px;
    flex-grow: 1;
    display: flex;
    flex-direction: column;
}
```

3. Estilización de Botones y Controles

- Botones con estilo retro/arcade
- Efectos al pasar el cursor y presionar
- Sistema de sombras para efecto 3D
- Transiciones suaves

```css
.menu-btn {
    background-color: var(--primary-color);
    color: var(--text-color);
    border: none;
    padding: 15px 30px;
    font-size: 1rem;
    cursor: pointer;
    transition: all 0.3s ease;
    width: 100%;
    max-width: 320px;
    border-radius: 5px;
    position: relative;
    text-align: center;
    box-shadow: 0 4px 0 rgba(0, 0, 0, 0.3);
}

.menu-btn:hover {
    background-color: var(--secondary-color);
    transform: translateY(-2px);
    box-shadow: 0 6px 0 rgba(0, 0, 0, 0.3);
}

.menu-btn:active {
    transform: translateY(2px);
    box-shadow: 0 2px 0 rgba(0, 0, 0, 0.3);
}
```

4. Pantalla de Juego y Canvas

- Diseño en columna con elementos centrados
- Panel de información como flex container con wrap para responsividad
- Canvas con borde y tamaño limitado para adaptarse a pantallas pequeñas

```css
.game-screen {
    display: none;
    flex-direction: column;
    align-items: center;
    max-width: 100%;
    margin: 0 auto;
}

.game-info {
    display: flex;
    justify-content: space-between;
    width: 100%;
    max-width: 600px;
    margin-bottom: 10px;
    flex-wrap: wrap;
    gap: 10px;
}

#game-canvas {
    border: 4px solid var(--primary-color);
    background-color: var(--grid-bg-color);
    margin-bottom: 20px;
    max-width: 100%;
    max-height: 70vh;
}
```

5. Efectos Visuales Retro

- Efecto de líneas de escaneo CRT mediante gradiente lineal repetido
- Efecto de viñeta/distorsión CRT mediante gradiente radial
- Elementos posicionados de forma fija cubriendo toda la pantalla
- pointer-events: none para que no interfieran con interacciones

```css
.scanlines {
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background: linear-gradient(
        to bottom,
        rgba(255, 255, 255, 0) 50%,
        rgba(0, 0, 0, 0.2) 50%
    );
    background-size: 100% 4px;
    z-index: 999;
    pointer-events: none;
    opacity: 0.15;
}

.crt-effect {
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background: radial-gradient(
        ellipse at center,
        transparent 50%,
        rgba(0, 0, 0, 0.4) 100%
    );
    z-index: 998;
    pointer-events: none;
}
```

6. Animaciones

- Animación glow para efecto neón en el título
- Animación pulse para elementos que deben llamar la atención
- Animación shake para pantalla de Game Over

```css
@keyframes glow {
    from {
        text-shadow: 0 0 5px #fff, 0 0 10px #fff, 0 0 15px var(--header-color),
        0 0 20px var(--header-color);
    }
    to {
        text-shadow: 0 0 10px #fff, 0 0 15px #ff2c6d, 0 0 20px #ff2c6d,
        0 0 25px #ff2c6d;
    }
}

@keyframes pulse {
    0% { transform: scale(1); }
    50% { transform: scale(1.05); }
    100% { transform: scale(1); }
}

@keyframes shake {
    0% { transform: translateX(0); }
    25% { transform: translateX(-10px); }
    50% { transform: translateX(10px); }
    75% { transform: translateX(-10px); }
    100% { transform: translateX(0); }
}
```

7. Diseño Responsivo

- Media queries para adaptar tamaños de fuente y elementos en pantallas pequeñas
- Reducción de padding y márgenes para optimizar espacio
- Ajuste de controles para facilitar uso táctil

```css
@media (max-width: 600px) {
    .game-title {
        font-size: 2rem;
    }

    .game-subtitle {
        font-size: 0.8rem;
    }

    .control-btn {
        width: 50px;
        height: 50px;
        font-size: 1.2rem;
    }

    /* Más ajustes... */
}
```

El CSS combina técnicas modernas (flexbox, variables CSS, transiciones) con una estética retro deliberada, creando una experiencia visual coherente que refuerza la temática de ciberseguridad con un toque nostálgico de videojuegos clásicos.

5.2. Lógica del Juego (JavaScript)

1. Análisis de las funciones principales

Inicialización y configuración

- init(): Función principal que inicializa el juego
- resizeCanvas(): Ajusta el tamaño del canvas según el tamaño de pantalla
- setupEventListeners(): Establece los controladores de eventos
- initSounds(): Inicializa y configura los elementos de audio
- playSound(): Reproduce efectos de sonido

Control del juego

- startGame(): Inicia una nueva partida
- startStoryMode(): Inicia el modo historia
- pauseGame(): Pausa el juego
- resumeGame(): Reanuda el juego
- startGameLoop(): Inicia el bucle principal del juego
- startTimer(): Inicia el temporizador de juego
- updateTimer(): Actualiza la visualización del temporizador

Manejo de piezas

- createNewPiece(): Crea una nueva pieza
- movePiece(): Mueve la pieza horizontalmente
- rotatePiece(): Rota la pieza
- rotateShape(): Matriz auxiliar para la rotación de piezas
- dropPiece(): Deja caer la pieza inmediatamente
- movePieceDown(): Mueve la pieza hacia abajo un paso
- canMoveTo(): Comprueba si una pieza puede moverse a una posición
- placePiece(): Coloca la pieza en el tablero
- checkLines(): Comprueba y elimina líneas completas

Renderizado

- drawGame(): Dibuja el juego
- drawGrid(): Dibuja la cuadrícula
- drawBoard(): Dibuja el tablero
- drawPiece(): Dibuja la pieza actual
- drawGhostPiece(): Dibuja la sombra de la pieza
- drawCell(): Dibuja una celda individual

Gestión de niveles y puntuación

- levelComplete(): Maneja la finalización de un nivel
- gameOver(): Maneja el final del juego
- showQuizScreen(): Muestra la pantalla de cuestionario
- generateQuiz(): Genera preguntas de cuestionario
- submitQuiz(): Procesa las respuestas del cuestionario
- startNextLevel(): Inicia el siguiente nivel
- restartGame(): Reinicia el juego desde el nivel 1

Gestión de interfaz y navegación

- checkSavedGame(): Comprueba si hay un juego guardado
- saveUsername(): Guarda el nombre de usuario
- saveScoreToLeaderboard(): Guarda la puntuación en el ranking
- showLeaderboard(): Muestra la tabla de puntuaciones
- showMainMenu(): Muestra el menú principal
- showStoryScreen(): Muestra la pantalla de historia
- showInfoScreen(): Muestra la pantalla de información
- hideAllScreens(): Oculta todas las pantallas

Manejo de eventos

- handleKeyDown(): Maneja los eventos de teclado

2. Interacción del usuario

- Controles por teclado (flechas) para mover y rotar piezas
- Botones en pantalla para control táctil
- Sistema de navegación entre diferentes pantallas
- Formulario para guardar nombre de usuario
- Quiz interactivo entre niveles

3. Mecánicas del juego

- Sistema de colisión para piezas
- Cálculo de puntuación basado en líneas completadas
- Sistema de niveles con dificultad progresiva
- Temporizador de cuenta regresiva
- Efecto "fantasma" para mostrar dónde caerá la pieza

4. Uso de localStorage

- Guardar nombre de usuario
- Guardar nivel actual para continuar partida
- Guardar tabla de puntuaciones

1. Estructura General y Configuración Inicial

El código JavaScript de Tewerly está organizado como un juego de tipo "Tetris" con elementos educativos sobre ciberseguridad. Comienza con la configuración del modo oscuro y definición de constantes fundamentales:

```
// Check for dark mode preference
if (window.matchMedia && window.matchMedia('(prefers-color-scheme: dark)').matches) {
    document.documentElement.classList.add('dark');
}
window.matchMedia('(prefers-color-scheme: dark)').addEventListener('change', event => {
    if (event.matches) {
        document.documentElement.classList.add('dark');
    } else {
        document.documentElement.classList.remove('dark');
    }
});

// Game constants
const ROWS = 20;
const COLS = 9;
const CELL_SIZE = 30;
const COLORS = [
    '#FF4081', // Pink
    '#7C4DFF', // Purple
    '#00E676', // Green
    '#FFEB3B', // Yellow
    '#FF5722', // Orange
    '#00B8D4'  // Cyan
];
```

Funciones Principales de Inicialización

- init(): Función de arranque que configura el entorno del juego.

```
function init() {
    resizeCanvas();
    window.addEventListener('resize', resizeCanvas);
    checkSavedGame();
    setupEventListeners();
    initSounds();

    if (username) {
        usernameInput.value = username;
        playerWelcome.textContent = `¡Bienvenido de nuevo, Agente ${username}!`;
    }

    showMainMenu();
}
```

- resizeCanvas(): Adapta el tamaño del canvas al dispositivo.

```
function resizeCanvas() {
    const containerWidth = Math.min(500, window.innerWidth - 40);
    const cellSize = Math.floor(containerWidth / COLS);

    canvas.width = COLS * cellSize;
    canvas.height = ROWS * cellSize;

    if (cellSize !== CELL_SIZE) {
        ctx.scale(cellSize / CELL_SIZE, cellSize / CELL_SIZE);
    }
}
```

2. Manejo del Ciclo de Juego

Control del Bucle Principal

- startGameLoop(): Establece el bucle de animación para actualizar el juego.

```
function startGameLoop() {
    function gameLoop(timestamp) {
        if (!gameActive) return;

        const deltaTime = timestamp - lastTime;
        lastTime = timestamp;

        dropCounter += deltaTime;
        if (dropCounter > dropInterval / gameSpeedFactor) {
            movePieceDown();
            dropCounter = 0;
        }

        drawGame();

        animationFrameId = requestAnimationFrame(gameLoop);
    }

    animationFrameId = requestAnimationFrame(gameLoop);
}
```

startTimer(): Administra el temporizador de cuenta regresiva.

```
function startTimer() {
    clearInterval(timerInterval);
    timerInterval = setInterval(() => {
        gameTimer--;
        updateTimer();

        // Ajuste gradual de la velocidad
        if (gameTimer <= 150 && gameTimer > 120) {
            gameSpeedFactor = 1.1 + (currentLevel - 1) * 0.15;
        } else if (gameTimer <= 120 && gameTimer > 90) {
            gameSpeedFactor = 1.2 + (currentLevel - 1) * 0.15;
        }
        // Más ajustes...

        if (gameTimer <= 0) {
            levelComplete();
        }
    }, 1000);
}
```

3. Manipulación de Piezas

Creación y Movimiento

createNewPiece(): Genera aleatoriamente una nueva pieza.

```
function createNewPiece() {

    // Define las 5 formas posibles tipo Tetris
    const shapes = [
        [
            [1, 1],
            [1, 1]
        ], // Cuadrado
        [
            [0, 1, 0],
            [1, 1, 1]
        ], // T
        [
            [1, 1, 1, 1]
        ], // Línea
        [
            [1, 1, 0],
            [0, 1, 1]
        ], // Z
        [
            [0, 1, 1],
            [1, 1, 0]
        ] // S
    ];

    // Selecciona forma y color aleatorios
    const shape = shapes[Math.floor(Math.random() * shapes.length)];
    const color = COLORS[Math.floor(Math.random() * COLORS.length)];
    currentPiece = {
        shape: shape,
        color: color,
        row: 0,
        col: Math.floor((COLS - shape[0].length) / 2)
    };

    // Comprueba si el juego ha terminado
    if (!canMoveTo(currentPiece.row, currentPiece.col, currentPiece.shape)) {
        gameOver();
    }
}
```

rotatePiece(): Rota la pieza con "wall kicks" para evitar colisiones.

```
function rotatePiece() {
    if (!gameActive) return;

    const newShape = rotateShape(currentPiece.shape);
    if (canMoveTo(currentPiece.row, currentPiece.col, newShape)) {
        currentPiece.shape = newShape;
        playSound(rotateSound);
    } else {
        // Intenta "wall kicks" (ajustar posición si la rotación causaría colisión)
        if (canMoveTo(currentPiece.row, currentPiece.col - 1, newShape)) {
            currentPiece.col -= 1;
            currentPiece.shape = newShape;
            playSound(rotateSound);
        }
        // Más intentos de ajuste...
    }
}
```

Detección de Colisiones

canMoveTo(): Verifica si una pieza puede moverse a una posición.

```
function canMoveTo(row, col, shape) {
    for (let r = 0; r < shape.length; r++) {
        for (let c = 0; c < shape[r].length; c++) {
            if (shape[r][c]) {
                const newRow = row + r;
                const newCol = col + c;

                // Comprueba límites
                if (newRow < 0 || newRow >= ROWS || newCol < 0 || newCol >= COLS) {
                    return false;
                }

                // Comprueba si el espacio está ocupado
                if (gameBoard[newRow][newCol]) {
                    return false;
                }
            }
        }
    }
    return true;
}
```

placePiece(): Coloca la pieza en el tablero y verifica líneas completadas.

```
function placePiece() {
    for (let r = 0; r < currentPiece.shape.length; r++) {
        for (let c = 0; c < currentPiece.shape[r].length; c++) {
            if (currentPiece.shape[r][c]) {
                const boardRow = currentPiece.row + r;
                const boardCol = currentPiece.col + c;
                gameBoard[boardRow][boardCol] = currentPiece.color;
            }
        }
    }

    // Comprueba líneas completadas
    checkLines();

    // Comprueba si las piezas alcanzaron la parte superior
    checkGameOver();
}
```

4. Sistema de Renderizado

drawGame(): Función principal de renderizado.

```
function drawGame() {
    // Limpia el canvas
    ctx.clearRect(0, 0, canvas.width, canvas.height);

    // Dibuja la cuadrícula
    drawGrid();

    // Dibuja el tablero
    drawBoard();

    // Dibuja la pieza actual
    drawPiece();
}
```

drawGhostPiece(): Renderiza una pieza "fantasma" que muestra donde caerá.

```
function drawGhostPiece() {
    const ghostPiece = {
        shape: currentPiece.shape,
        color: currentPiece.color,
        row: currentPiece.row,
        col: currentPiece.col
    };

    // Encuentra la posición más baja a la que puede llegar la pieza
    while (canMoveTo(ghostPiece.row + 1, ghostPiece.col, ghostPiece.shape)) {
        ghostPiece.row++;
    }

    // Omite el dibujo si la pieza fantasma está en la misma posición que la pieza actual
    if (ghostPiece.row === currentPiece.row) return;

    // Dibuja la pieza fantasma (versión con contorno)
    for (let row = 0; row < ghostPiece.shape.length; row++) {
        for (let col = 0; col < ghostPiece.shape[row].length; col++) {
            if (ghostPiece.shape[row][col]) {
                const x = (ghostPiece.col + col) * CELL_SIZE;
                const y = (ghostPiece.row + row) * CELL_SIZE;

                ctx.strokeStyle = ghostPiece.color;
                ctx.lineWidth = 2;
                ctx.strokeRect(x + 3, y + 3, CELL_SIZE - 6, CELL_SIZE - 6);
            }
        }
    }
}
```

5. Gestión de Niveles y Progresión

checkLines(): Detecta líneas completas y actualiza la puntuación.

```
function drawGhostPiece() {
    const ghostPiece = {
        shape: currentPiece.shape,
        color: currentPiece.color,
        row: currentPiece.row,
        col: currentPiece.col
    };

    // Encuentra la posición más baja a la que puede llegar la pieza
    while (canMoveTo(ghostPiece.row + 1, ghostPiece.col, ghostPiece.shape)) {
        ghostPiece.row++;
    }

    // Omite el dibujo si la pieza fantasma está en la misma posición que la pieza actual
    if (ghostPiece.row === currentPiece.row) return;

    // Dibuja la pieza fantasma (versión con contorno)
    for (let row = 0; row < ghostPiece.shape.length; row++) {
        for (let col = 0; col < ghostPiece.shape[row].length; col++) {
            if (ghostPiece.shape[row][col]) {
                const x = (ghostPiece.col + col) * CELL_SIZE;
                const y = (ghostPiece.row + row) * CELL_SIZE;

                ctx.strokeStyle = ghostPiece.color;
                ctx.lineWidth = 2;
                ctx.strokeRect(x + 3, y + 3, CELL_SIZE - 6, CELL_SIZE - 6);
            }
        }
    }
}
```

generateQuiz(): Crea cuestionarios de ciberseguridad según el nivel.

```
function generateQuiz() {
    // Preguntas de cuestionario por nivel
    const quizQuestions = [
        // Preguntas de nivel 1
        [
            {
                question: "¿Qué es el phishing?",
                options: [
                    "Una técnica de pesca deportiva",
                    "Un ataque que se hace pasar por una entidad confiable para obtener información sensible",
                    "Un software para proteger contraseñas",
                    "Una técnica para mejorar la velocidad de internet"
                ],
                answer: 1
            },
            // Más preguntas...
        ],
        // Preguntas para otros niveles...
    ];
```

```
    // Obtiene preguntas para el nivel actual
    const levelQuestions = quizQuestions[currentLevel - 1];

    // Limpia el cuestionario anterior
    quizContainer.innerHTML = '';

    // Añade preguntas al cuestionario con manipulación del DOM
    levelQuestions.forEach((q, index) => {
        const questionDiv = document.createElement('div');
        questionDiv.className = 'quiz-question';

        // Más código para crear la estructura del cuestionario...
    });
}
```

6. Sistema de Sonido

initSounds(): Inicialice los efectos de sonido.

```
function initSounds() {
    moveSound = document.getElementById('move-sound');
    rotateSound = document.getElementById('rotate-sound');
    dropSound = document.getElementById('drop-sound');
    lineClearSound = document.getElementById('line-clear-sound');
    levelCompleteSound = document.getElementById('level-complete-sound');
    gameOverSound = document.getElementById('game-over-sound');

    // Establece el volumen para todos los sonidos
    const sounds = [moveSound, rotateSound, dropSound, lineClearSound,
        levelCompleteSound, gameOverSound];
    sounds.forEach(sound => {
        if (sound) {
            sound.volume = 0.3;
        }
    });
}
```

playSound(): Reproduce efectos de sonido con manejo de errores.

```
function playSound(sound) {
    if (soundEnabled && sound) {
        // Reinicia el sonido para reproducirlo desde el principio
        sound.currentTime = 0;
        sound.play().catch(e => {
            // Ignora errores relacionados con requerimientos de interacción del usuario para audio
            console.log("Sound couldn't play automatically. This is normal if there hasn't been user interaction.");
        });
    }
}
```

7. Interacción del usuario

Gestión de eventos

setupEventListeners(): Configura todos los controladores de eventos.

```
function setupEventListeners() {
    // Botones del menú principal
    startBtn.addEventListener('click', startGame);
    storyModeBtn.addEventListener('click', showStoryScreen);
    infoBtn.addEventListener('click', showInfoScreen);
    leaderboardBtn.addEventListener('click', showLeaderboard);
    saveUsernameBtn.addEventListener('click', saveUsername);

    // Botones de regreso
    gameBackBtn.addEventListener('click', () => {
        pauseGame();
        showMainMenu();
    });
    // Más botones...

    // Controles de juego
    leftBtn.addEventListener('click', () => movePiece(-1));
    rightBtn.addEventListener('click', () => movePiece(1));
    rotateBtn.addEventListener('click', rotatePiece);
    downBtn.addEventListener('click', dropPiece);

    // Navegación de nivel
    nextLevelBtn.addEventListener('click', startNextLevel);
    retryBtn.addEventListener('click', restartGame);
    homeBtn.addEventListener('click', showMainMenu);

    // Controles por teclado
    document.addEventListener('keydown', handleKeyDown);
}
```

handleKeyDown(): Gestiona los controles de teclado.

```
function handleKeyDown(e) {
    if (!gameActive) return;
    switch(e.key) {
        case 'ArrowLeft':
            movePiece(-1);
            break;
        case 'ArrowRight':
            movePiece(1);
            break;
        case 'ArrowUp':
            rotatePiece();
            break;
        case 'ArrowDown':
        case ' ':
            dropPiece();
            break;
    }
}
```

8. Persistencia de Datos con localStorage

checkSavedGame(): Comprueba si hay un juego guardado.

```
function checkSavedGame() {
    const savedLevel = localStorage.getItem('tewerly_level');
    if (savedLevel) {
        currentLevel = parseInt(savedLevel);
        continueText.textContent = ` (Continuar Nivel ${currentLevel})`;
    } else {
        continueText.textContent = '';
    }
}
```

saveUsername(): Guarde el nombre del usuario.

```
function saveUsername() {
    const name = usernameInput.value.trim();
    if (name) {
        username = name;
        localStorage.setItem('tewerly_username', username);
        playerWelcome.textContent = `¡Bienvenido, Agente ${username}!`;
        alert(`¡Nombre guardado: ${username}!`);
    } else {
        alert('Por favor, ingresa un nombre válido.');
    }
}
```

saveScoreToLeaderboard(): Guarda evaluación en la tabla de clasificación.

```
function saveScoreToLeaderboard() {
    // Obtiene la tabla de clasificación existente o inicializa un array vacío
    let leaderboard = JSON.parse(localStorage.getItem('tewerly_leaderboard') || '[]');

    // Añade la puntuación actual a la tabla
    leaderboard.push({
        name: username || 'Anónimo',
        score: score,
        level: currentLevel,
        date: new Date().toLocaleDateString()
    });

    // Ordena la tabla por puntuación (mayor primero)
    leaderboard.sort((a, b) => b.score - a.score);

    // Mantiene solo las 10 mejores puntuaciones
    leaderboard = leaderboard.slice(0, 10);

    // Guarda la tabla actualizada
    localStorage.setItem('tewerly_leaderboard', JSON.stringify(leaderboard));
}
```

showLeaderboard(): Muestra la tabla de evaluación desde localStorage.

```
function showLeaderboard() {
    hideAllScreens();

    // Obtiene datos de la tabla de clasificación
    const leaderboard = JSON.parse(localStorage.getItem('tewerly_leaderboard') || '[]');

    // Limpia la tabla anterior
    leaderboardContainer.innerHTML = '';

    if (leaderboard.length === 0) {
        leaderboardContainer.innerHTML = '<p class="info-text">No hay puntuaciones guardadas.</p>';
    } else {
        // Crea tabla de clasificación con manipulación del DOM
        // ...
    }

    leaderboardScreen.style.display = 'flex';
}
```

9. Modo Historia y Narrativa

startStoryMode(): Inicia el modo historia con contenido narrativo.

```
function startStoryMode(level) {
    isStoryMode = true;
    currentLevel = level;

    // Muestra mensaje de la historia antes de empezar el nivel
    hideAllScreens();
    storyMessageTitle.textContent = storyContent[level-1].title;
    storyMessageContent.innerHTML = storyContent[level-1].intro;

    // Establece imagen de fondo según el nivel
    storyMessageScreen.style.backgroundImage = `url(https://images.unsplash.com/photo-${level === 1 ? '1550751827-4bd374c
                                               level === 2 ? '1558494949-ef010cbdcc31' :
                                               level === 3 ? '1526374965328-7f61d4dc18c5' :
                                               level === 4 ? '1544197150-b99a580bb7a8' :
                                               '1537498425277-c283d32ef9db'})`;

    storyMessageScreen.style.display = 'flex';
}
```

El código JavaScript de Tewerly implementa un juego completo tipo Tetris con funcionalidades adicionales como modo historia, cuestionarios educativos y persistencia de datos, todo organizado de manera modular y mantenible con funciones específicas para cada aspecto del juego.

9.PRUEBAS Y DEPURACIÓN

Métodos utilizados para probar el juego
A lo largo de la evolución de Tewerly desde su versión inicial hasta la versión 3.5 actual, se han empleado diversos métodos de prueba para garantizar una experiencia de usuario óptima:

1. Pruebas de Audio y Sistema de Sonido
El sistema completo de sonidos implementado en la versión 3.5 requirió pruebas extensas en diferentes navegadores y dispositivos:

```javascript
// Inicialización de efectos de sonido con manejo de volumen
function initSounds() {
    moveSound = document.getElementById('move-sound');
    rotateSound = document.getElementById('rotate-sound');
    dropSound = document.getElementById('drop-sound');
    lineClearSound = document.getElementById('line-clear-sound');
    levelCompleteSound = document.getElementById('level-complete-sound');
    gameOverSound = document.getElementById('game-over-sound');

    // Establece el volumen para todos los sonidos
    const sounds = [moveSound, rotateSound, dropSound, lineClearSound, levelCompleteSound, gameOverSound];
    sounds.forEach(sound => {
        if (sound) {
            sound.volume = 0.3;  // Volumen optimizado tras pruebas de equilibrio
        }
    });
}
```

Las pruebas revelaron que los navegadores tienen diferentes políticas de reproducción automática, lo que llevó a la implementación de un sistema de manejo de errores robusto:

```javascript
function playSound(sound) {
    if (soundEnabled && sound) {
        sound.currentTime = 0;
        sound.play().catch(e => {
            // Manejo de errores para reproducción automática
            console.log("Sound couldn't play automatically. This is normal if there hasn't been user interaction.");
        });
    }
}
```

2. Pruebas de Carga y Rendimiento de Imágenes

Las imágenes de fondo para el modo historia se sometieron a pruebas de rendimiento, resultando en una estrategia de precarga:

```html
<!-- Precarga de imágenes para el modo historia -->
<div style="display: none">
    <img id="bg-episode1" src="https://images.unsplash.com/photo-1550751827-4bd374c3f58b" alt="Episodio 1">
    <img id="bg-episode2" src="https://images.unsplash.com/photo-1558494949-ef010cbdcc31" alt="Episodio 2">
    <img id="bg-episode3" src="https://images.unsplash.com/photo-1526374965328-7f61d4dc18c5" alt="Episodio 3">
    <img id="bg-episode4" src="https://images.unsplash.com/photo-1544197150-b99a580bb7a8" alt="Episodio 4">
    <img id="bg-episode5" src="https://images.unsplash.com/photo-1537498425277-c283d32ef9db" alt="Episodio 5">
</div>
```

3. Pruebas de Equilibrio de Dificultad

Para la versión 3.5, se realizaron extensas pruebas de juego para calibrar la progresión de dificultad:

```javascript
// Ajustes de dificultad refinados tras pruebas de jugabilidad
gameSpeedFactor = 1 + (currentLevel - 1) * 0.15; // Incremento de 0.15 por nivel
dropInterval = 800 - (currentLevel - 1) * 70;   // Reducción de 70ms por nivel
```

4. Pruebas de Persistencia de Datos

El sistema de tabla de evaluación fue rigurosamente probado para garantizar la correcta clasificación y almacenamiento:

```
function saveScoreToLeaderboard() {
    let leaderboard = JSON.parse(localStorage.getItem('tewerly_leaderboard') || '[]');

    leaderboard.push({
        name: username || 'Anónimo',
        score: score,
        level: currentLevel,
        date: new Date().toLocaleDateString()
    });

    // Ordenamiento comprobado para garantizar precisión
    leaderboard.sort((a, b) => b.score - a.score);

    // Limita a 10 mejores puntuaciones tras pruebas de rendimiento
    leaderboard = leaderboard.slice(0, 10);

    localStorage.setItem('tewerly_leaderboard', JSON.stringify(leaderboard));
}
```

5. Pruebas de compatibilidad entre navegadores y dispositivos

La interfaz perfeccionada de la versión 3.5 requirió pruebas en múltiples plataformas:

```
// Detección y adaptación a preferencias de modo oscuro/claro
if (window.matchMedia && window.matchMedia('(prefers-color-scheme: dark)').matches) {
    document.documentElement.classList.add('dark');
}
window.matchMedia('(prefers-color-scheme: dark)').addEventListener('change', event => {
    if (event.matches) {
        document.documentElement.classList.add('dark');
    } else {
        document.documentElement.classList.remove('dark');
    }
});
```

Problemas Encontrados y Soluciones
Durante la evolución hasta la versión 3.5, se identificaron y solucionaron varios problemas significativos:

1. Balanceo del Sistema de Audio

Problema: Las primeras implementaciones tenían volúmenes inconsistentes entre efectos de sonido, resultando en una experiencia desequilibrada.

Solución: Normalización de volumen a 0.3 para todos los efectos de sonido, tras pruebas para encontrar el nivel óptimo entre audible y no intrusivo.

2. Rendimiento con Imágenes de Fondo

Problema: La carga de imágenes de fondo para el modo historia causaba retrasos perceptibles durante las transiciones entre pantallas.

Solución: Implementación de un sistema de precarga invisible que carga todas las imágenes al inicio:

```
<div style="display: none">
    <img id="bg-episode1" src="https://images.unsplash.com/photo-1550751827-4bd374c3f58b"
    alt="Episodio 1">
    <!-- Otras imágenes... -->
</div>
```

3. Curva de dificultad inconsistente

Problema: En versiones anteriores, la progresión de dificultad era demasiado abrupta o demasiado plana.

Solución: Refinamiento exhaustivo de los parámetros de dificultad:

- Aumento del factor de velocidad a 0.15 por nivel (antes era menor)
- Incremento de la reducción del intervalo de caída a 70ms por nivel (anteriormente 50ms)
- Implementación de cambios graduales dentro de cada nivel basados en el tiempo restante

```
// Ajuste gradual de velocidad durante el nivel
if (gameTimer <= 150 && gameTimer > 120) {
    gameSpeedFactor = 1.1 + (currentLevel - 1) * 0.15;
} else if (gameTimer <= 120 && gameTimer > 90) {
    gameSpeedFactor = 1.2 + (currentLevel - 1) * 0.15;
}
// Más incrementos graduales...
```

4. Pérdida de Datos en la Tabla de Puntuaciones

Problema: Ocasionalmente, las evaluación no se guardaban correctamente o se duplicaban en la tabla.

Solución: Mejora del algoritmo de clasificación y almacenamiento con validación adicional:

```
// Ordenamiento preciso por puntuación
leaderboard.sort((a, b) => b.score - a.score);

// Eliminación de duplicados potenciales manteniendo solo las 10 mejores
leaderboard = leaderboard.slice(0, 10);
```

Capturas de errores corregidos

Durante el desarrollo de la versión 3.5, se identificaron y corrigieron varios errores técnicos:

1. Sincronización de audio
Error original: Los efectos de sonido se sobreponían y creaban distorsión al reproducirse múltiples veces rápidamente.

Solución implementada:

```
function playSound(sound) {
    if (soundEnabled && sound) {
        // Reinicio del tiempo para evitar superposición
        sound.currentTime = 0;
        sound.play().catch(e => {
            // Manejo de error silencioso
        });
    }
}
```

2. Gestión de Memoria con Imágenes

Error original: Fugas de memoria por carga repetida de imágenes de fondo.

Solución implementada: Precarga única de todas las imágenes al inicio y referencia a las ya cargadas durante el juego.

3. Problemas de Cancelación de Animación

Error original: Múltiples bucles de animación ejecutándose simultáneamente causaban problemas de rendimiento.

Solución implementada:

```
// Asegurar que solo hay un bucle de animación activo
if (animationFrameId) {
    cancelAnimationFrame(animationFrameId);
    animationFrameId = null;
}
```

4. Error de Rotación en Piezas Completas

Error original: Algunas piezas (como la pieza "Z") causaban glitches visuales al rotar cerca de los bordes.

Solución implementada: Mejora del sistema de "wall kicks" con comprobaciones adicionales de colisión.

Comentarios Recibido y Mejoras Implementadas

1. "Los sonidos son demasiado fuertes/molestos"
Respuesta: Ajuste global del volumen a 0.3 y mejora del manejo de reproducción automática.

2. "La progresión entre niveles es inconsistente"
Respuesta: Recalibración completa de la curva de dificultad con incrementos más graduales:
Factor de velocidad aumentado a 0,15 por nivel
Cambios de velocidad más suaves dentro de cada nivel

3. "Necesito saber dónde caerá exactamente la pieza"
Respuesta: Mejora del sistema de pieza fantasma para mayor claridad visual:

```
function drawGhostPiece() {
    // Encuentra la posición más baja posible
    while (canMoveTo(ghostPiece.row + 1, ghostPiece.col, ghostPiece.shape)) {
        ghostPiece.row++;
    }

    // Contorno más visible pero no distractor
    ctx.strokeStyle = ghostPiece.color;
    ctx.lineWidth = 2;
    ctx.strokeRect(x + 3, y + 3, CELL_SIZE - 6, CELL_SIZE - 6);
}
```

4. "Los logros no se guardan correctamente"
Respuesta: Implementación de sistema robusto de tabla de evaluación con ordenamiento y almacenamientos mejorados.

5. "La experiencia narrativa podría ser más inmersiva"
Respuesta: Adición de imágenes de fondo específicas para cada episodio y efectos de sonido que refuerzan momentos clave de la historia.

La versión 3.5 de Tewerly representa una culminación de múltiples ciclos de prueba, análisis de errores y respuesta. Cada componente del juego ha sido refinado para crear una experiencia educativa inmersiva que equilibra el desafío y la accesibilidad, mientras mantiene una estética retro coherente con su temática de ciberseguridad.

10.CONCLUSIONES Y MEJORAS FUTURA

Información clave a considerar de la evolución mostrada:

Tewerly comenzó como un juego de bloques básico y evolucionó a una experiencia educativa completa con narrativa.
Ha pasado por Múltiples iteraciones incrementando progresivamente su complejidad técnica.
La versión 3.5 agregó sistemas de sonido, imágenes y mejoras de jugabilidad significativas.
El juego combina entretenimiento con educación sobre ciberseguridad.

Evaluación personal del proceso de desarrollo.

El desarrollo de Tewerly representa un caso exitoso de evolución progresiva de un concepto de juego, pasando de un simple rompecabezas basado en Tetris a una experiencia educativa inmersiva sobre ciberseguridad. Este proceso de desarrollo iterativo ha permitido:

Equilibrio entre entretenimiento y educación : El mayor logro de Tewerly es haber encontrado un punto óptimo donde la jugabilidad adictiva del formato Tetris sirve como vehículo para transmitir conocimientos sobre ciberseguridad sin sacrificar la diversión.

Arquitectura de código modular : La estructura implementada ha facilitado la expansión constante de funcionalidades sin necesidad de reescrituras completas. El código está organizado en funciones específicas con responsabilidades bien definidas, lo que ha permitido añadir nuevas características sin desestabilizar el sistema existente.

Interfaz adaptativa : La atención prestada a la responsividad desde las primeras versiones ha resultado en una interfaz que funciona consistentemente bien en diferentes dispositivos y tamaños de pantalla, extendiendo el alcance potencial del juego.
Narración integrada en la mecánica : La narrativa no es un simple adorno, sino que está integrada en la progresión del juego, donde cada nivel contribuye a un arco argumental coherente sobre amenazas cibernéticas.

Persistencia de datos efectiva : El uso estratégico de localStorage ha permitido implementar funciones como guardado de progreso y tabla de evaluación sin depender de infraestructura de servidor, manteniendo la aplicación simple pero funcional.
Desafíos enfrentados y cómo fueron superados.

1. Complejidad Creciente

Desafío : A medida que el juego evolucionaba de la versión 1.0 a la 3.5, la complejidad del código aumentaba, dificultando la adición de nuevas características sin introducir errores.
Solución : Implementación de una arquitectura basada en funciones independientes con responsabilidades claramente definidas. Por ejemplo, la función drawGame()coordina el renderizado pero delega las tareas específicas a drawGrid(), drawBoard()y drawPiece(), facilitando modificaciones aisladas.

3. Balance de Dificultad

Desafío : Crear una curva de dificultad que resultará desafiante pero no frustrante, manteniendo el interés a lo largo de los 5 niveles.
Solución : Implementación de un sistema multinivel de ajuste de dificultad:

- Incremento base por nivel
- Aceleración progresiva dentro de cada nivel
- Sistema de preguntas que recompensa el conocimiento en ciberseguridad

4. Experiencia Multimedia Cohesiva

Desafío : Integrar audio e imágenes de manera que enriquecerán la experiencia sin causar problemas de rendimiento o inconsistencias.

Solución :
Precarga de recursos multimedia
Manejo inteligente de reproducción de audio con control de errores
Limitación estratégica de efectos visuales para mantener el rendimiento

5. Narrativa Significativa
Desafío : Desarrollar una historia que complemente la mecánica del juego y transmitiera información educativa sin resultar forzada.

Solución : Creación de una narrativa sobre amenazas cibernéticas donde cada nivel representa un desafío diferente, haciendo que los cuestionarios de ciberseguridad formen parte natural de la progresión argumental.

Posibles mejoras futuras en el juego.

1. Expansión de Contenido Educativo

- Más preguntas de ciberseguridad : Ampliar el banco de preguntas para aumentar la variedad y evitar repeticiones.
- Consejos prácticos : Añadir pantallas con consejos útiles sobre seguridad digital después de cada nivel.
- Glosario de términos : Incluir un diccionario simple de términos de ciberseguridad accesible desde el menú principal.

2. Mejoras técnicas

- Optimización de código : Reducir redundancias en el código JavaScript para mejorar el rendimiento.
- Almacenamiento mejorado : Agregue compresión básica a los datos guardados en localStorage para almacenar más información.
- Precarga optimizada : Mejorar el sistema de precarga para reducir tiempos de espera entre pantallas.

3. Características Sociales y Competitivas

- Modo Multijugador Asíncrono : Permitir a los jugadores competir contra las evaluaciones/tiempos de sus amigos.
- Tablas de Clasificación Globales : Implementar API backend simple para tablas de clasificación más allá del almacenamiento local.
- Compartir Logros : Funcionalidad para compartir logros y evaluación en redes sociales.

4. Mejoras visuales y sonoras

- Temas Visuales Desbloqueables : Permitir a los jugadores ganar y seleccionar diferentes temas visuales.
- Efectos Partículas : Añadir sistemas de partículas para momentos clave como eliminación de líneas.
- Música Dinámica : Implementar sistema de música adaptativa que cambia según el estado del juego (tiempo restante, altura de la pila).

5. Accesibilidad y alcance

- Opciones de tamaño de texto : Agregue un selector simple para cambiar el tamaño de texto para una mejor legibilidad.
- Modo de alto contraste : Implementar un tema alternativo con mayor contraste para usuarios con dificultades visuales.
- Botón de silencio : Añade una opción fácilmente accesible para activar/desactivar todos los sonidos del juego.

Tewerly empezó como un juego de rompecabezas con un tema y ha crecido hasta ser una herramienta educativa muy útil que puede desarrollarse de muchas maneras. La buena base técnica y la historia que tiene en la versión 3.5 son un gran punto de partida para futuras versiones que pueden explorar más tanto en el juego como en la educación, siempre manteniendo el equilibrio que hace que el juego sea especial al combinar diversión y aprendizaje sobre ciberseguridad.

Desarrollo de Juegos y Programación

- Antonov, I. (2021). API Canvas: Desarrollo de juegos de navegador con HTML5 . Revista de Desarrollo Web, 12(3), 78-95. https://doi.org/10.1111/jwd.2021.0056
- Goldberg, E., y MacDonald, B. (2022). Fundamentos del desarrollo de juegos en JavaScript: Creación de juegos 2D desde cero (3.ª ed.). Packt Publishing.
- Lee, WM (2022). JavaScript: De principiante a ninja (4.ª ed.). SitePoint.
- Red de Desarrolladores de Mozilla. (2023). API de Canvas . Mozilla. https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API
- Weyl, E. (2020). HTML5 móvil: Uso de las últimas novedades (2.ª ed.). O'Reilly Media.

Tetris y juegos de rompecabezas

- Carr, D. (2019). El efecto Tetris: El juego que hipnotizó al mundo . Asuntos Públicos.
- Kim, SL y Park, JH (2021). Tetris como herramienta educativa: Una revisión sistemática. Revista Internacional de Aprendizaje Basado en Juegos , 11(2), 45-60. https://doi.org/10.4018/IJGBL.2021040104
- Koster, R. (2022). Una teoría de la diversión para el diseño de juegos (3.ª ed.). O'Reilly Media.
- Stein, A., y DeSalle, R. (2019). Una historia natural del color: La ciencia detrás de lo que vemos y cómo lo vemos . Pegasus Books.

Ciberseguridad

- Chong, M. (2022). Fundamentos de ciberseguridad para usuarios no técnicos . Wiley.
- Graham, J., Howard, R. y Olson, R. (2021). Concienciación sobre ciberseguridad: Ingeniería social y malware . Revista de Educación en Seguridad, 15(2), 105-117. https://doi.org/10.1080/jse.2021.4598732
- Stallings, W., y Brown, L. (2023). Seguridad informática: Principios y práctica (5.ª ed.). Pearson.

Recursos multimedia

- Font Awesome. (2023). Font Awesome 6.4.0 . [Software]. https://fontawesome.com/
- Fuentes de Google (2023). Fuente Press Start 2P . [Tipografía]. https://fonts.google.com/specimen/Press+Start+2P
- Mixkit (2023). Efectos de sonido de juegos . [Archivos de audio]. https://mixkit.co/free-sound-effects/game/
- Unsplash. (2023). Imágenes digitales y tecnológicas . [Fotografías]. https://unsplash.com/

Desarrollo Web y Técnicas de Programación

- Duckett, J. (2021). HTML y CSS: Diseño y creación de sitios web (2.ª ed.). Wiley.
- Flanagan, D. (2020). JavaScript: La guía definitiva (7.ª ed.). O'Reilly Media.
- Frain, B. (2022). Diseño web adaptable con HTML5 y CSS (4.ª ed.). Packt Publishing.
- Simpson, K. (2021). Aún no conoces JS: Comienza (2.ª ed.). O'Reilly Media.
- Verou, L. (2023). Secretos de CSS: Mejores soluciones a los problemas cotidianos de diseño web (2.ª ed.). O'Reilly Media.

Créditos
Sonidos

Todos los efectos de sonido utilizados en Tewerly fueron obtenidos de Mixkit (https://mixkit.co/) bajo su licencia de uso gratuito para proyectos personales y comerciales.

Imágenes

Las imágenes de fondo para los episodios del modo historia fueron obtenidas de Unsplash (https://unsplash.com/) bajo la Licencia Unsplash, que permite el uso gratuito sin atribución para fines comerciales y no comerciales.

Fuentes

La fuente "Press Start 2P" es una creación de CodeMan38, disponible a través de Google Fonts bajo la Licencia SIL Open Font License (OFL).

Iconos

Los iconos utilizados en la interfaz del juego provienen de Font Awesome 6.4.0, utilizados bajo los términos de la licencia Font Awesome Free.

TEWELY

UN PUZZLE DE CIBERSEGURIDAD