ГОУ ВПО

Уфимский государственный авиационный технический университет

Кафедра вычислительной математики и кибернетики

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** |
| **100** |  |  |  |  |  |  |  |  |  |  |  |  |
| **90** |  |  |  |  |  |  |  |  |  |  |  |  |
| **80** |  |  |  |  |  |  |  |  |  |  |  |  |
| **70** |  |  |  |  |  |  |  |  |  |  |  |  |
| **60** |  |  |  |  |  |  |  |  |  |  |  |  |
| **50** |  |  |  |  |  |  |  |  |  |  |  |  |
| **40** |  |  |  |  |  |  |  |  |  |  |  |  |
| **30** |  |  |  |  |  |  |  |  |  |  |  |  |
| **20** |  |  |  |  |  |  |  |  |  |  |  |  |
| **10** |  |  |  |  |  |  |  |  |  |  |  |  |

***Разработка алгоритма***

**«****Каркасная модель объекта»**

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовой работе**

**по дисциплине *«Интерактивные графические системы»***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Группа | Ф.И.О. | Подпись | Дата | Оценка |
| Студент | Доронин С.Г. |  |  |  |
| Консультант |  |  |  |  |
| Принял |  |  |  |  |

Уфа - 2014 г.

Федеральное агентство по образованию

Государственное образовательное учреждение высшего профессионального образования

«УФИМСКИЙ ГОСУДАРСТВЕННЫЙ АВИАЦИОННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра вычислительной математики и кибернетики

# ЗАДАНИЕ

На курсовую работу по дисциплине «Вычислительная математика»

Студент \_\_Доронин С.Г. Группа \_ПРО-301в Консультант\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ф.И.О номер акад. гр. Ф.И.О. .

1. Тема курсовой работы

Каркасная модель объекта

|  |
| --- |
| наименование темы |
|  |

|  |  |
| --- | --- |
| 1. Основное содержание: | Разбор работы алгоритма «Каркасная модель объекта» |
| на примере и проверка на программе, написанной для данной курсовой работы. | |
|  | |

1. Требования к структуре и оформлению курсовой работы

3.1. Курсовая работа должна содержать следующие разделы:

Введение

Глава 1. Постановка задачи

Глава 2. Теория по алгоритму

Глава 3. Задача с решением

Глава 4. Написание алгоритма на Scala

Глава 5. Обзор работы программы

Глава 6. Текст программы

Заключение

Список использованной литературы

1. Источники информации

Книги, учебники, учебные пособия, материалы сети Интернет.

На все источники, приведенные в списке литературы, должны быть ссылки по тексту курсовой работы.

Дата выдачи \_\_\_\_\_\_\_\_\_\_\_\_ Дата окончания \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Валеев Р.С.

Подпись

# Содержание

[Введение 4](#_Toc389130008)

[1 Постановка задачи 5](#_Toc389130009)

[2 Теория по алгоритму 6](#_Toc389130010)

[3 Задача с решением 7](#_Toc389130011)

[4 Написание алгоритма на Scala 8](#_Toc389130012)

[5 Обзор работы программы 9](#_Toc389130013)

[6 Текст программы 13](#_Toc389130014)

[Класс главной формы 13](#_Toc389130015)

[Класс фигуры Кресто-Куб 14](#_Toc389130016)

[Класс ComplexShape 15](#_Toc389130017)

[Класс RotateShape 15](#_Toc389130018)

[Заключение 16](#_Toc389130019)

[Список литературы 17](#_Toc389130020)

# Введение

Интерактивными графическими системами [1] будем называть системы, в которых проектирование графических объектов (ГО) осуществляется в процессе непосредственного взаимодействия человека и вычислительной системы. При этом ИГС позволяют создавать, модифицировать и редактировать графические объекты. На экране компьютера отображаются проектируемые графические объекты, являющиеся результатом работы человеко-машинной системы.

Пользователь интерактивных графических систем должен иметь в своем распоряжении некоторое множество команд, с помощью которых можно было бы управлять ходом выполнения программы. В этих командах содержатся две части: указание на необходимые действия и данные, подлежащие обработке.

В интерактивных графических системах - шаблон определенного цвета, фактуры и формы, используемый для рисования или закрашивания.

# 1 Постановка задачи

* Изучить теорию создания каркасной модели объекта;
* разобрать алгоритм для создания каркасной модели объекта;
* написать программу на языке Scala, которая бы при заданных пользователем параметрах, создавала каркасную модель объекта.

# 2 Теория по алгоритму

Каркасная модель [2] — модель объекта в трёхмерной графике, представляющая собой совокупность вершин и рёбер, которая определяет форму отображаемого многогранного объекта.

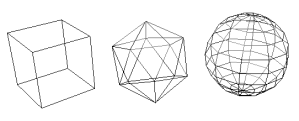


Рисунок . Примеры простейших каркасных сеток

# 3 Задача с решением

Дана фигура (Рисунок 2), построить ее каркасную модель с возможностью вращения по осям: X, Y, Z.



Рисунок . Фигура – Крестокуб

# 4 Написание алгоритма на Scala

Построение фигуры будет осуществляется разбиением ее на более простые части – кресты, кресты в свою очередь будут разбиты еще на более простые части – кубы, кубы на квадраты [3][4].

Для построение фигуры были созданы классы:

* RotateShape – абстрактная фигура, которую можно поворачивать по осям;
* Cube (наследник класса RotateShape) – куб, строится из 4-х прямоугольников;
* ComplexShape (наследник класса RotateShape) – сложная абстрактная фигура (состоящая из других фигур);
* Cross (наследник класса ComplexShape) – крест, строится из 5-ти кубов;
* CrossCube (наследник класса ComplexShape) – крестокуб, строится из 5 крестов.

# 5 Обзор работы программы

Программа запускается с файла «**run.bat**» (необходима установленная JRE 1.7).

После запуска откроется главная форма (Рисунок 3).

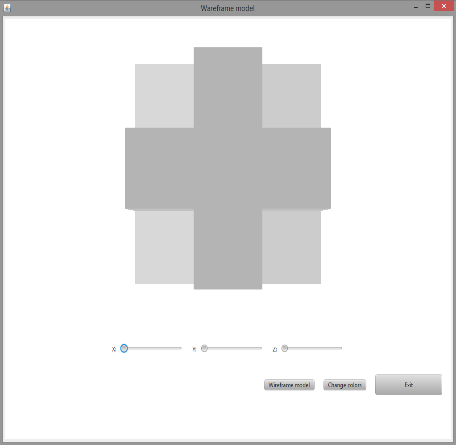


Рисунок . Главная форма

Для построения каркасной модели, нужно нажать на кнопку «**Wireframe model**», результат отобразится на форме (Рисунок 4).

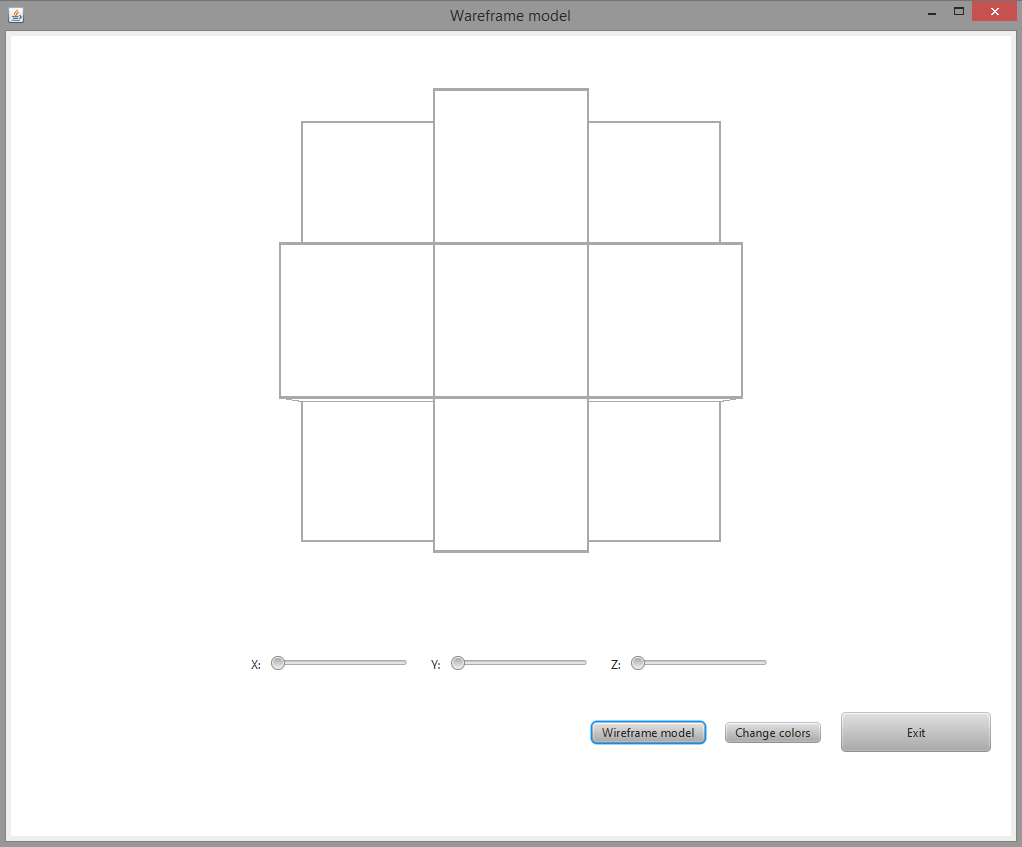


Рисунок . Каркасная модель объекта

Для вращения модели, используются ползунки расположенные ниже, у каждого ползунка есть подпись оси вращения (Рисунок 5).

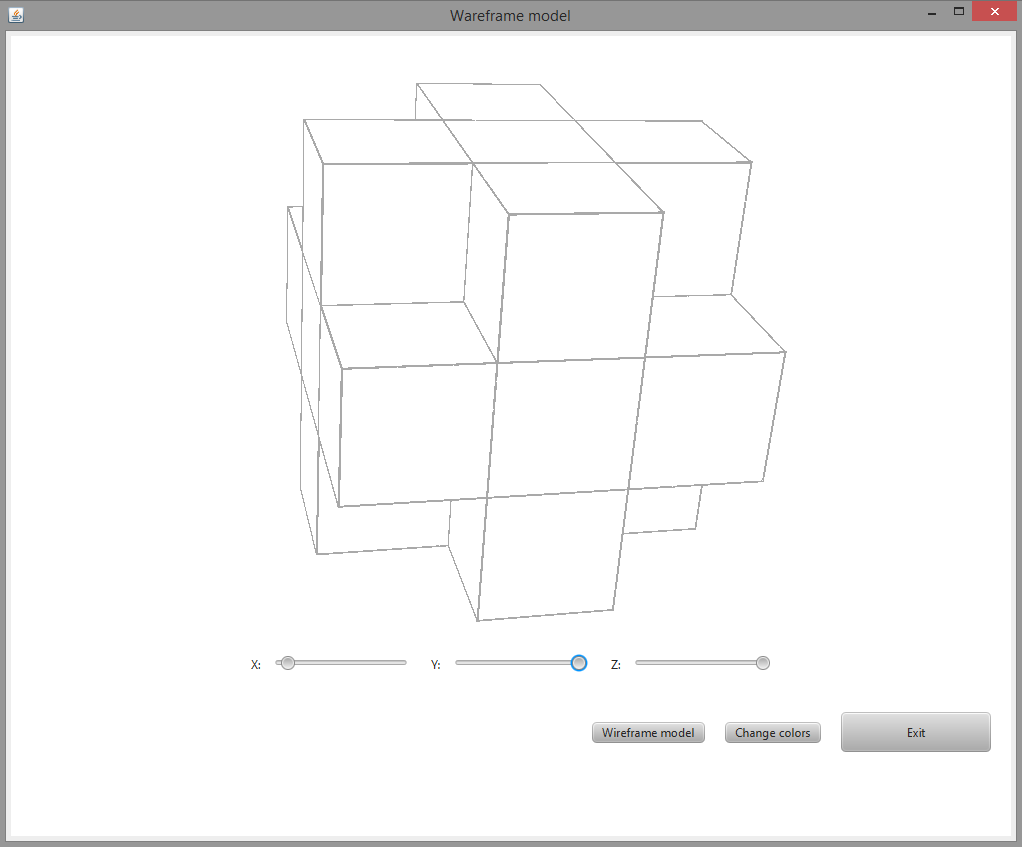


Рисунок . Вращение модели

Для покраски фигуры нужно нажать на кнопку «**Change colors**». Все кубы из которых состоит фигура, окрасятся в разные цвета (Рисунок 6).

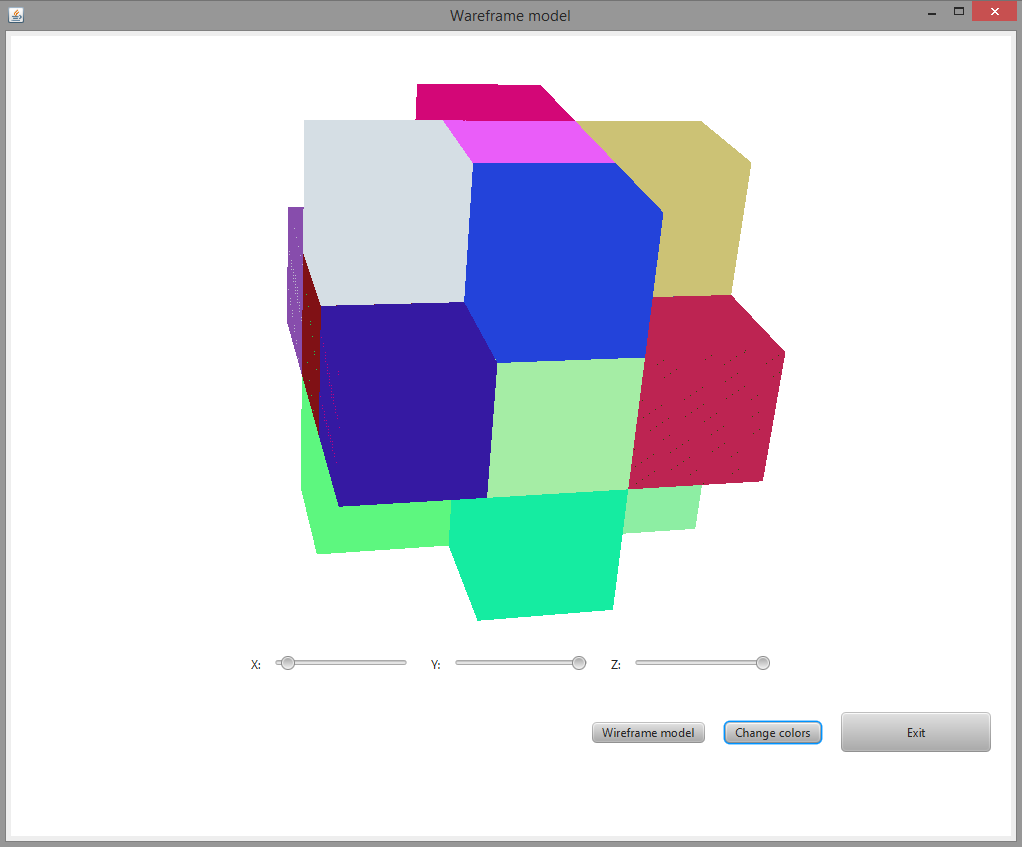


Рисунок . Покраска фигуры

Для выхода из программы нужно нажать на кнопку «**Exit**» или на крестик в правом верхнем углу экрана.

# 6 Текст программы

## Класс главной формы

/\*\*

\* Каркасная модель

\*/

**object** Main **extends** SimpleSwingApplication {

// Размеры формы

**private** **val** FormWidth **=** 800

**private** **val** FormHeight **=** 600

// Размеры модели

**private** **val** ModelWidth **=** FormWidth

**private** **val** ModelHeight **=** FormHeight

/\*\* Модель поверхности \*/

**private** **val** SizeModel **=** 300

**private** **val** ColorNum **=** 240

**private** **val** ColorModel **=** **if** (false) **null** **else** Color.rgb(ColorNum, ColorNum, ColorNum)

**private** **val** ShadeModel **=** 0.5

**private** **val** startPoint **=** **new** Point3D(**-**SizeModel/2, **-**SizeModel/2, **-**SizeModel/2)

**private** **val** objectModel **=** **new** CrossCube(startPoint, SizeModel, ColorModel, ShadeModel)

**private** **val** jfx **=** **new** JFXPanel

/\*\* Непрозрачность \*\*/

**private** **val** OpaqueModel **=** **true**

/\*\* Создание главной формы \*/

**def** top **=** **new** MainFrame {

title **=** "Wareframe model"

contents **=** **new** FlowPanel {

peer.add(jfx)

jfx.setPreferredSize(new Dimension(FormWidth, FormHeight))

}

centerOnScreen()

}

/\*\* Отложенный запуск создания Java FX компонентов \*/

Platform.runLater(new Runnable {

**override** **def** run() **=** createJavaFxComponents()

})

/\*\* Создание Java FX компонентов \*/

**private** **def** createJavaFxComponents() {

**val** root **=** **new** VBox(20)

**val** scene **=** **new** Scene(root, ModelWidth, ModelHeight, OpaqueModel)

scene.setCamera(new PerspectiveCamera)

jfx.setScene(scene)

// Модель объекта

**val** objectModelPanel **=** getObjectModelPanel(ModelWidth, ModelHeight)

root.getChildren.add(objectModelPanel)

// Слайдеры

**val** slidersBox **=** **new** HBox(20)

root.getChildren.add(slidersBox)

slidersBox.setAlignment(Pos.CENTER)

**val** sliderBoxX **=** createSliderBox(AxisTitle.X)

slidersBox.getChildren.add(sliderBoxX)

**val** sliderBoxY **=** createSliderBox(AxisTitle.Y)

slidersBox.getChildren.add(sliderBoxY)

**val** sliderBoxZ **=** createSliderBox(AxisTitle.Z)

slidersBox.getChildren.add(sliderBoxZ)

// Кнопки

**val** buttosnBox **=** **new** HBox(20)

root.getChildren.add(buttosnBox)

buttosnBox.setAlignment(Pos.CENTER\_RIGHT)

buttosnBox.setPadding(new Insets(20, 20, 20, 20))

// Построение каркасной модели

**val** wireframeModelButton **=** **new** Button("Wireframe model")

buttosnBox.getChildren.add(wireframeModelButton)

wireframeModelButton.setOnAction(new EventHandler[ActionEvent] {

**override** **def** handle(event: ActionEvent) **=** objectModel.wireframeModel()

})

// Смена цвета объекта

**val** changeColorsButton **=** **new** Button("Change colors")

buttosnBox.getChildren.add(changeColorsButton)

changeColorsButton.setOnAction(new EventHandler[ActionEvent] {

**override** **def** handle(event: ActionEvent) **=** objectModel.changeColors()

})

// Выход из программы

**val** exitButton **=** **new** Button("Exit")

buttosnBox.getChildren.add(exitButton)

exitButton.setPrefSize(150, 40)

exitButton.setOnAction(new EventHandler[ActionEvent] {

**override** **def** handle(event: ActionEvent) **=** System.exit(0)

})

}

/\*\* Создание панели с заголовком и ползунком \*/

**private** **def** createSliderBox(title: String) **=** {

// Slider

**val** minValue **=** 0

**val** maxValue **=** 360

**val** startValue **=** 0

**val** slider **=** **new** Slider(minValue, maxValue, startValue)

// Поворот объекта по оси при изменении положения ползунка

slider.valueProperty.addListener(new ChangeListener[Number]() {

**override** **def** changed(observableValue: ObservableValue[\_ **<**: Number], oldValue: Number, newValue: Number) {

**val** rotate **=** title **match** {

**case** AxisTitle.X **=>** objectModel.rx

**case** AxisTitle.Y **=>** objectModel.ry

**case** AxisTitle.Z **=>** objectModel.rz

}

rotate.setAngle(newValue.doubleValue)

}

})

// Label

**val** label **=** **new** Label(title)

// Box

**val** box **=** **new** HBox(10)

box.getChildren.addAll(label, slider)

box

}

/\*\* Панель с каркасной моделью \*/

**private** **def** getObjectModelPanel(width: Int, height: Int) **=** {

// Установка повортов по всем осям в 0

objectModel.rx.setAngle(0)

objectModel.ry.setAngle(0)

objectModel.rz.setAngle(0)

**val** stackPane **=** **new** StackPane()

stackPane.setPrefSize(width, height)

stackPane.getChildren.add(objectModel)

stackPane

}

}

## Класс фигуры Крестокуб

/\*\*

\* Кресто-Куб

\*/

**class** CrossCube(startPoint: Point3D, size: Double, color: Color, shade: Double) **extends** ComplexShape {

/\*\* Размер частей \*/

**private** **val** partSize **=** size

/\*\* Размер промежутков между частями \*/

**private** **val** indentSize **=** partSize / 3

// Отступы

**private** **val** sx **=** startPoint.getX

**private** **val** sy **=** startPoint.getY

**private** **val** sz **=** startPoint.getZ

init()

**def** init() {

/\*\* Построение креста с параметрами по умолчанию \*/

**val** newCross **=** (x: Double, y: Double, z: Double) **=>** {

**val** sPoint **=** **new** Point3D(x, y, z)

**new** Cross(sPoint, partSize, color, shade)

}

**val** frontCross **=** newCross(sx, sy, sz)

**val** middle **=** newCross(sx, sy, sz **+** indentSize)

**val** backCross **=** newCross(sx, sy, sz **+** indentSize \* 2)

**val** rightCross **=** newCross(**-**sz **+** indentSize \* **-**3, sy, sx **+** indentSize \* 2)

rightCross.ry.setAngle(90)

**val** leftCross **=** newCross(sz, sy, **-**sx **-** indentSize)

leftCross.ry.setAngle(**-**90)

partList **=** ListBuffer(middle, backCross, rightCross, leftCross, frontCross)

getChildren.addAll(JavaConversions.bufferAsJavaList(partList))

/\*\* Точка вращения \*/

**val** pivot **=** **new** Cube(new Point3D(0, 0, 0), 5, **new** Color(1, 0, 0, 1), shade)

getChildren.add(pivot)

}

}

## Класс ComplexShape

/\*\*

\* Сложная фигура - состоящая из других фигур

\*/

**abstract** **class** ComplexShape **extends** RotateShape {

/\*\* Части из которых состоит фигура \*/

**protected** **var** partList: ListBuffer[RotateShape] **=** **null**

/\*\* Построение каркасной модели \*/

**override** **def** wireframeModel() **=** partList.foreach(\_.wireframeModel())

/\*\* Изменение цвета \*/

**override** **def** changeColors() **=** partList.foreach(\_.changeColors())

}

## Класс RotateShape

/\*\*

\* Фигура, которую можно поворачивать по 3-м осям

\*/

**abstract** **class** RotateShape **extends** Group {

// Оси поворота объекта

**val** rx **=** **new** Rotate(0,Rotate.X\_AXIS)

**val** ry **=** **new** Rotate(0,Rotate.Y\_AXIS)

**val** rz **=** **new** Rotate(0,Rotate.Z\_AXIS)

getTransforms.addAll(rz, ry, rx)

**def** changeColors()

// Каркасная модель

/\*\* Цвет пустоты \*/

**protected** **val** VoidsColor **=** Color.WHITE

/\*\* Цвет каркаса \*/

**protected** **val** FrameColor **=** Color.DARKGRAY

/\*\* Построение каркасной модели \*/

**def** wireframeModel()

# Заключение

В соответствии с поставленной задачей в курсовой работе было выполнено следующее:

1. Изучен конкретный раздел интерактивных графических систем;
2. разобран алгоритм построения каркасной модели объекта;
3. разработана программа на Scala, реализующая данный алгоритм.

# Список литературы

1. www.ngpedia.ru [Электронный ресурс] // Большая Энциклопедия Нефти Газа - <http://www.ngpedia.ru/id430423p1.html> Режим доступа – свободный.
2. ru.wikipedia.org [Электронный ресурс] // Википедия – Свободная энциклопедия - <http://ru.wikipedia.org/w/index.php?oldid=56649411> Режим доступа – свободный.
3. ru.wikipedia.org [Электронный ресурс] // Википедия – Свободная энциклопедия - <http://ru.wikipedia.org/wiki/JavaFX> Режим доступа – свободный.
4. docs.oracle.com [Электронный ресурс] // Документация к JavaFX - <http://docs.oracle.com/javase/8/javase-clienttechnologies.htm> Режим доступа – свободный.