Scientific Visualization, 2023, volume 15, number 1, pages 83 - 89, DOI: 10.26583/sv.15.1.07

Universal System for Visualizing Geo-Referenced Data From a Text File on a Yandex Map

N.N. Voronina[1]

FSBSI FRC Marine Hydrophysical Institute RAS, Sevastopol, Russia

[1] ORCID: 0000-0001-7301-2609, voronina.nataly@mail.ru

Abstract

In 2016, an article [1] was published in the journal Scientific Visualization No. 1, which marked the beginning of a further series of publications, one way or another related to the visualization of geo-referenced data. The work is indirectly a continuation of this series, since, similarly to previous articles, the result of the work can also be seen on the Yandex satellite map. The article describes in sufficient detail the step-by-step process of developing and debugging a software product, the ultimate goal of which is the visualization of geophysical data with their subsequent display on the Yandex satellite map in accordance with the geographic coordinates that are set as input. The software product is a fully debugged complete module that can work both offline and, in a complex, aimed at laying out the result on the Yandex satellite map.

Keywords: georeferenced data, visualization, Yandex satellite map.

Introduction

There are many examples of building of images based on text files containing georeferenced data (data that has map reference coordinates). For these purposes, at various times, graphic editors Surfer [2], MathCAD [3], etc. were created, which can work with input text files and process these files in the form that is interesting to the user. In this article, we will not dwell on each of editors; the article has a different goal, namely, to show the process of creating an image from any text file for subsequent display on a Yandex satellite map without involving well-known powerful graphic editors.

API (abbr. from the English Application Programming Interface - a description of how one computer program interacts with others) Yandex [4] has all the necessary functionality for the implementation of this project, despite some limitations compared to other satellite maps (Google Maps, for example). In addition, Yandex is a domestic development, which is of no small importance. The authors set themselves the goal of universal image generation based on a text file that has in its arsenal only the values of a certain parameter (be it sea surface temperature or the number of visitors to supermarkets over a period), as well as georeference values (longitude/latitude) of the parameter. That is, if we are talking about the temperature of the sea surface, then the latitude and longitude of the point at which the temperature parameter takes exactly this value (obtained from observations or predicted) act as a georeference. If the parameter is the number of visitors to supermarkets for the period, then the file with georeferenced data will use the longitude/latitude of the point with the location of the supermarket of interest. In other words, longitude/latitude uniquely identify the locations of the values of the parameter that will be shown on the rendered image.

The following assumption should be noted: - any text file that has not been previously processed by graphic editors or anything similar, such as files of GRD type (the result of processing by the Surfer editor or Adobphotoshop [5]), can be used as input. As a software tool for the subsequent implementation of image construction, the scripting programming language PHP [6] was chosen, since it is best integrated into any site, namely the site with the

Yandex satellite map in our example, where, ultimately, the built image will be sent. In addition, PHP has the full range of graphics functions needed to create images.

So, the final goal, namely the visualization of arbitrarily given georeferenced data, will be implemented in several steps:

1 black and white rendering of image,
2 “filling” the constructed image with color,
3 laying out the image on the Yandex satellite map.

Step 1 – black and white rendering of image

The input data is a text file with the CSV extension, consisting of three columns:

- latitude/longitude in the first two columns;
- the actual value of the parameter (in our example, this is a truncated geostrophic, that is, only one of the two velocity components).

In this case, the extension in the name of the text file matters only so that a function designed specifically for this extension is used to open this file in the PHP program.

First, let's build a black and white image of the data contained in the input file. The black-and-white image is built only on the basis of the longitude/latitude of those points in which the parameter values are present. Actually, the value of the parameter itself will appear at the development stage in step 2, since the parameter values in the figures are actually a color palette, which we will see later.

The result of the script at step 1 is shown in Figure 1. The canvas size is calculated and depends on what is the maximum number of points in the input file represented by latitude and longitude:

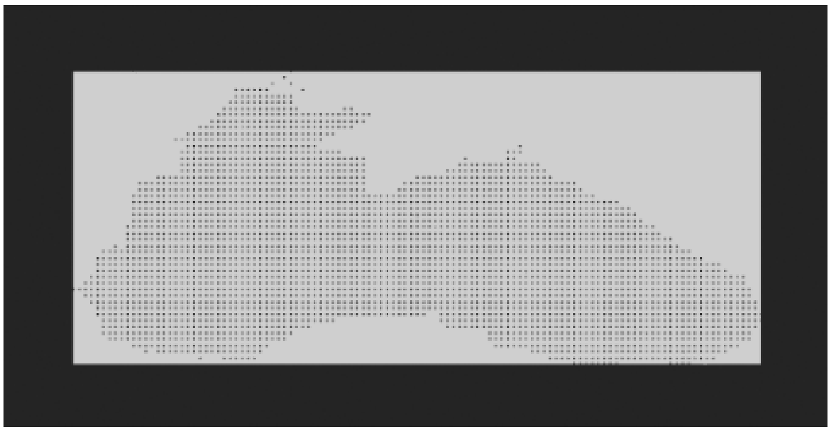


Figure1. *Primary black and white image of the input data*

x = (max_do - min_do)* const,
y = (max_sh - min_sh)*const,
where:
max_sh, min_sh – values of maximum and minimum latitudes in the input file,
max_do, min_do - the values of the maximum and minimum longitudes in the input file, respectively,
const – an arbitrary constant that specifies an increase / decrease in the image to improve viewing (a kind of scale).

In order to get the same image in a solid black and white representation, you can use the constant **const**, reducing our result, and see the image on the screen, see Figure 2.

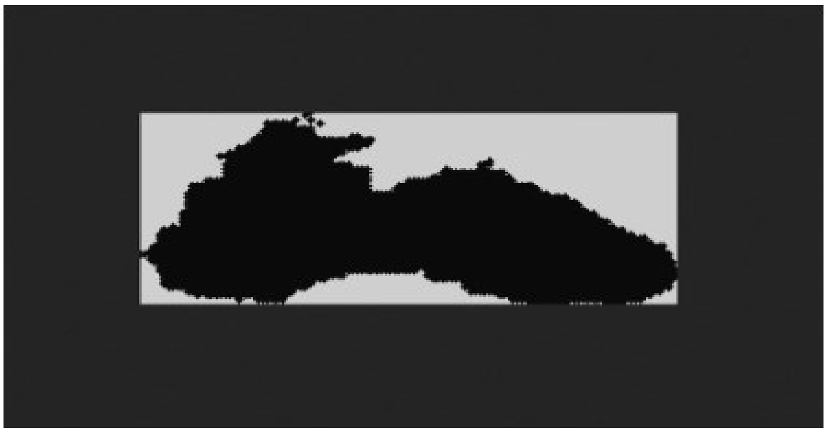


Figure 2. *Primary image of input data in solid black and white representation*

To clarify, points are clearly visible in Figure 1, but nevertheless, to construct them, the function for constructing an ellipse imagefilledellipse () was used with black (temporarily) as a fill color. The point in this case acts as a particular of the ellipse, that is, the point with equally given sizes of radii is considered an ellipse (functionally more universal).

In the PHP script, in addition to the above, the following functions were also used in step 1:

imagecreatetruecolor() – creating an image of a calculated size,

imagecolorallocate() – creating colors (white for building drawings on canvas and black for filling dots-ellipses),

imageFilledRectangle() – building a white rectangle on the canvas as the background of the image.

At this point, step 1 is considered fully implemented.

Step 2 – "filling" the constructed image with color

At this stage, we will additionally use the text data of the same file, but with an arbitrary palette file of the RGB type, that is, the palette file contains three columns corresponding to the values Red, Green, Blue. For now, for clarity, we will build the palette directly on the same canvas created earlier in step 1. To do this, we increase the size of the canvas calculated in step 1 by the desired width of the palette:

- x = ((max_do - min_do) + const_add_palett)* const,

where const_add_palett – the desired width of the palette image (for example, from 0 to 10). Zero in this case means no palette on the canvas.

So, it is necessary to paint over each point of the original black-and-white image with the color from the palette that corresponds to the parameter value at the point with the specified longitude/latitude. To establish this correspondence, we will proceed as follows:

- count the total number of points with parameter values (in our example 2993) and the total number of lines with color values in the palette, for example 150;

- since there are significantly more points with parameter values than colors in the palette, 2993 versus 150, and when choosing other input data in the future, this excess will remain unchanged to one degree or another, we will determine by simple calculations which color the range of parameter values from the input data will belong to;

- in the imagefilledellipse() ellipse construction function specified in step 1, change the fill color from black to the color of the parameter, the correspondence to which we have already determined above.

Figure 3 shows the output of the updated PHP script, displaying the used palette of different widths (left and right), as well as indicating the maximum and minimum values of the parameter corresponding to the start and end colors of the palette.

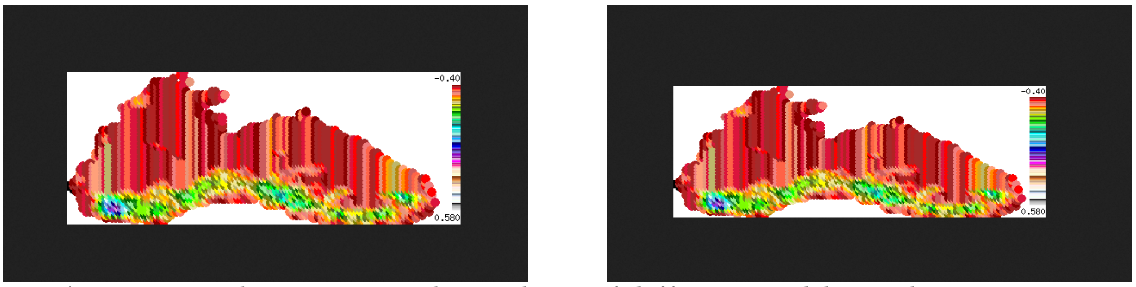


Figure 3. *Color image with a palette of different widths and inscriptions*

It should be noted that the indications of several parameter values, in addition to the maximum and minimum, can be specified in different ways - either by the corresponding inscriptions-divisions at the palette scale itself, or as a "pop-up" hint when "removing" these values from the canvas, similarly to "removing" coordinate values on the Yandex satellite map [7], about which certain assumptions should be made in the future.

In the PHP script, in addition to those already used earlier in step 1, the following functions were used in step 2:

imagecolorallocate() – creating different colors to build a palette,

imageFilledRectangle() – building a rectangle on the canvas with all colors as a palette,

ImageString () – to create inscriptions on the palette image.

The first two steps will be considered preliminary before laying out the generated image on the Yandex satellite map [8].

Step 3 - laying out the image on the Yandex satellite map

In the finished image, before laying it out on the map, one more thing needs to be done - a transparent background. But since this requires adding only one function Imagecolortransparent() for the background color (in our example, it is white), and the result of this function's work appears best after it is laid out on the map, it was decided to add transparency to the image at step 3 - when laying out on a satellite map.

Let's insert the finished PHP script for building the image into the HTML frame.

Let's transfer the already finished image with the necessary parameters (coordinates of points with the parameter, the center of the future map, etc.) to the JS script [9], since it is JS that is responsible for building the Yandex satellite map. As a result, we get the
image shown in Figure 4.

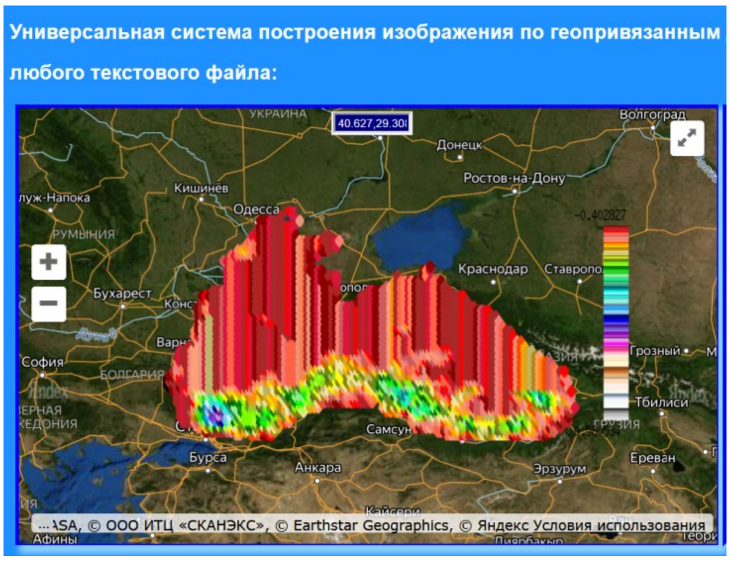
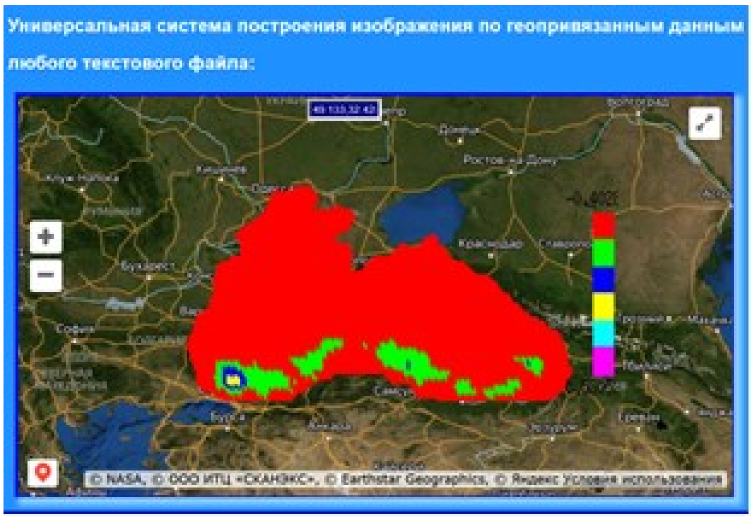

Универсальная система построения изображения по геопривязанным
любого текстового файла:

Figure 4. *Image on Yandex satellite map*

Now that everything is ready, as an experiment, let's try to replace the palette file, using a palette with a small number of colors for this. Then the same image [10] will look as shown in Figure 5.

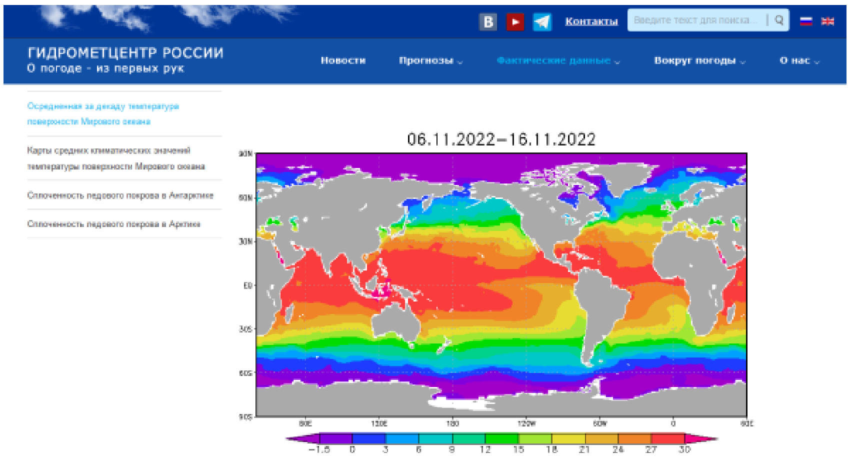

Универсальная система построения изображения по геопривязанным данным любого текстового файла:

Ростов-на-Дону
Краснодар
Бухарест
София
Самсун
Бурса
Анкара
Ереван
© NASA, © ООО ИТЦ «СКАНЭКС», © Earthstar Geographics, © Яндекс Условия использования

Figure 5. *Image on the Yandex satellite map with a different palette*

In other words, in the future, the user himself will decide and accordingly choose which palette is more convenient for him to use.

Finally, Figure 6 shows an example of the image of the temperature averaged over a decade (as a parameter) of the surface of the World Ocean on the website of the Hydrometeorological Center of Russia [11], made by completely different means. This example can be done with our fairly simple method described in the article.


ГИДРОМЕТЦЕНТР РОССИИ
О погоде - из первых рук

Осредненная за декаду температура поверхности Мирового океана

Карты средних климатических значений температуры поверхности Мирового океана

Сплоченность ледового покрова в Антарктике

Сплоченность ледового покрова в Арктике

06.11.2022–16.11.2022

Figure 6. *Actual data from the Hydrometeorological Center of Russia*

Conclusion

Note that these works with step-by-step imaging were carried out by a somewhat unconventional method, that is, the program was debugged not by separate modules, but each new step included the previously debugged one, that is, a certain kind of "stringing" took place. Thus, it was this approach that made it possible to reach the final goal with an already fully debugged, working PHP script.

This pivotal, working script will serve as the main basis for our further works towards universality. Note that this further works are already underway. Universality will consist in the ability to select any text file, as well as a file with a palette. In the future, we will make the palette image separate from the main image in order to be able to place the palette image anywhere behind the map vertically or horizontally at the request of the researcher. And most importantly, we have to make a software product for constructing the speed parameter, that is, arrows will be drawn on the image indicating the direction of movement of high-speed flows. I must say that all future results will be reflected in the article, which will be a continuation of this article.

If we turn to examples of the global level, then on the COPERNICUS website [12], where a huge number of images of geophysical parameters are presented, we note that the software product can also be used in analogue systems of Copernicus. Also, this software product can become the basis for works in any region on the cartographic surface. On the site with the Yandex map, in the future it is planned to create various kinds of selections of input data and palettes of interest, which will form the basis of the topic for another article.

Acknowledgements

*The work was carried out within the framework of the state task on the topic FNNN-2021-0004 "Fundamental research of oceanological processes that determine the state and evolution of the marine environment under the influence of natural and anthropogenic factors, based on observation and modeling methods" (code "Oceanological processes")***.**

References

1. Voronina N. N., Inyushina N. V., Mamchur N. L., Kryl' M. V. *Analiz i sopostavlenie programmnyh sredstv vizualizacii morskih prognozov na osnove raschetnyh dannyh operativnoj okeanografii po chernomorskomu bassejnu na primerah sevastopol'skogo i krymskogo regionov.*// Nauchnaya vizualizaciya. Elektronnyj zhurnal, ISSN 2079-3537, № 1, T. 8, 2016. – s. 146 - 155. [Electronic resource]. URL: http://sv-journal.org/2016-1/09.php?lang=ru
2. Dogan T., Kastner P., Mermelstein R. *Surfer: a fast simulation algorithm to predict surface temperatures and mean radiant temperatures in large urban models* // Building and Environment. 2021. T. 196. C. 107762.
3. Nikulin Ye. *Kompyuternaya 2d-grafika. Programmirovanie v MathCAD*// Lan, 2022
4. Koptenok Ye. V., Savenko A. V., Fomin I. I., Trunnikov M. V., Sukharev Ye. A. *Analiz vozmozhnostey servisov dlya realizatsii virtualnykh turov na primere ispolzovaniya API Yandeks.Kart.*//Molodoy uchenyy. № 19 (309), 2020. - S.130-133. URL: https://moluch.ru/archive/309/69891/
5. Folkner E., Chavez K. *Adobe Photoshop SS. Ofitsialnyy uchebnyy kurs* // Eksmo-Press, 2021.
6. Zandstra M. PHP 8. *Obekty, shablony i metodiki programmirovaniya* // Dialektika, 2021. [Electronic resource]. URL: https://php.ru/manual/
7. [Electronic resource]. URL: http://innovation.org.ru
8. Mokhov V.A., Kubil V.N., Kuznetsova A.V., Georgitsa I.V. *Rekursivnyy algoritm sinkhronizatsii Api-zaprosov k gis-servisu Yandeks.Karty*//Fundamentalnye issledovaniya. № 9-1, 2015. – S. 33-38. URL: https://fundamental-research.ru/ru/article/view?id=38961
9. Foerderer J., Bender M., Heinzl A. *Regulation of digital platform ecosystems: evidence from russia's google vs yandex ruling.//* V sbornike: International Conference on Information Systems 2018, ICIS 2018. 39. 2018.
10. [Electronic resource]. URL: http://innovation.org.ru/generate_image_do-sh.php
11. Servis Programmy «Gidromettsentr Rossii» // [Electronic resource]. URL: https://meteoinfo.ru/ocean
12. Servis Programmy «Kopernikus» // [Electronic resource]. URL: https://data.marine.copernicus.eu