Four frameworks for mobile development

Flutter, Ionic, NativeScript & ReactNative

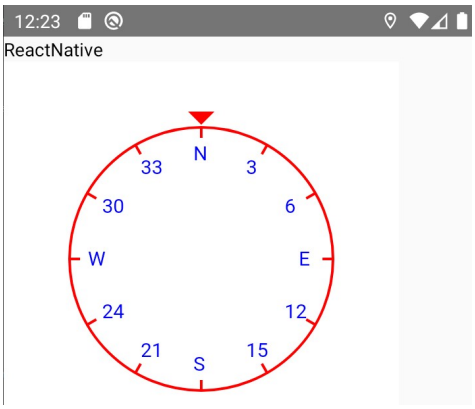
Bogusz Jelinski, 2020-01-04, v0.1

Introduction

The goal of the document is to describe steps needed to write and run the same little GPS-based functionality using different mobile frameworks. With this knowledge you could choose the framework which suites you best, which was my need too.

So I just wanted to check out what it is like to write mobile software in 2020 - which framework I would prefer because of ease of use, look&feel and technical details. Is there a "blocker"? This could very well be as prosaic as for example steep learning curve (as for now canvas in NativeScript is a blocker). The idea was to check if creating a simple compass application would be feasible. It would check both a native API (GPS here) and animation on the canvas. Only cross-platform frameworks were considered, although I think we don't need Apple any more[1], only Android was tested. So Flutter, Ionic, NativeScript and ReactNative were given a try and it turned out that you can have a working application after a few simple steps with all of them. What a great job of the particular teams!

You can find the source code both on GitHub (https://github.com/boguszjelinski/mobile_frameworks) and in appendices below.



You will need some more software than described in sections below - Android Studio with an emulator (“virtual device” in AVD Manager), Node.js with npm, Gradle and maybe Visual Studio Code. Here is a short list:

https://developer.android.com/studio
https://gradle.org/releases/
https://nodejs.org/en/download/
https://flutter.dev/docs/get-started/install
https://code.visualstudio.com/download

PATH needs to be updated to point at gradle, npm and Android SDK binaries[2].
ANDROID_SDK_ROOT, ANDROID_AVD_HOME and ANDROID_EMULATOR_HOME should be set[3]. You might want to check developers settings in your phone (USB install and debugging)[4].

1 unless they start producing a small urban EV for little money, which is unlikely.
2 On my PC I found them in *AppData\Local\Android\Sdk\platform-tools*
3 *AppData\Local\Android\Sdk*, *.android\avd* and *.android* accordingly.
4 https://developer.android.com/studio/debug/dev-options

Flutter 

Flutter and Dart (and Kotlin too) seem to be cards played by Google to exert pressure on Oracle in the court dispute about Java. But there are people who are totally in love with that proposal. No worry if you don't know Dart, it is not that different from Java or TypeScript. Although I like very much having types before variable names I would personally recommend sticking to the JavaScript family due to job opportunities.

Install

Download flutter (see an address above), unzip it and add 'bin' directory to Path. Then create a project in a chosen place on disk:

```
flutter create myapp
```

Code

compass_body.dart needs to be added in 'lib' and a few files changed – *main.dart*, *pubspec.yaml* (one line added) and *AndroidManifest.xml* (two lines added). See Appendix 1.

Run

Run Android emulator first: Tools -> AVD Manager -> Actions. Or connect your Android phone with a cable. If location is not acquired check user's consents in Settings/Apps/Permissions.

`flutter run` (it will deploy to emulator or phone depending on which is running/connected)

`flutter install` (this will install a smaller package without debug)

`adb devices` (list all available devices)

`adb -s <ID> install -f build\app\outputs\apk\app.apk` (installing a APK)

Ionic 

Install

```
npm install -g cordova ionic
ionic start MyGpsExample blank --type=angular
cd MyGpsExample
ionic cordova resources android --force
ionic cordova platform add android
ionic cordova plugin add cordova-plugin-geolocation
npm install --save @ionic-native/geolocation
```

Be sure 'gradle' is in PATH.

Code

Thre files need to be changed – *app.module.ts*, *home.page.html*l and *home.page.ts*. See Appendix 2.

Run

```
ionic cordova run android
```

```
ionic cordova run ios
```

NativeScript

There are two different canvas libraries but I have been unable to get the canvas running with any of them. It requires setting ANDROID_HOME, which was deprecated five years ago. Because you can use Angular and TypeScript with Ionic (an example here) and React too, I do not feel sorry. It is the least popular framework[5] most likely doomed to demise and oblivion if Flutter continues to gain popularity at current rate. There is no source code provided with this manual for NativeScript.

Install

```
npm install -g nativescript
```

```
tns create HelloWorld --template tns-template-hello-world
```

```
cd HelloWorld
```

```
npm install tns-android
```

```
tns platform add android
```

```
tns plugin add nativescript-canvas
```

Run

```
tns build android
```

```
tns run android --emulator
```

ReactNative

I was unable to draw a smooth movement of the compass on the canvas – something that worked smoothly with Ionic runs much slower on ReactNative, which is surprising (native should run faster than hybrid). There is no non-blocking, power efficient ‘sleep’ in JavaScript and calling setTimeout in a loop with different delay values did not make the trick with ReactNative.

```
var timer = () => {
	var h = position.coords.heading;
	setTimeout(() => {
		this.drawCompass(h);
	}, this.delay);
};
timer();
```

5 backed by a small company in stagnation in recent years

There are at least three CLI packages for React, I am presenting two. Expo does not seem to generate APK packages locally and it takes about 20 minutes to get one, you can download it then from https://expo.io/dashboard/. Advantage is you get a signed package.

expo

Install

```
npm install -g expo-cli
expo init HelloWorld          (choose 'blank', TypeScript available)
cd HelloWorld
expo install expo-location
npm install react-native-canvas –save
npm install react-native-webview –save
```

Code

One file has to be added – *MyLocation.js*. *App.js* has to be modified to refer to MyLocation component and *app.json* needs your package name. See Appendix 3.

Run

```
npm start                 (press 'a' for emulator)
npm run android
expo start                (on separate cmd window before 'expo ba')
expo build:android or expo ba
expo build:ios
expo build:status
```

react-native-cli

Install

```
npm install -g react-native-cli
react-native init MyGeoExample
cd MyGeoExample
npm install @react-native-community/geolocation –save
react-native link @react-native-community/geolocation
npm install react-native-canvas –save
npm install react-native-webview –save
```

Code

MyLocation.js is to be added and *App.js* and *AndroidManifest.xml* modified. See Appendix 4.

Run

Depending on what is connected the commands below deploy the app to a device.

`react-native run-android`

`react-native run-android –variant=release` (it builds APK)

Appendex 1 – flutter

pubspec.yaml

```
dependencies:
 flutter:
  sdk: flutter
 geolocator: '^5.1.1'
```

AndroidManifest.xml

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
```

main.dart

```
import 'package:flutter/material.dart';
import 'package:geolocator/geolocator.dart';
import 'dart:async';
import 'compass_body.dart';

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
 // This widget is the root of your application.
 @override
 Widget build(BuildContext context) {
  return MaterialApp(
   title: 'Flutter Demo',
   theme: ThemeData(
    primarySwatch: Colors.blue,
   ),
   home: MyHomePage(title: 'Flutter Demo Home Page'),
  );
 }
}

class MyHomePage extends StatefulWidget {
 MyHomePage({Key key, this.title}) : super(key: key);

 final String title;

 @override
 _MyHomePageState createState() => _MyHomePageState();
}
```

```
class _MyHomePageState extends State<MyHomePage> {

 Geolocator geolocator = Geolocator();
 double heading=0, prevHeading=0;
 StreamSubscription<Position> positionStream ;

 @override
 Future initState()  {
  super.initState();
  geolocator.forceAndroidLocationManager = true;
  geolocator.checkGeolocationPermissionStatus();

  positionStream = geolocator.getPositionStream(LocationOptions(
     accuracy: LocationAccuracy.best, timeInterval: 1000))
     .listen((position) {
      if (position == null || position.heading==null) return;
      animate(position.heading, prevHeading);
  });
 }

 void animate(double head, double prevHead) async {
  double diff = 1.0;
  if (abs(head - prevHead)>1.0) {
   if (head - prevHead < 0) diff = -1.0;
   if (abs(head - prevHead)>180) diff *= -1.0; // eg. 20 -> 340
   for (double h = prevHead + diff; abs(h - head) > 1.1; h+= diff) {
    if (h>360) h = 0;
    else if (h<0) h = 360;
    setState(() {
     heading = h;
    });
    await Future.delayed(new Duration(milliseconds: 15));
   }
  }
  setState(() {
   prevHeading = head;
   heading = head;
  });
 }

 abs(double d) {
  return d<0.0 ? -1*d : d;
 }

 void checkPermission() {
  geolocator.checkGeolocationPermissionStatus().then((status) { print('status: $status'); });
  geolocator.checkGeolocationPermissionStatus(locationPermission: GeolocationPermission.locationAlways).the
n((status) { print('always status: $status'); });
  geolocator.checkGeolocationPermissionStatus(locationPermission: GeolocationPermission.locationWhenInUse).
.then((status) { print('whenInUse status: $status'); });
 }

 @override
 Widget build(BuildContext context) {
```

```
   return Scaffold(
    appBar: AppBar(
     title: Text(widget.title),
    ),
    body: Center(
     child: Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: <Widget>[
       new CompassBody(heading)
      ],
     ),
    ), // This trailing comma makes auto-formatting nicer for build methods.
   );
  }
}
```

compass_body.dart

```
import 'package:flutter/material.dart';
import 'dart:math';

class CompassBody extends StatefulWidget {
 final double heading;

 CompassBody(this.heading);

 @override
 _StatefulWidgetExampleState createState() => _StatefulWidgetExampleState();
}

class _StatefulWidgetExampleState extends State<CompassBody> {

 @override
 Widget build(BuildContext context) {
  return Column(
     mainAxisAlignment: MainAxisAlignment.center,
     children: <Widget>[
      new CustomPaint(
       painter: new CompassPainter(widget.heading)
      ),
     ],
   );
 }
}

class CompassPainter extends CustomPainter{
 double heading;

 CompassPainter(double loc){
    this.heading = loc;
 }

 @override
 void paint(Canvas canvas, Size size) {
  drawCompass(canvas, heading);
 }
```

```
 void drawCompass (Canvas canvas, double hd) {
  final double width = 1.0;
  final double sze = 0;
  final double radius = 100;

  List<String> labels = ['N','3','6','E','12','15','S','21','24','W','30','33'];

  Paint line = new Paint()
    ..color = hd == 0.0 ? Colors.red : Colors.black
    ..strokeCap = StrokeCap.round
    ..style = PaintingStyle.stroke
    ..strokeWidth = width;

  Offset center  = new Offset(sze, sze);
  canvas.drawCircle(center, radius, line);
  drawTriangle(sze, sze, radius, canvas);

  double i =0;
  for (String lab in labels) {
     this.putText(canvas, sze, sze, i*30, hd, radius, lab, line);
     i++;
  }
  //canvas.restore();
 }

 void drawTriangle (double cx, double cy, double radius, Canvas canvas) {
  Paint red = new Paint()
    ..color = Colors.red
    ..strokeCap = StrokeCap.round
    ..style = PaintingStyle.stroke
    ..strokeWidth = 2;
  Path path = new Path();
  path.moveTo(cx, cy-radius-2);
  path.lineTo(cx-5, cy-radius-5-2);
  path.lineTo(cx+5, cy-radius-5-2);
  path.lineTo(cx, cy-radius-2);
  canvas.drawPath(path, red);
 }

 putText (canvas, double cx, double cy, double degree, double hdng, double radius, label, color) {
  TextSpan span = new TextSpan(text: label, style: new TextStyle(color: Colors.blue[800]));
  TextPainter tp = new TextPainter(text: span, textAlign: TextAlign.left, textDirection: TextDirection.ltr)
;
  tp.layout();

  Path path = new Path();
  double angle = degree - hdng;
  double sinus = sin(toRadians(angle));
  double cosinus = cos(toRadians(angle));
  double x = cx + (radius-18)*sinus-6;
  double y = cy - (radius-18)*cosinus-10;
  double x1 = cx + (radius-8)*sinus;
  double y1 = cy - (radius-8)*cosinus;
  double x2 = cx + (radius)*sinus;
```

```
    double y2 = cy - (radius)*cosinus;
    path.moveTo(x1,y1);
    path.lineTo(x2, y2);
    tp.paint(canvas, new Offset(x, y));
    canvas.drawPath(path, color);
  }

  toRadians (angle)  {
    return angle * (pi / 180);
  }

  @override
  bool shouldRepaint(CustomPainter oldDelegate) {
    return true;
  }
}
```

Appendix 2 – ionic

app.module.ts

```
import { Geolocation } from '@ionic-native/geolocation/ngx';
       providers: [
              ...
                Geolocation,
              ...
```

home.page.html

```
<ion-header>
  <ion-title>
    Compass
  </ion-title>
</ion-header>
<ion-content padding>
    <p>Ionic</p>
    <div class="ion-canvas">
      <canvas #myCanvas></canvas>
  </div>
</ion-content>
```

home.page.ts

```
import { Component, ViewChild, ElementRef  } from '@angular/core';
import { Geolocation ,GeolocationOptions } from '@ionic-native/geolocation/ngx';
import { Platform } from '@ionic/angular';

@Component({
  selector: 'app-home',
  templateUrl: 'home.page.html',
  styleUrls: ['home.page.scss'],
})

export class HomePage {

  @ViewChild('myCanvas') canvas : ElementRef;
  canvasElement: any;
```

```
private _CANVAS  : any;
private ctx : any;

circleRadius:number = 100;
cx:number = 150;
cy:number = 150;
delay:number = 20;

heading: number = undefined;
prevHeading : number = undefined;
watch: any;
error: any;

constructor(public platform: Platform, private geolocation: Geolocation) {
}

ngAfterViewInit(){
 let options : GeolocationOptions = {
  enableHighAccuracy: true
 };
 this.platform.ready().then(() => {
  this.watch = this.geolocation.watchPosition(options);
  this.watch.subscribe((resp) => {
   if (resp.coords && resp.coords.heading != 0) {
    this.prevHeading = this.heading;
    this.heading = resp.coords.heading;
    let wasAnimated : boolean;
    wasAnimated = this.animate();
   }
  }, error => this.error = error );
 });
 this._CANVAS      = this.canvas.nativeElement;
 this._CANVAS.width    = 300;
 this._CANVAS.height   = 300;
 this.initialiseCanvas();
 this.drawCompass(0);
}

animate() : boolean {
 if (this.heading && this.prevHeading) {
  let diff = 2.0;
  if (this.prevHeading && Math.abs(this.heading - this.prevHeading) > 1.1) {
   if (this.heading - this.prevHeading < 0) {
    diff = -2.0;
   }
   if (Math.abs(this.heading - this.prevHeading) > 180) {
    diff *= -1.0; // eg. 20 -> 340
   }
   let h = this.prevHeading + diff;
   let i = 1;
   while (Math.abs(h - this.heading) > 2.1) {
    if (h > 360) {
     h = 0;
    } else if (h < 0) {
```

```
        h = 360;
       }
       var timer = () => {
        var hh = h;
        var ii = i;
        setTimeout(() => {
         this.drawCompass(hh);
        }, ii * this.delay);
       };
       timer();
       h += diff;
       i++;
      }
      return true;
     }
    }
    return false;
}

initialiseCanvas() {
    if(this._CANVAS.getContext) {
     this.ctx = this._CANVAS.getContext('2d');
    }
}

clearCanvas() {
    this.ctx.fillStyle = '#FFFFFF';
    this.ctx.fillRect(0, 0, this._CANVAS.width, this._CANVAS.height);
}

private drawCompass (heading: number) : void {
    let labels = ['N','3','6','E','12','15','S','21','24','W','30','33'];
    this.clearCanvas();
    this.triangle();
    this.ctx.font = '15px Arial';
    this.ctx.fillStyle = 'blue'
    this.ctx.strokeStyle = 'black';
    this.ctx.lineWidth = 2;
    this.ctx.beginPath();
    this.ctx.arc(this.cx, this.cy, this.circleRadius, 0, 2*Math.PI);
    this.ctx.stroke();
    for (let i=0; i<12; i++)
       this.putText(i*30, heading, this.circleRadius-20, labels[i]); // 30: step of degree
    this.ctx.stroke();
}

toRadians = (angle) => { return angle * (Math.PI / 180); }

private triangle () {
    this.ctx.fillStyle = 'red'
    this.ctx.strokeStyle = 'red';
    this.ctx.beginPath();
    this.ctx.moveTo(this.cx, this.cy-this.circleRadius-2);
    this.ctx.lineTo(this.cx-10, this.cy-this.circleRadius-12);
```

```
    this.ctx.lineTo(this.cx+10, this.cy-this.circleRadius-12);
    this.ctx.lineTo(this.cx, this.cy-this.circleRadius-2);
    this.ctx.fill();
  }
  private putText (degree:number, heading: number, radius: number, label: string):void {
    let angle:number = degree - heading;
    let sinus:number = Math.sin(this.toRadians(angle));
    let cosinus: number = Math.cos(this.toRadians(angle));
    let x = this.cx + radius*sinus-6;
    let y = this.cy - radius*cosinus+5;
    this.ctx.fillText(label,x,y);
    let x1 = this.cx + (this.circleRadius-8)*sinus;
    let y1 = this.cy - (this.circleRadius-8)*cosinus;
    let x2 = this.cx + this.circleRadius*sinus;
    let y2 = this.cy - this.circleRadius*cosinus;
    this.ctx.moveTo(x1,y1);
    this.ctx.lineTo(x2, y2);
  }
}
```

Appendix 3 – react native (expo)

App.js

```
import MyLocation from './MyLocation';

...

return (
   <View>
     <MyLocation />
   </View>
 );
```

app.json

```
 "android": {
    "package": "com.my.company"
   }
```

MyLocation.js

```
import React, {Component} from 'react';
import Geolocation from '@react-native-community/geolocation';
import Canvas from 'react-native-canvas';
import {View} from 'react-native';

class MyLocation extends Component {
 ctx = null;
 radius = 100;
 delay = 100;
 cx = 150;
 cy = 150;
 labels = ['N', '3', '6', 'E', '12', '15', 'S', '21', '24', 'W', '30', '33'];

 state = {
  prevHead: undefined,
  currHead: undefined,
```

```
  error: '',
};

componentDidMount = () => {
  Geolocation.setRNConfiguration({
    enableHighAccuracy: true,
  });

  this.watchID = Geolocation.watchPosition(
    position => {
      if (position && position.coords) {
        this.drawCompass(position.coords.heading);
      }
    },
    error => {},
    {timeout: 1000, enableHighAccuracy: true, distanceFilter: 1},
  );
};

componentWillUnmount = () => {
  Geolocation.clearWatch(this.watchID);
};

render() {
  return (
    <View>
      <Canvas ref={this.handleCanvas} />
    </View>
  );
}

handleCanvas = canvas => {
  if (canvas) {
    this.ctx = canvas.getContext('2d');
    canvas.width = 300;
    canvas.height = 300;
    this.drawCompass(0);
  }
};

clearCanvas() {
  this.ctx.fillStyle = '#FFFFFF';
  this.ctx.fillRect(0, 0, 300, 600);
}

drawCompass(heading) {
  if (!this.ctx) {
    return;
  }
  this.clearCanvas();
  this.triangle();
  this.ctx.fillStyle = 'blue';
  if (heading === 0) {
    this.ctx.strokeStyle = 'red';
```

```
    } else {
      this.ctx.strokeStyle = 'black';
    }
    this.ctx.lineWidth = 2;
    // circle
    this.ctx.beginPath();
    this.ctx.arc(this.cx, this.cy, this.radius, 0, 2 * Math.PI);
    this.ctx.stroke();
    // labels with short lines depicting directions
    for (let i = 0; i < 12; i++) {
      this.putText(i * 30, heading, this.radius, this.labels[i]);
    } // 30: step of degree
    this.ctx.stroke();
  }

  toRadians(angle) {
    return angle * (Math.PI / 180);
  }

  triangle() {
    this.ctx.fillStyle = 'red';
    this.ctx.strokeStyle = 'red';
    this.ctx.beginPath();
    this.ctx.moveTo(this.cx, this.cy - this.radius - 2);
    this.ctx.lineTo(this.cx - 10, this.cy - this.radius - 12);
    this.ctx.lineTo(this.cx + 10, this.cy - this.radius - 12);
    this.ctx.lineTo(this.cx, this.cy - this.radius - 2);
    this.ctx.fill();
  }

  putText(degree, heading, r, label) {
    let angle = degree - heading;
    let sinus = Math.sin(this.toRadians(angle));
    let cosinus = Math.cos(this.toRadians(angle));
    let x = this.cx + (r - 20) * sinus - 6;
    let y = this.cy - (r - 20) * cosinus + 5;
    this.ctx.font = '15px Arial';
    this.ctx.fillText(label, x, y);
    let x1 = this.cx + (r - 8) * sinus;
    let y1 = this.cy - (r - 8) * cosinus;
    let x2 = this.cx + r * sinus;
    let y2 = this.cy - r * cosinus;
    this.ctx.moveTo(x1, y1);
    this.ctx.lineTo(x2, y2);
  }
}

export default MyLocation;
```

Appendix 4 – react native

App.js

```
import MyLocation from './MyLocation';
```

```
...
return (
  <View>
    <MyLocation />
  </View>
 );
```

AndroidManifest.xml

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
```

MyLocation.js

```
import React, {Component} from 'react';
import Geolocation from '@react-native-community/geolocation';
import Canvas from 'react-native-canvas';
import {View} from 'react-native';

class MyLocation extends Component {
 _CONTEXT = null;
 radius = 100;
 delay = 100;
 cx = 150;
 cy = 150;
 state = {
  prevHead: undefined,
  currHead: undefined,
  error: '',
 };

 componentDidMount = () => {
  Geolocation.setRNConfiguration({
   enableHighAccuracy: true,
  });

  this.watchID = Geolocation.watchPosition(
   position => {
    if (position && position.coords) {
     this.drawCompass(position.coords.heading);
    }
   },
   error => {},
   {timeout: 1000, enableHighAccuracy: true, distanceFilter: 1},
  );
 };

 componentWillUnmount = () => {
  Geolocation.clearWatch(this.watchID);
 };

 render() {
  return (
   <View>
    <Canvas ref={this.handleCanvas} />
   </View>
```

```
  );
}

handleCanvas = canvas => {
  if (canvas) {
    this._CONTEXT = canvas.getContext('2d');
    canvas.width = 300;
    canvas.height = 300;
    this.drawCompass(0);
  }
};

clearCanvas() {
  this._CONTEXT.fillStyle = 'rgba(0, 0, 0, 0)';
  this._CONTEXT.clearRect(0, 0, 300, 600);
  this._CONTEXT.stroke();
  this._CONTEXT.fillStyle = '#FFFFFF';
  this._CONTEXT.fillRect(0, 0, 300, 600);
}

drawCompass(heading) {
  let labels = [
    'N',
    '3',
    '6',
    'E',
    '12',
    '15',
    'S',
    '21',
    '24',
    'W',
    '30',
    '33',
  ];

  if (!this._CONTEXT) {
    return;
  }
  this.clearCanvas();
  // trangle
  this._CONTEXT.fillStyle = 'red';
  this._CONTEXT.strokeStyle = 'red';
  this._CONTEXT.beginPath();
  this._CONTEXT.moveTo(this.cx, this.cy - this.radius - 2);
  this._CONTEXT.lineTo(this.cx - 10, this.cy - this.radius - 12);
  this._CONTEXT.lineTo(this.cx + 10, this.cy - this.radius - 12);
  this._CONTEXT.lineTo(this.cx, this.cy - this.radius - 2);
  this._CONTEXT.fill();
  this._CONTEXT.fillStyle = 'blue';
  if (heading === 0) {
    this._CONTEXT.strokeStyle = 'red';
  } else {
    this._CONTEXT.strokeStyle = 'black';
```

```
    }
    this._CONTEXT.lineWidth = 2;
    this._CONTEXT.beginPath();
    this._CONTEXT.arc(this.cx, this.cy, this.radius, 0, 2 * Math.PI);
    this._CONTEXT.stroke();
    this._CONTEXT.beginPath();
    for (let i = 0; i < 12; i++) {
      this.putText(i * 30, heading, this.radius, labels[i]);
    } // 30: step of degree
    this._CONTEXT.closePath();
    this._CONTEXT.stroke();
  }

  toRadians(angle) {
    return angle * (Math.PI / 180);
  }

  putText(degree, heading, r, label) {
    let angle = degree - heading;
    let sinus = Math.sin(this.toRadians(angle));
    let cosinus = Math.cos(this.toRadians(angle));
    let x = this.cx + (r - 20) * sinus - 6;
    let y = this.cy - (r - 20) * cosinus + 5;
    this._CONTEXT.font = '15px Arial';
    this._CONTEXT.fillText(label, x, y);
    let x1 = this.cx + (r - 8) * sinus;
    let y1 = this.cy - (r - 8) * cosinus;
    let x2 = this.cx + r * sinus;
    let y2 = this.cy - r * cosinus;
    this._CONTEXT.moveTo(x1, y1);
    this._CONTEXT.lineTo(x2, y2);
  }
}

export default MyLocation;
```