Interactive Browsing System of 3D Lunar Model with Texture and Labels on Mobile Device

Yankui Sun(✉), Kan Zhang, and Ye Feng

Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
syk@mail.tsinghua.edu.cn

Abstract. This paper devotes to developing an interactive visualization system of 3D lunar model with texture and labels on mobile device. Using OpenGL ES 2.0 and Osg for android, we implement 3D lunar mesh model construction, lunar texture and mapping, GPU shader illumination programming, multi-level terrain labels and interactive browsing. In particular, a technique of terrain labels for mobile device is presented by developing a vertex shader and fragment shader which is dedicated to render texture, where the vertex shader is mainly used to determine the vertex's position attribute of the texture while the fragment shader is responsible for the color, font and transparency of the text, etc. The developed browsing system can be rendered on mobile device in real-time.

Keywords: Mobile device · Lunar model · Lunar texture · Terrain labels · OpenGL ES 2.0 · Osg for android · Chang'E-1

1 Introduction

China launched her own unmanned, lunar-orbiting spacecraft, CE-1 (Chang'E-1) in October 2007. CE-1 got 1.37 TB original research data which contain about 8.6 million LAM elevation values and 1,073 tracks of 2C-level CCD image data. In interactive visualization of lunar model, a series of research works have been done based on PC and/or internet [1–4]. In [1, 2], Sun et al. used the LAM data to construct an interactive lunar model using bicubic subdivision-surface wavelet and transmitted it through network using a client-server mode. In [3], Dong et al. constructed a lunar surface model, using the LAM data, the CCD data and hundreds of labels, and visualized it on personal computer with the converging problem in the polar regions being solved. Sun et al. further proposed and implemented an internet-based interactive visualization method of 3D lunar model with texture and labels [4]. In this paper, we devoted to developing the interactive browsing system of 3D lunar model with texture and labels on mobile device. It should be pointed out that, with the fast development of computing ability of mobile devices, rendering complex scene efficiently on portable device becomes an important research topic. Many researches in this aspect have been done based on client/service architecture [5, 6], where part of rendering work is done on the mobile devices. Recently, the research on doing rendering work completely on mobile devices has been done. [7] presented a technique which can manage the level-of-detail of 3D meshes in portable devices. Here we will develop techniques to run 3D lunar model interactively on mobile device using OpenGL ES 2.0 and OSG for android. This is not straightforward to do, and there are some technical difficulties to overcome, including: (1) The implementation of the terrain label could not be done directly on mobile device because OpenGL ES 2.0 has different rendering pipeline from OpenGL; (2) The texture size loaded into the memory once time is limited,where the maximum texture size is 4096 × 4096 for OpenGL ES 2.0; (3) GPU programming of programmable pipeline shader for mobile device is more difficult. The above difficulties will be overcome here. The main contributions of this paper include: (1) Proposing a transparent literal texture mapping technique to implement multi-level terrain labels for 3D lunar model on mobile device. (2) Developing a real-time browsing system of 3D lunar model with texture and labels on mobile device by realizing 3D lunar mesh model construction, texture mapping, illumination calculation, human-machine interaction.

The rest of this paper is organized as follows. The Sect. 2 gives the systematic flow chart, and introduces the key implementation techniques. Section 3 describes the system development and implementation. Conclusions are made in Sect. 4.

2 Algorithm Description

Our systematic flow chart is shown in Fig. 1. It consists of some models including lunar mesh modeling, texture mapping, illumination calculation, terrain labels and human-machine interactive.

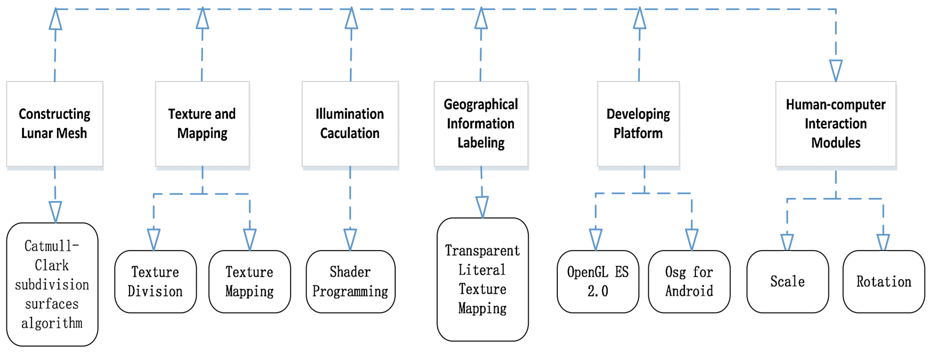


Fig. 1. Flow chart of our system

2.1 Lunar Mesh Modeling

A semi-regular lunar mesh with 50 thousand quadrilaterals was constructed by denoising, Catmull-Clark subdivision and resampling from the 8,610,511 data points obtained by CE-1, as was done in [3]. Its low resolution lunar 3D model with 24576 triangles and 12290 vertices is also obtained so that they can be browsed in multi-resolution. To render them on mobile device, the quadrangles of the lunar models are divided into triangles, and vertex array mapping method provided by OpenGL ES 2.0 is used to render the triangle mesh of the lunar models. More specifically, all the vertex information and the rendering information of the lunar triangle mesh are transmitted to GPU at once, and they are accessed and rendered sequentially according to vertex array.

2.2 Texture Division and Mapping

In [8], a lunar map with size of 18024 × 9012 has been effectively completed from CE-1 CCD image data. The texture size is about 120 M, which is impossible to load it into the memory of a mobile device once time since the texture size loaded into the memory once time is limited, where the maximum texture size is 4096 × 4096 for OpenGL ES 2.0. To solve this problem, a texture division method is proposed to load large size texture. First, the original lunar texture is divided into four parts: South Polar texture and North Polar texture with size of 5740 × 5740; Eastern and Western Hemisphere textures with size of 9100 × 3640. Then, these textures are all divided into two patches further so that the total texture size 122 M can be loaded into the memory in small patch sizes.

For the lunar mesh models mentioned above, each vertex's texture coordinate is calculated by using the texture mapping method used in [3], and they are stored in vertex array by allocating storage space for them. Texture rendering is realized in OpenGL ES 2.0 as follows. The function InputStream is used to decode a texture bitmap, and the texture is bound to a serial number by the function glBindTexture in OpenGL ES 2.0. Then, specifying the filling mode as texture filling when texture rendering is done in shader.

2.3 Illumination Calculation

Lighting is very important for enhancing the rendering quality of a lunar model. In [3], two point light sources were positioned in front and back of the lunar model. Lighting calculation is not complex in programming on PC because OpenGL support specifying materials and reflection characteristics of the object, location and intensity of the light source directly in the application, so illumination calculation can be done automatically. Yet the things become very different for OpenGL ES 2.0 because it renders pixel one by one. In this case, each vertex's rendering mode is specified by setting light source, normal, material attributes, and then illumination for each vertex is computed by illumination model. In particular, a normal parameter is computed and stored in a vertex array, and then it is transferred to GPU to be used in illumination calculation by shader code.

2.4 Terrain Labels

Terrain labels can be implemented on PC using the text information output function DrawListText() and Chinese byte stream reading function fread() in OpenGL. Unfortunately, these two functions are cut down in OpenGL ES 2.0, so there are some difficulties to realize terrain labels on 3D lunar models in mobile terminals. This paper designs and implements a technique to tag terrain information on the lunar model by using OpenGL

ES 2.0, called transparent literal texture mapping technique. Here a set of specific vertex shader and fragment shader are developed to render texture, which are different from the vertex shader and fragment shader for rendering the lunar mesh. In the new design, the vertex shader is mainly used to determine the vertex position attribute of a texture while the fragment shader to realize the color, font and transparent attributes of a text. Then, terrain text information is added into a texture bitmap, and it is output by using texture mapping embedded with the text. The following steps show how to realize text information labeling in 3D space by texture mapping using OpenGL ES 2.0.

Step 1: **Merging Terrain Text Information into a Texture Bitmap.** For each geographical name, a new bitmap and a new canvas are created. Then, the canvas is set to be transparent, and the attributes of the landmark such as font, color and size, are set. After all the settings are completed, we begin to draw the landmarks.

Step 2: **Displaying the Text in Transparency on the Merged Bitmap.** Get a vertex's color by Bitmap.getPixel and Bitmap.SetPixel functions. For each vertex, the corresponding alpha channel is set 0 to assure the text can be seen. Transparent attribute is the inherent attribute of the vertices. By setting the transparency, we can set the background color to be transparent, and show the text information in the texture.

Step 3: **Labelling Geographical Names in a Right Position.** Obtain 3D position coordinate of a texture and label geographical names in the right position of the lunar model. Bind texture rendering and camera so that terrain labels can be rotated and scaled together with viewpoint. In the meanwhile, backface cutting technique is used to hide invisible landmark to speed up the rendering speed.

Because of quantity of geographical names and small display screen of a mobile device, we propose a multi-level terrain label technique. In our application, we classified the geographical labels into different levels according to the distance, denoted distance by D, between a viewpoint and the center of the lunar model. The smaller the distance, the more labels displayed.

2.5 Human-Machine Interaction

Human-computer interaction is an important part for an interactive browsing system on mobile device. For Android application, the class used to correspond to user's gesture definition is OnTouchEvent. We realized three kinds of gesture interaction functions in our browsing system: rotation, zoom in and zoom out so that we can browse the lunar model from all directions and various perspectives.

3 System Development and Experimental Results

With the aforementioned techniques, we developed a lunar interactive visualization system with texture and labels under lighting environment by using OpenGL ES 2.0 and

OSG for android. One Plus mobile phone (5.5 inch screen, 1920 × 1080 resolution, four core processor, 3 Gb of memory, 64G external memory), which runs Android 4.3 operating system, is used as a mobile device to test the performance of our system.

OpenGL ES 2.0 uses programmable pipeline instead of fixed one in rendering, where GPU running code needs be written and loaded into the video card to compile when the program runs. In programming with OpenGL ES 2.0, how to render every pixel needs to be specified by developers. Obviously the great degree of freedom in programming also means more difficulties to meet. Compared with OpenGL ES 2.0, Osg for android is an advanced 3D interactive graphics development engine, where some powerful functions are provided so that the programming for developers is simplified greatly. In our application development, Osg is mainly used to manage 3D viewpoint (camera) including setting up and modifying the parameters of the camera such as location and direction. In human-computer interaction module, camera class in Osg engine is applied to make the camera do actions according to what we need, such as rotating around an orbit and so on, which simplifies the programming greatly compared to using OpenGL ES 2.0 directly.

An integrated software development framework for mobile device is designed and implemented, which is showed in Fig. 2. A cpp file programmed by VS2010 is loaded into Osg for android and complied in local computer (PC). To make the local codes run correctly in Java virtual machine, modify CMake List file by adding header files and cpp files, and then run it in NDK command line to generate .so file (a dynamic link library in Linux kernel) in target folder. Finally, compile the Java program by Java SDK to generate APK file, a set up file which can be installed in the mobile terminal.

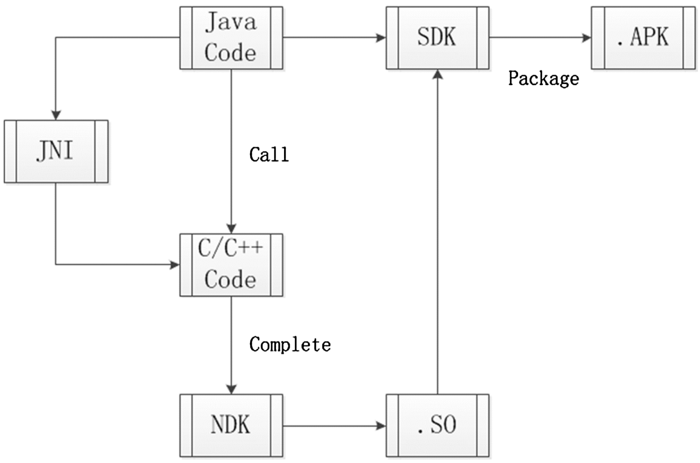
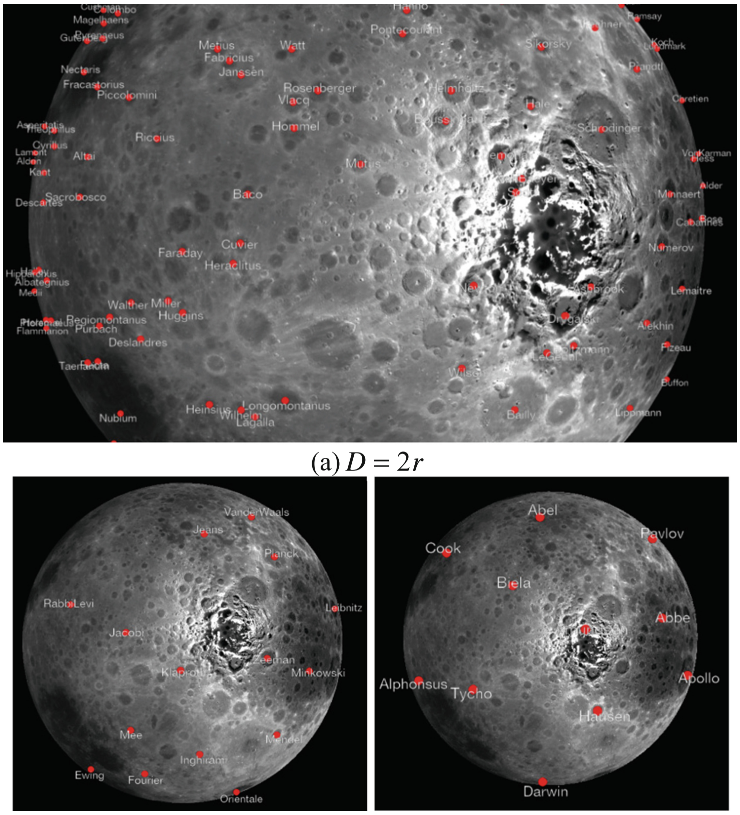


Fig. 2. An integrated software development framework in our system

Our experimental data is the same as that used in [3], including lunar mesh data, texture and label information. Here two-resolution lunar mesh models are used, one is the model with 24576 triangles when the viewpoint is far from the moon; the other is the one with 98304 triangles when the viewpoint is near the moon. The original lunar texture is divided into eight patches in all, as is described in Sect. 2.2. The number of geographical names is 468 in total both in English and Chinese, including 367 craters, 1 plain, 3 cliffs, 15 ridges, 20 mountains, 1 ocean, 2 valleys, 22 lunar mares, 17 lunar lacuses, 11 lunar sinuses, 3 lunar paluses and 6 capes. Three labelling levels are used, which correspond to $D < L_1$, $L_1 \le D \le L_2$ and $D > L_2$ respectively for the thresholds L_1 and L_2. Let r denote the lunar radius, we chose the threshold $L_1 = 2r$ and $L_2 = 5r$. Some experimental results are given in Fig. 3, where D is the distance between a viewpoint and the center of lunar model. By the way, our method can also label Chinese geographical names in corresponding coordinates.



(a) $D = 2r$

(b) $D = 3.5r$ (c) $D = 5r$

Fig. 3. Lunar model visualization with label geographical names at different levels

Our browsing system supports three kinds of gesture interactions including rotation, zoom in and zoom out, which can make us browse the lunar model on mobile device in all-round and multi-angle. The average frame rate of the system is 62 fps for the lunar mesh with 24576 triangles and 33 fps for the model with 98304 triangles.

4 Conclusions

We developed an interactive visualization system of 3D lunar model with texture and labels on mobile device successfully. The system achieves real-time rendering frame rate. Specifically, we develop a vertex shader and fragment shader to render texture so that a technique of terrain labels for mobile device is provided for OpenGL ES 2.0. It is also provide a way to process a large size texture to overcome the limit of OpenGL ES 2.0 programming in texture loading size once time. In addition, a general software development framework for mobile terminal is described and implemented to show how to call C++ project by Java Virtual Machine based on OpenGL ES 2.0 and Osg for android. Therefore, our work provides a typical application for mobile terminal rendering. Everyone can download the lunar browsing system for mobile device from http://www.115.com/?lang=en (username: 13581683182; password: 19850415).

Acknowledgments. This work was supported by the National High Technology Research and Development Program of China (“863” Program) under Grant No. 2013AA013702.

References

1. Sun, Y.K., Mao, K.M., Zhang, T., et al.: A 3D multiresolution lunar surface model using bicubic subdivision-surface wavelets, with interactive visualization tools. Comput. Geosci. **37**(9), 1460–1467 (2011)
2. Sun, Y.K., Dong, Y.F., Mao, K.M., et al.: View-dependent progressive transmission and rendering for lunar model based on bicubic subdivision-surface wavelet. Adv. Space Res. **53**(12), 1848–1857 (2014)
3. Dong, Y.F., Sun, Y.K., Tang, Z.S.: Interactive visualization of 3D lunar model with texture and labels, using Chang’E-1 data. Sci. China Phys. Mech. Astron. **56**(10), 2002–2008 (2013)
4. Sun, Y.K., Dong, Y.F., Tang, Z.S.: Internet-based interactive visualization method of 3D lunar model with texture. Multimedia Tools Appl. (2014). doi:10.1007/s11042-014-1863-z
5. Noguera, J.M., Segura, R.J., Ogáyar, C.J., Joan-Arinyo, R.: Navigating large terrains using commodity mobile devices. Comput. Geosci. **37**(9), 1218–1233 (2011)
6. Noguera, J.M., Segura, R.J., Ogáyar, C.J., Joan-Arinyo, R.: A scalable architecture for 3D map navigation on mobile devices. Pers. Ubiquit. Comput. **17**(7), 1487–1502 (2013)
7. Francisco, R., Oscar, R., Miguel, C.: Efficient visualization of 3D models on hardware-limited portable devices. Multimedia Tools Appl. **73**(2), 961–976 (2014)
8. Ye, M.J., Li, J., Liang, Y.Y., et al.: Automatic seamless stitching method for CCD images of Chang’E-1 lunar mission. J. Earth Sci. **22**(5), 610–618 (2011)