# Discovering the Best Developer Framework Through Benchmarking

## Delphi, WPF with .NET Framework, & Electron On Microsoft Windows 10

16 DECEMBER 2020 | VERSION 1.0

**Authors**

Jim McKeeth | Adam Leone | Eli M.

embarcadero

Embarcadero Technologies
10801 North Mopac Expressway
Building 1, Suite 100
Austin, TX, 78759
WWW.EMBARCADERO.COM

## Executive Summary

When businesses choose a software framework, they commit to a long-term relationship for the duration of their application's lifecycle. Given the strategic consequences of this decision, businesses must carefully consider a framework's developer **productivity**, business **functionality**, application **flexibility**, and product **performance**. The best framework demonstrates strength in each category and will minimize product time-to-market, lower maintenance costs, maximize product variety, and provide a superior customer experience.

This paper evaluates three frameworks for Windows application development - **Delphi**, **Windows Presentation Foundation (WPF)** with the **.NET Framework**, and **Electron**. Each development framework will create Windows applications but calls upon different languages, libraries, IDEs, and compilation models. To assess these frameworks, this paper defines four evaluation categories, describes 23 metrics, defines the benchmark application, and scores each framework using a weighted evaluation. The benchmark, a Windows 10 Calculator clone, assesses frameworks' ability to re-create a known GUI and target the Windows desktop environment.

Evaluation conclusions include:

1. Delphi and its RAD Studio IDE profoundly enhance development productivity and product time-to-market. Not only that, developing one codebase to reach every desktop and mobile platform simplifies successive releases and product maintenance.
2. WPF with the .NET Framework offers small teams native entry to Windows applications and a solid IDE but struggles to match Delphi's productivity, IP security, and performance while also missing Delphi and Electron's cross-platform features.
3. Electron offers a free alternative to Delphi and WPF, familiarity to front-end developers, and cross-platform capability at the cost of IP protection, standard IDE tooling, and application performance

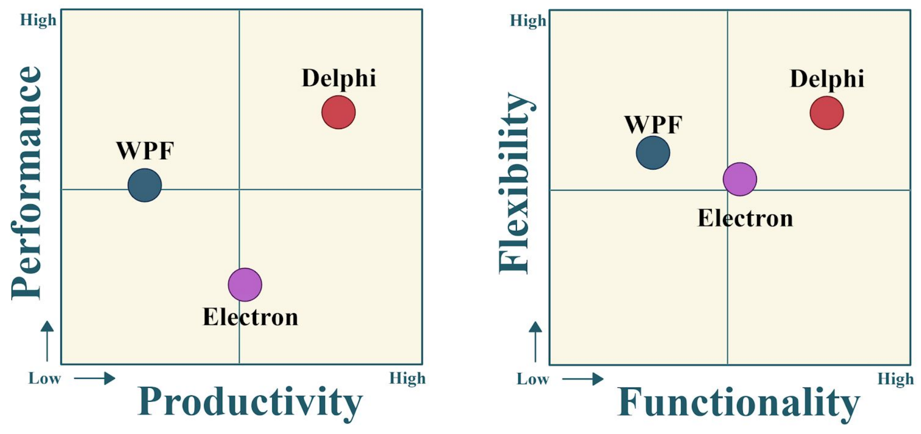


*Figure 1 - Depiction of Framework Category Scores*

# Table of Contents

## 1. Introduction

Today's proliferation of software development tools is a cause for celebration while also requiring more effort from decision-makers to critically assess the benefits and drawbacks of common frameworks and IDEs. Although some tasks may require a specific tool, software companies developing Windows desktop applications may choose from a variety of frameworks. Given a choice, developers would benefit from choosing the framework, IDE, or tool that measurably maximizes productivity and provides outstanding functionality, flexibility, and performance. While this choice is ideally informed by solid research, popularity trends within the industry and interest in the next "shiny" tool undeniably influence framework, IDE, and tool selection. This paper aims to counter capriciousness and lay the groundwork for developer tool comparisons by defining a benchmark methodology, applying it to Delphi, Windows Presentation Foundation (WPF) with .NET Framework, and Electron using a calculator application, and drawing conclusions about the productivity, functionality, flexibility, and performance of each framework.

This paper is structured as follows: Section 2 introduces the history of developer tools and discusses related academic tool comparisons. Section 3 describes the broad methodology of this comparison in four categories, lays out specific metrics per category, introduces the frameworks under comparison, and describes the evaluation benchmark application and weighted comparison. Section 4 analyzes the metrics by category and Section 5 draws conclusions.

## 2. Related Work

It should be no surprise that software developers and academics have conducted tests and comparisons since the second-ever framework was released. A time-honored method of comparison is the *benchmark*. Formalized in the 80s, Robert Camp's widely accepted definition of a benchmark is "the search for the best industry practices which will lead to exceptional performance through the implementation of these best practices".[1] Software industry benchmarks are most frequently manifested as quantitative performance tests and evaluations, such as task completion speeds, but can also be qualitative via scoring systems and weighted assessments. The key to any benchmark is to apply the test equally to *like systems* from different companies, avoiding an "apples-to-oranges" comparison, and to incorporate the objective results into business decisions.

Since 2015, academia has produced over a dozen comparison papers[2] focusing with increasing intensity on cross-platform frameworks for mobile application development, seeking to fill that gap in academic literature.[3] While each paper's focus varied, many used performance

[1] Camp, Robert C. *Benchmarking: the search for industry best practices that lead to superior performance*. Asq Press, 1989.

[2] Majchrzak, Tim, and Tor-Morten Grønli. "Comprehensive analysis of innovative cross-platform app development frameworks." In *Proceedings of the 50th Hawaii International Conference on System Sciences*. 2017, Table 1.

[3] Biørn-Hansen, Andreas, Tim A. Majchrzak, and Tor-Morten Grønli. "Progressive web apps: The possible web-native unifier for mobile development." In *International Conference on Web Information Systems and Technologies*, vol. 2, pp. 344-351. SCITEPRESS, 2017, p. 349.

benchmarking techniques and qualitative assessments to compare applications written to the same level of functionality in different frameworks.[4] Seeing that recent literature lacked evaluation cohesion and consistent metrics, Rieger and Majchrzak proposed a detailed evaluation schema for future comparisons.[5] Their four-category, 31 metric schema, and corresponding weighted evaluation table provide an excellent starting point for both qualitative and quantitative comparisons.[6]

## 3. Methodology

### 3.1. Evaluation Categories

This paper evaluates frameworks supporting Windows development according to four aspects of effectiveness: developer productivity, business functionality, framework application flexibility, and end-product performance. Combined, these aspects examine a framework's impact on the entire business and product lifecycle. An excellent framework will speed product development but should also nurture a maintainable codebase and easily pair with in-house or 3rd party tools. Some of the frameworks investigated offer cross-platform options but analysis of them is outside the scope of this Windows-oriented document.

Developer productivity is the measure of effort and code required for developers to complete typical development tasks. Productivity directly impacts product time-to-market and long-term labor costs so tools that increase developer productivity have substantial impacts on business timelines and bottom lines. Productivity can be realized in two distinct ways - reduced coding requirements due to native libraries and also IDE tools like code-completion and visual design. IDEs with greater library breadth generally result in fewer lines of code per application and produce a clean, lean codebase that minimizes opportunities for bugs or maintenance problems later in the product life cycle.

Business functionality refers to a framework's business suitability and impact on long-term plans. Excellent functionality allows companies to easily build custom tools or extensions, develop on a platform of their choosing, protect their source code from exploitation, and have confidence that their applications will be maintainable for decades.

Application flexibility assesses the breadth of tasks addressable with the framework. While IDEs and frameworks are Turing-complete, and thus technically infinitely flexible, some are better suited to a task than others. Flexible frameworks allow businesses to target a broad audience, build software for every application tier, and access operating system functions and consumer hardware.

[4] Willocx, Michiel, Jan Vossaert, and Vincent Naessens. "A quantitative assessment of performance in mobile app development tools." In *2015 IEEE International Conference on Mobile Services*, pp. 454-461. IEEE, 2015, pp. 455-456.

[5] Rieger, Christoph, and Tim A. Majchrzak. "Weighted evaluation framework for cross-platform app development approaches." In *EuroSymposium on Systems Analysis and Design*, pp. 18-39. Springer, Cham, 2016, p. 8-14.

[6] Rieger and Majchrzak, 2016, p. 16.

Consumers judge applications in part by runtime performance. Businesses choosing the framework with superior performance avoid customer dissatisfaction by minimizing wait times and resource-use on slow machines.

## 3.2. Metrics

### 3.2.1. Productivity

Framework *productivity* will be evaluated according to the following metrics:

**[P1] Development Time:** Total hours spent writing the fully functional application from scratch. This measurement assesses the value a framework's productivity tools add to an average developer with no prior task knowledge. Comprehensive documentation, plentiful native libraries, code completion, and other IDE tools will allow the developer to design and build the benchmark application more efficiently than would be the case in a "standard" text editor.

**[P2] UI Design Approach:** Does the framework's IDE allow for graphical/visual application creation and provide a "What you see is what you get" (WYSIWYG) view model?[7] IDEs that support development through "drag and drop" components or other visual methods allow users to engage different methods of thought and creativity as they work. Visual creation through WYSIWYG editors preclude businesses from needing every version of physical hardware to view platform-native styling.

**[P3] Developer Environment Tools:** Does the framework IDE standard installation include auto-completion, debugging, and emulation tools? Are multiple IDEs available for the framework?[8] Frameworks with multiple development tools and a choice of IDE better support individual development preferences, techniques, and requirements.

**[P4] Speed Implementation Time:** Total hours required to "speedrun" the application using a known solution. This measures the number of actions and volume of code required to complete the full application by an expert developer with perfect knowledge of a working solution. Productive frameworks reduce development time on repetitive, but slightly altered tasks.

**[P5] Code Size:** Total lines of code the developer must write, adhering to accepted formatting and styles, to create a fully functional application. This objective measure of code volume sheds light on the difficulty of future code maintenance - more code typically requires more time to learn and troubleshoot.

**[P6] Application Store Deployment:** Does the framework's IDE facilitate direct deployment to native platform application stores (i.e. iOS App Store, Android's Google Play, Microsoft Store)? Frameworks with built-in deployment features reduce product

[7] Rieger and Majchrzak, 2016, p. 11.
[8] Rieger and Majchrzak, 2016, p. 10.

deployment complexity, limiting errors that could occur or compound, and time-to-market for initial products and updates/bug-fixes.

### 3.2.2. Functionality

Framework *functionality* will be evaluated according to the following metrics:

**[F1] License:** Does the license allow the development of commercial applications and at what cost? Is the licensing difficult to understand? Proprietary frameworks and tools may require one-time or recurring payments and have different levels of licenses according to the commercial applications desired.[9] Open-source frameworks may have license-specific restrictions.

**[F2] Long-term Feasibility:** Does the framework have a history of stability, backward compatibility between major releases, bug fixes, and security updates?[10] This metric highlights the confidence businesses can enjoy or the strategic risk they may take when choosing a framework.

**[F3] Supported Development Platforms:** Can application development occur on any major operating system or does the framework IDE impose limitations? This metric indicates how a framework may hinder a team without homogenous equipment.

**[F4] Testing Support:** Does the framework ship with a testing suite, test coverage analysis, and runtime monitoring capability?[11]

**[F5] Tool Extension:** Can the framework be extended in its own language? Frameworks that require plug-ins, extensions, or modifications to be written in a different language impose costs on businesses that require altered functionality. Rather than creating the required tool from resident knowledge, businesses may have to invest time and resources to hire an external contractor or build in-house skills in that alternate language.

**[F6] Accessibility:** Do programs built with the framework support the major OS accessibility features like screen readers and font size/color changes?

**[F7] Intellectual Property Security:** How secure is the intellectual property of the source code in a deployable project? After businesses invest resources into their projects, they face the challenge of putting their product into the hands of the public while protecting the code and techniques that produce revenue. This qualitative metric evaluates the ability of a user to access source code via decompilation.

### 3.2.3. Flexibility

Framework *flexibility* will be evaluated according to the following metrics:

[9] Rieger and Majchrzak, 2016, p. 8.
[10] Rieger and Majchrzak, 2016, p. 10.
[11] Rieger and Majchrzak, 2016, p. 11.

**[X1] Supported Target Platforms:** How many user-platforms can the framework deploy an application to? Great frameworks will support most platforms on the market, whether mobile, desktop, 32-bit, or 64-bit. Businesses benefit from multi-platform support because they can develop and maintain one codebase to reach many customers. One codebase rather than separate code for each target application reduces development time, bug potential, maintenance requirements, and time-to-market for new features.

**[X2] Project Variety:** Does the framework support development of different types of applications from stand-alone desktop apps to Windows services? Flexible frameworks allow developers to create mobile applications, desktop services, and everything in between.

**[X3] Scalability:** Can the code be partitioned into subcomponents based on architectural design? Is the framework suitable for client, middle-tier, and backend systems? Frameworks that support modularity and multiple design tiers are better suited for large, enterprise applications and specialization among multiple teams working on the same project.[12]

**[X4] Database Access:** Does the framework contain native libraries supporting database access? Data persistence is critical for many applications and must be user-friendly and integrated with any good development framework.

**[X5] Access to Device-specific Hardware:** Does the framework facilitate access to data from device sensors (GPS, microphone, accelerometers, camera, etc.) and physical action through similar devices?[13] Frameworks that "throw open the doors" to the plethora of sensors and actuators available on smart devices today create business opportunities and novel solutions to consumer pain.

**[X6] Access to Platform-specific Functionality:** Does the framework allow applications to interact with the host platform's operating system and access its services like file system access, contact list, battery state, and push notifications?[14] Access to core OS functions prevents code duplication, avoids presenting potentially inconsistent data to users, and provides increased ways to collect and analyze information.

### 3.2.4. Performance

Framework *performance* will be evaluated according to the following metrics:

**[R1] Deployment Requirements:** What is the file size/number of files for the compiled project? Larger application sizes require more storage on user devices and longer download times while numerically more files can increase deployment complexity.

[12] Rieger and Majchrzak, 2016, p. 10.
[13] Rieger and Majchrzak, 2016, p. 12.
[14] Rieger and Majchrzak, 2016, p. 12.

**[R2] Startup Time:** Over 100 executions, what is the average time from command to a visible application ready for user input when started on a local machine and over a network? Frameworks producing applications with shorter start-up times facilitate good user experiences and minimize system resources required before the application is useful.

**[R3] Standing Memory Usage:** How much memory is required for the application to run while idle as measured by a task manager tool? Better frameworks produce applications requiring lower overhead when the user isn't actively using them, thus conserving total system resources.

**[R4] Peak Memory Usage:** What is the maximum memory required for the program from startup through heavy use as measured by the Windows Task Manager?

## 3.3. Frameworks

This paper compares Embarcadero Technologies' RAD Studio, Microsoft's WPF with .NET Framework, and Electron.

### 3.3.1. Delphi

Delphi, encapsulated in the Rapid Application Development (RAD) Studio IDE, is Embarcadero Technologies' flagship product. A proprietary version of the Object Pascal language, Delphi features graphical application development with "drag and drop" components, a WYSIWYG viewer for most mobile platforms, and robust style options including platform-standard and unique IDE palettes. Among other features, included libraries provide GUI controls, database access managers, and direct access target platform hardware and platform operating systems.

Delphi offers two distinct frameworks - the Visual Component Library (VCL) for 32-bit and 64-bit Windows applications and the FireMonkey (FMX) framework for 32-bit and 64-bit cross-platform applications on Windows, macOS, Linux, Android, and iOS. The FireMonkey framework allows businesses to develop and maintain one codebase while reaching most of the market. Delphi VCL has been available for over 25 years and Delphi FMX for nine years.

### 3.3.2. Windows Presentation Foundation with .NET Framework

Microsoft's Windows Presentation Foundation is a GUI development system that uses DirectX and the eXtensible Application Markup Language (XAML) to separate user interfaces from business logic. Microsoft's Visual Studio is the native IDE for WPF and provides a WYSIWYG view of WPF applications along with drag-and-drop components for visual design. .NET Framework libraries included with every Windows installation provide a plethora of user interface controls and WPF supports complete style creation and modification. WPF will compile to a native Windows binary for installation on the

Windows desktop or to an XAML Browser Application for cross-platform use in web browser sandboxes. WPF has been available for over 15 years.

### 3.3.2. Electron

Electron is an open-source (MIT License), Chromium-based framework that utilizes web technologies to build desktop applications on Windows, macOS, and Linux. It was originally developed by and is still maintained by GitHub (a subsidiary of Microsoft). Electron combines the Chromium-based rendering engine with the Node.js runtime. As such, the user interface for an Electron application is available via HTML5 and CSS. Generally, Electron works with most Javascript frameworks such as Angular, Vue.js, and React. The HTML5, CSS, and Javascript-based technologies found in Chromium provide for a rich ecosystem of user customization familiar to any web developer. Electron has been available for over 5 years.

## 3.4. Evaluation Strategy

### 3.4.1. Benchmark Application

The benchmark application for this comparison is a clone of the Windows 10 “Standard” calculator that ships with every Windows installation. While the calculator logic itself is not complex, the application will test each framework’s ability to emulate a known product and allow side-by-side comparisons. Additionally, it evaluates the ability of a framework to create a small application for the target platform - Windows - and to mimic the Window’s style used for the calculator.

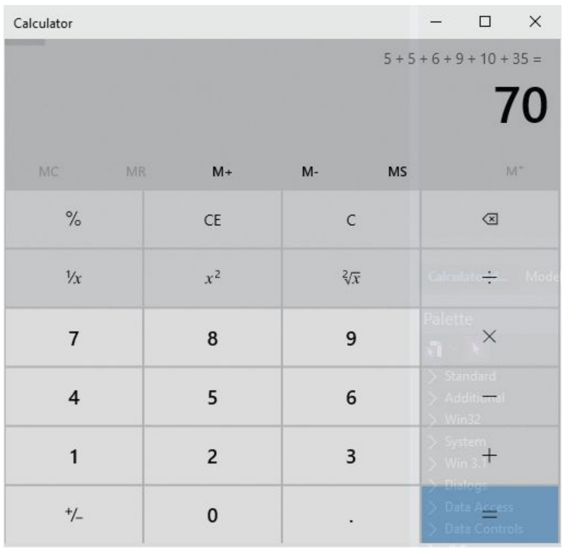


*Figure 2 - Windows 10 Calculator Benchmark*

This benchmark application is targeted to Windows specifically. While some frameworks are multi-platform capable, analysis of compiled code on non-Windows platforms will not be accomplished in this paper. This calculator is also a “lightweight” application which limits how much performance differences may be extrapolated to more complex applications. Reference ***Appendix 1*** for the full specification of the calculator benchmark application as it was implemented by independent contractors.

The benchmark application will be analyzed according to the qualitative metrics laid out in section 3.2. The source code of each application will be examined to count total lines of code, the number of developer-typed lines according to their application instructions (one of the contract deliverables), and the number of lines focused on the user interface specifically. Application startup time from local and network storage locations will be tested using AppTimer, a free startup benchmarking tool. Finally, application memory use will be measured using the free MiTeC Task Manager Deluxe tool.

### 3.4.2. Weight Profiles

In order to facilitate the comparison of frameworks that serve similar but not identical purposes, this paper will use a weighted evaluation mechanism similar to that proposed by Rieger and Majchrzak.[15] Each of the 23 metrics will be given a weight between *1* and *7* points, summing to 100.[16] Frameworks will be evaluated in each category and assigned a score between *0* (unsatisfied) and *5* (optimally satisfied).[17] When the metric is a direct, quantitative comparison between frameworks (i.e. startup times), the objectively “winning” framework will score a *5*, the “middle” framework a 3, and the “losing” framework a *1*. Once calculated, the weighted score will fall between *0* and *5* and indicate which framework better satisfies these 23 criteria. See *Table 1* for the metric weights and the emphasis placed on each evaluation category.

[15] Rieger and Majchrzak, 2016, p. 15.
[16] Rieger and Majchrzak, 2016, p. 15.
[17] Rieger and Majchrzak, 2016, p. 15.

| Framework Comparison | | | | |
|---|---|---|---|---|
| **Criterion** | **Weight (%)** | **Delphi** | **WPF** | **Electron** |
| P1: Development Time | 7 | | | |
| P2: UI Design Approach | 7 | | | |
| P3: Developer Environment Tools | 5 | | | |
| P4: Speed Implementation Time | 2 | | | |
| P5: Code Size | 4 | | | |
| P6: App Store Deployment | 4 | | | |
| ***Productivity Total*** | 29% | | | |
| F1: License | 2 | | | |
| F2: Long-term Feasibility | 6 | | | |
| F3: Development Platforms | 2 | | | |
| F4: Testing Support | 6 | | | |
| F5: Tool Extension | 3 | | | |
| F6: Accessibility | 1 | | | |
| F7: IP Security | 4 | | | |
| ***Functionality Total*** | 24% | | | |
| X1: Target Platforms | 7 | | | |
| X2: Project Variety | 3 | | | |
| X3: Scalability | 5 | | | |
| X4: Database Access | 4 | | | |
| X5: Hardware Access | 6 | | | |
| X6: Platform Access | 6 | | | |
| ***Flexibility Total*** | 31% | | | |
| R1: Deployment Requirements | 5 | | | |
| R2: Startup Time | 4 | | | |
| R3: Standing Memory Usage | 4 | | | |
| R4: Peak Memory Usage | 3 | | | |
| ***Performance Total*** | 16% | | | |
| **Weighted Score (5 is best)** | 100% | | | |

*Table 1 - Weighted Criteria Schema*

## 4. Analysis

The benchmark application - a Windows 10 Calculator clone - was recreated in each framework by three Delphi Most Valuable Professionals (MVPs) volunteers, one expert freelance WPF developer, and one expert Electron freelance developer. Furthermore, proposals were received from 16 other WPF developers and 8 other Electron developers to gauge the average development time expected by professionals in each field. Quantitative analysis of each benchmark application and expert-assisted qualitative research resulted in the completed Framework Comparison of *Table 2*. Each section will be discussed in more detail in the following sections and the detailed analysis of each metric can be found in Appendix 2.

| Framework Comparison | | | | |
|---|---|---|---|---|
| **Criterion** | **Weight (%)** | **Delphi** | **WPF** | **Electron** |
| P1: Development Time | 7 | 5 | 1 | 3 |
| P2: UI Design Approach | 7 | 5 | 3 | 3 |
| P3: Developer Environment Tools | 5 | 4 | 4 | 4 |
| P4: Speed Implementation Time | 2 | 3 | 1 | 5 |
| P5: Code Size | 4 | 3 | 1 | 5 |
| P6: App Store Deployment | 4 | 5 | 1 | 2 |
| ***Productivity Total*** | 29% | 1.28 | 0.58 | 1 |
| F1: License | 2 | 3 | 3 | 5 |
| F2: Long-term Feasibility | 6 | 5 | 4 | 3 |
| F3: Development Platforms | 2 | 2 | 2 | 5 |
| F4: Testing Support | 6 | 4 | 3 | 4 |
| F5: Tool Extension | 3 | 5 | 3 | 3 |
| F6: Accessibility | 1 | 4 | 4 | 5 |
| F7: IP Security | 4 | 5 | 2 | 1 |
| ***Functionality Total*** | 24% | 1.03 | 0.73 | 0.8 |
| X1: Target Platforms | 7 | 5 | 2 | 3 |
| X2: Project Variety | 3 | 5 | 4 | 3 |
| X3: Scalability | 5 | 5 | 5 | 5 |
| X4: Database Access | 4 | 5 | 5 | 4 |
| X5: Hardware Access | 6 | 5 | 3 | 3 |
| X6: Platform Access | 6 | 5 | 4 | 3 |
| ***Flexibility Total*** | 31% | 1.55 | 1.13 | 1.07 |
| R1: Deployment Requirements | 5 | 5 | 3 | 1 |
| R2: Startup Time | 4 | 5 | 3 | 1 |

| | | | | |
|---|---|---|---|---|
| R3: Standing Memory Usage | 4 | 5 | 1 | 3 |
| R4: Peak Memory Usage | 3 | 5 | 3 | 1 |
| ***Performance Total*** | 16% | 0.8 | 0.4 | 0.24 |
| **Weighted Score (5 is best)** | 100% | **4.66** | **2.84** | **3.11** |

*Table 2 - Scored Evaluation Metrics*

## 4.1. Productivity

Framework productivity was evaluated according to five metrics that sought to capture how a framework and IDE can speed application time-to-market. Productivity scores are found in *Table 3* and development data in *Table 4*.

| **Productivity Comparison** | | | | |
|---|---|---|---|---|
| **Criterion** | **Weight (%)** | **Delphi** | **WPF** | **Electron** |
| P1: Development Time | 7 | 5 | 1 | 3 |
| P2: UI Design Approach | 7 | 5 | 3 | 3 |
| P3: Developer Environment Tools | 5 | 4 | 4 | 4 |
| P4: Speed Implementation Time | 2 | 3 | 1 | 5 |
| P5: Code Size | 4 | 3 | 1 | 5 |
| P6: App Store Deployment | 4 | 5 | 1 | 2 |
| ***Productivity Total*** | 29% | 1.28 | 0.58 | 1 |

*Table 3 - Productivity Scores*

| **Framework Productivity** | | | |
|---|---|---|---|
| | **Delphi** | **WPF** | **Electron** |
| Development Time (hrs) | **4.667** | 30.000 | 10.000 |
| Final Speedrun (hrs) | 1.347 | 2.050 | **0.550** |
| Total Lines of Code | 398 | 680 | **293** |
| Lines of Code for UI | **72** | 383 | 115 |
| UI % of Code | **18%** | 56% | 39% |

*Table 4 - Benchmark Productivity Indicators*

Product time-to-market can make or break a business. This benchmark evaluated "time-to-market", called Development Time, and found that three expert Delphi developers completed their VCL or FMX Win10 calculator clone in an average of 4.66 hours, the Electron application took twice as long, and the WPF application six times as long. This disparity can be partially attributed to requirements each framework imposes for GUI development. Although the Delphi applications were not the shortest programs of the comparison, the lines of code developers wrote for the GUI comprised only 18% of their typed total. The RAD Studio IDE allowed them to rapidly design and initialize the GUI using standard control components in the drag-and-drop Form Designer and property-modification Object Inspector, minimizing the developers' need to initialize the calculator interface in code. In comparison, 39% of the Electron application's developer-written code forms the GUI and 49% of the WPF developer's code supports the calculator GUI. While Microsoft's Visual Studio supports drag-and-drop controls and visual GUI development for WPF, the framework requires more XAML code to make the GUI work than Delphi, slowing the design process. Another factor influencing development time is the amount of code required to connect the calculator logic to the user interface. Requiring 680 total lines of code between three files and in two languages, it isn't a surprise that the WPF application took the last place. Clearly, Delphi's visual-development interface and native control libraries are a substantial asset to initial development productivity, allowing the work to occur more quickly or a more-sophisticated GUI to be developed in the same amount of time.

Speed implementation time - evaluated by a "speedrun" re-implementation - tested the level of effort each framework required to complete a known task and was influenced by code size and IDE tools and features. The Electron application, with the fewest lines of code, was the fastest to rebuild at 35 minutes. The Delphi calculator average was twice as long, owing to slightly more code and the process of visually designing the application. Bringing up the rear was WPF with the largest codebase. Overall, speed implementation time is directly related to code size and shows that Delphi and Electron are more concise than WPF, an advantage for developers who frequently implement similar functions.

A final aspect of product development productivity is the time required to get the application to the user. Delphi scores top marks in this metric. The RAD Studio IDE automates application deployment to the app stores for all major desktop and mobile applications, eliminating the headache of manual deployment and ensuring the process is bug-free and repeatable. WPF and Electron struggle in this regard - WPF cannot be deployed directly to the Microsoft Store without conversion to a different framework and Electron can only deploy to the Microsoft Store with the help of 3rd party tools. Businesses should keep this "last mile" aspect of product development and deployment in mind when selecting a framework for their application.

## 4.2. Functionality

Framework functionality was examined qualitatively through research and conversation with experts in Delphi, WPF, and Electron and sought to analyze the business use of each framework from investing in the technology through long-term maintenance of the products created. Functionality scores are displayed in *Table 5*.

| Functionality Comparison | | | | |
|---|---|---|---|---|
| **Criterion** | **Weight (%)** | **Delphi** | **WPF** | **Electron** |
| F1: License | 2 | 3 | 3 | 5 |
| F2: Long-term Feasibility | 6 | 5 | 4 | 3 |
| F3: Development Platforms | 2 | 2 | 2 | 5 |
| F4: Testing Support | 6 | 4 | 3 | 4 |
| F5: Tool Extension | 3 | 5 | 3 | 3 |
| F6: Accessibility | 1 | 4 | 4 | 5 |
| F7: IP Security | 4 | 5 | 2 | 1 |
| ***Functionality Total*** | 24% | 1.03 | 0.73 | 0.8 |

*Table 5 - Functionality Scores*

When businesses choose Delphi as their development framework, they are investing in a proprietary framework (that includes runtime library source code) with up-front costs and an optional annual update fee. For this price, they gain a stable, backward compatible, and growing framework and can be confident that applications developed today will be supported and maintainable in 2045. Delphi ships with testing software and also gives businesses the opportunity to develop tools and extensions for the framework using the same talent that builds their product (the Delphi IDE is programmed in Delphi). Some drawbacks to Delphi include its Windows-only IDE and limited accessibility support for compiled programs under one of its frameworks, a shortfall that Embarcadero Technologies is working to remedy.

Windows Presentation Foundation with .NET Framework offers businesses an economical framework with the full backing of Microsoft but includes all the challenges Microsoft's choices induce. WPF has a shorter history than Delphi but was open-sourced in 2018, giving the GUI aspect of application development a brighter long-term outlook despite its ties to the proprietary .NET Framework for most Windows development. WPF offers testing libraries through Visual Studio and businesses can enjoy the large 3rd party tool and extension environment but may need to outsource work to build their own extensions or invest in talent for non-WPF languages. WPF offers slightly greater accessibility than Delphi, especially in its browser app deployment. Like Delphi, WPF applications using .NET Framework must be compiled on Windows machines.

Electron is a free, open-source platform offering businesses the opportunity to develop applications from any major operating system. Electron's future is uncertain, however. It is the newest of the three frameworks and still in its honeymoon phase. It lacks a native IDE, giving businesses a choice but also removing some conveniences like integrated compilation and included testing libraries. Businesses developing in-house tools would have a more difficult time with Electron than the other frameworks. Electron provides excellent accessibility options for all major desktop platforms through its Chromium foundation.

Intellectual property protection is fundamentally important to long-term business plans. If a product solves a new problem or utilizes a novel technique, the developers should understand how their choice of framework affects IP vulnerability. Delphi programs compile into platform-native machine code rather than intermediate code. Decompilation using free tools can recover the GUI form but only yields assembly code for the logic. IP security is more tenuous in WPF. Decompiling executable and library files with free tools results in recognizable C# business logic and nearly recognizable XAML text. Finally, Electron has the most significant problem - it gives away source code with each installation by default. Electron application code can be recovered with a simple text editor - a function of how the framework is structured - but can be somewhat obfuscated using 3rd party tools. *Appendix 3* describes available decompiler tools and their results when applied to each framework's calculator application.

Overall, Delphi provides the most assured long-term outlook, best intellectual property security, and easiest in-house customization at the cost of a one-time commercial license purchase. WPF's barrier to entry is lower and it offers better accessibility options but is subject to Microsoft's .NET overhauls, is more difficult to customize, and can be decompiled with ease. Electron is absolutely free and can be developed on each of the three major desktop platforms but pays for that flexibility via its uncertain long-term outlook and by relying on corporate sponsorships and community support for additional development.

## 4.3. Flexibility

Framework flexibility was examined qualitatively through research and conversation with experts in Delphi, WPF, and Electron and sought to analyze the application of each framework to business problems and requirements. Flexibility scores are displayed in *Table 6*.

| Flexibility Comparison | | | | |
|---|---|---|---|---|
| **Criterion** | **Weight (%)** | **Delphi** | **WPF** | **Electron** |
| X1: Target Platforms | 7 | 5 | 2 | 3 |
| X2: Project Variety | 3 | 5 | 4 | 3 |
| X3: Scalability | 5 | 5 | 5 | 5 |
| X4: Database Access | 4 | 5 | 5 | 4 |
| X5: Hardware Access | 6 | 5 | 3 | 3 |
| X6: Platform Access | 6 | 5 | 4 | 3 |
| ***Flexibility Total*** | 31% | 1.55 | 1.13 | 1.07 |

*Table 6 - Flexibility Scores*

Delphi's major advantage over WPF and Electron is that its FMX framework can deploy one body of source code as a binary to any major desktop or mobile platform, maximizing a business's reach to customers and minimizing code duplication and maintenance/upgrade headaches. It can support projects of every size from logic controllers for industrial automation to world-wide inventory management and be developed for every tier from a database-heavy back end to the GUI client-side of an application. Finally, Delphi's standard libraries provide easy access to nearly every database type available and allow developers to access operating system functionality on every platform as well as interact with I/O devices and hardware sensors.

WPF with .NET Framework targets Windows computers directly and provides cross-platform support through a browser deployment from a similar codebase. The framework is primarily geared toward client-side desktop applications but can incorporate business logic in C# for middle-tier or back-end functions and access the ADO .NET Entity Framework for databases. WPF can access Windows operating system functionality and I/O devices through .NET libraries but with managed code after compilation rather than native code.

Electron is an open-source framework targeting all desktop operating systems through its Chromium browser base. It focuses on client-side applications, typically web-centric, but uses node.js for middle-tier and back-end services. Electron provides hardware access from its node.js process and can access some but not all operating system functions via node.js libraries.

After reviewing all three frameworks, Delphi holds the lead in the flexibility category due to its flexible and automated deployment to all major platforms, scalability to every level of development, and visual design system. WPF with .NET Framework is competitive on the Windows platform but lacks the ability to compete on macOS or mobile devices. Finally, Electron has the fewest barriers to entry and the most development tool options but relies heavily on manual deployments, cannot target mobile devices directly by default, is the least scalable, and lacks the same hardware and operating system access of its competitors.

## 4.4. Performance

Delphi, WPF, and Electron were evaluated according to the performance of their deployed applications using startup times, peak and idle memory use, and file numbers and sizes. Performance scores are found in *Table 7.*

| Performance Comparison | | | | |
|---|---|---|---|---|
| **Criterion** | **Weight (%)** | **Delphi** | **WPF** | **Electron** |
| R1: Deployment Requirements | 5 | 5 | 3 | 1 |
| R2: Startup Time | 4 | 5 | 3 | 1 |
| R3: Standing Memory Usage | 4 | 5 | 1 | 3 |
| R4: Peak Memory Usage | 3 | 5 | 3 | 1 |
| ***Performance Total*** | 16% | 0.8 | 0.4 | 0.24 |

*Table 7 - Performance Scores*

| Framework Startup Times (sec) and Deployment Sizes | | | | |
|---|---|---|---|---|
| | **Delphi** | **WPF** | **Electron** | **Win10** |
| Deployed File Size (MB) | 6.4 | **0.1** | 198.0 | **0.3** |
| # Deployed Files | **1** | 2 | 161 | 1 |
| Startup (local) | **0.239** | 0.471 | 0.483 | 0.243 |
| Startup (network) | **0.439** | 0.643 | 0.848 | **0.259** |
| Fastest startup (local) | **0.175** | 0.413 | 0.391 | **0.068** |
| Fastest startup (network | **0.264** | 0.564 | 0.569 | **0.106** |
| Slowest startup (local) | **0.687** | 0.814 | 0.846 | 0.872 |
| Slowest startup (network) | 2.416 | **2.106** | 19.669 | **0.925** |
| Startup Std Dev (local) | 0.070 | 0.075 | **0.054** | 0.127 |
| Startup Std Dev (network) | 0.329 | **0.178** | 1.902 | 0.117 |
| Peak Memory Use (MB) | **30.4** | 50.4 | 57.8 | 64.1 |
| Idle Memory Use (MB) | 21.1 | 37.3 | **20.4** | 36.0 |

*Table 8 - Benchmark Performance Indicators*

Each framework deployed its calculator differently. Delphi created one executable file that averaged 6.4 MB.  WPF created an executable file and library file totaling less than 0.1 MB.  The heavy-weight of the group, Electron produced 161 files totaling 198 MB due to its Chromium browser. Although lighter weight due to its use of the .NET Framework installed on every Windows computer, WPF scored lower than Delphi due to producing two files. In general, one file will be easier to manage than multiple files, can negate the need for an installer or scripts to update the application, and reduces network bandwidth requirements and hard drive use.

*Table 8* shows the average startup times of each framework's application from a local hard drive. The three Delphi applications posted the shortest times, with WPF taking about twice as long and Electron longer still. When started from a networked hard drive, WPF took the lead from Delphi but the small size of both framework applications limits the extrapolation of that data point. A standard deviation analysis identified Electron as the framework with the most consistent local startup times but least consistent networked times. In fact, the slowest Electron network startup time was 19.66 seconds - twenty-three times slower than its slowest local time - indicating that Electron apps would be best deployed locally for consistent user experiences and might pose a significant problem for enterprises with large networked services or remote employees. WPF's slowest time was faster than the other frameworks, consistent with its small standard deviation. Delphi's times varied more than WPF's but its faster average times provide the best user launch experience.

Testing found that the Delphi VCL framework used the least peak and idle memory, settling down to just 3.5 MB. Delphi's FMX framework consumed quite a bit more, peaking at 41 MB and idling at 32 MB. Both Delphi variants posted the lowest peak memory use, followed by WPF and then Electron. Electron edged out the Delphi FMX idle memory use at 20.4 MB but had the highest startup memory requirements of the three frameworks.

## 5. Conclusions

This paper sought to compare Delphi, Windows Presentation Foundation, and Electron - three competing frameworks for modern application development - in the areas of developer productivity, functionality for decision makers, flexibility for product development, and product performance using a benchmark application. Calculator development by experts in each framework, qualitative research, and consultation resulted in several salient conclusions for business decision-makers: First, Delphi and its RAD Studio IDE profoundly enhance development productivity and product time-to-market. Not only that, developing just one codebase to reach every desktop and mobile platform provides businesses advantages through simplified successive releases and product maintenance. Second, WPF with the .NET Framework offers small teams native entry to Windows applications and a solid IDE but struggles to match Delphi's productivity, IP security, and performance while also missing Delphi and Electron's cross-platform features. Lastly, Electron offers a free alternative to Delphi and WPF, familiarity to front-end developers, and cross-platform capability at the cost of IP protection, standard IDE tooling, and application performance.

Overall, the three frameworks this paper evaluated showed their strengths in different areas of product development and performance but Delphi demonstrated consistent strength across each evaluation category and scored 4.66 out of 5 points, outperforming Electron (3.11 points) and WPF with the .NET Framework (2.85 points). Based on this comparison, businesses seeking to build robust products with long lifecycles and wide market reach should strongly consider investing in Delphi, and it's RAD Studio IDE.

## 6. Future Work

This study and its benchmark application examined only three frameworks for Windows and focused on productivity related to GUI design and application performance. Future white papers from Embarcadero will work to round-out the study of these frameworks by examining database support with an RSS reader/PostgreSQL application, website interactions through REST services and APIs using a GitHub Recent Explorer application, operating system support and interaction using a File Browser application, and multi-screen interaction with a Screenshot History application.  Other groups wishing to contribute to this comparison effort should consider unmentioned functions frameworks must handle, incorporate a wider variety of frameworks (Xamarin, Spring, Cordova, etc.), and challenge existing conclusions with new tests and more research.

This paper may undergo several revisions as Embarcadero refines its understanding of the data collected in this study.

## References

Biørn-Hansen, Andreas, Tim A. Majchrzak, and Tor-Morten Grønli. "Progressive web apps: The possible web-native unifier for mobile development." In *International Conference on Web Information Systems and Technologies*, vol. 2, pp. 344-351. SCITEPRESS, 2017.

Dalmasso, Isabelle, Soumya Kanti Datta, Christian Bonnet, and Navid Nikaein. "Survey, comparison and evaluation of cross platform mobile application development tools." In *2013 9th International Wireless Communications and Mobile Computing Conference (IWCMC)*, pp. 323-328. IEEE, 2013.

Delia, Lisandro, Nicolas Galdamez, Pablo Thomas, Leonardo Corbalan, and Patricia Pesado. "Multi-platform mobile application development analysis." In *2015 IEEE 9th International Conference on Research Challenges in Information Science (RCIS)*, pp. 181-186. IEEE, 2015.

Majchrzak, Tim, and Tor-Morten Grønli. "Comprehensive analysis of innovative cross-platform app development frameworks." In *Proceedings of the 50th Hawaii International Conference on System Sciences*. 2017.

Rieger, Christoph, and Tim A. Majchrzak. "Weighted evaluation framework for cross-platform app development approaches." In *EuroSymposium on Systems Analysis and Design*, pp. 18-39. Springer, Cham, 2016.

Willocx, Michiel, Jan Vossaert, and Vincent Naessens. "A quantitative assessment of performance in mobile app development tools." In *2015 IEEE International Conference on Mobile Services*, pp. 454-461. IEEE, 2015.

## Source Data and Community Input

The complete codebase for this comparison paper along with project specifications and notes on individual calculator adherence to the specification are available on GitHub in the public ComparisonResearch/calculator repository.  Embarcadero encourages readers to examine the calculators submitted by contractors and MVPs, compare code and methods, find errors and improvements, and to learn from this project.

**Have a suggestion for improving this paper series? Submit a GitHub repository *Issue*!**

## Contributors

### Embarcadero Technologies, Inc.

| | |
|---|---|
| Marco Cantù | RAD Studio Senior Product Manager |
| Adam Leone | Software Development Intern |
| Jim McKeeth | Chief Developer Advocate & Engineer |
| David Millington | RAD Studio Senior Product Manager |

### Embarcadero Most Valuable Professionals (MVPs)

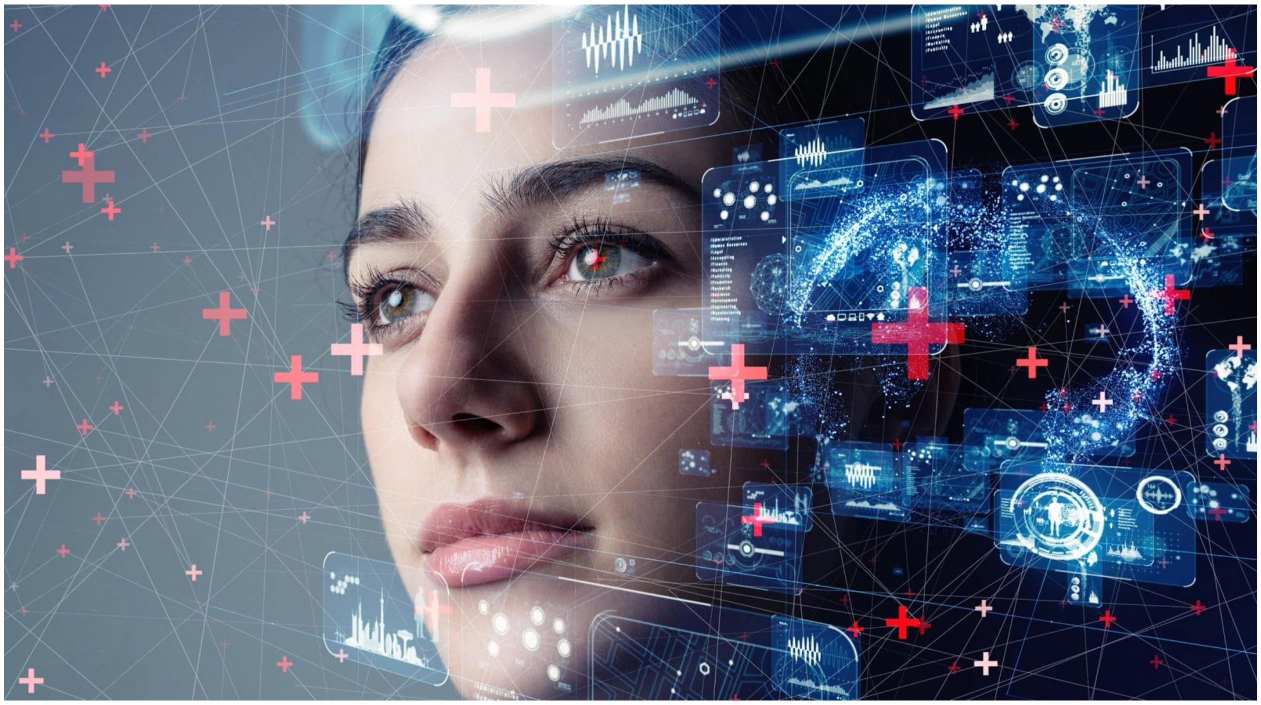
| | |
|---|---|
| Ian Barker | codedotshow.com |
| Bob Calco | Apex Data Solutions |
| Javier Gutiérrez Chamorro | javiergutierrezchamorro.com |
| Olaf Monien | developer-experts.net |
| François Piette | francois-piette.blogspot.com |
| Patrick Prémartin | developpeur-pascal.fr |
| Yılmaz Yörü | yyoru.com |

### Independent Contractors

| | |
|---|---|
| Serhii K. | upwork.com/fl/serhiik |
| Eli M. | upwork.com/freelancers/~015a0a19afc2593d77 |
| Martin P. | github.com/martin-pettersson |
| Dhiraj S. | upwork.com/freelancers/~01139eb7cc53906988 |
| Heru S. | upwork.com/freelancers/~0195227b473e36e942 |
| Victor V. | upwork.com/freelancers/~01393e5253b24c66e9 |

## About Embarcadero Technologies

Embarcadero Technologies, Inc. is a leading provider of award-winning tools for application developers and database professions so they can design systems right, build them faster, and run them better regardless of platform or programming language. Ninety of the Fortune 100 and an active community of more than three million users worldwide rely on Embarcadero products to increase productivity, reduce costs, simplify change management and compliance, and accelerate innovation. Founded in 1993, Embarcadero is headquartered in San Francisco with offices located around the world. Download a free trial at www.embarcadero.com.



# Appendix 1. Benchmark Application Specification

The following specification was provided to the independent contractors and MVPs that developed the benchmark application in Delphi, WPF, and Electron. Embarcadero project managers answered questions as needed, updated the specification with additional details, and strove to keep each application true to the specification.

## Calculator

October 29, 2020

### Overview

The goal of this Calculator project is to build a generic calculator that looks and functions nearly identically to the standard view of the Windows 10 calculator. This is a prototype! The emphasis of this project is on the lessons learned during the development process and documentation phase. The end result does not require a huge amount of polish but should look reasonably similar and function as closely as possible unless otherwise noted in this specification.

Your goal while building the Calculator is to explore the strengths and weaknesses of the framework you are using. The app should be built completely once to figure out your approach. Once complete, build the app again from scratch while recording your screen. Finally, document the app creation process in a step-by-step manner (similar to a recipe - what would someone else need to know to build the app in its entirety - configuration, code, testing, etc.), noting where your framework/language/toolset helps or hinders the build process.

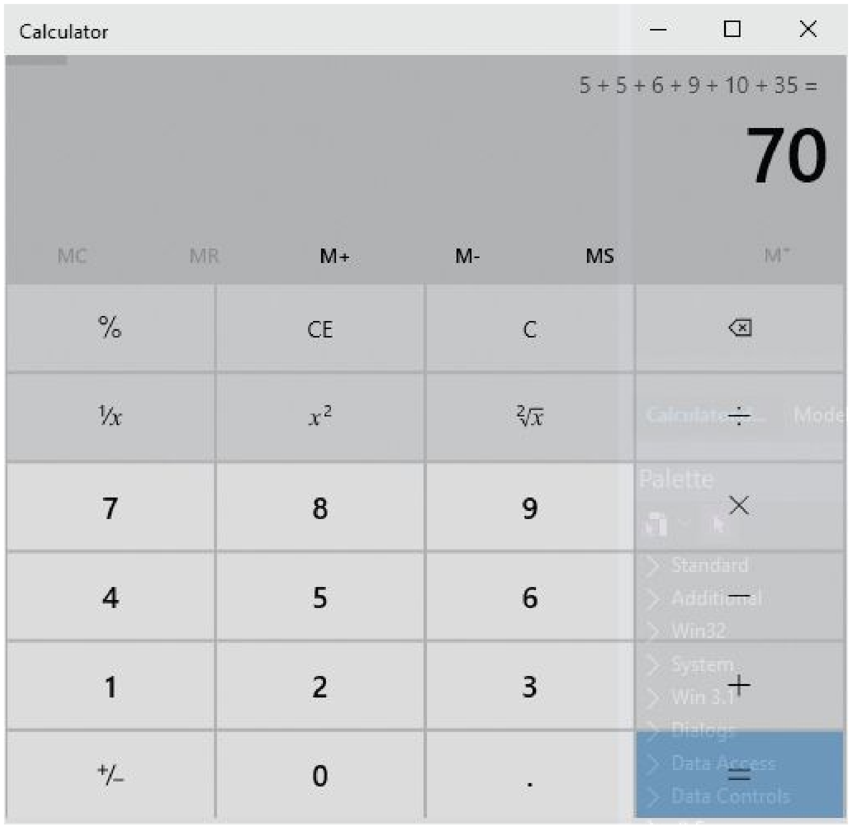
### Key Features

- Responsive Layout
- Semi-Transparent Window
- Basic Math Operations

## Requirements

### Theme

The Calculator should feature a look similar to the below. Note the brighter color palette for number, digit, and sign buttons, darker palette for operator/function buttons, and dark grey palette for the display field. The “equals” button is a cornflower blue shade. All buttons but the equals button change to a darker grey shade when the mouse rolls over and becomes darker again when clicked. The equals button has the same behavior but in blue shades. Note that the window and controls are semi transparent.



## Transparency

The calculator window should be semi-transparent if it will not take more than 5 hours to add that feature.

## Layout

The goal here is to get close to the layout of the Windows calculator. It is broken up into two sections. The top section which shows the equation entered and the number entered/result. The bottom section contains input buttons for digits, mathematical operations, and functions.

## Responsive

The Calculator interface should resize automatically to the size of the window that contains it. Changing the width/height should not result in blank areas or awkward spaces between buttons.

## Memory Buttons

The memory buttons are not required.

## Functions

The calculator must imitate Windows 10 calculator function to the maximum extent possible. The following is not an all-inclusive list of behaviors:

- Does not respect operator precedence. Operands are calculated from left-to-right.
- If a number is entered and “=” pressed, it appears in the Equation View as the number with an equals sign. Ex. “0 =”
- After an operator (+, -, *, /) is clicked, the first operand remains visible until a new number is entered for the second operand.
- Numbers and operators can be “chained” and appear in sequential order in the Equation View. The running result is displayed in the Entry/Answer View (largest font) whenever a new operator is clicked.
- If the equals button is clicked after already solving an equation, the last operation is applied to the current result and the Equation View and Entry/Answer Views are updated.

- The backspace button will remove one digit at a time from the current number in the Entry/Answer View. If the equation has been solved, it will erase the Equation View.
- The ‘CE’ button will completely erase the current number in the Entry/Answer View. If the equation has been solved, it will clear both the Equation View and the Entry/Answer View.
- The ‘C’ button will fully reset the calculator Views whenever pressed.
- The square, square root, and 1/x buttons will immediately act on the current number in the Entry/Answer field so that the result is displayed and will update the equation in the Equation View..
- The change sign button will immediately act on the current number in the Entry/Answer field so that the result is displayed.
- The percent button will behave in accordance with the Microsoft algorithm found here.

## Project Items

1. Complete source code for your working calculator. Include a compiled executable if applicable or instructions for executing the code if not.
2. A video capture of the second build process. This must be in real-time (not sped up) and executed manually (without auto-typing or other speed features). The intent is to get a realistic view of the effort required to make this calculator by a competent programmer.
3. A document with step-by-step instructions that walk someone unfamiliar with your development environment, tools, and language through the process of building this calculator to its full functionality. This document can be a .docx, .pdf, or Google Document format. Markdown usage is preferred.

## Iterative Feedback

Please provide feedback to us during the development process so we can help speed up the development. We have many many years of experience and are here to help you get the project done as fast as possible

# Appendix 2. Detailed Framework Analysis

| Framework | Evaluation | Score | | |
|---|---|---|---|---|
| | | D | W | E |
| **P1: Development Time** | | | | |
| *Delphi* | Three expert Delphi developers completed the Calculator in an average of 4.66 hours using RAD Studio. One developer used his Delphi calculator code and a 3rd party library to create an Electron calculator in 7 minutes, demonstrating the code-reusability of Delphi. | **5** | | |
| *WPF* | One expert WPF developer completed the Calculator in 30 hours using Visual Studio. 16 other WPF estimates were received ranging from 8 hour to 100 hours with a mean of 53 hours and a mode of 80 hours. | | **1** | |
| *Electron* | One expert Electron developer completed the Calculator in 10 hours using Angular for the calculator logic and Electron for the GUI. Eight other Electron estimates were received ranging from 15 to 80 hours with a mean of 47 hours and a mode of 20 hours. | | | **3** |
| **P2: UI Design Approach** | | | | |
| *Delphi* | Delphi's RAD Studio IDE offers a What-You-See-Is-What-You-Get (WYSIWYG) design experience with drag-and-drop components for visual GUI design. The designed GUI can be viewed using native Android/iOS/Windows/macOS styling or custom styles and in a simulated mobile device of varying screen sizes. Components can also be resized and have their properties adjusted in the Object Inspector without touching code, allowing rapid prototyping through visual development. Delphi also offers the ability for a developer to edit the UI using a simple YAML style language definition. | **5** | | |
| *WPF* | WPF in Visual Studio offers a WYSIWYG design experience, immediately reflecting code changes in the GUI representation, and drag-and-drop visual design. Developers can also change object properties through context menus apart from the code. WPF also offers the ability to edit the UI via an XML-like language definition called XAML. | | **3** | |
| *Electron* | Electron lacks a native IDE but can be developed using text editors and command line tools, Electron doesn't include a WYSIWYG design experience or drag-and-drop components by default. The UI can be created using HTML5 and CSS styling. Unless the developer chooses an IDE like Visual Studio, Electron applications must be compiled and run to view the project's GUI. | | | **3** |
| **P3: Developer Environment Tools** | | | | |

| | | | | |
|---|---|---|---|---|
| *Delphi* | Delphi's IDE, RAD Studio, offers a plethora of developer tools including Code Insight (suggestions, completion, etc.), advanced debugger, code formatting, refactoring assistance, keystroke macros, and integration with common software version control systems. RAD Studio provides an Android emulator feature and can tie into an iOS simulator on a macOS machine. RAD Studio is the only IDE available for Delphi and the only method of compiling Delphi projects, however, both the code and UI definitions can be edited using standard text editors. | **4** | | |
| *WPF* | WPF's IDE, Visual Studio, offers many tools including CodeLens (suggestions, completion, etc.), IntelliSense (API suggestions), advanced debugger, integration with version control systems and cloud services, and team collaboration tools. Visual Studio is the only full-fledged IDE for developing WPF applications and includes Blend, a separate system for WPF UI design. WPF could be developed using text editors and command line tools but would be impractical for large projects. | | **4** | |
| *Electron* | Electron applications can be written in code editors such as Visual Studio, Atom, and WebStorm as well as full IDEs. All offer robust features and tools to enhance developer productivity. Electron must be compiled, run, and packaged using the command line - integration with Visual Studio Code hasn't been completed. Third party solutions may be available. | | | **4** |
| **P4: Speed Implementation Time** | | | | |
| *Delphi* | Three expert Delphi developers completed their Calculator speedruns in an average of 1.34 hours. | **3** | | |
| *WPF* | One expert WPF developer completed the Calculator speedrun in 2.05 hours. | | **1** | |
| *Electron* | One expert Electron developer completed the Calculator speedrun 0.55 hours. | | | **5** |
| **P5: Code Size** | | | | |
| *Delphi* | The average Delphi Calculator required 398 lines of typed code. | **3** | | |
| *WPF* | The WPF Calculator required 680 lines of typed code. | | **1** | |
| *Electron* | The Electron calculator required 293 lines of typed code. | | | **5** |
| **P6: App Store Deployment** | | | | |
| *Delphi* | Delphi's VCL framework can deploy directly to the Microsoft Store. Delphi's FMX framework can deploy applications directly to the Microsoft Store, Apple App Store, and Google Play app store for Android. In some cases this deployment results in a platform package such as an APK or IPA which must be uploaded. | **5** | | |

| | | | | |
|---|---|---|---|---|
| *WPF* | WPF applications cannot be directly deployed to any app store. Conversion to the Universal Windows Platform (UWP) enables WPF .NET Framework apps to deploy to the Microsoft Store and conversion to Xamarin provides access to mobile app stores. | | **1** | |
| *Electron* | Electron applications can be packaged for the Microsoft Store but will not be deployed there directly by default. Third party options are available. Electron apps can also be packaged for the Apple App Store but the process lacks automation help. | | | **2** |
| **F1: License** | | | | |
| *Delphi* | Delphi is a proprietary software with three paid license tiers and a free Community Edition and Academic Program. The free tier allows for development as long as annual revenue does not exceed $5,000 USD per year. The first license for full commercial use costs $1,599 USD and the tier that fully unlocks the software suite is priced at $5,999 USD at the time of this writing. An annual subscription is offered at one-third the initial license cost in order to receive updates and new software versions. | **3** | | |
| *WPF* | WPF with .NET Framework is a proprietary environment and generally requires a Visual Studio license for ease-of-use. WPF can also be used with the open source .NET Core. Microsoft's Visual Studio IDE offers licenses with subscription fees between $45/month and $250/month at the time of this writing. First-year subscription fees range from $1,199 to $5,999 with additional years available at a lower cost. Additionally, a community edition of Visual Studio is available for free to small teams. | | **3** | |
| *Electron* | Electron is a free and open-source (MIT license) framework allowing full commercial use without any licenses or fees. It is not tied to an IDE but can be developed in Visual Studio to take advantage of the IDE's tools and 3rd party ecosystem. | | | **5** |
| **F2: Long-term Feasibility** | | | | |
| *Delphi* | Delphi has been growing, maturing, and expanding since 1995. It's development maintains backward compatibility to the degree that a 1995 application can be ported to the current Delphi version with minimal changes. Comprehensive documentation aids maintenance and a full support team is available for upgrade, migration, or troubleshooting help. | **5** | | |
| *WPF* | Released in 2006, WPF has developed along with the .NET framework. It was open-sourced by Microsoft in 2018 and has provided several roadmaps indicating community engagement and growth in the near future. Significant | | **4** | |

| | | | | |
|---|---|---|---|---|
| | .NET changes and Microsoft's shifting design decisions impact the long-term feasibility of WPF. | | | |
| *Electron* | Released in 2013, Electron is actively developed and maintained by GitHub and has rapidly provided support for emerging technologies like Apple Silicon (circa Nov 2020). It lacks the history and stable longevity needed to determine if Electron apps built in 2020 will survive through 2030. | | | **3** |
| **F3: Development Platforms** | | | | |
| *Delphi* | Delphi can only be used on Windows machines. Virtualization solutions such as VMware, Parallels, and Virtual Box with a virtual Windows machine allow Delphi use on other platforms. | **2** | | |
| *WPF* | WPF applications using.NET Framework can be written in Visual Studio on any platform but must be compiled on a Windows machine. | | **2** | |
| *Electron* | Electron can be developed on Windows, macOS, and Linux. | | | **5** |
| **F4: Testing Support** | | | | |
| *Delphi* | Delphi ships with the DUnitX unit testing package but lacks a native integration testing system. Numerous 3rd party unit and integration testing tools are available but may not be free. | **4** | | |
| *WPF* | WPF's IDE, Microsoft Visual Studio, includes a unit testing framework. Open-source projects or testing libraries like XUnit.net and Moq are also available. | | **3** | |
| *Electron* | Electron does not install with a native unit or integration testing package. Open-source projects and libraries are available for both functions. | | | **4** |
| **F5: Tool Extension** | | | | |
| *Delphi* | The RAD Studio IDE for Delphi is written in Delphi. Users can build their own extensions and tools in Delphi, eliminating the need to learn a new language and handle language boundary problems. Additionally, extensions and tools can be built in C++ via the C++Builder side of RAD Studio. | **5** | | |
| *WPF* | Visual Studio, the native WPF IDE, can be extended in a number of ways and in multiple languages. Macros are written in Visual Basic, Add-Ins are written in .NET, and Packages can be written in .NET, C#, C++, or Visual Basic. Because WPF is written in XAML and ties into a C# logical back-end, businesses might not have in-house experience to build tools they need to enhance their development environments without out-sourcing the work or investing in training. | | **3** | |

| | | | | |
|---|---|---|---|---|
| *Electron* | Electron lacks a native IDE but can use plug-ins available in IDEs such as Visual Studio Code. Additional Electron tools might have to be developed in-house from scratch or integrated with a 3rd party tool such as Visual Studio Code. There are a large number of open source projects around tooling and functionality for Electron. | | | **3** |
| **F6: Accessibility** | | | | |
| *Delphi* | Delphi VCL applications are fully accessibility-compatible. Delphi FMX applications are not accessible-friendly in the latest release but work is being done to re-issue a free accessibility package for Windows applications. Both VCL and FMX based applications can be compiled to Win32 and Win64. | **4** | | |
| *WPF* | WPF .NET Framework applications can compile to Win32 or Win64 and have full accessibility compatibility on Windows machines. When compiled as a browser app, accessibility depends on browser implementation. | | **4** | |
| *Electron* | Chromium supports many accessibility tools such as screen readers and magnifiers and is functional on all desktop platforms. Electron provides support for both Win32 and Win64. | | | **5** |
| **F7: IP Security** | | | | |
| *Delphi* | Delphi compiles to native machine code, eliminating much of the source code structure and metadata necessary for accurate decompilation and interpretation. Decompilation using a tool like *DeDe* will provide full details about the UI but only assembly code for the logic/back-end. | **5** | | |
| *WPF* | WPF compiled to a Windows desktop application is converted to .dll and .baml files. Decompilation back to recognizable C# and near-perfect XAML is possible through 3rd party tools. | | **2** | |
| *Electron* | Electron source code is packaged and deployed to the end-user's system. Unless a developer uses 3rd party tools to obfuscate code, the source code can be read verbatim using a simple text editor or by unpacking with a tool like *asar*. | | | **1** |
| **X1: Target Platforms** | | | | |
| *Delphi* | Delphi can compile to native 32-bit or 64-bit code for Windows using the VCL framework and compile to 32-bit or 64-bit code for Windows, macOS, Android, iOS, and Linux using the FMX framework. | **5** | | |
| *WPF* | WPF can compile to native code for Windows and to a browser executable for cross-platform use. | | **2** | |
| *Electron* | Electron packages for cross-platform use within the Chromium browser rather than compiling to native code. | | | **3** |
| **X2: Project Variety** | | | | |

| | | | | |
|---|---|---|---|---|
| *Delphi* | Delphi can be used to create applications on all levels from Windows services to Programmable Systems-on-Chip (PSOC) to enterprise applications with database, UI, and network components. 3rd party tools extend Delphi applications to the web. | **5** | | |
| *WPF* | WPF with the .NET Framework focuses on developing "visually stunning desktop applications". It has access to all Windows .NET functionality, including database access and multimedia tools. | | **4** | |
| *Electron* | Electron applications mimic desktop applications by running in the Chromium browser and are typically web-centric (i.e. collaboration, messaging, etc.). Electron uses node.js for native services, utilities, and back-end applications. | | | **3** |
| **X3: Scalability** | | | | |
| *Delphi* | Delphi applications can be separated according to a chosen design pattern. Delphi supports client, middle-tier, and back-end applications and each tier can be divided and owned by different teams. | **5** | | |
| *WPF* | WPF applications can be developed and tested modularly according to design patterns or with the aid of the open source Prism library. WPF is primarily a client-focused framework but can incorporate C# logic for middle-tier and back-end. | | **5** | |
| *Electron* | Electron can be developed and tested modularly for projects of any size. Electron uses node.js for middle-tier and back-end functions. | | | **5** |
| **X4: Database Access** | | | | |
| *Delphi* | Delphi ships with multiple database libraries that connect to nearly every database type on the market. Database access, queries, and data display are smoothly integrated through components accessible in the free Community Edition and at the first commercial license tier. | **5** | | |
| *WPF* | WPF ships with access to database libraries, including ADO .NET Entity Framework, that enable database connections, queries, and entries through C# code. | | **5** | |
| *Electron* | Electron does not include a native database access library. Multiple open source libraries are available to harness server and server-less databases, including JavaScript implementations. | | | **4** |
| **X5: Hardware Access** | | | | |
| *Delphi* | Delphi's FMX framework includes libraries that allow interaction with a device's peripheral sensors and components regardless of platform. These libraries compile into native code. The Delphi RTL, direct memory access, and other | **5** | | |

| | | | | |
|---|---|---|---|---|
| | low level features give it full access to the hardware platform, including inline assembly code on x86 desktop platforms. | | | |
| *WPF* | WPF .NET Framework can access numerous Windows libraries for sensors, I/O devices, and other peripherals for PCs. WPF's access to hardware is through managed code rather than native code, but there is a native (unmanaged) interface through P/Invoke. This bridge limits some access. | | **3** | |
| *Electron* | Electron can access operating system functions and hardware peripherals through node.js libraries. It's cross-platform Chromium base facilitates high level hardware access on all major desktop platforms. Electron's access to hardware is through managed code rather than native code and can only access features exposed through libraries. | | | **3** |
| **X6: Platform Access** | | | | |
| *Delphi* | Delphi VCL and FMX are fully capable of accessing and using native OS APIs and features on all major desktop and mobile platforms. Delphi applications can push native OS messages and notifications and access such platform functions as storage, contacts, battery status, etc. | **5** | | |
| *WPF* | WPF applications have full access to Windows APIs and can use/initiate Windows OS functions with the .NET Framework. These interactions occur through managed code, not native code. | | **4** | |
| *Electron* | Electron applications are unable to utilize operating system functions without bridging libraries developed with other tools and frameworks. | | | **3** |
| **R1: Deployment Requirements** | | | | |
| *Delphi* | Delphi compiled to one executable binary file averaging 2-8 MB in size. | **5** | | |
| *WPF* | WPF compiled to 2 files that were just 55 KB in size. | | **3** | |
| *Electron* | Electron compiled to 151 files that measured 198 MB in size. | | | **1** |
| **R2: Startup Time** | | | | |
| *Delphi* | The Delphi calculators averaged a startup time of 0.239 seconds from local files and 0.439 seconds from network files with a standard deviation of 0.07 and 0.329 seconds respectively. The slowest startup times were 0.687 seconds locally and 2.416 seconds networked. | **5** | | |
| *WPF* | The WPF calculator averaged a startup time of 0.471 seconds from local files and 0.643 seconds from network files with a standard deviation of 0.075 and 0.178 seconds respectively. The slowest startup times were 0.814 seconds locally and 2.106 seconds networked. WPF was slightly slower than Delphi overall but it's slowest network startup time bested the other two frameworks and network startup time was the most consistent. | | **3** | |

| | | | | |
|---|---|---|---|---|
| *Electron* | The Electron calculator averaged a startup time of 0.483 seconds from local files and 0.848 seconds from network files with a standard deviation of 0.054 and 1.902 seconds respectively. The slowest startup times were 0.846 seconds locally and 19.669 seconds networked. Electron had the most consistent local startup time but was the slowest overall. | | | **1** |
| **R3: Standing Memory Usage** | | | | |
| *Delphi* | The Delphi calculators averaged 21.1 MB standing memory use but this number isn't a clear representation of the framework. Calculators written in the Windows-only Visual Component Library (VCL) used 3.6 MB of memory when idle, far less than the competition. Calculators written in FireMonkey (FMX), the cross-platform library for Android, iOS, macOS, Windows, and Linux, used 32MB of memory when idle. | **5** | | |
| *WPF* | The WPF calculator used 37.3 MB of memory when idle. | | **1** | |
| *Electron* | The Electron calculator consumed 20.4 MB of memory when idle, beating Delphi FMX and WPF but still six times more than Delphi VCL. | | | **3** |
| **R4: Peak Memory Usage** | | | | |
| *Delphi* | The Delphi VCL calculators used 13 MB of memory at their peak. Delphi FMX calculators peaked at 44 MB. | **5** | | |
| *WPF* | The WPF calculator peaked at 50.4 MB. | | **3** | |
| *Electron* | The Electron calculator peaked at 57.8 MB. | | | **1** |

# Appendix 3. Framework Decompilation Analysis

## Overview

The goal of this decompilation exercise was to determine the feasibility of retrieving *both* the UI and the original code from each framework's calculator application using open-source or free tools. The frameworks assessed were Delphi VCL, Delphi FMX, WPF (C#), and Electron (with Angular).

When the Delphi VCL and FMX calculators were decompiled, all UI elements were successfully extracted and the logic code was presented as assembly. This exercise wasn't able to extract function and procedure structure but it may be possible.

Decompiling the WPF calculator yielded the UI elements and mostly recognizable C# code.

The UI elements and Javascript code of the Electron calculator are easily exposed using a standard text editor. The Typescript code was transpiled into Javascript and could not be recovered. Overall, Electron's packaging provided a very limited level of obfuscation.

## Tools

### Delphi

DeDe - one of the most popular Delphi decompilers.

Interactive Delphi Reconstructor - a decompiler for Delphi executables and dynamic libraries.

MiTeC DFM Editor - a standalone editor for Delphi Form files (*.dfm) in both binary and text format.

### WPF

WPF StylesExplorer - a WPF .baml decompiler and tool to explore .baml resources.

Snoop WPF - a tool to spy/browse the visual tree of a running WPF application without the need for a debugger.

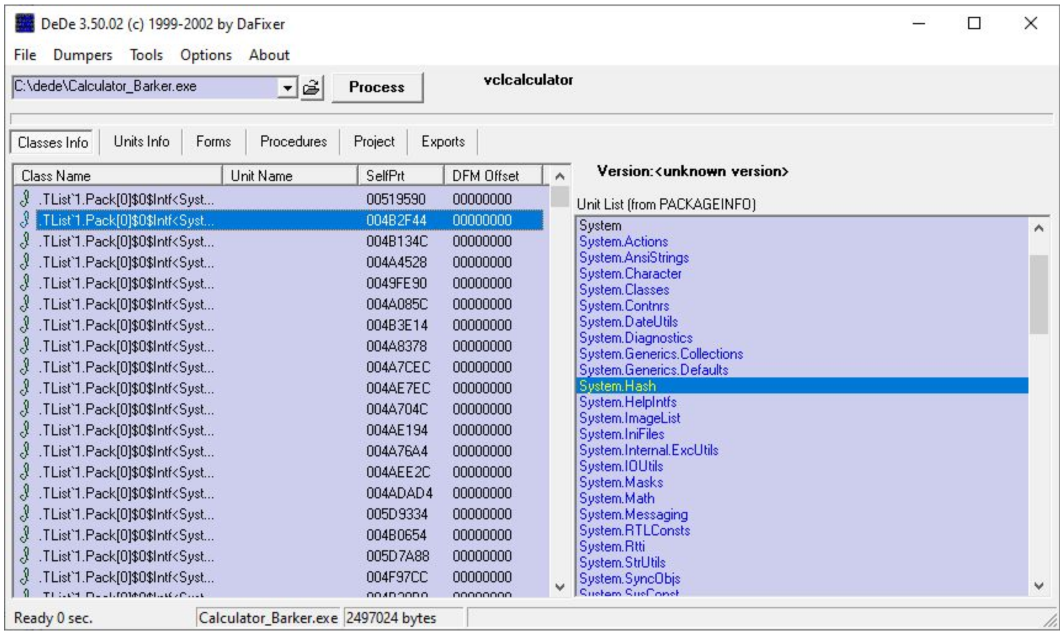
JetBrains dotPeek - a .NET decompiler and assembly browser.

### Electron

TextPad - a general purpose text editor for plaintext files.

## Decompiler Results

### Delphi VCL

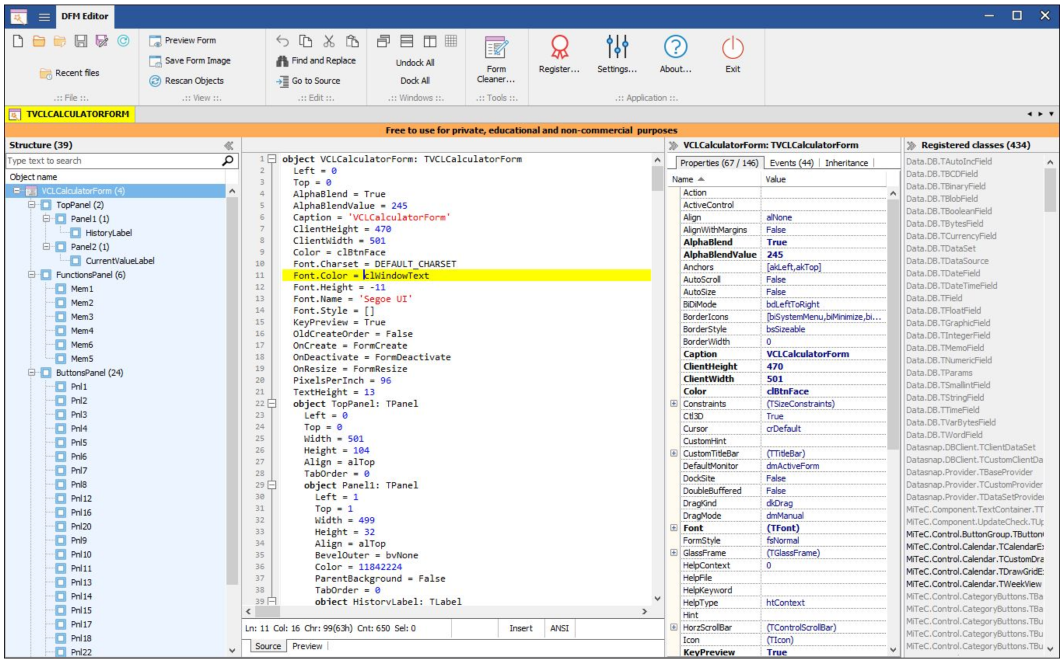

DeDe 3.50.02 (c) 1999-2002 by DaFixer
File
Dumpers
Tools
Options
About
C:\dede\Calculator_Barker.exe
Process
vclcalculator
Classes Info
Units Info
Forms
Procedures
Project
Exports
Version:<unknown version>
Unit List (from PACKAGEINFO)

| Class Name | Unit Name | SelfPrt | DFM Offset |
|---|---|---|---|
| .TList`1.Pack[0]$0$Intf<Syst... | | 00519590 | 00000000 |
| .TList`1.Pack[0]$0$Intf<Syst... | | 004B2F44 | 00000000 |
| .TList`1.Pack[0]$0$Intf<Syst... | | 004B134C | 00000000 |
| .TList`1.Pack[0]$0$Intf<Syst... | | 004A4528 | 00000000 |
| .TList`1.Pack[0]$0$Intf<Syst... | | 0049FE90 | 00000000 |
| .TList`1.Pack[0]$0$Intf<Syst... | | 004A085C | 00000000 |
| .TList`1.Pack[0]$0$Intf<Syst... | | 004B3E14 | 00000000 |
| .TList`1.Pack[0]$0$Intf<Syst... | | 004A8378 | 00000000 |
| .TList`1.Pack[0]$0$Intf<Syst... | | 004A7CEC | 00000000 |
| .TList`1.Pack[0]$0$Intf<Syst... | | 004AE7EC | 00000000 |
| .TList`1.Pack[0]$0$Intf<Syst... | | 004A704C | 00000000 |
| .TList`1.Pack[0]$0$Intf<Syst... | | 004AE194 | 00000000 |
| .TList`1.Pack[0]$0$Intf<Syst... | | 004A76A4 | 00000000 |
| .TList`1.Pack[0]$0$Intf<Syst... | | 004AEE2C | 00000000 |
| .TList`1.Pack[0]$0$Intf<Syst... | | 004ADAD4 | 00000000 |
| .TList`1.Pack[0]$0$Intf<Syst... | | 005D9334 | 00000000 |
| .TList`1.Pack[0]$0$Intf<Syst... | | 004B0654 | 00000000 |
| .TList`1.Pack[0]$0$Intf<Syst... | | 005D7A88 | 00000000 |
| .TList`1.Pack[0]$0$Intf<Syst... | | 004F97CC | 00000000 |

```
System
System.Actions
System.AnsiStrings
System.Character
System.Classes
System.Contnrs
System.DateUtils
System.Diagnostics
System.Generics.Collections
System.Generics.Defaults
System.Hash
System.HelpIntfs
System.ImageList
System.IniFiles
System.Internal.ExcUtils
System.IOUtils
System.Masks
System.Math
System.Messaging
System.RTLConsts
System.Rtti
System.StrUtils
System.SyncObjs
```

```
Ready 0 sec.    Calculator_Barker.exe    2497024 bytes
```

Table 1 - DeDe Decompilation of Delphi VCL



Free to use for private, educational and non-commercial purposes

```
VCLCalculatorForm (4)
  TopPanel (2)
    Panel1 (1)
      HistoryLabel
    Panel2 (1)
      CurrentValueLabel
  FunctionsPanel (6)
    Mem1
    Mem2
    Mem3
    Mem4
    Mem6
    Mem5
  ButtonsPanel (24)
    Pnl1
    Pnl2
    Pnl3
    Pnl4
    Pnl5
    Pnl6
    Pnl7
    Pnl8
    Pnl12
    Pnl16
    Pnl20
    Pnl9
    Pnl10
    Pnl11
    Pnl13
    Pnl14
    Pnl15
    Pnl17
    Pnl18
    Pnl22
```

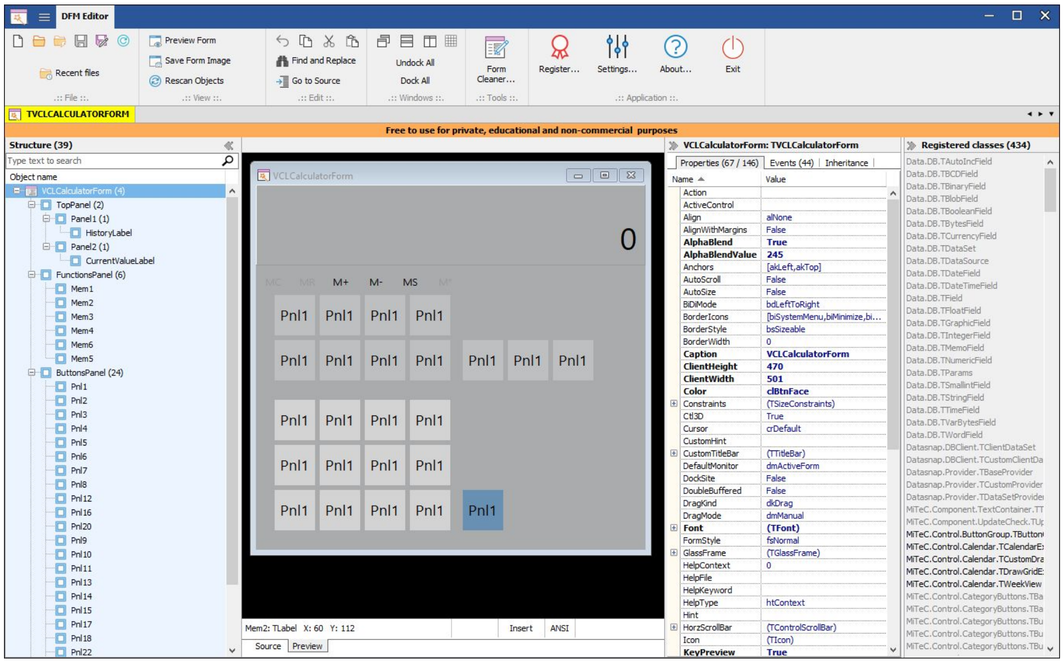
```
object VCLCalculatorForm: TVCLCalculatorForm
  Left = 0
  Top = 0
  AlphaBlend = True
  AlphaBlendValue = 245
  Caption = 'VCLCalculatorForm'
  ClientHeight = 470
  ClientWidth = 501
  Color = clBtnFace
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -11
  Font.Name = 'Segoe UI'
  Font.Style = []
  KeyPreview = True
  OldCreateOrder = False
  OnCreate = FormCreate
  OnDeactivate = FormDeactivate
  OnResize = FormResize
  PixelsPerInch = 96
  TextHeight = 13
  object TopPanel: TPanel
    Left = 0
    Top = 0
    Width = 501
    Height = 104
    Align = alTop
    TabOrder = 0
    object Panel1: TPanel
      Left = 1
      Top = 1
      Width = 499
      Height = 32
      Align = alTop
      BevelOuter = bvNone
      Color = 11842224
      ParentBackground = False
      TabOrder = 0
      object HistoryLabel: TLabel
```

```
Ln: 11 Col: 16 Chr: 99(63h) Cnt: 650 Sel: 0    Insert    ANSI
```

| Name | Value |
|---|---|
| Action | |
| ActiveControl | |
| Align | alNone |
| AlignWithMargins | False |
| **AlphaBlend** | **True** |
| **AlphaBlendValue** | **245** |
| Anchors | [akLeft,akTop] |
| AutoScroll | False |
| AutoSize | False |
| BiDiMode | bdLeftToRight |
| BorderIcons | [biSystemMenu,biMinimize,bi... |
| BorderStyle | bsSizeable |
| BorderWidth | 0 |
| **Caption** | **VCLCalculatorForm** |
| **ClientHeight** | **470** |
| **ClientWidth** | **501** |
| **Color** | **clBtnFace** |
| Constraints | (TSizeConstraints) |
| Ctl3D | True |
| Cursor | crDefault |
| CustomHint | |
| CustomTitleBar | (TTitleBar) |
| DefaultMonitor | dmActiveForm |
| DockSite | False |
| DoubleBuffered | False |
| DragKind | dkDrag |
| DragMode | dmManual |
| **Font** | **(TFont)** |
| FormStyle | fsNormal |
| GlassFrame | (TGlassFrame) |
| HelpContext | 0 |
| HelpFile | |
| HelpKeyword | |
| HelpType | htContext |
| Hint | |
| HorzScrollBar | (TControlScrollBar) |
| Icon | (TIcon) |
| **KeyPreview** | **True** |

```
Data.DB.TAutoIncField
Data.DB.TBCDField
Data.DB.TBinaryField
Data.DB.TBlobField
Data.DB.TBooleanField
Data.DB.TBytesField
Data.DB.TCurrencyField
Data.DB.TDataSet
Data.DB.TDataSource
Data.DB.TDateField
Data.DB.TDateTimeField
Data.DB.TField
Data.DB.TFloatField
Data.DB.TGraphicField
Data.DB.TIntegerField
Data.DB.TMemoField
Data.DB.TNumericField
Data.DB.TParams
Data.DB.TSmallintField
Data.DB.TStringField
Data.DB.TTimeField
Data.DB.TVarBytesField
Data.DB.TWordField
Datasnap.DBClient.TClientDataSet
Datasnap.DBClient.TCustomClientDa
Datasnap.Provider.TBaseProvider
Datasnap.Provider.TCustomProvider
Datasnap.Provider.TDataSetProvide
MiTeC.Component.TextContainer.TT
MiTeC.Component.UpdateCheck.TUp
MiTeC.Control.ButtonGroup.TButton
MiTeC.Control.Calendar.TCalendarE
MiTeC.Control.Calendar.TCustomDra
MiTeC.Control.Calendar.TDrawGridE
MiTeC.Control.Calendar.TWeekView
MiTeC.Control.CategoryButtons.TBa
MiTeC.Control.CategoryButtons.TBa
MiTeC.Control.CategoryButtons.TBu
MiTeC.Control.CategoryButtons.TBu
MiTeC.Control.CategoryButtons.TBu
```

Table 2 - DFM Editor GUI Code View of Delphi VCL

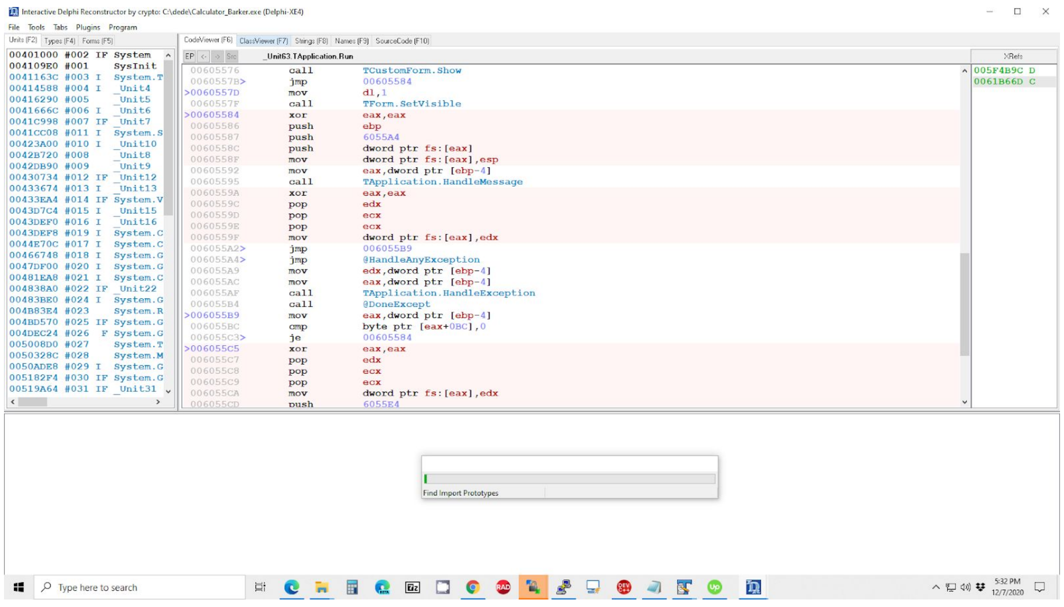


VCLCalculatorForm: TVCLCalculatorForm

Properties (67 / 146) | Events (44) | Inheritance

| Name | Value |
|---|---|
| Action | |
| ActiveControl | |
| Align | alNone |
| AlignWithMargins | False |
| **AlphaBlend** | **True** |
| **AlphaBlendValue** | **245** |
| Anchors | [akLeft,akTop] |
| AutoScroll | False |
| AutoSize | False |
| BiDiMode | bdLeftToRight |
| BorderIcons | [biSystemMenu,biMinimize,bi... |
| BorderStyle | bsSizeable |
| BorderWidth | 0 |
| **Caption** | **VCLCalculatorForm** |
| **ClientHeight** | **470** |
| **ClientWidth** | **501** |
| **Color** | **clBtnFace** |
| Constraints | (TSizeConstraints) |
| Ctl3D | True |
| Cursor | crDefault |
| CustomHint | |
| CustomTitleBar | (TTitleBar) |
| DefaultMonitor | dmActiveForm |
| DockSite | False |
| DoubleBuffered | False |
| DragKind | dkDrag |
| DragMode | dmManual |
| **Font** | **(TFont)** |
| FormStyle | fsNormal |
| GlassFrame | (TGlassFrame) |
| HelpContext | 0 |
| HelpFile | |
| HelpKeyword | |
| HelpType | htContext |
| Hint | |
| HorzScrollBar | (TControlScrollBar) |
| Icon | (TIcon) |
| **KeyPreview** | **True** |

Table 3 - DFM Editor GUI Design View of Delphi VCL


Interactive Delphi Reconstructor by crypto: C:\dede\Calculator_Barker.exe (Delphi-XE4)
File Tools Tabs Plugins Program
Units (F2) Types (F4) Forms (F5)
CodeViewer (F6) ClassViewer (F7) Strings (F8) Names (F9) SourceCode (F10)
Find Import Prototypes
Type here to search

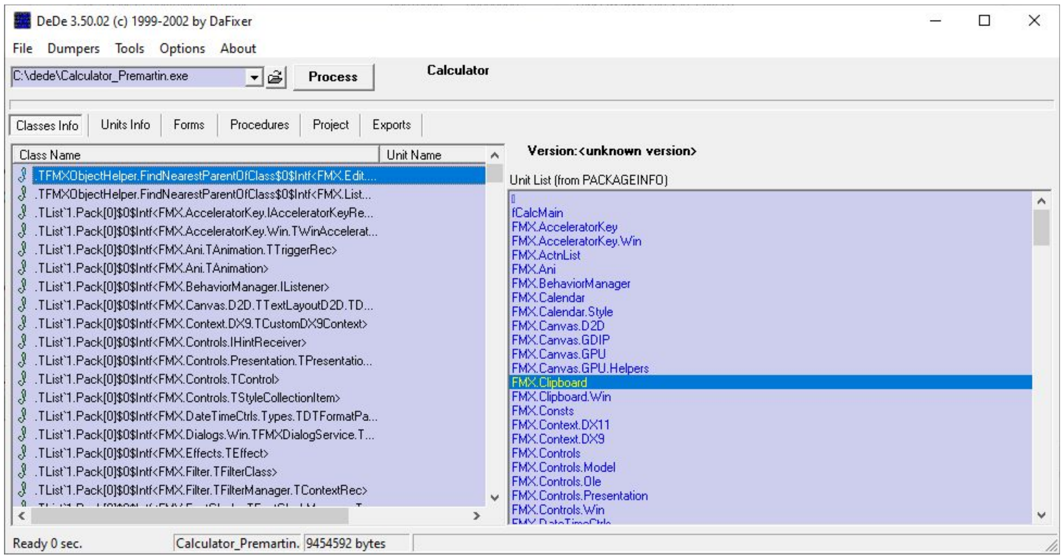
```
00401000 #002 IF System
004109E0 #001    SysInit
0041163C #003 I  System.T
00414588 #004 I  _Unit4
00416290 #005    _Unit5
0041666C #006 I  _Unit6
0041C998 #007 IF _Unit7
0041CC08 #011 I  System.S
00423A00 #010 I  _Unit10
0042B720 #008    _Unit8
0042DB90 #009    _Unit9
00430734 #012 IF _Unit12
00433674 #013 I  _Unit13
00433EA4 #014 IF System.V
0043D7C4 #015 I  _Unit15
0043DEF0 #016 I  _Unit16
0043DEF8 #019 I  System.C
0044E70C #017 I  System.C
00466748 #018 I  System.G
0047DF00 #020 I  System.G
00481EA8 #021 I  System.C
004838A0 #022 IF _Unit22
00483BE0 #024 I  System.G
004B83E4 #023    System.R
004BD570 #025 IF System.G
004DEC24 #026  F System.G
005008D0 #027    System.T
0050328C #028    System.M
0050ADE8 #029 I  System.G
005182F4 #030 IF System.G
00519A64 #031 IF _Unit31
```

```
_Unit63.TApplication.Run
 00605576     call    TCustomForm.Show
 0060557B>    jmp     00605584
>0060557D     mov     dl,1
 0060557F     call    TForm.SetVisible
>00605584     xor     eax,eax
 00605586     push    ebp
 00605587     push    6055A4
 0060558C     push    dword ptr fs:[eax]
 0060558F     mov     dword ptr fs:[eax],esp
 00605592     mov     eax,dword ptr [ebp-4]
 00605595     call    TApplication.HandleMessage
 0060559A     xor     eax,eax
 0060559C     pop     edx
 0060559D     pop     ecx
 0060559E     pop     ecx
 0060559F     mov     dword ptr fs:[eax],edx
 006055A2>    jmp     006055B9
 006055A4>    jmp     @HandleAnyException
 006055A9     mov     edx,dword ptr [ebp-4]
 006055AC     mov     eax,dword ptr [ebp-4]
 006055AF     call    TApplication.HandleException
 006055B4     call    @DoneExcept
>006055B9     mov     eax,dword ptr [ebp-4]
 006055BC     cmp     byte ptr [eax+0BC],0
 006055C3>    je      00605584
>006055C5     xor     eax,eax
 006055C7     pop     edx
 006055C8     pop     ecx
 006055C9     pop     ecx
 006055CA     mov     dword ptr fs:[eax],edx
 006055CD     push    6055E4
```

```
005F4B9C D
0061B66D C
```

Table 4 - Delphi VCL Assembly Code Generated by IDR

## Delphi FMX


DeDe 3.50.02 (c) 1999-2002 by DaFixer

C:\dede\Calculator_Premartin.exe — Process — Calculator

Version:<unknown version>

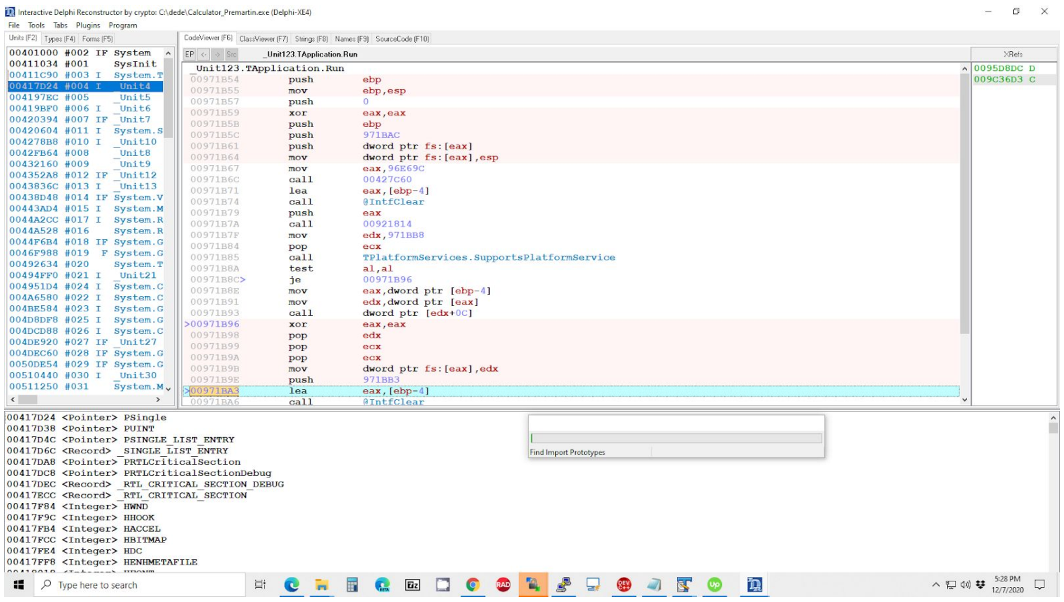
| Class Name | Unit Name |
|---|---|
| .TFMXObjectHelper.FindNearestParentOfClass$0$Intf<FMX.Edit.... | |
| .TFMXObjectHelper.FindNearestParentOfClass$0$Intf<FMX.List... | |
| .TList`1.Pack[0]$0$Intf<FMX.AcceleratorKey.IAcceleratorKeyRe... | |
| .TList`1.Pack[0]$0$Intf<FMX.AcceleratorKey.Win.TWinAccelerat... | |
| .TList`1.Pack[0]$0$Intf<FMX.Ani.TAnimation.TTriggerRec> | |
| .TList`1.Pack[0]$0$Intf<FMX.Ani.TAnimation> | |
| .TList`1.Pack[0]$0$Intf<FMX.BehaviorManager.IListener> | |
| .TList`1.Pack[0]$0$Intf<FMX.Canvas.D2D.TTextLayoutD2D.TD... | |
| .TList`1.Pack[0]$0$Intf<FMX.Context.DX9.TCustomDX9Context> | |
| .TList`1.Pack[0]$0$Intf<FMX.Controls.IHintReceiver> | |
| .TList`1.Pack[0]$0$Intf<FMX.Controls.Presentation.TPresentatio... | |
| .TList`1.Pack[0]$0$Intf<FMX.Controls.TControl> | |
| .TList`1.Pack[0]$0$Intf<FMX.Controls.TStyleCollectionItem> | |
| .TList`1.Pack[0]$0$Intf<FMX.DateTimeCtrls.Types.TDTFormatPa... | |
| .TList`1.Pack[0]$0$Intf<FMX.Dialogs.Win.TFMXDialogService.T... | |
| .TList`1.Pack[0]$0$Intf<FMX.Effects.TEffect> | |
| .TList`1.Pack[0]$0$Intf<FMX.Filter.TFilterClass> | |
| .TList`1.Pack[0]$0$Intf<FMX.Filter.TFilterManager.TContextRec> | |

Unit List (from PACKAGEINFO)

```
fCalcMain
FMX.AcceleratorKey
FMX.AcceleratorKey.Win
FMX.ActnList
FMX.Ani
FMX.BehaviorManager
FMX.Calendar
FMX.Calendar.Style
FMX.Canvas.D2D
FMX.Canvas.GDIP
FMX.Canvas.GPU
FMX.Canvas.GPU.Helpers
FMX.Clipboard
FMX.Clipboard.Win
FMX.Consts
FMX.Context.DX11
FMX.Context.DX9
FMX.Controls
FMX.Controls.Model
FMX.Controls.Ole
FMX.Controls.Presentation
FMX.Controls.Win
```

Ready 0 sec. | Calculator_Premartin. | 9454592 bytes

Table 5 - DeDe Decompilation of Delphi FMX


Interactive Delphi Reconstructor by crypto: C:\dede\Calculator_Premartin.exe (Delphi-XE4)

```
00401000 #002 IF System
00411034 #001    SysInit
00411C90 #003 I  System.T
00417D24 #004 I  _Unit4
004197EC #005    _Unit5
00419BF0 #006 I  _Unit6
00420394 #007 IF _Unit7
00420604 #011 I  System.S
004278B8 #010 I  _Unit10
0042FB64 #008    _Unit8
00432160 #009    _Unit9
004352A8 #012 IF _Unit12
0043836C #013 I  _Unit13
00438D48 #014 IF System.V
00443AD4 #015 I  System.M
0044A2CC #017 I  System.R
0044A528 #016    System.R
0044F6B4 #018 IF System.G
0046F988 #019  F System.G
00492634 #020    System.T
00494FF0 #021 I  _Unit21
004951D4 #024 I  System.C
004A6580 #022 I  System.C
004BE584 #023 I  System.G
004D8DF8 #025 I  System.G
004DCD88 #026 I  System.C
004DE920 #027 IF _Unit27
004DEC60 #028 IF System.G
0050DE54 #029 IF System.G
00510440 #030 I  _Unit30
00511250 #031    System.M
```

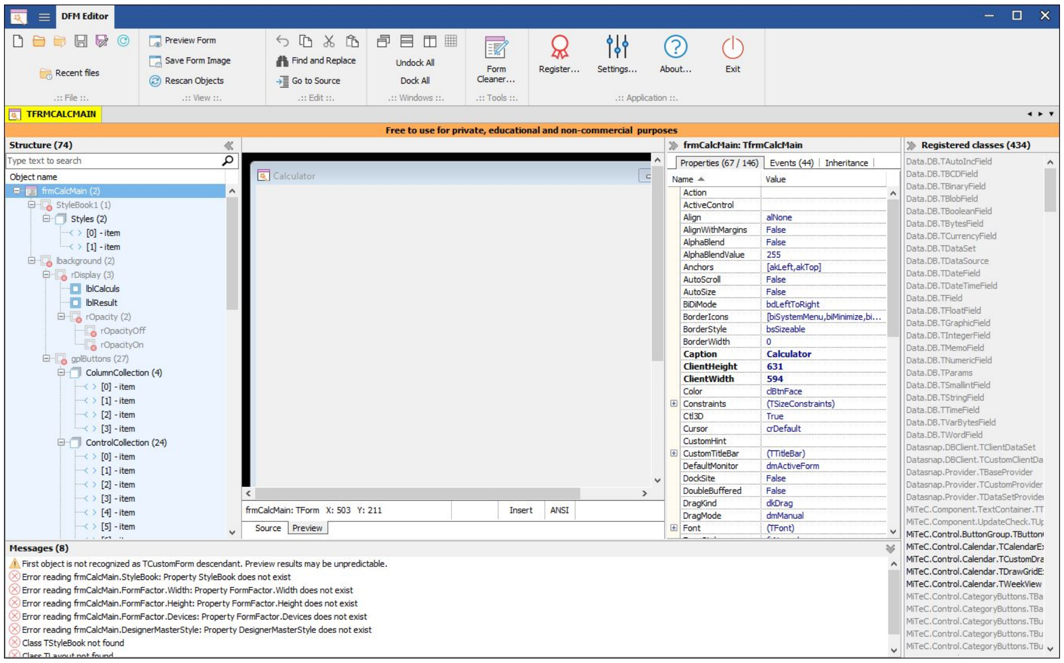
```
_Unit123.TApplication.Run
00971B54        push     ebp
00971B55        mov      ebp,esp
00971B57        push     0
00971B59        xor      eax,eax
00971B5B        push     ebp
00971B5C        push     971BAC
00971B61        push     dword ptr fs:[eax]
00971B64        mov      dword ptr fs:[eax],esp
00971B67        mov      eax,96E69C
00971B6C        call     00427C60
00971B71        lea      eax,[ebp-4]
00971B74        call     @IntfClear
00971B79        push     eax
00971B7A        call     00921814
00971B7F        mov      edx,971BB8
00971B84        pop      ecx
00971B85        call     TPlatformServices.SupportsPlatformService
00971B8A        test     al,al
00971B8C>       je       00971B96
00971B8E        mov      eax,dword ptr [ebp-4]
00971B91        mov      edx,dword ptr [eax]
00971B93        call     dword ptr [edx+0C]
>00971B96       xor      eax,eax
00971B98        pop      edx
00971B99        pop      ecx
00971B9A        pop      ecx
00971B9B        mov      dword ptr fs:[eax],edx
00971B9E        push     971BB3
>00971BA3       lea      eax,[ebp-4]
00971BA6        call     @IntfClear
```

```
XRefs
0095D8DC D
009C36D3 C
```

```
00417D24 <Pointer> PSingle
00417D38 <Pointer> PUINT
00417D4C <Pointer> PSINGLE_LIST_ENTRY
00417D6C <Record> _SINGLE_LIST_ENTRY
00417DA8 <Pointer> PRTLCriticalSection
00417DC8 <Pointer> PRTLCriticalSectionDebug
00417DEC <Record> _RTL_CRITICAL_SECTION_DEBUG
00417ECC <Record> _RTL_CRITICAL_SECTION
00417F84 <Integer> HWND
00417F9C <Integer> HHOOK
00417FB4 <Integer> HACCEL
00417FCC <Integer> HBITMAP
00417FE4 <Integer> HDC
00417FF8 <Integer> HENHMETAFILE
```

File Tools Tabs Plugins Program
Units (F2) Types (F4) Forms (F5)
CodeViewer (F6) ClassViewer (F7) Strings (F8) Names (F9) SourceCode (F10)
EP Src
Find Import Prototypes
Type here to search
5:28 PM
12/7/2020

Table 6 - Delphi FMX Assembly Code Generated by IDR



Free to use for private, educational and non-commercial purposes

Structure (74)

```
frmCalcMain (2)
  StyleBook1 (1)
    Styles (2)
      [0] - item
      [1] - item
  lbackground (2)
    rDisplay (3)
      lblCalculs
      lblResult
    rOpacity (2)
      rOpacityOff
      rOpacityOn
    gplButtons (27)
      ColumnCollection (4)
        [0] - item
        [1] - item
        [2] - item
        [3] - item
      ControlCollection (24)
        [0] - item
        [1] - item
        [2] - item
        [3] - item
        [4] - item
        [5] - item
```

frmCalcMain: TForm X: 503 Y: 211 Insert ANSI

frmCalcMain: TfrmCalcMain — Properties (67 / 146) | Events (44) | Inheritance

| Name | Value |
|---|---|
| Action | |
| ActiveControl | |
| Align | alNone |
| AlignWithMargins | False |
| AlphaBlend | False |
| AlphaBlendValue | 255 |
| Anchors | [akLeft,akTop] |
| AutoScroll | False |
| AutoSize | False |
| BiDiMode | bdLeftToRight |
| BorderIcons | [biSystemMenu,biMinimize,bi… |
| BorderStyle | bsSizeable |
| BorderWidth | 0 |
| **Caption** | **Calculator** |
| **ClientHeight** | **631** |
| **ClientWidth** | **594** |
| Color | clBtnFace |
| Constraints | (TSizeConstraints) |
| Ctl3D | True |
| Cursor | crDefault |
| CustomHint | |
| CustomTitleBar | (TTitleBar) |
| DefaultMonitor | dmActiveForm |
| DockSite | False |
| DoubleBuffered | False |
| DragKind | dkDrag |
| DragMode | dmManual |
| Font | (TFont) |

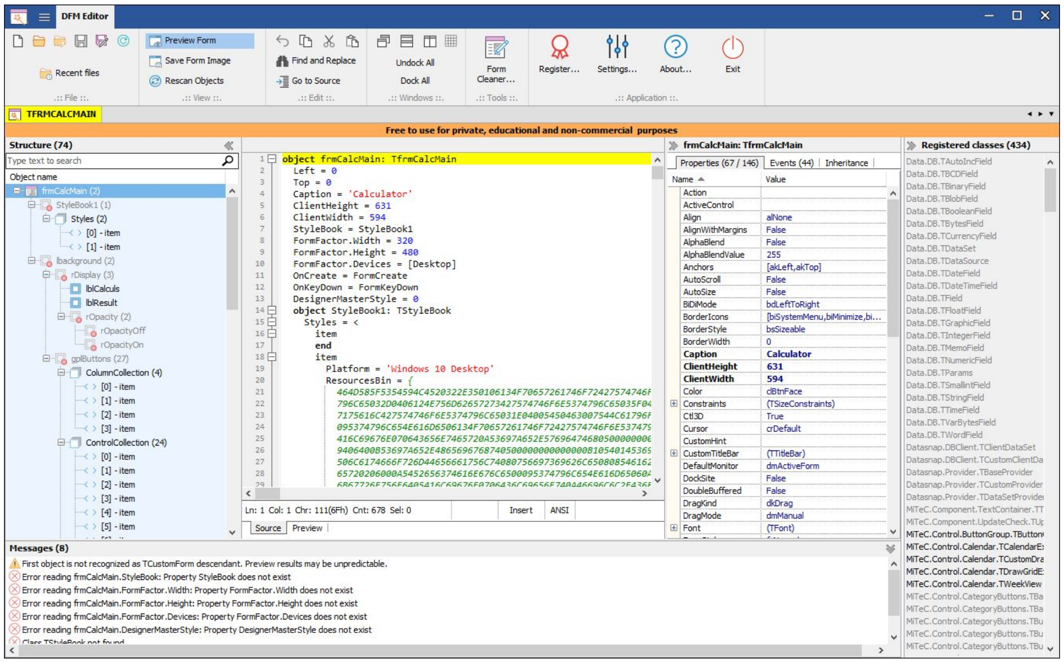
Registered classes (434)

```
Data.DB.TAutoIncField
Data.DB.TBCDField
Data.DB.TBinaryField
Data.DB.TBlobField
Data.DB.TBooleanField
Data.DB.TBytesField
Data.DB.TCurrencyField
Data.DB.TDataSet
Data.DB.TDataSource
Data.DB.TDateField
Data.DB.TDateTimeField
Data.DB.TField
Data.DB.TFloatField
Data.DB.TGraphicField
Data.DB.TIntegerField
Data.DB.TMemoField
Data.DB.TNumericField
Data.DB.TParams
Data.DB.TSmallintField
Data.DB.TStringField
Data.DB.TTimeField
Data.DB.TVarBytesField
Data.DB.TWordField
Datasnap.DBClient.TClientDataSet
Datasnap.DBClient.TCustomClientDa
Datasnap.Provider.TBaseProvider
Datasnap.Provider.TCustomProvider
Datasnap.Provider.TDataSetProvide
MiTeC.Component.TextContainer.TT
MiTeC.Component.UpdateCheck.TUp
MiTeC.Control.ButtonGroup.TButton
MiTeC.Control.Calendar.TCalendarE
MiTeC.Control.Calendar.TCustomDra
MiTeC.Control.Calendar.TDrawGridE
MiTeC.Control.Calendar.TWeekView
MiTeC.Control.CategoryButtons.TBa
MiTeC.Control.CategoryButtons.TBa
MiTeC.Control.CategoryButtons.TBu
MiTeC.Control.CategoryButtons.TBu
MiTeC.Control.CategoryButtons.TBu
```

Messages (8)

```
First object is not recognized as TCustomForm descendant. Preview results may be unpredictable.
Error reading frmCalcMain.StyleBook: Property StyleBook does not exist
Error reading frmCalcMain.FormFactor.Width: Property FormFactor.Width does not exist
Error reading frmCalcMain.FormFactor.Height: Property FormFactor.Height does not exist
Error reading frmCalcMain.FormFactor.Devices: Property FormFactor.Devices does not exist
Error reading frmCalcMain.DesignerMasterStyle: Property DesignerMasterStyle does not exist
Class TStyleBook not found
```

Table 7 - DFM Editor GUI Design View of Delphi FMX



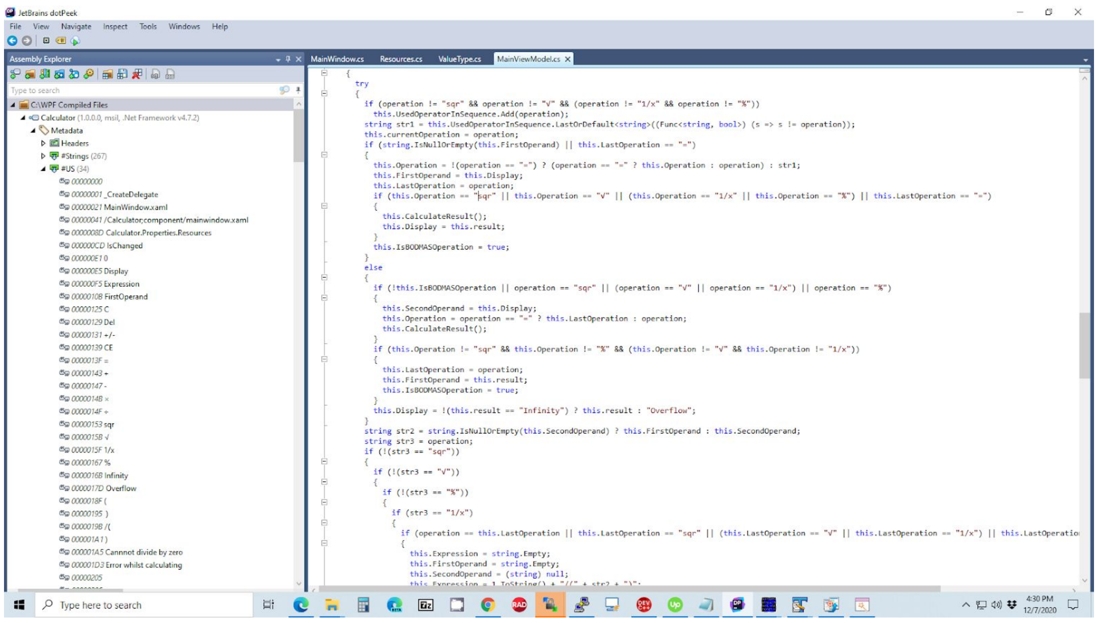
```
object frmCalcMain: TfrmCalcMain
  Left = 0
  Top = 0
  Caption = 'Calculator'
  ClientHeight = 631
  ClientWidth = 594
  StyleBook = StyleBook1
  FormFactor.Width = 320
  FormFactor.Height = 480
  FormFactor.Devices = [Desktop]
  OnCreate = FormCreate
  OnKeyDown = FormKeyDown
  DesignerMasterStyle = 0
  object StyleBook1: TStyleBook
    Styles = <
      item
      end
      item
        Platform = 'Windows 10 Desktop'
        ResourcesBin = {
          464D585F5354594C4520322E350106134F70657261746F72427574746F
          796C65032D0406124E756D626572734275747446F6E5374796C65035F04
          7175616C427574746F6E5374796C65031E040054504630007544C61796F
          095374796C654E616D6506134F70657261746F72427574746F6E537479
          416C69676E070643656E7465720A53697A652E57696474680500000000
          9406400B53697A652E486569676874050000000000000000B1054014536
          506C6174666F726D44656661756C7408075669736962 6C650808546162
          65720206000A5452656374616E676C6500095374796C654E616D65060A
```

Ln: 1 Col: 1 Chr: 111(6Fh) Cnt: 678 Sel: 0 Insert ANSI

Messages (8)
First object is not recognized as TCustomForm descendant. Preview results may be unpredictable.
Error reading frmCalcMain.StyleBook: Property StyleBook does not exist
Error reading frmCalcMain.FormFactor.Width: Property FormFactor.Width does not exist
Error reading frmCalcMain.FormFactor.Height: Property FormFactor.Height does not exist
Error reading frmCalcMain.FormFactor.Devices: Property FormFactor.Devices does not exist
Error reading frmCalcMain.DesignerMasterStyle: Property DesignerMasterStyle does not exist
Class TStyleBook not found

Table 7 - DFM Editor GUI Code View of Delphi FMX

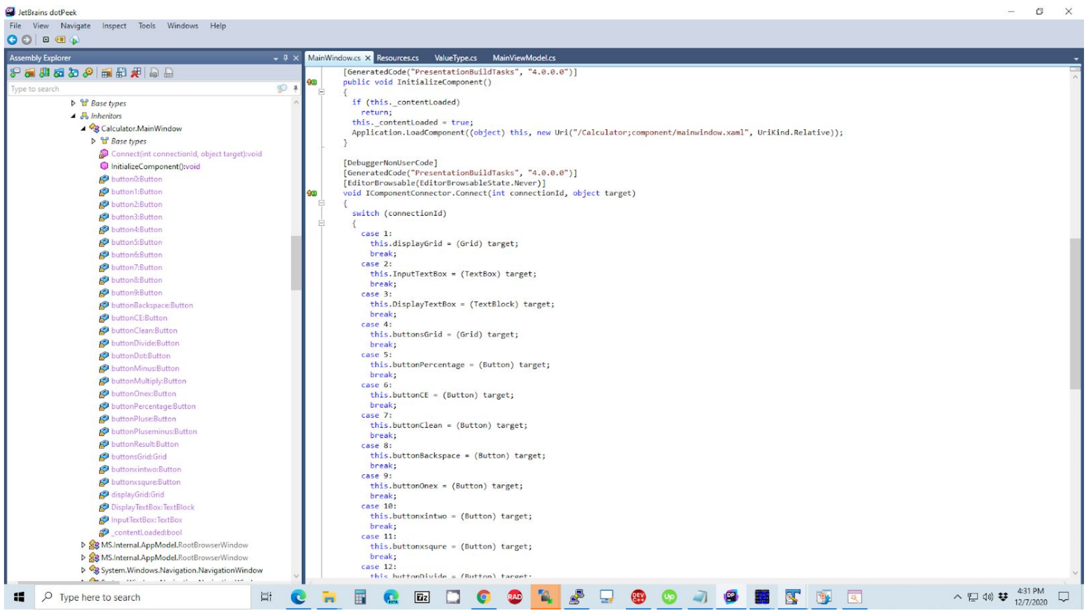
## WPF



```
C:\WPF Compiled Files
  Calculator (1.0.0.0, msil, .Net Framework v4.7.2)
    Metadata
      Headers
      #Strings (267)
      #US (34)
        00000000
        00000001 _CreateDelegate
        00000021 MainWindow.xaml
        00000041 /Calculator;component/mainwindow.xaml
        0000008D Calculator.Properties.Resources
        000000CD IsChanged
        000000E1 0
        000000E5 Display
        000000F5 Expression
        0000010B FirstOperand
        00000125 C
        00000129 Del
        00000131 +/-
        00000139 CE
        0000013F =
        00000143 +
        00000147 -
        0000014B ×
        0000014F ÷
        00000153 sqr
        0000015B √
        0000015F 1/x
        00000167 %
        0000016B Infinity
        0000017D Overflow
        0000018F (
        00000195 )
        0000019B /(
        000001A1 )
        000001A5 Cannnot divide by zero
        000001D3 Error whilst calculating
        00000205
```

```
{
  try
  {
    if (operation != "sqr" && operation != "√" && (operation != "1/x" && operation != "%"))
      this.UsedOperatorInSequence.Add(operation);
    string str1 = this.UsedOperatorInSequence.LastOrDefault<string>((Func<string, bool>) (s => s != operation));
    this.currentOperation = operation;
    if (string.IsNullOrEmpty(this.FirstOperand) || this.LastOperation == "=")
    {
      this.Operation = !(operation == "=") ? (operation == "=" ? this.Operation : operation) : str1;
      this.FirstOperand = this.Display;
      this.LastOperation = operation;
      if (this.Operation == "sqr" || this.Operation == "√" || (this.Operation == "1/x" || this.Operation == "%") || this.LastOperation == "=")
      {
        this.CalculateResult();
        this.Display = this.result;
      }
      this.IsBODMASOperation = true;
    }
    else
    {
      if (!this.IsBODMASOperation || operation == "sqr" || (operation == "√" || operation == "1/x") || operation == "%")
      {
        this.SecondOperand = this.Display;
        this.Operation = operation == "=" ? this.LastOperation : operation;
        this.CalculateResult();
      }
      if (this.Operation != "sqr" && this.Operation != "%" && (this.Operation != "√" && this.Operation != "1/x"))
      {
        this.LastOperation = operation;
        this.FirstOperand = this.result;
        this.IsBODMASOperation = true;
      }
      this.Display = !(this.result == "Infinity") ? this.result : "Overflow";
    }
    string str2 = string.IsNullOrEmpty(this.SecondOperand) ? this.FirstOperand : this.SecondOperand;
    string str3 = operation;
    if (!(str3 == "sqr"))
    {
      if (!(str3 == "√"))
      {
        if (!(str3 == "%"))
        {
          if (str3 == "1/x")
          {
            if (operation == this.LastOperation || this.LastOperation == "sqr" || (this.LastOperation == "√" || this.LastOperation == "1/x") || this.LastOperatio
            {
              this.Expression = string.Empty;
              this.FirstOperand = string.Empty;
              this.SecondOperand = (string) null;
              this.Expression = 1.ToString() + "/(" + str2 + ")";
```

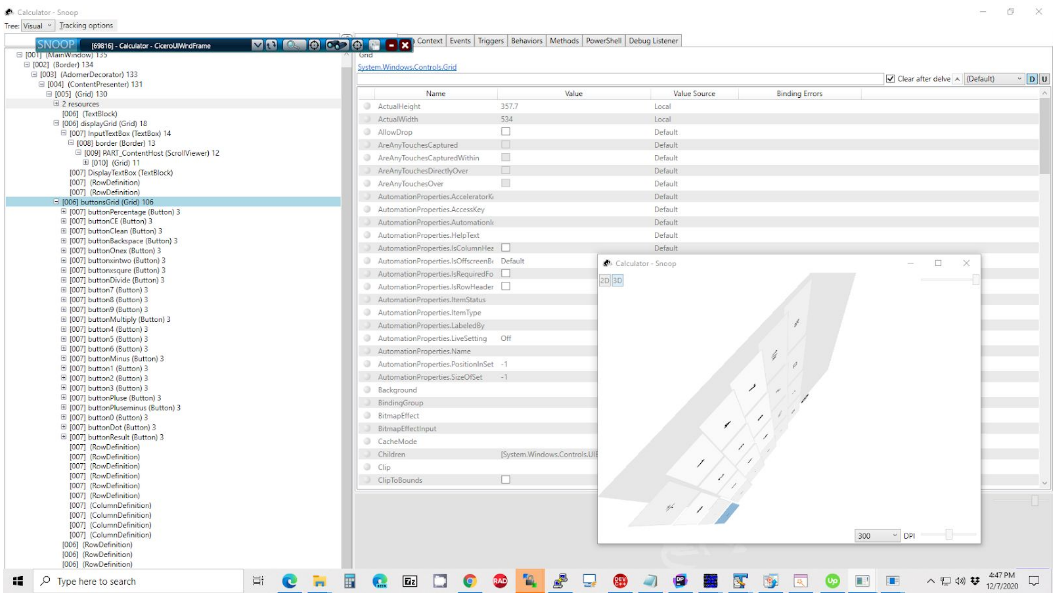
Table 9 - dotPeek Decompilation of WPF Logic



```
Base types
Inheritors
  Calculator.MainWindow
    Base types
    Connect(int connectionId, object target):void
    InitializeComponent():void
    button0:Button
    button1:Button
    button2:Button
    button3:Button
    button4:Button
    button5:Button
    button6:Button
    button7:Button
    button8:Button
    button9:Button
    buttonBackspace:Button
    buttonCE:Button
    buttonClean:Button
    buttonDivide:Button
    buttonDot:Button
    buttonMinus:Button
    buttonMultiply:Button
    buttonOnex:Button
    buttonPercentage:Button
    buttonPluse:Button
    buttonPluseminus:Button
    buttonResult:Button
    buttonsGrid:Grid
    buttonxintwo:Button
    buttonxsqure:Button
    displayGrid:Grid
    DisplayTextBox:TextBlock
    InputTextBox:TextBox
    _contentLoaded:bool
  MS.Internal.AppModel.RootBrowserWindow
  MS.Internal.AppModel.RootBrowserWindow
  System.Windows.Navigation.NavigationWindow
```

```
[GeneratedCode("PresentationBuildTasks", "4.0.0.0")]
public void InitializeComponent()
{
  if (this._contentLoaded)
    return;
  this._contentLoaded = true;
  Application.LoadComponent((object) this, new Uri("/Calculator;component/mainwindow.xaml", UriKind.Relative));
}

[DebuggerNonUserCode]
[GeneratedCode("PresentationBuildTasks", "4.0.0.0")]
[EditorBrowsable(EditorBrowsableState.Never)]
void IComponentConnector.Connect(int connectionId, object target)
{
  switch (connectionId)
  {
    case 1:
      this.displayGrid = (Grid) target;
      break;
    case 2:
      this.InputTextBox = (TextBox) target;
      break;
    case 3:
      this.DisplayTextBox = (TextBlock) target;
      break;
    case 4:
      this.buttonsGrid = (Grid) target;
      break;
    case 5:
      this.buttonPercentage = (Button) target;
      break;
    case 6:
      this.buttonCE = (Button) target;
      break;
    case 7:
      this.buttonClean = (Button) target;
      break;
    case 8:
      this.buttonBackspace = (Button) target;
      break;
    case 9:
      this.buttonOnex = (Button) target;
      break;
    case 10:
      this.buttonxintwo = (Button) target;
      break;
    case 11:
      this.buttonxsqure = (Button) target;
      break;
    case 12:
      this.buttonDivide = (Button) target;
```

Table 10 - dotPeek Decompilation of WPF GUI


Calculator - Snoop

Tree: Visual | Tracking options

SNOOP [69816] - Calculator - CiceroUIWndFrame

```
⊟ [001] (MainWindow) 135
  ⊟ [002] (Border) 134
    ⊟ [003] (AdornerDecorator) 133
      ⊟ [004] (ContentPresenter) 131
        ⊟ [005] (Grid) 130
          ⊞ 2 resources
            [006] (TextBlock)
          ⊟ [006] displayGrid (Grid) 18
            ⊟ [007] InputTextBox (TextBox) 14
              ⊟ [008] border (Border) 13
                ⊟ [009] PART_ContentHost (ScrollViewer) 12
                  ⊞ [010] (Grid) 11
              [007] DisplayTextBox (TextBlock)
              [007] (RowDefinition)
              [007] (RowDefinition)
          ⊟ [006] buttonsGrid (Grid) 106
            ⊞ [007] buttonPercentage (Button) 3
            ⊞ [007] buttonCE (Button) 3
            ⊞ [007] buttonClean (Button) 3
            ⊞ [007] buttonBackspace (Button) 3
            ⊞ [007] buttonOnex (Button) 3
            ⊞ [007] buttonxintwo (Button) 3
            ⊞ [007] buttonxsqure (Button) 3
            ⊞ [007] buttonDivide (Button) 3
            ⊞ [007] button7 (Button) 3
            ⊞ [007] button8 (Button) 3
            ⊞ [007] button9 (Button) 3
            ⊞ [007] buttonMultiply (Button) 3
            ⊞ [007] button4 (Button) 3
            ⊞ [007] button5 (Button) 3
            ⊞ [007] button6 (Button) 3
            ⊞ [007] buttonMinus (Button) 3
            ⊞ [007] button1 (Button) 3
            ⊞ [007] button2 (Button) 3
            ⊞ [007] button3 (Button) 3
            ⊞ [007] buttonPluse (Button) 3
            ⊞ [007] buttonPluseminus (Button) 3
            ⊞ [007] button0 (Button) 3
            ⊞ [007] buttonDot (Button) 3
            ⊞ [007] buttonResult (Button) 3
              [007] (RowDefinition)
              [007] (RowDefinition)
              [007] (RowDefinition)
              [007] (RowDefinition)
              [007] (RowDefinition)
              [007] (RowDefinition)
              [007] (ColumnDefinition)
              [007] (ColumnDefinition)
              [007] (ColumnDefinition)
              [007] (ColumnDefinition)
            [006] (RowDefinition)
            [006] (RowDefinition)
            [006] (RowDefinition)
```

Context | Events | Triggers | Behaviors | Methods | PowerShell | Debug Listener

Grid

System.Windows.Controls.Grid

☑ Clear after delve (Default) D U

| Name | Value | Value Source | Binding Errors |
|---|---|---|---|
| ActualHeight | 357.7 | Local | |
| ActualWidth | 534 | Local | |
| AllowDrop | ☐ | Default | |
| AreAnyTouchesCaptured | ☐ | Default | |
| AreAnyTouchesCapturedWithin | ☐ | Default | |
| AreAnyTouchesDirectlyOver | ☐ | Default | |
| AreAnyTouchesOver | ☐ | Default | |
| AutomationProperties.AcceleratorK | | Default | |
| AutomationProperties.AccessKey | | Default | |
| AutomationProperties.Automationl | | Default | |
| AutomationProperties.HelpText | | Default | |
| AutomationProperties.IsColumnHea | ☐ | Default | |
| AutomationProperties.IsOffscreenB | Default | | |
| AutomationProperties.IsRequiredFo | ☐ | | |
| AutomationProperties.IsRowHeader | ☐ | | |
| AutomationProperties.ItemStatus | | | |
| AutomationProperties.ItemType | | | |
| AutomationProperties.LabeledBy | | | |
| AutomationProperties.LiveSetting | Off | | |
| AutomationProperties.Name | | | |
| AutomationProperties.PositionInSet | -1 | | |
| AutomationProperties.SizeOfSet | -1 | | |
| Background | | | |
| BindingGroup | | | |
| BitmapEffect | | | |
| BitmapEffectInput | | | |
| CacheMode | | | |
| Children | [System.Windows.Controls.UIE | | |
| Clip | | | |
| ClipToBounds | ☐ | | |

Calculator - Snoop

2D 3D

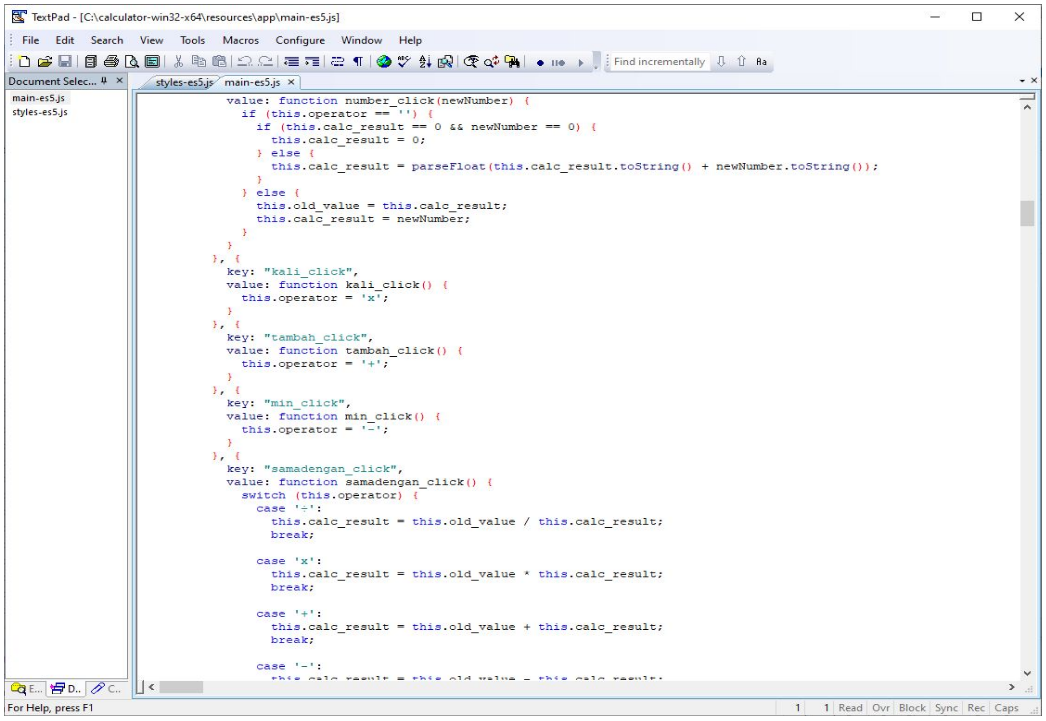
300 DPI

Type here to search

4:47 PM 12/7/2020

Table 11- Snoop WPF Decompilation of WPF GUI

## Electron


TextPad - [C:\calculator-win32-x64\resources\app\main-es5.js]

File Edit Search View Tools Macros Configure Window Help

Find incrementally

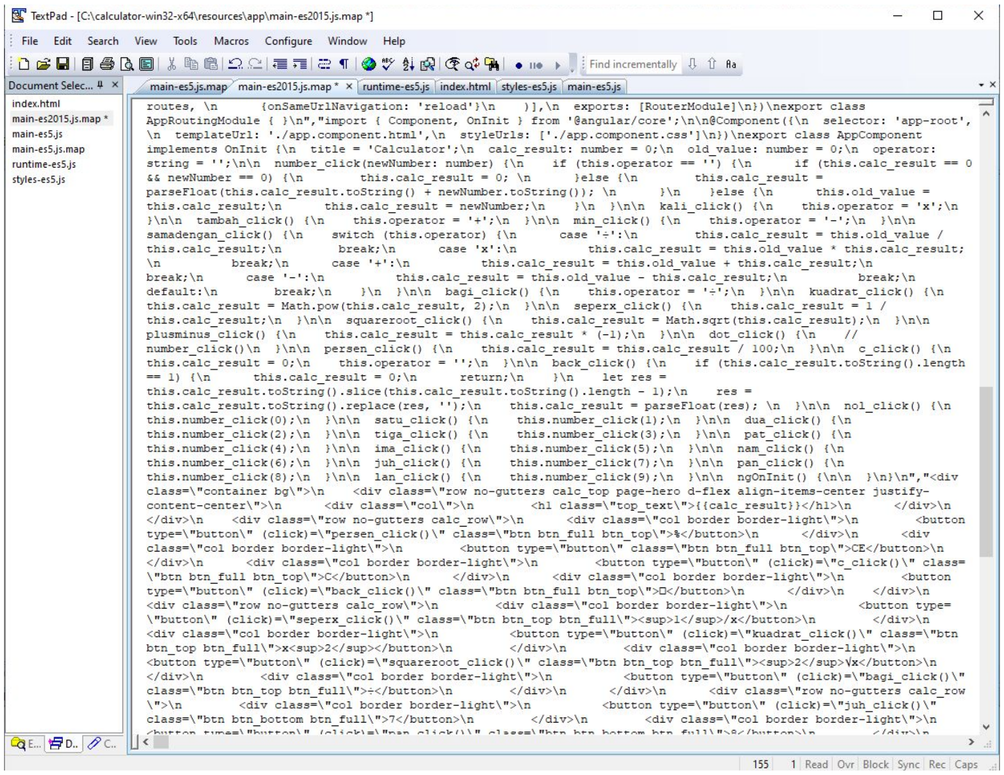
Document Selec... main-es5.js styles-es5.js

styles-es5.js main-es5.js

```
      value: function number_click(newNumber) {
        if (this.operator == '') {
          if (this.calc_result == 0 && newNumber == 0) {
            this.calc_result = 0;
          } else {
            this.calc_result = parseFloat(this.calc_result.toString() + newNumber.toString());
          }
        } else {
          this.old_value = this.calc_result;
          this.calc_result = newNumber;
        }
      }
    }, {
      key: "kali_click",
      value: function kali_click() {
        this.operator = 'x';
      }
    }, {
      key: "tambah_click",
      value: function tambah_click() {
        this.operator = '+';
      }
    }, {
      key: "min_click",
      value: function min_click() {
        this.operator = '-';
      }
    }, {
      key: "samadengan_click",
      value: function samadengan_click() {
        switch (this.operator) {
          case '÷':
            this.calc_result = this.old_value / this.calc_result;
            break;

          case 'x':
            this.calc_result = this.old_value * this.calc_result;
            break;

          case '+':
            this.calc_result = this.old_value + this.calc_result;
            break;

          case '-':
            this.calc_result = this.old_value - this.calc_result;
```

For Help, press F1 | 1 | 1 | Read | Ovr | Block | Sync | Rec | Caps

Table 12- Textpad Displaying Electron Logic Code



```
routes, \n      {onSameUrlNavigation: 'reload'}\n    )],\n  exports: [RouterModule]\n})\nexport class
AppRoutingModule { }\n","import { Component, OnInit } from '@angular/core';\n\n@Component({\n  selector: 'app-root',
\n  templateUrl: './app.component.html',\n  styleUrls: ['./app.component.css']\n})\nexport class AppComponent
implements OnInit {\n  title = 'Calculator';\n  calc_result: number = 0;\n  old_value: number = 0;\n  operator:
string = '';\n\n  number_click(newNumber: number) {\n    if (this.operator == '') {\n      if (this.calc_result == 0
&& newNumber == 0) {\n        this.calc_result = 0; \n      }else {\n        this.calc_result =
parseFloat(this.calc_result.toString() + newNumber.toString()); \n      }\n    }else {\n      this.old_value =
this.calc_result;\n      this.calc_result = newNumber;\n    }\n  }\n\n  kali_click() {\n    this.operator = 'x';\n
}\n\n  tambah_click() {\n    this.operator = '+';\n  }\n\n  min_click() {\n    this.operator = '-';\n  }\n\n
samadengan_click() {\n    switch (this.operator) {\n      case '÷':\n        this.calc_result = this.old_value /
this.calc_result;\n        break;\n      case 'x':\n          this.calc_result = this.old_value * this.calc_result;
\n          break;\n      case '+':\n          this.calc_result = this.old_value + this.calc_result;\n
break;\n      case '-':\n          this.calc_result = this.old_value - this.calc_result;\n          break;\n
default:\n        break;\n    }\n  }\n\n  bagi_click() {\n    this.operator = '÷';\n  }\n\n  kuadrat_click() {\n
this.calc_result = Math.pow(this.calc_result, 2);\n  }\n\n  seperx_click() {\n    this.calc_result = 1 /
this.calc_result;\n  }\n\n  squareroot_click() {\n    this.calc_result = Math.sqrt(this.calc_result);\n  }\n\n
plusminus_click() {\n    this.calc_result = this.calc_result * (-1);\n  }\n\n  dot_click() {\n    //
number_click()\n  }\n\n  persen_click() {\n    this.calc_result = this.calc_result / 100;\n  }\n\n  c_click() {\n
this.calc_result = 0;\n    this.operator = '';\n  }\n\n  back_click() {\n    if (this.calc_result.toString().length
== 1) {\n      this.calc_result = 0;\n      return;\n    }\n    let res =
this.calc_result.toString().slice(this.calc_result.toString().length - 1);\n    res =
this.calc_result.toString().replace(res, '');\n    this.calc_result = parseFloat(res); \n  }\n\n  nol_click() {\n
this.number_click(0);\n  }\n\n  satu_click() {\n    this.number_click(1);\n  }\n\n  dua_click() {\n
this.number_click(2);\n  }\n\n  tiga_click() {\n    this.number_click(3);\n  }\n\n  pat_click() {\n
this.number_click(4);\n  }\n\n  ima_click() {\n    this.number_click(5);\n  }\n\n  nam_click() {\n
this.number_click(6);\n  }\n\n  juh_click() {\n    this.number_click(7);\n  }\n\n  pan_click() {\n
this.number_click(8);\n  }\n\n  lan_click() {\n    this.number_click(9);\n  }\n\n  ngOnInit() {\n\n  }\n}\n","<div
class=\"container bg\">\n    <div class=\"row no-gutters calc_top page-hero d-flex align-items-center justify-
content-center\">\n      <div class=\"col\">\n        <h1 class=\"top_text\">{{calc_result}}</h1>\n      </div>\n
</div>\n    <div class=\"row no-gutters calc_row\">\n      <div class=\"col border border-light\">\n        <button
type=\"button\" (click)=\"persen_click()\" class=\"btn btn_full btn_top\">%</button>\n      </div>\n      <div
class=\"col border border-light\">\n        <button type=\"button\" class=\"btn btn_full btn_top\">CE</button>\n
</div>\n      <div class=\"col border border-light\">\n        <button type=\"button\" (click)=\"c_click()\" class=
\"btn btn_full btn_top\">C</button>\n      </div>\n      <div class=\"col border border-light\">\n        <button
type=\"button\" (click)=\"back_click()\" class=\"btn btn_full btn_top\">□</button>\n      </div>\n    </div>\n
<div class=\"row no-gutters calc_row\">\n      <div class=\"col border border-light\">\n        <button type=
\"button\" (click)=\"seperx_click()\" class=\"btn btn_top btn_full\"><sup>1</sup>/x</button>\n      </div>\n
<div class=\"col border border-light\">\n        <button type=\"button\" (click)=\"kuadrat_click()\" class=\"btn
btn_top btn_full\">x<sup>2</sup></button>\n      </div>\n      <div class=\"col border border-light\">\n
<button type=\"button\" (click)=\"squareroot_click()\" class=\"btn btn_top btn_full\"><sup>2</sup>√x</button>\n
</div>\n      <div class=\"col border border-light\">\n        <button type=\"button\" (click)=\"bagi_click()\"
class=\"btn btn_top btn_full\">÷</button>\n      </div>\n    </div>\n    <div class=\"row no-gutters calc_row
\">\n      <div class=\"col border border-light\">\n        <button type=\"button\" (click)=\"juh_click()\"
class=\"btn btn_bottom btn_full\">7</button>\n      </div>\n      <div class=\"col border border-light\">\n
<button type=\"button\" (click)=\"pan_click()\" class=\"btn btn_bottom btn_full\">8</button>\n      </div>\n
```

Table 13- Textpad Displaying Electron UI Code

embarcadero®

# Discovering the Best Developer Framework Through Benchmarking

Delphi, WPF with .NET Framework, & Electron
On Microsoft Windows 10

16 December 2020, Version 1.0

Authors
Jim McKeeth
Adam Leone
Eli M.