Fips: An OpenGL based FITS viewer

Matwey Kornilov, Konstantin Malanchev

Sternberg Astronomical Institute, Lomonosov Moscow State University
Universitetsky pr. 13, Moscow 119234, Russia
National Research University Higher School of Economics
21/4 Staraya Basmannaya Ulitsa, Moscow 105066, Russia

E-mail: `malanchev@physics.msu.ru`

Abstract. FITS (Flexible Image Transport System) is a common format for astronomical data storage. [1]. Even though astronomical data is now processed mostly using software, visual data inspection by a human is still important during equipment or software commissioning and while observing. We present Fips [2, 3], a cross-platform FITS file viewer released as open source software[1]. To the best of our knowledge, it is for the first time that the image rendering algorithms are implemented mostly on GPU (graphics processing unit). We show that it is possible to implement a fully-capable FITS viewer using OpenGL [4] interface, including movie support for representing 3D data. We also emphasise the advantages of using GPUs for efficient image handling.

1. Introduction

Modern GPUs have many hard-wired features accelerating typical 2D and 3D-rendering tasks. GPU acceleration has been already used for various astronomical tasks, i.e. solving visualisation problems [5], computational linear algebra problems [6, 7], or real-time rendering and colouring of 3D FITS data [8].

In this work, we concentrate on how typical image manipulation operations may be offloaded to the GPU. Here we prove that the end-to-end OpenGL rendering of FITS file data can be practically implemented as a software application. We propose a FITS viewer software implementation based on the GPU acceleration. Raw FITS file data loaded into the GPU memory, and then the geometric and colour transformations are handled by GPU. The sample application screenshot is given in Fig. 1.

2. OpenGL based FITS viewer

OpenGL is essentially a programming interface, i.e. a library with a standardised set of functions allowing us to control GPU hardware. In an OpenGL application, different kinds of objects may exist: a vertex defining surfaces, and a texture containing images to be drawn on the surfaces. In our case, it is enough to have only four vertexes to define a rectangular plane. The texture containing a FITS image is drawn on this plane. The plane position, orientation, and size are controlled by special GPU routines, so-called vertex shaders. It is useful that the image is drawn on the plane using its local coordinate frame, while the user sees automatically transformed the final image. Therefore, OpenGL gives us a way to perform such operations as pan, zoom,

[1] `https://fips.space`

rotation and flipping. All these operations require us to have only a few lines of source code that executes on GPU in a very efficient manner.

It appeared that handling FITS image as a texture is a possible but complex task. First, we need to load a FITS image into the graphics memory as a texture. Second, we should render the image using the required colour-mapping, contrast level, etc.

If any transformation between the FITS image memory representation and GPU texture representation is needed, it would use additional CPU-based calculations. This is why we focus on cases where byte-representations are similar. This saves us a huge amount of computational time that is usually required for FITS file data parsing and transformation. FITS image data loading is performed quickly and is only limited by CPU to GPU memory transfer rate. It can be done only for some FITS data formats. For instance, a 16-bit FITS image is an array of subsequent 16-bit integers. A monochrome 16-bit texture is also an array of subsequent 16-bit integers. Since it is monochrome (i.e. single-channel), a single integer represents a whole single pixel. This means that memory representations of the 16-bit FITS image and the 16-bit single channel texture should be identical except for the byte order. Unfortunately, it isn't always the case: 32- and 64-bit integers are not supported by OpenGL texture formats. To overcome this kind of difficulties, we present the colour deinterleaving technique that allows us to recover FITS image pixel value in the fragment shader. An example of the technique applied to a 64-bit integer FITS is given in Fig. 2.

Since FITS pixel values are available within the texture, we may apply an arbitrary transformations to them using a fragment shaders. In particular, we use a fragment shader to recover the pixel value from the multi-channel representation in cases when colour deinterleaving is necessary (see Fig. 2). Also, the fragment shader is used to apply colour maps, adjust brightness and contrast levels without precision loss. Note, that the fragment shader is executed on GPU in parallel threads which is a fast and energy-efficient way to make such transformations. Detailed benchmarks are presented in [2].

The OpenGL application may be helpful for astronomical purposes not only due to its ability to perform simple FITS image rendering. The so-called texture arrays available with OpenGL 3 may be properly used to support 3D FITS data. A 3D data cube stored in GPU video memory in the form of texture array representation may allow us to implement efficient GPU video playback. Since data cube is preloaded into GPU memory the next frame rendering is as simple as setting frame number in the fragment shader. It saves a lot of computation time on CPU side while playing the movie. The representation of data cubes in the form of a video is currently implemented in e.g. Ginga and FITSWebQL FITS viewers [9, 10] and it is also currently supported in the coming FIPS 3.4 release.

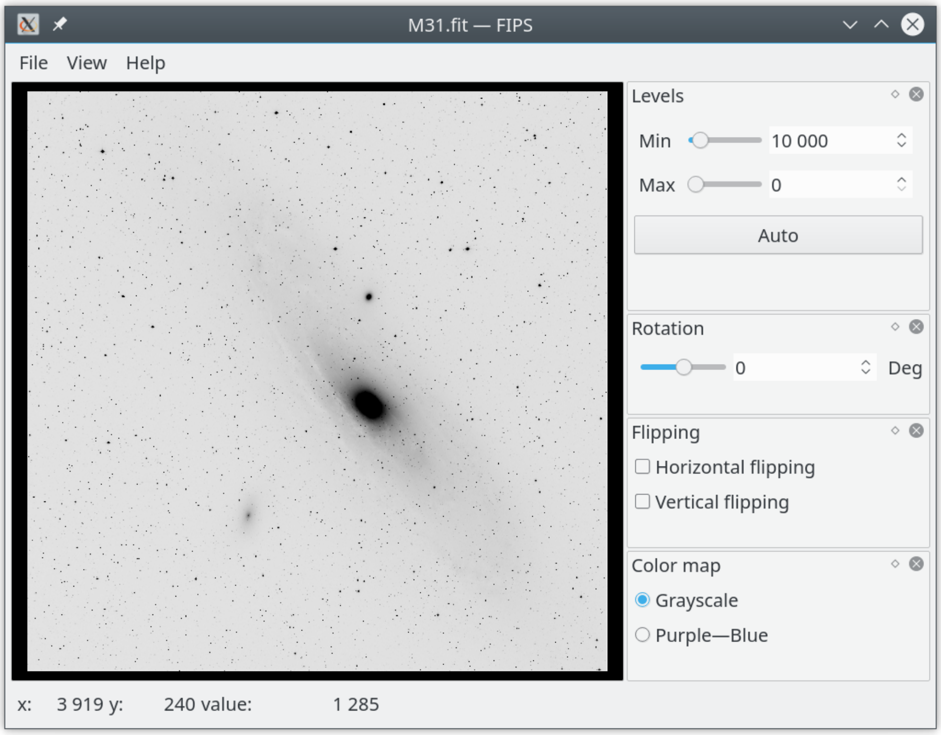
3. Installation

FIPS can be easily installed on all supported operation systems: Linux-based openSUSE and Fedora official repositories contain FIPS packages, other Linux-based operation systems are supported via FlatPak[2], cross-distribution Linux desktop package manager with isolation, macOS package can be installed using Homebrew Caskroom[3], macOS and Windows packages are available on project's GitHub page. Find below detailed information on FIPS installation.

3.1. openSUSE Leap 15.1 and openSUSE Tumbleweed

```
zypper in fips
```

[2] https://flatpak.org
[3] https://brew.sh



Min 10 000
Max 0
Rotation 0 Deg
[ ] Horizontal flipping
[ ] Vertical flipping
[✓ Grayscale] Purple—Blue
x: 3 919 y: 240 value: 1 285

Figure 1: FIPS interface on openSUSE operating system. The user interface looks the same both on Linux and Windows. A M31 galaxy image obtained by the MASTER robotic telescopes network [11] is shown here.

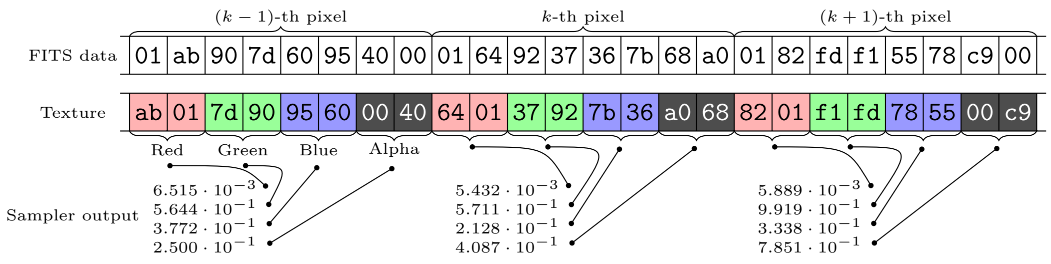


Figure 2: Memory layout example. At the top layer, FITS image linear memory data represent 64-bit integer pixels in big-endian order. At the middle layer, the texture memory representation for a 16-bit RGBA (Red, Green, Blue, Alpha) pixel in little-endian architecture is given. At the bottom layer, floating point vectors are shown that are returned by the sampler when the texture is accessed.

3.2. Fedora 30

```
dnf in fips
```

3.3. FlatPak

```
flatpak install flathub space.fips.Fips
```

3.4. macOS
Download release package from GitHub releases (https://github.com/matwey/fips3) or install using Homebrew Caskroom:
```
brew cask install fips
```

3.5. Windows
Download release package from GitHub releases (https://github.com/matwey/fips3).

3.6. Building from sources
Fips can be built on target operation system using modern C++ compiler and Qt[4] library.
```
  git clone https://github.com/matwey/fips3
  mkdir -p fips3/build
  cd fips3/build
  cmake ..
  make
```

4. Conclusion
In this proceeding, we have presented the software for rendering astronomical data in the form of FITS images and movies. The major design novelty is using GPU acceleration: the image geometry and colour transformation are programmed in GPU using the OpenGL programming interface. It turns out that the full processing stack, starting with loading bytes from a FITS file into the GPU memory, to rendering the picture on the user screen, may be implemented by applying all necessary data transformations in the GPU. Note, that OpenGL is still supported in environments without hardware GPU by efficient CPU software implementations such as Mesa llvmpipe [5], that provides Fips users with a good experience even in such cases.

We offer astronomical community to enjoy a modern graphical user interface of Fips and to contribute new ideas and code to the project.

Acknowledgements
Authors thank the referee for the constructive comments which helped to improve the paper. The study was partially supported by RBFR grants 18-32-00426 and 18-32-00553, and MSU Development Program "Outstanding Scientific Schools of Moscow State University: Physics of Stars, Relativistic Compact Objects and Galaxies".

References
[1] Wells D C, Greisen E W and Harten R H 1981 A&AS **44** 363
[2] Kornilov M and Malanchev K 2018 Fips: An OpenGL based FITS viewer Astrophysics Source Code Library ascl:1808.006
[3] Kornilov M and Malanchev K 2019 *Astronomy and Computing* **26** 61 (*Preprint* 1901.10189)
[4] Segal M and Akeley K 2018 The OpenGL Graphics System: A Specification (Version 4.6) Tech. rep. The Khronos Group Inc. URL https://khronos.org/registry/OpenGL/specs/gl/glspec46.core.pdf
[5] Perkins S, Questiaux J, Finniss S, Tyler R, Blyth S and Kuttel M M 2014 *New Astronomy* **30** 1 – 7 ISSN 1384-1076
[6] Fromang S, Hennebelle P and Teyssier R 2006 *A&A* **457** 371–384
[7] Liska M, Hesp C, Tchekhovskoy A, Ingram A, van der Klis M and Markoff S 2018 *MNRAS* **474** L81–L85
[8] Vohl D, Fluke C J, Barnes D G and Hassan A H 2017 *MNRAS* **471** 3323–3346 (*Preprint* 1707.00442)
[9] Jeschke E, Inagaki T and Kackley R 2013 Introducing the Ginga FITS Viewer and Toolkit *Astronomical Data Analysis Software and Systems XXII* (*Astronomical Society of the Pacific Conference Series* vol 475) ed Friedel D N p 319

[4] https://qt.io
[5] https://www.mesa3d.org/llvmpipe.html

[10] Zapart C, Shirasaki Y, Ohishi M, Mizumoto Y, Kawasaki W, Kobayashi T, Kosugi G, Morita E, Yoshino A and Eguchi S 2018 (*Preprint* 1812.05787)

[11] Lipunov V, Kornilov V, Gorbovskoy E, Shatskij N, Kuvshinov D, Tyurina N, Belinski A, Krylov A, Balanutsa P, Chazov V, Kuznetsov A, Kortunov P, Sankovich A, Tlatov A, Parkhomenko A, Krushinsky V, Zalozhnyh I, Popov A, Kopytova T, Ivanov K, Yazev S and Yurkov V 2010 *Advances in Astronomy* **2010** 349171