What’s new for WPF in .NET 7

WPF community is filled with so many passionate individuals with truly amazing experiences and this post aims to highlight what has been done in the dotnet/wpf repo in past few months and thanking the amazing people behind all this. We are really grateful for the contributors who have consistently worked towards improving WPF. Here is a quick recap of what was accomplished in the past few months in the dotnet/wpf repository.

Performance

WPF in .NET 7 ships with number of improvements in the areas not just limited to unnecessary boxing/unboxing, use of Span for string manipulation, better allocation/deallocation of objects, memory improvements, font rendering etc. but also code cleanup and making way for future readiness.

Boxing/Unboxing

Boxing and Unboxing are computationally expensive processes. When a value type is boxed, a new object must be allocated and constructed. To a lesser degree, the cast required for unboxing is also expensive computationally.

- Avoid boxing when setting DependencyObject properties *(Thanks Ilya)*
- Stop boxing WeakReferenceListEnumerator in PresentationSource use
- Reuse bool box objects in UncommonField.SetValue
- Stop boxing in Visual.SetDpiScaleVisualFlags
- Avoid enumerator boxing in XamlSchemaContext.UpdateNamespaceByUriList
- Avoid boxing list/array enumerator in CreateTextLSRuns
- Avoid boxing list enumerator in XamlObjectWriter.Logic_ConvertPositionalParamsToArgs

Allocations

The more objects allocated on the heap, more is GC overhead in reclaiming those object post their lifetimes. Reducing such allocations in memory lowers GC overhead.

- Avoid Hashtable-related allocations in AccessorTable
- Avoid Hashtable-related allocations in DataBindEngine
- Remove closure/delegate allocation in ItemContainerGenerator
- Avoid delegate allocation to call ListCollectionView.PrepareComparer
- Avoid unnecessary byte[] allocation in Baml2006Reader.Process_Header
- Avoid allocating Stack<BranchNode> just to peek at it
- Stop allocating unnecessary StringBuilders in ParsePropertyComments
- Avoid unnecessary StringBuilder reallocation in LookupAndSetLocalizabilityAttribute
- Avoid exceptional string allocation in StaticExtension.ProvideValue
- Remove unnecessary string and string[] allocations from MS.Internal.ContentType
- Remove some unnecessary StringBuilders
- Remove substring allocation from Baml2006Reader.Logic_GetFullXmlns
- Don’t allocate fallback name in XamlNamespace.GetXamlType unless it’s needed
- Avoid unnecessary enumerator allocations in XamlDirective.GetHashCode

Miscellaneous

- Avoid excessive calls to the PropertyValues index getter. *(Thanks paulozemek)*
- Harden events against race conditions *(Thanks Bruno Martinez)*
- Eliminate memory copy when reading font data *(Thanks Bradley Grainger)*
- Small performance improvement of PathParser *(Thanks ThomasGoulet73)*
- Use span slice instead of substring in AbbreviatedGeometryParser.ReadNumber
- Avoid unnecessary duplication of fields in NullableBooleanBoxes
- Change most non-generic sorts to be generic
- Some improvements to FrugalList
- Avoid unnecessary Collection<T> wrapper in CombineSources
- Avoid unnecessary List<> wrapper in GetTextRunSpans

Special thanks to Stephen Toub for contributing many other performance fixes.

Accessibility

With a commitment to ensure WPF controls are accessible, below are the product improvements that made its way to WPF.

- WPF DataGrid/GridView column width can be changed using the keyboard shortcut ALT+left or right arrow key – Datagrid column width can be adjusted using keyboard shortcut Alt + Left arrow / Right arrow.
- Sort Datagrid Column by Keyboard F3 – Datagrid columms can now be sorted (if sorting is enabled for a column) using keyboard shortcut F3.
- Narrator announcement for Checkable Menuitems – Onscreen narrators can correctly announce the presence of checkable menuitems.

Bug fixes

While WPF remains fully supported and serviced on .NET Framework, most fixes and all new features will go only into .NET Core, where we have the opportunity to make bigger changes. Our community helped address some long-standing bugs in this release.

- FocusVisualStyle can't be overwritten globally *(Thanks Bastian Schmidt)*
- CommandParameter invalidates CanExecute by miloush *(Thanks Jan Kučera)*
- Tooltip issues
  - .NET 6 Tooltip behavior change from .NET 5 (bug?)
  - Comboboxitem tooltip bug
- ContextMenu stops working if its owner is removed from the visual tree
- Fixes rounding error while glyphrun serialization

The above list is **NOT** exhaustive and many more bug fixes went thanks to our community contributors for their efforts in fixing them.

Infrastructure upgrades

One of the major areas for improvement voiced by community is the rate at which community contributions were accepted. With the below infrastructure upgrades, we intend to address the rate at which we accept community contributions.

Title	Description
Test repository migration	WPF codebase has more than 30K integration tests. These tests ensure sanity of the build with various OS and .NET framework combination matrix. As part of open sourcing the test infrastructure, we aim at moving all the tests from internal to Github. This would also enable community to add their own tests to the test repo. We have open-sourced most of the basic regression tests that enables the community contributors to run those tests locally and debug them, in case there's an issue with any of the PR submissions .
Running basic tests on each incoming PR	Basic regression tests, *(aka Daily Regression Tests or DRT(s))* that validates the basic behavior of controls, need to be run on each submitted PR to ensure that the changes do not cause any regressions. The end goal of this exercise is to make sure that we have a framework in place that allows running tests on incoming PRs. This would reduce the turn-around times on PRs, thereby increasing the velocity at which new changes / fixes can be merged into the repository. We are now running these tests as a part of build on every incoming PR. In upcoming months, we plan to enhance these pipelines in terms of the reporting the status if test execution, to avoid manual lookup in logs for failures.

Ongoing activities

- Tests repository migration still has a larger subset of tests that are yet to be ported.
- Enabling running the bigger suite of test cases on community submitted PRs which would improve the turnaround time for feedback on PRs.
- Clearing the backlog of PRs and issues.

Community

The community run-project started with the intent to enable developers in making big difference in shaping WPF going forward. To all the WPF developers, your work is invaluable. We'd like to thank the below contributors for their efforts in fixing long standing issues and contributing performance and functional improvements.

- Andrii Kurdiumov
  - Fix ZeroForNow parameter
  - Remove sorcery for getting consistent names in the traces
  - Explicit delegate types
  - Update MSBuild tasks
  - Improve Linux build
  - Remove flacky and redundant codegen

- Austin Wise
  - Fix setting DPI awareness

- Bastian Schmidt
  - Fixing TextBoxView memory leak for 2 seconds after unloading host control
  - Use regular resource lookup for FocusVisualStyle
- Bradley Grainger
  - Eliminate memory copy when reading font data

- Bruno Martinez
  - Harden events against race conditions

- Ilya
  - Avoid boxing when setting DependencyObject properties

- Jan Kučera
  - Check script of combining marks during font fallback
  - CommandParameter invalidates CanExecute
  - Ignore NotImplementedException from ITaskbarList

- lindexi
  - Fix the stream do not be closed in ImageSourceTypeConverter
  - Fix GetStreamCore in ContentFilePart
  - Fix create BitmapDecoder with async file stream.
  - Using the Clone method to fast clone the array in StylusPoint
  - Using ArrayPool in RenderData
  - Using Array.Copy to make array copy faster in StylusPointCollection
  - Use pattern matching in TaskExtensions

- paulozemek
  - Avoid excessive calls to the PropertyValues index getter.

- Pomain
  - Remove 'Zero Width No-Brake Space' character from multiple files

---

- ThomasGoulet73
  - Small performance improvement of PathParser
  - Use params and char overload
  - Inline VerifyAccess
  - Fix Invalid IInputElement resource
  - Disable Indeterminate animation when hiding ProgressBar
  - Replace IsAssignableFrom with is in converters

- Use generic Marshal.StructureToPtr
- Migrate DPI awareness initialization to managed
- Use generic Marshal.PtrToStructure

Summary

We’d encourage you to try out WPF on .NET 7 and let us know how these improvements have helped. We are always looking for feedback on how to improve the product and look forward to your contributions. We would like to thank everyone that is committed to making WPF better as a product. Our goal is to continue improving WPF, while growing our community so that we can bring you the best developer experience possible. Your help and input is very much required. Whether that is through triaging issues, updating documentation, participating in discussions or writing code, we appreciate all of your help!