Introduction

What is the Skia graphics library?

It's an open-source 2D graphics library that serves as the graphics engine for many hardware and software platforms such as Google Chrome, ChromeOS, Android, Flutter, and many other products. It provides common APIs that work across these platforms.

Why use Skia for this project?

Much of the application demands the data visualization; however, React Native does not have much support when it comes to graphics. We want to use Skia with its vast image effects, filters, and capabilities when there's a visualization we cannot make with other out-of-the-box packages provided by React Native.

- Provided developers know how to use Skia, the possibilities are endless, meaning components made with Skia are fully customizable.
- Runs on the UI thread, and not the main JS thread, which prevents any frame drops so it's highly performant.
- Compatibility with the Web, thanks to CanvasKit that is a WebAssembly build of Skia.
- Considering we will be fetching data in real-time, having graphics that perform well is what we desire.
- The Skia DOM API
    - Allows us to express any Skia drawing declaratively
    - Is platform-agnostic and framework-agnostic
    - Recomputes only necessary parts of the drawing if changed, optimizing performance levels

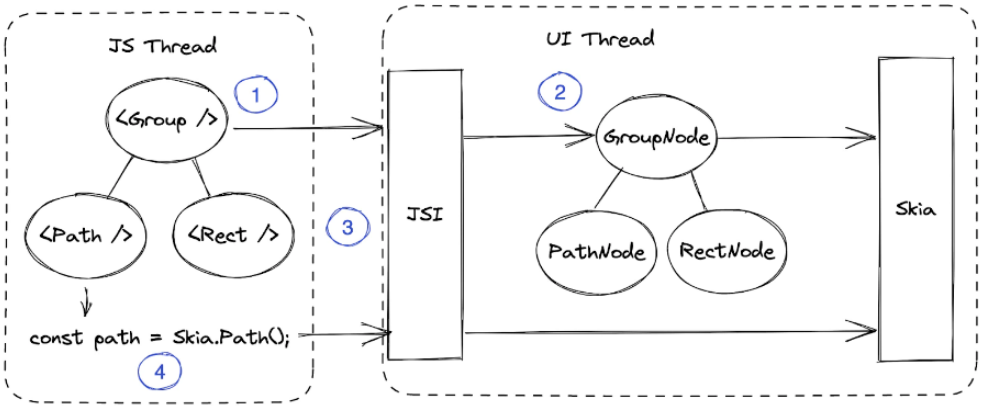

Figure 1.1. A diagram of how the Skia DOM API works.

1. A declarative representation of a Skia drawing (SkiaDOM) is built by the React reconciler via the JSI.
2. SkiaDOM has no dependencies on the JS thread, therefore it can very quickly draw on the UI thread.
3. The SkiaDOM API only draws once. It only recomputes when it gets updates from the JS thread for necessary parts of a drawing.
4. The Skia is directly available from a thin JSI layer. This is particularly useful for paths and shaders.

Source: https://shopify.engineering/react-native-skia-2022

Notes on Skia

Canvas

Canvas is basically the background for the drawing. One must set a size for the Canvas, which will be the “canvas” for the drawing. Once the size has been set, the component cannot cross the borders from the limitations of the Canvas. If it does, the part of the component crossing over is simply not displayed. Working Canvas alongside Views can be tricky, where the View must also be set a size for the Canvas to work.

- Props
    - Style? - set view style
    - Ref? - set reference to the SkiaView object
    - Mode?: default or continuous
        - Set component to continuously redraw every frame or not
    - onSize? - Reanimated value (uses SharedValue) that can set the Canvas size
    - onLayout? - invoked on mount and layout changes

Painting

We call painting attributes the things we want to specify for drawings, for example style, color, and how it blends with the background. Painting attributes can either be props or children of a drawing component (Rect, Circle).

- Fills and Strokes
    - Paint components can be passed as children to add strokes and fills.
- Inheritance
    - Descendants inherit paint attributes, for example from a *Group.*
    - Furthermore, complex attributes like shaders and image filters may also be passed down as children to a group or drawing.

- Manual Assignment
    - A variable may be set to Paint, which can be referenced to within the component as well.
- Properties
    - https://shopify.github.io/react-native-skia/docs/paint/properties

Group

Basically a component that passes properties down to its children. Group components can be deeply nested with each other.

- Paint Properties
    - Any paint attribute will be inherited by the children of a Group.
- Transformations
    - In React Native, we know transformations use the center of the object as the origin. **However, in Skia, the origin is the top-left position of the object.**
- Clipping Operations
    - Sets part of the children that should be shown. Displays inside the region, but not outside of it. *invertClip* does the opposite.
- Layer Effects
    - Creates a bitmap drawing of the children. Useful to create effects for a group of children, but not one in particular.
- Fitbox
    - Allows one to scale drawings to fit into a destination rectangle automatically.
    - For example, scales path to the size of the canvas.

Shapes

Path

Semantically identical to SVG paths.

- Notations that can be used:
    - SVG Notation
    - Path object
- Trim
    - Parts of the path may be trimmed through the *start* and *end* props.
- Fill Type
    - Defines the algorithm to use to fill the inside part of the shape.
    - **winding** - default, just fills the shape inside
    - **evenOdd** - if number of boundary lines is odd, fills the area
    - **inverseWinding** - the inverse of winding, fills outside

- **inverseEvenOdd** - opposite of evenOdd

Polygons
https://shopify.github.io/react-native-skia/docs/shapes/polygons

Ellipses
https://shopify.github.io/react-native-skia/docs/shapes/ellipses

Vertices
https://shopify.github.io/react-native-skia/docs/shapes/vertices

Patch
https://shopify.github.io/react-native-skia/docs/shapes/patch

Pictures

Renders previously recorded list of drawing operations. Pictures are immutable and cannot be changed in any way. Pictures may be serialized to a byte array.
https://shopify.github.io/react-native-skia/docs/shapes/pictures

Images

Images are loaded using the useImage hook. Images can be loaded using a network URL or the *require* statement.
https://shopify.github.io/react-native-skia/docs/images

Animated Images

- Supported formats are GIF and animated WebP.
- Reanimated library can be used with the provided *useAnimatedImageValue* hook.
- To load the image using a manual API, Skia provides the *useAnimatedImage* hook instead.

SVG Images

- *ImageSVG* and *useSVG* from Skia to import SVG and use it.
- *Skia.SVG.MakeFromString* may also be used if you want to use an inlined string.
- Scaling the SVG also possible.
- ImageSVG does not follow the same painting rules as other components, as the SVG module is used from Skia instead. Effects may be applied using the layer property.

```
        <Group
        transform={fitbox("contain", src, dst)}
        layer={<Paint><ColorMatrix matrix={OpacityMatrix(0.5)} /></Paint>}>
        <ImageSVG svg={tiger} x={0} y={0} width={800} height={800} />
```

- CSS styles are not supported with SVG in Skia.
- RGBA color syntax not supported. Use fill-opacity and stroke-opacity instead.
- Inlined SVGs not supported.
- Xlink:href not supported. Avoid href.

Overall, SVG module in Skia is less forgiving.

Snapshot Views

https://shopify.github.io/react-native-skia/docs/snapshotviews