**高德，百度，Google地图定位偏移以及坐标系转换**

参考：<http://blog.csdn.net/a13570320979/article/details/51366355>

一。在进行地图开发过程中，我们一般能接触到以下三种类型的地图坐标系：

1.WGS－84原始坐标系，一般用国际GPS纪录仪记录下来的经纬度，通过GPS定位拿到的原始经纬度，Google和高德地图定位的经纬度（国外）都是基于WGS－84坐标系的；但是在国内是不允许直接用WGS84坐标系标注的，必须经过加密后才能使用；

2.GCJ－02坐标系，又名“火星坐标系”，是我国国测局独创的坐标体系，由WGS－84加密而成，在国内，必须至少使用GCJ－02坐标系，或者使用在GCJ－02加密后再进行加密的坐标系，如百度坐标系。高德和Google在国内都是使用GCJ－02坐标系，可以说，GCJ－02是国内最广泛使用的坐标系；

3.百度坐标系:bd-09，百度坐标系是在GCJ－02坐标系的基础上再次加密偏移后形成的坐标系，只适用于百度地图。(目前百度API提供了从其它坐标系转换为百度坐标系的API，但却没有从百度坐标系转为其他坐标系的API)

二。为什么会发生偏移？

1.由于坐标系之间不兼容，如在百度地图上定位的经纬度拿到高德地图上直接描点就肯定会发生偏移；只考虑国内的情况，高德地图和Google地图是可以不经过转换也能够准确显示的（在国内用的都是GCJ－02坐标系）；下面是收录了网上的WGS－84，GCJ－02，百度坐标系(bd-09)之间的相互转换的方法，经测试，是转换后相对准确可用的：

**[java]** [view plain](http://blog.csdn.net/a13570320979/article/details/51366355) [copy](http://blog.csdn.net/a13570320979/article/details/51366355)

1. **package** com.asiabasehk.cgg.util;
3. /\*\*火星坐标系 (GCJ-02) 与百度坐标系 (BD-09) 的互转
4. \* Created by macremote on 16/5/3.
5. \*/
6. **public** **class** GPSUtil {
7. **public** **static** **double** pi = 3.1415926535897932384626;
8. **public** **static** **double** x\_pi = 3.14159265358979324 \* 3000.0 / 180.0;
9. **public** **static** **double** a = 6378245.0;
10. **public** **static** **double** ee = 0.00669342162296594323;
12. **public** **static** **double** transformLat(**double** x, **double** y) {
13. **double** ret = -100.0 + 2.0 \* x + 3.0 \* y + 0.2 \* y \* y + 0.1 \* x \* y
14. + 0.2 \* Math.sqrt(Math.abs(x));
15. ret += (20.0 \* Math.sin(6.0 \* x \* pi) + 20.0 \* Math.sin(2.0 \* x \* pi)) \* 2.0 / 3.0;
16. ret += (20.0 \* Math.sin(y \* pi) + 40.0 \* Math.sin(y / 3.0 \* pi)) \* 2.0 / 3.0;
17. ret += (160.0 \* Math.sin(y / 12.0 \* pi) + 320 \* Math.sin(y \* pi / 30.0)) \* 2.0 / 3.0;
18. **return** ret;
19. }
21. **public** **static** **double** transformLon(**double** x, **double** y) {
22. **double** ret = 300.0 + x + 2.0 \* y + 0.1 \* x \* x + 0.1 \* x \* y + 0.1
23. \* Math.sqrt(Math.abs(x));
24. ret += (20.0 \* Math.sin(6.0 \* x \* pi) + 20.0 \* Math.sin(2.0 \* x \* pi)) \* 2.0 / 3.0;
25. ret += (20.0 \* Math.sin(x \* pi) + 40.0 \* Math.sin(x / 3.0 \* pi)) \* 2.0 / 3.0;
26. ret += (150.0 \* Math.sin(x / 12.0 \* pi) + 300.0 \* Math.sin(x / 30.0
27. \* pi)) \* 2.0 / 3.0;
28. **return** ret;
29. }
30. **public** **static** **double**[] transform(**double** lat, **double** lon) {
31. **if** (outOfChina(lat, lon)) {
32. **return** **new** **double**[]{lat,lon};
33. }
34. **double** dLat = transformLat(lon - 105.0, lat - 35.0);
35. **double** dLon = transformLon(lon - 105.0, lat - 35.0);
36. **double** radLat = lat / 180.0 \* pi;
37. **double** magic = Math.sin(radLat);
38. magic = 1 - ee \* magic \* magic;
39. **double** sqrtMagic = Math.sqrt(magic);
40. dLat = (dLat \* 180.0) / ((a \* (1 - ee)) / (magic \* sqrtMagic) \* pi);
41. dLon = (dLon \* 180.0) / (a / sqrtMagic \* Math.cos(radLat) \* pi);
42. **double** mgLat = lat + dLat;
43. **double** mgLon = lon + dLon;
44. **return** **new** **double**[]{mgLat,mgLon};
45. }
46. **public** **static** **boolean** outOfChina(**double** lat, **double** lon) {
47. **if** (lon < 72.004 || lon > 137.8347)
48. **return** **true**;
49. **if** (lat < 0.8293 || lat > 55.8271)
50. **return** **true**;
51. **return** **false**;
52. }
53. /\*\*
54. \* 84 to 火星坐标系 (GCJ-02) World Geodetic System ==> Mars Geodetic System
55. \*
56. \* @param lat
57. \* @param lon
58. \* @return
59. \*/
60. **public** **static** **double**[] gps84\_To\_Gcj02(**double** lat, **double** lon) {
61. **if** (outOfChina(lat, lon)) {
62. **return** **new** **double**[]{lat,lon};
63. }
64. **double** dLat = transformLat(lon - 105.0, lat - 35.0);
65. **double** dLon = transformLon(lon - 105.0, lat - 35.0);
66. **double** radLat = lat / 180.0 \* pi;
67. **double** magic = Math.sin(radLat);
68. magic = 1 - ee \* magic \* magic;
69. **double** sqrtMagic = Math.sqrt(magic);
70. dLat = (dLat \* 180.0) / ((a \* (1 - ee)) / (magic \* sqrtMagic) \* pi);
71. dLon = (dLon \* 180.0) / (a / sqrtMagic \* Math.cos(radLat) \* pi);
72. **double** mgLat = lat + dLat;
73. **double** mgLon = lon + dLon;
74. **return** **new** **double**[]{mgLat, mgLon};
75. }
77. /\*\*
78. \* \* 火星坐标系 (GCJ-02) to 84 \* \* @param lon \* @param lat \* @return
79. \* \*/
80. **public** **static** **double**[] gcj02\_To\_Gps84(**double** lat, **double** lon) {
81. **double**[] gps = transform(lat, lon);
82. **double** lontitude = lon \* 2 - gps[1];
83. **double** latitude = lat \* 2 - gps[0];
84. **return** **new** **double**[]{latitude, lontitude};
85. }
86. /\*\*
87. \* 火星坐标系 (GCJ-02) 与百度坐标系 (BD-09) 的转换算法 将 GCJ-02 坐标转换成 BD-09 坐标
88. \*
89. \* @param lat
90. \* @param lon
91. \*/
92. **public** **static** **double**[] gcj02\_To\_Bd09(**double** lat, **double** lon) {
93. **double** x = lon, y = lat;
94. **double** z = Math.sqrt(x \* x + y \* y) + 0.00002 \* Math.sin(y \* x\_pi);
95. **double** theta = Math.atan2(y, x) + 0.000003 \* Math.cos(x \* x\_pi);
96. **double** tempLon = z \* Math.cos(theta) + 0.0065;
97. **double** tempLat = z \* Math.sin(theta) + 0.006;
98. **double**[] gps = {tempLat,tempLon};
99. **return** gps;
100. }
102. /\*\*
103. \* \* 火星坐标系 (GCJ-02) 与百度坐标系 (BD-09) 的转换算法 \* \* 将 BD-09 坐标转换成GCJ-02 坐标 \* \* @param
104. \* bd\_lat \* @param bd\_lon \* @return
105. \*/
106. **public** **static** **double**[] bd09\_To\_Gcj02(**double** lat, **double** lon) {
107. **double** x = lon - 0.0065, y = lat - 0.006;
108. **double** z = Math.sqrt(x \* x + y \* y) - 0.00002 \* Math.sin(y \* x\_pi);
109. **double** theta = Math.atan2(y, x) - 0.000003 \* Math.cos(x \* x\_pi);
110. **double** tempLon = z \* Math.cos(theta);
111. **double** tempLat = z \* Math.sin(theta);
112. **double**[] gps = {tempLat,tempLon};
113. **return** gps;
114. }
116. /\*\*将gps84转为bd09
117. \* @param lat
118. \* @param lon
119. \* @return
120. \*/
121. **public** **static** **double**[] gps84\_To\_bd09(**double** lat,**double** lon){
122. **double**[] gcj02 = gps84\_To\_Gcj02(lat,lon);
123. **double**[] bd09 = gcj02\_To\_Bd09(gcj02[0],gcj02[1]);
124. **return** bd09;
125. }
126. **public** **static** **double**[] bd09\_To\_gps84(**double** lat,**double** lon){
127. **double**[] gcj02 = bd09\_To\_Gcj02(lat, lon);
128. **double**[] gps84 = gcj02\_To\_Gps84(gcj02[0], gcj02[1]);
129. //保留小数点后六位
130. gps84[0] = retain6(gps84[0]);
131. gps84[1] = retain6(gps84[1]);
132. **return** gps84;
133. }
135. /\*\*保留小数点后六位
136. \* @param num
137. \* @return
138. \*/
139. **private** **static** **double** retain6(**double** num){
140. String result = String .format("%.6f", num);
141. **return** Double.valueOf(result);
142. }
144. }

2.在国内定位的经纬度，然后在国外网络下显示也会发生偏移（谷歌和高德会依据网络的情况选择使用WGS－84坐标还是GCJ－02坐标，百度地图则一直使用bd-02坐标系）

3.定位方式，在iOS定位的经纬度是通过GPS获取的，在android则可以通过网络或GPS获取经纬度。通过地图SDK定位获取的经纬度，地图SDK会自动选择加密的方式（如Google地图会根据国内国外选择不同的坐标系）然后再将点显示在地图上，这个时候是没有偏移的；如果直接将经纬度在地图上显示，可能就会因为地域或网络的问题导致使用的坐标系不同，进而发生来偏移。