12 Athens - Release 1

Go Up to Main Page

Go Up to What Was New in Past Releases

Updated RAD Studio 12.1 Athens release available (April 04th, 2024).

> ***Note:***
> - *See what is included in version **12 Athens***

RAD Studio 12 Athens - Release 1 (also known as 12.1) is available for installation. RAD 12.1 builds on the feature set of RAD Studio 12, enhancing existing features throughout the product and adding new capabilities.

RAD Studio 12.1 strongly focuses on quality improvements, but it includes some significant enhancements in the C++ toolchain and the IDE. Key features and quality focus areas include:

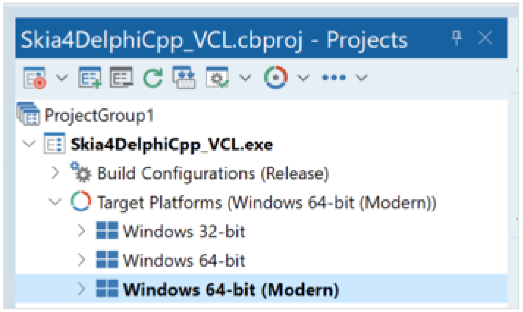
- C++ Clang Toolchain
- Split Editor
- IDE
- Debugger
- Android Target API level 34
- VCL
- VA
- Delphi Compiler
- Other Libraries

Key Enhancements by product area in 12.1

Contents

C++ Clang Toolchain

RAD Studio 12.1 integrates the upgraded Clang toolchain for Win64 into the IDE.


Skia4DelphiCpp_VCL.cbproj - Projects
ProjectGroup1
Skia4DelphiCpp_VCL.exe
Build Configurations (Release)
Target Platforms (Windows 64-bit (Modern))
Windows 32-bit
Windows 64-bit
Windows 64-bit (Modern)

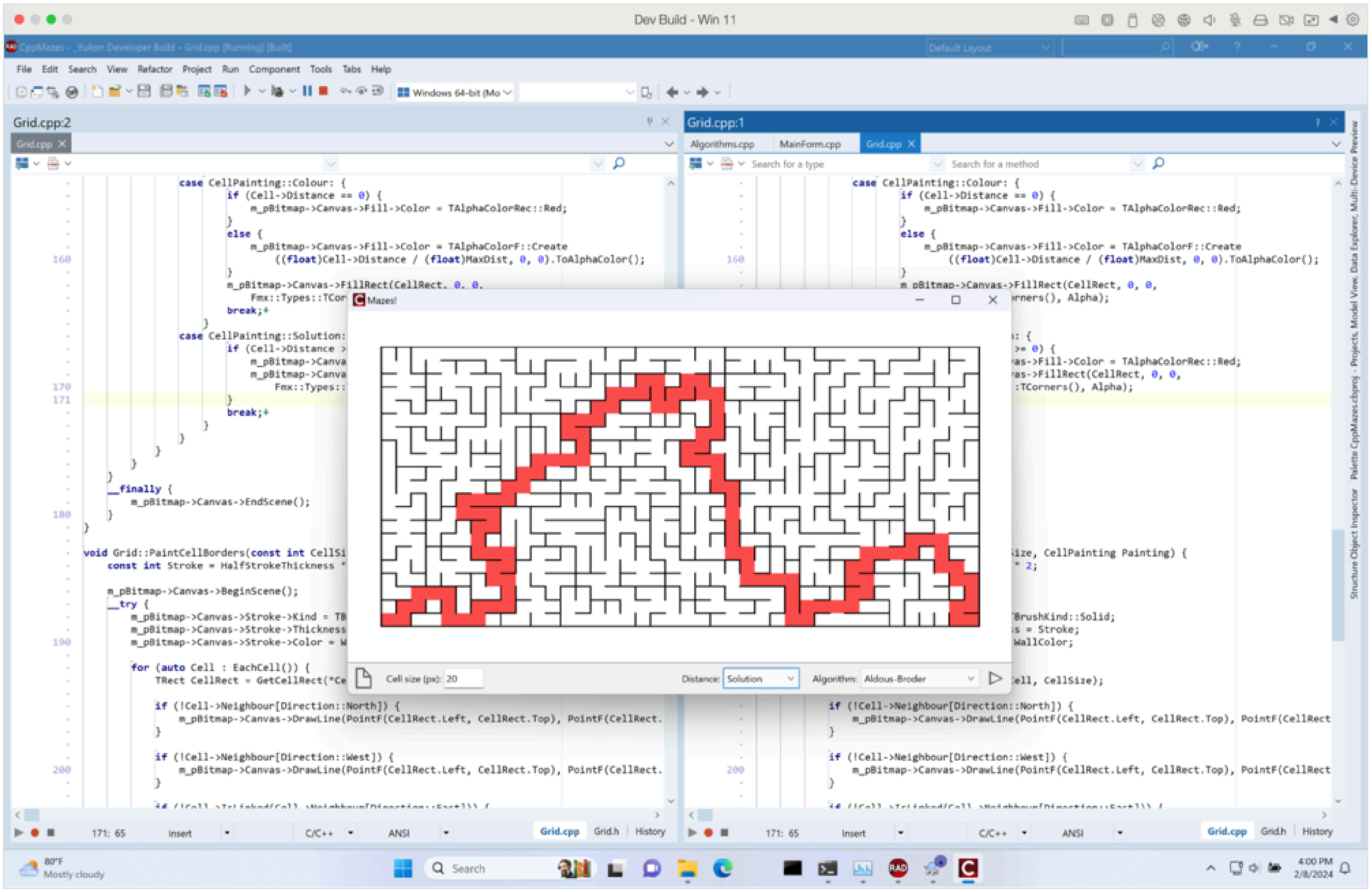
An entire new toolchain with full IDE integration as a normal platform. The toolchain includes the new Clang 15 compiler, LLVM lld linker, libc++ STL, UCRT C runtime, and more.

The toolchain ships parallel to the existing Win64 platform, including VCL and FMX support, DLLs, static libraries, and console applications; right-click the Target Platforms in the IDE and add **Win64 (Modern)**.

What can you do?

The toolchain is ready to test using VCL and FMX with real-world applications. The IDE is fully integrated, sending the right options to the compiler driver and linker. You should not need any special workarounds or compiler command-line flags.

Please note regardless of the dynamic packages setting, your apps will statically link packages.


Dev Build - Win 11
CppMazes - Yukon Developer Build - Grid.cpp [Running] [Built]
File Edit Search View Refactor Project Run Component Tools Tabs Help
Windows 64-bit (Mo
Grid.cpp:2
Grid.cpp:1
Algorithms.cpp MainForm.cpp Grid.cpp
Mazes!
171: 65 Insert C/C++ ANSI
Grid.cpp Grid.h History
80°F Mostly cloudy
Search
4:00 PM 2/8/2024

```
                    case CellPainting::Colour: {
                        if (Cell->Distance == 0) {
                            m_pBitmap->Canvas->Fill->Color = TAlphaColorRec::Red;
                        }
                        else {
                            m_pBitmap->Canvas->Fill->Color = TAlphaColorF::Create
                                ((float)Cell->Distance / (float)MaxDist, 0, 0).ToAlphaColor();
                        }
                        m_pBitmap->Canvas->FillRect(CellRect, 0, 0,
                            Fmx::Types::TCor
                        break;
                    }
                    case CellPainting::Solution:
                        if (Cell->Distance >
                            m_pBitmap->Canva
                            m_pBitmap->Canva
                                Fmx::Types::
                        }
                        break;
                    }
                }
            }
        }
    }
    __finally {
        m_pBitmap->Canvas->EndScene();
    }
}

void Grid::PaintCellBorders(const int CellSi
    const int Stroke = HalfStrokeThickness *

    m_pBitmap->Canvas->BeginScene();
    __try {
        m_pBitmap->Canvas->Stroke->Kind = TB
        m_pBitmap->Canvas->Stroke->Thickness
        m_pBitmap->Canvas->Stroke->Color = W

        for (auto Cell : EachCell()) {
            TRect CellRect = GetCellRect(*Ce

            if (!Cell->Neighbour[Direction::North]) {
                m_pBitmap->Canvas->DrawLine(PointF(CellRect.Left, CellRect.Top), PointF(CellRect.
            }

            if (!Cell->Neighbour[Direction::West]) {
                m_pBitmap->Canvas->DrawLine(PointF(CellRect.Left, CellRect.Top), PointF(CellRect.
            }
```

Cell size (px): 20 Distance: Solution Algorithm: Aldous-Broder

The above image shows an example of the FireMonkey Mazes app (from our Github) using the new Clang toolchain.

Limitations

- Some libraries, like SOAP, are not included in this toolchain. This is to reduce the testing matrix or focus on the most needed libraries for quality. Find a complete list of the available libraries/packages below.
- Packages (like VCL, etc.) are linked into your app statically. RAD Studio 12.1 is not shipping dynamic package consumption, i.e., dynamically linking to BPL files distributed side by side with the EXE in 12.1. The contents of packages are statically linked to the EXE regardless of the 'Runtime packages' setting in the Project Options.
- Creating packages written in C++. RAD Studio 12.1 currently supports packages written in Delphi only.

Special Notes

Virtual File System

The IDE maintains a virtual file system (VFS), which allows you to create a project or file, modify an existing file, and compile and run it without saving it to disk.

In the past, our toolchains all had DLL versions of the compiler and linker to allow this, which hooked into the IDE VFS. In 12.1, the new Clang Upgrade is using the Clang VFS overlay.

Temporary VFS files are removed by default after compiling. Turn this off in Project Options > C++ Compiler > Advanced and uncheck the **Remove temporary VFS files after compilation** option.

POSIX and other RTL methods

RAD Studio never fully complied with the standard Windows set of RTL methods and provided many POSIX versions and custom RTL functions. Some of those may now be missing.

C++ Standards Compability

RAD Studio uses libc++ version 15.0. Check the language standards compliance on the following pages:

- C++14 (https://libcxx.llvm.org/Status/Cxx14.html)
- C++17 (https://libcxx.llvm.org/Status/Cxx17.html)
- C++20 (https://libcxx.llvm.org/Status/Cxx20.html) (not supported by RAD)
- C++23 (https://libcxx.llvm.org/Status/Cxx23.html) (not supported by RAD)

> *Note:* *The Format library has support introduced in libc++ 16 (https://libcxx.llvm.org/Status/Format.html). Therefore, RAD Studio 12.1 does not yet support it. (The {fmt} library is untested but should work well.) Other items like Parallelism TS, the spaceship operator, etc, are in libc++, but the version is not yet documented*

.

Packages and Libraries

The new lld linker is considerably faster (about 4x faster) than ilink. It handles larger file sizes easily, so dynamic linking is not a required workaround for either link time or linker memory.

Dynamic package consumption is important for architectural reasons, and RAD Studio plans to upgrade the dynamically linking Delphi runtime packages in a future release.

Find the list of included and not included libraries on our packages documentation page.

Split Editor

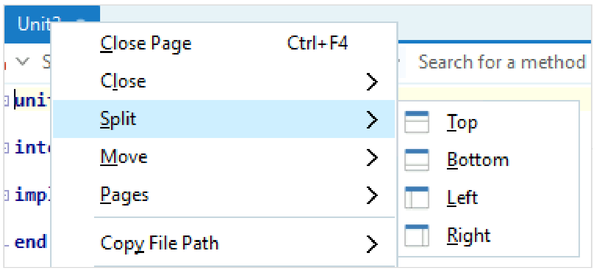
RAD Studio 12.1 adds support for split editor views in the IDE. Use multiple editors side by side, above and below each other, or a mix, all in the same window. This includes dragging entire sets of tabs out to floating windows and back into the IDE.

RAD Studio implements split editor views in an advanced way, allowing complex custom layouts and manipulation. This means you can have a simple two-column layout, multiple-row layout, or something customized exactly for your workflow.

You can have multiple tabs open, all editing the same unit, and have multiple tabs open with a form designer in one tab plus one or more code tabs for the same unit. This includes moving which tab displays the designer. There is **no limit** to the number of split editors, horizontally or vertically.

Using Split Editor

You can access the Split Editor view by right-clicking a tab and choosing **Split >** or **Move >**. Split will create another version of the selected tab, while move will close the tab and open it in the selected place in your IDE.



Alternatively, you can use Split Editor by dragging a file from the Projects view into any open editor to open that file as a tab in that tab group.

Once two tabs are in a window, both sets have title bars. You can:

- Drag an editor tab, or set of tabs, by dragging the title bar and dock it to any side of the IDE window.
- Drag and dock it to any other existing editor's side (left, right, top, bottom).
- Merge it with an existing set of tabs by dragging it over the center of an existing editor, including dragging a floating window of tabs to merge.
- Drag it out to become its own window (either one tab or a set of tabs).

Most operations can be done on individual tabs, such as moving a tab between docked sets of tabs by dragging the individual tab from one location to another.

Each set of tabs can be dragged out to become its own window. Drag the title bar to move it. (Hold Ctrl while dragging to ensure it does not try to dock, to make it float.)

Similarly, each window can be docked next to another set of tabs, to do this drag the title bar.

> *Note:* *Most editors/groups, when closed, will simply vanish. But the first editor, the one that used to be the only editor for the main window, when closed with the 'X' button, will close all the tabs it holds. Still, the space for it will remain open. The original*

> *editor can't be closed and always has space on the screen. This won't affect anything until you close tab groups. There is no visible impact when you create, move, or dock them in everyday work.*

Designer and Code

Split editors allow the same file to be edited next to itself. Both editors have views over the same file contents. As you type in one, if you scroll to the same location, you will see the text appear in the other.

But both views can be scrolled independently, allowing you to edit the same file in multiple locations. You can have any number of tabs editing the same file.

> ***Note:*** *Remember, there is only one file (module or file buffer in internal terms), but multiple editor views over that file, so a change you make in one editor is always reflected in every other editor for the same file. A change made in one editor is not 'copied' or 'duplicated' or anything else: there is only one file and one content for that file, even if you have multiple editors viewing/editing that file.*
> *Another way of saying this is that when you edit a file in one editor, that change is in the file itself, not the editor. By definition, that means that change is always present in the other editors also showing the same file.*

Focus Tracking

A key tenet of RAD Studio's UI design is to make focus clear. In versions before RAD Studio 12.1, the currently focused window was a strong blue color, whether a docked window on one side or the (then solo) editor tab.

The same technique applies now that we have multiple editors, but the tab group's title bar and the specific tab are a solid blue. This is to make it really, really clear.

Editor groups that do not hold the focused editor have a pale 'background' title bar, and even their foremost tabs (the ones that would be focused if that group were focused) are gray, indicating a lack of focus.

Docking and Layouts

Starting with 12.1, RAD Studio allows you to move around and dock panes such as Object Inspector anywhere you want. In practice, this means there can be little distinction between dockable windows (like the Object Inspector, Structure view, and so forth) and dockable editors/editor groups, allowing you to create powerful layouts tailored to your needs.

A desktop layout is the set of open editors (and editor groups, their location, where they're docked, etc.) along with the old-style dockable panes and can include multiple windows, each with multiple sets of tabs.

The configuration of editors needs to be saved as your desktop layout; otherwise, it is easily lost.

In addition, the configuration will change when the desktop layout changes, such as when the IDE starts debugging and switches to the Debug Layout. When the layout changes, any tab group (which can be an individual tab, too, so long as it is separately docked) without a place in the layout being switched to will be undocked to a floating window.

IDE Improvements

DelphiLSP Code completion

RAD Studio 12.0 added the Code completion feature, making code completion automatically pop up as you type. This pre-DelphiLSP feature provides a list of helpful suggestions rather than appearing only on dot or only when manually invoked.

However, appearing automatic as you typed often made it hard to type specific things, such as new variable names, because (as previously unknown identifiers) they would not be suggested. Code completion accepted the top item when you pressed `Space` , which overrode your writing.

RAD Studio version 12.1 improved the functionality so when the auto-invoke function is `On` , and code completion appears automatically, only pressing `Tab` and `Enter` accepts the currently selected completion. Other keys, such as `Space` , open brace, dot, etc., do not work. This means you must choose to accept a completion when code completion is auto-invoked; you won't accidentally get something completed.

When code completion is invoked normally, those keys accept what is suggested, as in 11.3 and earlier. This means that behavior is unchanged for all previously normal code completions.

The auto-invoke feature is `Off` by default, turn it on and learn more about the new completion keys tab on the Code Insight documentation.

Debugger

In version 12.1, all debuggers other than Delphi Win32, Delphi Win64, and C++ Win32 use LLDB 15. This is the same debugger used by the Clang Upgrade.

Previous versions used LLDB 12 for these platforms and have been upgraded.

Android Target API level 34

RAD Studio version 12.1 is updating the Android API level to 34. This change required changes to the Android SDK tooling, some additional Android platform tools, and an update of the Java runtime. Here are more details:

- Version 12.1 now installs among platform features and recommends Eclipse Temurin JDK OpenJDK 17 (Hotspot) JVM.
- Version 12.1 updated the Command-line Tools Android SDK components from our default Android SDK installation.
- The default `targetSdkVersion` manifest attribute is now 34.

RAD Studio 12.1 provides a noticeable performance boost in the `.apk` file creation because version 12.1 takes advantage of Google's **zipflinger** and **signflinger** libraries in the Packager command-line tool. The following are the improvements in the Android packaging:

- The Android DEX operations (DEX compilation and DEX merging) are now incremental (they are only triggered when the input files are considered out-of-date in relation to the output files).
- Version 12.1 migrated the build system from the old and deprecated Google's AAPT command-line tool to the new Google's **AAPT2** tool.
- Google's AAPT2 command-line is now used to compile each Android resource file and package them together in a `.zip` file containing the manifest file and the Android resource table.
- The new Packager command-line tool puts the packaged resource files, native libraries, asset files, and `.dex` files into the final `.apk` or base module file.

Quality Improvements

VCL

- Enabled VCL TEdit to work correctly with the new Ease of Access Text Cursor in recent versions of Windows. This way, we bring back accessibility to the IDE, which was disabled by default due to that issue. Using JAWS with the IDE should now work (again) by default.
- NumberBox improvements when working with negative values.
- ProgressBar State is now supported under VCL Styles.

Visual Assist for C++

- Solved an issue where VA was left with a cache if a file was closed unsaved.
- Renaming now also renames event handlers.
- Renaming a component will include renaming it in the designer, a missing feature in 12.0.
- Code completion on object properties is now working correctly.
- Code completion results no longer show unexpected values.
- Solved an issue where code completion was not triggering.
- Solved an issue where VA was not working on 3rd party libraries.
- `#include` preprocessor directive autocomplete with `<` and `“”` should now work correctly.
- Minor UI improvements on menus.

Other Libraries

RAD Studio 12.1 introduces the following improvements in the database space:

- Improved the TIBDataSet editor (IBX).
- Added FireDAC support for Firebird version 5. This includes support for parallel operation in TFDFBBackup for Firebird 5.
- Added FireDAC support for PostgreSQL version 16.
- The latest versions of IBToGo and IBLite (aligned with InterBase 2020 Update 5) should now work on Android.

In FireMonkey, 12.1 includes the following:

- On macOS, transparent bitmaps saved to non-transparent formats behave like on Windows (i.e., the transparent color will turn black).

In the Internet space, 12.1 includes the following:

- TRestClient supports using certificates from the local machine list and not only from the current user list.
- Support for web hex color syntax in StringToColor function.
- The ability for the REST Debugger to copy values from table cells.
- Ability to change the username of RAD Server users.

See Also

- Installation Notes
- Release Notes

- New features and fixed issues
- What Was New in Past Releases

Retrieved from "https://docwiki.embarcadero.com/RADStudio/Athens/e/index.php?title=12_Athens_-_Release_1&oldid=276635"