

CREATING SOFTWARE

Building non-trivial Windows Applications

How do we make non-trivial software?



1. **UNDERSTAND** THE
REQUIREMENTS



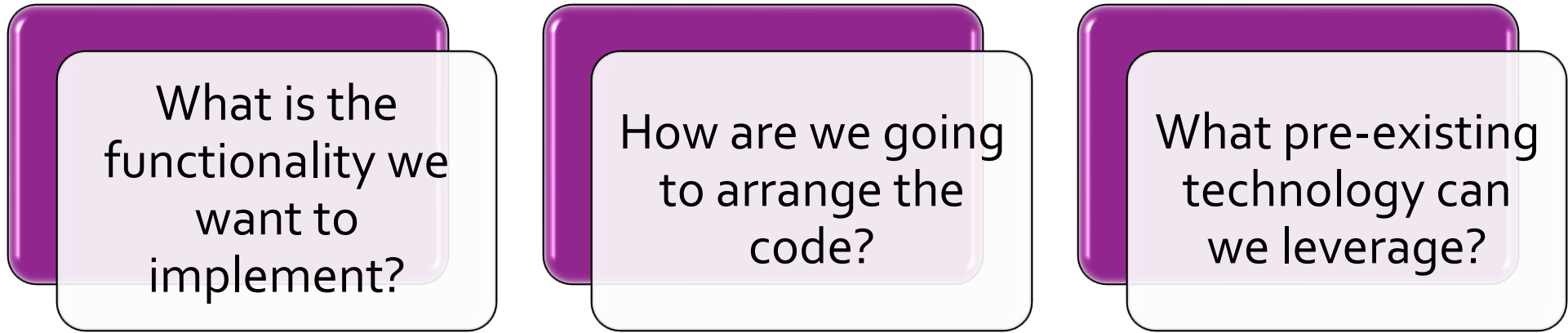
2. **PLAN** WHAT WE
ARE GOING TO DO



3. **IMPLEMENT** THE
CODE



4. **VALIDATE** THE
FUNCTIONALITY



What is the
functionality we
want to
implement?

How are we going
to arrange the
code?

What pre-existing
technology can
we leverage?

Three Questions

Relationship

Structure and arrangement of code should reflect the functionality

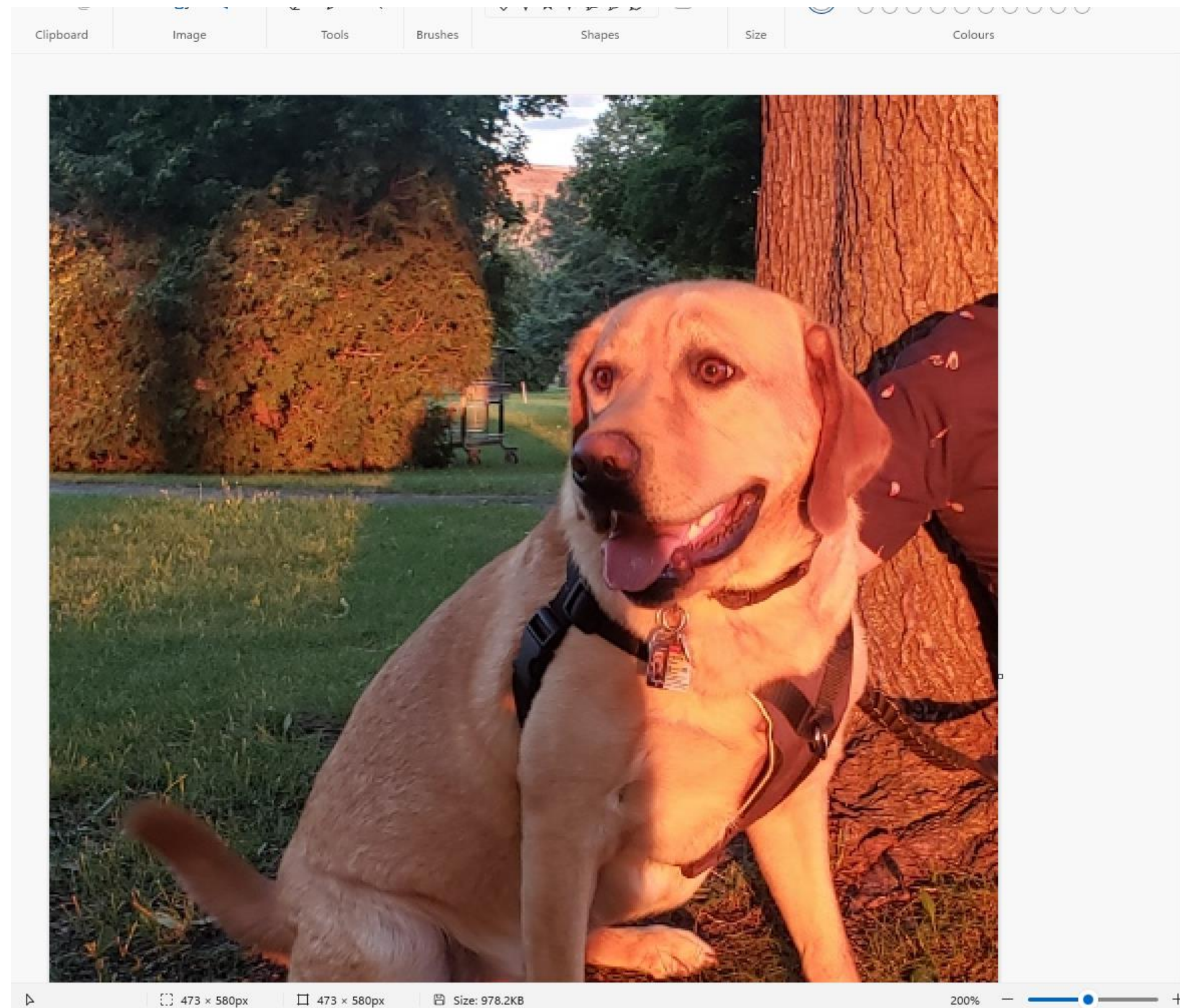
- Make the code easier to read, understand, and manage

Knowing what the software does is critical to making choices

- For example which technology or libraries to use

The choice of library and technology feeds back into process

- Affects what is feasible to do in a reasonable amount of time



**CONSIDER
MSPAINT**

Basics



Read and write image data from file



Display image data in a window



Allow some editing

What can it do in more detail

Draw lines, shapes (outlined or filled), images on a canvas

Open and save files, import images

Select a region and cut/copy/paste

Update the canvas

Flip, rotate, resize, crop regions

Zoom in / out

Undo / redo

Move/Resize/Minimize/Maximize/Close window

Major Areas

Files - open, save, check if modified, recent files

Keyboard and mouse

Window and UI – title, menu, toolbar, context, and shortcuts

Image Edits Selection - skew, crop, flip, rotate, resize

Clipboard - cut, copy, paste, import

Drawing Tools – shapes, lines, fill, erase,

Properties of Drawing Tools – color, fill, brush

What are the UI components?

- Window border
- Title
- Menu
- Toolbar
- Status bar
- Border
- Shadow
- Minimize/Maximize/Close

Drawing Tools

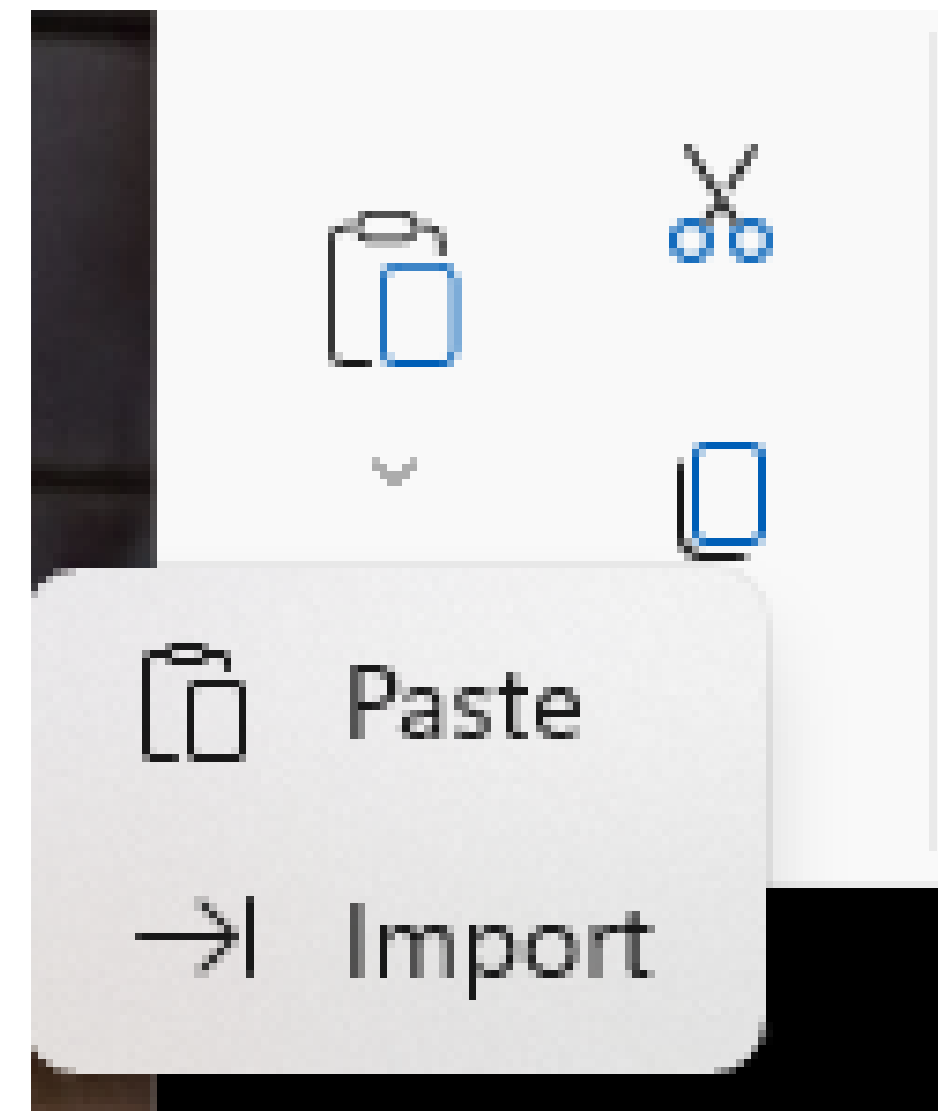
- Write text
- Draw brush
- Make shape (filled or not)
- Select and edit region

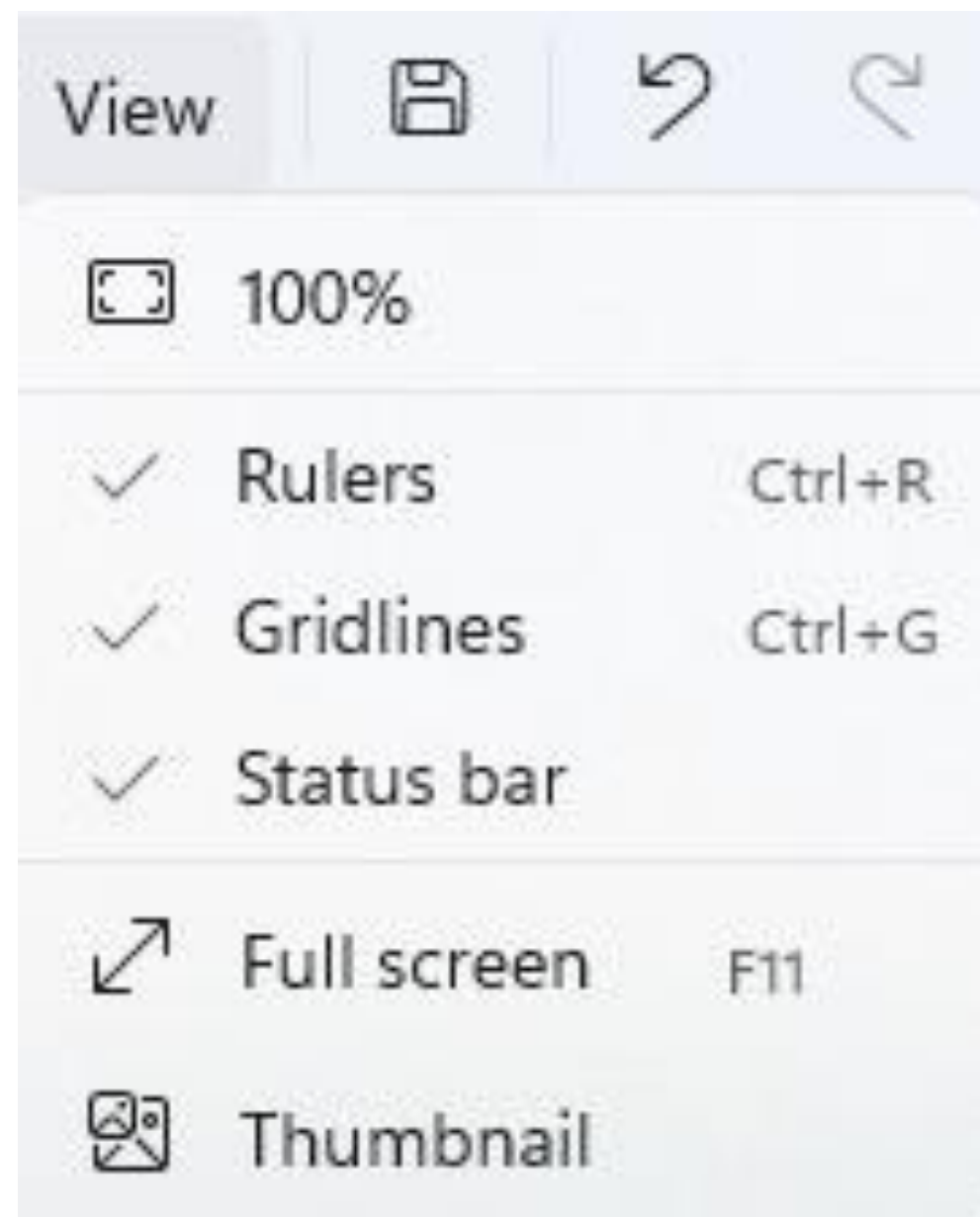
Standardized Stuff

- Color picker
- Font chooser
- Open file dialog
- Save file dialog
- Page set-up
- Print

Clipboard

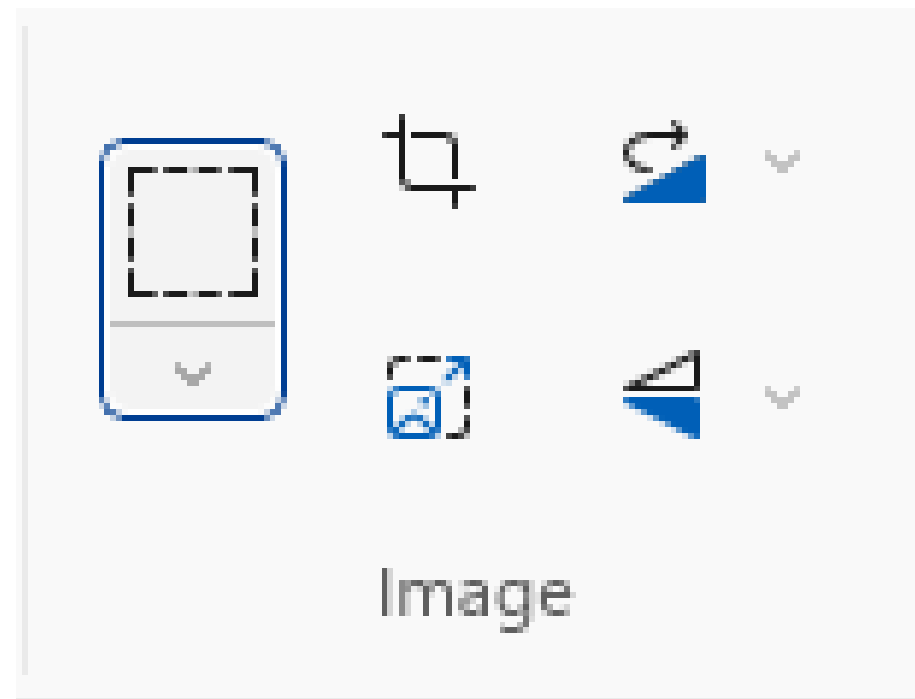
- Cut
- Copy
- Paste
- Import



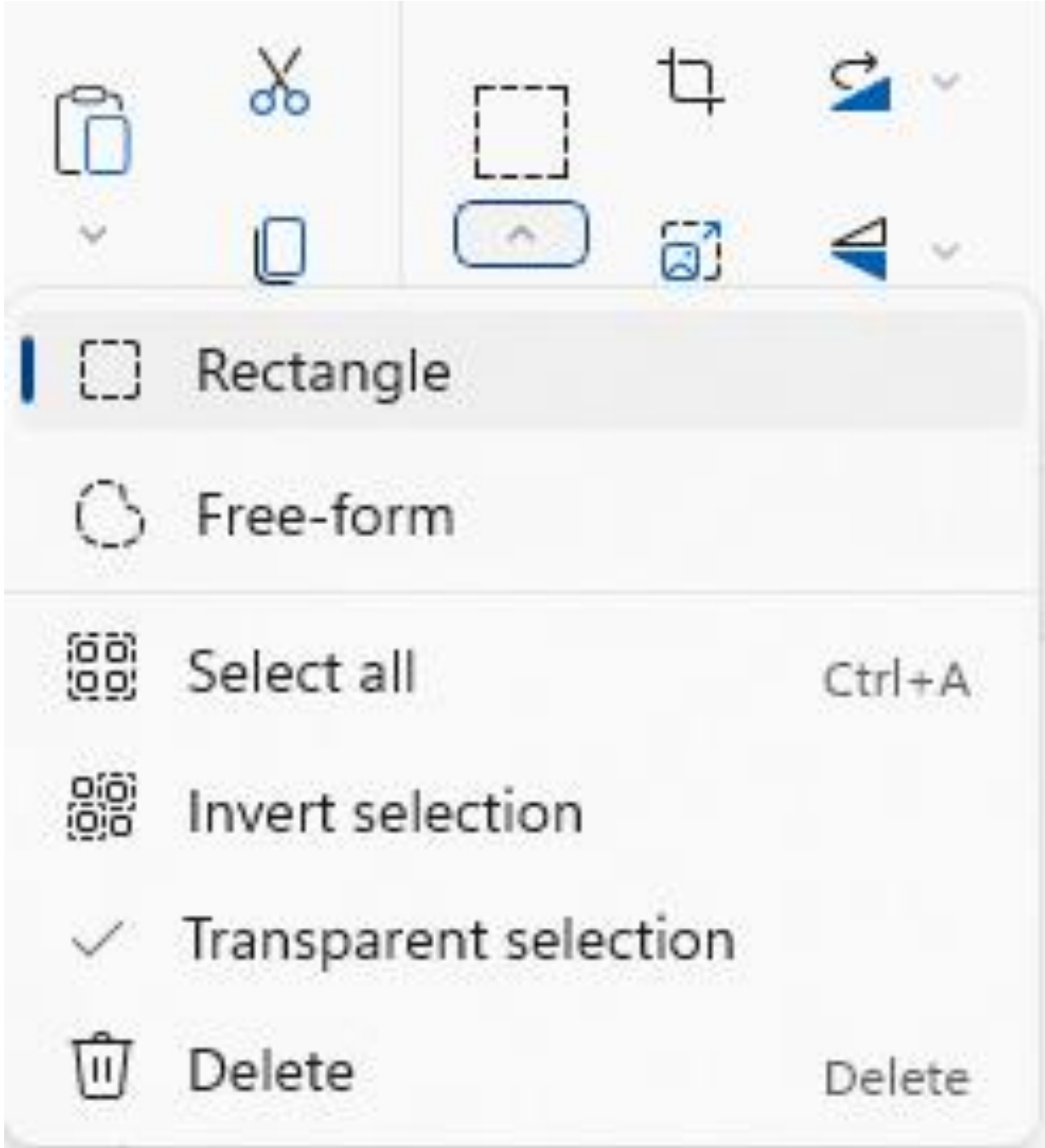


VIEW OPTIONS

Image and Selection Tools



- Select
- Crop
- Resize
- Rotate
- Flip

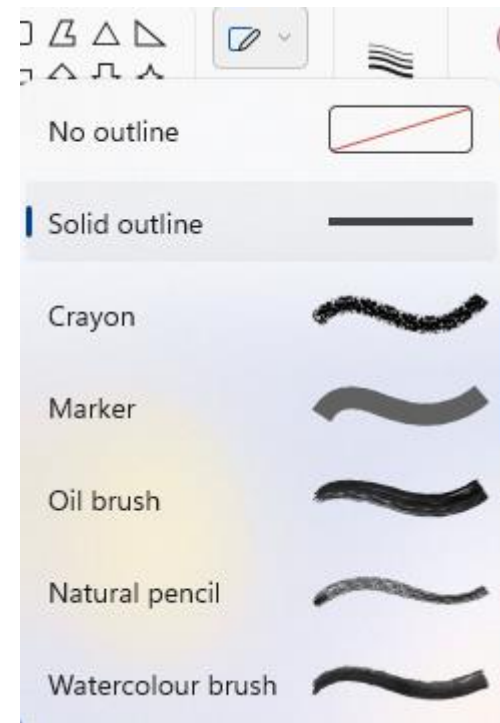


Selection is Complicated

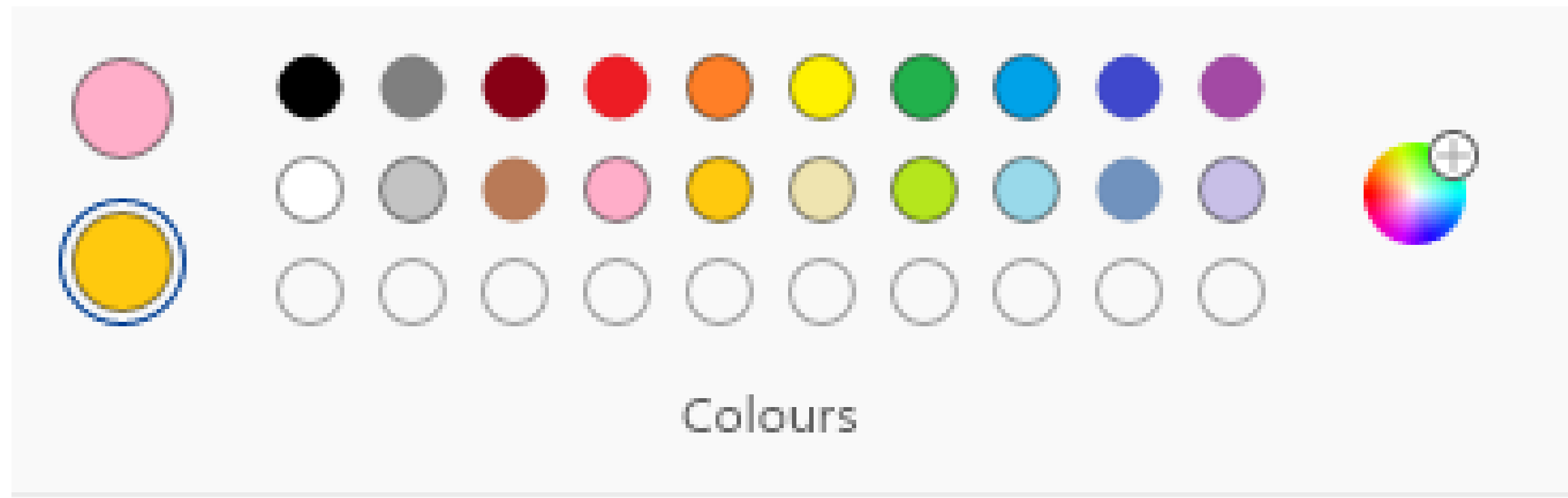
- Two types
- Selection can be inverted
- Transparent

Brush Types

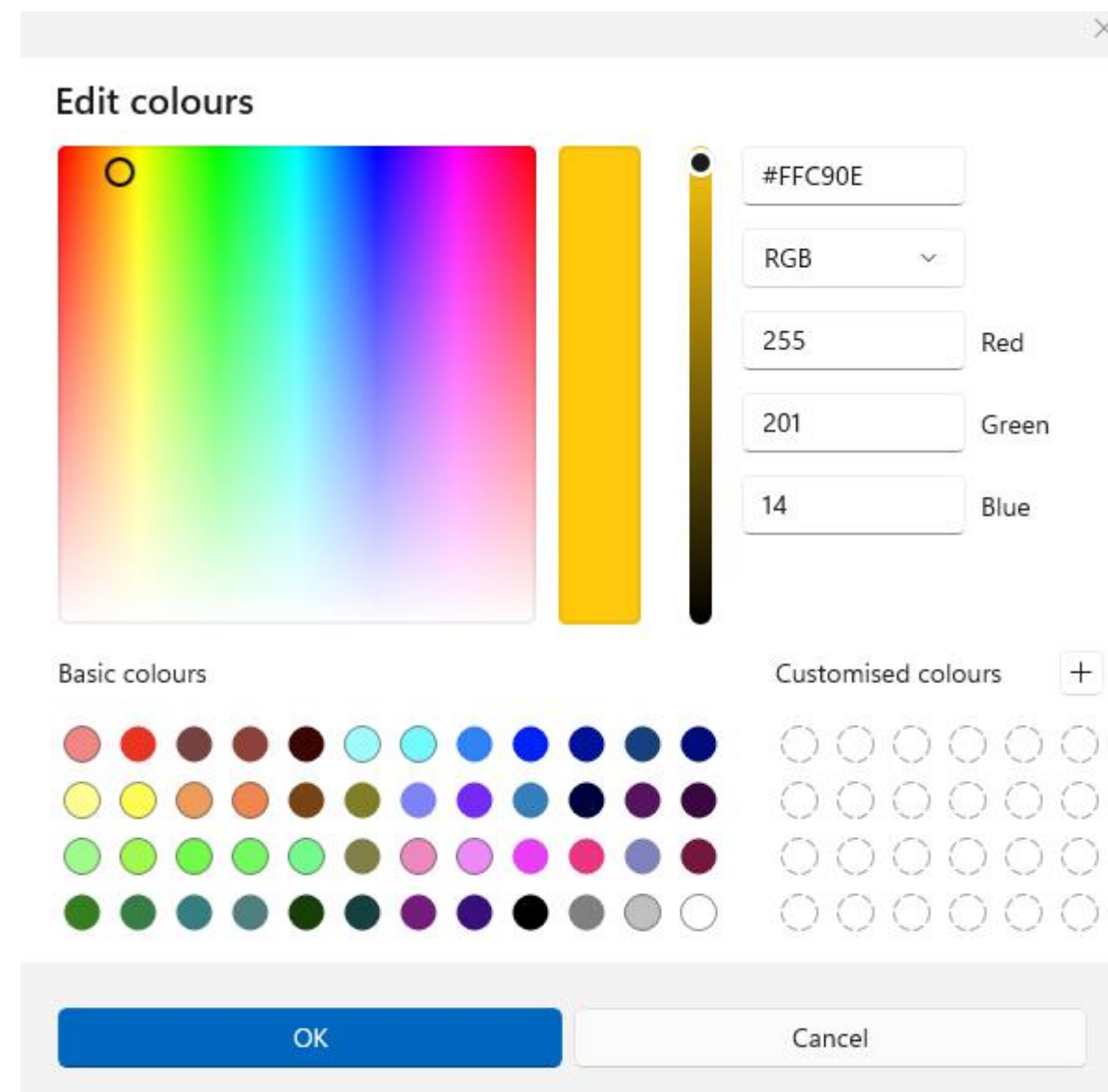
- Shape Fill Brush
- Outline Fill Brush

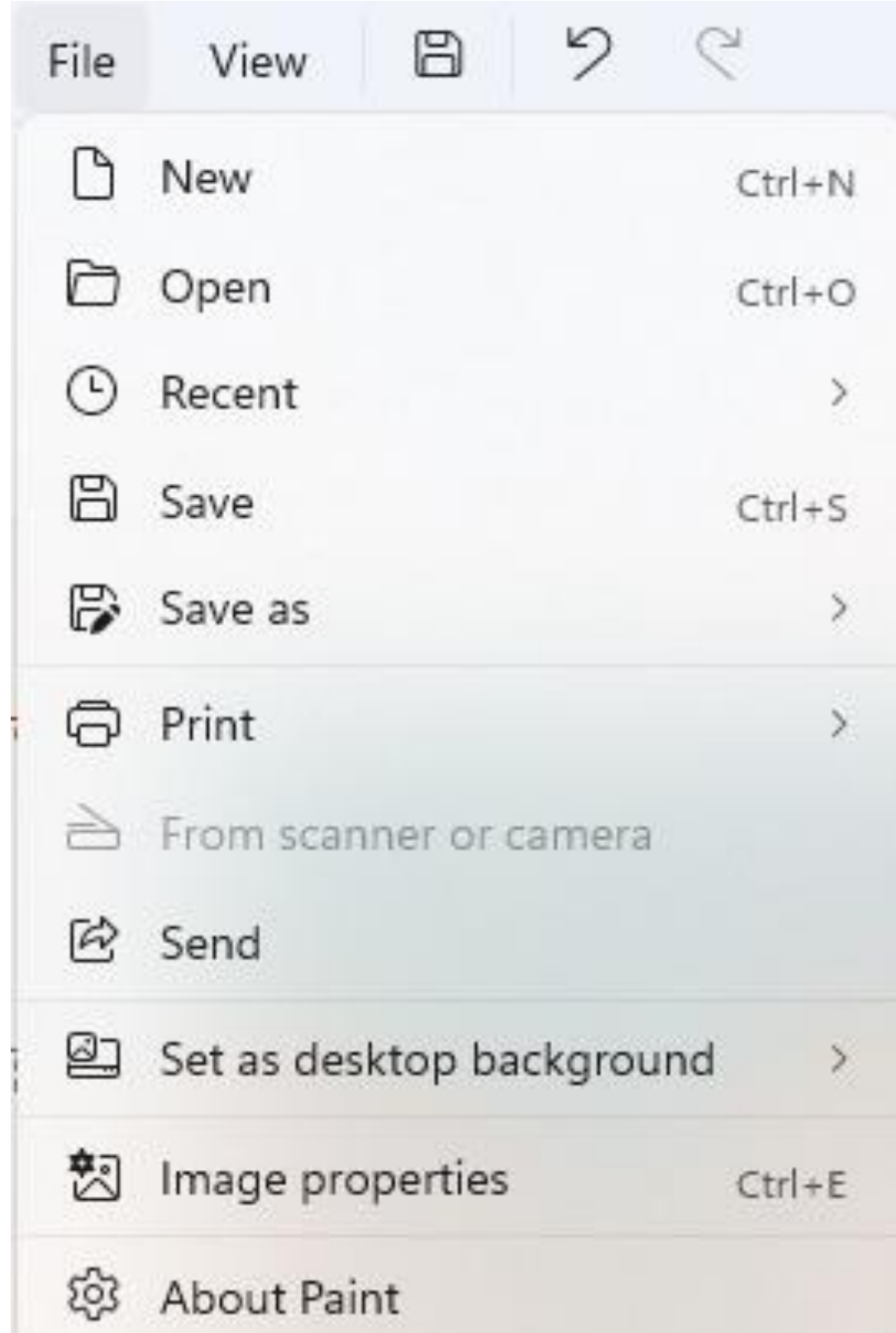


Colours – Pen / Brush (Foreground) and Eraser (Background)



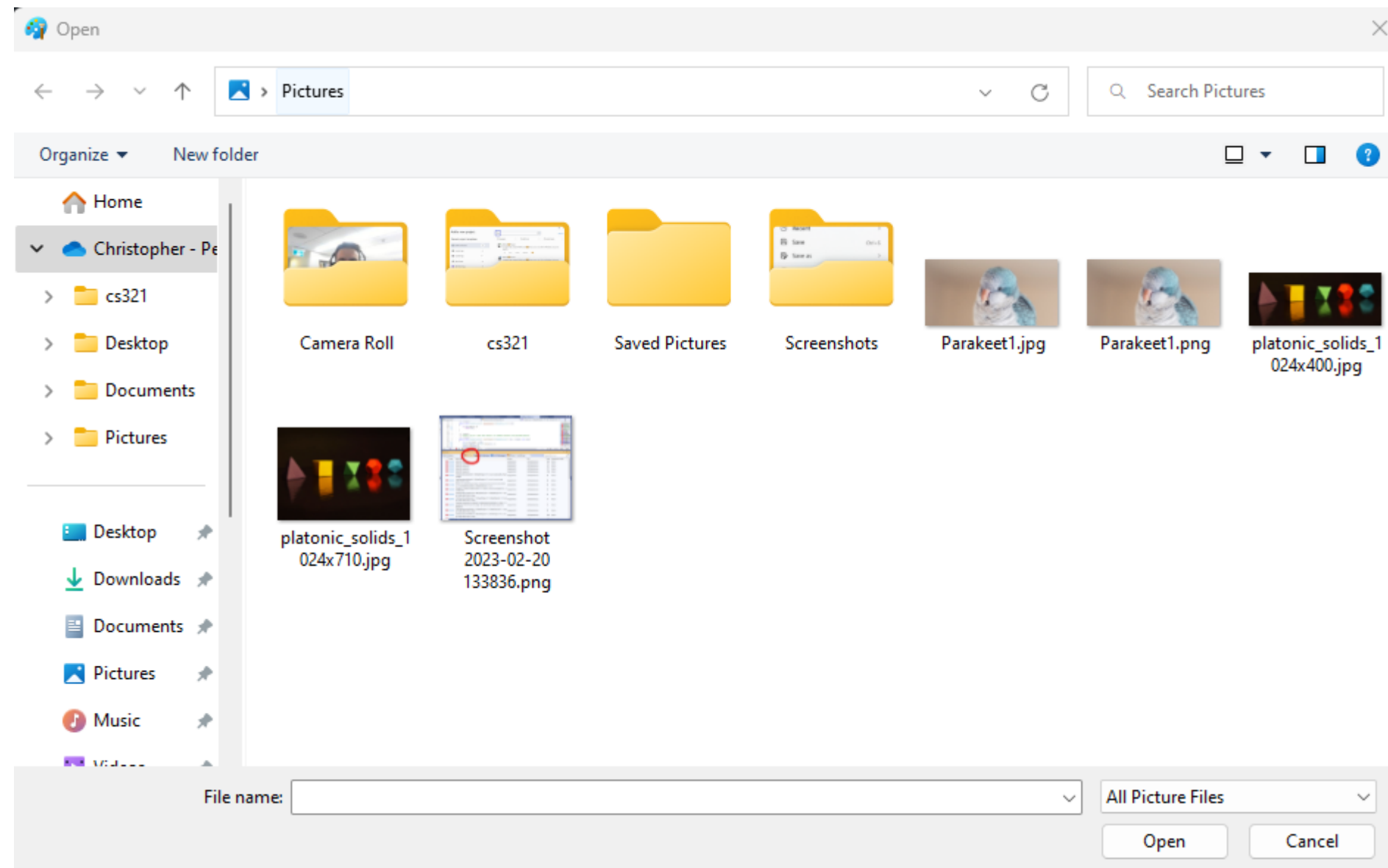
Just the Colour Chooser is a lot of work!

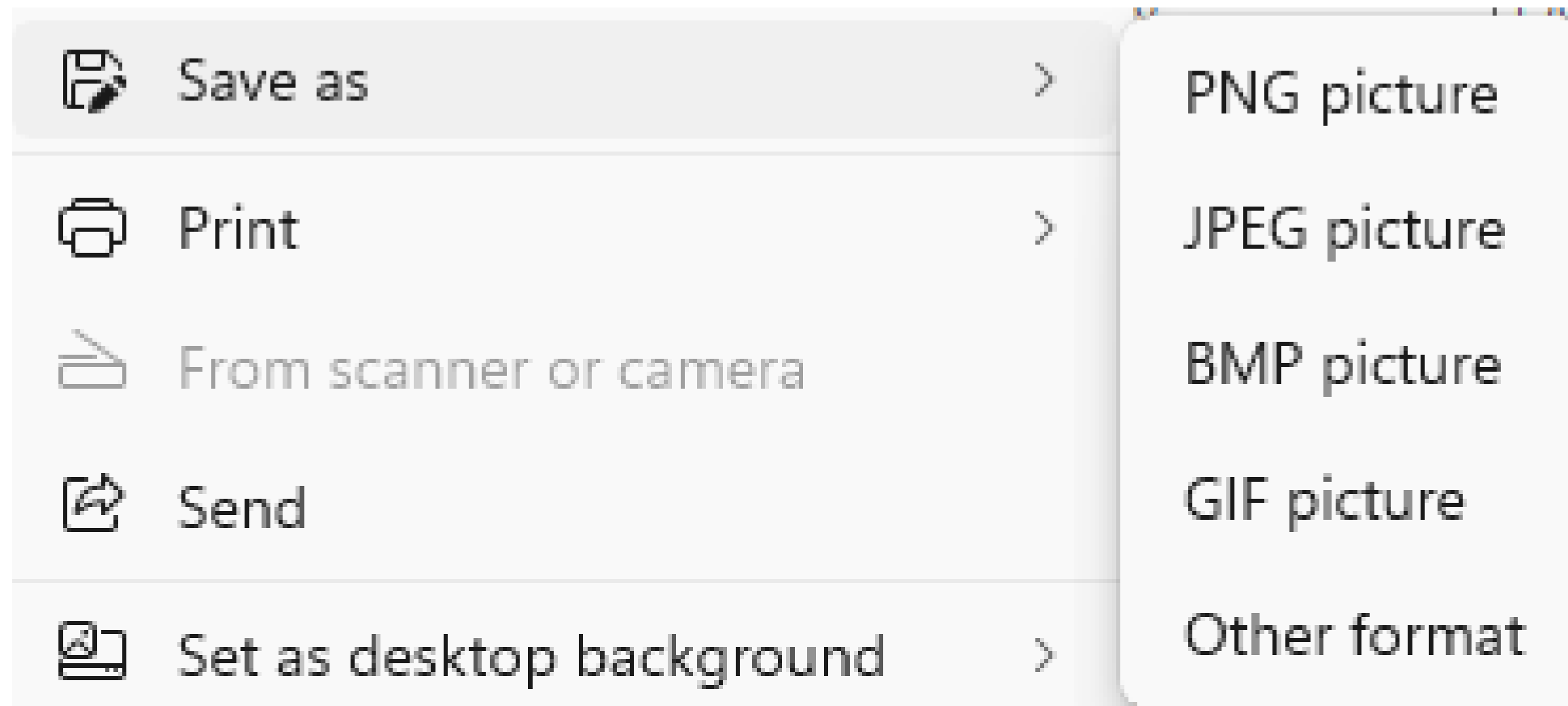




FILE MENU

Open File Dialog



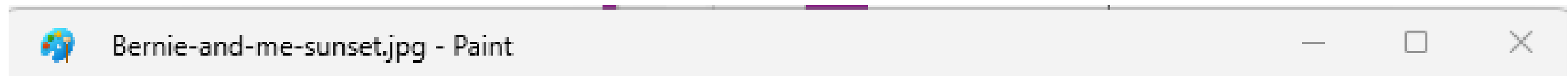


SAVE AS OPTIONS

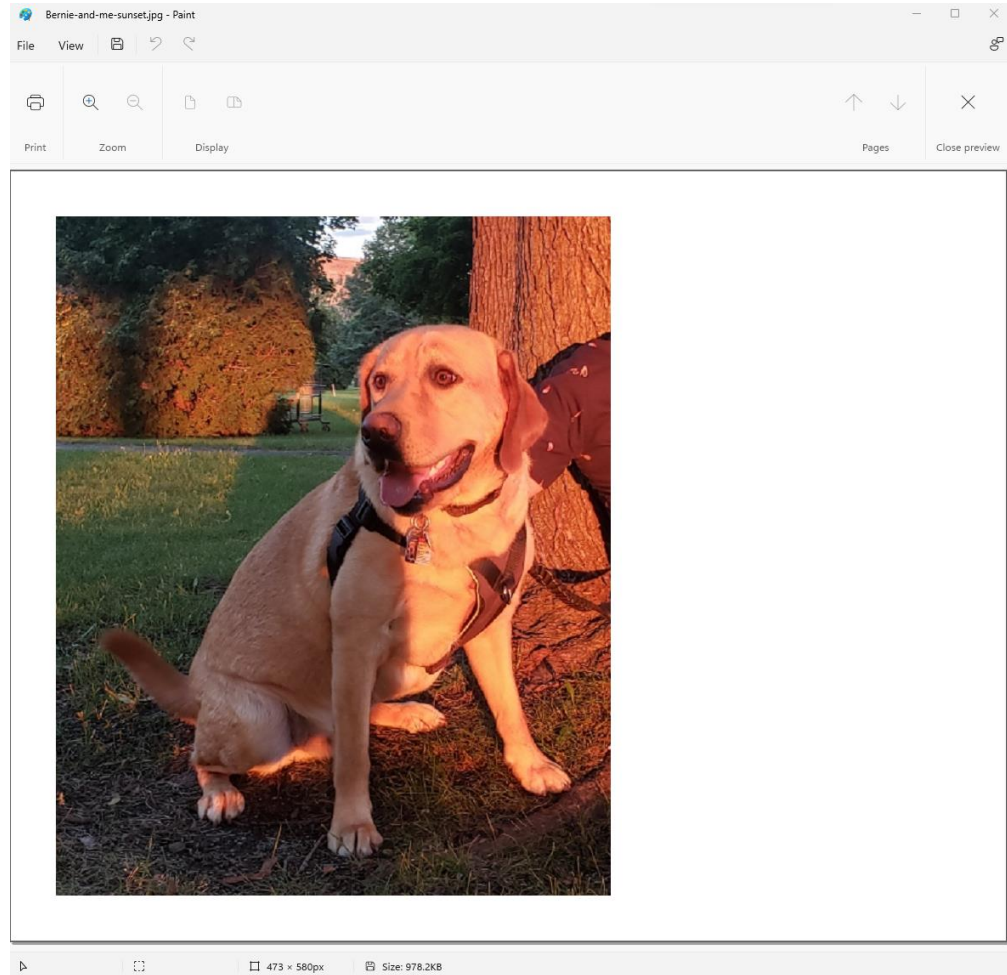
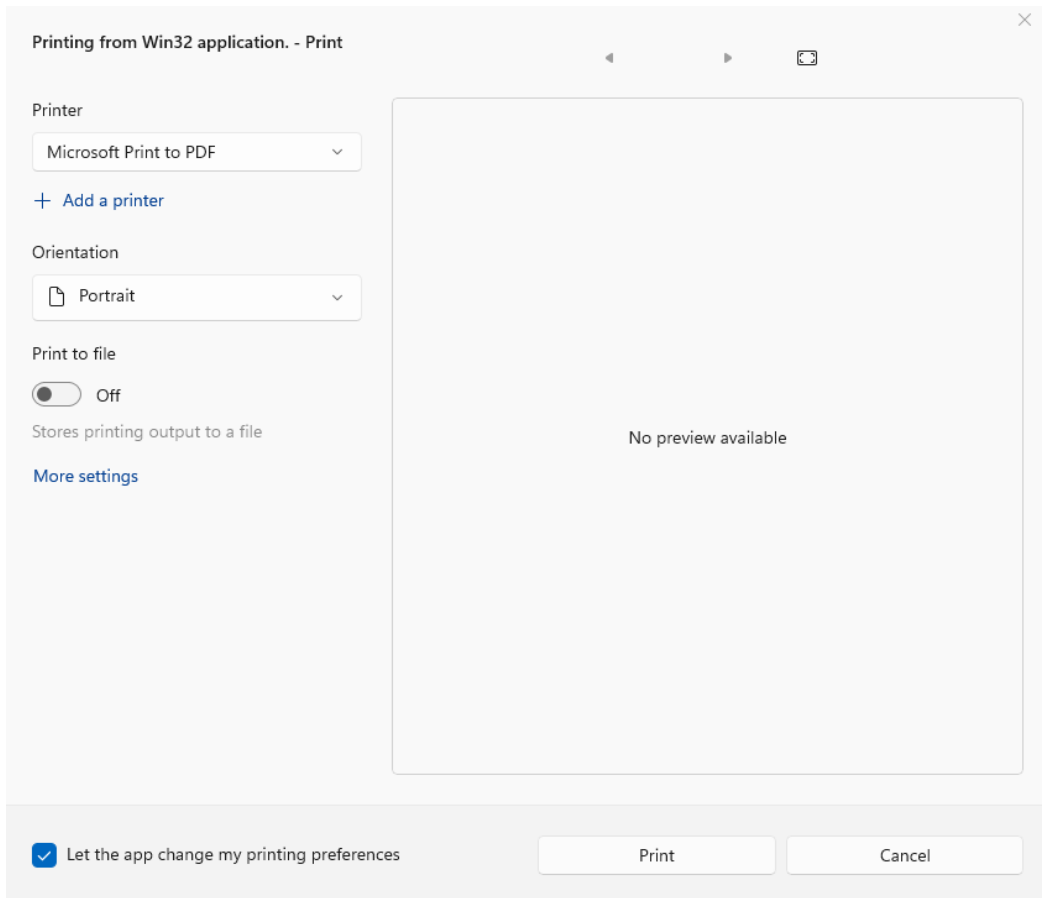
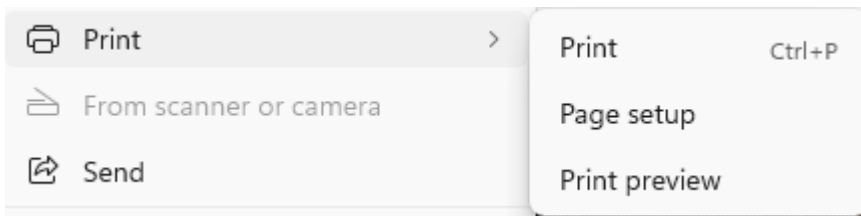
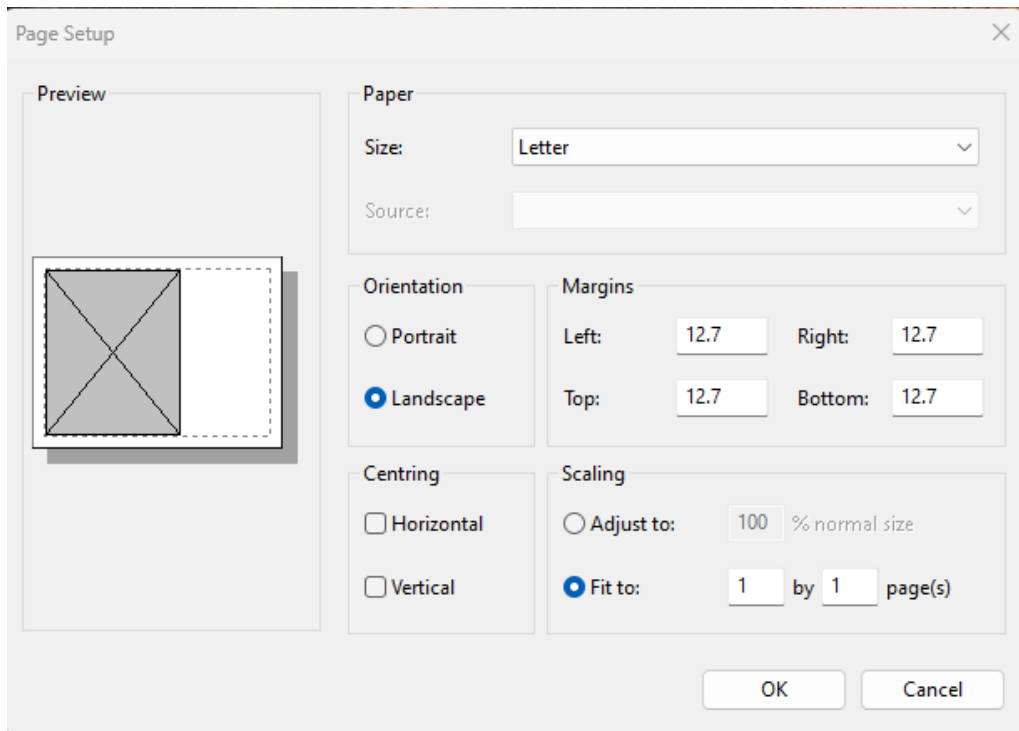
Status Bar



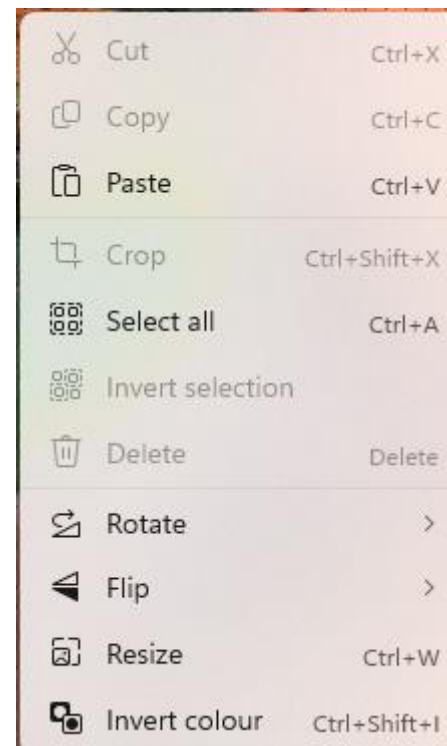
Title Bar



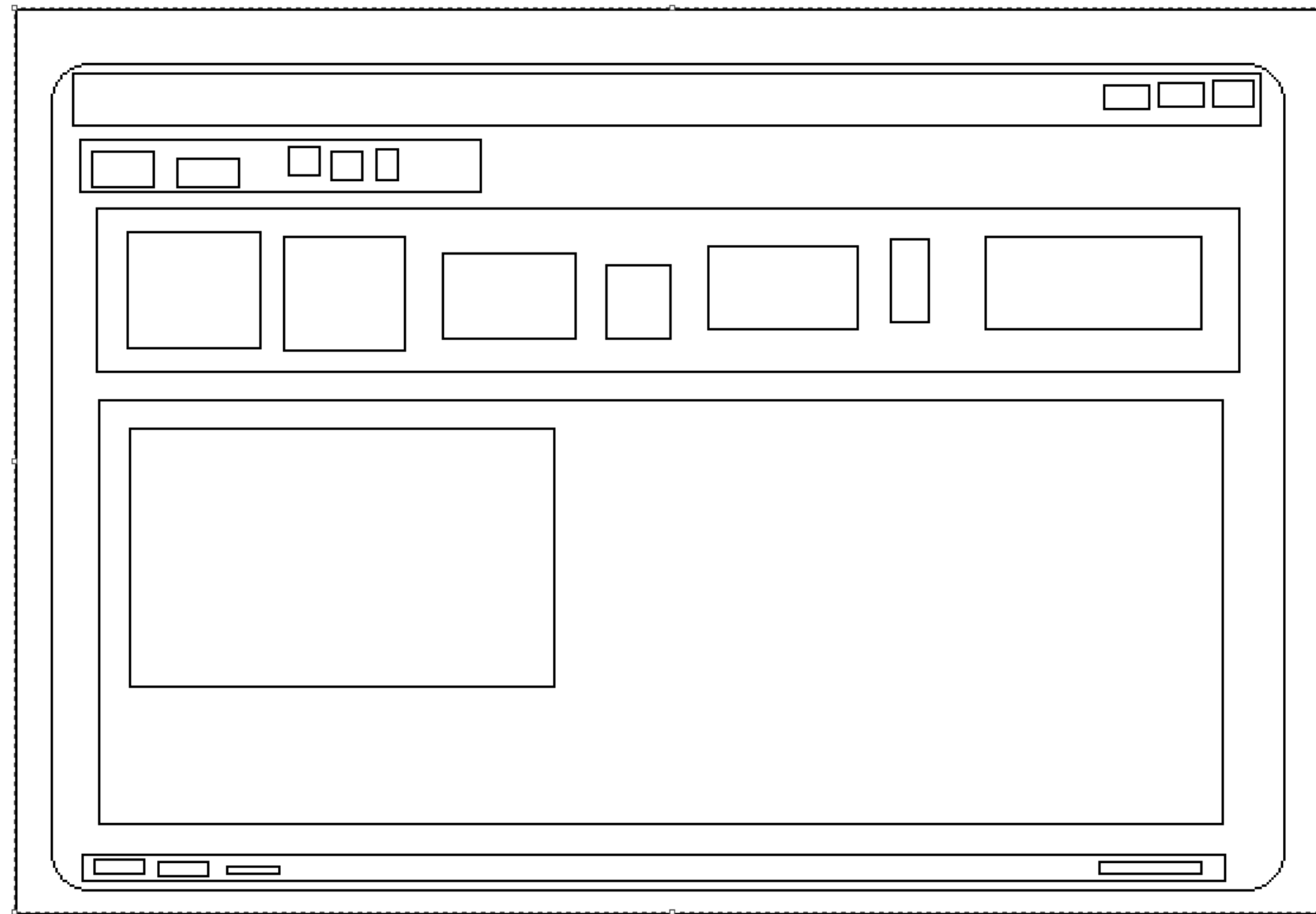
Print Dialog



Context Menu



The UI – Lots of Rectangles (Controls)





WOW!

That's more than it seems

Describing Requirements

- Not an exact science
- A complete specification would be source code
- Some ambiguity is unavoidable
- Different ways to do it
- Hard but important
- Be flexible and lightweight

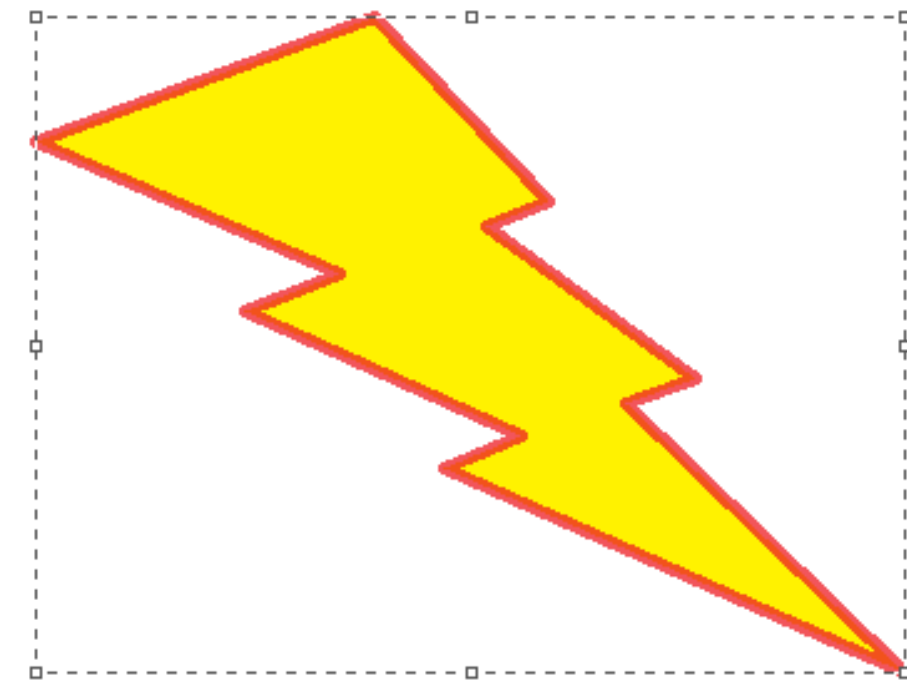
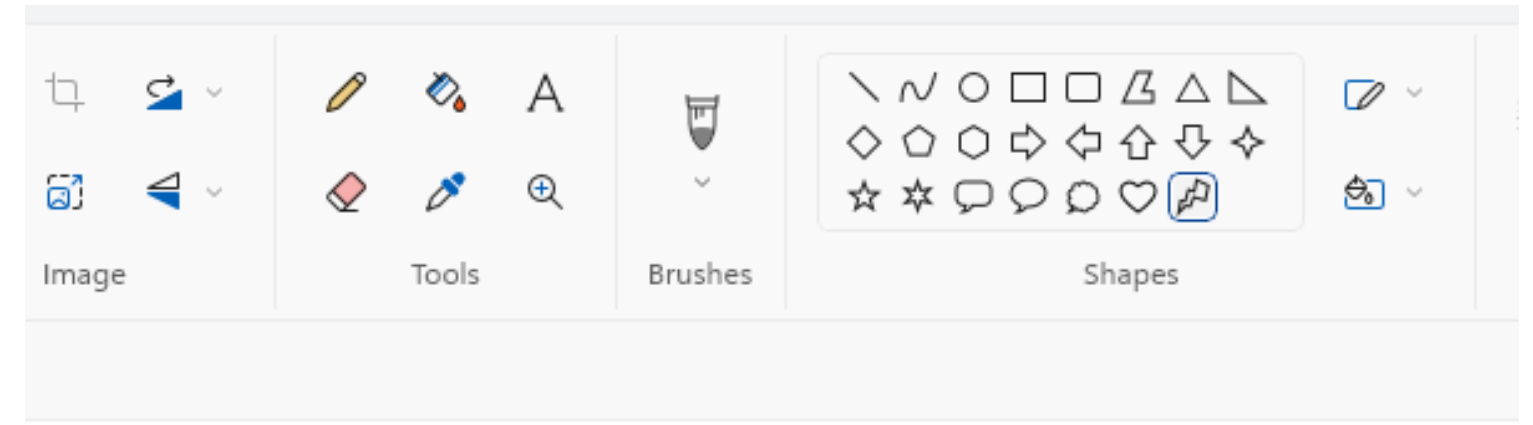


Describing a Feature

- How is it enabled?
- Is it modal?
- Does it change how keyboard or mouse input is handled?
- Are there multiple stages?
- Can it be cancelled?
- Can it be undone?
- What is the impact on the data?

Example

- The Windows Paint program: creating a shape
- How would you describe it this feature?



Shape Drawing Operation

- When the active tool is a “shape”
- Left-clicking on the canvas with the mouse will start the drawing process for a shape
- While the left mouse button is held down a shape will be drawn to fit a bounding box
- The box is defined by the original location of the mouse and the current position of the mouse
- Moving the mouse will update the drawn shape
- When the left mouse button is released the shape will be selected with a rectangular selection
- Selection tools are enabled (e.g., flip, rotate, move, resize) for the shape but won't affect background
- Pressing escape or delete will discard shape, and cancel the operation
- Clicking anywhere in the application will complete the drawing operation and cancel selection

But wait ...
there's more

The shape is drawn is using current pen and brush settings

- border pen width
- border brush style and color
- fill brush style

These can be changed while the shape is selected and before the shape is completed.

That's a lot!

Complexity Management

Primarily challenge of writing software is complexity management.

Problem has little to do with hardware

We are limited primarily by creativity and human ability to process complex information

Software could do virtually anything given enough

- time
- brainpower



Five Steps of Complexity Management

1

Eliminate
redundancy

2

Maximize code
reuse

3

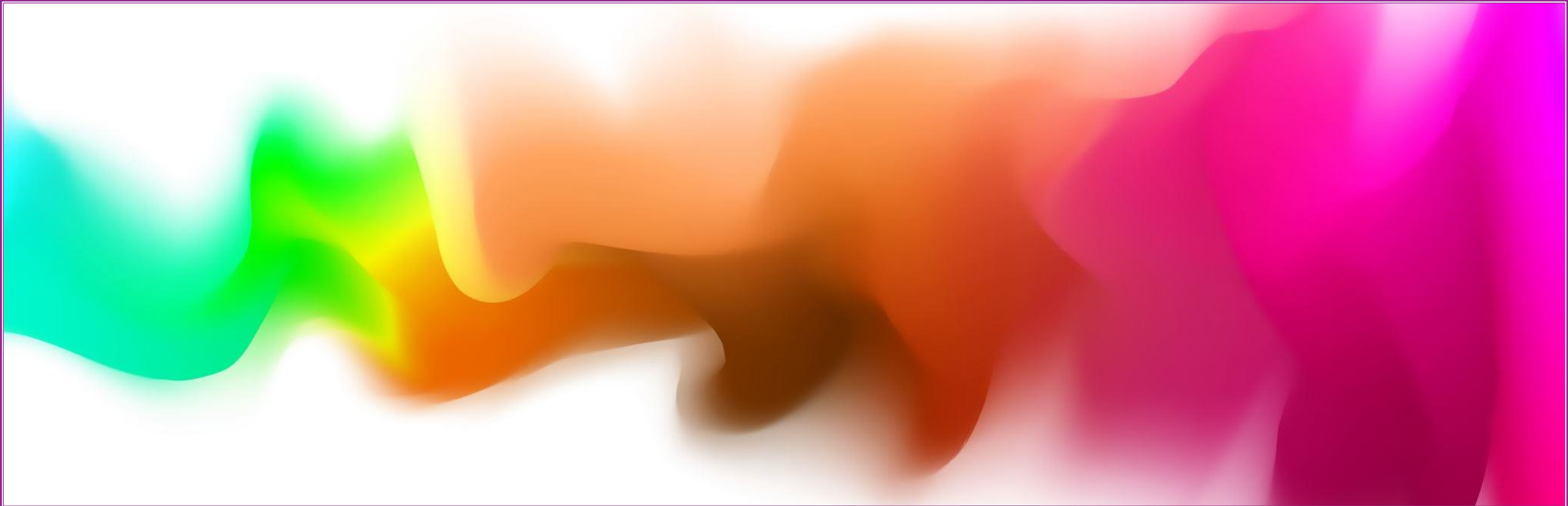
Arrange code
logically

4

Make
dependencies
explicit

5

Minimize
dependencies



KISS PRINCIPLE

Keep it super simple

Grouping Functionality

Namespaces

Types

Projects

Executables

Libraries

Packages

Modules, Components, and Units

The terms “module”, “component”, and “unit” are used to describe arbitrary groupings of code generically, independent of language. There is no formal meaning in the context of C#.

Namespaces

Namespace allow type to be grouped together

Can be nested

At most one namespace per file

One namespace may occur in multiple files and libraries

Allows the same name to be used in different contexts

Libraries (Managed Assemblies)

Any managed assembly (DLL or EXE) can be used as a library

You can add a reference directly using a file path

Any .NET libraries can reuse types from an assembly

You can view exported contents of an assembly using “object browser”

Assemblies are versioned: you should only include one version of an assembly

Resolving References to Assemblies

Referenced assemblies have to be discoverable

They are either registered in the global assembly cache (GAC)

Or they are in the same folder as your executable or library

Don't forget to distribute dependencies with your executable or library

Look-up happens at run-time, the first time the referenced type is used

Project References



Within a solution you can create references between projects



This allows visual studio to take care of compiling things in order



Also copies the dependencies to the executable folder

An introduction to NuGet

Article • 10/11/2022 • 10 minutes to read • [12 contributors](#)

[Feedback](#)

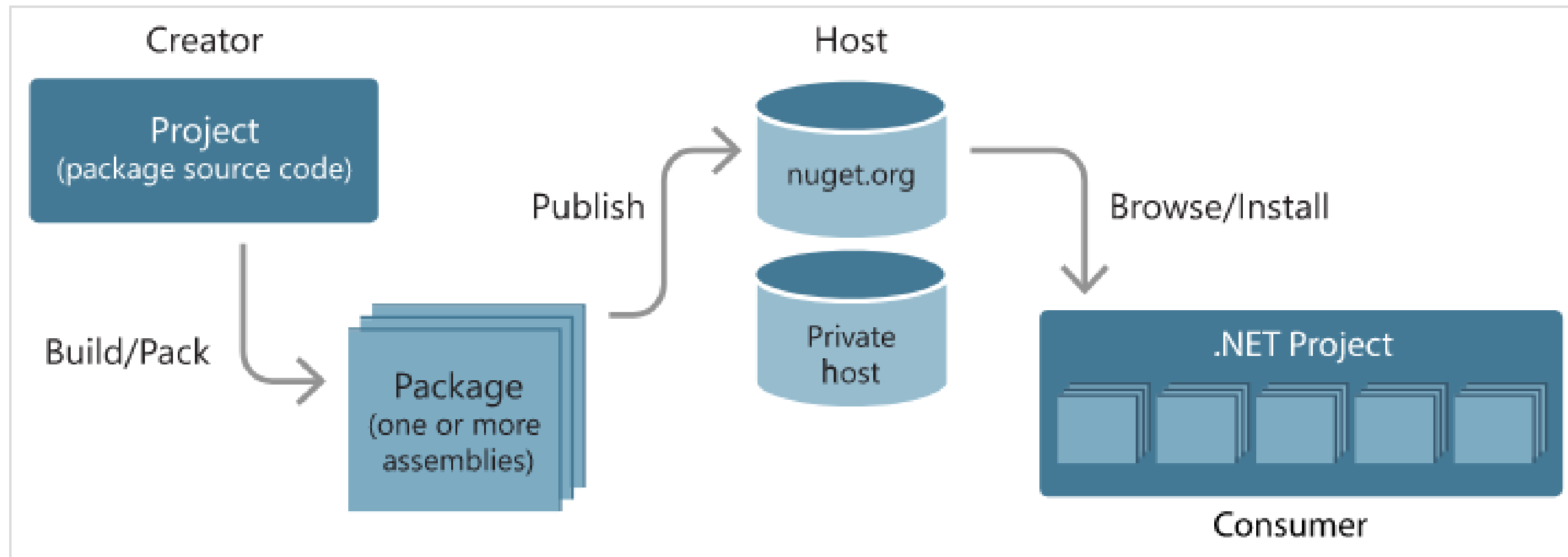
An essential tool for any modern development platform is a mechanism through which developers can create, share, and consume useful code. Often such code is bundled into "packages" that contain compiled code (as DLLs) along with other content needed in the projects that consume these packages.

For .NET (including .NET Core), the Microsoft-supported mechanism for sharing code is **NuGet**, which defines how packages for .NET are created, hosted, and consumed, and [provides the tools](#) for each of those roles.

Put simply, a NuGet package is a single ZIP file with the `.nupkg` extension that contains compiled code (DLLs), other files related to that code, and a descriptive manifest that includes information like the package's version number. Developers with code to share create packages and publish them to a public or private host. Package consumers obtain those packages from suitable hosts, add them to their projects, and then call a package's functionality in their project code. NuGet itself then handles all of the intermediate details.

NUGET PACKAGES

About Nuget

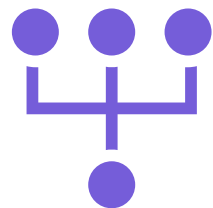


<https://learn.microsoft.com/en-us/nuget/what-is-nuget>

Examples of Libraries as Nuget Package



Manipulating, reading, and writing image data – ImageSharp



Parsing the command-line – CommandLine



Saving arbitrary data to Json – Newtonsoft

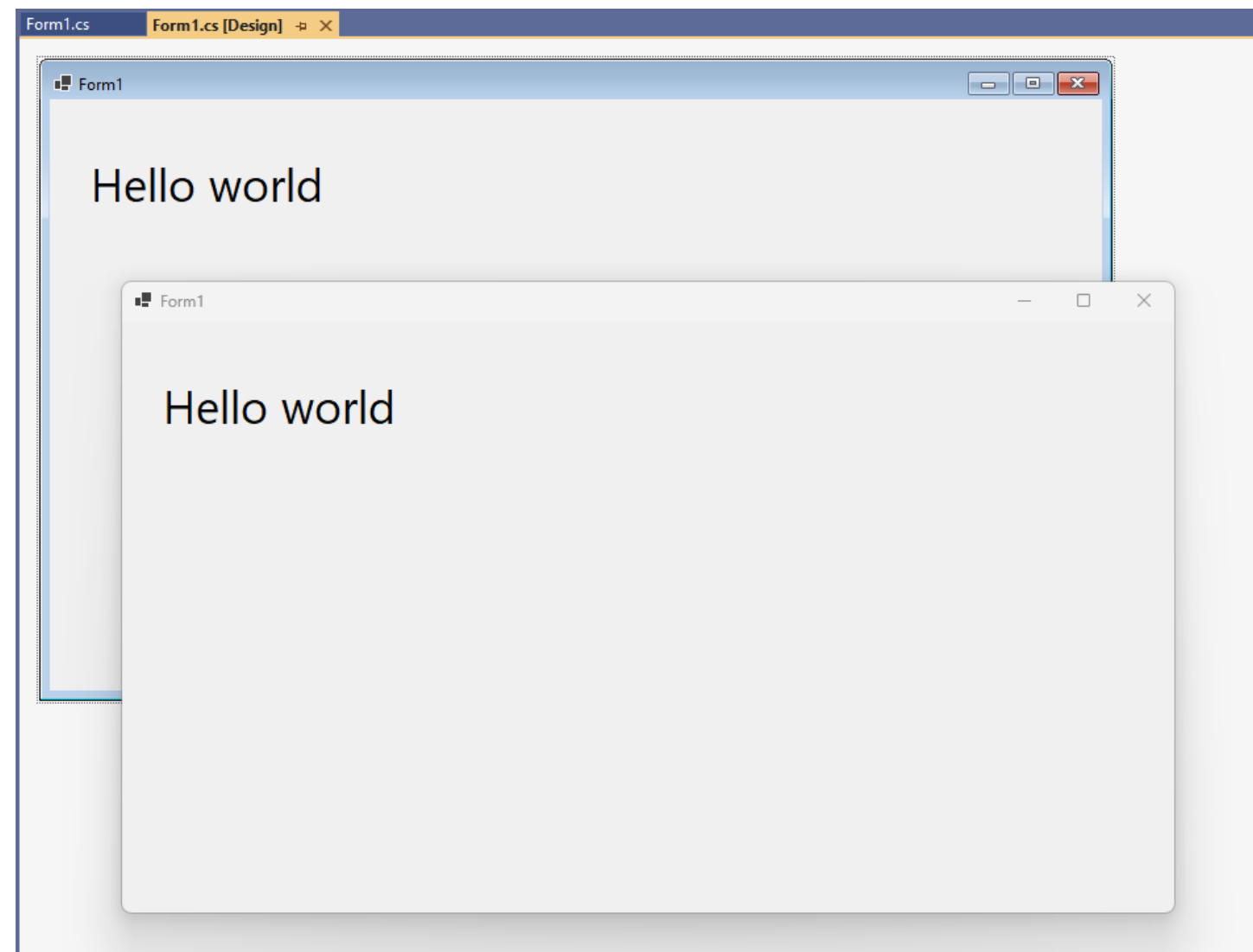


Writing and Running Unit Tests – NUnit

THE GOLDEN RULE

Don't copy code!

Bonus Lab + 1 Point



Create a “Hello world” Windows Forms Application (aka Winforms).

Open the form in the designer

Go to “Window > Toolbox” in Visual Studio

Drag and Drop a “Label” Control onto the form

Change the text of the label using the “properties” of the control

Screenshot it