BabylonJS /
BabylonReactNative

Code | 43 Issues | 75 Pulls | Discussions | Actions

BabylonReactNative / Modules / @babylonjs / react-native /

CedricGuillemet up bjs version (#594) last month

Name	Name	Last commit date
..		
shared	Update BN submodule (#589)	2 months ago
.gitattributes	Move @babylonjs/react-nati...	3 years ago
.gitignore	Move @babylonjs/react-nati...	3 years ago
.npmignore	Move @babylonjs/react-nati...	3 years ago
BabylonModule.ts	Refactor of frame buffer cod...	2 years ago
EngineHook.ts	Update Babylon Native for n...	10 months ago
EngineView.tsx	Android ZOrder for transpar...	last year
FontFace.ts	Update to latest BabylonNat...	2 years ago
NativeCapture.ts	Update to Babylon 5.0.0-alp...	2 years ago
NativeEngineHook.ts	fix(NativeEngineHook): useA...	last year
NativeEngineView.tsx	Android ZOrder for transpar...	last year
README.md	Add 0.70 to @babylonjs/rea...	7 months ago
ReactNativeEngine.ts	up bjs version (#594)	last month
VersionValidation.ts	Validate Babylon.js version (...	3 years ago
index.ts	Implement FontFace polyfill ...	2 years ago
jest.config.js	Move @babylonjs/react-nati...	3 years ago

Name	Name	Last commit date
package-lock.json	up bjs version (#594)	last month
package.json	up bjs version (#594)	last month
tsconfig.json	Various tooling and packagi...	2 years ago

README.md

Babylon React Native

Usage

This quick overview will help you understand the constructs provided by Babylon React Native and how to use them in a React Native application.

Dependencies

This package has several **peer dependencies**. If these dependencies are unmet, `npm install` will emit warnings. Be sure to add these dependencies to your project.

The `react-native-permissions` dependency is required for XR capabilities of Babylon.js (to request camera permissions automatically). Be sure to follow the `react-native-permissions` instructions to update your `Podfile` and `Info.plist` (iOS) and/or `AndroidManifest.xml` (Android).

Android Configuration

The minimum Android SDK version is 18. This must be set as `minSdkVersion` in the consuming project's `build.gradle` file.

iOS Configuration

The minimum deployment target version is 12. This must be set as `iOS Deployment Target` in the consuming project's `project.pbxproj`, and must also be set as `platform` in the consuming project's `podfile`.

Platform Native Packages

Babylon React Native platform native packages must also be installed for the platforms and React Native versions being targeted. This is only needed for ***apps*** using Babylon React Native, not for ***libraries (React Native packages)*** building on top of Babylon React Native.

	React Native 0.63 - 0.64	React Native 0.65 - 0.66	React Native 0.69	React Na
Android	@babylonjs/react-native-iosandroid-0-64	@babylonjs/react-native-iosandroid-0-65	@babylonjs/react-native-iosandroid-0-69	@babylo native-io: 0-70
iOS	@babylonjs/react-native-iosandroid-0-64	@babylonjs/react-native-iosandroid-0-65	@babylonjs/react-native-iosandroid-0-69	@babylo native-io: 0-70
Windows	@babylonjs/react-native-windows-0-64	@babylonjs/react-native-windows-0-65	@babylonjs/react-native-windows-0-69	@babylo native-wi 0-70

`useEngine`

`useEngine` is a **custom React hook** that manages the lifecycle of a Babylon engine instance in the context of an owning React component. `useEngine` creates an engine instance **asynchronously** which is used to create and configure scenes. Typically scene initialization code should exist in a `useEffect` triggered by an `engine` state change. For example:

```
import { useEngine } from '@babylonjs/react-native';
import { Engine, Scene } from '@babylonjs/core';

const MyComponent: FunctionComponent<MyComponentProps> = (props: MyComponentPr
    const engine = useEngine();

    useEffect(() => {
        if (engine) {
            const scene = new Scene(engine);
            // Setup the scene!
        }
    }, [engine]);

    return (
        <>
        </>
    );
}
```

EngineView

`EngineView` is a **custom React Native view** that presents a `camera` from a Babylon `scene`. A `camera` therefore is assigned to the `EngineView`. For example:

```
import { useEngine, EngineView } from '@babylonjs/react-native';
import { Engine, Scene, Camera } from '@babylonjs/core';

const MyComponent: FunctionComponent<MyComponentProps> = (props: MyComponentPr
    const engine = useEngine();
    const [camera, setCamera] = useState<Camera>();

    useEffect(() => {
        if (engine) {
            const scene = new Scene(engine);
            scene.createDefaultCamera(true);
            setCamera(scene.activeCamera!);
            // Setup the scene!
        }
    }, [engine]);

    return (
        <>
            <EngineView style={{flex: 1}} camera={camera} />
        </>
    );
}
```

Also the `EngineView` has a boolean `isTransparent` flag which defines whether the background of the scene should be transparent or not.

e.g.

```
<EngineView style={{flex: 1}} camera={camera} isTransparent={true} />
```

To configure anti-aliasing, a property called `antiAliasing` can be changed to a value of 0 or 1 (disable anti-aliasing, default), 2, 4, 8 or 16 (anti-aliasing samples).

e.g.

```
<EngineView style={{flex: 1}} camera={camera} MSAA={4} />
```

Note: Currently only one `EngineView` can be active at any given time. Multi-view will be supported in a future release.

The Android specific `androidView` property can help set the type of the view used for rendering. Depending on user needs and performance, refer to the table below. `TextureView` can be inserted anywhere in the view hierarchy, but is less efficient. `SurfaceView` can only be full above or fully below the rest of the UI, but is more efficient.

isTransparent	androidView	Description
False	TextureView	Opaque TextureView.
False	SurfaceView	Simple surfaceView (default when no `androidView` set with `isTransparent=false`).
False	SurfaceViewZTopMost	SurfaceView with ZTopMost set to `true` .
False	SurfaceViewZMediaOverlay	SurfaceView with ZMediaOverlay set to `true` .
True	TextureView	Transparent TextureView.
True	SurfaceView	SurfaceView will stay opaque
True	SurfaceViewZTopMost	SurfaceView with ZTopMost set to `true` . Transparent but top most. (default when no `androidView` set with `isTransparent=true`)
True	SurfaceViewZMediaOverlay	SurfaceView with ZMediaOverlay set to `true` . Only Transparent on top of other SurfaceViews.

More infos on TextureView Vs SurfaceView performance here: https://developer.android.com/reference/android/view/TextureView