面向安卓系统的矢量室内地图绘制优化技术

刘 政，罗 红
（北京邮电大学计算机学院，北京 100876）

摘要：地理位置服务（location based services，LBS）的迅猛发展，要求在手机上快捷流畅地进行地图展示。而 Android 手机系统，受限于硬件条件及自身架构体系，效率问题一直备受诟病。对此，研究从软件优化的角度提出了一种优化矢量室内地图绘制的解决方案。提出了三个方面的优化思路，并针对 Android 手机进行相关实验验证了优化设计思路的可行性与正确性。

关键词：计算机应用；Android；矢量地图；性能优化；双缓冲；预加载
中图分类号：TP311.1 **文献标识码：**A **文章编号：**1674-2850(2016)04-0411-06

Optimized render technology of vectorial indoor map on Android system

LIU Zheng, LUO Hong
(*School of Computer Science*, *Beijing University of Posts and Telecommunications*, *Beijing* 100876, *China*)

Abstract: The rapid development of location based service (LBS) requires quick and fluent map display on mobile phones. However, the efficiency of the Android mobile phone system has been much criticized because of its hardware and structure system. To solve this problem, this paper proposes a solution which can optimize indoor vector map rendering from the perspective of software optimization. This paper will present prioritization scheme in three aspects. And we have done related experiments on Android mobile phones to verify the feasibility and correctness of these prioritization schemes.

Keywords: computer application; Android; vectorial map; performance optimization; double buffering; pre-loading

0 引言

随着移动互联网的兴起及智能硬件的飞速发展，尤其是苹果推出 iPhone、Google 推出 Android 后，围绕移动互联网的业务创新、模式创新和全球性战略竞争与布局已全面展开。智能硬件与移动互联网的融合，催生了 LBS 的兴起。LBS 是通过移动终端和移动网络的配合，确定移动用户的实际地理位置，从而提供用户所需与位置相关服务信息的一种移动通信与导航融合的服务形式。LBS 直接面向用户提供个性化服务，大大方便了人们的生活，增强了人与人之间的沟通，具有极大的商业价值和社会价值。各大企业看到该行业的发展前景，纷纷开始提供 LBS. 目前，随着大型建筑物内部结构的日益复杂和人们室内活动的不断增加，室内地图的需求越来越大，成为指导人们日常出行的重要工具[1]，作为 LBS 重要载体之一的地图在未来移动互联网中仍将发挥着基石作用。地图服务需要与用户进行频繁的交互，因此对系统的性能要求比较高。然而移动设备受限于硬件条件，很难在硬件条件上对性能做出更多的提升。所

基金项目：高等学校博士学科点专项科研基金（20110005110007）
作者简介：刘政（1990—），男，硕士研究生，主要研究方向：物联网技术
通信联系人：罗红，教授，主要研究方向：现代通信网络技术及物联网、通信软件. E-mail: luoh@bupt.edu.cn

以，研究如何以软件方法来优化移动端的地图服务具有重大的现实意义。

1 矢量室内地图数据格式

目前在移动端的电子地图呈现技术主要有瓦片拼接的位图技术以及矢量图形技术两种。然而移动手机端信号的不稳定性及流量和手机内存的限制使得矢量地图越来越受到人们的欢迎。矢量图形是在平面上表现的图形，一般采用二维坐标系，通过 x 和 y 坐标来描绘图形的端点位置。矢量地图采用单纯的点和线及对应的渲染数据就能灵活地描述图像，数据量非常小，而且不用拼接技术而是直接绘制在画布上，可以使计算的复杂度降低，极大地满足了受限移动端的绘制需求。

常用的矢量图形标准有可伸缩矢量图形（scalable vector graphics，SVG）、SWF（shock wave flash）、制图交换格式（drawing exchange format，DXF）等。目前有很多的矢量图形格式，但是，有些格式的设计本身有缺陷，而有些是某家公司私有标准，扩展和兼容性比较差，这些都不太合适手机端直接使用[2]。正因为如此，许多应用往往会采用自定义的矢量数据格式，这样不但可以以最小的代价满足自己的需求，而且还方便自身的扩展与优化。

2 Android 二维图像绘制技术

Android 系统提供非常强大的二维图形库，主要涉及的类库包括 paint、canvas、matrix、bitmap 等。paint 类用于保存画笔的颜色、样式等绘制属性信息；canvas 画布类作为图像绘制的载体；matrix 类用于实现图形的变换操作，如平移、缩放和旋转等；bitmap 类用于实现对位图的基本操作。

利用 Android 原生的图形库可以绘制几何图形、文本、图像和 path 等元素。其中的 path 代表任意多条直线连接而成的任意图形，当 canvas 根据 path 绘制图形时，它可以绘制出任意由关键点组成的图形。除此之外，通过 matrix 作用于 path 对象可以方便地实现对任意图形的平移、旋转和缩放操作。

3 地图绘制优化方案

本文研究对象针对的是一套结合室内地图和 Android 系统特点而自定义的矢量图形格式，该格式对应的每一个显示元素主要由 3 个方面的元素组成：商铺基本信息、渲染数据和其他附带数据。这里研究的是矢量地图绘制优化问题，地图绘制性能的关键部分便是地图多边形轮廓的绘制。因此，这里主要针对元素关键的轮廓数据部分进行研究。每个显示元素主要由各顶点表示的多边形组成，例如 $\{x_1, y_1; x_2, y_2; ...; x_n, y_n\}$ 表示一个由 n 个顶点组成的多边形[3]。Android 端将这样的一个多边形保存为一个 path 对象，如此便可以通过 paint 画笔对象将对应的 path 对象绘制到 canvas 中显示出来。

Android 系统通过监听用户每次对屏幕的触摸事件来对 path 对象对应的坐标做相应的加减或缩放操作之后再将 path 对象绘制到 canvas 中，从而实现用户对显示元素（地图）的拖动和缩放等交互操作。然而过多的重绘消耗过多的系统资源，影响了系统的运行效率，从而导致用户使用时出现卡顿等现象，带来不良的用户体验。本文就是在此基础上对相关实现进行优化，从而实现系统性能上的提升。

系统运行效率取决于用户操作过程中地图更新操作的时间，这个时间主要包括 2 个方面：重绘的频率以及图形绘制过程中的耗时。因为每次屏幕显示的数据内容大小一定，不可能为了提高系统效率而隐藏某些需要显示的部分。因此，只能通过减小系统重绘的频率来提升系统效率。本优化方案主要通过双缓冲、预加载和选择性重绘的技术来减少不必要的重绘，从而提升系统的效率。

3.1 bitmap 和 canvas 双缓冲

双缓冲技术是游戏开发中图形绘制的一个重要技术。主要原理是：当多个元素争先显示时，程序又

在改变它，前面的画面还没显示完，程序又要求重新绘制，这样屏幕就会不停闪烁。为避免闪烁，可以使用双缓冲技术，将要处理的图片都放在内存中处理好之后，再将其显示到屏幕上。这样一来每次显示出来的就是完整图像，不会出现闪烁现象[4]。

Android 中的 SurfaceView 类其实就是一个双缓冲机制[5]，适用于游戏这类交互、更新比较频繁的场景，然而对于基础的地理数据显示，主要是用户的手势操作之后地图做出被动的响应操作，完全可以直接用一般的 view 类来进行处理。然而当显示的数据过多时，每次重绘时 onDraw 都需要对所有的绘制元素进行绘制，这样会阻塞主线程从而使界面重绘时出现卡顿现象。在此情况下采用缓冲技术异步地将矢量地图数据先绘制到缓存 bitmap 对象中，绘制完成之后再将绘制好的 bitmap 在 view 类的 onDraw 回调函数中直接绘制到 canvas 中，绘制流程如图 1 所示。如此循环的缓冲过程不但提高了重绘时的效率，很大程度上解决了主线程导致的界面卡顿现象，更为后面的拖动和缩放的优化打下了良好的基础。

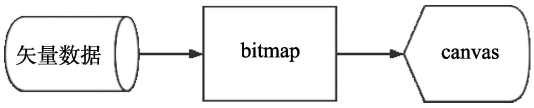

图 1 双缓冲绘制

Fig. 1 Double buffer drawing

3.2 临近部分预加载

系统用于展示给用户的数据可能是很大的一部分，但受限于硬件条件，不可能将所有的数据都加载到内存中。例如对于一个地图应用，系统提供给用户的可能是整个世界地图的数据，不可能因为用户可能查看世界地图的任何一个部分就将所有的数据都预加载到内存中供用户查看。

传统的做法是按一定的比例仅仅绘制并显示当前屏幕区域的数据，然后每次缩放或拖动时都得重新计算当前应显示的区域并重新绘制和显示，这样，每次简单的拖动或缩放操作都需要重绘所有的显示元素。简单的处理带来的结果就是导致出现卡顿现象及界面响应不够流畅，给用户带来不好的体验。

图 2 缓存预加载

Fig. 2 Cache preloading

冯诺伊曼体系结构的计算机系统中的程序执行遵循着一个“局部性”原理，即 CPU 访问存储器时，无论是存取指令还是存取数据，所访问的存储单元都趋于聚集在一个较小的连续区域中[6]。对于地图类的展示数据也是一样，用户将要看见的部分大部分都是当前所见部分的临近周边部分。基于这个原理，当通过计算当前屏幕需要显示区域 A 时，可以将 A 区域周边的一定大小区域提前用双缓冲的方式绘制到一张 bitmap 中，然后再将此 bitmap 绘制到 canvas 中，当用户拖动地图导致当前屏幕应显示区域没超过预加载部分时，只需要拖动画布而不需要重新绘制整个区域便可以将对应区域展示给用户。当用户拖动阈值使显示区域超过预加载部分时便回收之前的bitmap，并计算当前需显示哪部分的矢量数据（区域 B），同样地，预加载 B 周边的数据到新建的 bitmap 中，然后再将此 bitmap 绘制到 canvas 中，如图 2 所示。如此便可以减少大部分拖动操作过程中的重绘频率，从而提高系统的响应时间，增加系统的流畅性。

3.3 选择性重绘

所有的动画效果其实都是利用人眼在观察事物时的“视觉暂留”现象，通过在一定的时间内变换画面场景来达到人眼所看到的动画效果。更换画面的频率越高，眼睛看到的画面也就越流畅。然而频率太高会导致性能问题，尤其对于资源有限的移动设备。因此，可以在“视觉暂留”效应的时间内尽可能地减少不必要的重绘来提高绘制效率。如上文提到的可以在加载的时候预先加载多余的临近部分，这样就

不会在每次拖动的过程中都需要重绘，只有当拖动的距离超过预加载的部分之后才会发生重绘。

除了通过预加载的方式来减少拖动过程中的重绘次数外，还可以在缩放的过程中减少重绘的次数。这同样也是在双缓冲的基础上来实现的。地图缩放的过程可以两步处理。在缩放的过程中（用户手指尚未抬起）只需对之前已经绘制好的 bitmap 进行缩放然后将此 bitmap 绘制到 canvas 中，在缩放结束后（手指抬起）在对相应区域的坐标进行缩放计算之后，再将整个预加载区域绘制到 bitmap 中，之后再将 bitmap 绘制到 canvas 中，如图 3 所示。在缩放的过程中因为是对 bitmap 进行缩放，这样就会出现位图缩放的弊端，即当缩放到一定程度时显示区域会变得模糊。不过由于在缩放结束后会对 bitmap 进行重绘，最后呈现给用户的地图仍然是清晰的。为进一步解决 bitmap 缩放模糊的情况还可以在缩放的过程中对缩放的比例进行阈值控制，在缩放的过程中当缩放的倍数超过一定比例后就进行重绘，这样便可以通过调节阈值来控制不重绘的情况下 bitmap 最大的模糊度。

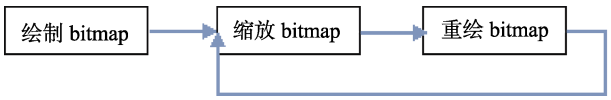

图 3 选择性重绘

Fig. 3 Selective redraw

4 核心算法

电子地图对应点的坐标有 2 个：地图坐标和屏幕坐标。地图坐标是之前测量好点的绝对位置（不变项）。屏幕坐标是每次对地图进行偏移、旋转和缩放后点对应显示在屏幕上的位置坐标（可变项）。以上分析中对重绘的优化是在拖动未超过预加载部分以及缩放过程中仅仅只是对上一次绘制好的bitmap整体进行相应的偏移和缩放，这个过程不用考虑坐标映射的问题。然而当拖动超过预加载的部分或是缩放手势抬起时就需要将对应的地图坐标映射到拖动或缩放后的位置再进行重绘。算法过程描述如下。

算法 1：平移操作算法

1）通过用户的手势操作获得水平和垂直方向的偏移量并计算偏移量是否超过预加载部分，若超过预加载部分则执行 2），否则执行 4）。

2）回收上一次重绘的 bitmap 资源，新建 bitmap. 根据偏移量生成偏移转换矩阵 matrixOffset.

3）将偏移矩阵 matrixOffset 作用于所有的地图坐标 path 对象生成新的屏幕坐标 path 对象之后将其绘制到新建的 bitmap 中（这一过程中需要绘制其临近部分），重绘后将偏移量置 0.

4）根据手势操作的偏移量对 canvas 进行偏移后再将 bitmap 绘制到 canvas 中。

算法 2：缩放操作算法

1）用户按下手指准备开始缩放操作时，记录下起始时两手指间的距离 disStart 及其中心 midPoint（mid$x$，mid$y$），之后缩放的过程中记录手指间的距离 disCurrent 并计算缩放比例 scale=disCurrent/disStart，若手指未抬起则执行 4），否则此次缩放结束，转到 2）执行。

2）回收上一次重绘的 bitmap 资源，新建 bitmap. 根据缩放中心 midPoint 及缩放比例 scale 生成缩放转换矩阵 matrixScale.

3）将缩放转换 matrixScale 作用于所有的地图坐标 path 对象生成新的屏幕坐标 path 对象之后将其绘制到新建的 bitmap 中（这一过程中需要绘制其临近部分），重绘后将缩放比例 scale 置为 1.

4）根据缩放中心 midPoint 及缩放比例 scale 对 bitmap 进行缩放之后再将其绘制到 canvas 中。

5 测试结果

为验证上述方案的可行性与正确性，针对上述设计，完成了对旧版本 SDK 的优化升级。然后在基于 Android 4.4 版本的小米 3 手机上进行了相关测试实验。测试结果如下：

图 4 为新版 SDK 运行界面，加载、拖动、旋转及缩放操作流畅，无明显卡顿现象，达到了预期设想的效果。

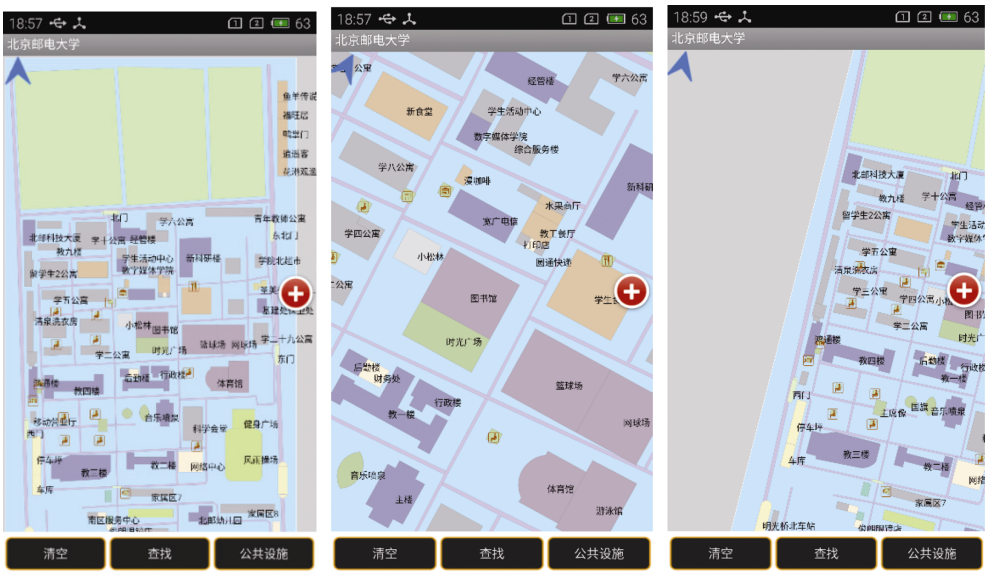

图 4 测试效果界面

Fig. 4 Interface of test effect

除此之外，还对相同的数据在新、旧 SDK 版本分别进行了拖动和缩放性能测试，测试图包含 13 829 个地图元素，地图实际大小约为 40 万 m^2. 用谷歌的性能测试工具“GPU 呈现模式分析（profile GPU rendering）”[7]得到的测试数据结果如图 5 所示。

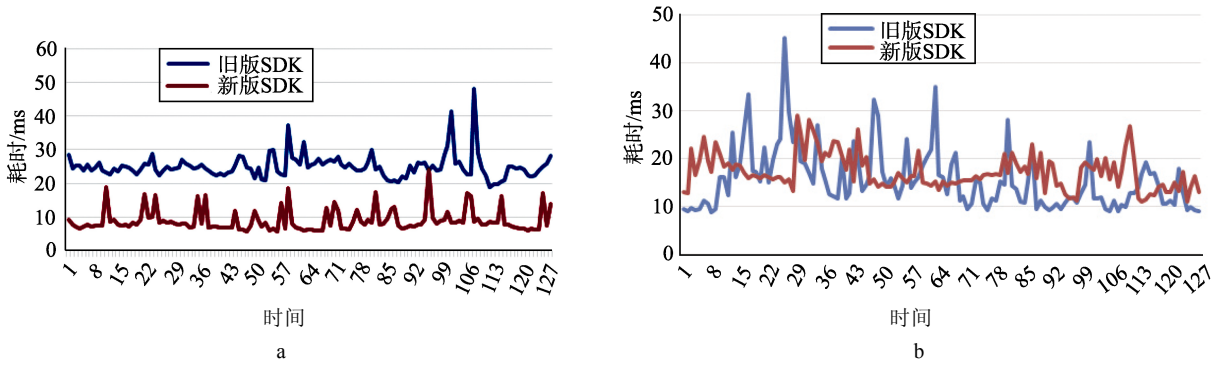

图 5 新、旧版本 SDK 性能对比

Fig. 5 Comparion of original and new SDK characters

a—拖拽操作；b—缩放操作

a-Drag-and-drop operation; b-Scale operation

图 5 数据显示，新版 SDK 在拖拽操作中性能提高了 50%左右，缩放操作性能也提高了约 30%，以上测试数据显示证明该性能优化方案切实可行并达到了良好的优化效果。

6 结论

移动互联网的兴起对移动设备提出了更高的要求。更复杂的界面交互、更复杂的逻辑流程要求更流畅的用户体验。在有限的硬件资源条件下，通过良好的软件设计思路、实现方式来提高系统的运行效率

变得越来越重要。这里给出了 Android 手机矢量图形绘制的优化方案，并基于此方案，通过实验验证了优化思路的可行性并达到了预期的效果。

[参考文献] (References)

[1] 齐晓飞，崔秀飞，李怀树. 室内地图设计现状分析[J]. 测绘与空间地理信息，2013，36（2）：10-14.
QI X F, CUI X F, LI H S. Analysis of design on indoor map[J]. Geomatics & Spatial Information Technology, 2013, 36(2): 10-14. (in Chinese)

[2] 李春卉. 矢量图形标准与规范研究现状[J]. 情报科学，2007，25（4）：636-640.
LI C H. Research on the standards of vector graphic[J]. Information Science, 2007, 25(4): 636-640. (in Chinese)

[3] 吴晓泉. Android 室内矢量地图 SDK 的设计与实现[D]. 北京：北京邮电大学，2014.
WU X Q. The design and realization of the vector map of SDK Android indoor[D]. Beijing: Beijing University of Posts and Telecommunications, 2014. (in Chinese)

[4] 刘婷婷，潘彤，陈彬茹. 利用双缓存技术提高绘图速率[J]. 电脑编程技巧与维护，2012（18）：22-24.
LIU T T, PAN T, CHEN B R. Improving the efficiency of computer drawing with the help of double cache technology[J]. Computer Programming Skills & Maintenance, 2012(18): 22-24. (in Chinese)

[5] 曹海涛. 基于并发控制和双缓存技术的移动 gis 切片地图展现方法：中国，CN103473732A[P]. 2013.
CAO H T. Mobile gis slice method based on the map to show the concurrency control and dual caching techniques: China, CN103473732A[P]. 2013. (in Chinese)

[6] 陈军民. 循序渐近原则在“冯·诺依曼体系结构”教学中的渗透[J]. 魅力中国，2014（13）：185.
CHEN J M. Principle of gradual penetration in the ‘von Neumann architecture’ teaching[J]. Charm China, 2014(13): 185. (in Chinese)

[7] GUY R. Android performance case study[OL]. [2015-12-30]. http://www.curious-creature.com/docs/android-performance-case-study-1.html.