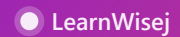


AI Coding Exercise

Build this module's VisualOperationsStudio sample with ChatGPT or Claude from a ready-made prompt that points the model at this module's specification, then review, run and extend what it gives you.



You are not going to type this sample. You are going to direct it.

Almost nobody writes Wisej.NET code from a blank file any more — an assistant writes the first draft and the developer decides whether it is right. That only works if you know the material, which is what the rest of this module gave you. Here you hand that knowledge to ChatGPT or Claude, get this module's sample back as a working Visual Studio solution, and review it the way you would review a colleague's pull request.

What you are asking for

Add a TopologyImageRenderer to VisualOperationsStudio that turns the topology model into a PNG without touching a Wisej control. Give it a Render method taking the scene and an output size, create the Bitmap, obtain its Graphics with Graphics.FromImage, draw nodes, edges, labels and a legend with pens, brushes and a GraphicsPath, and dispose everything it owns while leaving the caller's objects alone. Resolve its Font through a ResolveFont method that tries an application-supplied family and falls back to a known-present one instead of assuming a system font exists, save to a MemoryStream as PNG and hand the bytes to the page for download. Then check the project references to confirm you are not adding System.Drawing.Common alongside Wisej's managed implementation, run or stage the renderer on Windows and in a Linux container, and write up how the same code behaves on iOS, Android and macOS Hybrid targets.

The module asks for these deliverables, and the prompt below asks the assistant for exactly them:

- ✓ TopologyImageRenderer with no dependency on a Wisej control, rendering through Graphics.FromImage into a Bitmap
- ✓ ResolveFont method with an explicit fallback chain instead of an assumed system font
- ✓ PNG written to a MemoryStream and downloaded from the page, with every owned graphics object disposed
- ✓ Reference check confirming Managed.System.Drawing is in use and System.Drawing.Common is not being added alongside it
- ✓ CrossPlatform.md with Windows plus Linux container evidence and the iOS, Android and macOS implications

Why it is shaped this way

Everything VisualOperationsStudio has drawn so far belongs to a live session: the gauge repaints because a control asked for it, and the Canvas topology from Module 5 exists only as commands the browser has already executed. Operations now wants that same topology in a nightly incident e-mail and attached to a ticket, which means an image with no control behind it. At the same time the application is moving into a Linux container, so the drawing code has to keep working on a host with no GDI+ and no libgdiplus.

These are the terms the assistant has to get right. If what comes back uses them loosely, it has not understood the module:

Term	What it means here
System.Drawing.Common	Microsoft's GDI+ wrapper, Windows-specific from .NET 6; the Unix switch went in .NET 7, so it is no option for Linux, macOS, iOS or Android.
Managed.System.Drawing	The NuGet package id, reversed because the System.* package root is reserved; Wisej.NET 4 already references it for you.
System.Drawing.Managed.dll	The assembly that package delivers — the managed implementation itself, seen in a build log rather than in your source.
The System.Drawing namespace	Unchanged: Bitmap, Graphics, Pen and Font keep their names, so existing drawing code compiles without a rewrite.
Managed graphics engine	A pure managed implementation on a customized, integrated ImageSharp engine; on modern .NET it avoids GDI+, libgdiplus and native calls.
Graphics.FromImage	The off-screen entry point: a drawing surface over a Bitmap you own, with no control, no Paint event and no session.
PrivateFontCollection	Loads font files the application ships, so output depends on what you deployed, not on what happens to be installed on the host.

Build it — pick your assistant

Both buttons open a new tab with the **same ready-made prompt** already filled in. The prompt points the model at this module's specification — a single Markdown file holding the lesson, the lab brief and the deliverables — so it works from the real material instead of guessing what Module 6 is about. You can open [module-06.md](#) yourself to see exactly what it will read.



Use ChatGPT

Opens [chatgpt.com](#) with this module's prompt loaded. Works with GPT-5 or Codex; any plan that can browse the web will fetch the spec.



Use Claude

Opens [claude.ai](#) with the same prompt loaded. Claude Code users can paste it into a terminal in an empty folder instead.



Send it as-is and you get a project, not a snippet

The prompt ends by asking for the whole solution as a single `.zip` with build-and-run steps. Download it, open the solution in Visual Studio, press F5 and VisualOperationsStudio comes up on `localhost`. You need the .NET SDK and the `Wisej-4` NuGet package; everything else the prompt asks the assistant to write itself.

The prompt, in full — read it, then edit it if you want a different sample

I am working through Module 6 – "System.Drawing.Managed Across Windows, Linux, macOS, iOS and Android" – of the LearnWisej Drawing & Painting in Wisej.NET course, and I want to build the module sample with you.

Start by reading the module specification. It is a single Markdown file holding the full lesson, the lab brief and the deliverables:

<https://learnwisej.com/assets/courses/drawing-and-painting/specs/module-06.md>

Then build a complete Wisej.NET 4 project I can open in Visual Studio:

1. Project "VisualOperationsStudio", C#, net8.0 or later, NuGet package Wisej-4 4.1.x, Kestrel host calling `app.UseWisej()`.
2. Build what the specification's lab asks for: Add a `TopologyImageRenderer` to `VisualOperationsStudio` that turns the topology model into a PNG without touching a Wisej control. Give it a `Render` method taking the scene and an output size, create the `Bitmap`, obtain its `Graphics` with `Graphics.FromImage`, draw nodes, edges, labels and a legend with pens, brushes and a `GraphicsPath`, and dispose everything it owns while leaving the caller's objects alone. Resolve its `Font` through a `ResolveFont` method that tries an application-supplied family and falls back to a known-present one instead of assuming a system font exists, save to a `MemoryStream` as PNG and hand the bytes to the page for download. Then check the project references to confirm you are not adding `System.Drawing.Common` alongside Wisej's managed implementation, run or stage the renderer on Windows and in a Linux container, and write up how the same code behaves on iOS, Android and macOS Hybrid targets.
3. Produce these deliverables, and tell me where each one lives in the solution:
 - `TopologyImageRenderer` with no dependency on a Wisej control, rendering through `Graphics.FromImage` into a `Bitmap`
 - `ResolveFont` method with an explicit fallback chain instead of an assumed system font
 - PNG written to a `MemoryStream` and downloaded from the page, with every owned graphics object disposed
 - Reference check confirming `Managed.System.Drawing` is in use and `System.Drawing.Common` is not being added alongside it
 - `CrossPlatform.md` with Windows plus Linux container evidence and the iOS, Android and macOS implications
4. Follow the specification wherever it names a Wisej.NET class, property or event – use exactly that one. Where it leaves a choice open, tell me which API you picked and why.
5. Keep the server authoritative: state and validation live in .NET, and the browser renders and reports. Never move a decision into JavaScript because it is easier there.
6. Make the result demonstrable: I should be able to run it and see the behaviour this module is about, including what happens when something goes wrong.

Before you write any code, restate in your own words what you are going to build and where each piece will live, so I can correct the design while it is still cheap.

When the code is done, package the whole solution as a single .zip I can download, and give me the exact steps to build and run it.

Finally, answer these questions from the specification, using your own implementation as the evidence:

- A colleague says the renderer proves the app is cross-platform because the code compiles on Linux. What would you actually have to observe in the container before you accepted that claim, and which part of the output is most likely to differ?
- Your renderer disposes the `Bitmap` it created. Why would disposing the `Graphics` inside a

control's Paint handler be a bug, and what rule covers both cases in one sentence?

- Someone proposes replacing this renderer with SkiaSharp because it benchmarks faster. What would you need to know about the workload before agreeing, and what does the team lose by leaving the System.Drawing API behind?

Ask me about anything that is ambiguous before you start.

Then review what came back

Code that compiles is not yet code that is right. The interesting questions are design questions: is this the Wisej.NET control the job actually calls for, does it follow the lifecycle the framework gives you, is the authoritative state on the server rather than parked in the browser because that was convenient? Read the result as a reviewer, not as a customer.

This module gives you the checklist. Answer each one against the code you just received:

- ✓ A colleague says the renderer proves the app is cross-platform because the code compiles on Linux. What would you actually have to observe in the container before you accepted that claim, and which part of the output is most likely to differ?
- ✓ Your renderer disposes the Bitmap it created. Why would disposing the Graphics inside a control's Paint handler be a bug, and what rule covers both cases in one sentence?
- ✓ Someone proposes replacing this renderer with SkiaSharp because it benchmarks faster. What would you need to know about the workload before agreeing, and what does the team lose by leaving the System.Drawing API behind?

Anything you had to correct is worth writing down. A short note of the conventions and decisions you settled on, pasted into the first message of your next session, is worth more than any single clever prompt.



No reference sample for this course yet — yours is the first

The [LearnWisej samples repository](#) carries worked builds for most courses, but not yet for this one. That means there is nothing to compare against and no shortcut: the specification and your own review are the whole safety net. Keep what you build — and if it turns out well, we would like to see it.

Three prompts to take it further

Point your assistant at the project you just built and try these. Each one is a review-and-extend exercise, not a rewrite.

- ✓ **Add the second case.** *"Extend this VisualOperationsStudio project with a second case that follows the same pattern as the first — another entity, screen or control of the same shape, wired the same way. Factor out what is genuinely shared into a base class or service, but only where it earns its place; two similar things do not always want an abstraction. Show me the diff and justify each thing you extracted."*
- ✓ **Prove it actually works.** *"Go through this project deliverable by deliverable against the module specification at <https://learnwisej.com/assets/courses/drawing-and-painting/specs/module-06.md> and show me the evidence that each one is genuinely done — the code path, what to click, and*

what I should see. Where a deliverable is only partly met, say so plainly instead of arguing it through. Then add whatever is missing."

- ✓ **Review it like a pull request.** *"Walk me through this VisualOperationsStudio code as if you were reviewing a colleague's pull request and did not write it. Where does state live, what happens end to end on one user action, what would you send back for changes, and which parts would you not be comfortable putting in front of a customer? Be specific and quote the lines you mean — do not summarise the design back to me."*



Carry it into the lab

The hands-on lab that closes this module asks for the same deliverables you just had the assistant produce. Use what you have here as the shape of your answer and the evidence behind it — not as the answer itself. The lab is graded on your reasoning, and that part has to be yours.