Flutter and AGL
Deep Dive

AGL Workshop
Joel Winarske
Toyota Connected North America
Introduction

• Qualcomm Windows BSP organization (~8 years)
  • 1st FTE in Redmond, WA
  • MSM7200 to Snapdragon 855
  • Window Phone, Windows on ARM
  • Brought up UEFI on QC silicon - MSM8660
  • Primary QC engineer involved in Windows on ARM bring-up. UEFI, HAL, SD, USB, WiFi, etc.
  • Implemented Parking algorithm (multi-core)
  • Integrated/Demonstrated first functional case of the Windows Kernel power collapsing to Steven Sinofsky
  • Many other highlights

• INRIX (OpenCar) – Automotive Javascript UI/UX
  • Chromium Browser customizations
  • Re-wrote the OpenCar server. Went from 20 second load to 750ms. 1+GB RAM runtime footprint down to 1.5MB.
  • Target Demos + device input drivers
  • Made the Android solution a viable product
  • Started work on embedded flutter - cross compile engine + store app demo

• PureWatercraft – Electric Drivetrain
  • Owner of Throttle (Embedded Linux + UI) and Battery Charger
  • Implemented J1939 Stack for auto-addressing multiple Batteries (STM32/C)
  • Manufacturing stations
  • Enabled series A fundings (General Motors)

• MSFT Surface - PLE Team (Post Launch Engineering)
  • Surface ProX (8CX)

• Author/Creator of https://github.com/meta-flutter/meta-flutter

• Author/Owner of ivi-homescreen (flutter-auto) @ Toyota
Embedded Flutter
Components

• Development Environment
• Build Environment
• Target Environment
Flutter Engine

• Core component of the Flutter technology
• Written in C++ 17
• Common source tree for all platforms
  • Desktop - Mac/Windows/Linux
  • Mobile - Android / iOS
  • Web
  • Fuchsia
  • Custom Embedder
Flutter Engine

- Build environment based on Google GN
- Dart + SKIA
- Impeller
- Custom Embedder Backend support
  - OpenGL
  - Software
  - Metal
  - Vulkan
Benefits

• Premium User Experience
• Developer Experience
• Commodity Talent Pool
• Reduced NRE
• Time To Market
Development Environment
Goals

• Shortest path to run a Flutter App on AGL
• Easy to change between Flutter SDK versions
• Support unique configurations
• Archive friendly
• Ubuntu 20+ support
Solution

• Flutter Workspace Automation
What does it do?

• Creates a Flutter Workspace
  • Clones Flutter SDK
  • Sets up local Flutter SDK config
  • Sets up local pub cache
  • Fetches defined artifacts and installs runtime dependencies
  • Clones defined repositories
  • Creates setup_env.sh
Workspace Components

• Flutter SDK
• Sandboxed Flutter SDK config
• Sandboxed pub cache
• Platform Setup
  • Runtime
    • Binary
    • Required Dependencies
    • Custom-Device Config
• Development Repositories
  • VS Code launch.json
Install Method – AGL Source Tree

- cd $AGL_TOP
- external/meta-flutter/tools/setup_flutter_workspace.py meta-agl-devel/meta-agl-flutter/tools/flutter_workspace_config.json
Install Method – Tip of Tree

- **mkdir -p $HOME/workspace && cd $HOME/workspace**
- **curl --proto "https" --tlsv1.2 --ssFf https://gerrit.automotivelinux.org/gerrit/gitweb?p=AGL/metadataagl-devel.git;a=blob_plain;f=meta-agl-flutter/tools/flutter_workspace_config.json;hb=HEAD -o flutter_workspace_config.json**
- **curl --proto "https" --tlsv1.2 --ssFf https://raw.githubusercontent.com/meta-flutter/meta-flutter/kirkstone/tools/setup_flutter_workspace.py | python 3**
Script Options

- ./setup_flutter_workspace.py --help
  
  [--target-user TARGET_USER] [--target-address TARGET_ADDRESS]

options:
- -h, --help show this help message and exit
- --clean Wipes workspace clean
- --workspace-cfg WORKSPACE_CFG
  Selects custom workspace configuration file
- --flutter-version FLUTTER_VERSION
  Select flutter version. Overrides config file key: flutter-version
- --target-user TARGET_USER
  Sets custom-device target user name
- --target-address TARGET_ADDRESS
  Sets custom-device target address
Setup Caveats

• Flutter IDE tooling
  • Flutter tooling uses File Watching to trigger events. This conflicts with the setup script. To prevent this interaction run from a new system terminal instance, or temporarily disable the tooling.

• Multiple ‘flutter’ entries in system path
  • The setup script will attempt to remove the first occurring instance of ‘flutter’ from PATH when it runs, if the resolved path matches. To prevent this, remove all entries of ‘flutter’ from your path.

• Your Host Machine is expected to have hardware Hypervisor support enabled
  • If you explicitly want software Hypervisor support remove ‘-enable-kvm -cpu kvm64’ from the QEMU arguments in flutter_workspace_config.json before running.
Debug on flutter-auto desktop

- Login via GDM Wayland Session
- Open Terminal and type
  - `source ${FLUTTER_WORKSPACE}/setup_env.sh`
- Navigate to your favorite app
- `flutter run --d desktop-auto`
Debug on AGL QEMU

• Open Terminal and type
• `source ${FLUTTER_WORKSPACE}/setup_env.sh`
• Type `qemu_run`
• Wait until QEMU image reaches login prompt
• Run `ssh -p 2222 root@localhost who` to add remote host to `~/.ssh/known_hosts`
• Navigate to your favorite app
• `flutter run --d AGL-qemu`
Debug using Visual Studio Code

• Open Terminal and type
  • source ${FLUTTER_WORKSPACE}/setup_env.sh
  • code .

• Navigate to the debug pane

• Select application + runtime environment from drop down combo box

• Click the play icon to start debug session
VS Code launch.json creation

• `setup_flutter_workspace.py` creates a `.vscode/launch.json` file if one is not present.
• It uses the repo configuration key `pubspec_path`.
• If this key is present in the repo entry, then it will add entry to `.vscode/launch.json`.
How Custom Devices work

• flutter config –enable-custom-devices
• flutter doctor –vv
  • Enables viewing ping interaction with custom device
• Ping callback is made by flutter tooling, if pingSuccessRegex matches ping response, then flutter tooling lists device as being available
  • E.g. If not running a Wayland session, then desktop-auto will not be available
• Additional callbacks
  • postBuild – creates staged bundle folder
  • uninstall – removes bundle folder from target device
  • install – installs staged bundle folder into /tmp folder of target device
  • runDebug – invokes flutter-auto in platform environment
• See flutter_workspace_config.json -> “custom-device” for specific platform implementation
Resources

• https://github.com/meta-flutter/meta-flutter/tree/kirkstone/tools
• https://gerrit.automotivelinux.org/gerrit/gitweb?p=AGL/meta-agl-devel.git;a=blob;f=meta-agl-flutter/README.md
Labs

• Setup Flutter Workspace
• Create AGL Flutter Application
• Debug AGL Flutter Application – CLI
• Debug AGL Flutter Application – Visual Studio Code
Setup Flutter Workspace (1/3)

• Copy and paste the command below

```bash
rm -rf ~/workspace
```

![Screenshot of the command execution in a terminal](image.png)
• The command brings up a new terminal to set up the flutter workspace.
Once the setup is completed, the "Setting Up Flutter Workspace" terminal will close automatically.
Create AGL Flutter Application (1/2)

• Setup environment with the command below:

source ${FLUTTER_WORKSPACE}/setup_env.sh
cd ${FLUTTER_WORKSPACE}/app
flutter create hello_world -t app
cd hello_world
flutter run -d desktop-auto
source workspace/setup_env.sh
cd workspace/app/gallery
flutter run -d desktop-auto
Debug AGL Flutter Application – Visual Studio Code (1/3)

```
cd workspace
source setup_env.sh
code .
```
Debug AGL Flutter Application – Visual Studio Code (2/3)
Debug AGL Flutter Application – Visual Studio Code (3/3)
Linux GTK Embedder

• Canonical is primary development partner
• Flutter SDK only supports host only builds
  • No cross compilation support
• Runtime library dependency list is very big
• Applicable to Desktop class processors
• meta-flutter supports cross compiling the required target artifacts
• Flutter SDK support is missing to consume artifacts
Platform Views

- Using Platform views in Flutter dramatically decreases your potential framerate
- Avoid usage
- flutter-auto does not support for this reason
Platform Channels

• Dart - Native code bridge
• Platform Channel communication adds ~10ms latency per message
• Suitable for lifecycle calls or to support pre-existing platform constructs
Foreign Function Interface (FFI)

- Enables calling native C APIs from Dart code
- Zero latency
- No message passing
- No async/await on Dart
- No garbage collection
1P Linux Plugins

• 1P Linux Plugins are only intended for the Linux GTK embedder
• The use of the term “Linux Plugins” was poorly chosen
• In no way does it mean that “1P Linux Plugins” work with any Flutter embedder that runs on Linux
• It should really be “1P Linux GTK Plugins”
• The fact “1P Linux Plugin” Dart code runs in Flutter Debug builds is a Flutter bug and tracked here:
  • https://github.com/flutter/flutter/issues/103660
• Some solutions involve forking Flutter SDK to support a custom plugin type (not linux). Not a longterm solution
Build Environment
Components

- Yocto Layers
  - meta-agl-demo
  - meta-agl-devel/meta-agl-flutter
  - meta-flutter
- flutter-auto
  - Toyota ivi-homescreen *agl* branch
meta-agl-demo

https://gerrit.automotivelinux.org/gerrit/gitweb?p=AGL%2Fmeta-agl-demo.git
Flutter Image

• agl-ivi-demo-platform-flutter
  • Runtime = Release
  • Flutter Apps
    • Dashboard
    • HAVC
    • Navigation
    • Media Player
meta-agl-devel

https://gerrit.automotivelinux.org/gerrit/gitweb?p=AGL/meta-agl-devel.git
Images

• agl-image-flutter-runtimedebug
  • Runtime = Debug
  • SSH server
  • Flutter Engine SDK
• agl-image-flutter-runtimeprofile
  • Runtime = Profile
  • Same pattern as agl-image-flutter-runtimedebug + Apps
• agl-image-flutter-runtimereslease
  • Runtime = Release
  • Same pattern as agl-image-flutter-runtimereslease
bbappends

- Disables Gstreamer build flag for flutter-auto
- Enables network access for Archiver
- Flutter Gallery
  - Adds User Service
meta-flutter

https://github.com/meta-flutter/meta-flutter
Flutter App Bundle

• `<Bundle folder>`
  • data
    • flutter_assets
  • lib
    • libapp.so

• Origin - Flutter GTK runtime folder structure
recipes-devtools

- depot-tools – consumed by engine build
- flutter-rust-bridge-example
- membrane-example
- Rust (proc2) support
recipes-graphics

- agl-flutter-apps
  - flutter-app-igalia-homescreen
  - flutter-app-pumped-fuel-ped
- flutter-apps
  - flutter-gallery
  - flutter-test-animated-background
  - flutter-test-frb
  - flutter-test-localization
  - flutter-test-membrane
  - flutter-test-plugins
  - flutter-test-secure-storage
  - flutter-test-texture-egl
  - flutter-test-video-player
recipes-graphics

• flutter-engine
• flutter-pi
• flutter-sdk
• sony
• toyota
  • flutter-auto – AGL branch
  • ivi-homescreen – Quarterly release
tools

- Flutter Workspace Automation
  - `flutter_workspace_config.json`
    - Specific to builds available on meta-flutter
  - `setup_workspace_flutter.py`
    - Authoritative Repo
Kirkstone CI Jobs

- Linux-dummy – Layer Canary Build
- AGL QEMU x86_64 – master
- AGL Renesas M3 - master
- imx8mmevk
- Qualcomm DragonBoard 410C + 820C
- Raspberry PI Zero 2W
- STM32MP15
- Workspace
Dunfell CI Jobs

• Linux-dummy – Layer Canary Build
• Nvidia Jetson Nano
• Nvidia Jetson Xavier NX
• RPI4
• STM32MP15
• Variscite dart-mx8m-mini
Honister CI Jobs

- Linux-dummy – Layer Canary Build
- RPI Zero2W
- RPI3 32-bit
- RPI3 64-bit
- RPI4 32-bit
- RPI4 64-bit
flutter-auto

https://github.com/toyota-connected/ivi-homescreen/tree/agl
Features

• Wayland based
  • aegl_shell
  • xdg_shell
• Same code runs on Desktop and Target
• Multi-View
  • Single Process Multiple Engines/Surfaces
• Backend support (compile time)
  • EGL
  • Vulkan
• JSON configuration
• Bundle Override Logic
• more planned...
Command Line Options

- **--a={int value}** - Sets the Engine's initial state of Accessibility Feature support. Requires an integer value.
- **--c** - Disables the cursor.
- **--d** - Outputs backend debug information. If Vulkan and Validation Layer is available, it will be loaded.
- **--f** - Sets window to fullscreen.
- **--w={int value}** - Sets View width. Requires an integer value.
- **--h={int value}** - Sets View height. Requires an integer value.
- **--t={String}** - Sets cursor theme to load. e.g. **--t=DMZ-White**
- **--b={path to folder}** - Sets the Bundle Path.
- **--j={json config}** - Sets the JSON configuration file.
- Dart VM arguments - any additional command line arguments not handled get directly passed to the Dart VM instance.
JSON Configuration

• Parameter Loading Order
  • JSON – View
  • JSON – Global
  • CLI Arguments

• If there are redundant key/values they will be overwritten.

• e.g., CLI Arguments override all
Bundle Override Logic

• Optional paths are checked first. If file exists, it will be used.

• <Bundle folder>
  • data
    • flutter_assets
    • icudtl.dat <optional>
  • lib
    • libapp.so
    • libflutter_engine.so <optional>
Default Build Flags

- BUILD_BACKEND_WAYLAND_DRM:BOOL=OFF
- BUILD_BACKEND_WAYLAND_EGL:BOOL=ON
- BUILD_EGL_TRANSPARENCY:BOOL=ON
- BUILD_PLUGIN_ACCESSIBILITY:BOOL=ON
- BUILD_PLUGIN_GSTREAMER_EGL:BOOL=ON
- BUILD_PLUGIN_ISOLATE:BOOL=ON
- BUILD_PLUGIN_MOUSE_CURSOR:BOOL=ON
- BUILD_PLUGIN_NAVIGATION:BOOL=ON
- BUILD_PLUGIN_OPENGL_TEXTURE:BOOL=ON
- BUILD_PLUGIN_PACKAGE_INFO:BOOL=ON
- BUILD_PLUGIN_PLATFORM:BOOL=ON
- BUILD_PLUGIN_PLATFORM_VIEW:BOOL=OFF
- BUILD_PLUGIN_RESTOREATION:BOOL=ON
- BUILD_PLUGIN_SECURE_STORAGE:BOOL=OFF
- BUILD_PLUGIN_TEXT_INPUT:BOOL=ON
- BUILD_PLUGIN_URL_LAUNCHER:BOOL=ON
- BUILD_TEXTURE_TEST_EGL:BOOL=OFF
CI Job

Labs

• Run flutter apps in runtime=release image
• Auto-run flutter app using system service
• Multi-View
• Run flutter app in runtime=profile image
• Setup a Flutter workspace for flutter-auto

$ mkdir workspace
$ cd workspace
$ wget https://raw.githubusercontent.com/meta-flutter/meta-flutter/kirkstone/tools/setup_flutter_workspace.py
$ python3 setup_flutter_workspace.py
Run flutter apps in runtime=release image (2/5)

demo@demo::--$ mkdir workspace
demo@demo::--$ cd workspace
demo@workspace$ wget https://raw.githubusercontent.com/bill9412618/flutter-auto-demo/main/run_flutter_app_in_release_image/flutter_workspace_config.json
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.111.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 5595 (5.5K) [text/plain]
Saving to: ‘flutter_workspace_config.json’
flutter_workspace_c 100%[================================]=> 5.46K --.KB/s in 0s
2022-10-06 16:17:45 (31.1 MB/s) - ‘flutter_workspace_config.json’ saved [5595/5595]
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 48065 (47K) [text/plain]
Saving to: ‘setup_flutter_workspace.py’
setup_flutter_works 100%[================================]=> 46.94K --.KB/s in 0.02s
2022-10-06 16:17:58 (2.88 MB/s) - ‘setup_flutter_workspace.py’ saved [48065/48065]
demo@workspace$ python3 setup_flutter_workspace.py
Run flutter apps in runtime=release image (3/5)

Setup the environment and run the image on QEMU

```
$ source setup_env.sh
$ qemu_run
```
Run flutter apps in runtime=release image (4/5)

• Log in as root, delete agl-driver's password, exit
  
  $ passwd -d agl-driver
  $ exit

• Login as agl-driver
Run flutter apps in runtime=release image (5/5)

- Log in as agl-driver, and run the gallery app
  
  $ flutter-auto --b=/usr/share/flutter/gallery --window-type=BG
Run release image on QEMU and login as root

```bash
$ source setup_env.sh
$ qemu_run
```
Auto-run flutter app using system service (2/3)

• copy flutter-gallery.service to /usr/lib/systemd/user
• make a symbolic link for flutter-gallery.service in /usr/lib/systemd/user/agl-session.target.wants
• reboot
  $ cd /usr/lib/systemd/user
  $ cd agl-session.target.wants
  $ ln -s ../flutter-gallery.service flutter-gallery.service
  $ reboot
Auto-run flutter app using system service (3/3)

• After reboot, the gallery app runs automatically
Multi-View (1/4)

• Run release image on QEMU and login as agl-driver
Multi-View (2/4)

• Copy multi-view.json to /tmp

• Content of multi-view.json

```json
{
    "view": [
        {
            "bundle_path": "/usr/share/flutter/pumped_end_device",
            "window_type": "BG",
            "width": 1920,
            "height": 1080,
            "accessibility_features": 31
        },
        {
            "bundle_path": "/usr/share/flutter/animated_background_example",
            "window_type": "PANEL_TOP",
            "width": 1920,
            "height": 360,
            "accessibility_features": 31
        },
        {
            "bundle_path": "/usr/share/flutter/test_egl",
            "window_type": "PANEL_LEFT",
            "width": 320,
            "height": 1080,
            "accessibility_features": 31
        }
    ]
}
```
Multi-View (4/4)

• Run flutter-auto with the json file
• 3 Apps run simultaneously

$ flutter-auto --j=/tmp/multi-view.json
Run flutter app in runtime=profile image (1/5)

• Make a Flutter workspace for flutter-auto

$ mkdir workspace
$ cd workspace
$ wget https://raw.githubusercontent.com/meta-flutter/meta-flutter/kirkstone/tools/setup_flutter_workspace.py
$ python3 setup_flutter_workspace.py
Run flutter app in runtime=profile image (2/5)

- Setup environment
  - $ source setup_env.sh
  - $ qemu_run

- Login as root

- Delete agl-driver's password
  - $ passwd -d agl-driver
  - $ exit

- Login as agl-driver
Run flutter app in runtime=profile image (3/5)

- Run the gallery app
  
  $ flutter-auto --window-type=BG --b=/usr/share/flutter/gallery --f --observatory-host=0.0.0.0 --observatory-port=1234

- Note the URL which Dart VM service is listening on
Run the following commands on the host

$ flutter pub get
$ flutter attach --device-id=AGL-qemu --debug-url=http://127.0.0.1:1234/YLSuCEGH52A=/

Note the output shows the URL for debugger and profiler.

Press v to bring up the debugger and profiler on Chrome browser
Run flutter app in runtime=profile image (5/5)
Resources

- https://docs.flutter.dev
- https://github.com/flutter/flutter/wiki
- https://www.yoctoproject.org
- https://github.com/meta-flutter/meta-flutter
- https://github.com/toyota-connected/ivi-homescreen
- https://www.automotivelinux.org
- https://docs.automotivelinux.org/en/needlefish/#5_Component_Documentation/1_agl-compositor/