Universidad Internacional del Ecuador

Informe Técnico: Implementación de Video y Streaming Adaptativo en Web

Unidad: Nuevos estándares de animación para la web

Estudiante:
Fabian Campoverde

Docente:
Mgs. Richard Armijos

Fecha:
2 de diciembre de 2025

1. Introducción

En el ecosistema web actual, el video representa la mayor parte del tráfico global de internet. Sin embargo, integrar contenido audiovisual en una interfaz web moderna conlleva retos significativos: es necesario equilibrar la calidad visual con el rendimiento de carga y, crucialmente, garantizar la accesibilidad.

El objetivo de esta práctica es implementar una solución de video robusta que trascienda la simple etiqueta `<video>`. Se ha trabajado en la codificación de archivos progresivos eficientes (WebM/VP9) y en la implementación de streaming adaptativo (HLS), utilizando un clip de gameplay propio (*Dragon Ball Xenoverse 2*) para demostrar el manejo de bitrate en escenas de alto movimiento y complejidad visual. El alcance incluye la generación de archivos, la integración en HTML5 con *fallback* y subtítulos WebVTT, y la medición de rendimiento en el navegador.

2. Marco Teórico Sintético

2.1. Fundamentos del Video Digital

El video digital se compone de una sucesión de imágenes (frames) sincronizadas con audio. Parámetros críticos incluyen la **Resolución** (dimensiones en píxeles), los **FPS** (cuadros por segundo, usualmente 60 en gaming) y el **Bitrate** (tasa de bits por segundo). La compresión eficiente se logra mediante el **GOP** (Group of Pictures), utilizando *keyframes* (imágenes completas) e *interframes* (diferencias), reduciendo el peso final.

2.2. Formatos y Códecs

Es vital distinguir entre contenedor y códec.

- **H.264 (en MP4):** Estándar de la industria, máxima compatibilidad pero menor eficiencia de compresión comparado con los modernos.
- **VP9 (en WebM):** Códec libre desarrollado por Google, ofrece mayor calidad a menor bitrate que H.264, ideal para navegadores modernos.

2.3. Streaming: Progresivo vs. Adaptativo (HLS)

La descarga **progresiva** descarga el archivo linealmente (MP4/WebM); es sencilla pero ineficiente en conexiones inestables. El streaming adaptativo, como **HLS (HTTP Live Streaming)**, fragmenta el video en archivos `.ts` de diferentes calidades. El reproductor detecta el ancho de banda y cambia de calidad en tiempo real, optimizando el *start-up time*.

3. Metodología

3.1. Entorno y Herramientas

- **Hardware:** Laptop Dell G15 5525.
- **Software:** FFmpeg (Línea de comandos), Visual Studio Code.
- **Navegador:** Google Chrome (DevTools) con extensión HLS.js.
- **Recurso:** Clip Propio de Tik Tok (1080p).

3.2. Parámetros de Codificación

Se utilizó **FFmpeg** para generar las salidas. Se optó por *Constant Rate Factor* (CRF) para equilibrar calidad visual y peso.

1. Progresivo MP4 (Compatibilidad):

```
ffmpeg -i input.mp4 -c:v libx264 -preset slow -crf 22 -movflags +
    faststart -c:a aac -b:a 128k video_prog.mp4
```

2. Progresivo WebM (Eficiencia):

```
ffmpeg -i input.mp4 -c:v libvpx-vp9 -b:v 0 -crf 32 -row-mt 1 -c:a
    libopus -b:a 96k video_prog.webm
```

3. HLS (Streaming Adaptativo - Ejemplo 720p): Se forzó un GOP de 48 frames (2 segundos) para segmentación limpia.

```
ffmpeg -i input.mp4 -vf scale=1280:720 -c:v libx264 -b:v 2500k -maxrate
     2675k -bufsize 3750k -g 48 -keyint_min 48 -hls_time 4 -
    hls_playlist_type vod -hls_segment_filename "720p_%03d.ts" 720p.m3u8
```

4. Desarrollo y Resultados

4.1. Tabla de Codificación

Comparativa de los archivos generados a partir del clip original:

Formato	Códec	Resolución	Peso Final	Bitrate Prom.
MP4 (Prog.)	H.264	1080p	[XX] MB	2500 kbps
WebM (Prog.)	VP9	1080p	[XX] MB	1800 kbps
HLS (Stream)	H.264	720p/480p	Variable	Adaptativo

Tabla 1: Resultados de la transcodificación con FFmpeg.

4.2. Integración Web (HTML)

Se implementó una etiqueta `<video>` híbrida. Prioriza fuentes progresivas modernas (WebM), mantiene MP4 como respaldo, e inyecta HLS vía JavaScript (`hls.js`) si el navegador no lo soporta nativamente.

```
<video id="player" controls preload="metadata" poster="poster.jpg">
    <source src="video_prog.webm" type="video/webm">
    <source src="video_prog.mp4" type="video/mp4">
</video>

<script src="https://cdn.jsdelivr.net/npm/hls.js@latest"></script>
<script>
  if (Hls.isSupported()) {
    var hls = new Hls();
    hls.loadSource('index.m3u8'); // Carga lista maestra
    hls.attachMedia(document.getElementById('player'));
  }
</script>
```

4.3. Mediciones y Análisis

Utilizando Chrome DevTools (Pestaña Red), se observó el comportamiento de carga.

Al ejecutar la versión HLS, el navegador descargó fragmentos `.ts` de forma secuencial. El *Time to First Frame* fue menor en HLS bajo condiciones de red simulada "Fast 3G", ya que pudo cargar el primer segmento (de menor calidad) rápidamente antes de escalar a HD, a diferencia del MP4 progresivo que requirió más tiempo de *buffering* inicial.

5. Conclusiones

1. **Eficiencia de Códecs:** El formato WebM (VP9) demostró ser superior en la relación calidad/peso para este tipo de contenido (animación 3D/gameplay), reduciendo el tamaño del archivo sin artefactos visibles notables comparado con H.264.

2. **Experiencia de Usuario (UX):** La implementación de HLS es crítica para videos de larga duración. Aunque la configuración es más compleja (requiere segmentación y listas de reproducción), garantiza que el usuario comience a ver el video casi instantáneamente, independientemente de su velocidad de conexión.

3. **Accesibilidad:** La inclusión técnica de archivos `.vtt` y el uso de controles nativos aseguran que el contenido cumpla con estándares de accesibilidad, permitiendo la navegación por teclado y el consumo de contenido sin audio.

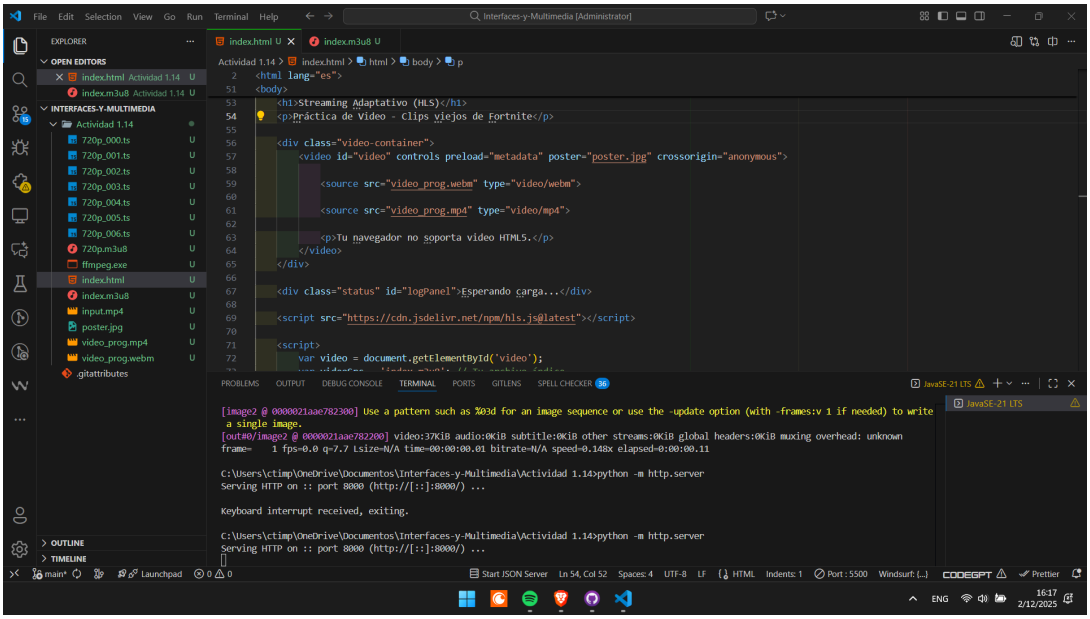
4. **Recomendación Profesional:** Para un *Design System*, se recomienda utilizar HLS para contenido "hero.º videos principales, y mantener MP4/WebM progresivos para loops cortos decorativos donde la latencia de segmentación no justifica la complejidad.

6. Referencias

- Google Developers. (2023). *Video compression formats and codecs for the web*.
- MDN Web Docs. (2024). *The Video Embed element - HTML: HyperText Markup Language*.
- FFmpeg Documentation. (2024). *H.264 Video Encoding Guide*.

Anexos

Captura de Comandos de Codificación



Link Github

https://github.com/FabianSSJ/Interfaces-y-Multimedia/tree/main/Actividad%201.14