Version: 2.6.0 – 2.12.0

RotationGestureHandler

DANGER

Consider using the new gestures API instead. The old API is not actively supported and is not receiving the new features. Check out RNGH 2.0 section in Introduction for more information.

A continuous gesture handler that can recognize a rotation gesture and track its movement.

The handler activates when fingers are placed on the screen and change position in a proper way.

Gesture callback can be used for continuous tracking of the rotation gesture. It provides information about the gesture such as the amount rotated, the focal point of the rotation (anchor), and its instantaneous velocity.

The handler is implemented using UIRotationGestureRecognizer on iOS and from scratch on Android.

Properties

Properties provided to `RotationGestureHandler` do not extend common set of properties from base handler class.

Event data

See set of event attributes from base handler class. Below is a list of gesture event attributes specific to `RotationGestureHandler`:

`rotation`

Amount rotated, expressed in radians, from the gesture's focal point (anchor).

`velocity`

Instantaneous velocity, expressed in point units per second, of the gesture.

`anchorX`

X coordinate, expressed in points, of the gesture's central focal point (anchor).

`anchorY`

Y coordinate, expressed in points, of the gesture's central focal point (anchor).

Example

See the scale and rotation example from Gesture Handler Example App.

```
class RotableBox extends React.Component {
  _rotate = new Animated.Value(0);
  _rotateStr = this._rotate.interpolate({
    inputRange: [-100, 100],
    outputRange: ['-100rad', '100rad'],
  });
  _lastRotate = 0;
  _onRotateGestureEvent = Animated.event(
    [{ nativeEvent: { rotation: this._rotate } }],
    { useNativeDriver: USE_NATIVE_DRIVER }
  );
  _onRotateHandlerStateChange = (event) => {
    if (event.nativeEvent.oldState === State.ACTIVE) {
      this._lastRotate += event.nativeEvent.rotation;
      this._rotate.setOffset(this._lastRotate);
      this._rotate.setValue(0);
    }
  };
  render() {
    return (
      <RotationGestureHandler
        onGestureEvent={this._onRotateGestureEvent}
        onHandlerStateChange={this._onRotateHandlerStateChange}>
        <Animated.Image
          style={[
            styles.pinchableImage,
            {
              transform: [{ perspective: 200 }, { rotate: this._rotateStr }],
            },
          ]}
        />
      </RotationGestureHandler>
      );
    }
  }
```