World Wind 瓦片地图插件设计与实现①

杨 龙, 李 家

(辽宁师范大学 城市与环境学院, 大连 116029)

摘 要: 根据 WorldWind 的插件扩展机制, 研究了在 WorldWind 上显示网络瓦片地图的一些关键知识和技术, 如网络瓦片地图使用的坐标投影、插件加载、下载瓦片的过程和本地文件缓存以及内存缓存等, 最后实现了网络瓦片地图的插件, 并提出优化瓦片地图的一些方法.

关键词: World Wind; 网络瓦片地图; 插件; 多线程队列; 缓存

Design and Implementation of Tile Map Plugin Based on World Wind

YANG Long, LI Jia

(City & Environment Institute, Liaoning Normal University, Dalian 116029, China)

Abstract: According to extension mechanism of the World Wind plugin, this paper studies some of knowledge and technology about displaying tile map, such as the tile map coordinates projection, the plug-in loaded, the download process of the tiles and the local file cache and memory cache etc., and finally achieves the the tile map plugin, and puts forward a number of methods to optimize the tile map

Key words: World Wind; tile map; plugin; multi-thread queue; cache

1 前言

1.1 瓦片地图技术

随着谷歌 GoogleMaps 的发布, Esri、微软、百度等公司纷纷推出自己的网络瓦片地图[1]以及以网络地图为数据源的其他应用. 网络瓦片地图技术已经成为一种主流的在互联网上发布地图的技术, 出现于各行各业的应用中. 网络瓦片地图是将地图和影像采用特定的预切割方式进行切片, 并将切好的地图图像存储在服务器上, 当用户访问地图时, 将从地图服务器上请求图片到本机缓存, 这样就很大程度上减轻服务器压力, 提高了请求数量和访问速度.

1.2 World Wind 介绍

World Wind[2]是 NASA 发布的一个开放源代码的三维虚拟地球显示项目, 以 NASA 网站提供的数据为数据源, 提供了插件机制以扩展其应用. 它是一个优秀的客户端框架引擎, XML 实现数据描述和软件设置, 通过 WorldModel、图层、插件、HTTP 和 WMS 请求、三维渲染等实现交互式浏览. World Wind 具有良好的可扩展性.

1.3 插件设计目标

插件是一种遵循一定规范的应用程序接口编写出来的辅助程序. 使用插件技术能够在分析、设计、开发、产品扩展等很多方面带来诸多好处. World Wind 对自身的功能扩展都是通过插件技术来实现的. 本文通过插件技术使 World Wind 支持主流的瓦片数据, 例如 Google、Esri、Bing 等, 使其成为兼容多种瓦片数据源的流畅、易用的 World Wind 插件.

2 瓦片地图技术要点

2.1 投影及坐标系统

网络瓦片地图所使用的地图投影, 常被称作 Web Mercator 或 Spherical Mercator[3], 它与常规墨卡托投影的主要区别就是把地球模拟为球体而非椭球体. 赤道作为标准纬线, 本初子午线作为中央经线, 两者交点

① 收稿时间:2012-09-13;收到修改稿时间:2012-10-26

为坐标原点，向东向北为正，向西向南为负. 由于赤道半径为 6378137 米，赤道周长为 2*PI*R = 2*20037508.3427892，因此 X 轴的取值范围为：[-20037508.3427892, 20037508.3427892]. 由墨卡托投影可知当纬度接近 90°时，y 值趋向于无穷. 为了计算方便，把 Y 轴的取值范围也限定在[-20037508.3427892, 20037508.3427892]之间，取成正方形. 对应的地理坐标，经度取全球范围: [-180, 180]，依据投影可知纬度不可能到达90°，经过反计算，可得到纬度85.0511288. 因此纬度取值范围是[-85.0511288, 85.0511288].

2.2 瓦片与坐标

瓦片地图是在服务器端根据瓦片地图金字塔模型[4]进行切片的. 瓦片金字塔是一种多分辨率层次模型. 在地形场景绘制时，在保证显示精度的前提下为提高显示速度，不同区域通常需要不同分辨率的数字高程模型数据和纹理影像数据. 地图瓦片是栅格图像，并不具备坐标信息. 但是切片运用了相关切片算法之后，可以计算出具体定位的位置. 例如采用 WGS84 大地坐标系为空间参考，用经纬度步长等比例分割形成的地图瓦片，当需要时可以根据瓦片的行列号反推出起地理坐标以及投影坐标，反之也能根据地理经纬度坐标推算出其在一定层级中的行列号.

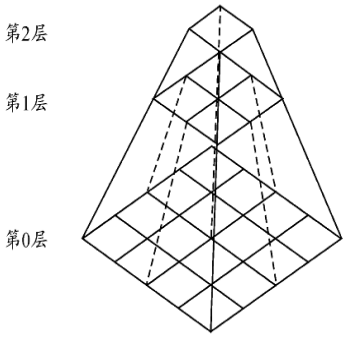

图 1 瓦片金字塔示意图

3 插件设计过程

3.1 插件的加载过程分析

WorldWind 提供了完善的插件扩展机制. 本文研究了 World Wind 的启动过程. 首先在 Main 函数中调用 OpenStartupWorld()，用来初始化启动 World 对象. 加载完 World 对象后，通过 InitializePluginCompiler()函数加载各个插件. 此函数会根据插件形态，dll 或者.cs 源代码生成 Plugin 类. 然后调用该类的虚函数 Load(). 因此，在插件中只要实现抽象基类 Plugin 类的虚函数即可. Plugin 类提供了四个虚函数，只需实现 Load(), Unload()这两个.

3.2 插件模块的架构

通过上面的分析，在插件模块中定义类 KOMPlugin，继承于 Plugin，实现 Load(), Unload()方法. 在 Load()中启动自定义窗体，窗体设计如下:

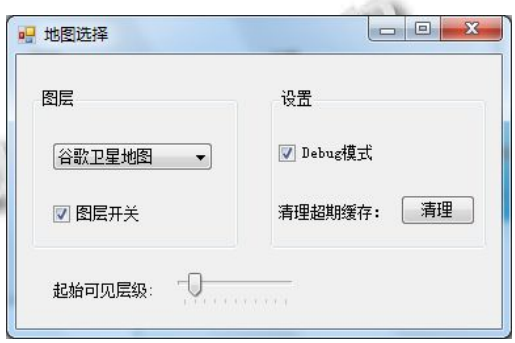

图 2 插件界面设计

当在组合框中切换地图种类时，主窗体加载对应的地图. 其实质是 MapProvider 根据行列号提供不同的瓦片 URL. 一个瓦片数据对应一个 URL 地址，因此需要构造不同的 URL. 以 Google 中国地图[5]为例，某瓦片对应的 URL 如下:

http://mt1.google.cn/vt/lyrs=s@88&gl=cn&x=1&y=2&z=3&s=gali

其中 mtX 中的 X 为数字，代表某个服务器，“x=1, y=2, z=3”的意思为行号为 1、列号为 2、层级为 3 的瓦片，lyrs=s 中的 s 为地图的种类. 根据这些规则可设计以下功能类来产生 URL.

设计 MapProvider 类，它是一个抽象类，只有接口 imageUrl()，用来根据行列号产生一个对应的 URL. 派生的子类主要是为其他属性赋值以及实现该方法. 例如 Google 地图为 GoogleMapProvider，继承于 MapProvider，其 imgUrl()实现代码大致为:

format="http://{0}{1}{2}/{3}/lyrs={4}&hl={5}&gl=cn&x={6}{7}&y={8}&z={9}&s={10}"

url= string.format(format, …x, y, z…); //x, y, z 为行列号和层级

3.3 设计瓦片多线程下载队列

由于互联网络环境的复杂性，从网络下载瓦片数据相对于渲染是一个缓慢的过程. 当一个下载请求发

送出去后，大部分时间在等待请求的返回. 如果在工作线程中等待数据的返回，势必影响工作线程的执行，实际表现为浏览过程卡顿现象特别严重. 因此下载数据的任务应该单独使用其他线程来完成，需设计下载队列来优化这一过程.

设计瓦片下载队列应注意以下几点. 首先是下载瓦片的范围. 根据视野的范围确定哪些瓦片需要下载渲染. 第二点是下载任务项的优化. 当快速切换瓦片层级时，应该剔除上一层级的瓦片. 例如任务队列下载 level=4 的瓦片，而新加入的瓦片 level=5 了，此时应该放弃所有 level=4 的任务. 第三，下载失败，任务项应该重新投回队列. 第四，一些值的估测. 例如任务队列的最大容量(属于优化部分)， 下载线程的个数等. 这些值可多次测试得到满意经验值.

具体设计方案需要三个类:TaskItem， TaskQueue 和 DownloadQueue 来协同完成.

TaskItem，即任务项，它表示一项具体的瓦片下载任务. TaskItem 数据结构设计如下:

```
public class TaskItem
{
    public string url;   //瓦片地址
    public int level;    //瓦片层级
    public KOMTile tile;   //保存数据
    public string savedFilePath;//本地缓存路径
    publicDownloadComplete completeCallback;
}
```

TaskQueue 是一个队列类，主要实现线程安全的队列操作. 主要数据成员： List< TaskItem > queue 用来模拟队列， Object syncObject 用来做同步 . 主要有两个操作 addTaskItem() 和 getTaskTtem(). addTaskItem()由工作线程调用，工作线程会随着视野位置移动不断产生 TaskItem，并由 addTaskItem()加入 TaskQueue 中. 这两个函数不同线程调用，因此需要线程同步:

```
lock (syncObject) [6]   //加锁
{
 Queue.Add(item); //加入队列
 if (queue.Count == 1)
  {
    Monitor.PulseAll(syncObject); //启动下载线程
  }
}
```

getTaskTtem()由下载线程调用:

```
lock (syncObject)
{
    if (queue.Count > 1 && queue [0].level != queue [Queue.Count - 1] .level)
      {
        … //优化，图层切换发生，移除过时任务
      }
   else if(queue.Count > 0)
      {
        item = Queue[0]; //从队列首取出一项
        queue.RemoveAt(0);
      }
   if(queue.Count >= TaskQueue.MaxQueueLength)
      {
        //优化，移除快速浏览时一些过时任务
        Queue.RemoveRange(0,  20);
      }
}
```

DownloadQueue 里有一个 TaskQueue 和线程函数，它负责启动各个下载线程，轮询 TaskQueue 中是否有下载任务项. 提供的主要接口为 addTaskItem()，它实际是调用了 TaskQueue 的 addTaskItem(). 线程函数利用 .NetFremwork 提供的 HttpWebRequest，HttpWebResponse 等类来实现具体的发送 Http 请求，并调用回调函数处理下载的瓦片数据.

3.4 本地瓦片缓存机制

为了提高瓦片的加载速度以及重复利用率，需要建立瓦片的缓存机制. 瓦片缓存分为两类:

(1) 本地文件缓存. 瓦片数据不可能每次需要时都去从网络下载，所以应该把下载到的瓦片保存到本地. 有两种方案，一是保存到数据库中，根据行列号和层级唯一确定一张瓦片数据;二是保存到本地文件系统中. 方案一是一种可扩展的方案. 例如一个局域网中多个 World Wind 客户端，用一台数据库服务器作为缓存服务器，可以提高数据的利用率. 方案二适用于一般情况，简单灵活，目录层次依次可以为:本地建缓存目录、地图种类目录、层级目录、行目录、行的列瓦片文件.

(2) 内存中缓存瓦片数据. 因为加载瓦片数据是

一个相对耗时的操作，所以在内存缓存瓦片数据可以提高渲染速度．因为内存大小限制，所以不能在内存中保留所有图片．那只能保留最急切需要的部分瓦片．假设这样一种情况:现在已经显示了 50 张瓦片，鼠标拖动地球使地图移动，可能有 10 张新瓦片需要从网络下载或者本地瓦片文件加载，并渲染，而另外 40 张是之前的数据，只需要从新渲染即可，不需要加载．因此，可以缓存上一次的瓦片数据，以供下一次使用．利用一个 Dictionaty 数据结构缓存本次显示的瓦片，下次显示前根据行列号和层级生成的键值来提取需要的数据，同时适当时候把这次新数据加入字典．字典的键可根据行列号生成，在插入一个新缓存时根据剔除离插入位置远的键值对．

4 结论及优化

4.1 结论

借助 WorldWind 的插件机制，本文实现了网络瓦片的加载显示，并能在不同种类的瓦片地图之间切换，如图:

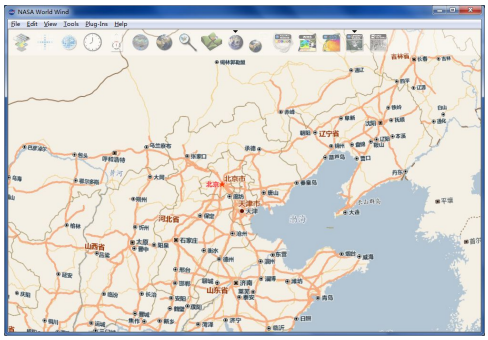

图 4 Bing 卫星地图数据的显示

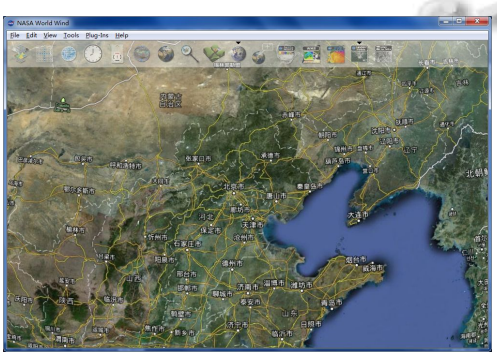

图 5 Google 地图数据的显示

4.2 可优化的方向

瓦片地图的显示涉及到大量瓦片的加载和渲染，其响应速度涉及到整个系统的流畅程度，因此最后提出一些可优化的方向，来提高插件的用户体验．

(1) 适当扩大瓦片覆盖的范围．可以适当扩大瓦片的申请范围，例如下载 1.5 倍屏幕的瓦片，这样小范围移动地图时，只要不超出已经扩大的范围，就不用下载或者加载新的瓦片．

(2) 使用优先级队列，屏幕中心优先下载．为了达到更好的用户体验，当切换地图层级时，应该考虑优先从屏幕的中心加载瓦片，然后向四周扩散．可以使用优先级队列，将屏幕中心的瓦片设置成最高的优先级，然后离中心越远，优先级递减，这样中心的瓦片可以优先得到处理．

参考文献

1 http://wiki.osgeo.org/wiki/Tile_Map_Service_Specification.

2 http://goworldwind.org.

3 寇曼曼,王勤忠.Google Map 数字栅格地图算法及应用.计算机技术与发展,2012,(4):204–206.

4 崔金红,王旭.Google 地图算法研究及实现.计算机科学,2007,(11):193–195.

5 刘升容,刘学锋.全国第二次土地调查中海量遥感影像瓦片金字塔的建立与无缝组织.测绘通报,2011,(7):37–39.

6 Threading(C#Programming Guide),Microsoft MSDN Library.