Virtual image generator for asteroid mission navigation

Jiri Burant

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo

Thesis supervisor:

Prof. Esa Kallio

Aalto University
School of Electrical Engineering

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Jiri Burant

Title: Virtual image generator for asteroid mission navigation

Date:	Language: English	Number of pages: 4+10

Department of Radio Science and Technology

Professorship: Circuit theory

Supervisor and advisor: Prof. Esa Kallio

This project deals with testing of possible setups of a camera mounted on spacecraft. The idea is to simulate simplified movement of this spacecraft above the surface and capture images along the way. Based on those images, it can be decided which camera is best suitable for the given mission and at which angle it should be facing. The generated images can also be used in the next stage to test image analysis tools or image reconstruction software.
To do so, a simple application has been developped in java programming language, simulating the movement of the spacecraft and generating the pictures. It consists of a big canvas with 2D image, above which the spacecraft moves and simple GUI, allowing the user to change basic parameters such as the altitude of the spacecraft, the angle of the mounted camera and do on.
The basic application has been developped and tested, giving satisfying results, but there is a lot of room for future extensions, some of them being described in this report.

Keywords: Image generator, space technology project, openGL, java

Preface

I want to thank Professor Esa Kallio for the opportunity to work on this project and for his guidance and optimism.

Helsinky, 6.11.2016

Jiri Burant

Contents

1 Introduction

The project of the virtual image generator has been developped as a helper software tool for the ASPECT project, which is part of the ESA's Asteroid Impact Misstion (AIM).

The goal of the AIM is to demonstrate new technologies, mainly in the telecommunications domain. The AIM mission will consist of the main spacecraft carrying the Mascot-2 asteroid lander, as well as two or more CubeSats. The mission should launch in in October 2020, then travel to the Didymos asteriod pair. The spacecraft would perform high-resolution visual, thermal and radar mapping of the moon and build detailed maps of its surface and interior structure.

(Official site of the AIM project: *AIM website*)

The ASPECT project focuses on developping a single three-unit CubeSat equipped with a visible/near-infrared spectrometer to assess the asteroid composition and effects of space weathering and metamorphic shock, as well as post-impact plume observations.

The Cubesat will be deployed from a 10 km height from the target's surface and orbit it at around 1 to 3 km from the surface. From this distance it will measure the reflectance which will give an understanding of the composition of Didymoon and may provide answers to how this is influenced by space weathering and shock effect.

(Official site of the ASPECT project: *ASPECT website*)

The navigation of the ASPECT CubeSat is a problematic task, and one of the things that could help it are the camera images. By using appropriate camera and appropriate software, position and speed could be estimated by analyzing the captured images. In order to determine a feasible camera setup and to test the analysis software, before it is deployed to the spacecraft, The Virtual Image Generator, VGI tool has been developped.

It simulates the movement of the spacecraft above surface along specified trajectory (assumed as line), capturing images below. Those images are saved as screenshots, and can later be used for testing image reconstruction, mapping software or speed and position estimation software.

The outcome of this work is a basic software application, ready for future extensions. Some of possible future improvements or applications are discussed in section 4. of this report.

2 Software design

Basic information

The software project, as well as the source codes and corresponding libraries can be found on Github (*Space – Camera project*), from there it can be forked or copied by anyone. The application has been developped in java programming language (JDK 1.8), using NetBeans IDE. Therefore, in order to run it, installed java is required. Apart from basic java libraries, the libraries: *jinput*, *lwjgl ver.*2.9.3 and *lwjgl util* have been used. It can be run on any standard operation system (e.g. Windows, LinuxMint, Ubuntu etc.)

The hardware requirements are not strict, any standard computer with installed java and decent graphics card is suitable. During testing, there was an issue with performance, using a notebook with just integrated graphics card, therefore a PC with standard graphics card is recommended.

The library lwjgl (lightweight java openGL library) enables to use the openGL in the java application, providing interface to the lower lever C++ openGL methods. The usage of this library is straightforward, and is very similar to using normaml openGL in C++. There are also many tutorials on the internet, providing guidance for the development. An important thing to notice is that the application is using lwjgl2. There is already newer version, lwjgl3, whose interface is slightly different and should not be mixed with version 2. The older version has been chosen, because the newer one didn't provide support for any GUI libraries at the time of development of this application.

The Github bundle contains also prepared NetBeans project, which has those libraries already imported. Only thing that must be changed in the project specification is the java path to the native libraries. To do that, right click on libraries in the project tab and select properties, in the newly opened window, in the categories select Run and specify VM Options as :-Djava.library.path="Path-to-lwjgl-library/lwjgl-2.9.3/native/Desired-OS" (1).

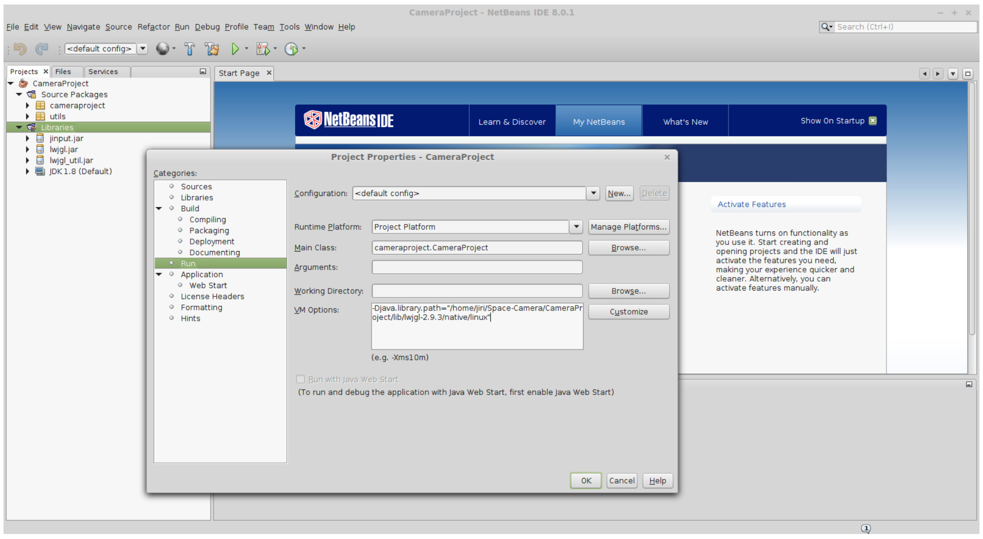
In case, that future development will be carried on using another IDE, the libraries must be added, and the path to native libraries must be set. Here is an example how to do that in eclipse: *Set up lwjgl in Eclipse*.

After the initial imports and after setting the path, no more setup should be needed.

Architecture description

The application project itself consists of two packages. Package called *utils* consists of parts of the source code from the joml java library. It contains classes and methods for manipulating vectors and matrices, which is useful when rendering the openGL objects. Package *cameraproject* comprises all the application logic. It contains classes concerning the openGL tasks, as texture loading and scene rendering, it also contains the GUI setup and operation logic.

The code itself is thoroughly documented and self-explanatory. Nevertheless, a


Project Properties - CameraProject

Configuration: <default config>

Runtime Platform: Project Platform

Main Class: cameraproject.CameraProject

Arguments:

Working Directory:

VM Options: -Djava.library.path="/home/jiri/Space-Camera/CameraProject/lib/lwjgl-2.9.3/native/linux"

(e.g. -Xms10m)

Run with Java Web Start

(To run and debug the application with Java Web Start, first enable Java Web Start)

NetBeans turns on functionality as you use it. Start creating and opening projects and the IDE will just activate the features you need, making your experience quicker and cleaner. Alternatively, you can activate features manually.

Figure 1: Setting up native path in NetBeans IDE

brief description of the most important classes follows.

The main class is called *CameraProject.java* In this class, the basic GUI initialization is carried out as well as the openGL initialization. Moreover, the main rendering loop is also located there.

For each of the rendered objects, that means the target location, the surface plane and the spacecraft, a dedicated class containing the parameters for the given object, has been created. Those classes are named: *Target.java*, *Land.java* and *Carrier.java*, respetivelly. These three classes have a common parent named *SceneObject.java*, which contains the common parameters, that each scene object must have, as Texture *Texture.java* and Model *Model.java*. Class *Texture.java* contains the loaded texture for the given object and the class *Model.java* contains parameters and methods for rendering of the scene objects.

Class named *Shader.java* is purely openGL based, and creates the shaders for rendering. Those shaders are compiled from files *shaderFrag.frag* (for fragment shader) and *shaderVert.vert* (for vertex shader). The shaders specify, how should each pixel be rendered (color, texture, attenuation and so on.)

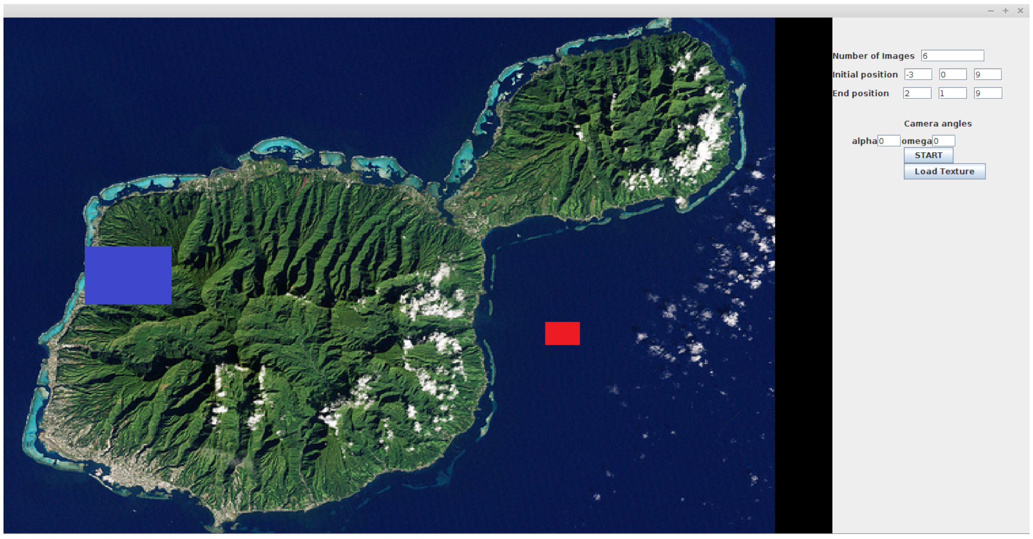
Class *Camera.java* contains the parameter of the camera, such as field of view, angles of the camera and so on. From those parameters, a transformation matrix is computed, which serves for rendering of the "observed" objects.

The remaining classes *TextFieldListener.java*, *FloatWrapper.java* and *InputData.java* are helper classes, which facilitate the transport of data from the GUI text fields into the program.

Graphical User Interface

The GUI has been developped using the java's native Swing library. The GUI is simple and comprises of several labels and text fields for entering the input data and two buttons, one for starting the simulation, one for loading the texture. The layout has been chosen as box layout, because it is straightforward and easy to modify.

In the fig.2, there is an example of how the running apllication looks like. On the left side, there is a surface (land) covered with texture of an island. Over it, a blue and a smaller red rectangles are drawn. They symbolize the spacecraft (carrier) and the target. On the right side, the GUI panel is shown. First field is number of images to be taken during the flight, Underneath is initial position of the spacecraft, the three coordinates specify the x,y,z position in the respective coordinate system. The line end position specified the target location. Last fields are camera parameters, namely two angles of the camera measured as deviation from the y and x axis. Finally, there are buttons for starting the simulation and for loading of the surface texture.



Number of Images 6

Initial position -3 0 9

End position 2 1 9

Camera angles

alpha 0 omega 0

Figure 2: Example run of the application

The coordinate system of the rendered left part of the window is classical 3D Cartesian system, which spans from -4.5 to 4.5 for the x and y coordinates. For the z coordinates, it spans from 0 to 10, while 10 being the coordinate of the surface plane. Therefore the z coordinate is somewhat reversed altitude, as 0 is at the overview, top camera location and 10 is on the surface.

Once the simulation starts, the camera switches to view of the camera mounted on the spacecraft, with the specified parameters. The images captured by the spacecraft are rendered and consequently saved into a file. A few examples of such images are presented in the section 3.

3 Results

Tässä osassa esitetään tulokset ja vastataan tutkielman alussa esitettyihin tutkimuskysymyksiin. Tieteellisen kirjoitelman arvo mitataan tässä osassa esitettyjen tulosten perusteella.

Tutkimustuloksien merkitystä on aina syytä arvioida ja tarkastella kriittisesti. Joskus tarkastelu voi olla tässä osassa, mutta se voidaan myös jättää viimeiseen osaan, jolloin viimeisen osan nimeksi tulee »Tarkastelu». Tutkimustulosten merkitystä voi arvioida myös »Johtopäätökset»-otsikon alla viimeisessä osassa.

Tässä osassa on syytä myös arvioida tutkimustulosten luotettavuutta. Jos tutkimustulosten merkitystä arvioidaan »Tarkastelu»-osassa, voi luotettavuuden arviointi olla myös siellä.

4 Conclusions and future works

The aim of this work was to develop an application capable of generating simulated images captured by a camera mounted on a spacecraft during a flight over 2D surface. It can be stated that such application, for simplified setup, was successfully developped and the results are convincing.

Still, there is a lot of room for future development and more improvements. Several suggestions for improvement are listed below.

Improved GUI and rendering

The GUI, as stated earlier, is very simple. One of the possible improvements could be switching to another GUI library, for example TWL or nifty. That would allow more convenient control, for example adding of sliders for setting the position. Also, the rectangles symbolising the carrier and target are very primitive, therefore a better depiction could be introduced.

Besides those cosmetic changes, also coordinates system could be adjusted. Currently, the span of the x and y coordinates is fixly set from -4.5 to 4.5 for the top view, and the altitude is defined as distance from camera in the range from 0 to 10. For the real world applications, it would be a good idea to add the possibility to specify for example the real size of the 2D plane, and based on this values, the coordinates would be recalculated.

Curved surface

The primary use of this software is presented as part of the asteroids mission program, but in case it finds its use in another projects, for example concerning Earth imaging, than a nice feature could become a slight curvature of the 2D plane, imitating the curvature of the Earth's surface.

In order to do so, the 2D plane would have to be changed to 3D sphere. This can be done by using the sphere object from the *lwjgl.util.glu* package. Of course, such sphere would have quite big diameter, compared to the altitude of the spacecraft.

The problem with rendering of the sphere is the texture mapping. In this work, no sphere textures were used so far, but if it is required in the future, it could be useful to express the coordinates in the spherical coordinate system, or to use the UV mapping technique (*UV mapping on wikipedia*). Although, given the error introduced by the curvature is small, it might be possible to just map the texture as if it was 2D plane and accept this error.

Custom 3D surface

Another extension, similar to the previous one, but more complicated could be loading of arbitrary surface of the 'land'. Such feature could be useful specifically i a project such as the asteroid impact mission, because the surface of the asteriod is very different from 2D plane, and the spacecraft would be close enough to recognize those irregularities.

References

[1] Kauranen, I., Mustakallio, M. ja Palmgren, V. *Tutkimusraportin kirjoittamisen opas opinnäytetyön tekijöille.* Espoo, Teknillinen korkeakoulu, 2006.

[2] Itkonen, M. *Typografian käsikirja.* 3. painos. Helsinki, RPS-yhtiöt, 2007.

[3] Koblitz, N. *A Course in Number Theory and Cryptography. Graduate Texts in Mathematics 114.* 2. painos. New York, Springer, 1994.

[4] Bardeen, J., Cooper, L. N. ja Schrieffer, J. R. Theory of Superconductivity. *Physical Review,* 1957, vol. 108, nro 5, s. 1175–1204.

[5] Deschamps, G. A. Electromagnetics and Differential Forms. *Proceedings of the IEEE,* 1981, vol. 69, nro 6, s. 676–696.

[6] Sihvola, A. et al. Interpretation of measurements of helix and bihelix superchiral structures. Teoksessa: Jacob, A. F. ja Reinert, J. (toim.) *Bianisotropics '98 7th International Conference on Complex Media.* Braunschweig, 3.–6.6.1998. Braunscweig, Technische Universität Braunschweig, 1998, s. 317–320.

[7] Lindblom-Ylänne, S. ja Wager, M. Tieteellisten opinnäytetöiden ohjaaminen. Teoksessa: Lindblom-Ylänne, S. ja Nevgi, A. (toim.) *Yliopisto- ja korkeakoulu-opettajan käsikirja.* Helsinki, WSOY, 2004, s. 314–325.

[8] Miinusmaa, H. Neliskulmaisen reiän poraamisesta kolmikulmaisella poralla. Diplomityö, Teknillinen korkeakoulu, konetekniikan osasto, Espoo, 1977.

[9] Loh, N. C. High-Resolution Micromachined Interferometric Accelerometer. Master's Thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1992.

[10] Lönnqvist, A. Applications of hologram-based compact range: antenna radiation pattern, radar cross section, and absorber reflectivity measurements. Väitöskirja, Teknillinen korkeakoulu, sähkö- ja tietoliikennetekniikan osasto, 2006.

[11] SFS 5342. Kirjallisuusviitteiden laatiminen. 2. painos. Helsinki, Suomen standardisoimisliitto, 2004. 20 s.

[12] Palmgren, V. Suunnittelija. Teknillinen korkeakoulu, kirjasto. Otaniementie 9, 02150 Espoo. Haastattelu 15.1.2007.

[13] Ribeiro, C. B., Ollila, E. ja Koivunen, V. Stochastic Maximum-Likelihood Method for MIMO Propagation Parameter Estimation. *IEEE Transactions on Signal Processing,* verkkolehti, vol. 55, nro 1, s. 46–55. Viitattu 19.1.2007. Lehti ilmestyy myös painettuna. DOI: 10.1109/TSP.2006.882057.

[14] Stieber, T. GnuPG Hacks. *Linux Journal,* verkkolehti, 2006, maaliskuu, nro 143. Viitattu 19.1.2007. Lehti ilmestyy myös painettuna. Saatavissa: `http://www.linuxjournal.com/article/8732.`

[15] Pohjois-Koivisto, T. Voiko kone tulevaisuudessa arvata tahtosi? *Apropos,* verkkolehti, helmikuu, nro 1, 2005. Viitattu 19.1.2007. Saatavissa: `http://www.apropos.fi/1-2005/prima.php.`

[16] Adida, B. Advances in Cryptographic Voting Systems. Verkkodokumentti. Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2006. Viitattu 19.1.2007. Saatavissa: `http://crypto.csail.mit.edu/~cis/theses/adida-phd.pdf.`

[17] Kilpeläinen, P. WWW-lähteisiin viittaaminen tutkielmatekstissä. Verkkodokumentti. Päivitetty 26.11.2001. Viitattu 19.1.2007. Saatavissa: `http://www.cs.uku.fi/~kilpelai/wwwlahteet.html.`

A Esimerkki liitteestä

Liitteet eivät ole opinnäytteen kannalta välttämättömiä ja opinnäytteen tekijän on kirjoittamaan ryhtyessään hyvä ajatella pärjäävänsä ilman liitteitä. Kokemattomat kirjoittajat, jotka ovat huolissaan tekstiosan pituudesta, paisuttavat turhan helposti liitteitä pitääkseen tekstiosan pituuden annetuissa rajoissa. Tällä tavalla ei synny hyvää opinnäytettä.

Liite on itsenäinen kokonaisuus, vaikka se täydentääkin tekstiosaa. Liite ei siten ole pelkkä listaus, kuva tai taulukko, vaan liitteessä selitetään aina sisällön laatu ja tarkoitus.

Liitteeseen voi laittaa esimerkiksi listauksia. Alla on listausesimerkki tämän liitteen luomisesta.

```
\clearpage
\appendix
\addcontentsline{toc}{section}{Liite A}
\section*{Liite A}
...
\thispagestyle{empty}
...
teksti\"a
...
\clearpage
```

Kaavojen numerointi muodostaa liitteissä oman kokonaisuutensa:

$$d \wedge A = F, \tag{A1}$$
$$d \wedge F = 0. \tag{A2}$$

B Toinen esimerkki liitteestä

Liitteissä voi myös olla kuvia, jotka eivät sovi leipätekstin joukkoon: Liitteiden taulukoiden numerointi on kuvien ja kaavojen kaltainen: Kaavojen numerointi

Table B1: Taulukon kuvateksti.

9.00–9.55	Käytettävyystestauksen tiedotustilaisuus (osanottajat ovat saaneet sähköpostitse valmistautumistehtävät, joten tiedotustilaisuus voidaan pitää lyhyenä).
9.55–10.00	Testausalueelle siirtyminen

muodostaa liitteissä oman kokonaisuutensa:

$$T_{ik} = -pg_{ik} + wu_i u_k + \tau_{ik}, \tag{B1}$$

$$n_i = nu_i + v_i. \tag{B2}$$