WINDOWS FORMS

Ondřej Pavlica

PV178

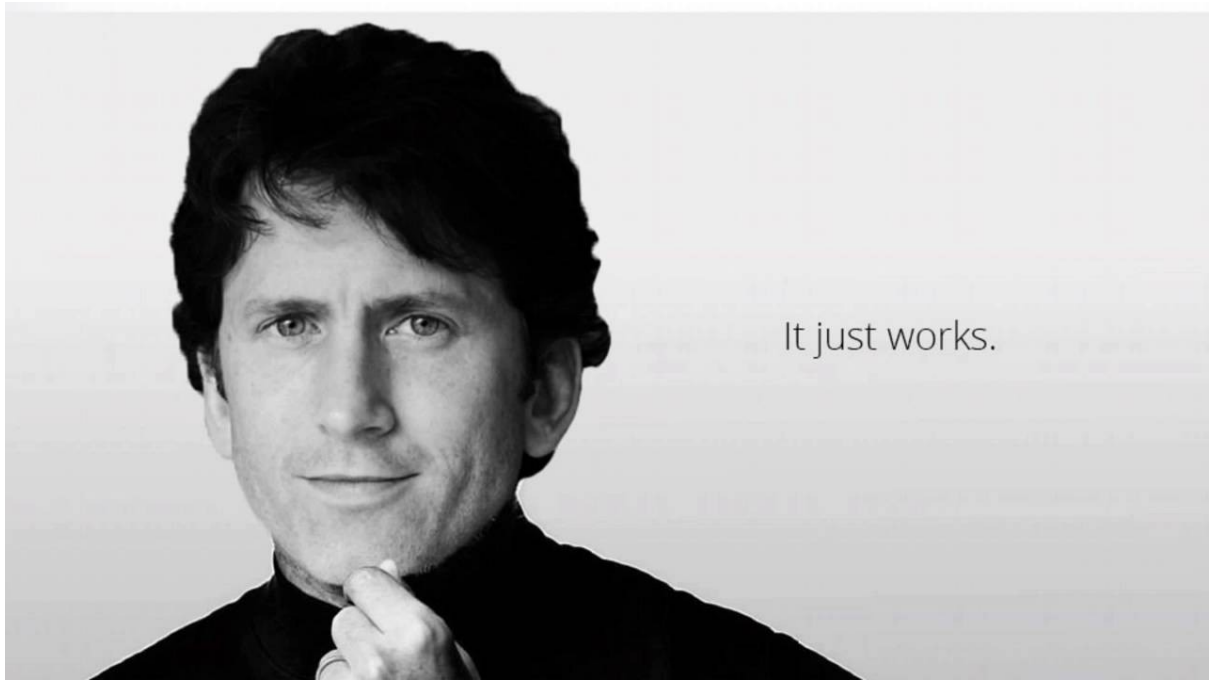
Spring 2022

Coming up

- Theory
  - General info
  - History lesson
  - Alternatives
- Practical use
  - Setting up Visual Studio
  - Standard controls
  - Application logic
  - Dialogs
  - Custom controls
  - Scaling
  - Asynchronous code
- Q&A + Bonuses (if time permits)

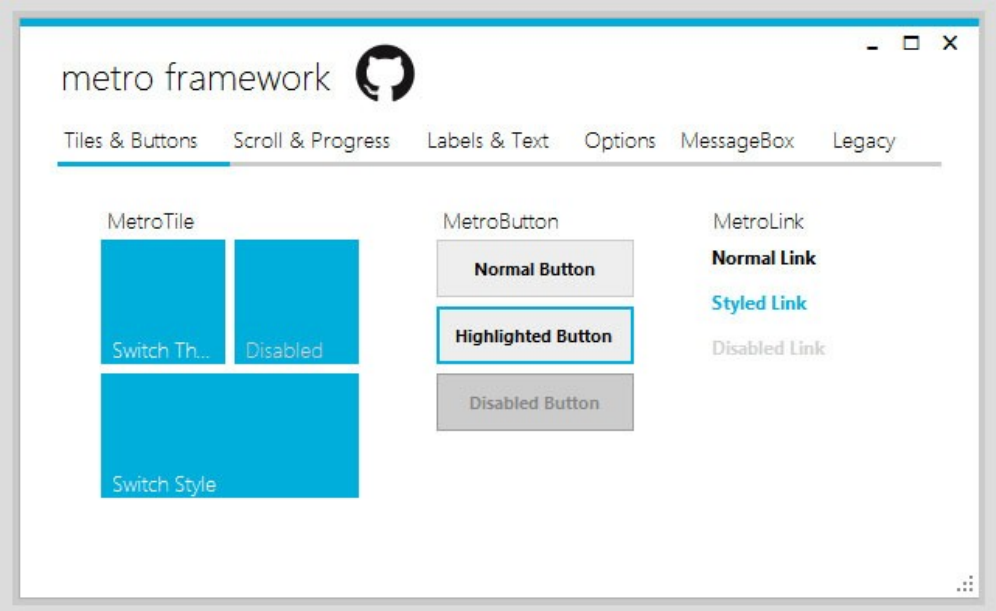
General information

- Windows only
- Event-based
- Very beginner-friendly
  - Drag & Drop
  - It Just Works™
- Very senior-unfriendly
  - Limited UI scaling
  - No adaptive font sizes
  - Very low-level for advanced scenarios
  - Esoteric bugs

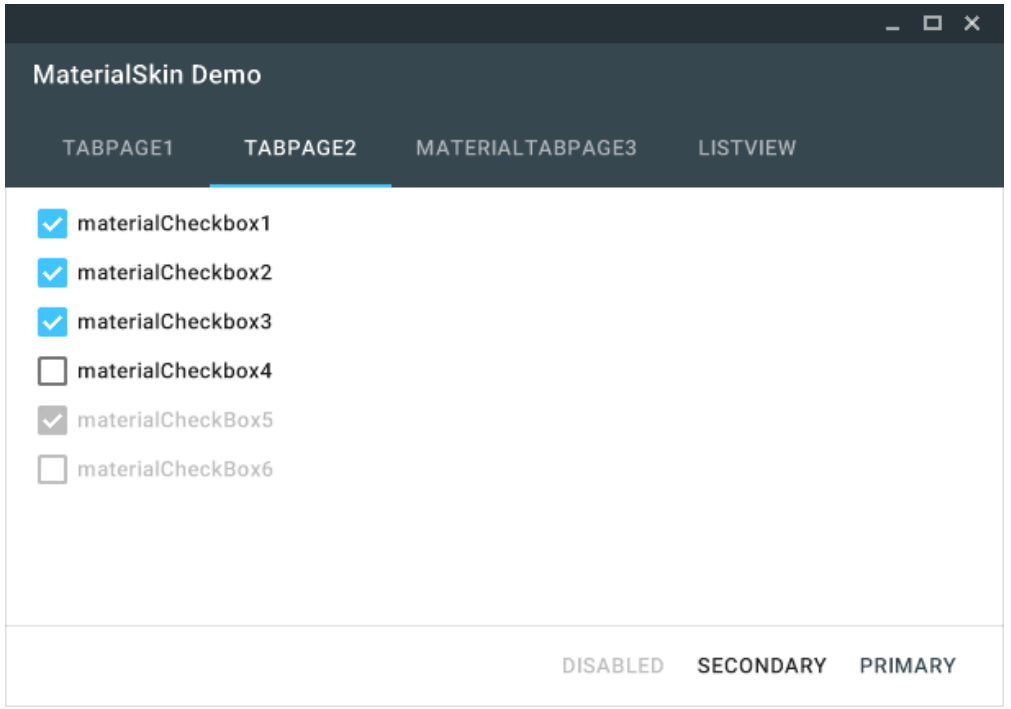


WinForms don‘t need to be ugly

MetroFramework



MaterialSkin



History lesson

- Oldest C# UI library (2002, in beta since 2000 - .NET 1.0)
  - Therefore has a lot of skeletons in its closet (to maintain backwards-compatibility)
- Internally uses a lot of Win32 API calls
  - Rendering native form controls (checkboxes, radio buttons, etc.)
  - Window management (e.g., moving to foreground)
- Uses GDI / GDI+ for rendering
  - Not maintained anymore by MS
  - A lot of known bugs
  - CPU rendering only – no HW acceleration, performance hit

History lesson

- Open-source rewrite in .NET Core starting in 2018
  - More-or-less feature-complete by .NET Core 3.1
  - DPI scaling
  - Modern dialog windows
  - Still Windows-only
  - API compatibility with the .NET 3 version of WinForms (still pretty bad)
  - Still rendered by the ancient GDI+ library
- No multiplatform port planned (as of yet)
  - This use case will be (most probably) be covered by MAUI (former Xamarin.Forms)

Real use cases

- Quick&Dirty UIs
  - Prototypes
  - Internal tools
  - Software where a UI is an afterthought
- Legacy support
  - Yes, a lot of companies still support Windows XP or even older systems
- Student projects ☺

Alternatives - Windows

WPF (Windows Presentation Foundation)

- Declarative approach (XML files)
- HW acceleration (DirectX)

WinUI (UWP)

- Windows 10+
- Prettier, but more limited WPF (access to low-level system APIs)
- Look up „XAML Controls Gallery“ in Windows Store

WinUI

The modern native UI platform of Windows.

Alternatives - Unix

Mono WinForms

- A port of .NET 4 WinForms
- Not feature-complete
- In maintenance mode
- Can often run existing WinForms code without modifications

Alternatives - Multiplatform

Avalonia

- Open-source multi-platform WPF
- Mobile platforms in beta

UNO Platform

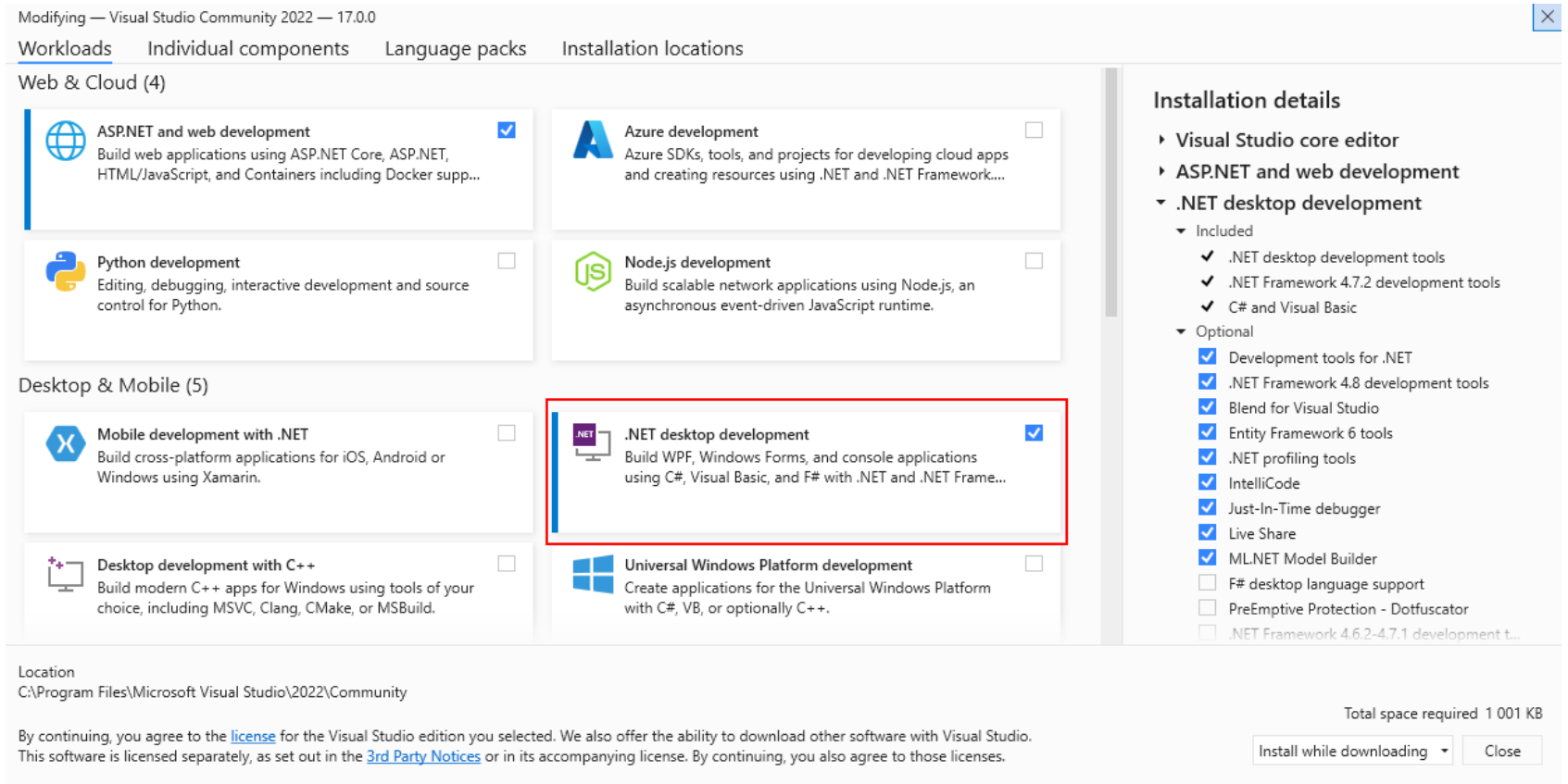
- Multi-platform UWP
- Can run in a web browser (WebAssembly)

MAUI

- UNO Platform, but from Microsoft
- Faster, less mature

WINFORMS IN PRACTICE

Setting up Visual Studio (Installer)



Visual Studio Tour

- Creating a WinForms project
- Toolbox
- Document outline
- Form editor
- Control properties & events

VS Tour + Standard Controls

Application Logic

- Based on handling events
- Switching between UI and code-behind: **(Shift +) F7**
- Don‘t touch the generated code (*.Designer.cs) if you are not sure about what you‘re doing

- **Don‘t put business logic in the code-behind, interact with business logic classes instead!**
  - This is a common bad practice even in software companies
  - The code becomes unmaintainable very quickly
  - There is a high potential of (even unintentionally) storing business data in the UI controls

Application Logic

Dialogs

Custom Controls

- Composition
  - Creating a new UserControl and then drag&dropping existing controls onto it
  - Quite easy to create, reduces repetition of common UI groupings
    - Progress bar with status text
    - Listbox with button controls
- Creating an entirely new control
  - Quite low-level
  - Extending an existing control
    - Differently styled button (e.g., material design)
  - Creating a new control from scratch
    - OnPaint event + drawing basic shapes (points, lines, rectangles, …)

Creating an Entirely New Control

- Out of scope of this lecture
- Taught in PB069

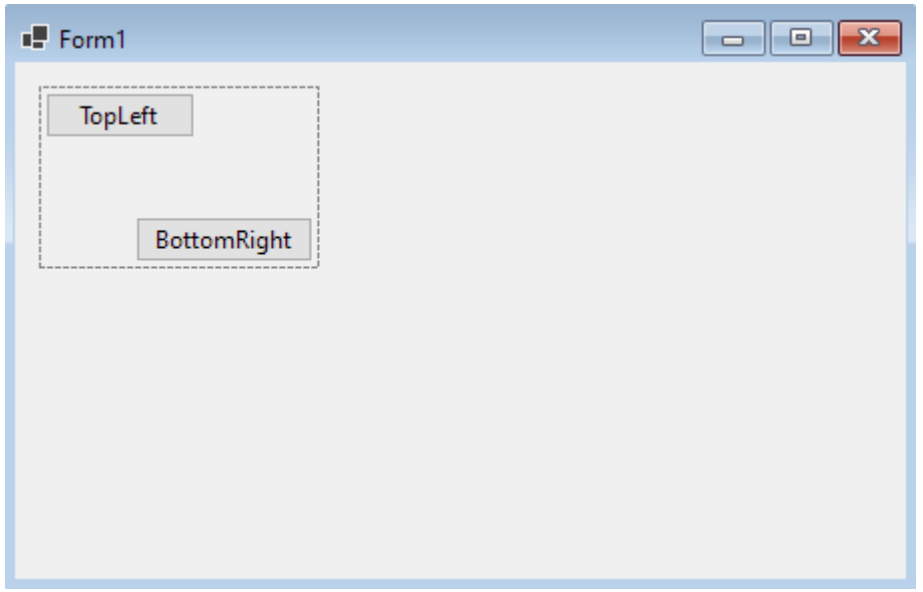
Custom Controls - Composition

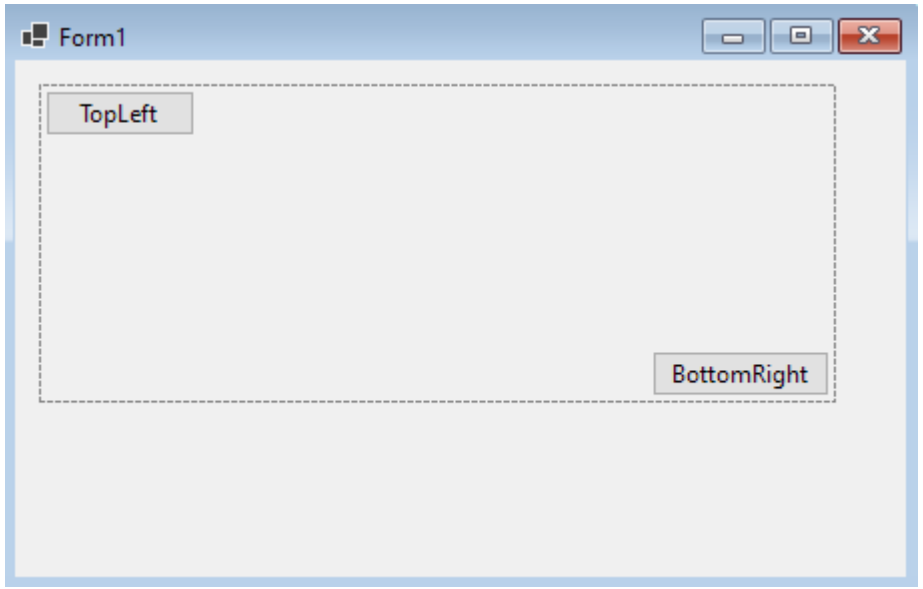
Scaling

- Smart usage of different types of panels combined with:
  - Docking
  - Anchoring
- Major disadvantage – no sane out-of-the-box font scaling
  - Has to be implemented at a pretty low level – using Graphics.MeasureString

Panel

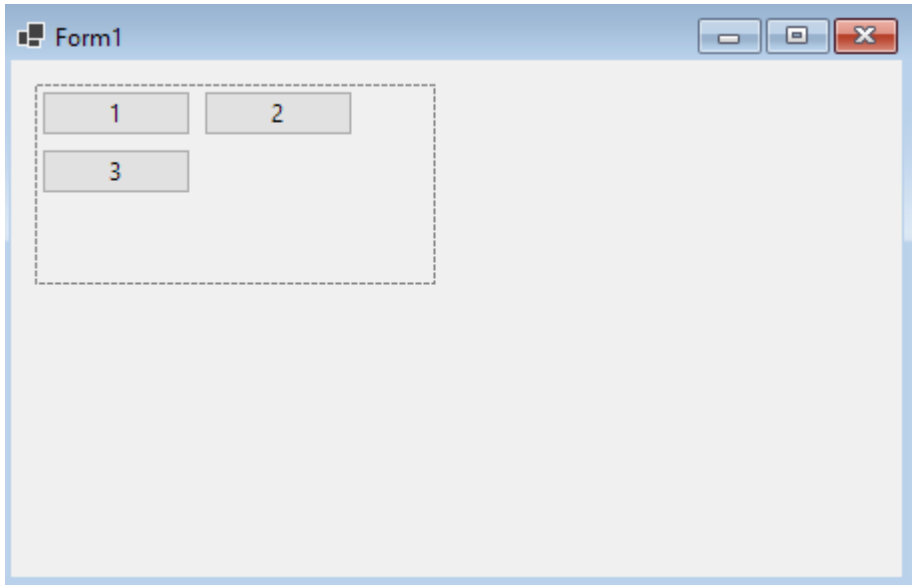
- A container for a group of controls
- Primarily used to scale or move this group of controls together

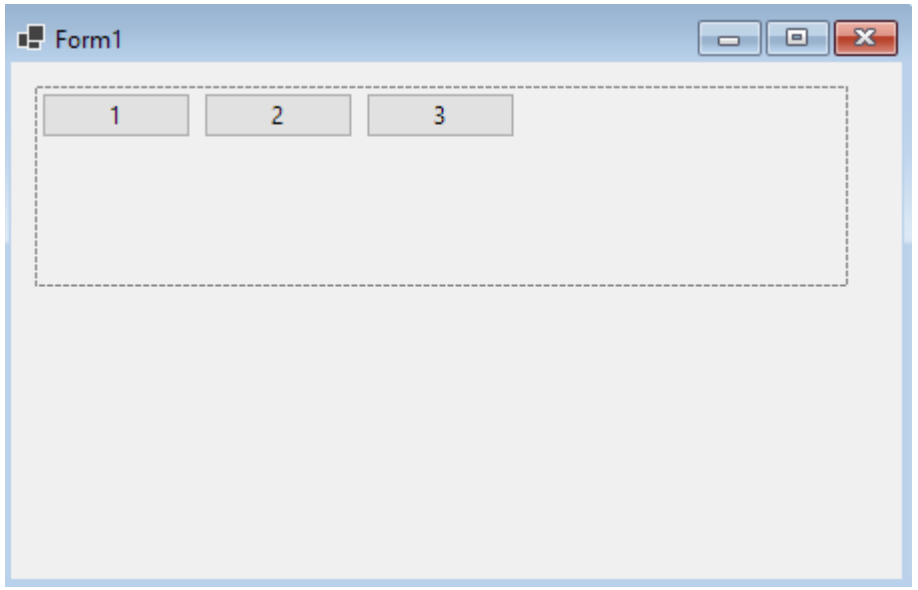




FlowLayoutPanel

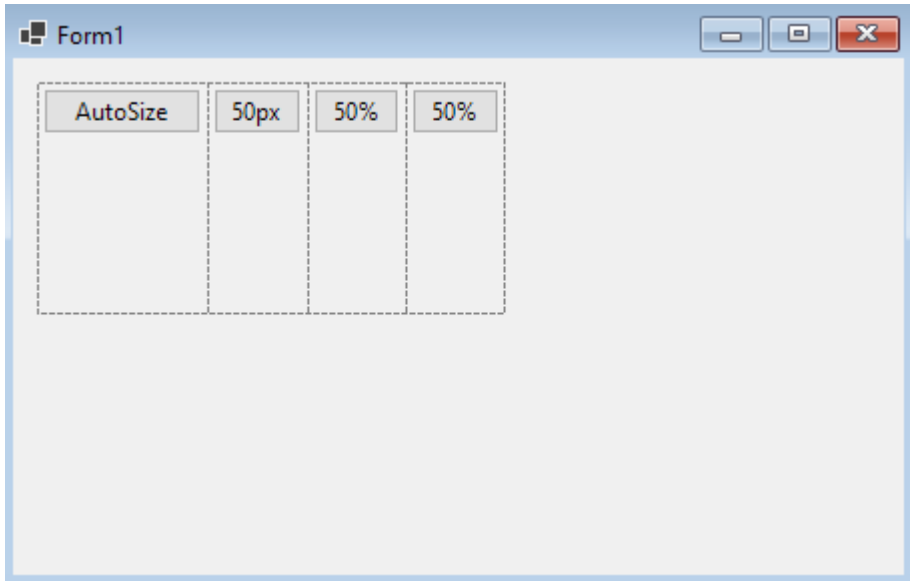
- Used for stacking controls after each other in a certain direction

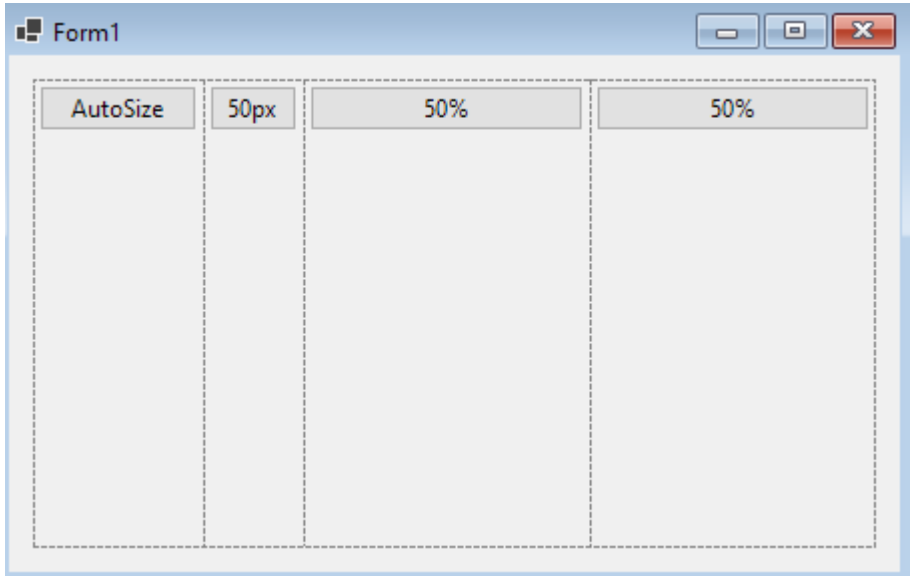




TableLayoutPanel

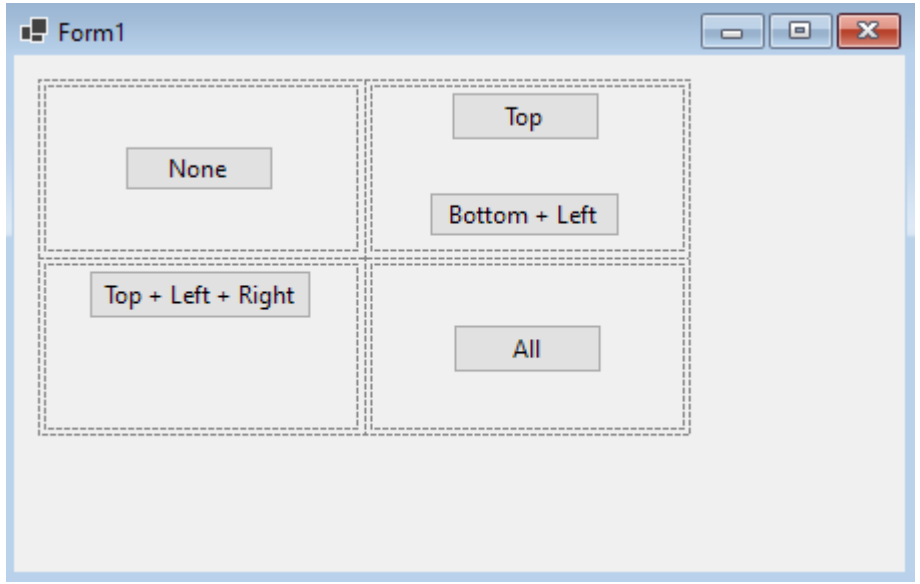
- More fine-tuned control of placements
- Columns and rows sized by:
  - Number of pixels
  - Percentage of available space
  - Contents of the given cell

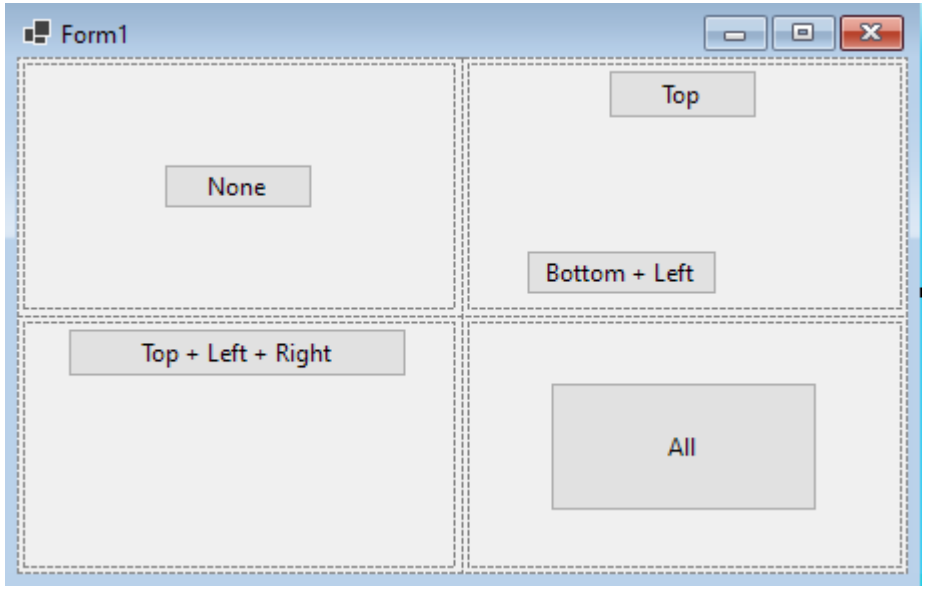




Anchor

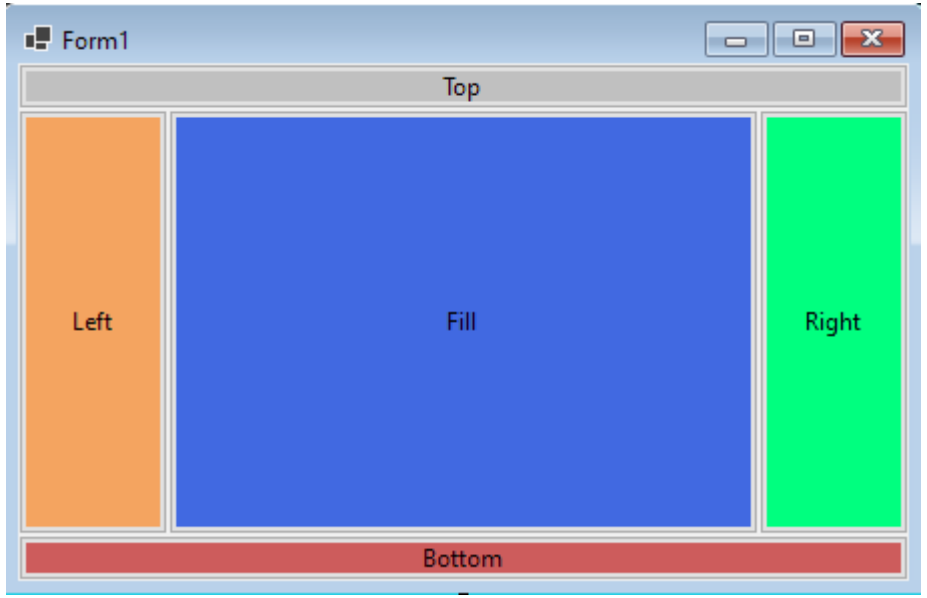
- Controls‘ position is computed relative to the parent‘s position
- This property defines the sides of parent from which the position is computed
- Behavior when resizing:
  - **Zero sides** – relative position to all sides stays the same (great for centering)
  - **1-2 adjacent sides** – the distances (in px) to the chosen sides stay the same
  - **2 non-adjacent, any 3+ sides** – the control is stretched, if possible (e.g., AutoSize = true)

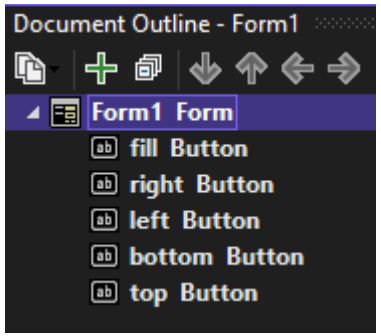


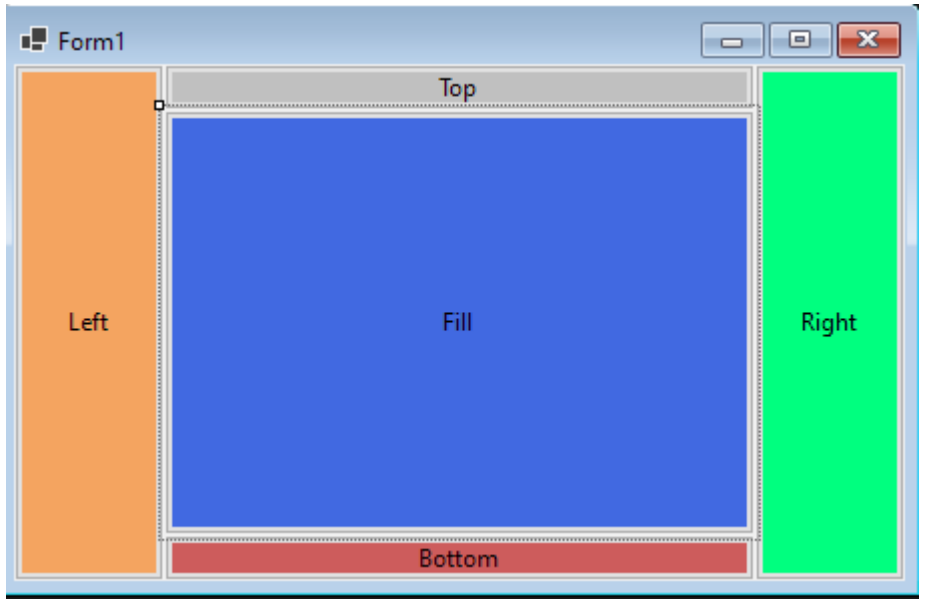


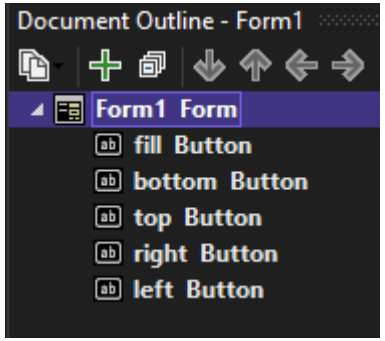
Dock

- Dock property - „Sticking" and stretching a control to fit one of the parent container's sides
- Does not play nice with non-docked controls in the same container
- Docking priority is set by the order of controls in the document tree









Scaling

Asynchronous Code

- Problem 1 – You can access UI only from the UI thread
  - Writing to the UI elements gets more complicated
  - Solution: control.Invoke() and BeginInvoke()
    - BeginInvoke() is fire&forget – less deadlock-prone
- Problem 2: Events have only synchronous delegates
  - Therefore async/await parallelism can only be done by void-returning methods
    - No completion signaling
    - Exceptions get ignored
  - Solution: Side channels in the event handler code
  - Beware using ConfigureAwait(false) -> after awaiting, you **must** use Invoke() to access UI

Asynchronous Code

Further study

- PB069 course
- Microsoft docs
- CodeProject.com – best resource for obscure parts of WinForms
- Experimenting ☺

Q&A

THANK YOU FOR ATTENDING