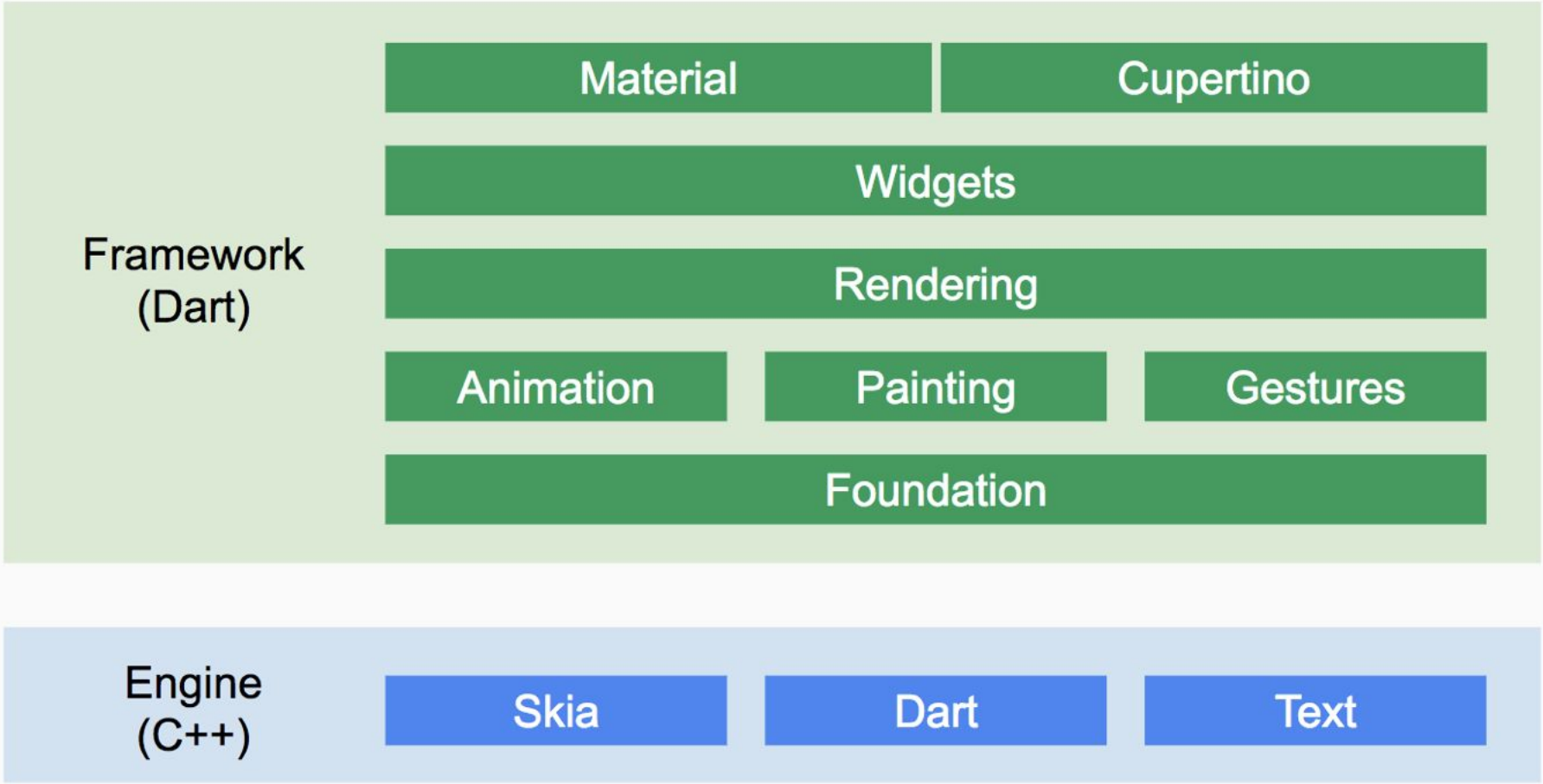
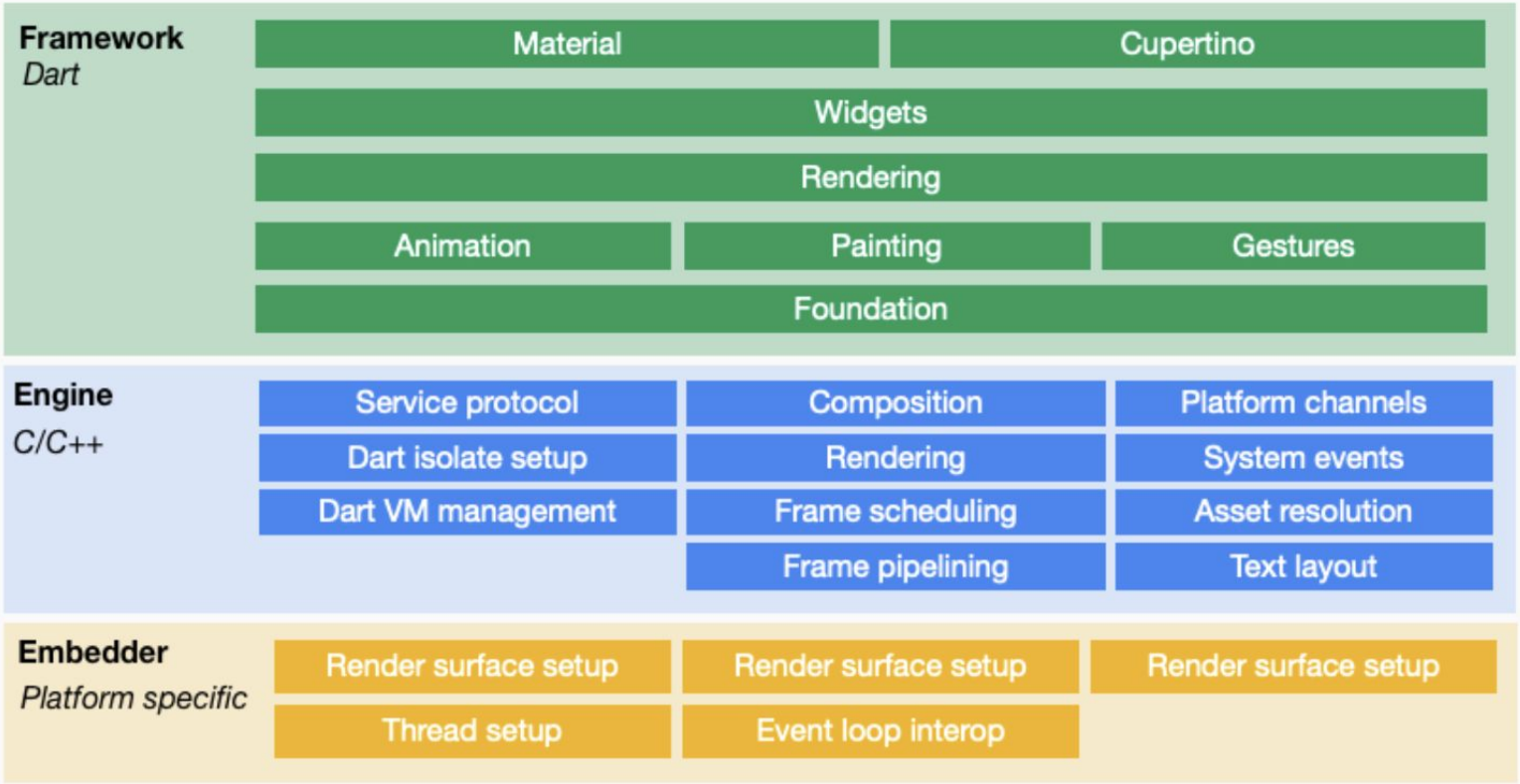
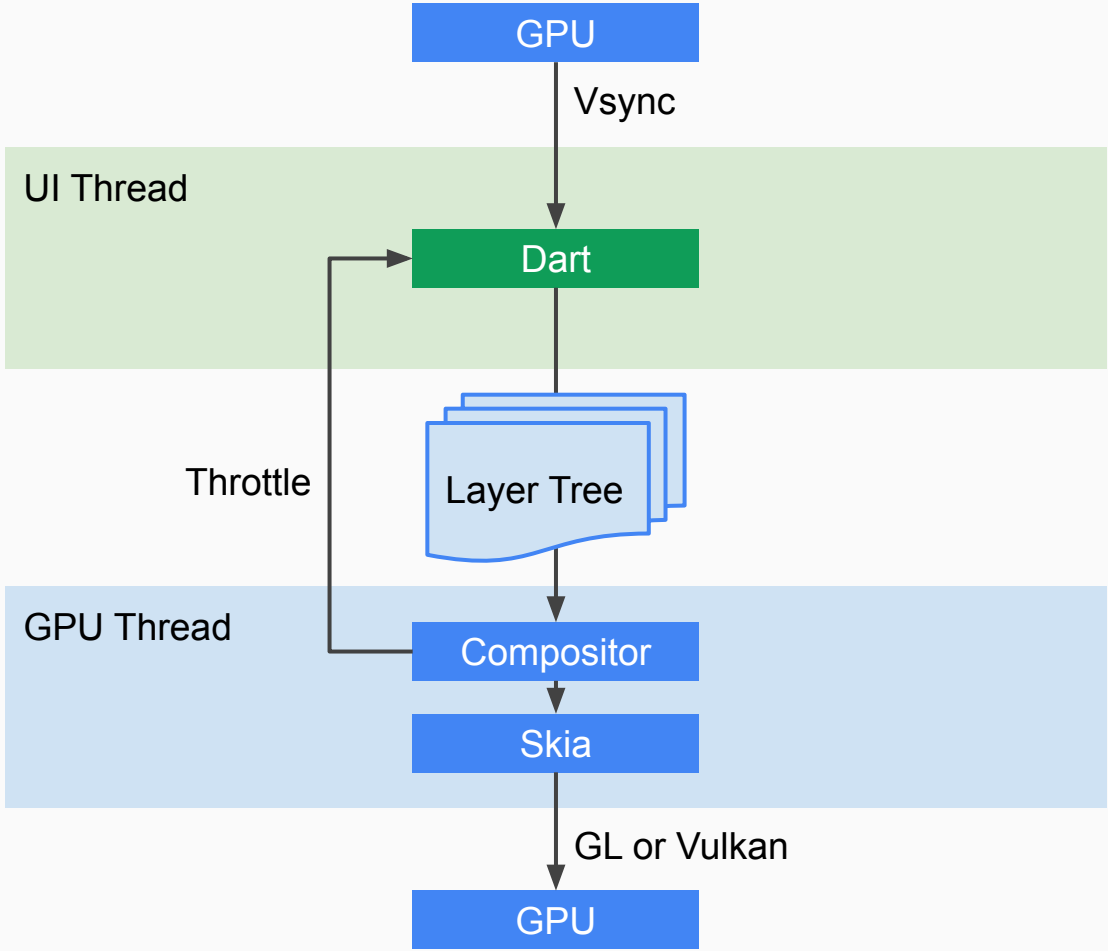
Flutter : And how it flows in Android

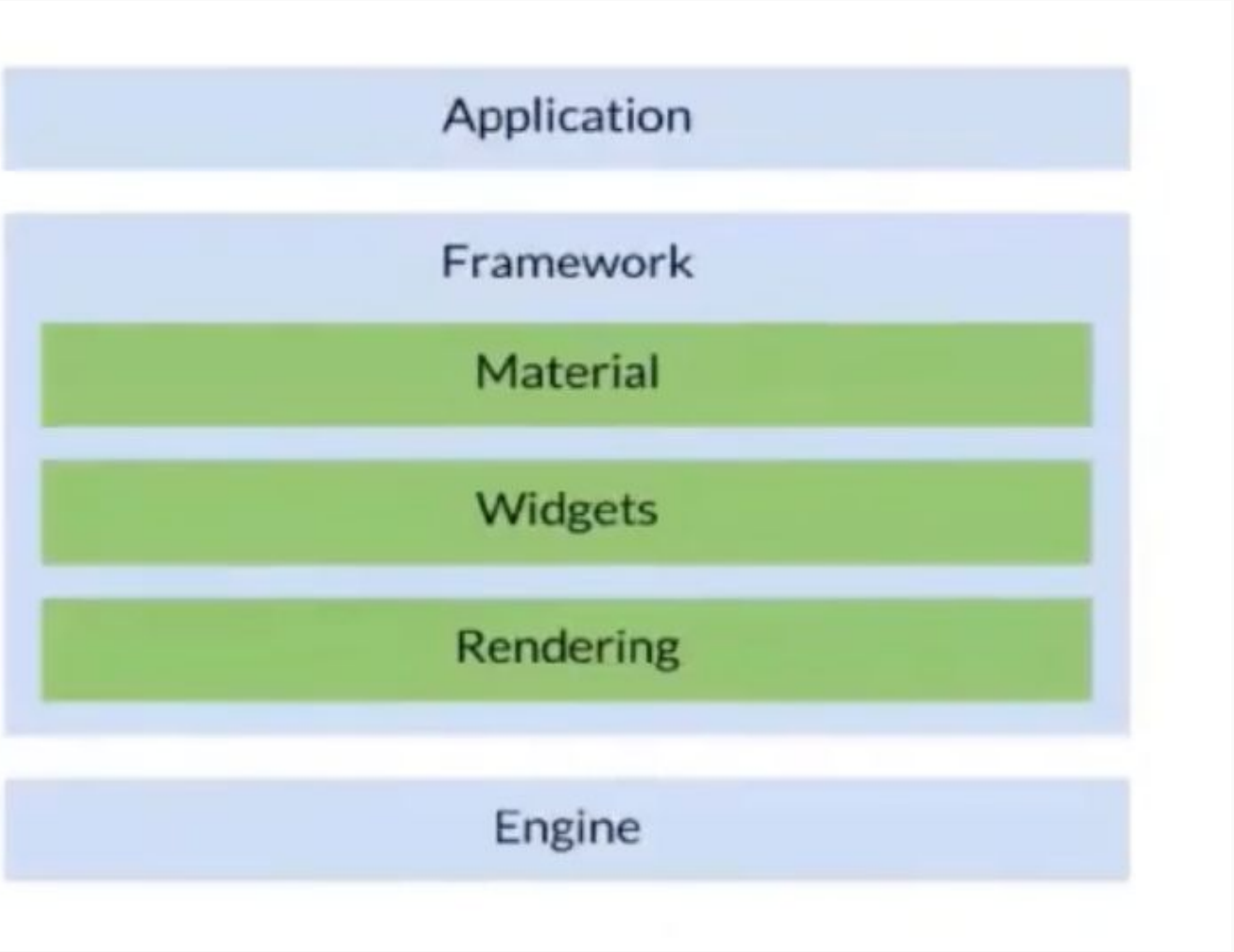
- Anupam Singh
- Sr. Android Dev
- UrbanClap
- Androidiots Podcast
- Not a Flutter Enthusiast or Evangelist


Framework
(Dart)
Material
Cupertino
Widgets
Rendering
Animation
Painting
Gestures
Foundation
Engine
(C++)
Skia
Dart
Text

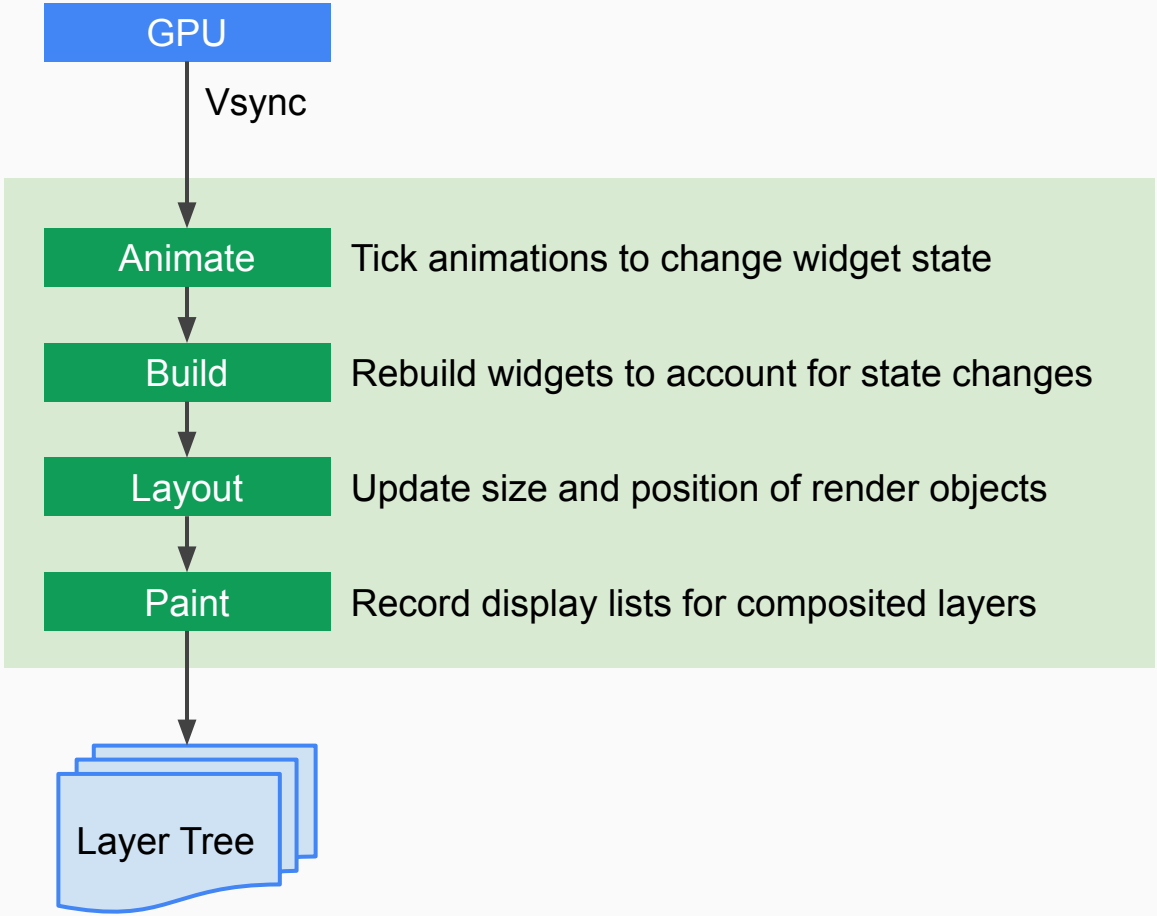

Framework
Dart
Material
Cupertino
Widgets
Rendering
Animation
Painting
Gestures
Foundation
Engine
C/C++
Service protocol
Composition
Platform channels
Dart isolate setup
Rendering
System events
Dart VM management
Frame scheduling
Asset resolution
Frame pipelining
Text layout
Embedder
Platform specific
Render surface setup
Render surface setup
Render surface setup
Thread setup
Event loop interop

Graphics Pipeline

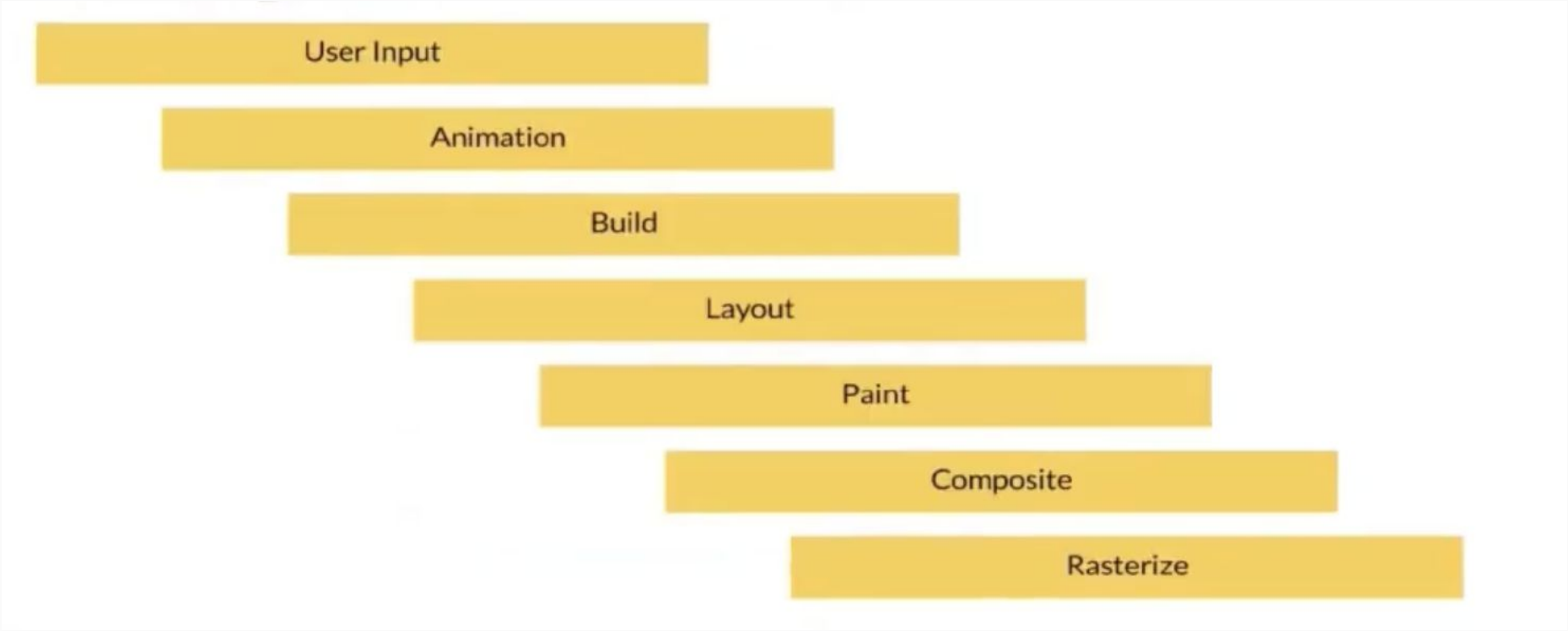



Application
Framework
Material
Widgets
Rendering
Engine

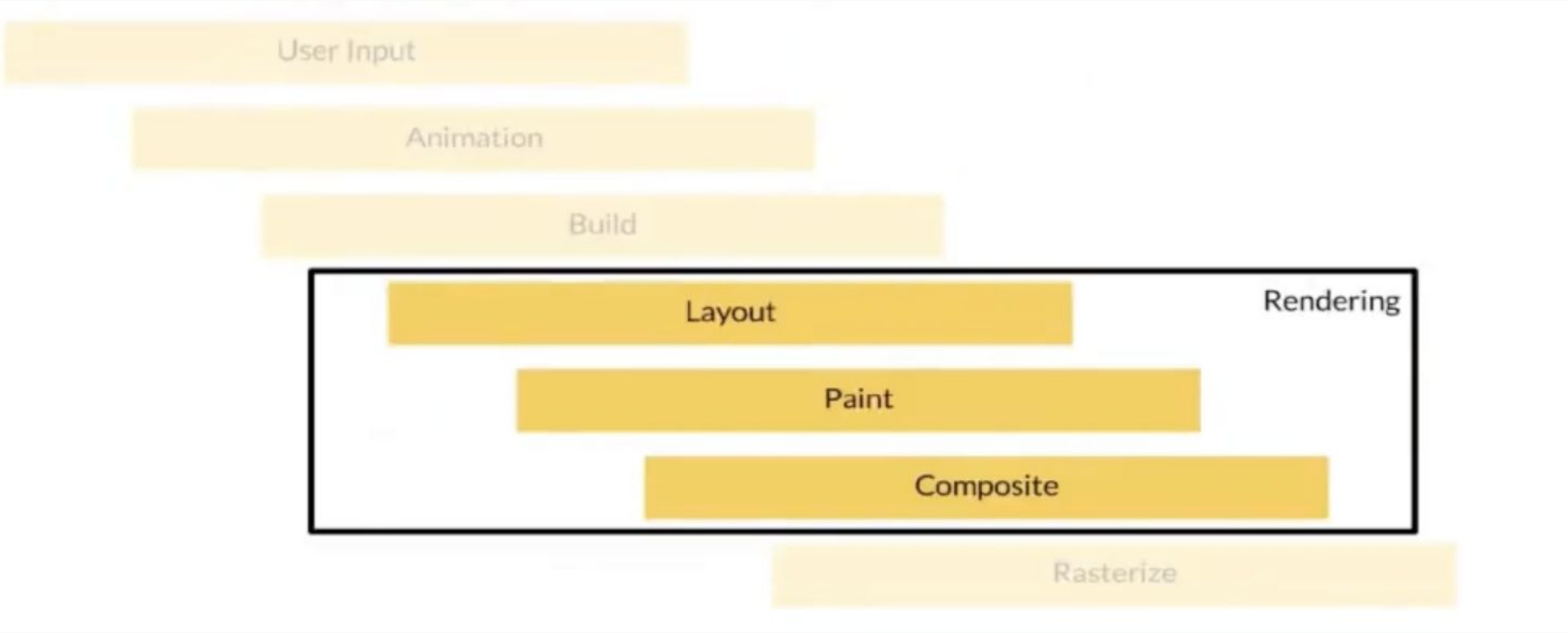
Rendering Pipeline



Flutter Pipeline



Rendering


User Input
Animation
Build
Layout
Rendering
Paint
Composite
Rasterize

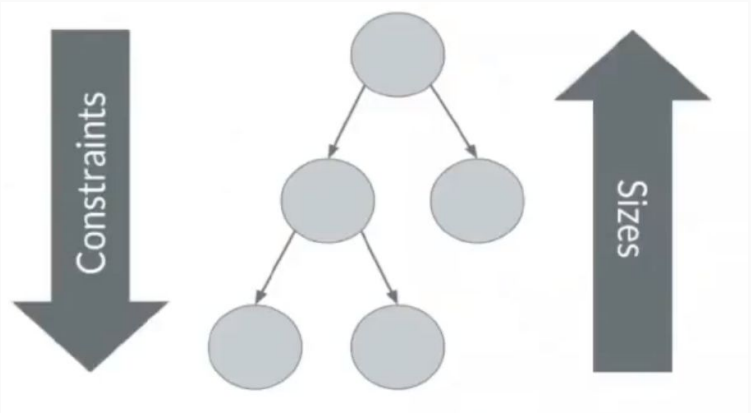
In Text

1. Layout Phase: in this phase, Flutter determines exactly how big each object is, and where it will be displayed on the screen.
2. Painting Phase: in this phase, Flutter provides each widget with a *canvas*, and tells it to paint itself on it.
3. Compositing Phase: in this phase, Flutter puts everything together into a *scene*, and sends it to the GPU for processing.
4. Rasterizing Phase: in this final phase, the scene is displayed on the screen as a matrix of pixels.

1) Layout

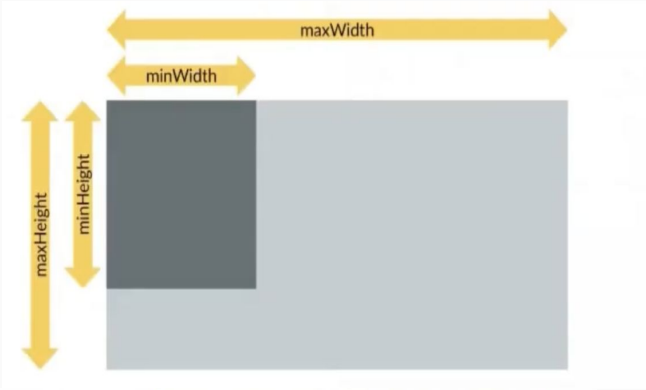
- Simple one pass linear algorithm
  - Walk the tree from top to bottom
  - Box constraints (min width, min height, max width, max height)
  - Structural Repainting using composting (from tracking rectangles to subtree)

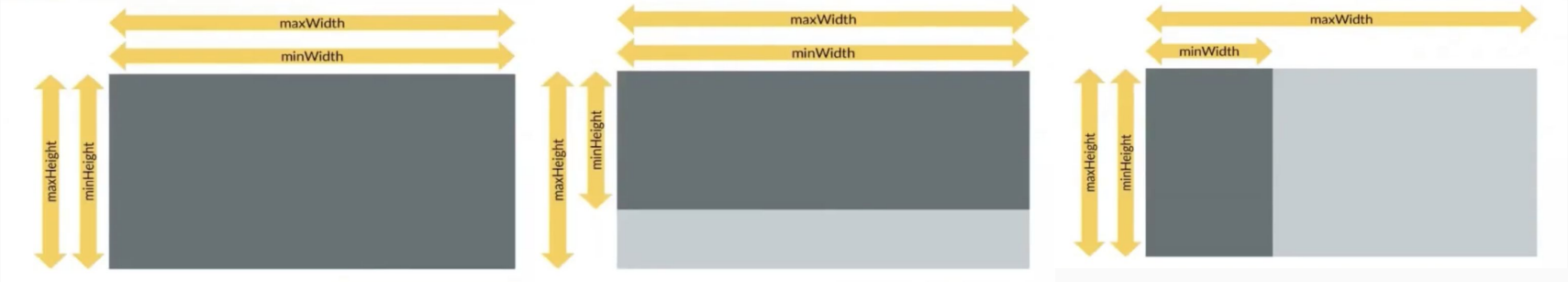
- RenderObject: Base class
  - Owner : Which drives the pipeline
  - Parent
  - layout(), parent(), parent Data
  - visitChildren()
    - Different render object are free to have different child models eg single, list etc
  - No coordinates just a parent



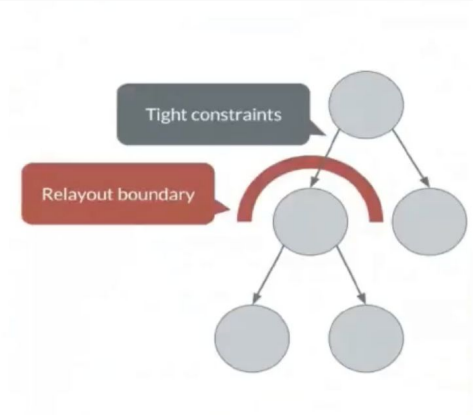
Too Abstract what about Coordinates?

- Render Box
  - Size
    - getMinIntrinsicWidth,getMaxIntrinsicHeight
    - getMinIntrinsicHeight, getMaxIntrinsicHeight



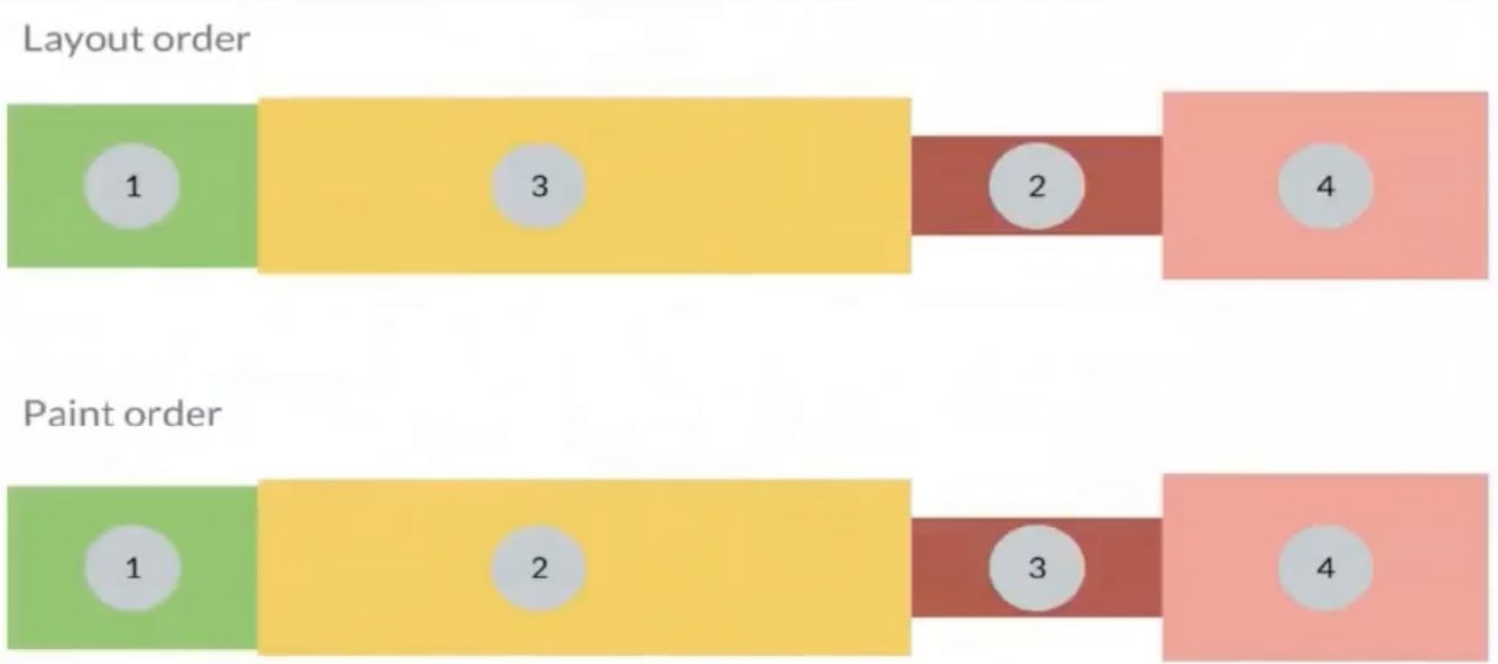


Edge cases

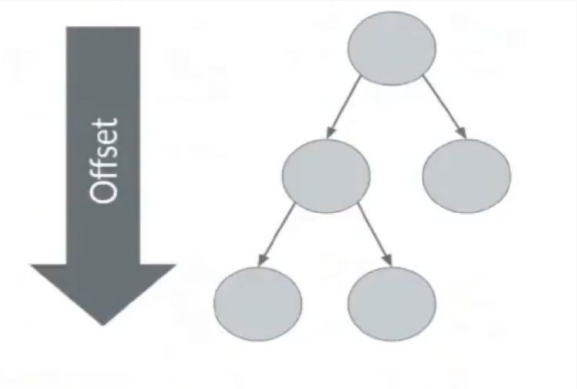


- Flex layout
  - Inflexible children are resolved first with infinite max width
  - Flexible children as a function of spans
- Tight Constraint by Parents
- ParentUsesSize
  - But child says no
- sizedByParent

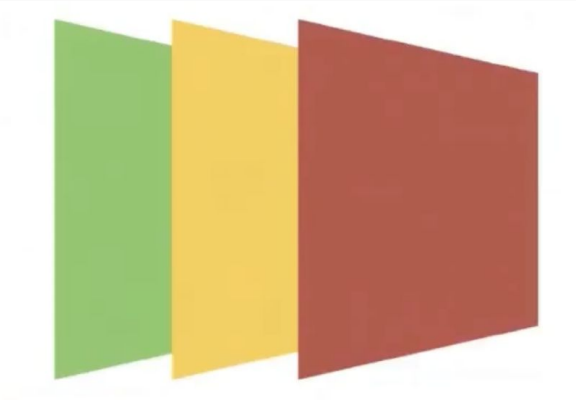
2 : Lets Paint
Paint order is not similar to information order of layout



Painting seems simple

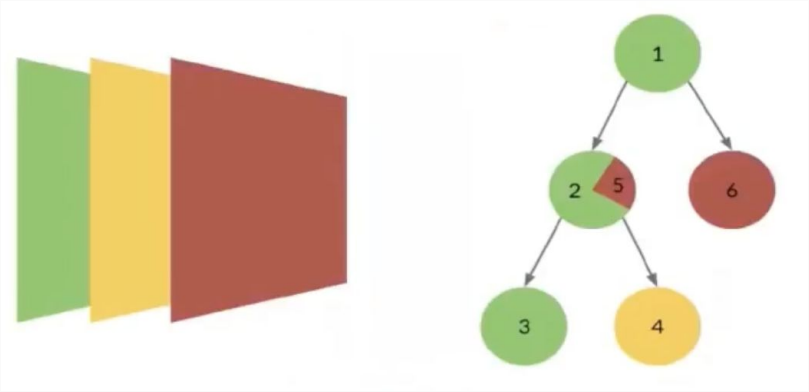


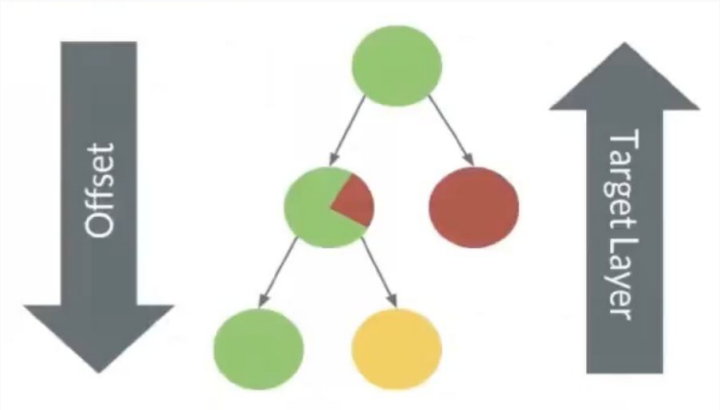
- With Painting Context and Canvas
- Walk the tree and pass around the offset??

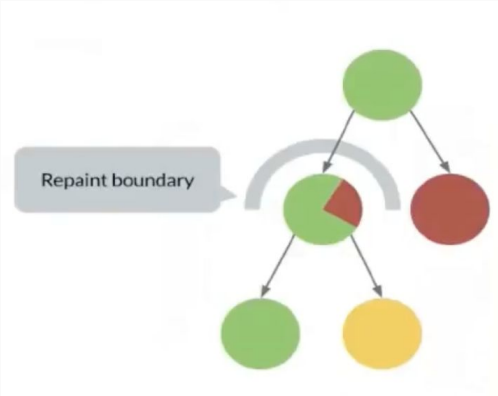


Naaah : Layers

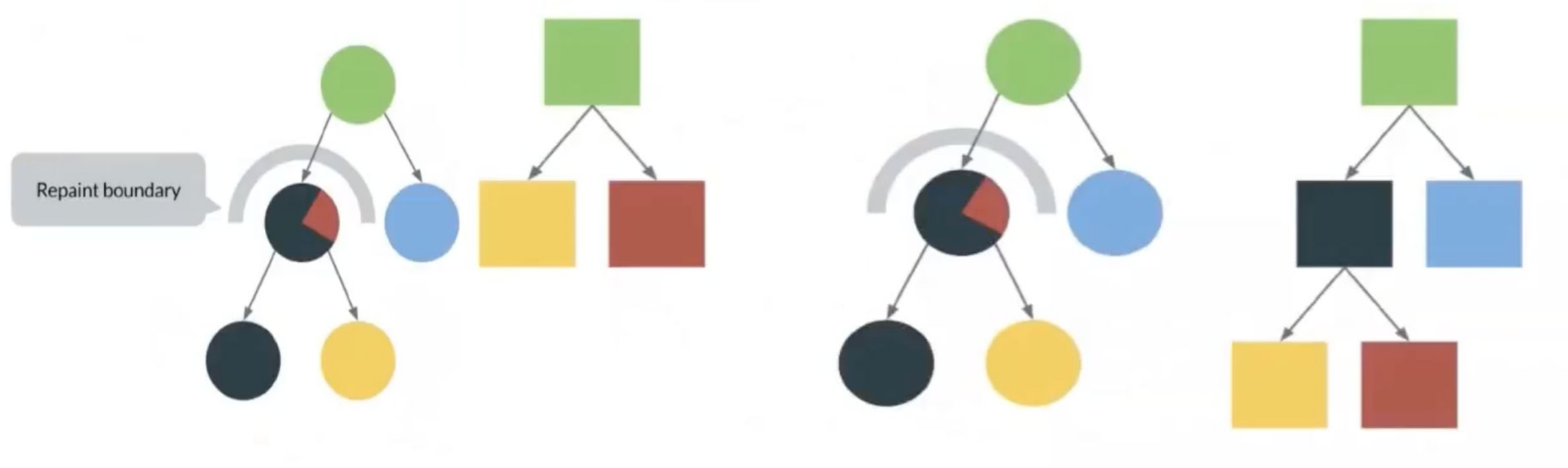
Let's paint a Video with background and Play button on top







Repaint Boundaries and effects



So What if i write my own custom complex widget??

3 : Composition
Update Visual Appearance, and do it fast

- Pixels are recorded in scene/textures and gpu renders these scenes as finite pixels
- Cocoa : Lots of memory make every display list a texture
- Android : No textures, as not much memory hence redraw every display list but make it very fast
- Flutter sits in between

Lets scroll

Every item is a layer

- Each layer can be vector
  - Display list of drawing command to execute
- Or list is baked as a texture, and system blitz the pixel on the screen

Question : When do we texturise?
When we don't?

4 : Rasterization

as per Flutter FAQ page

- The engine's C and C++ code are compiled with Android's NDK.
- The Dart code (both the SDK's and yours) are ahead-of-time (AOT) compiled into a native, ARM and x86 libraries.
- Those libraries are included in a "runner" Android project, and the whole thing is built into an APK. When launched, the app loads the Flutter library.
- Any rendering, input or event handling, and so on, is delegated to the compiled Flutter and app code. This is similar to the way many game engines work.

Thanks

No Questions Please

References

https://api.flutter.dev/flutter/rendering/rendering-library.html

https://medium.com/saugo360/flutters-rendering-engine-a-tutorial-part-1-e9eff68b825d

https://docs.google.com/presentation/d/1cw7A4HbvM_Abv320rVgPVGiUP2msVs7tfGbkgdrTy0I/edit#slide=id.p