web: suruse.uni-ruse.bg

GRAPHICAL EDITOR FOR ANDROID

Valentin Velikov, Ivelin Mitev

Angel Kanchev University of Ruse

Abstract: *This paper present a Graphical Editor for Android – fully functional application, with intuitive and simple interface. The system is created to demonstrate some techniques, advanced technologies and knacks in the subject area and to show them in the teaching process. The architecture, basic principles and base components of the application are presented. The idea of the screen organization is shown.*

Keywords: *Computer Science, Android, Graphical editor, Android teaching.*

INTRODUCTION

There are many graphics processing applications for mobile devices. Some of them offer excellent functionality. They have a large number of filters and effects, a diverse set of brushes, and various interesting and innovative options. But applications that offer such a wide variety of functionalities are paid.

Free versions are good choice for everyone who does not need more sophisticated image processing than just some basic functionality. The only drawback of free applications is the presence of ads in most of them, although they are not particularly intrusive.

With the development of this project two goals are realized:

• to create an Android application offering basic tools for image editing;

• to have own application code with fragments of techniques and functionalities that can be used to demonstrate and illustrate relevant topics in the teaching process of the Mobile Device Programming course.

RELATED WORK

Several other similar applications have been explored before creating this graphics editor: ***Photo suite*** [4], ***ArtFlow: Paint Draw Sketchbook*** [6], ***Painter Mobile*** [9], ***Sketch Master*** [8], ***My Paints Free, SketchBook Express*** [7], ***Silk paints drawing*** [11] and more [3]. Each of them has its own advantages and disadvantages that have been analyzed and on this basis it has been decided what functionalities the current application should have.

Many apps have a paid and free version. In paid versions, it is difficult to specify functional flaws (though - it should be borne in mind that these are not desktop applications and Adobe Photoshop capabilities are not to be expected).

With free, demo, and trial versions, a variety of shortcomings - random or deliberate - can be mentioned. Among the main drawbacks are: insufficient effects, insufficient tools, insufficient filters, lack of layers, limited undo/redo functionality, difficulty or lack of direct image insertion from the camera, missing rotate/flip, missing options for placing figures and text, lack of predefined figures, difficulty or lack of ability to work with their own colours, etc.

At the same time, many useful ideas can be taken from various applications: screen organization, tools, filters, templates, textures, tutorials, image

transformations, redefined shapes and emoticons, direct sharing on social networks, etc.

DETAILED DESCRIPTION

The application offers basic features that can be upgraded over time. These features are easily accessible and their use is not a problem for non-professionals or enthusiasts on the subject. That is people who did not deal with image processing have no difficulty in using the application. For this purpose a clean and simple interface should be created.

The overall structure of the application and its components [10] [5] [2] is so developed that it allows the addition of new functionality and expansion of the existing without need for major code changes.

In order to facilitate the app usage in the teaching purposes, some basic requirements are met during the implementation:

- compliance with Java code-writing conventions;
- the application is built on a modular principle;
- program units are short, clear, well-structured;
- program units are well documented and described with appropriate comments.

1. <u>Architecture and basic application components</u>

Fig. 1 shows a diagram of the application main components and their relations.

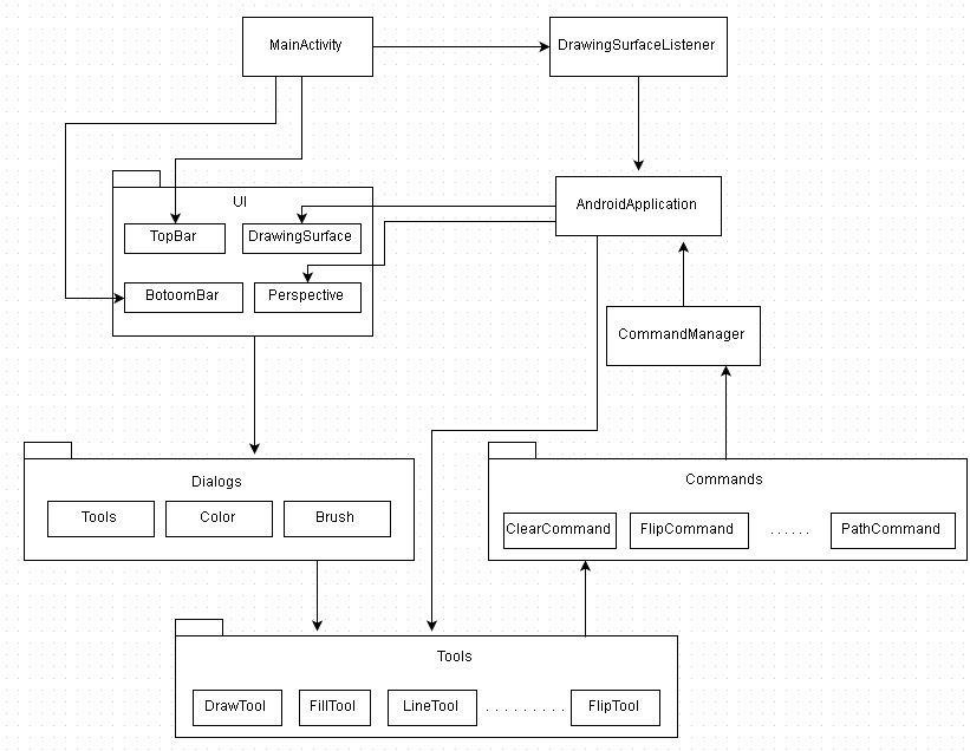


Fig.1. Main components of the application

The input application point is the MainActivity class. This class includes the menu bars (*TopBar* and *BottomBar*). In addition, *DrawingSurfaceListener* is also

initialized here. This is the class that implements the *onTouchListener*, which is responsible for processing user touches and movements on the screen.

The classes *TopBar* and *BottomBar* take care of the two strips at the top and bottom of the screen. They include the buttons on the various menus. Different dialogs (ToolsDialog, ColorDialog, BrushDialog) appear in their use.

DrawingSurfaceListener sends to *AndroidApplication* the coordinates of the user's finger and the action performed. *AndroidApplication* in turn passes them on to the current tool.

In addition to having the coordinates and action of the user, the tool listens through the listener if the user does not change any of the options for colour change or brush settings. If there is a change to any of them, the dialog notifies the tool through these listeners. Then the tool invokes a command.

The command contains the way in which the object has to be drawn. All commands are added to *CommandManager*.

The *CommandManager*, in turn, transmits all commands to a *DrawingSurface,* found in *AndroidApplication*. The *DrawingSurface* class performs the drawing of the commands on the screen itself. The class *Perspective* is a helper class that is used when scaling and translating the object.

2. Basic components of the application

The application is developed on a modular principle. *Fig. 2* shows the interaction of the separated modules and the communication between them.

- **AndroidApplication**

Inherits the *Application* class. For this reason, an instance of *AndroidApplication* is created when the application itself starts. This instance is global and can be accessed anywhere in the app.

AndroidApplication holds the application's global status. This means that it contains all the important elements of the application. These are the context of the application; the object where the drawing is performed *(DrawingSurface)*; the object *CommandManager* (responsible for the command management); the current tool, used from the user (*Tool*); the application menu, as well as some variables, which contain information about whether the image is saved or not.

- **Tools**

These are all available tools in the app. When a tool is selected from the tool selection dialog, an instance of the selected tool is created.

All tools implement the Tool Interface. They do not draw directly into the end object that the user has already drawn, they use commands for this purpose. The role of the tools is to validate the data, draw the intermediate state of the object and split the actions for better readability of the code.

The classes implemented by Tool receive the following methods:

- handleDown – the method is called when the user touches the screen. This method determines the starting point;

- handleMove – the method is called when the user performs some movement on the screen. Depending on the Tool, different actions are taken with the coordinates that this method receives. The intermediate drawing of the object is performed here (for those tools where some kind of painting is done). That is, the drawing of the object that is still drawn by the user is done by the draw () method of the Tool itself, not by CommandManager.

- handleUp – the method is called when the user no longer touches the screen. This method creates a new command and places it in CommandManager.

• **DrawingSurface**

DrawingSurface present the surface on which to draw (Canvas, Bitmap). Drawing in this class is done in a separate thread.

The class *DrawingSurface* contains the nested class *DrawLoop* that implements the *Runnable* interface. The *onDraw* method form *DrawingSurface* is invoked every 20 milliseconds. This method draws all commands from *CommandManager*.

The class *DrawingSurfaceLisneter* is an *onTouchListener* into a *DrawingSurface* which handles the *touchEvents* and invokes the methods *handleDown*(), *handleMove*() and *handleUp*() of the active tool.

• **Undo/Redo functionality**

The Command Manager is designed to reduce the use of resources (mainly memory) in undo / redo functionality.

Fig. 2 shows the Undo / Redo workflow of the application. When selecting a tool and then using it, the tool invokes a command. After executing the command, it enters the command manager. Immediately after that, *UndoRedoManager* is called. Its task, depending on whether there are previous commands in the command manager, is to update the graphical interface buttons. For example, the user has returned the image in its original state and the undo button has to be disabled or at the beginning of the application when the user has not done anything yet the redo button is not active.

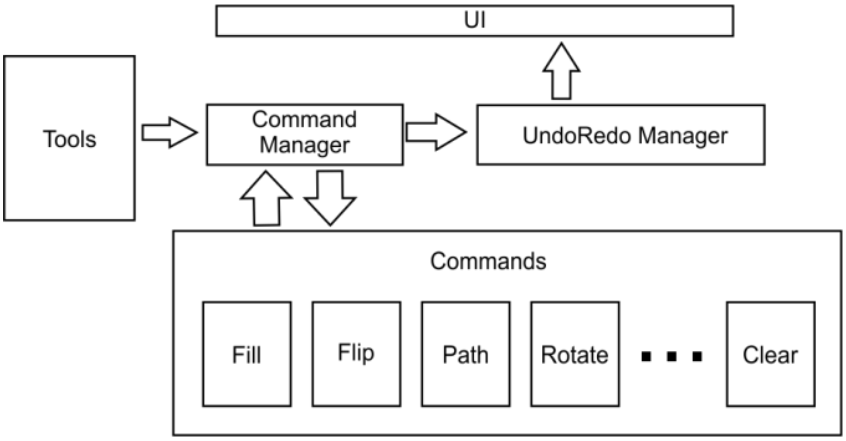


Fig.2. Undo/Redo functionality

• **CommandManager**

The command manager itself is a *LinkedList* of *Commands* and a counter of all active commands. When starting the application, there is a single command in *CommandManager* and this is the *Clear* command. This command deletes everything that has been done, i.e. the status of the application is a ***white sheet*** - what it was when it was started. The command *Clear* cannot be removed from *CommandManager*. It is the initial (default) command. If the user cancels all of his actions, he will get to the starting state (the *clear* command).

Fig. 3 shows that the user has performed 5 actions, no matter what they exactly are. At this point, the *LinkedList* has 6 commands (clear + 5 custom), the counter is 5.

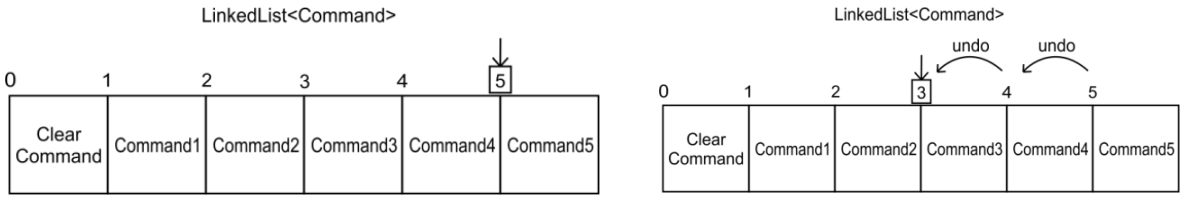


Fig.3. CommandManager – example with 5 actions, and UnDo command

In Fig. 3, the user has pressed the undo button 2 times, and the current commands counter is already 3. Thus *drawingSurface* draws only the start command (clear) along with the first 3 user's commands. The last 2 commands are not drawn due to the command counter, which indicates that only 3 commands are active at this time. If the undo is pressed again, the counter will become 2 and only 2 commands will be drawn. When you press the redo button, it does exactly the same, but the counter is incremented.

A LinkedList was selected instead of the more widely used *ArrayList* because it is more optimized for this application. Each user action inserts a new command into the *CommandManager*, that is a very common insert operation. In addition, when **undo** and a new action is performed, it is possible to delete one or more elements, which means often using **remove**.

The *ArrayList* insert time is O(1) at best and O(n) at worst. In *LinkedList*, the time is always O(1), that is it is constant (Fig.4).

The *ArrayList* remove time is O (n), while in *LinkedList* it can be from O (1) to O (n).

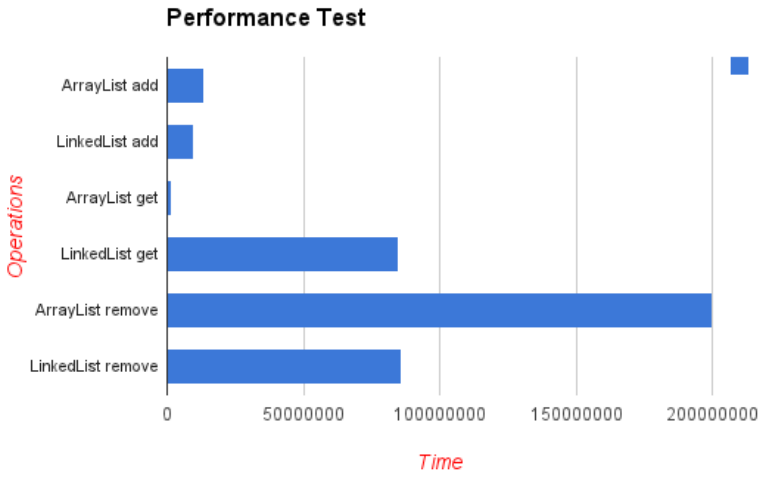


Fig.4. Performance test for ArrayList and LinkedList

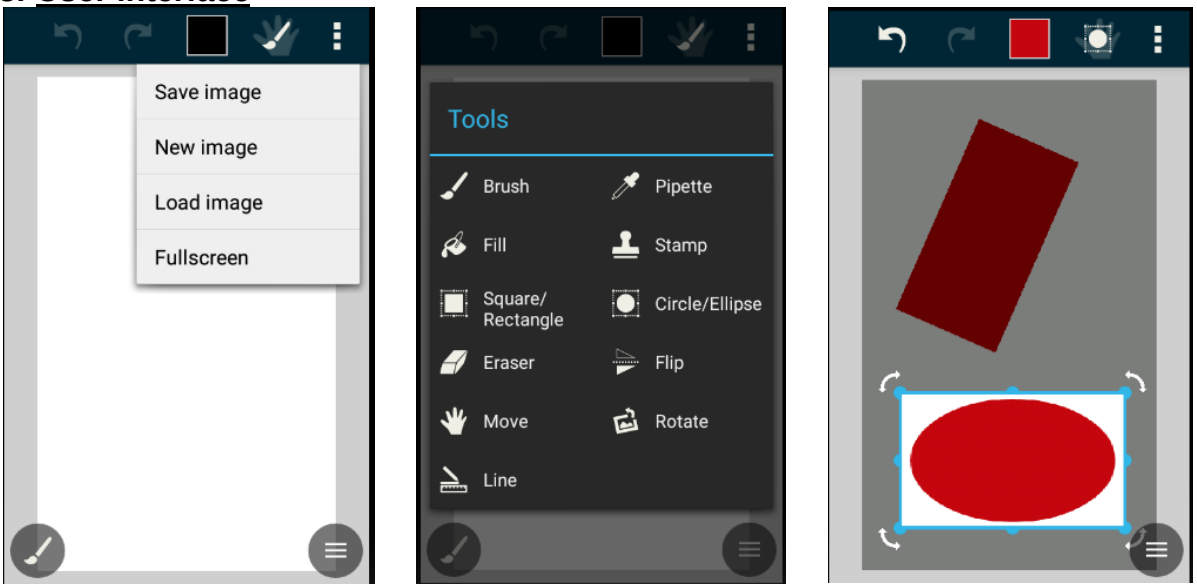
- **Observable**

CommandManager and *BaseCommand* inherit the *Observable* class. The *Observable* class takes care of this particular group of objects being notified when an event occurs. In this case, when executing any command, it must notify the *CommandManager* if it has been successfully executed or an error has occurred during execution. This is necessary for the *CommandManager* to know if the command has failed to successfully execute and to remove it from the *LinkedList* of all executed commands.

• **QueueLinearFlood algorithm**

This algorithm [1] is used in Fill functionality - filling a user-selected area of the image with a colour.

3. User interface



main menu tools drawing area

Fig.5. A part of the User Interface

A basic requirement for the user interface (Fig.5) is to be simple and straightforward. Due to the nature of the application, the main part of the screen is reserved for a drawing area. The main features are assembled in a bar at the top, when a button is pressed it is opens a window or a menu tool for selection or setup.

TEST RESULTS

Several types of tests have been performed with this application. They can be united into two main groups: the first - for performance and the second - Unit and Integration.

Performance tests: Memory consumption and CPU load were tested to perform some of the heaviest operations: Fill, Stamp, Flip, Save and Load an Image. Appropriate size diagrams were used for the study.

Fill - to fill up to 90% of the screen - it uses a lot of processor time (about 5 seconds, 50% of the CPU – despite of the dedicated resource rescue algorithm), the memory use is at a good level (1.5 MB).

Stamp - When taking a small part of the image, a small amount of memory is allocated, which is released immediately after it is placed. The processor is almost idle.

Flip - about 2 MB of memory is allocated (for rearranging the pixels) which is released immediately after the task is completed. The processor is hardly used - there are hardly any calculations, just copying and moving pixels.

Saving and loading an image - The processor is not used much, unlike the memory. However, its amount depends on the image itself.

Unit and Integration tests: During the development of the project upon completion of each code snippet (method), it was tested until it was clear from errors and showed the desired behaviour with **junit**.

Upon completion of each module (Tools, Dialogs), integration tests were performed to verify that the specified tools successfully work with commands, the CommandManager and the different features of the tool itself (colour, line thickness, etc.). Dialogues have been tested if they properly visualize the state of the tools and whether they correctly create a new tool and change its features.

The conclusions from the tests are:

- The application offers very good basic functionality, along with some additions such as full-screen mode, stamp option, camera capture capability and direct image import;
- The options are very flexible. For each of them the colour, size, shape and blurring of the line can be changed. There is nothing that is embedded and cannot be changed by the user;
- Simple interface where all options require no more than 2 clicks to select.

CONCLUSION

A Graphical Editor for Android with simple user interface was implemented. It was created as application in a modular principle, with a good documentation to easily use it for a demonstration in the teaching process. In the future it is planned to extend its functionality by adding effects filters, manipulating the features of the image itself, adding new tools, increasing the number of predefined filters.

ACKNOWLEDGEMENTS: This work is supported by the National Scientific Research Fund under the contract NSRF- I02/13.

REFERENCES

[1] J.Dunlap, Queue-Linear Flood Fill: A Fast Flood Fill Algorithm, http://www.codeproject.com/, 15.04.2017.

[2] Marko Vitas, ART vs Dalvik - introducing the new Android runtime in KitKat, https://infinum.co/the-capsized-eight/, 14.04.2017.

[3] AppCrawlr, Softonic International S. A., Best Android apps for Photo Suite, http://appcrawlr.com/android-apps/best-apps-photo-suite, 15.04.2017.

[4] Apps2Apk, Photo Suite 4 pro, http://www.apps2apk.com/android-apps/photosuite-4-pro-4-3-688-apk.html, http://www.mobisystems.com/photosuite/android/help/ 15.04.2017

[5] ART and Dalvik, https://source.android.com/index.html, 11.04.2017.

[6] ArtFlow Studio, Paint Draw Sketchbook, http://artflowstudio.com/, http://www.apkmonk.com/app/com.bytestorm.artflow/, 15.04.2017.

[7] Autodesk, Sketch Book Express, https://www.sketchbook.com/ios, https://sketchbook-express.en.uptodown.com/android, 15.04.2017.

[8]BaryLab, APK4Fun, Sketch Master, https://www.apk4fun.com/apps/com.barilab.katalksketch.googlemarket/ , 15.04.2017.

[9] Corel, Painter Mobile, http://www.painterartist.com/en/product/painter-mobile/, 15.04.2017.

[10] Developer Workflow, http://developer.android.com/, 20.04.2017.

[11] KoPlayer Inc., Silk Paints, http://silk-paints.com/, http://apk.koplayer.com/download-Silk-paints-drawing-for-pc.html, 15.04.2017.

CONTACT ADDRESSES

Pr. Assist. Valentin Velikov,PhD
Department of Informatics and Information Technologies
Faculty of Natural Sciences and Education
Angel Kanchev University of Ruse
8 Studentska Str., 7017 Ruse,Bulgaria
Phone: (++359 82) 888 326,
Cell Phone: (++359) 886 011 544,
E-mail: val @ ami. uni-ruse. bg

Ivelin Mitev, BSc student
Department of Informatics and Information Technologies
Faculty of Natural Sciences and Education
Angel Kanchev University of Ruse
8 Studentska Str., 7017 Ruse,Bulgaria
Cell Phone: (++359) 886462476
E-mail: fc_cska@yahoo.com

ПОДСИСТЕМА ЗА СЪЗДАВАНЕ НА ГРАФИЧЕН ПОТРЕБИТЕЛСКИ ИНТЕРФЕЙС

Валентин Великов, Ивелин Митев

Русенски университет „Ангел Кънчев”

Резюме: *Статията представя графичен редактор за Android – функционално пълно приложение, с интуитивен и елементарен интерфейс. Системата е създадена за демонстрация на някои техники, съвременни технологии и прийоми в предметната област и да ги покаже в процеса на обучение. Представени са архитектурата, основни принципи и основни компоненти на приложението. Показани са идеите за организация на екрана.*

Ключови думи*: информатика, Android, графичен редактор, обучение на Android.*

Requirements and guidelines for the authors - "Proceedings of the Union of Scientists - Ruse"

The Editorial Board of "Proceedings of the Union of Scientists - Ruse" accepts for publication annually both scientific, applied research and methodology papers, as well as announcements, reviews, information materials, adds. No honoraria are paid.

The paper scripts submitted to the Board should answer the following requirements:

1. Papers submitted in Bulgarian, Russian and English are accepted. Their volume should not exceed 8 pages, formatted following the requirements, including reference, tables, figures and abstract.

2. The text should be computer generated (MS Word 97 for Windows or higher versions up to Word 2003) and printed in one copy, possibly on laser printer and on one side of the page. Together with the printed copy the author should submit a disk (or send an e-mail copy to: desi@ami.uni-ruse.bg).

3. Compulsory requirements on formatting:

font - Ariel 12;
paper Size - A4;
page Setup - Top: 20 mm, Bottom: 15 mm, Left: 20 mm, Right: 20mm;
Format/Paragraph/Line spacing - Single;
Format/Paragraph/Special: First Line, By: 1 cm;
Leave a blank line under Header - Font Size 14;
Title should be short, no abbreviations, no formulas or special symbols - Font Size 14, centered, Bold, All Caps;
One blank line - Font Size 14;
Name and surname of author(s) - Font Size: 12, centered, Bold;
One blank line - Font Size 12;
Name of place of work - Font Size: 12, centered;
One blank line;
abstract – no formulas - Font Size 10, Italic, 5-6 lines ;
keywords - Font Size 10, Italic, 1-2 lines;
one blank line;
text - Font Size 12, Justify;
references;
contact address - three names of the author(s) scientific title and degree, place of work, telephone number, Email - in the language of the paper.

4. At the end of the paper the authors should write:

The title of the paper;
Name and surname of the author(s);
abstract; keywords.

Note: If the main text is in Bulgarian or Russian, parts in item 4 should be in English. If the main text is in English, they should be in Bulgarian and have to be formatted as in the beginning of the paper.

5. All mathematical signs and other special symbols should be written clearly and legibly so as to avoid ambiguity when read. All formulas, cited in the text, should be numbered on the right.

6. Figures (black and white), made with some of the widespread software, should be integrated in the text.

7. Tables should have numbers and titles above them, centered right.

8. Reference sources cited in the text should be marked by a number in square brackets.

9. Only titles cited in the text should be included in the references, their numbers put in square brackets. The reference items should be arranged in alphabetical order, using the surname of the first author, and written following the standard. If the main text is in Bulgarian or Russian, the titles in Cyrillic come before those in Latin. If the main text is in English, the titles in Latin come before those in Cyrillic. The paper cited should have: for the first author – surname and first name initial; for the second and other authors – first name initial and surname; title of the paper; name of the publishing source; number of volume (in Arabic figures); year; first and last page number of the paper. For a book cited the following must be marked: author(s) – surname and initials, title, city, publishing house, year of publication.

10. **The author(s) and the reviewer, chosen by the Editorial Board, are responsible for the contents of the materials submitted**.

Important for readers, companies and organizations

1. Authors, who are not members of the Union of Scientists - Ruse, should pay for publishing of materials.

2. Advertising and information materials of group members of the Union of Scientists – Ruse are published free of charge.

3. Advertising and information materials of companies and organizations are charged on negotiable (current) prices.

Editorial Board

ISSN 1314-3077