Flutter Deep Dive

Toyota Connected North America - UI/UX Team

Joel Winarske

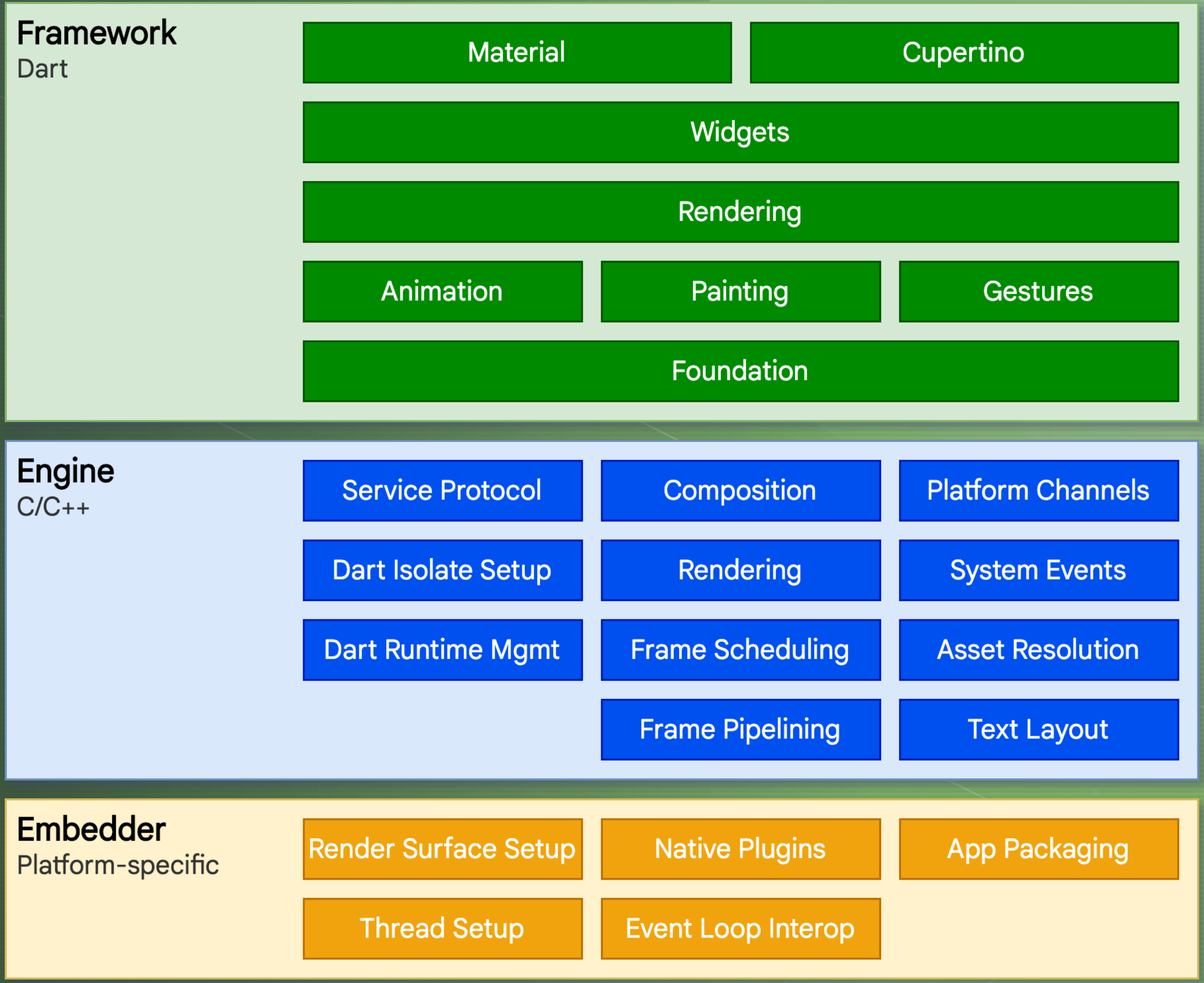
Flutter

- Flutter is Google's UI toolkit for building beautiful, natively compiled applications for mobile, web, desktop, and embedded devices from a single codebase.
- Dart is a programming language designed for client development, such as for the web and mobile apps.
- Skia is an open source 2D graphics library which provides common APIs that work across a variety of hardware and software platforms. Skia is used in the Chrome browser.
- Flutter is the combination of Dart + Skia.

Flutter Advantages

- Reduced Code Development Time
- Increased Time-to-Market Speed
- Similar to Native App Performance
- Same UI and Business Logic across All Platforms
- Custom, Animated UI of Any Complexity Available
- Own Rendering Engine
- BSD 3-Clause "New" or "Revised" *License*
- Non-Proprietary / Open Source
- Zero Licensing Fees
- Smaller RAM / ROM footprint than Browser based UI
- Hot Reload

Flutter Architecture



Flutter Engine

- The core of Flutter, which is mostly written in C++ and supports the primitives necessary to support all Flutter applications. The engine is responsible for rasterizing composited scenes whenever a new frame needs to be painted. It provides the low-level implementation of Flutter's core API, including graphics (through Skia), text layout, file and network I/O, accessibility support, plugin architecture, and a Dart runtime and compile toolchain.
- The engine is exposed to the Flutter framework through dart:ui, which wraps the underlying C++ code in Dart classes. This library exposes the lowest-level primitives, such as classes for driving input, graphics, and text rendering subsystems.
- https://github.com/flutter/engine

Dart Framework

- Developers interact with Flutter through the Flutter Framework.
- A modern reactive framework written in the Dart language.
- The Flutter framework is relatively small; many higher-level features that developers might use are implemented as packages, including platform plugins like camera and webview, as well as platform-agnostic features like characters, http, and animations that build upon the core Dart and Flutter libraries. Some of these packages come from the broader ecosystem, covering services like in-app payments, Apple authentication, and animations.

Dart Framework - continued

- It includes a rich set of platform, layout, and foundational libraries, composed of a series of layers.
- Basic **foundational** classes, and building block services such as **animation**, **painting**, **and** **gestures** that offer commonly used abstractions over the underlying foundation.
- The **rendering layer** provides an abstraction for dealing with layout. With this layer, you can build a tree of renderable objects. You can manipulate these objects dynamically, with the tree automatically updating the layout to reflect your changes.
- The **widgets layer** is a composition abstraction. Each render object in the rendering layer has a corresponding class in the widgets layer. In addition, the widgets layer allows you to define combinations of classes that you can reuse. This is the layer at which the reactive programming model is introduced.
- The **Material** and **Cupertino** libraries offer comprehensive sets of controls that use the widget layer's composition primitives to implement the Material or iOS design languages.
- https://flutter.dev

Embedder

- The embedder is written in a language that is appropriate for the platform.
- Currently Java and C++ for Android
- Objective-C/Objective-C++ for iOS and macOS
- C++ for Windows and Linux
- Flutter includes a number of embedders for common target platforms
- Generic Interface for communicating between Dart and Native code – Platform Channels
- Toyota Connected has a custom Embedder specific to Embedded Linux (Wayland) and Automotive use cases

Flutter Versioning Semantics

- Flutter SDK: https://github.com/flutter/flutter
- Flutter engine: https://github.com/flutter/engine
- The SDK compiles Dart apps (using artifacts produced by engine build output)
- Both SDK & Engine have distinct versions
- For stability purposes, it's best to assume that apps compiled with a given SDK version MUST be run by the engine version locked to the same SDK commit.
- **This introduces a requirement that either:**
  - **All Flutter apps running on a system must be compiled against the same SDK version, or**
  - **The system must support packaging an app in a way that is strongly coupled to its engine, and shipping an engine-per-app**

Flutter Channels & Stability

- https://github.com/flutter/flutter/wiki/Flutter-build-release-channels
- stable = quarterly releases
- beta* = monthly releases
- dev = as often as the full test suite passes
- master = tip of tree

** beta is currently our "default" channel for development purposes, with a balance of latest updates with good stability, but this can be freely modified.*

ivi-homescreen

- Flutter embedder specifically for Wayland
- Strongly Typed (C++)
- Lightweight
- Clang 11 Release Stripped = 151k
  - GCC 9.3 Release Stripped = 168k
- Runs on Linux Desktop and Yocto Linux
  - Ubuntu 18+
  - Fedora 33+
  - Yocto Dunfell+
- Platform Channels enabled/disabled via CMake
- OpenGL Texture Framework
- Hot Reload is functional

Developer Workflows

- Execute/Debug Application on Linux Desktop
  - Developers are encouraged to use Linux Desktop for Application Development
  - CI validation – headless in cloud
  - Establish perf and regression baselines using profile runs
- Execute/Debug Flutter Application on Target
  - CI validation - board farm
  - Establish perf and regression baselines using profile runs

Execute Application on Desktop

- Install ivi-homescreen and flutter-engine packages
- Build your flutter application using flutter build bundle
  - cd ~/development/my_flutter_app
  - flutter channel beta
  - flutter upgrade
  - flutter config --enable-linux-desktop
  - flutter create .
  - flutter build bundle
- Create application symlink in /usr/local/homescreen
  - cd /usr/local/share/homescreen && sudo rm -rf bundle/
  - sudo ln -sf ~/development/my_flutter_app/build/ bundle
- Run Flutter Application
  - /usr/local/bin/homescreen

Debug Application on Desktop

- Follow steps under Execute Application on Desktop with following exception
  - Install flutter-engine "debug" package
- After Flutter embedder is running attach to process
  - flutter attach --debug-port 41795 --host-vmservice-port 41795
- Hot Reload, Restart, etc. is available via console only

Create Ahead Of Time (AOT) image

- flutter build bundle
- dart ${ENGINE_SDK}/frontend_server.dart.snapshot --aot --tfa --target=flutter --sdk-root ${ENGINE_SDK} --output-dill app.dill lib/main.dart
- ${ENGINE_SDK}/clang_x64/gen_snapshot --deterministic --snapshot_kind=app-aot-elf --elf=libapp.so --strip app.dill
- Notes:
  - Requires flutter-engine “Release” or "Profile" variant to run
  - Desktop ENGINE_SDK=/usr/local/share/flutter/sdk
  - Target ENGINE_SDK=/usr/share/flutter/sdk
- See integration example here: https://github.com/meta-flutter/meta-flutter/blob/main/recipes-graphics/flutter-apps/flutter-gallery_git.bb

Building with Yocto

- https://github.com/meta-flutter/meta-flutter
- Add meta-flutter and meta-clang layers
- At minimum append ivi-homescreen and flutter-engine to image.
- Common variables overridden in "local.conf"
  - FLUTTER_CHANNEL. Generally good idea to set in local.conf.
  - SRCREV_flutter-engine. Defaults to FLUTTER_CHANNEL value.
  - TARGET_GCC_VERSION. Defaults to 9.3. If GCC version != 9.3 override with value.
- For application use cases see meta-flutter CI validation jobs
  - https://github.com/meta-flutter/meta-flutter/blob/main/.github/workflows/ci.yml

AGL Build Example

- meta-flutter validation CI include AGL Lamprey build example
- https://github.com/meta-flutter/meta-flutter/blob/main/.github/workflows/ci.yml#L222

Long Term Support

- Flutter SDK
  - Flutter storage buckets for milestone releases
    - https://storage.googleapis.com/flutter_infra/releases/releases_windows.json
    - https://storage.googleapis.com/flutter_infra/releases/releases_macos.json
    - https://storage.googleapis.com/flutter_infra/releases/releases_linux.json
- Flutter Engine
  - What we need is a tar.xc of Engine source posted to match SDK milestone releases. This request has been made to the Flutter PM. This would follow the pattern of how Chromium archives source.
  - This would enable a dedicated Yocto recipe per LTS release, eliminate use of gclient, and cache everything in DL folder.

Flutter Gallery App on AGL Emulator Image

- Ubuntu 20.04.3 LTS
- Download and extract "agl-image-weston-flutter-qemux86-64-linux" package from: https://github.com/meta-flutter/meta-flutter/actions/runs/1361062278
- Login to console
- # vi /etc/xdg/weston/weston.ini. Change transform to 0, and mode to 1920x720. "wq" to write and quit.
- # systemctl restart weston@root
- # homescreen
- https://docs.automotivelinux.org/en/master/#0_Getting_Started/1_Quickstart/Using_Ready_Made_Images/#1-qemu-emulation

Important Notes

- In order to run a Flutter application, the Flutter SDK Dart VM version must match the Flutter Engine Dart VM version.
  - You can check the Flutter Engine Dart VM version via:
    - Desktop
      - cat /usr/local/share/flutter/sdk/dart.version
    - Target
      - cat /usr/share/flutter/sdk/dart.version
- A flutter channel will roll or change commit hash periodically. The channels are master, dev, beta, stable. The rate of change is greatest with the master channel, where the beta channel rolls generally once a week.
- Flutter applications built by default are machine agnostic and run as interpreted code.
- To achieve native run time performance, build your application for Ahead Of Time (AOT). This compiles project into native machine code.

x86_64 binary packages

- ivi-homescreen
  - https://github.com/toyota-connected/ivi-homescreen/actions/runs/1251358078
- flutter-engine
  - https://github.com/jwinarske/flutter_embedded/actions/runs/1259970904
- Deployment to JArtifactory based Debian/RPM/Opkg repository, including versioning and dependent packages planned. Possibly an alternative that is hosted on AGL site?

Repositories

- https://github.com/toyota-connected/ivi-homescreen
  - Linux Flutter Embedder
- https://github.com/meta-flutter/meta-flutter
  - Yocto layer for embedded Flutter
  - includes recipes for flutter-engine and ivi-homescreen
- https://github.com/jwinarske/flutter_embedded
  - X86_64 Debian, RPM, tar.gz package of libflutter_engine.so
  - Runtime – debug, profile, release
- https://github.com/jwinarske/flutter-channel-watch
  - Checks flutter channel every 15 minutes for a change in commit
  - Used to trigger flutter-engine builds

Planned/Active Work

- Open Source AGL integration
  - Full featured homescreen
- v2 ivi-homescreen
  - Vulkan Based
  - 3D Rendering engine integration
  - Flutter Engine renders to RenderTarget of type vkImage

Why Vulkan in v2?

- https://flutter.dev/go/embedder-vulkan-support
- Mixed compatibility with EGL
- Higher performance (better advantage of parallelism)
- Better inter-operability with 3d scenes
- In the future we plan to integrate https://github.com/google/filament as a 3D plugin tool - which benefits from Vulkan