Four Years of Cross-platform Improvements

Nirbheek Chauhan

(*nirbheek* on GitLab / Matrix / Twitter / Hachyderm.io)

https://nirbheek.in

gstreamer

Centricular

Windows, macOS, iOS, Android

Build system updates

Platform-specific elements

Amy Spark, Jan Schmidt, Matthew Waters, Nirbheek Chauhan, Piotr Brzeziński,
Sebastian Dröge, Seungha Yang, Stéphane Cerveau, Tim Müller, Xavier Claessens

gstreamer

Centricular

Cerbero: Windows Builds

- Run `cerbero-uninstalled` from any terminal now
- MSYS2 move is complete, stop using MSYS
  - Much faster and more reliable builds
- Visual Studio is the default compiler
- Moving recipes away from Autotools
  - 22 recipes ported to Meson, 4 to CMake, 19 left
- Universal Windows Platform
  - Began in 1.18, demoted in 1.22 when Mozilla dropped HoloLens
  - ARM64 builds are still tested in CI
- MinGW-GCC toolchain was updated to 8.2
  - Also became outdated in the last 4 years 😅
  - Medium-term plan is to use third-party MinGW toolchain
- Windows 11 SDK is required (VS 2019 or 2022)
- Rust plugins shipped out of the box
- WiX installers when cross-compiling

Monorepo: Windows Builds

- Functional gstreamer-full support
  - Better static compilation support
- CI tests MSYS2, VS x86, VS x86_64, VS arm64
- More unit tests pass now
- `gst-plugin-scanner` has a win32 loader implementation
  - Plugin scanning is done out of process now
- More meson wraps were added
  - pango, cairo, opus, libjpeg-turbo, fdkaac, libvorbis, libogg, lame, libsrtp2, gtk4, webrtc-audio-processing
- gst-plugins-rs meson subproject
- Development environment works most of the time
  - `PATH` length limitation is the main issue
  - Workaround is to install to a prefix
  - Can use it from powershell too, now

Cerbero: macOS and iOS Builds

- Minimum iOS version is 11.0
- Minimum macOS version went to 10.10... and then 10.13 in 1.24
  - Qt5 supports 10.13
- Apple bitcode support was added... and then removed
- Framework improvements
  - RPATH set correctly, fully relocatable
  - pkg-config shipped with the framework
  - Plugin pkgconfig files are correctly relocatable
- Apple Silicon support
  - Cross compilation was easy, same as iOS
  - Native support also works now, took some work
  - Universal binaries
  - CI builds on Apple Silicon, much faster
- Plans to move to xcframework for better ARM64 simulator support

Monorepo: macOS Builds

- Lots of fixes for the monorepo development environment
  - Especially static linking
  - Need to use `eval $(./gst-env.py --only-environment)` in some cases
- Builds are tested on the CI, should break less often
- gstreamer-full should work
- gst-plugins-rs meson subproject
- Homebrew builds use the monorepo now
  - Small issue with the libnice plugin
- Machine file to use with Homebrew:
  - `data/machine-files/macos-native-file.ini`

Windows Features: General

- High resolution timers (sleep, events, etc)
  - Default timers are about 15ms
  - Enabling high precision timers uses more power
  - Application-specific, only enabled by default for gst-play, gst-launch and in d3d11/amf elements
  - See: `subprojects/gstreamer/tools/gst-launch.c`
- Use native atomic operations in more places
  - Pending GLib changes to use C11 atomics
  - Visual Studio supports C11, C17, C23
  - We will likely require C11 soon
- `gst-ptp-helper` ported to Rust
  - Old C helper didn't support Windows
  - Used to sync with a PTP clock
  - First code that uses Rust in gstreamer core

Windows Features: D3D11 DXVA2

- Lots of video formats, HDR support (10-bit 12-bit 16-bit), etc
- VP8, VP9, AV1, MPEG-2, H264, H265 decoders
  - d3d11testsrc
  - d3d11convert (convert, scale, cropping, flip/rotate)
  - d3d11deinterlace
  - d3d11compositor
  - d3d11overlay
  - d3d11screencapturesrc
  - d3d11videosink (deprecates d3dvideosink)
- Zerocopy IPC elements
- Qt6 video sink
- WinRT / Universal Windows Platform

Windows Features: Media Foundation

- Hardware-agnostic
  - AMD, Intel, Nvidia, Qualcomm
- Camera capture
  - Windows Kernel Streaming is deprecated
- Video encoders: H264 H265 VP9
  - Software fallbacks
  - HDR support
- Audio software codecs: AAC MP3
- WinRT / Universal Windows Platform

Windows Features: Nvidia

- Encoder rewritten, reorganized, optimized repeatedly based on real-world usage
  - HDR YUV and RGB 10-bit 16-bit
  - Supported inputs: CUDA, D3D11, OpenGL
  - Automatic GPU selection mode ((CUDA vs D3D11) + GPU)
- Decoder rewritten to be stateless, optimized based on real-world usage
  - AV1 VP8 VP9 H264 H265 MPEG2
  - HDR YUV and RGB 10-bit 12-bit 16-bit
- New gstcuda library
- cudaconvertscale
- cudaupload / cudadownload
  - D3D11 interop
  - GL interop
- Zerocopy IPC elements

Windows Features: AMD and Intel

- Intel MSDK support
  - AV1 H264 H265 JPEG VP9 encoder + decoder
  - Only system memory is supported
- Intel QSV support (recommended)
  - H264 H265 JPEG VP9 encoder + decoder
  - AV1 only encoder right now
  - D3D11 zerocopy and system memory
- AMD AMF video codec support
  - H264 H265 encoder + decoder
  - D3D11 zerocopy and system memory

Windows Features: Even More

- WASAPI2 audio source/sink elements
  - New plugin, replacement for WASAPI elements
  - Written in C++, uses WinRT APIs, requires Windows 10
  - Automatic stream routing
  - Rewritten! Now has a higher rank than WASAPI
- Directwrite elements, HW-accelerated
  - New plugin, replacement for pango elements
  - clockoverlay
  - subtitlemux
  - subtitleoverlay
  - textoverlay
  - timeoverlay

Windows Features: Still More

- Windows Image Component plugin
  - jpeg decoders
  - png decoders
  - Faster in some case
  - Leaner app builds
- DirectShow elements (deprecated)
  - Audio capture from directshow graph
  - Video capture from directshow graph
  - Ancient audio / video decoders
  - wma, wmv, divx, cinepak, mpeg 1, 2, 3, 4
  - Video render to directshow graph
  - Purely for legacy use

Windows Summary: Capture & IPC

- Screen capture
  - dshowvideosrc (DirectShow, deprecated)
  - gdiscreencapsrc (GDI, deprecated)
  - dx9screencapsrc (Direct3D 9, deprecated)
  - dxgiscreencapsrc (**REMOVED**)
  - d3d11screencapturesrc (Direct3D 11)
  - d3d11screencapturesrc (WinRT Windows Graphics Capture)
- Camera capture
  - ksvideosrc (Windows Kernel Streaming, deprecated)
  - mfvideosrc (Media Foundation)
- Video IPC
  - cudaipc
  - d3d11ipc
  - win32ipc (software memory)

Windows Summary: Render

- Software video sinks
  - dshowvideosink (deprecated)
  - gtksink
- Direct3D video sinks
  - d3dvideosink (dx9, deprecated)
  - d3d11videosink
  - d3d12videosink
  - qml6d3d11sink
- OpenGL, Vulkan video render
  - glimagesink
  - qmlglsink
  - qml6glsink
  - gtk4paintablesink
  - gtkglsink
  - vulkansink

macOS Features: General

- Cocoa event loop changes
  - `gst_macos_main()` runs `NSRunLoop` in the main thread
  - Custom GLib patches that did that automatically are now dropped
  - No longer need a GLib main loop now
  - Tutorials and examples have been updated
  - See: `subprojects/gst-plugins-base/tools/gst-play.c`
- Documentation is getting some love again
- Better developer experience overall
- Development is speeding up, lots of fixes are happening
- Number of GStreamer developers using it natively have doubled
  - Or something like that 😉
- Expect native macOS support to start matching Windows soon

macOS/iOS Features: VT & AVF

- VideoToolBox encoder/decoder elements improvements
  - Apple ProRes, H265
  - Encoder output is pushed via a separate task
  - Zerocopy decoder support already exists
  - Zerocopy encoder support is WIP
  - YUV support in GL sinks
- AVFoundation element improvements
  - Screen capture fix default framerate when unspecified
  - Screen capture add cropping properties
  - Screen capture report latency correctly
  - Camera capture fix framerate detection

macOS/iOS Features: CoreAudio

- CoreAudio elements are much improved
  - Extreme low latency achievable
- Device provider can uniquely identify devices across reboots
- Source provides a better clock now
- Source accurately configures the segment sizes now
- Source sets sample timestamps based on CoreAudio reporting
  - Some bugs need fixing before 1.24 is released
- Hidden device support, virtual devices
- Support devices IDs that are both input and output
  - Previously would only be a sink

Android Improvements

- Cerbero updated at first to NDK r21 and now r25
- Can cross-compile to Android using the monorepo
  - `data/machine-files/android_arm64-cross-file.ini`
- OpenSLES audio source/sink improvements
- Small bugfixes, no big changes

Thanks!