

# Plasma Internals

The next few years

**Presented By:**  
- *Marco Martin*

# Scope of the talk






18 – 25 June 2021

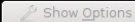
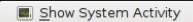


- Here “Plasma” is in the context of purely the desktop shell and its accompanying framework
- An introduction on how some internal things in plasma-framework and plasmashell work
- How the startup process works
- How plasmoids are loaded
- What’s the api between plasmoid and the framework
- What we can improve for Plasma 6
- A proposed approach

The KDE 4.0  
Desktop

# HISTORY

Enter the name of an application, location or search term below. 

wp:clock  



18 - 25 June 2021

# Plasma4: parts



18 – 25 June 2021

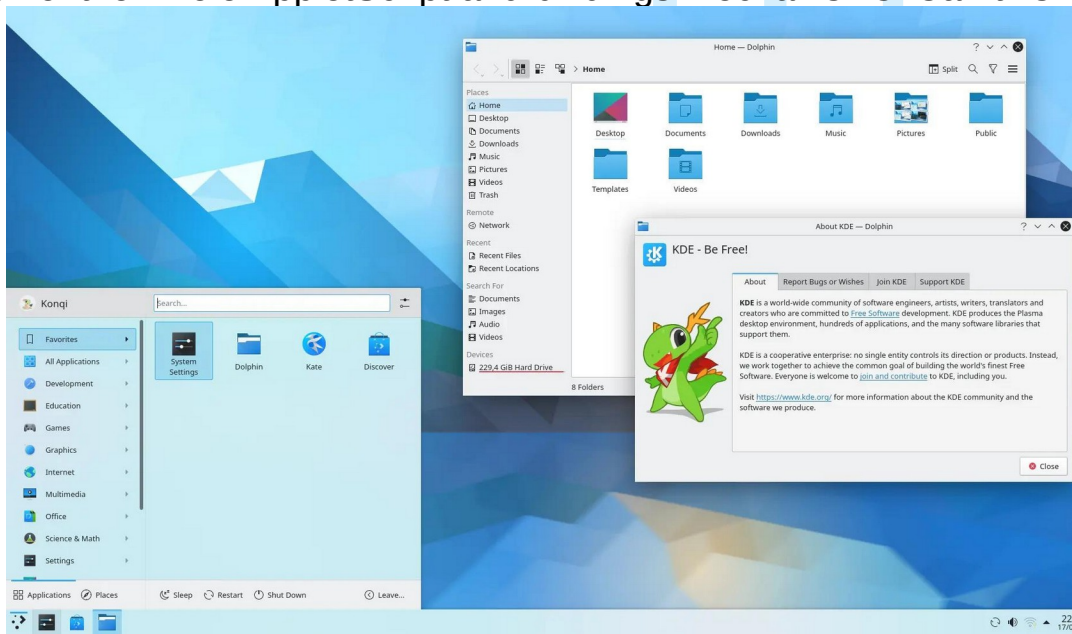
- Plasma was born in KDE 4.0
- Based on a scene (at the times QGraphicsScene) called “Corona” (it was an innocent name at the time ) that manages views (panels and desktops)
- Each view shows a graphic object called a “Containment”
- Each Containment can indeed, contain many “Applets”
- Both Containments and applets are “Plasmoids” (Containment is a subclass of Applet, comes together a package of file, together they make a plasmoid)
- Applets were a QGraphicsWidget instance, most plasmoids were entirely in C++
- Bindings were there for multiple languages: JavaScript, Python and Ruby, later in the lifecycle, QML

# Plasma 5



18 – 25 June 2021

- Out with QGraphicsScene, all in QML scene graph!
- Corona, Applet and Containment are just QObjects
- Only one language binding: QML which is mandatory to use, even if Applets still have an optional C++ plugin-based components
- Even if the only one, there is still the crutch of the whole AppletScript and bindings mechanisms. Can this be simplified?



# What parts are still needed?



18 – 25 June 2021

- The Layout loading part with Corona (needing a better name ) , Containment and Applet is largely fine
- Just some minor API fixes are needed
- Biggest part to fix there is the Containment screen ownership by number, for better multiscreen support
- Some pieces should be independent frameworks
- Some other pieces could be not needed anymore at all

# Svg Framesvg, Theme



18 – 25 June 2021

- Plasma Svg loading is pretty nice
- On disk caching of the rendering cuts down load times of often-used assets
- Datasheet support to have graphics that recolors itself from the system colorscheme
- 9-patches svgs (FrameSvg) are scaled using hardware acceleration
- Should be usable from everywhere without many dependencies, so should be its own framework
- Both from QML with some basic acceleration for FrameSvg or in qwidget apps as well with a QPainter API



# Dataengines



18 – 25 June 2021

- Done in the beginning to separate data and representation in the widgets
- Designed together with the JavaScript Plasmoid bindings (pre-QML) which were very limited
- So the applet written in JavaScript didn't have any heavy logic at all, most of the operations were done with the jobs-based Dataengine and service API
- It was a good idea, but ended up being way more clumsy to use than a simple QML binding with a simple QObject with properties and methods (that or C++ Applet subclasses which are bound with QML as well)
- In Plasma6 we ideally wouldn't need at all
- At most as a Plasma5 support library that just to be used for things that aren't ported away yet, then we can retire them



# Layout loading



18 – 25 June 2021

- What you have in the desktop is stored in the appletsrc config file
- Corresponds to the Corona (from now “the scene”)
- First level of config represents containments
- The scene will read this first level and create the containments
- Each containment will then restore its own layout, reading the second level of the configuration nodes under its own, and create the applets
- Each Applet (and Containments) will load the QML Scriptengine, which will create an AppletInterface instance for each which is a QQuickItem and will load with Kpackage the qml files of the applet ui implementation
- The AppletInterface instance is what is seen on qml side as that global “**plasmoid**” object

# Shell packages

- Introduced in Plasma 4 times with Plasma Active
- Became a central part of Plasma 5 on all form factors
- Some behavioral parts on how the “desktop” screen is loaded
- Panels behavior
- Look of configuration dialogs and applets popups in panels
- Can we make them a bit more powerful?



18 – 25 June 2021



# Scriptengines

- They duplicate and wrap almost the whole api of Applet and Containment
- Not really needed anymore
- Can we load qml applets more “directly” without them?



18 – 25 June 2021



# Prototype

- <https://invent.kde.org/mart/plasma-framework> fork
- work/mart/pf6 branch
- Still not completely functional, but has basic refactored layout loading in place (only a very stupid and basic shell, real plasmashell still to port)



18 – 25 June 2021

# Proposed lifecycle



18 – 25 June 2021

- As is now, slightly different between panels and desktop views
- The scene creates a desktop view for each screen, each view gets assigned a containment
- The view gets the qml instance for the containment, or loads it if it doesn't exist yet
- The containment at this point instantiates an AppletContainer instance for each Applet
- AppletContainer instantiates the qml files of the applet, and the optional expander from the shell (ie popups from the panel)
- Since AppletInterface is dead, “plasmoid” will be the actual Applet\* instance directly, removing a level of indirection
- For panels, a view is created for each panel containment, and then the same process happens

# Applet QML structure



18 – 25 June 2021

- In a KPackage, as always
- As in QQC2/Kirigami applications need to have an instance of ApplicationWindow as their root element, plasmoids will need to have an instance of PlasmoidRepresentation as their root element
- It's a simple QObject, not a graphical QQuickItem
- It provides the compactRepresentation and fullRepresentation component properties, which will be instantiated as needed
- I would have a single one, not splitted popupapplets/normal applets like Plasma4, perhaps explicitly set null compactRepresentation would be ok and semantically very self explanatory
- Anywhere in the plasmoid's QML will be possible to access the Plasmoid.\* context property
- Not a contextproperty anymore, they give problems, even if they weren't removed from Qt6 in the end, is better to avoid
- That will be the Applet\* instance, anywhere the attached property value will point to the same Applet\* pointer, so one single instance even if is an attached one

# A simple Plasma 6 plasmoid



18 – 25 June 2021

```
import QtQuick 2.15 //3
import QtQuick.Layouts 1.0
import org.kde.plasma.core 6.0 as PlasmaCore
import org.kde.kirigami 2.15 as Kirigami

PlasmaCore.AppletRepresentation { // This non graphical element is the mandatory root, won't load with a different one
    // As is today, 99% on the times shouldn't need a compactrepresentation
    compactRepresentation: Kirigami.Icon { // Hopefully this can replace PlasmaCore.IconItem
        source: "start-here"
        TapHandler {
            onTapped: PlasmaCore.Plasmoid.expanded = !PlasmaCore.Plasmoid.expanded
        }
    }
    fullRepresentation: Item {
        Layout.minimumWidth: Kirigami.Units.gridUnit * 10 //plasmacore units should be migrated to Kirigami
        Layout.minimumHeight: Kirigami.Units.gridUnit * 22
        Text {
            text: PlasmaCore.Plasmoid.title // all accesses to PlasmaCore.Plasmoid are pointers to the same Applet* instance
        }
    }
}
```

# More power to shell packages



18 - 25 June 2021

- Proposal to make shell packages significantly more powerful:
- Right now the customization qml for a view like desktop or panel, is an Item that gets loaded into the view, created from plasmashell directly
- Makes it impossible to make significantly different window behaviors on different shells, maybe embedded (maybe there you would even like to have the panel or config dialogs in the same window as the desktop there)
- Those qml files should probably instantiate their own window, for easier customization
- May make easier to integrate better with lattedock, so it can be loaded in plasma using its own plugin special panelview



# Will be done? When?



18 – 25 June 2021

- A big concern is to measure the level of disruption which is acceptable in order to keep having plasmoids easily portable
- Not all of those ideas are guaranteed to make it to the final version
- For now is just a quick and dirty fork, proper kf6 branching needs to happen in frameworks before development goes full speed
- To have more discussion on it, join the Bof on Wednesday, 16:00 UTC

# Thanks everybody



18 – 25 June 2021

## Questions?

