https://doi.org/10.26160/2474-5901-2020-22-79-89

РАЗРАБОТКА КРОССПЛАТФОРМЕННОГО ПРИЛОЖЕНИЯ С ПОМОЩЬЮ БИБЛИОТЕКИ QT НА ПРИМЕРЕ ГРАФИЧЕСКОГО ВЕКТОРНОГО РЕДАКТОРА

Баланков Н.В., Букунов С.В.

Ключевые слова: программирование, векторная графика, кроссплатформенное приложение, язык программирования C++, библиотека Qt.

Аннотация. В статье с помощью языка C++ и библиотеки Qt разработано кроссплатформенное приложение, которое представляет собой графический редактор, позволяющий создавать, обрабатывать и сохранять двумерные векторные изображения в формате SVG (Scalable Vector Graphics). Описаны графический интерфейс пользователя, основные классы и основные функциональные возможности приложения. Разработанное приложение может корректно функционировать в операционных системах Windows, MacOS, Linux.

DEVELOPING A CROSS-PLATFORM APPLICATION USING QT LIBRARY ON THE EXAMPLE OF A GRAPHICAL VECTOR EDITOR

Balankov N.V., Bukunov S.V.

Keywords: programming, vector graphics, cross-platform application, C ++ programming language, Qt library.

Abstract. A cross-platform application using the C ++ language and the Qt library, which is a graphical editor that allows you to create, process and save two-dimensional vector images in the SVG format (Scalable Vector Graphics) is considered in this article. The graphical user interface, the main classes and the main functionality of the application have described. The developed application can be correctly used in Windows, MacOS, Linux operating systems.

Введение

В настоящее время компьютерная графика является одним из самых быстроразвивающихся направлений в сфере информационных технологий [1]. Для создания и обработки графических изображений каждый год разрабатывается или улучшается большое количество как узкоспециализированных (Paint), так и масштабных приложений (Photoshop).

Компьютерная графика широко используется в различных сферах деятельности человека, например:

– в строительстве – для выполнения различных чертежей, схем, планов зданий (2D графика), для информационного моделирования зданий (3D визуализация зданий, BIM-технологии) [2];

– в математическом моделировании сложных поверхностей [3];

– в маркетинге и рекламе – для производства различных баннеров, плакатов и прочих видов рекламной продукции;

– в сфере мультимедиа – для создания компьютерных игр, анимации;

– в медицине – для визуализации внутренних органов человека, автоматизированного проектирования имплантатов суставов и костей.

Графические изображения создаются с помощью графических редакторов, среди которых можно выделить следующие редакторы [1]:

– редакторы растровой графики;

– редакторы векторной графики;
– редакторы фрактальной графики;
– редакторы трехмерной графики.

На рынке приложений для компьютерной графики предлагается достаточно большое количество различных графических редакторов. Однако подавляющее большинство из них относятся к категории растровых нативных приложений [4]. В современном же мире, где все большую популярность набирают кроссплатформенность (способность одного приложения с одним набором кода запускаться на разных операционных системах) [4, 5] и векторная графика, возникает нехватка решений, которые бы удовлетворяли этим двум критериям.

Цель данной работы заключается в разработке кроссплатформенного графического векторного редактора, который бы отвечал всем современным стандартам и паттернам проектирования.

Среда разработки

Огромный процент успешного создания приложения заключается в правильном выборе языка программирования и среды разработки.

К наиболее популярным современным средствам разработки кроссплатформенного программного обеспечения можно отнести:

JavaFX – кроссплатформенный фреймворк, разработанный компанией Oracle и предоставляющий огромный спектр инструментов разработки и создания графического интерфейса пользователя (Graphic User Interface, GUI) на языке Java [6];

Xamarin − кроссплатформенный фреймворк для разработки приложений с помощью языка программирования C# [7];

Qt − кроссплатформенный фреймворк, который предоставляет не только широкий спектр библиотек классов, но и шаблоны разработки приложений на языке C++ [8, 9].

В данной работе для разработки приложения был выбран кроссплатформенный фреймворк Qt. Данный фреймворк в связке с высокопроизводительным языком C++ дает великолепные возможности в сфере графики и создания интерфейса пользователя с помощью декларативного языка QML [10, 11]. В Qt есть огромный набор классов (более 700), которые специализируются на 2D и 3D визуализации [12].

К дополнительным преимуществам Qt можно отнести открытую модульную платформу, что позволяет значительно расширить функционал базовых классов с помощью подключения сторонних модулей, детальную документацию по API (Application Programming Interface), большое количество примеров и развитую техподдержку [12].

Qt имеет очень гибкий набор классов, позволяющий работать с тем вариантом UI (User Interface), который предпочтителен для разработчика, вытекает из требований проекта или требований заказчика. Разработка UI поддерживается с помощью нескольких технологий: QML, C++, HTML5, любых их комбинаций.

Для создания динамических пользовательских интерфейсов в Qt используется набор технологий QT Quick, основанных на языке QML. QT Quick использует все возможности такой популярной библиотеки для работы с графикой как OpenGL. Это позволяет создавать красивые нагруженные интерфейсы без каких-либо заметных последствий в ухудшении производительности.

В библиотеке Qt есть специальный класс для работы с файлами в формате SVG. SVG (Scalable Vector Graphic, или масштабируемая векторная графика) – это графический формат файла, который позволяет отображать двухмерные векторные изображения [13].

Проектирование приложения

При разработке программного обеспечения все требования к нему принято разделять на два основных типа [14]:

– функциональные требования – требования к поведению системы;

– нефункциональные требования – особые свойства или ограничения, накладываемые на систему.

При проектировании данного приложения были выделены следующие функциональные и нефункциональные требования:

– пользователь должен иметь возможность восстанавливать проект с возможностью изменения всех восстановленных объектов;

– пользователь должен иметь возможность сохранять полученный результат во все популярные форматы графики (PNG, JPEG, GIF, PDF);

– пользователь должен иметь возможность изменять геометрию всех предоставленных фигур и их параметры (ширина линии, цвет и т.д.);

– пользователь должен иметь возможность получить справку по горячим клавишам приложения;

– пользователь должен иметь возможность выделять как все объекты в рабочей области, так и часть из них;

– пользователь должен иметь возможность удалять любой из объектов в рабочей области;

– приложение должно иметь модульный интерфейс, что позволит пользователю настраивать панели инструментов так, как ему удобно;

– приложение должно корректно функционировать на следующих операционных системах: Windows, MacOS, Linux.

Логика приложения подразумевает то, что одновременно доступ к управлению приложением имеет только один пользователь. Диаграмма вариантов использования разработанного приложения приведена на рисунке 1.

Описание вариантов взаимодействия пользователя и приложения:

– создать проект – создание пустого проект (расширение файла SVG);

– сохранить проект – пользователю предоставляется на выбор несколько вариантов сохранения проекта: в файл-проект с расширением SVG, формат файла PNG, JPG;

– выбор инструментов рисования – пользователь выбирает графический примитив, с помощью которого будет происходить рисование (круг, прямоугольник, линия и т. д.);

– настройка инструментов рисования – пользователь может настраивать различные параметры графических примитивов: толщина линии, цвет линии, цвет заливки, тип заливки;

– открыть справку о приложении – открыть окно с информацией о комбинациях клавиш, текущей версии программы и т. д.;

– редактирование графического объекта – пользователь в любое время может выделить графический объект на листе и настроить его параметры; дополнительно он может изменять размер, положение в двумерном пространстве, угол поворота объекта;

– создание графического объекта – при выборе графического объекта, его можно разместить на графическом листе приложения.

Дерево классов, используемых при проектировании и разработке приложения, приведено на рисунке 2.

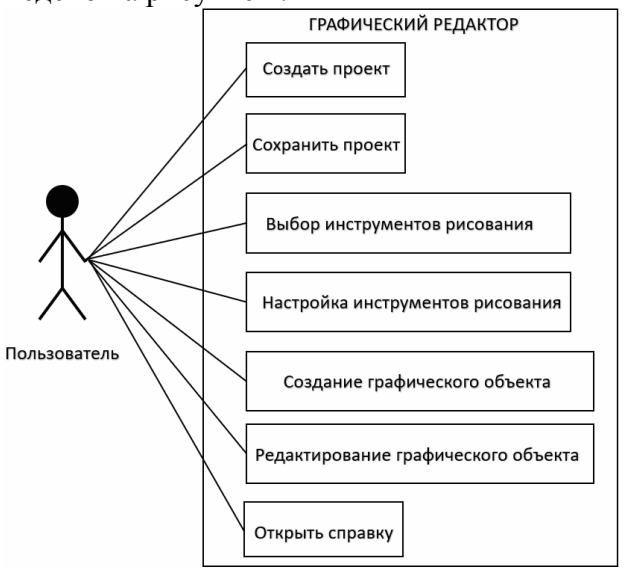


Рис. 1. Диаграмма вариантов использования приложения

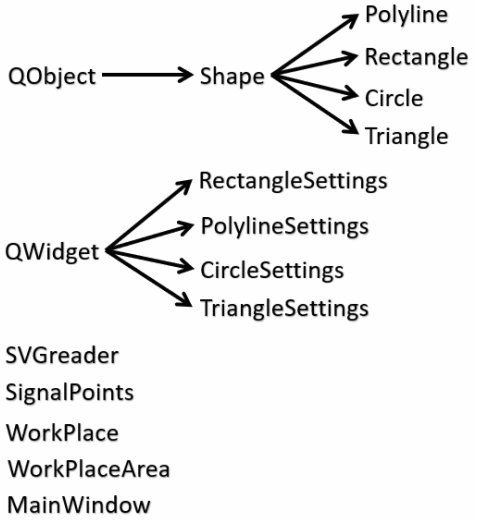


Рис. 2. Дерево классов

Описание разработанных классов:

QObject – основной класс всех объектов в среде Qt;

Shape – абстрактный класс в котором вынесены параметры, которые будут во всех классах-наследниках;

Polyline – ломанная линия;

Rectangle – прямоугольник;

Circle – круг;

Triangle – треугольник;

SVGreader – сохранение проекта в файл с форматом SVG и считывание всей информации о фигурах из SVG-файла при восстановлении проекта;

SignalPoints – точки, с помощью которых пользователь осуществляет манипуляции над фигурами;

MainWindow – логика главного окна приложения.

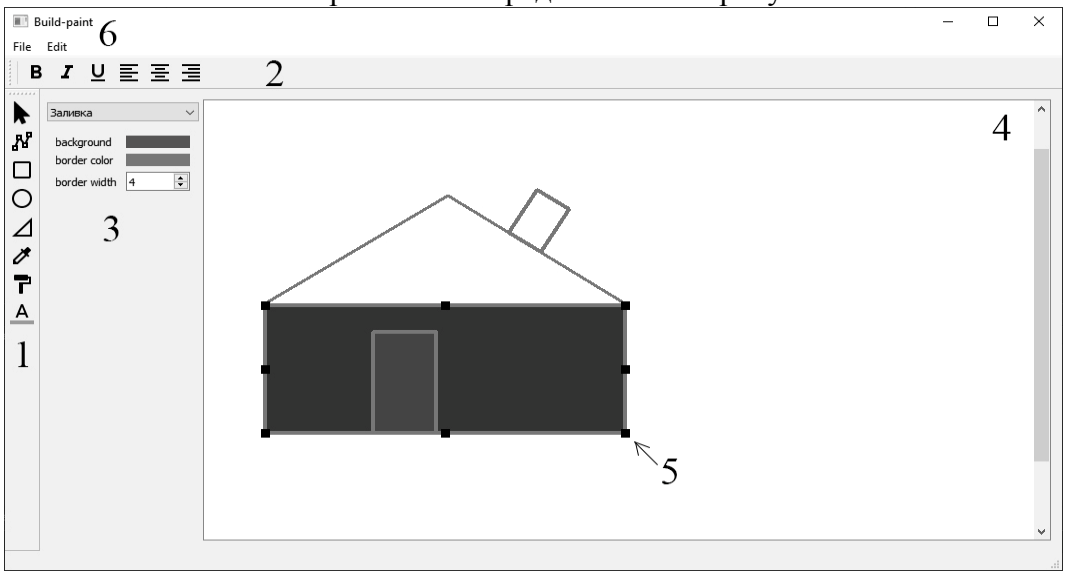
Программная реализация приложения

При запуске приложения пользователь видит главное окно графического редактора с основными панелями управления приложением и разработанными инструментами для рисования (рис. 3).

Пользователь может выбрать один из представленных объектов: ломаная линия, прямоугольник, эллипс, треугольник, инструмент пипетка, инструмент заливка, текст (рис. 3). После выбора объекта пользователь может использовать его функции в рабочей области (класс QGraphicsScene).

У геометрических фигур с помощью дополнительного меню форматирования объектов можно настроить такие параметры как: толщина линии, цвет линии, цвет заливки, тип заливки. С помощью сигнальных точек можно изменять угол поворота фигур и их размер.

Основное меню приложения представлено на рисунке 4.



1 – меню инструментов; 2 – меню форматирования объекта; 3 – дополнительное меню форматирования объекта; 4 – рабочая область; 5 – сигнальная точка; 6 – основное меню

Рис. 3. Интерфейс графического редактора:

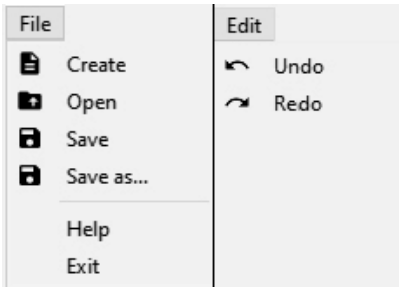


Рис. 4. Основное меню приложения

В основном меню представлены функции:

create – создать проект;
open – открыть существующий проект с SVG форматом файла;
save – сохранить изменения в проекте;
save as – сохранить как;
help – справка;
exit – выход;
undo – шаг назад;
redo – шаг вперед.

С помощью меню можно создать проект, который будет сохранен в формате SVG. Это сделано для того, чтобы при открытии данного файла в разработанном редакторе, была возможность распознавать и изменять все созданные ранее объекты. При выборе функции «save as» можно сохранить все объекты на рабочей области в популярных растровых форматах изображений: PNG, JPEG, GIF, PDF.

Qt обладает специальным классом QSvgRenderer для чтения SVG-файлов и помещения графических объектов на сцену QGraphicsScene. Данный подход восстанавливает файл, как один графический объект, что не соответствуют поставленным целям для разрабатываемого приложения. Именно поэтому в приложении был разработан специальный класс, который бы не только позволял считывать файл, но и обладал бы возможностью разбивать графический объект на его составляющие (QGraphicsItem).

SVG-файл - это текстовый файл, обладающий структурой XML-формата, поэтому его легко можно разложить на составляющие графические объекты. Для этого можно воспользоваться классом QDomDocument, встроенного в Qt и предназначенного для представления XML-документа.

При разборе XML-документ представляется в виде дерева объектов, в качестве которых можно использовать классы QDom. Все объекты классов QDom являются всего лишь ссылками (reference) на внутреннее дерево. Объекты внутреннего дерева существуют до тех пор, пока на них есть ссылки объектов классов QDom и пока существует объект QDomDocument.

При создании элементов, текстовых узлов и т.д. используются функции-фабрики, реализованные в этом классе. Использование конструкторов по

умолчанию классов QDom приведет к созданию пустых объектов, которыми нельзя будет манипулировать или вставлять в документ.

Содержимое документа устанавливается с помощью функции setContent(). Эта функция анализирует строку как XML-документ и создает по нему дерево DOM. Корневой элемент доступен через функцию documentElement(). Текстовое представление документа может быть получено с помощью функции toString().

Для хранения объектов класса QDomDocument использовался контейнер QDomNodeList. Для получения списков использовалась функция elementsByName(), которая возвращает список всех элементов документа с заданным именем. Порядок элементов в списке определяется тем, как они встречаются при обходе в ширину всего дерева.

На рисунке 5 приведена реализация считывания всех ломанных линий из считываемого SVG-файла в лист объектов.

```
QList<VEPolyline *> SvgReader::getPolylines(const QString filename)
{
    QList<VEPolyline *> polylineList;

    QDomDocument doc;
    QFile file(filename);
    if (!file.open(QIODevice::ReadOnly) || !doc.setContent(&file))
        return polylineList;

    QDomNodeList pList = doc.elementsByTagName("g");
    for (int i = 0; i < pList.size(); i++) {
        QDomNode pNode = pList.item(i);
        QDomElement pathElement = pNode.firstChildElement("path");
        if (pathElement.isNull()){
            continue;
        } else {
            VEPolyline *polyline = new VEPolyline();
            auto pElement = pNode.toElement();

            polyline->setBrush(QBrush(Qt::transparent));

            QColor strokeColor(pElement.attribute("stroke", "#000000"));
            strokeColor.setAlphaF(pElement.attribute("stroke-opacity").toFloat());
            polyline->setPen(QPen(strokeColor, pElement.attribute("stroke-width", "0").toInt()));

            QPainterPath path;
            QStringList listDotes = pathElement.attribute("d").split(" ");
            QString first = listDotes.at(0);
            QStringList firstElement = first.replace(QString("M"),QString("")).split(",");
            path.moveTo(firstElement.at(0).toInt(),firstElement.at(1).toInt());
            for(int i = 1; i < listDotes.length(); i++){
                QString other = listDotes.at(i);
                QStringList dot = other.replace(QString("L"),QString("")).split(",");
                path.lineTo(dot.at(0).toInt(),dot.at(1).toInt());
            }
            polyline->setPath(path);
            polylineList.append(polyline);
        }
    }
    file.close();
    return polylineList;
}
```

Рис. 5. Считывание ломаной линии

После того, как получение всех объектов было выполнено, необходимо получить их размер и координаты расположения на графической сцене приложения (рис. 6).

```
QRectF SvgReader::getSizes(const QString filename)
{
    QDomDocument doc;
    QFile file(filename);
    if (!file.open(QIODevice::ReadOnly) || !doc.setContent(&file))
        return QRectF(0,0,200,200);

    QDomNodeList list = doc.elementsByTagName("svg");
    if(list.length() > 0) {
        auto svgElement = list.item(0).toElement();
        auto parameters = svgElement.attribute("viewBox").split(" ");
        return QRectF(parameters.at(0).toInt(),
                      parameters.at(1).toInt(),
                      parameters.at(2).toInt(),
                      parameters.at(3).toInt());
    }
    return QRectF(0,0,200,200);
}
```

Рис. 6. Считывание размера многоугольника

В результате пользователь сможет разобрать сохраненный SVG-файл, чтобы забрать из него созданные ранее фигуры. Сложность заключалась в том, что невозможно написать уникальный парсер (от англ. parser - анализатор данных) объектов, т. к. все они имею разную геометрию и набор полей, которые их описывают. Поэтому для обеспечения возможности восстановления всех разработанных в редакторе фигур в приложении был реализован отдельный класс-парсер для каждой фигуры.

Следует отметить, что, несмотря на то, что в разработанной программе есть возможность считывания фигур, она не может воспроизвести аналогичный прием для файлов, разработанных, например, в среде CorelDraw или Inscape. Дело в том, что версии SVG-файлов могут сильно отличаться генерацией документа. Поэтому приложение может корректно считывать только формат SVG, созданный непосредственно в ней.

В приложении реализованы функции Undo (назад) и Redo (вперед), цель которых отменить совершенное действие и отменить отмену соответственно. Данные функции реализованы по принципу operation-oriented (ориентированный на операции). Данный подход реализуется на основе паттерна «Command» (команда) [15].

Результаты

На рисунках 7,8 представлены некоторые графические изображения, созданные с помощью разработанного приложения. Так, например, неоднократные изменения размера изображения никак не отражаются на его качестве, что продемонстрировано на рисунке 7.

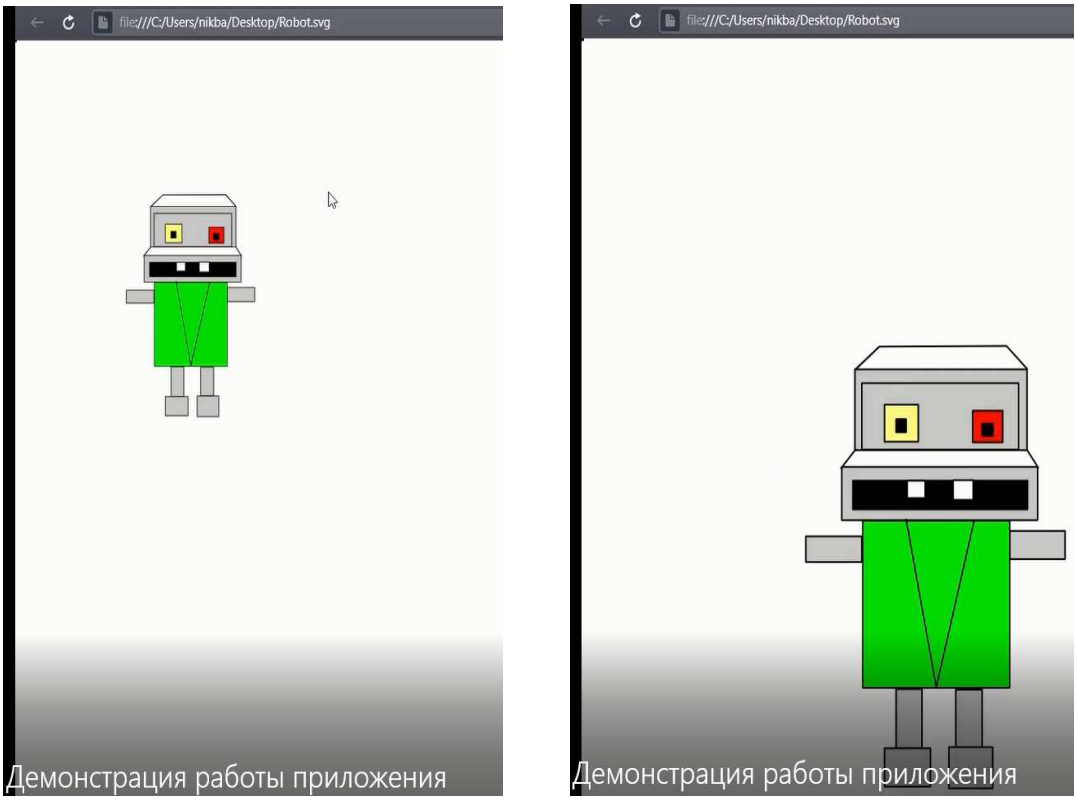


Рис. 7. Неизменность качества изображения при изменении масштаба

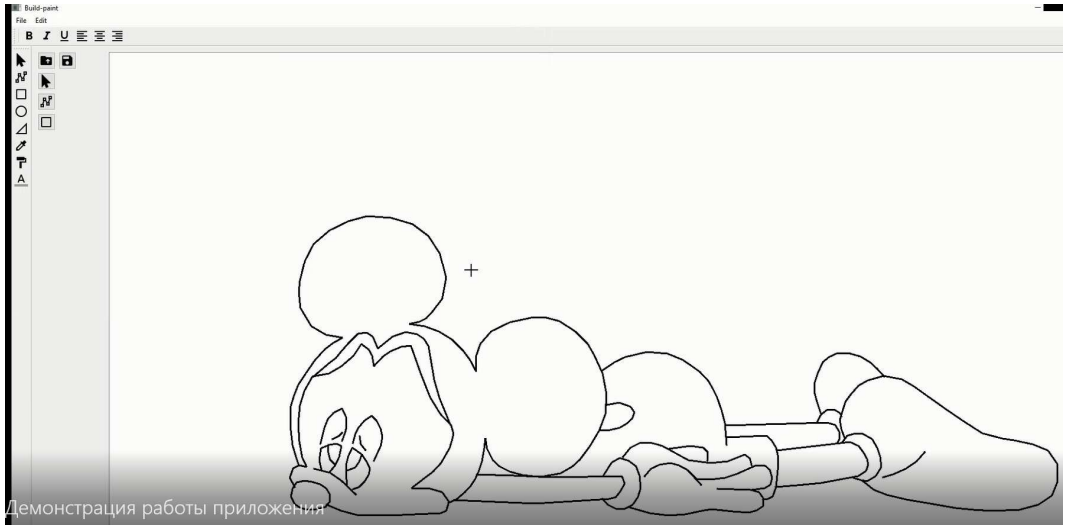


Рис. 8. Изображение, созданное с помощью приложения

Заключение

С помощью библиотеки Qt и языка программирования C++ разработан кроссплатформенный графический редактор, позволяющий создавать, сохранять и обрабатывать изображения в векторном формате SVG.

Список литературы

1. Васильев В.Е. Компьютерная графика / В.Е. Васильев, А.В. Морозов. – СПб: СЗТУ, 2005. – 101 с.
2. Льянов Д.Р. Использование BIM-технологий для создания энергоэффективного будущего // Инженерный вестник Дона. – 2019. – №2. URL: ivdon.ru/magazine/archive/n2y2019/5797/.
3. Рачковская Г.С. Математическое моделирование и компьютерная визуализация сложных геометрических форм // Инженерный вестник Дона. – 2013. – №1. URL: ivdon.ru/magazine/archive/n1y2013/1498/.
4. Маилян А. Что такое нативные и кроссплатформенные приложения? Плюсы и минусы. URL: itvdn.com/ru/blog/article/native-cross-platform.
5. Каратанов А.В. Кроссплатформенное программирование / А.В. Каратанов, Н.Б. Еремеев. – Харьков: ХАИ, 2016. – 108 с.
6. Java Platform Standard Edition 8 Documentation. URL: docs.oracle.com/javase/8/docs.
7. Xamarin. URL: xamarin.com.
8. Боровский А.Н. Qt 4.7+. Практическое программирование на C++. – СПб: БХВ-Петербург, 2012. – 496 с.
9. Шлее М. Qt 5.3. Профессиональное программирование на C++. – СПб: БХВ-Петербург, 2015. – 928 с.
10. Бланшет Ж., Qt 4: программирование GUI на C++ / Ж. Бланшет, М. Свммерфилд. – М.: КУДИЦ-ПРЕСС, 2007. – 641 с.
11. Алексеев Е.Р. Программирование на языке C++ в среде Qt Creator / Е.Р. Алексеев, Г.Г. Злобин, Д.А. Костюк, О.В. Чеснокова, А.С. Чмыхало. – М.: ALT Linux, 2015. – 448 с.
12. QT Documentation. URL: doc.qt.io.
13. Scalable Vector Graphics (SVG) 1.1 (Second Edition). URL: w3.org/TR/SVG/.
14. Арлоу Д. UML 2 и Унифицированный процесс. Практический объектно-ориентированный анализ и проектирование / Д. Арлоу, А. Нейштадт. – М.: Символ-Плюс, 2007. – 624 с.
15. Undo и Redo.URL: habr.com/ru/post/306398.

References

1. Vasiliev V.E., Morozov A.V. Computer graphics. Spb: SZTU, 2005. 101 p.
2. Lyanov D.R. Use of BIM Technologies to Create an Energy Efficient Future // Don’s Engineering Bulletin, 2019. №2 URL: ivdon.ru/magazine/archive/n2y2019/5797/.
3. Rachkovskaya G.S. Mathematical modeling and computer visualization of complex geometric shapes // Don’s Engineering Bulletin, 2013. №1 URL: ivdon.ru/magazine/archive/n1y2013/1498/.
4. Mailyan A. What are native and cross-platform applications? Pluses and minuses. URL: itvdn.com/ru/blog/article/native-cross-platform.

5. Karatanov A.V., Eremeev N.B. Cross-platform programming. Harkov: HAI, 2016. 108 p.
6. Java Platform Standard Edition 8 Documentation. URL: docs.oracle.com/ javase/8/docs.
7. Xamarin. URL: xamarin.com.
8. Borovski A.N. Qt 4.7+. Practical programming in C++. Spb: BHV-Peterburg, 2012. 496 p.
9. Shlee M. Qt 5.3. Professional programming in C++. Spb: BHV-Peterburg, 2015. 928 p.
10. Blanshet Zh., Sammerfild M. Qt 4: GUI programming in C++. M.: KUDITS-PRESS, 2007. 641 p.
11. Alekseev E.R., Zlobin G.G., Kostyuk D.A., Chesnokova O.V., Chmyhalo A.S. C++ programming in the Qt Creator. M.: ALT Linux, 2015. 448 p.
12. QT Documentation. URL: doc.qt.io.
13. Scalable Vector Graphics (SVG) 1.1 (Second Edition). URL: w3.org/TR/SVG/.
14. Arlou D., Neishtadt A. UML 2 and Universal process. Practical object-oriented analysis and design. M.: Simvol-Plyus, 2007. 624 p.
15. Undo и Redo.URL: habr.com/ru/post/306398.

Баланков Никита Валерьевич – магистрант, nikbalsea@gmail.com	**Balankov Nikita Valerievich** – magister student, nikbalsea@gmail.com
Букунов Сергей Витальевич – кандидат технических наук, доцент кафедры информационных технологий, sergeybukunov@yandex.ru	**Bukunov Sergei Vitalievich** – candidate of technical sciences, associate professor of Information Technology Department, sergeybukunov@yandex.ru
Санкт-Петербургский государственный архитектурно-строительный университет, Санкт-Петербург, Россия	Saint-Petersburg State University of Architecture and Civil Engineering, Saint-Petersburg, Russia

Received 08.11.2020