Министерство цифрового развития, связи и

Массовых коммуникаций Российской Федерации

Ордена Трудового Красного Знамени

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский технический университет связи и информатики»

Кафедра «Математической кибернетики и информационных технологий»

Отчет по лабораторной работе №6

по дисциплине «Кроссплатформенные технологии программирования»

Выполнил: студент группы БСТ2001

Багдасарян Артём

Москва 2022

Оглавление

[**1.** **Цель и задание** 3](#_Toc97797644)

[**2.** **Ход выполнения лабораторной работы** 3](#_Toc97797645)

[**2.1** **Код программы** 3](#_Toc97797646)

[**2.2** **Результат работы программы** 4](#_Toc97797647)

[**3.** **Вывод** 5](#_Toc97797648)

[**Список использованных источников** 6](#_Toc97797649)

1. **Цель и задание**

В данной лабораторной работе мы дополним нашу программу. Добавим поддержку многопоточности.

1. **Ход выполнения лабораторной работы** 
   1. **Код программы**

Ниже представлен код FractalGenerator.java

package com.company;  
  
import java.awt.geom.Rectangle2D;  
  
  
*/\*\*  
 \* This class provides the common interface and operations for fractal  
 \* generators that can be viewed in the Fractal Explorer.  
 \*/*public abstract class FractalGenerator {  
  
 */\*\*  
 \* This static helper function takes an integer coordinate and converts it  
 \* into a double-precision value corresponding to a specific range. It is  
 \* used to convert pixel coordinates into double-precision values for  
 \* computing fractals, etc.  
 \*  
 \** ***@param*** *rangeMin the minimum value of the floating-point range  
 \** ***@param*** *rangeMax the maximum value of the floating-point range  
 \*  
 \** ***@param*** *size the size of the dimension that the pixel coordinate is from.  
 \* For example, this might be the image width, or the image height.  
 \*  
 \** ***@param*** *coord the coordinate to compute the double-precision value for.  
 \* The coordinate should fall in the range [0, size].  
 \*/* public static double getCoord(double rangeMin, double rangeMax,  
 int size, int coord) {  
  
 assert size > 0;  
 assert coord >= 0 && coord < size;  
  
 double range = rangeMax - rangeMin;  
 return rangeMin + (range \* (double) coord / (double) size);  
 }  
  
  
 */\*\*  
 \* Sets the specified rectangle to contain the initial range suitable for  
 \* the fractal being generated.  
 \*/* public abstract void getInitialRange(Rectangle2D.Double range);  
  
  
 */\*\*  
 \* Updates the current range to be centered at the specified coordinates,  
 \* and to be zoomed in or out by the specified scaling factor.  
 \*/* public void recenterAndZoomRange(Rectangle2D.Double range,  
 double centerX, double centerY, double scale) {  
  
 double newWidth = range.width \* scale;  
 double newHeight = range.height \* scale;  
  
 range.x = centerX - newWidth / 2;  
 range.y = centerY - newHeight / 2;  
 range.width = newWidth;  
 range.height = newHeight;  
 }  
  
  
 */\*\*  
 \* Given a coordinate <em>x</em> + <em>iy</em> in the complex plane,  
 \* computes and returns the number of iterations before the fractal  
 \* function escapes the bounding area for that point. A point that  
 \* doesn't escape before the iteration limit is reached is indicated  
 \* with a result of -1.  
 \*/* public abstract int numIterations(double x, double y);  
}

Ниже представлен код FractalExplorer.java

package com.company;  
  
import java.awt.\*;  
import javax.swing.\*;  
import java.awt.geom.Rectangle2D;  
import java.awt.event.\*;  
import javax.swing.JFileChooser.\*;  
import javax.swing.filechooser.\*;  
import javax.imageio.ImageIO.\*;  
import java.awt.image.\*;  
import java.io.File;  
import java.io.FilenameFilter;  
/\*  
Этот класс позволит исследовать  
различные области фрактала, путем его создания, отображения через  
графический интерфейс Swing и обработки событий, вызванных  
взаимодействием приложения с пользователем.  
\*/  
public class FractalExplorer {  
 //Строки  
 private int rowsRemaining;  
 //Целочисленный размер отображения - это ширина и высота отображения в пикселях  
 private int displaySize;  
 /\*  
 Ссылка JImageDisplay, для обновления отображения в разных  
 методах в процессе вычисления фрактала.  
 \*/  
 private JImageDisplay display;  
 /\*  
 Объект FractalGenerator, использующий ссылку на базовый класс для отображения  
 других типов фракталов в будущем.  
 \*/  
 private FractalGenerator fractal;  
 /\*  
 Объект Rectangle2D.Double, указывающий диапазона комплексной  
 плоскости, которая выводится на экран.  
 \*/  
 private Rectangle2D.Double range;  
 private JButton saveButton;  
 private JButton resetButton;  
 private JComboBox myComboBox;  
 /\*  
 Конструктор, который принимает размер дисплея, сохраняет его и  
 инициализирует объекты диапазона и фрактал-генератора.  
 \*/  
 public FractalExplorer(int size) {  
 //Сохраняет размер дисплея  
 displaySize = size;  
 //Инициализирует фрактальный генератор и объекты диапазона  
 fractal = new Mandelbrot();  
 range = new Rectangle2D.Double();  
 fractal.getInitialRange(range);  
 display = new JImageDisplay(displaySize, displaySize);  
 }  
 /\*  
 Этот метод инициализирует графический интерфейс Swing с помощью JFrame, содержащий  
 JImageDisplay объект и кнопку для сброса дисплея.  
 \*/  
 public void createAndShowGUI() {  
 //Устанавливаем фрейм на использование java.awt.BorderLayout для его содержимого  
 display.setLayout(new BorderLayout());  
 JFrame myframe = new JFrame("Fractal Explorer");  
 //Добавляем объект отображения изображения в BorderLayout.CENTER  
 myframe.add(display, BorderLayout.*CENTER*);  
 //Создаем кнопку сброса  
 resetButton = new JButton("Reset");  
 //Экземпляр ResetHandler на кнопке сброса  
 resetButton.addActionListener(new Button());  
 //Экземпляр MouseHandler в компоненте фрактального отображения  
 MouseHandler click = new MouseHandler();  
 display.addMouseListener(click);  
 //Устанавливает операцию закрытия фрейма по умолчанию на "exit"  
 myframe.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 //Установить комбо бокс  
 myComboBox = new JComboBox();  
 //Добавиь каждый объект фрактала в комбо бокс  
 FractalGenerator Mandelbrot = new Mandelbrot();  
 myComboBox.addItem(Mandelbrot);  
 FractalGenerator Tricorn = new Tricorn();  
 myComboBox.addItem(Tricorn);  
 FractalGenerator BurningShip = new BurningShip();  
 myComboBox.addItem(BurningShip);  
 //Экземпляр ButtonHandler в поле со списком  
 Button fractalChooser = new Button();  
 myComboBox.addActionListener(fractalChooser);  
 //Создать новые объекты JPanel,JLabel,JComboBox и добавить панель сверху  
 JPanel myPanel = new JPanel();  
 JLabel myLabel = new JLabel("Fractal:");  
 myPanel.add(myLabel);  
 myPanel.add(myComboBox);  
 myframe.add(myPanel, BorderLayout.*NORTH*);  
 /\*  
 Создать кнопку сохранения, добавить ее в JPanel в BorderLayout.SOUTH  
 вместе с кнопкой сброса.  
 \*/  
 saveButton = new JButton("Save");  
 JPanel myBottomPanel = new JPanel();  
 myBottomPanel.add(saveButton);  
 myBottomPanel.add(resetButton);  
 myframe.add(myBottomPanel, BorderLayout.*SOUTH*);  
 //Экземпляр обработчика сохранения в кнопке сохранения.  
 ActionListener save = new Button();  
 saveButton.addActionListener(save);  
 /\*  
 Разместить содержимое фрейма так, чтобы оно было видно, и  
 запретить изменение размера окна.  
 \*/  
 myframe.setTitle("this is my fractal");  
 Toolkit toolkit = Toolkit.*getDefaultToolkit*();  
 Dimension dimension = toolkit.getScreenSize();  
 myframe.setBounds(dimension.width/2 - 300,dimension.height/2 - 300, 600, 600);  
 myframe.pack();  
 myframe.setVisible(true);  
 myframe.setResizable(false);  
 }  
 /\*  
 Вспомогательный метод для отображения фрактала. Этот метод  
 должен циклически проходить через каждый пиксель в отображении (т.е.  
 значения x и y будут меняться от 0 до размера отображения)  
 и вычислять количество итераций для соответствующих координат в области  
 отображения фракталов.  
 \*/  
 private void drawFractal()  
 {  
 /\*  
 Отключение всех элементов  
 пользовательского интерфейса во время рисования  
 \*/  
 enableUI(false);  
 /\*  
 Устанавливает значение «rows remaining» равное общему количеству строк,  
 которые нужно нарисовать  
 \*/  
 rowsRemaining = displaySize;  
 /\*  
 Вызываем Fractal Worker на каждой строке дисплея (вычисление значений  
 цвета на каждой строке получается)  
 \*/  
 for (int x=0; x<displaySize; x++){  
 FractalWorker drawRow = new FractalWorker(x);  
 drawRow.execute();  
 }  
 }  
 /\*  
 Включает или отключает кнопки интерфейса  
 и поле со списком в зависимости от заданного значения  
 \*/  
 private void enableUI(boolean val) {  
 myComboBox.setEnabled(val);  
 resetButton.setEnabled(val);  
 saveButton.setEnabled(val);  
 }  
 //Внутренний класс для обработки событий java.awt.event.ActionListener  
 private class Button implements ActionListener {  
 //Проверка выборки источника  
 public void actionPerformed(ActionEvent e) {  
 //Получить источник действия  
 String command = e.getActionCommand();  
 /\*  
 Если источником является комбо бокс, получить фрактал, выбранный пользователем,  
 и отобразить его.  
 \*/  
 if (e.getSource() instanceof JComboBox) {  
 JComboBox mySource = (JComboBox) e.getSource();  
 fractal = (FractalGenerator) mySource.getSelectedItem();  
 fractal.getInitialRange(range);  
 drawFractal();  
 }  
 //Если источником является кнопка сохранения, сохранить изображение текущего фрактала  
 else if (command.equals("Save")) {  
 //Разрешить пользователю выбрать файл для сохранения изображения  
 JFileChooser myFileChooser = new JFileChooser();  
 //Сохранять только изображения PNG  
 FileFilter extensionFilter = new FileNameExtensionFilter("PNG Images", "png");  
 myFileChooser.setFileFilter(extensionFilter);  
 //Гарантирует, что средство выбора файлов не разрешит не - ". Png" имена файлов  
 myFileChooser.setAcceptAllFileFilterUsed(false);  
 /\*  
 Всплывает окно «Сохранить файл», в котором пользователь может выбрать  
 каталог и файл для сохранения.  
 \*/  
 int userSelection = myFileChooser.showSaveDialog(display);  
 /\*  
 Если результат операции выбора файла  
 APPROVE\_OPTION, продолжить операцию сохранения файла.  
 \*/  
 if (userSelection == JFileChooser.*APPROVE\_OPTION*) {  
 //Получить файл и имя файла  
 java.io.File file = myFileChooser.getSelectedFile();  
 String file\_name = file.toString();  
 //Сохранить фрактальное изображение на диск  
 try {  
 BufferedImage displayImage = display.getImage();  
 javax.imageio.ImageIO.*write*(displayImage, "png", file);  
 }  
 //Исключения и сообщения об исключениях  
 catch (Exception exception) {  
 JOptionPane.*showMessageDialog*(display, exception.getMessage(),  
 "Cannot Save Image", JOptionPane.*ERROR\_MESSAGE*);  
 }  
 }  
 }  
 //Если источником является кнопка сброса, сбросить дисплей и перерисовать фрактал  
 else if (command.equals("Reset")) {  
 fractal.getInitialRange(range);  
 drawFractal();  
 }  
 else return;  
 }  
 }  
 //Внутренний класс для обработки событий MouseListener с дисплея  
 private class MouseHandler extends MouseAdapter  
 {  
 /\*  
 При получении события о щелчке мышью, класс должен  
 отобразить пиксельные кооринаты щелчка в область фрактала, а затем вызвать  
 метод генератора recenterAndZoomRange() с координатами, по которым  
 щелкнули, и масштабом 0.5. Таким образом, нажимая на какое-либо место на  
 фрактальном отображении, вы увеличиваете его!  
 \*/  
 //@Override //переопределяем метод базового класса  
 public void mouseClicked(MouseEvent e)  
 {  
 //Возвращает значение, если оставшиеся строки ненулевые.  
 if (rowsRemaining != 0) {  
 return;  
 }  
 //Получить координату x области отображения щелчка мыши  
 int x = e.getX();  
 double xCoord = fractal.*getCoord*(range.x,range.x + range.width, displaySize, x);  
 //Получить координату y области отображения щелчка мышью  
 int y = e.getY();  
 double yCoord = fractal.*getCoord*(range.y,range.y + range.height, displaySize, y);  
 /\*  
 Вызов метода генератора RecenterAndZoomRange () с  
 координатами, по которым щелкнули, и масштаблм 0,5.  
 \*/  
 fractal.recenterAndZoomRange(range, xCoord, yCoord, 0.5);  
 //Перерисовываем фрактал после изменения области отображения  
 drawFractal();  
 }  
 }  
 /\*  
 Класс для вычисления значений цвета  
 для одной строки фрактала  
 \*/  
 private class FractalWorker extends SwingWorker<Object, Object>  
 {  
 // целочисленная y-координата вычисляемой строки  
 int yCoordinate;  
 //массив чисел типа int для хранения  
 //вычисленных значений RGB для каждого пикселя в этой строке  
 int[] arrayRGB;  
 //Конструктор принимает координату y  
 //в качестве аргумента и сохраняет ее  
 private FractalWorker(int row) {  
 yCoordinate = row;  
 }  
 protected Object doInBackground() {  
 arrayRGB = new int[displaySize];  
 //Цикл сохраняет каждое значение RGB в соответствующем элементе  
 //целочисленного массива  
 for (int i = 0; i < arrayRGB.length; i++) {  
 //Нахождение координат  
 double xCoord = fractal.*getCoord*(range.x,  
 range.x + range.width, displaySize, i);  
 double yCoord = fractal.*getCoord*(range.y,  
 range.y + range.height, displaySize, yCoordinate);  
 // Вычисление количества итераций для координат  
 int iteration = fractal.numIterations(xCoord, yCoord);  
 //Если -1, то значения RGB int в черный цвет  
 if (iteration == -1){  
 arrayRGB[i] = 0;  
 }  
 else {  
 // В противном случае выберите  
 // значение оттенка на основе числа итераций  
 float hue = 0.7f + (float) iteration / 200f;  
 int rgbColor = Color.*HSBtoRGB*(hue, 1f, 1f);  
 //Обновление массива  
 arrayRGB[i] = rgbColor;  
 }  
 }  
 return null;  
 }  
 /\*  
 Вызывается после завершения фоновой задачи. Рисует пиксели  
 для текущей строки и обновляет отображение  
 \*/  
 protected void done() {  
 //Рисование в пикселях,  
 // которые были вычислены в doInBackground ()  
 for (int i = 0; i < arrayRGB.length; i++) {  
 display.drawPixel(i, yCoordinate, arrayRGB[i]);  
 }  
 //метод перерисовки (рекомендуется для сложных анимаций и компонентов)  
 display.repaint(0, 0, yCoordinate, displaySize, 1);  
 //Уменьшение значения «rows remaining» на 1, как  
 //последний шаг данной операции. Затем, если после уменьшения значение «rows  
 //remaining» равно 0, вызов метода enableUI (true).  
 rowsRemaining--;  
 if (rowsRemaining == 0) {  
 enableUI(true);  
 }  
 }  
 }  
 /\*  
 Инициализирует новый экземпляр класса FractalExplorer с  
 размером отображения 800. Вызовает метод createAndShowGUI () класса FractalExplorer.  
 Вызовает метод drawFractal() класса FractalExplorer для отображения начального представления.  
 \*/  
 public static void main(String[] args) {  
 FractalExplorer displayExplorer = new FractalExplorer(700);  
 displayExplorer.createAndShowGUI();  
 displayExplorer.drawFractal();  
 }

Ниже представлен код JImageDisplay.java

package com.company;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.image.BufferedImage;  
  
public class JImageDisplay extends JComponent {  
  
 private BufferedImage displayImage;  
  
 public JImageDisplay(int height, int width){  
 displayImage = new BufferedImage(width, height, BufferedImage.*TYPE\_INT\_RGB*);  
 Dimension imageDimension = new Dimension(width, height);  
 super.setPreferredSize(imageDimension);  
 }  
  
 @Override  
 public void paintComponent(Graphics g){  
 super.paintComponent(g);  
 g.drawImage (displayImage, 0, 0, displayImage.getWidth(), displayImage.getHeight(), null);  
 }  
  
 public void clearImage(){  
 int[] blankArray = new int[getWidth() \* getHeight()];  
 displayImage.setRGB(0, 0, getWidth(), getHeight(), blankArray, 0, 1);  
 }  
  
 public void drawPixel(int x, int y, int rgbColor)  
 {  
 displayImage.setRGB(x, y, rgbColor);  
 }  
  
 public BufferedImage getImage() {  
 return displayImage;  
 }  
}

Ниже представлен код Mandelbort.java

package com.company;  
  
import java.awt.geom.Rectangle2D;  
  
public class Mandelbrot extends FractalGenerator{  
 public static final int *MAX\_ITERATIONS* = 2000;  
  
 public void getInitialRange(Rectangle2D.Double range){  
  
 range.x = -2;  
 range.y = -1.5;  
 range.width = 3;  
 range.height = 3;  
 }  
  
 public int numIterations(double x, double y){  
 int iteration = 0;  
 double zreal = 0;  
 double zimaginary = 0;  
 while (iteration<=*MAX\_ITERATIONS* && zreal \* zreal + zimaginary \* zimaginary < 4)  
 {  
 double zrealUpdated = zreal \* zreal - zimaginary \* zimaginary + x;  
 double zimaginaryUpdated = 2 \* zreal \* zimaginary + y;  
 zreal = zrealUpdated;  
 zimaginary = zimaginaryUpdated;  
 iteration += 1;  
 }  
 if (iteration == *MAX\_ITERATIONS*)  
 {  
 return -1;  
 }  
 return iteration;  
 }  
}

Ниже представлен код BurningShip.java

package com.company;  
  
import java.awt.geom.Rectangle2D;  
  
public class BurningShip extends FractalGenerator  
{  
 public static final int *MAX\_ITERATIONS* = 2000;  
  
 public void getInitialRange(Rectangle2D.Double range)  
 {  
 range.x = -2;  
 range.y = -2.5;  
 range.width = 4;  
 range.height = 4;  
 }  
  
 public int numIterations(double x, double y)  
 {  
  
 int iteration = 0;  
  
 double zreal = 0;  
 double zimaginary = 0;  
  
 while (iteration < *MAX\_ITERATIONS* &&  
 zreal \* zreal + zimaginary \* zimaginary < 4)  
 {  
 double zrealUpdated = zreal \* zreal - zimaginary \* zimaginary + x;  
 double zimaginaryUpdated = 2 \* Math.*abs*(zreal) \* Math.*abs*(zimaginary) + y;  
 zreal = zrealUpdated;  
 zimaginary = zimaginaryUpdated;  
 iteration += 1;  
 }  
  
 if (iteration == *MAX\_ITERATIONS*)  
 {  
 return -1;  
 }  
 return iteration;  
 }  
  
 public String toString() {  
 return "Burning Ship";  
 }  
}

Ниже представлен код tricorn.java

package com.company;  
  
import java.awt.geom.Rectangle2D;  
//Этот класс является подклассом FractalGenerator. Он используется для вычисления Фрактал Мандельброта  
public class Tricorn extends FractalGenerator  
{  
 public static final int *MAX\_ITERATIONS* = 2000;  
 /\*  
 метод позволяет генератору  
 фракталов определить наиболее «интересную» область комплексной плоскости  
 для конкретного фрактала. Методу в качестве  
 аргумента передается прямоугольный объект, и метод должен изменить поля  
 прямоугольника для отображения правильного начального диапазона для  
 фрактала. Этот метод  
 должен установить начальный диапазон в (-2 - 1.5i) - (1 + 1.5i). Т.е. значения x  
 и y будут равны -2 и -1.5 соответственно, а ширина и высота будут равны 3.  
 \*/  
 public void getInitialRange(Rectangle2D.Double range)  
 {  
 range.x = -2;  
 range.y = -2;  
 range.width = 4;  
 range.height = 4;  
 }  
 /\*  
 Этот метод реализует итерационную функцию для фрактала Мандельброта.  
 Требуется два дабла для действительной и мнимой частей комплекса.  
 Вычисляется количество итераций для соответствующей координаты.  
 \*/  
 public int numIterations(double x, double y)  
 {  
 //Начать итерацию с нуля  
 int iteration = 0;  
 //Инициализация zreal и zimaginary  
 double zreal = 0;  
 double zimaginary = 0;  
 /\*  
 Вычислить Zn = Zn-1 ^ 2 + c, где значения представляют собой комплексные числа, представленные  
 по zreal и zimaginary, Z0 = 0, а c - особая точка в  
 фрактале, который мы отображаем (заданный x и y). Это повторяется  
 до Z ^ 2> 4 (абсолютное значение Z больше 2) или максимум  
 достигнуто количество итераций.  
 \*/  
 while (iteration < *MAX\_ITERATIONS* &&  
 zreal \* zreal + zimaginary \* zimaginary < 4)  
 {  
 double zrealUpdated = zreal \* zreal - zimaginary \* zimaginary + x;  
 double zimaginaryUpdated = -2 \* zreal \* zimaginary + y;  
 zreal = zrealUpdated;  
 zimaginary = zimaginaryUpdated;  
 iteration += 1;  
 }  
 /\*  
 В случае, если алгоритм дошел до значения MAX\_ITERATIONS нужно  
 вернуть -1, чтобы показать, что точка не выходит за границы.  
 \*/  
 if (iteration == *MAX\_ITERATIONS*)  
 {  
 return -1;  
 }  
 return iteration;  
 }  
 //Возврат названия фрактала  
 public String toString() {  
 return "Tricorn";  
 }

* 1. **Результат работы программы**

Результат работы программы показан на рисунке 1

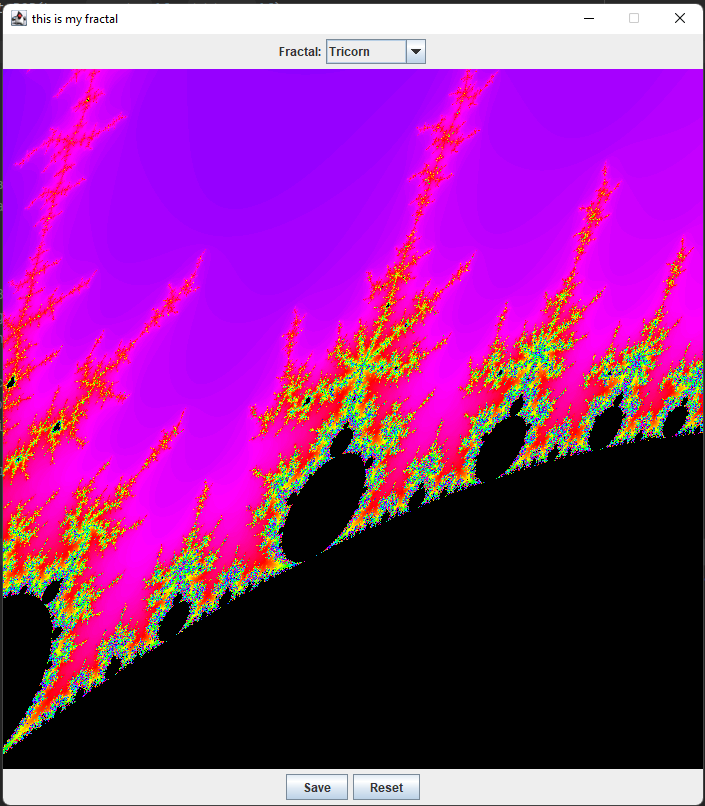


Рисунок 1 – Результат работы программы

* 1. **Программа на Git hub**

Ссылка на репозиторий на Git hub: https://github.com/Artem2406/KTP

1. **Вывод**

В данной лабораторной работе мы дополнили нашу программу. Добавили поддержку многопоточности.

# **Список использованных источников**

1. Камаев В.А., Костерин В.В. Технологии программирования. М.: Высшая школа, 2006.

2. Жоголев Е.А.Технология программирования. – М.: Научный мир, 2004.