

Copilot-Chat Sicherung - hübsch formatiert

Erstellt am 2026-01-23 13:25

Enthält die von dir in diesem Chat geposteten Teile (1-10 + letzter Code-Block).

Yesterday

You

Hi, ich habe ein Powershell script mit dem ich automatisch beim Computerstart Videos abspielen kann. Man kann auch auswählen welches Video abgespielt werden soll und im Script wird definiert auf welchem Bildschirm das abgespielt werden soll. Jetzt möchte ich das einwenig besser machen und ein richtiges Programm erstellen in dem man Einstellungen machen kann, Videos importieren usw. und das automatisch bei Computerstart gestartet werden kann. Kannst du mir dabei helfen? Ich kann dir auch den Code des Scripts mal zeigen

Copilot

Copilot

Reasoned for 1s

Zusammenfassung anbieten

Vorschläge unterbreiten

Ja, sehr gern — das ist ein super nächster Schritt. Aus deinem PowerShell-Script hast du im Prinzip schon die „Business-Logik“: Video auswählen → auf Monitor X abspielen → beim Systemstart starten. Daraus lässt sich gut ein richtiges Windows-Programm machen, mit UI für Einstellungen, Videoliste/Import, Autostart usw.

Wenn du willst, kannst du mir gleich den Code deines Scripts schicken (gern komplett oder die wichtigsten Teile). Dann kann ich dir sehr konkret zeigen, wie man das 1:1 in eine App überführt.

1) Zielbild: Was soll die App können?

Typische Features (wir können das gemeinsam priorisieren):

Grundfunktionen

Videos importieren (Datei hinzufügen, ggf. Ordner hinzufügen)

Video auswählen (Standardvideo)

Ziel-Monitor auswählen (z. B. Monitor 1/2/3)

Startoptionen:

Vollbild

Loop (Endlosschleife)

Ton an/aus, Lautstärke

Startverzögerung (z. B. 5–10 Sekunden nach Login/Boot)

„Starten“ / „Stoppen“ / „Test abspielen“

Automatisierung

App startet automatisch beim Systemstart (sauber über Task Scheduler oder Autostart)

Optional: im Hintergrund starten und sofort abspielen

Verwaltung / Stabilität

Einstellungen speichern (JSON)

Videofadverwaltung (prüfen ob Dateien noch existieren)

Logs/Diagnose (falls ein Codec fehlt oder Monitor nicht verfügbar ist)

2) Welche Technologie eignet sich am besten?

Da du Windows + PowerShell nutzt, ist .NET naheliegend.

Option A (empfohlen): C# + WPF (.NET 8) + LibVLCSharp

Warum: Sehr robust bei Videoformaten (H.264/H.265, MKV, etc.), Multi-Monitor sauber steuerbar, Vollbild/Loop zuverlässig.

Unterstützt fast alle Codecs (bringt VLC Engine mit)

Sehr stabile Wiedergabe

Einfaches Packaging möglich

Option B: C# + WPF/WinUI + MediaElement

Keine zusätzlichen Libraries

Hängt stark von installierten Windows Codecs ab

Kann bei MKV/H.265 schnell Probleme machen

Wenn du „es soll einfach immer laufen“ willst: LibVLCSharp.

3) Architekturvorschlag (übersichtlich & wartbar)

Datenmodell

AppSettings.json im App-Datenordner:

Standardvideo

MonitorIndex
Vollbild/Loop/Mute/Volume
Liste importierter Videos (oder einfach letzter Importordner)
Bausteine
UI (Settings Window)

Videoliste + Import Button
Monitor Drop-Down (alle Displays)
Checkboxen (Loop, Vollbild, Mute, Start Delay)
Button „Test abspielen“
Player Window (separates Fenster)

Öffnet auf dem ausgewählten Monitor
Vollbild, ohne Rahmen (optional)
Startet Wiedergabe
Startup Manager

Autostart ein/aus:
Entweder Startup-Ordner (simpel)
Oder Task Scheduler (zuverlässiger, auch mit Delay/Privilegien)
4) Multi-Monitor: so macht man's in .NET (Prinzip)
WPF kann Monitore über System.Windows.Forms.Screen abfragen und Fenster gezielt platzieren.

Idee:

Alle Screens auflisten → Nutzer wählt z. B. „Display 2 (1920×1080)“
Player-Fenster wird auf dessen Bounds positioniert und in Vollbild gesetzt.
5) Autostart: „sauber“ umsetzen
Du hast zwei gute Wege:

A) Autostart-Ordner (einfach)
Verknüpfung deiner App in:
%AppData%\Microsoft\Windows\Start Menu\Programs\Startup
B) Task Scheduler (empfohlen für Delay / „nur wenn angemeldet“ / höhere Zuverlässigkeit)
Task: „At log on“
Optional: „Delay task for 10 seconds“
Startet deine exe mit Parametern z. B. --autoplay
Das ist meist stabiler als Registry-Run Keys.

6) Minimaler MVP-Plan (damit du schnell etwas Funktionierendes hast)
Phase 1 (MVP, 1-2 Abende):

WPF App
Settings speichern (JSON)
Monitor auswählen
Video auswählen + Test abspielen
Player Fenster Vollbild auf Monitor X

Phase 2:

Videoimport (Liste)
Loop/Mute/Volume
Autostart-Toggle (Startup folder oder Task Scheduler)

Phase 3 (Polish):

Packaging (MSIX oder Installer)
Logging, Fehleranzeigen
Tray-Icon (optional)
7) Was ich von dir brauche (damit ich zielgenau helfen kann)
Schick mir bitte:

Dein PowerShell Script (oder zumindest die Teile für: Videoauswahl + Monitorwahl + Start bei Boot)
Welche Player-Methode nutzt du gerade?
VLC? Windows Media Player? mpv? ffplay? Start-Process mit einem Player?
Muss das Video ohne sichtbare UI laufen (kioskartig), oder ist ein Fenster ok?
Soll es schon beim Boot (vor Login) starten oder nach Login?
8) Wenn du willst: Ich kann direkt mit dir den Startcode aufsetzen
Wenn du sagst „WPF + LibVLCSharp“, kann ich dir:

Projektstruktur vorschlagen
AppSettings-Klasse + JSON-Speicherung
Player-Fenster mit Monitorplatzierung
Argumente wie --autoplay / --settings
... als sauberes Grundgerüst erstellen.
 Schick mir gern den Code deines Scripts (einfach hier rein kopieren).
Dann sag ich dir auch, welche Teile wir 1:1 übernehmen und welche wir „app-mäßig“ besser lösen.

You

```
#Get current script location
$thisLoc = $PSScriptRoot

#Get all video files in the script location
$arr = Get-ChildItem $thisLoc .\* -include (*.mp4, *.webm, *.mov, *.wmv, *.avi) -recurse |
Foreach-Object {$_.Name}

echo "Default video will start in 10 Seconds. To select a specific video press any key!"

#Output for 10 seconds, if no input select first video to show, else show video selection
function Check-Keypress ($sleepSeconds = 10) {
$timer = [Diagnostics.Stopwatch]::StartNew()
$interrupted = $false

while ($timer.elapsed.TotalSeconds -lt 10) {
if ($Host.UI.RawUI.KeyAvailable)
{
$selection = $Host.UI.RawUI.ReadKey("NoEcho, IncludeKeyUp, IncludeKeyDown")
if ($selection.KeyDown -eq "True") {
$interrupted = $true
break
}
}
}
}

$timer.Stop()
return $interrupted
}

If (Check-Keypress){
$i = 0
foreach ($v in $arr){
echo $i": "$v
++$i
}

$selection = read-host -Prompt "Enter number to play specific video"
}
else{
write-host 0
}

#start VLC player on the buildin display (--video-x=1920 need to be adjusted depending on resolution
#and position of the screen, negative value = left side)
& "C:\Programme\VideoLAN\VLC\vlc.exe" --one-instance --no-embedded-video --audio --repeat
--loop --no-random --autoscale --fullscreen --no-osd --playlist-tree --start-time= --stop-time=
--video-x=1920 --video-y=1 --quiet "$(($thisLoc)\$($arr[$selection]))" vlc://quit
```

Copilot

Copilot

Hier ist Teil 2 des Chats

Reasoned for 3s

Vorschlagen von Technologien

Vorschlagen von Einstellungen

Mega – danke fürs Script! ☐ Damit kann man sehr gut erkennen, welche Kernfunktionen du brauchst und wie wir sie „app-tauglich“ nachbauen.

Ich fasse kurz zusammen, was dein Script aktuell macht, und zeige dir dann einen sauberen Weg zu einer echten Windows-App (mit UI, Import, Einstellungen, Autostart, Multi-Monitor) – inklusive konkretem C#/.NET-Grundgerüst.

1) Was dein PowerShell-Script genau tut (als Funktionsliste)

Dein Script:

Ermittelt den Script-Ordner (\$PSScriptRoot)

Sucht rekursiv Videodateien (mp4/webm/mov/wmv/avi) und nimmt nur die Dateinamen (\$_.Name)

Wartet 10 Sekunden auf Tastendruck:

wenn Taste gedrückt → zeigt Liste → User wählt Index

sonst → spielt Standard (0)

Startet VLC mit Parametern:

Vollbild, Loop/Repeat, keine OSD, etc.

Positionierung über --video-x / --video-y (Pixel-Offset)

spielt die gewählte Datei ab und beendet VLC danach (vlc://quit)

☐ Das ist bereits eine perfekte Spezifikation fürs spätere Programm.

2) Empfehlung für die „richtige App“ (robust + unkompliziert)

☐ Technologie-Empfehlung (Windows)

C# + WPF (.NET 8) + LibVLCSharp

Warum genau das zu deinem Szenario passt:

VLC kann praktisch alle Formate zuverlässig (auch MKV, H.265 etc.)

Multi-Monitor / Fullscreen ist sauber machbar

UI + Settings + Autostart ist in .NET sehr gut lösbar

Alternative wäre „MediaElement“, aber das ist codec-abhängig und macht bei manchen Formaten Probleme. Du nutzt schon VLC – also lohnt sich LibVLCSharp.

3) Wie wir dein Pixel-Offset --video-x=1920 „richtig“ ersetzen

In einer App sollten wir nicht mit harten Pixelwerten arbeiten, sondern den Monitor korrekt abfragen:

Windows kennt Monitore als Screen.AllScreens

Jeder Screen hat Bounds (Left/Top/Width/Height)

Wenn der Monitor links ist, sind Coordinates sogar negativ → alles automatisch korrekt

Damit wird dein „VLC-video-x“ ersetzt durch:

Player-Fenster wird genau auf Bounds des gewählten Monitors gesetzt

Dann Fullscreen/Borderless

4) App-Architektur (simpel, aber sauber)

Zwei Modi:

Settings-Modus (UI)

Videos importieren

Standardvideo wählen

Monitor wählen

Loop/Mute/Delay setzen

Autostart an/aus

Autoplay-Modus (--autoplay)

lädt Settings

wartet ggf. Delay

öffnet Player-Fenster auf Monitor X

spielt Standardvideo ab (Loop optional)

5) Konkreter Start: Projekt erstellen + NuGet

Projekt erstellen

Code (Shell)

```
dotnet new wpf -n VideoBootPlayer
```

```
cd VideoBootPlayer
```

```
NuGet Packages
```

```
In Visual Studio: Manage NuGet Packages → installieren:
```

```
LibVLCSharp.WPF
```

```
VideoLAN.LibVLC.Windows (bringt die VLC engine mit)
```

6) Settings speichern (JSON) – 1:1 passend zu deinem Script
AppSettings.cs

Code (C#)

```
using System.Text.Json;

public class AppSettings
{
    public List<string> Videos { get; set; } = new();
    public int DefaultVideoIndex { get; set; } = 0;
    public int MonitorIndex { get; set; } = 0;

    public bool Fullscreen { get; set; } = true;
    public bool Loop { get; set; } = true;
    public bool Mute { get; set; } = false;
    public int Volume { get; set; } = 100;

    public int StartDelaySeconds { get; set; } = 10;
}
```

SettingsStore.cs

Code (C#)

```
using System.Text.Json;

public static class SettingsStore
{
    private static string Folder =>
        Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData),
        "VideoBootPlayer");

    private static string FilePath => Path.Combine(Folder, "settings.json");

    public static AppSettings Load()
    {
        Directory.CreateDirectory(Folder);
        if (!File.Exists(FilePath)) return new AppSettings();
        var json = File.ReadAllText(FilePath);
        return JsonSerializer.Deserialize<AppSettings>(json) ?? new AppSettings();
    }

    public static void Save(AppSettings settings)
    {
        Directory.CreateDirectory(Folder);
        var json = JsonSerializer.Serialize(settings, new JsonSerializerOptions { WriteIndented = true });
        File.WriteAllText(FilePath, json);
    }
}
```

7) Multi-Monitor Auswahl (wie die UI später die Monitore listet)

In WPF nutzt man hierfür praktisch:

Hier ist Teil 3 des Chats

Code (C#)

```
using System.Windows.Forms; // Verweis auf WindowsForms

var screens = Screen.AllScreens;
for (int i = 0; i < screens.Length; i++)
{
    var s = screens[i];
    Console.WriteLine("{i}: {s.DeviceName} {s.Bounds.Width}x{s.Bounds.Height} @
{s.Bounds.Left},{s.Bounds.Top}");
}
```

8) Player-Fenster mit LibVLCSharp (Vollbild auf Monitor X)
PlayerWindow.xaml

Code (XML)

```
<Window x:Class="VideoBootPlayer.PlayerWindow"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
```

```
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:vlc="clr-namespace:LibVLCSharp.WPF;assembly=LibVLCSharp.WPF"
WindowStyle="None" ResizeMode="NoResize" Topmost="True"
ShowInTaskbar="False" Background="Black">
<Grid>
<vlc:VideoView x:Name="VideoView" />
</Grid>
</Window>
PlayerWindow.xaml.cs
```

Code (C#)

```
using LibVLCSharp.Shared;
using System.Windows;
using System.Windows.Forms;

namespace VideoBootPlayer;

public partial class PlayerWindow : Window
{
    private readonly LibVLC _libVlc;
    private readonly MediaPlayer _player;

    public PlayerWindow(AppSettings settings)
    {
        InitializeComponent();
        Core.Initialize();

        _libVlc = new LibVLC();
        _player = new MediaPlayer(_libVlc);
        VideoView.MediaPlayer = _player;

        PlaceOnMonitor(settings.MonitorIndex);
        ApplyPlayerSettings(settings);
        Play(settings);
    }

    private void PlaceOnMonitor(int monitorIndex)
    {
        var screens = Screen.AllScreens;
        if (monitorIndex < 0 || monitorIndex >= screens.Length) monitorIndex = 0;

        var b = screens[monitorIndex].Bounds;

        Left = b.Left;
        Top = b.Top;
        Width = b.Width;
        Height = b.Height;
        WindowState = WindowState.Normal;
    }

    private void ApplyPlayerSettings(AppSettings s)
    {
        if (s.Mute) _player.Mute = true;
        else _player.Volume = Math.Clamp(s.Volume, 0, 100);

        if (s.Fullscreen)
        {
            WindowStyle = WindowStyle.None;
            WindowState = WindowState.Maximized;
        }
    }

    private void Play(AppSettings s)
    {
        if (s.Videos.Count == 0) return;
        var index = Math.Clamp(s.DefaultVideoIndex, 0, s.Videos.Count - 1);
        var path = s.Videos[index];

        if (!File.Exists(path)) return;
```

```
var media = new Media(_libVlc, new Uri(path));

if (s.Loop)
{
    player.EndReached += (_, _) => Dispatcher.Invoke(() => _player.Play(media));
}

_player.Play(media);
}

protected override void OnClosed(EventArgs e)
{
    _player?.Dispose();
    _libVlc?.Dispose();
    base.OnClosed(e);
}
```

You

Zu deinen Frage,...
1. ja sehr gerne würde ich das über LibVLCSharp umsetzen
2. Muss/Soll nicht mehr interaktiv beim Start sein...

Copilot

Copilot

Reasoned for 1s

Setting up WPF app

Setting up project steps

Perfekt – damit haben wir ein sehr klares Zielbild. □

...

Hier ist Teil 4 des Chats

Code (Shell)

```
dotnet new wpf -n VideoBootPlayer
cd VideoBootPlayer
2) NuGet Pakete
LibVLCSharp.WPF
VideoLAN.LibVLC.Windows
3) Tray (NotifyIcon) aktivieren
...
PlayerWindow.xaml.cs
```

Hier ist Teil 5 des Chats

Code (C#)

```
using LibVLCSharp.Shared;
using System.Windows;
using System.Windows.Forms;

namespace VideoBootPlayer;

public partial class PlayerWindow : Window
{
    private readonly LibVLC _libVlc;
    private readonly MediaPlayer _player;

    private Media? _currentMedia;

    public PlayerWindow()
    {
        InitializeComponent();
        Core.Initialize();
        _libVlc = new LibVLC();
        _player = new MediaPlayer(_libVlc);
        VideoView.MediaPlayer = _player;
```

```
}

public void Play(AppSettings s, string videoPath)
{
PlaceOnMonitor(s);
ApplyPlayerSettings(s);

_currentMedia?.Dispose();
_currentMedia = new Media(_libVlc, new Uri(videoPath));

if (s.Loop)
{
_player.EndReached -= OnEndReached;
_player.EndReached += OnEndReached;
}
else
{
_player.EndReached -= OnEndReached;
}

_player.Play(_currentMedia);
Show();
Activate();
}

public void Stop()
{
_player.Stop();
Hide();
}

private void OnEndReached(object? sender, EventArgs e)
{
Dispatcher.Invoke(() =>
{
if (_currentMedia != null) _player.Play(_currentMedia);
});
}

private void PlaceOnMonitor(AppSettings s)
{
var screen = Screen.AllScreens.First(sc =>
sc.DeviceName.Equals(s.MonitorDeviceName, StringComparison.OrdinalIgnoreCase));

var b = screen.Bounds;
Left = b.Left;
Top = b.Top;
Width = b.Width;
Height = b.Height;

if (s.Fullscreen)
{
WindowStyle = WindowStyle.None;
WindowState = WindowState.Maximized;
}
else
{
WindowState = WindowState.Normal;
}
}

private void ApplyPlayerSettings(AppSettings s)
{
_player.Mute = s.Mute;
if (!s.Mute) _player.Volume = Math.Clamp(s.Volume, 0, 100);
}

protected override void OnClosed(EventArgs e)
{
_currentMedia?.Dispose();
```

```
_player.Dispose();  
_libVlc.Dispose();  
base.OnClosed(e);  
}  
}
```

□ Tray Icon + Schnellauswahl Menü

...

□ SettingsWindow (als nächster Schritt)

Das bauen wir so:

Hier ist Teil 6 des Chats

□ UI Elemente:

ListBox der Videos (mit Entfernen / Up/Down optional)

Button: „Video hinzufügen...“ (OpenFileDialog)

Button: „Ordner hinzufügen...“ (FolderBrowserDialog + Rekursiv)

Dropdown: Monitore (anzeigen: DeviceName + Auflösung + Position)

Checkboxen: Fullscreen, Loop, Mute, Autoplay

Volume Slider

Delay Numeric

Button: „Test“ (spielt selected/default)

Button: „Speichern“

Und: Wenn ein Monitor fehlt → oben ein roter Hinweis.

□ Single Instance (damit Autostart nicht doppelt startet)

...

You

zu deinen Fragen,...1. ich finde einen APP-Ordner gut2. ...

Copilot

Copilot

...

Hier ist Teil 7

You

Die Eigenschaft "Min" ist im XML-Namespace "http://schemas.lepo.co/wpfui/2022/xaml" nicht vorhanden. Zeile 43 Position 61.

"Application" ist ein mehrdeutiger Verweis zwischen "System.Windows.Forms.Application" und "System.Windows.Application".

...

Copilot

Copilot

Super – danke, damit ist klar, was los ist. □

...

You

Hier die Fehlerliste... Snowblind-Mod Player

Copilot

Copilot

...

You

jetzt habe ich noch diesen Fehler

Copilot

Copilot

...

You

jetzt hab ich diese Fehler

Copilot

Copilot

I wasn't able to generate a response. Send a new prompt or start a new chat.

...

Hier ist Teil 8

You

Also ich hätte gern die Tabs lieber wie in Windows auf der Linken Seite... mit Leertaste pausieren ...

Copilot

Copilot

... (v3, UX-Wünsche umgesetzt) ...

You

- Im Video Tab beim ändern des Defaults ...

Copilot

Copilot

I wasn't able to generate a response. Send a new prompt or start a new chat.

...

Hier ist Teil 9

You

C + D

Copilot

Copilot

Klar — C + D sind jetzt umgesetzt. ☐

... (v6 Beschreibung) ...

You

Kannst du die Punkte 1 und 2 ...

Copilot

Copilot

... (v7 Snackbar + OSD erweitert) ...

Teil 10

You

Jetzt funktioniert die Steuerung des Videos nicht mehr und die Settings fehlen auch noch

Copilot

Copilot

I wasn't able to generate a response. Send a new prompt or start a new chat.

You

ok kannst du bitte das ganze OSD Zeug wieder raus nehmen ...

Copilot

Copilot

... (Patch PlayerWindow.xaml/.cs) ...

Hier ist der letzte Teil

Code (C#)

```
using LibVLCSharp.Shared;
using System;
using System.Linq;
```

```
using System.Windows;
using System.Windows.Input;

namespace SnowblindModPlayer;

public partial class PlayerWindow : Window
{
    private readonly LibVLC _libVlc;
    private readonly MediaPlayer _player;
    private Media? _media;

    private bool _isFullscreen;

    public PlayerWindow()
    {
        InitializeComponent();

        Core.Initialize();
        _libVlc = new LibVLC();
        _player = new MediaPlayer(_libVlc);
        VideoView.MediaPlayer = _player;

        Hide();
    }

    public void Play(AppSettings settings, string path)
    {
        PlaceOnMonitor(settings);

        ApplyPlayerSettings(settings);

        _media?.Dispose();
        _media = new Media(_libVlc, new Uri(path));

        // Loop optional
        if (settings.Loop)
        {
            _player.EndReached -= OnEndReached;
            _player.EndReached += OnEndReached;
        }
        else
        {
            _player.EndReached -= OnEndReached;
        }

        _player.Play(_media);

        Show();
        Activate();

        // ☐ Fokus-Fix (damit KeyDown zuverlässig funktioniert)
        Focus();
        Keyboard.Focus(this);
    }

    public void Stop()
    {
        _player.Stop();
        Hide();
    }

    private void OnEndReached(object? sender, EventArgs e)
    {
        Dispatcher.Invoke(() =>
        {
            if (_media != null)
                _player.Play(_media);
        });
    }
}
```

```
private void PlaceOnMonitor(AppSettings settings)
{
var screens = System.Windows.Forms.Screen.AllScreens;

var target = screens.FirstOrDefault(s =>
s.DeviceName.Equals(settings.MonitorDeviceName, StringComparison.OrdinalIgnoreCase))
?? screens.First();

var b = target.Bounds;
Left = b.Left;
Top = b.Top;
Width = b.Width;
Height = b.Height;

WindowState = WindowState.Normal;
WindowStyle = WindowStyle.None;
ResizeMode = ResizeMode.NoResize;
Topmost = true;
ShowInTaskbar = false;

_isFullscreen = settings.Fullscreen;
if (_isFullscreen)
WindowState = WindowState.Maximized;
}

private void ApplyPlayerSettings(AppSettings s)
{
_player.Mute = s.Mute;
if (!s.Mute)
_player.Volume = Math.Clamp(s.Volume, 0, 100);
}

private void Window_KeyDown(object sender, KeyEventArgs e)
{
switch (e.Key)
{
case Key.Space:
_player.Pause();
e.Handled = true;
break;

case Key.Left:
SeekBySeconds(-5);
e.Handled = true;
break;

case Key.Right:
SeekBySeconds(5);
e.Handled = true;
break;

case Key.F11:
ToggleFullscreen();
e.Handled = true;
break;

case Key.Escape:
// ESC: zuerst Fullscreen verlassen, wenn schon windowed -> Stop
if (_isFullscreen)
SetFullscreen(false);
else
Stop();

e.Handled = true;
break;
}
}

private void SeekBySeconds(int seconds)
{
```

```

var newTime = _player.Time + seconds * 1000L;
if (newTime < 0)
newTime = 0;

if (_player.Length > 0 && newTime > _player.Length)
newTime = _player.Length;

_player.Time = newTime;
}

private void ToggleFullscreen()
{
SetFullscreen(!_isFullscreen);
}

private void SetFullscreen(bool enable)
{
_isFullscreen = enable;

if (_isFullscreen)
{
WindowStyle = WindowStyle.None;
ResizeMode = ResizeMode.NoResize;
Topmost = true;
ShowInTaskbar = false;
WindowState = WindowState.Maximized;
}
else
{
WindowState = WindowState.Normal;
WindowStyle = WindowStyle.SingleBorderWindow;
ResizeMode = ResizeMode.CanResize;
Topmost = false;
ShowInTaskbar = true;
}

// ☐ Fokus nach Umschalten wieder sicherstellen
Activate();
Focus();
Keyboard.Focus(this);
}

protected override void OnClosed(EventArgs e)
{
_media?.Dispose();
_player.Dispose();
_libVlc.Dispose();
base.OnClosed(e);
}
}

```

Damit ist gar kein OSD mehr drin, aber die Steuerung funktioniert wieder stabil.