Version: 3.x

useAnimatedSensor

ⓘ INFO

This page was ported from an old version of the documentation.

As we're rewriting the documentation some of the pages might be a little outdated.

With the `useAnimatedSensor` hook, you can easily create cool interactive animations based on data from sensors in the device such as gyroscope, accelerometer etc.

```
useAnimatedSensor(sensorType: [SensorType], config?: [UserConfig]) ->
[AnimatedSensor]
```

Arguments

`sensorType` **- [SensorType]**

You can select the sensor available in [SensorType] enum.

`config` **- [UserConfig]**

Optionally, you can pass configuration to customize the sensor behavior.

Returns

Hook `useAnimatedSensor` returns an instance of [AnimatedSensor];

Types

`AnimatedSensor: [object]`

Properties:

- `sensor` : [**SharedValue**] contains [**3DVector**] or [**RotationVector**] or `null`
  contains actual sensor measurements as a shared value
- `unregister: [function]`
  allows you to stop listening to sensor updates
- `isAvailable: [boolean]`
  the flag contains information on the availability of sensors in a device
- `config` : [**UserConfig**]
  the configuration provided by a user

`SensorType: [enum]`

`SensorType` is an enum that contains possibly supported sensors. Values:

- `ACCELEROMETER`
  measurements output as [**3DVector**]. Measured in m/s^2, excluding gravity.
- `GYROSCOPE`
  measurements output as [**3DVector**]. Measured in rad/s.
- `GRAVITY`
  measurements output as [**3DVector**]. Measured in m/s^2.
- `MAGNETIC_FIELD`
  measurements output as [**3DVector**]. Measured in μT.
- `ROTATION`
  measurements output as [**RotationVector**]. [qx, qy, qz, qw] is a normalized quaternion. [yaw, pitch, roll] are rotations measured in radians along respective axes. We follow the iOS **convention**.

`UserConfig: [object]`

Properties:

- `interval: [number | auto]` - interval in milliseconds between shared value updates. Pass `'auto'` to select interval based on device frame rate. Default: `'auto'` .
- `iosReferenceFrame: [[IOSReferenceFrame](#iosreferenceframe-enum)]` - reference frame to use on iOS. Default: `Auto` .

- `adjustToInterfaceOrientation: [boolean]` - whether to adjust measurements to the current interface orientation. For example, in the landscape orientation axes x and y may need to be reversed when drawn on the screen. It's `true` by default.

`IOSReferenceFrame: [enum]`

`IOSReferenceFrame` is an enum describing reference frame to use on iOS. It follows Apple's documentation. Possible values:

- `XArbitraryZVertical`
- `XArbitraryCorrectedZVertical`
- `XMagneticNorthZVertical`
- `XTrueNorthZVertical`
- `Auto` - on devices without magnetometer (for example iPods) `XArbitraryZVertical`, on devices with magnetometer `XArbitraryCorrectedZVertical`

`3DVector: [object]`

Properties:

- `x: number`
- `y: number`
- `z: number`
- `interfaceOrientation: [[InterfaceOrientation](#interfaceorientation-enum)]`

`RotationVector: [object]`

Properties:

- `qw: number`
- `qx: number`
- `qy: number`
- `qz: number`
- `yaw: number`
- `pitch: number`
- `roll: number`

- `interfaceOrientation: [[InterfaceOrientation](#interfaceorientation-enum)]`

`InterfaceOrientation: [enum]`

Values:

- `ROTATION_0` - default rotation on Android, portrait orientation on iOS
- `ROTATION_90` - 90 degrees rotation on Android, landscape right orientation on iOS (landscape and home button on the right)
- `ROTATION_180` - 180 degrees rotation on Android, upside down orientation on iOS
- `ROTATION_270` - 270 degrees rotation on Android, landscape left orientation on iOS (landscape and home button on the left)

Example

```
function UseAnimatedSensorExample() {
  const animatedSensor = useAnimatedSensor(SensorType.ROTATION, {
    interval: 10,
  }); // <- initialization
  const style = useAnimatedStyle(() => {
    const yaw = Math.abs(animatedSensor.sensor.value.yaw);
    const pitch = Math.abs(animatedSensor.sensor.value.pitch);
    return {
      height: withTiming(yaw * 200 + 20, { duration: 100 }), // <- usage
      width: withTiming(pitch * 200 + 20, { duration: 100 }), // <- usage
    };
  });

  return (
    <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
      <Animated.View style={[{ backgroundColor: 'black' }, style]} />
    </View>
  );
}
```

Live example

Tips

CAUTION

On iOS, if you want to read sensor data you need to enable location services on your device (`Settings > Privacy > Location Services`).

Edit this page