NEWS · DECEMBER 17, 2025

**A Christmas Carol for Delphi Developers: “AI Humbug!” … or Maybe Not? **

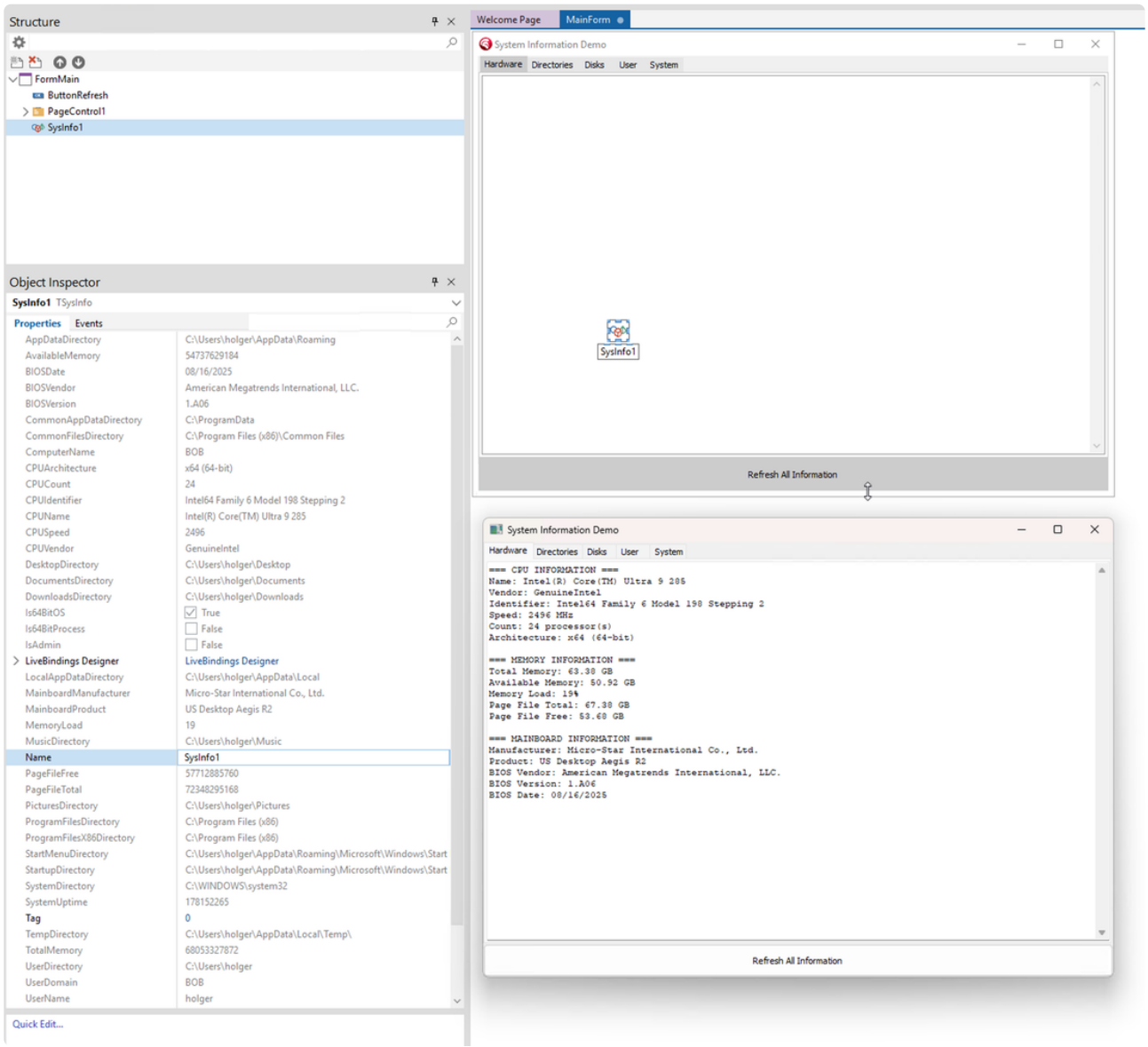
By Dr. Holger Flick

Every December, you can hear it echo across developer forums: *“AI? Bah, humbug! It doesn’t help with real Delphi code!”*

If you’ve spent decades refining your Delphi craft — from Turbo Pascal to today’s VCL and FireMonkey projects — you might share that sentiment. But let me show you why those days are fading faster than a pine tree in January.

Recently, I built a **new Delphi system information component**, `TSysInfo`, in about *one hour*.

No, not because I worked all night — but because **AI helped**.



Screenshot of component in RAD Studio IDE

A Modern Helper for a Classic IDE

`TSystemInfo` is a drop-in VCL component that gives your Delphi application quick access to essential system details — without digging through Windows APIs or typing endless `GetSystemInfo` calls.

The component makes it easy to fetch:

- CPU details (model name, number of cores/threads)
- Memory statistics (total and available)
- Hard drive information (drive names, capacity, and free space)

Think of it as your *one-stop shop for environment diagnostics* — ideal for about dialogs, diagnostics windows, or logging system information for support tickets.

How to Use It — A Simple Example

Once installed and dropped onto a form, you can access all system information directly through its properties or by calling simple methods.

Here's an example:

```
uses
  SysInfo;

procedure TForm1.Button1Click(Sender: TObject);
var
  i: Integer;
begin
  // Retrieve CPU information
  Memo1.Lines.Add('CPU: ' + SysInfo.CPU.Name);
  Memo1.Lines.Add('Cores: ' + SysInfo.CPU.PhysicalCores.ToString);
  Memo1.Lines.Add('Threads: ' + SysInfo.CPU.LogicalCores.ToString);
  Memo1.Lines.Add('');

  // Retrieve memory information
  Memo1.Lines.Add('Total Memory: ' + SysInfo.Memory.TotalGB.ToString + ' GB');
  Memo1.Lines.Add('Available Memory: ' + SysInfo.Memory.AvailableGB.ToString + ' GB');
  Memo1.Lines.Add('');

  // List all hard drives
  Memo1.Lines.Add('Drives:');
  for i := 0 to SysInfo.Drives.Count - 1 do
  begin
    Memo1.Lines.Add(Format('%s - %.2f GB (Free: %.2f GB)',
      [SysInfo.Drives[i].Letter,
       SysInfo.Drives[i].TotalSpaceGB,
       SysInfo.Drives[i].FreeSpaceGB]));
  end;
end;
```

That's it — no WinAPI digging, no juggling of units, and no head-scratching conversions. Just *Intel inside*, memory data, and storage insights — all ready to go.

How AI Helped

Now, here's the interesting part. The concept for `TSystemInfo` took me minutes to outline. But usually, creating a Delphi component involves quite a bit of boilerplate:

- Declaring published properties
- Handling RTTI and registration
- Writing repetitive wrapper code around the Windows API

That's where **AI came in** — it drafted the structure, outlined unit organization, built the initial interface for CPU/Memory/Drives, and handled most of the verbose setup.

I reviewed and adjusted the details, ensuring it matched Delphi conventions and quality.

Together, we got something *finished, tested, and published* in about an hour.

Challenge Accepted?

So, dear fellow Delphi developer, before you utter *“AI? Bah, humbug!”* — try it yourself.

Clone the repo: ⇉https://github.com/holgerflick/delphi.sysinfo

Drop it into your IDE, call `SysInfo`, and see the results.

Ask yourself honestly — could you have built it faster without a digital helper?

Final Thought — The Spirit of Future Code

AI doesn’t take away what makes you a great Delphi developer — it **frees** you to focus on the parts that matter. The design, the logic, the polish — not the scaffolding.

Maybe this Christmas, it’s time to hang up the “humbug” and start saying, *“AI, compile me something beautiful.”* <…

INDEPENDENT · AD-FREE · READER-SUPPORTED

Free to read. Not free to make.

Every tutorial, article, and line of example code on Holger's Code is published without a paywall and without ads. This page is about what keeps it that way.

Professional work, published free

The content here is held to the same standard as professional client work: every tutorial is built as a real project, tested against current versions, documented step by step, and revisited when the frameworks move on. A single in-depth tutorial routinely takes days — and that's before the bills behind it: servers, storage, build infrastructure, licenses, and the hardware everything is verified on.

If an article or tutorial here has saved you an afternoon of debugging, taught you a technique, or helped you ship, it created real value for you. Supporting it isn't charity — it's paying for professional work you found useful, at whatever price you choose. And if you can't, keep reading anyway. That's what the free part means.

Ways to support

KO-FI

Buy me a coffee

A one-time tip — no account, no subscription, any amount.

BOOKS & COURSES

Buy the paid work

The paid tier of everything published free here — and you get something lasting in return.

SPREAD THE WORD

Share an article

Send a tutorial to a colleague, cite an article, subscribe to the RSS feed.

Nothing here goes behind a paywall, whether you chip in or not. But if something on this site earned its keep today, this is where to say so.

Support on Ko-fi —