УДК 38

Модуль графического представления информации в программном комплексе обработки экспериментальных данных

Костиков Юрий Александрович
Кандидат физико-математических наук,
заведующий кафедрой 812,
Московский авиационный институт
(национальный исследовательский университет),
125993, Российская Федерация, Москва, Волоколамское шоссе, 4;
e-mail: jkostikov@mail.ru

Павлов Виталий Юрьевич
Кандидат физико-математических наук,
заведующий кафедрой 318,
Московский авиационный институт
(национальный исследовательский университет),
125993, Российская Федерация, Москва, Волоколамское шоссе, 4;
e-mail: kafedra@istmati.ru

Романенков Александр Михайлович
Кандидат технических наук, доцент,
кафедра 812,
Московский авиационный институт
(национальный исследовательский университет),
125993, Российская Федерация, Москва, Волоколамское шоссе, 4;
e-mail: romanaleks@gmail.com

Терновсков Владимир Борисович
Кандидат технических наук, доцент,
кафедра 318,
Московский авиационный институт
(национальный исследовательский университет),
125993, Российская Федерация, Москва, Волоколамское шоссе, 4;
e-mail: vternik@mail.ru

Аннотация

В статье рассмотрена реализация модуля обработки графических данных натурного эксперимента. Проанализированы характерные особенности этих медиаданных и предложены методы повышения скорости рутинных действий при анализе графических материалов, которые основаны на подходах, используемых в цифровой обработке сигналов. Предлагаемые решения предназначены для повышения эффективности деятельности, связанной с оцифровкой данных. С целью обеспечения высокой производительности и следуя современным методам разработки программного обеспечения, графический модуль поддерживает асинхронную загрузку медиаданных в настроенную специальным образом среду обработки данных. Автор подробно рассматривает структуру графического модуля и подходы к реализации каждой части модуля. Организация работы оператора ПК с помощью решений, рассматриваемых в данной статье, позволит значительно сократить время, затрачиваемое на обработку графических данных, что позволит перенаправить освобожденные временные ресурсы на решение текущих актуальных задач.

Для цитирования в научных исследованиях

Костиков Ю.А., Павлов В.Ю., Романенков А.М., Терновсков В.Б. Модуль графического представления информации в программном комплексе обработки экспериментальных данных // Экономика: вчера, сегодня, завтра. 2017. Том 7. № 10А. С. 118-125.

Ключевые слова

Цифровая обработка, распознавание изображения, асинхронная загрузка изображения, OpenTK.

Введение

Необходимость распознавания графических данных очень часто возникает как побочная задача при подготовке данных эксперимента к анализу. На практике часто бывает нужно сравнить полученные результаты с уже имеющимися эмпирическими данными. Однако наличие оцифрованных данных не всегда может быть обеспечено: данные могут быть в печатном формате, например, в журнале, или необходимо затратить неприемлемо большое время для получения цифрового аналога требуемых данных, например в случае ручного воспроизведения изображения. В этих ситуациях может помочь наше решение, в котором оцифровка данных происходит почти моментально в автоматическом режиме. С целью обеспечения высокой производительности и следуя современным методам разработки программного обеспечения, графический модуль поддерживает асинхронную загрузку медиаданных в специальным образом настроенную среду обработки данных.

Структура графического модуля

Для обеспечения корректной и надежной работы модуля распознавания изображения производится ряд вспомогательных операций. Применяемые операции условно можно разделить на несколько блоков, таких как создание вспомогательных опорных элементов, реализация обработки вспомогательных опорных элементов, непосредственная загрузка изображения и прямая реализация цифровых методов обработки загруженного изображения для

выполнения поставленной задачи.

Первый блок – создание вспомогательных опорных элементов – включает следующие операции:

1. Создание вкладки графического интерфейса «Распознавание изображения» под элементами «График 1», «График 2», «Графики сверху/снизу», «Графики слева/справа» на вкладке «Графики» (в верхнем меню программы). Создание вкладки «Распознавание изображения» происходит в разметке XAML главного окна в Grid вместе с остальными вкладками этого меню:

```
<ListBoxx:Name="TreeViewMenu">
<ListBoxItem Content="График 1"IsSelected="True"/>
<ListBoxItem Content="График 2"/>
<ListBoxItem Content="Графикисверху/снизу"/>
<ListBoxItemContent="Графики слева/справа"/>
</ListBox>
```

2. Верстка с использованием средств XAML разметки кнопки «Конфигурация распознавания» со знаком настроек, выделенным красным цветом, справа на графике, пятая по счету, под кнопками со знаком настроек, выделенным зеленым цветом, плюса, минуса и кнопкой «Домой» (иконка домика). Процесс создания кнопки с иконкой красной шестеренки происходит в разметке XAML. Там ей задается стандартный стиль, как у всех кнопок, размер, индекс, отвечающий за приоритет вложенности, и загружается фон. Создание кнопки выглядит следующим образом:

```
<Style
x:Key="SettingButton"
BasedOn="{StaticResourceButtonBaseStyle}"
TargetType="{x:TypeToggleButton}">
<Style.Setters>
<Setter Property="DataContext" Value="{StaticResourceSettingIcon}" />
</Style.Setters>
</Style>
<Button
x:Name="ToolButton"
Margin="5"
Background="{StaticResource&#103;}"
Foreground="{StaticResource&#103;}"
Grid.ZIndex="1"
Style="{StaticResourceToolButton}" />
```

3. Создание модального диалога при нажатии на кнопку «Конфигурация распознавания» со знаком настроек, выделенным красным цветом, справа на графике. В коде в файле Generic.xaml создается элемент, который вызывается при нажатии на кнопку «Конфигурация распознавания»:

```
<md:PopupBox.PopupContent>
<Border>
Width="200"
Height="300"
 Margin="3"
 Background="White"
BorderBrush="{StaticResource&#101;}"
```

```
BorderThickness="0.5"
CornerRadius="3"
Effect="{StaticResource&#102;}">
</Border>
</md:PopupBox.PopupContent>
</md:PopupBox>
```

4. Создание двух параметров модального диалога, вызываемого при нажатии на кнопку «Конфигурация распознавания» со знаком настроек, выделенным красным цветом, справа на графике: «Загрузить изображение» и «Очистить». В том же блоке создания окна появляются и параметры «Загрузить изображение» и «Очистить»:

```
<StackPanel>
<StackPanel.Background>
<ImageBrushImageSource="{StaticResource Overlay}" />
</StackPanel.Background>
<DockPanel>
<ListBoxItem
x:Name="ListBox"
Text=”Загрузить изображение”
  Margin="5"
HorizontalAlignment="Left"
VerticalContentAlignment="Center"
Content="Grid"
FontSize="15"
FontWeight="Light"
/><ListBoxItem
x:Name="ListBox"
Text=”Очистить”
  Margin="5"
HorizontalAlignment="Left"
VerticalContentAlignment="Center"
Content="Grid"
FontSize="15"
FontWeight="Light"/>
</DockPanel>
</StackPanel>
</StackPanel>
```

5. Вызов модального диалога при нажатии на кнопку «Конфигурация распознавания». При наведении на кнопку генерируется событие и срабатывает функция privatevoidOnGotFocus (object sender, RoutedEventArgs e), в которой происходит выставление необходимых элементов управления и настройка значений параметров. При нажатии кнопки вызывается функция protectedoverridevoidOnKeyDown(KeyEventArgse).

Второй блок – реализация обработки вспомогательных опорных элементов – содержит следующие операции:

1. Установление ассоциаций между появлением кнопки «Конфигурация распознавания» и выбором вкладки «Распознавание изображения» слева от графика во вкладке «Графики» (в верхнем меню программы).

2. Вызов модального диалога при нажатии на кнопку «Конфигурация распознавания» со

знаком настроек, выделенным красным цветом, справа на графике.

3 Обработка двух параметров модального диалога, вызываемого при нажатии на кнопку «Конфигурация распознавания» со знаком настроек, выделенным красным цветом, справа на графике: «Загрузить изображение» и «Очистить».

Третий блок – загрузка изображения – включает такие операции:

1. Обращение к стандартному диалогу загрузки изображения с фильтрацией имен файлов. Загрузка изображения осуществляется с помощью стандартного диалога открытия файла: var dialog = newOpenFileDialog() { Filter = "PNG Files(*.png)|*.png" }. В результате создается и активируется модальное окно, в котором имеется возможность выбрать необходимый файл. Загружаемый файл может быть только формата «.png».

2. Осуществление асинхронной загрузки изображения для оптимизации работы программы. Для оптимизации работы программы мы используем загрузку изображения в асинхронном режиме. Для этого создается метод с пометкой async, что означает, что он будет выполняться асинхронно – не задерживая основной поток и не блокируя интерфейс на время загрузки изображения. Этот метод возвращает ссылку на объект типа Task<T>, внутри которого происходит загрузка изображения с помощью метода с пометкой await. Bdispatch сообщается, что загрузка закончилась, и изображение на окне обновляется. Для отображения изображения используется достаточно большое количество методов.

К примеру, примитивы создаются следующим образом:

```
glBegin(BeginMode.Points );  // указываем, что будем рисовать
glVertex[2 3 4][s i f d](...); // первая вершина
  ...                         // остальные вершины
glVertex[2 3 4][s i f d](...); // последняя вершина
glEnd();                 // закончили рисовать примитив
```

В glBegin передается соответствующий параметр, который является тем, что мы будем рисовать. Возможные значения mode разнообразны. Далее указываются вершины, определяющие объекты указанного типа. Обычно задается вершина одним из четырех способов:

```
glVertex2d(x,y);    // две переменных типа double
glVertex3d(x,y,z);  // три переменных типа double
glVertex2dv(array); // массив из двух переменных типа double
glVertex3d(array);  // массив из трех переменных типа double
```

И наконец, происходит вызов glEnd, чтобы указать, что окончили рисовать объекты типа, указанного в glBegin.

Однако у нас реализован метод DrawArrays(…),который позволяет отображать список вершин. В этом случае происходит загрузка всех вершин в массив (а также выполняется загрузка массива цветов) и с помощью специальной команды **glDrawArrays(...)** отображается вся фигура. Кроме того, эта команда реализована аппаратно. И такой подход намного эффективнее, чем рисование всех вершин. Данный метод принимает следующие параметры:

а) Mode – указывает, какие примитивы нужно отображать; символьные константы GL_POINTS, GL_LINE_STRIP, GL_LINE_LOOP, GL_LINES, GL_TRIANGLE_STRIP, GL_TRIANGLE_FAN, GL_TRIANGLES, GL_QUAD_STRIP, GL_QUADS и GL_POLYGON принимаются;

б) First – задает начальный индекс в разрешенных массивах;

в) Count – определяет количество отображаемых индексов.

Команда glDrawArrays указывает множество геометрических примитивов с очень небольшим количеством вызовов подпрограмм. Вместо вызова процедуры GL для прохождения каждой отдельной вершины, нормальной, текстурной координаты, граничного флага или цвета,

мы имеем возможность задавать отдельные массивы вершин, нормалей и цветов и использовать их для построения последовательности примитивов с одним вызовом glDrawArrays. Когда вызывается glDrawArrays, он использует подсчет последовательных элементов из каждого разрешенного массива для построения последовательности геометрических примитивов, начиная с элемента mode, который определяет, какие примитивы построены и как элементы массива конструируют эти примитивы. Если GL_VERTEX_ARRAY не включен, геометрические примитивы не генерируются. Атрибуты вершины, которые были изменены glDrawArrays, имеют неопределенное значение после возвращения glDrawArrays. Например, если включена опция GL_COLOR_ARRAY, значение текущего цвета не определено после выполнения glDrawArrays. Атрибуты, которые не изменены, остаются четко определенными.

Для загрузки и отображения выбранного пользователем изображения используется метод GL.DrawElements (BeginMode, intLength, DrawElementsType, object).

Параметр glDrawElements указывает множество геометрических примитивов с очень небольшим количеством вызовов подпрограмм. Вместо вызова функции GL для передачи каждой отдельной вершины, нормальной, текстурной координаты, граничного флага или цвета, можно задавать отдельные массивы вершин, нормалей и др. и использовать их для построения последовательности примитивов с одним вызовом к glDrawElements.

При вызове glDrawElements используется подсчет последовательных элементов из разрешенного массива, начиная с заданных индексов, чтобы построить последовательность геометрических примитивов. Mode определяет, какие примитивы построены и как элементы массива конструируют эти примитивы. Если включено более одного массива, то каждый из них используется. Если GL_VERTEX_ARRAY не установлен, геометрические примитивы не создаются.

Атрибуты вершин, которые были изменены glDrawElements, имеют неопределенное значение после возвращения glDrawElements. Например, если установлен GL_COLOR_ARRAY, значение текущего цвета не определено после выполнения glDrawElements. Атрибуты, которые не изменены, сохраняют свои предыдущие значения.

3. И наконец, последняя операция третьего блока – представление загруженного изображения.

Заключение

Таким образом, нами предложен подход к реализации модуля для распознавания графической информации, а также описана структура и детально изложены подходы к реализации каждой части модуля. С целью обеспечения высокой производительности и следуя современным методам разработки программного обеспечения, данный графический модуль поддерживает асинхронную загрузку медиаданных в специальным образом настроенную среду обработки данных. Организация работы оператора ПК с помощью решений, рассматриваемых в данной статье, позволит значительно сократить время, затрачиваемое на обработку графических данных, что позволит перенаправить освобожденные временные ресурсы на решение текущих актуальных задач.

Библиография

1. Ву М., Девис Т., Нейдер Дж., Шрайнер Д. OpenGL. Руководство по программированию. Библиотека программиста. 4-е изд. СПб.: Питер, 2006.
2. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования. Паттерны

проектирования. СПб.: Питер, 2016.
3. Дэйт К.Дж. Введение в системы баз данных. М.: Вильямс, 2017.
4. Мак-Дональд М. WPF: Windows Presentation Foundation в .NET 4.5 с примерами на C# 5.0 для профессионалов. М.: Вильямс, 2015.
5. Поляков В.П. Аспекты информационной безопасности в информационной подготовке. М.: ИУО РАО, 2016. 135 с.
6. Поляков В.П. Развитие информационной подготовки в контексте Стратегии национальной безопасности Российской Федерации // Наукоград: наука, производство, общество. 2016. № 2. С. 46-51.
7. Троелсон Э. Язык программирования C# 5.0 и платформа .NET 4.5. М.: Вильямс, 2015.

Module of graphical representation of information in the software package of experimental data processing

Yurii A. Kostikov

PhD in Physical and Mathematical Sciences,
Head of the Department 812,
Moscow Aviation Institute (National Research University),
125993, 4 Volokolamskoe highway, Moscow, Russian Federation;
e-mail: jkostikov@mail.ru

Vitalii Yu. Pavlov

PhD in Physical and Mathematical Sciences,
Head of the Department 318,
Moscow Aviation Institute (National Research University),
125993, 4 Volokolamskoe highway, Moscow, Russian Federation;
e-mail: kafedra@istmati.ru

Aleksandr M. Romanenkov

PhD in Technical Sciences, Associate Professor,
Department 812,
Moscow Aviation Institute (National Research University),
125993, 4 Volokolamskoe highway, Moscow, Russian Federation;
e-mail: romanaleks@gmail.com

Vladimir B. Ternovskov

PhD in Technical Sciences, Associate Professor,
Department 318,
Moscow Aviation Institute (National Research University),
125993, 4 Volokolamskoe highway, Moscow, Russian Federation;
e-mail: vternik@mail.ru

Abstract

The article describes the implementation of a module for experimental data processing. The authors analyze the characteristics of these media data and propose the methods for enhancing the speed of routine actions during the analysis of graphic materials, which are based on approaches used in digital signal processing. The proposed solutions are intended to improve the efficiency of activities related to the digitization of data. Following modern methods of software development, graphic module supports asynchronous loading of media data in a special data processing environment. The authors considers in detail the structure of the graphic module and the approaches to implementation of each part of the module. To ensure correct and reliable operation of the recognition module of the image a number of supporting operations are produced. These operations can be divided into several blocks, such as the establishment of auxiliary support elements, the implementation of processing auxiliary supporting elements, immediate loading of images, a direct implementation of digital methods of processing the uploaded images for the task solution. Organization of work of keyboarder due to the solutions discussed in this article, will significantly reduce the time spent on processing the image data, that will redirect the freed time resources to the solution of urgent problems.

For citation

Kostikov Yu.A., Pavlov V.Yu., Romanenkov A.M., Ternovskov V.B. (2017) Modul' graficheskogo predstavleniya informatsii v programmnom kom-plekse obrabotki eksperimental'nykh dannykh [Module of graphical representation of information in the software package of experimental data processing]. *Ekonomika: vchera, segodnya, zavtra* [Economics: Yesterday, Today and Tomorrow], 7 (10A), pp. 118-125.

Keywords

Digital processing, image recognition, asynchronous loading of images, OpenTK.

References

1. Deit K.Dzh. (2017) *Vvedenie v sistemy baz dannykh* [Introduction to database systems]. Moscow: Vill'yams Publ.
2. Gamma E., Khelm R., Dzhonson R., Vlissides Dzh. (2016) Priemy ob"ektno-orientirovannogo proektirovaniya. Patterny proektirovaniya [Techniques of object-oriented design. Design patterns]. Saint Petersburg: Piter Publ.
3. Mak-Donal'd M. (2015) *WPF: Windows Presentation Foundation v .NET 4.5 s primerami na C# 5.0 dlya professionalov* [Windows Presentation Foundation in .NET 4.5, with examples in C# 5.0 for professionals]. Moscow: Vill'yams Publ.
4. Polyakov V.P. (2016) Razvitie informatsionnoi podgotovki v kontekste Stra-tegii natsional'noi bezopasnosti Rossiiskoi Federatsii [Development of information training in the context of the National Security Strategy of the Russian Federation]. *Naukograd: nauka, proizvodstvo, obshchestvo* [Naukograd: science, science, production and society], 2, pp. 46-51.
5. Polyakov V.P. (2016) *Aspekty informatsionnoi bezopasnosti v informatsionnoi podgotovke* [Aspects of information security in information training]. Moscow Institute of Education Management of the Russian Academy of Education.
6. Troelson E. (2015) *Yazyk programmirovaniya C# 5.0 i platforma .NET 4.5* [The programming language C# 5.0 and the platform .NET 4.5]. Moscow: Vill'yams Publ.
7. Vu M., Devis T., Neider Dzh., Shrainer D. (2006) OpenGL. Rukovodstvo po pro-grammirovaniyu. Biblioteka programmista [OpenGL. Programming Guide. Library for programmer], 4th ed. Saint Petersburg: Piter Publ.