ELECTRONICS AND ELECTRICAL ENGINEERING
ISSN 1392 – 1215 *2007. No. 1(73)*
ELEKTRONIKA IR ELEKTROTECHNIKA

MEDICINE TECHNOLOGY
T 115
MEDICINOS TECHNOLOGIJA

Improving the Performance of Program Package for 3D Simulation of Low Frequency Magnetic Field in Medical Therapy

D. Tz. Dimitrov
Faculty of Communication Technique and Technologies, Technical University of Sofia,
8, Kliment Ohridsky str. 1000 Sofia, Bulgaria, tel.: +359 2 9652278, e-mail:dcd@tu-sofia.bg

Introduction

The purpose of the article is to cover some aspects of the implementation of the program for a 3D simulation of a low frequency magnetic field. This simulation is required for an efficient design of apparatuses for magnetotherapy in medicine, as far as the computer-aided analysis of magnetic field distribution becomes the most effective way for evaluating such electromagnetic devices and their component's performance. The field simulation requires input of the current in the coils , parameters of the coils, their space disposition, some conditions, a lot of calculations and output of the desired parameters [1–2].

A typical cycle of the proposed simulation program includes the following steps: entering the parameters of the field sources and their geometry (relative positions) as well as the points, surface or volume of interest; calculating the magnetic induction in each point; visualizing the results in different ways. If the results are found unsatisfactory, the design is modified and the magnetic induction vector data is recalculated. These steps are repeated until the desired results are obtained. The factors that influence the effectiveness of the described process are the ease of use, the accuracy of results, and the speed of the program.

The method for a magnetic induction calculation used in the presented program solution is specific, because it concerns the particular case of low frequency magnetic field with an axisymetry generated by a circular current contour. Other methods such as FEM and BEM are much more universal, but they require more time for calculations [3–4]. The applied method provides a good precision and permits the parameters (magnetic induction vector) to be calculated in every 3-dimensional point of interest. The goal of the program is to perform the calculation method multiple times for different sets of points. The sets of points can be generated as a result of analytical description as well as selected by the user, or entered by keyboard [5]. After calculating the magnetic induction for every point of the set, the information was saved for subsequent analysis and visualization. As a result of the analysis and the visualization the user can follow:

- the behavior of the field as a function of the current and contour radius;
- the behavior of the resultant field, generated of separate contours, as a function of the number of contours and their relative positions;
- the influence of the low frequency magnetic field on the human body represented as a geometric model.

3D simulation of magnetic field includes as its obligatory pan support for the fastest possible visualization of the vector field. The vector fields are characterized with a n-component vector (3D coordinates, components of the magnetic induction, color information) in each point of the field. This vector data tends to grow when the number of points grows and it raises the question about the structure and content of the data and the way of its preparation and processing [6]. The requirement for dynamic change in the number and position of field sources and parameters of the sources leads to frequent changes in number and configuration of the set of points of interest. Depending on the chosen visualization technique (numerical data, vectors, field lines, surfaces) additional calculations and analysis of the vector data will take place.

Choosing of the most suitable technology for the application

From the implementation point of view the presented software solution uses the advantages of the object oriented and component oriented approaches that are inherent to the chosen programming language C# and development environment .NET.

.NET Framework supports three core technologies for developing applications: Windows Forms, ASP .NET Web Forms and XML Web services. The choice of technology depends on the characteristics of the particular software solution and the requirements of the end users.

Because the 3-dimensional simulation of magnetic field requires a great deal of calculations, high quality color graphics and highly interactive user interface, the best suited technology is Windows Forms. Windows

Forms are used to develop classic Microsoft Win32 desktop applications in which the desktop computer handles all of the application processing: calculations and data visualization. The main advantage of this technology is the responsiveness. In the presented software solution the 3-dimensional vector data is encapsulated into two connected classes. The first class named *magneticVector* contains member data that match the components of the magnetic induction vector and the properties needed for setting and obtaining member data values. The second class named vector Data provides a wrapper for an Array List of magnetic induction vectors and includes the necessary methods and properties for accessing data members. Most of the computational methods of the solution are implemented as instance methods of the separate class named calculate.

All mentioned classes are combined into a single namespace named calculate Component and are encapsulated into a single component with the same name. The component is separately compiled to an intermediate (MSIL) code and stored as a *.dll file. The file consists of a single self-explanatory assembly with embedded metadata that provides all the information required for a software component interaction. In the presented software solution the methods and properties exposed by the component, are called by the simple Windows Forms client application. The main responsibilities of the client are to supply a flexible and highly interactive user interface and thus to allow the users easily to change the configuration as well as the parameters and to see the graphical or numerical results as quickly as possible.

The components, developed using C#, are the basic approach to create reusable classes. Thus the fields, properties and methods, once defined into the component, can be used by different kind of applications - not only desktop applications, but also ASP .NET applications, without any changes in the contents (code) of the components. Using components by ASP .NET client wiil avoid the problem with the necessity of .NET Framework running on the client computer, which is the strong requirement of the Windows Forms applications, but ASP NET applications are out of the scope of this article.

Achieving Better Application Performance

As stated above, methods and properties exposed by the component are called from the Windows Forms client, which supports flexible and highly interactive user interface. In order to guarantee high responsiveness of the application and to prevent the blocking of the user interface for long periods of time the client is realized as a multithread application. In such a way the user can continue his work on the main thread reviewing or visualizing already prepared data, while on another thread computational methods called asynchronously calculate and populate different dataset..NET Framework provides a rich support (*System.Threading* namespace) that greatly simplifies working with multithread applications. In addition because the asynchronous programming is a core concept, the .NET Framework provides a common design pattern to handling asynchronous execution (*IAsyncResult* interface and asynchronous delegate classes), that enables a programmer to avoid some of the implementation details of threading. In the presented program solution most of the computational methods encapsulated into the component *calculate Component* are implemented in such a way to provide both synchronous and asynchronous calls.

.NET Framework Class Library provides classes with build-in asynchronous support (for example File I/O, ASP .NET pages). Such classes have a pair of asynchronous methods for each synchronous method they contain. Unfortunately, classes into the *calculateComponent* are user classes and therefore have no build-in asynchronous support. The developer can still make asynchronous calls to each synchronous method, but some additional steps are required. During this steps must be used the mechanism of the asynchronous delegates, to guarantee that the behaviour of the user classes will correspond to the .NET Framework asynchronous design pattern.

One of the goals of this article is to formalize and explain the procedure for creating and using methods that support both synchronous and asynchronous calls. For illustration purposes the method calculate *B vector()* is used from the class *calculate* of the name space *calculateComponent* as shown below:

```
{
class calculate
void calculateBVector(IEnumrable list)
{ // code of the method.........            }
//other methods of the class
public delegate void
calculateBVectorDelegate(IEnumerable list);
//*************
}
//other classes
} // end of the namespace
```

The first step of the procedure is to explicitly declare a delegate for a method *calculateBVector()* (the statement commented with an '*'s in the above code).
Note that the delegate must have the same signature as the target method. Such a delegate must be declared for each method that must support asynchronous calls and is an instance or static method of a class with no build-in asynchronous support.

When the delegate that references *calculateBVector()* method is declared, the C# compiler generates the *Invoke* method together with the *Beginlnvoke* and *EndInvoke* methods for the target *calculateBVectors()* method.

Invoke method has to be used when a synchronous call of the target method is needed. *BeginInvoke* and *EndInvoke* methods are related to the asynchronous call of the same target method. After declaring all needed delegates calculate Component can be compiled to a .dll and supplied to the Windows Forms client or ASP NET client.

All other programming actions concern the code of the Windows Forms client only.

The second step in the procedure is to choose the completion mechanism. The most appropriate for Windows Forms applications completion mechanism is the callback method. The callback method is called automatically by .NET Framework when the asynchronous operation is completed and must have the following signature: **void *method*(IASyncResult obj);**

Inside the callback function the *EndInvoke* method for the target method must be explicitly called to retrieve the results of the asynchronous operation. Next, the callback method is responsible for returning to the main thread and updating the user interface. Updating user interface is a mechanism for synchronization preferred in the Windows Form applications and requires a separate function to be defined with the following signature: **void *method*(); .**

The code of this function usually changes the properties *Enabled* and *Text* of the Windows Form controls and thus alters both the appearance and the behavior of the controls.

The best suited place to define both functions (callback function and function that updates UI elements) is the Windows Form, from which the asynchronous call is made. The requirement for special signature of these two functions is obligatory, because they are called through the delegates. The developer should not explicitly create delegates that reference these methods because the .NET Framework supports needed delegate classes *AsyncCallback* and *MethodInvoke*.

The contents of these two functions is as follow:

```
public void updateButtonAppearance()
{
//changes the property Eenabled of the button,
named calculate Field
this. calculateField. Enabled = true;
}
public void calculateBVectorCallback(lAsyncResult
obj}
{
//create an object of the delegate class
calculateBVectorDelegate
calculate.calculateBVectorDelegate a;
// initialize the object
a = (calculate.
calculateBVectorDelegate)((AsyncResult)obj).Async
Delegate;
//call the Endlnvoke method for the target method
calculate BVector()
a.Endlnvoke(obj);
//return to the main thread
//create an object of the .NBT Framework delegate
class Methodlnvoker that
//refferes to the updateUI function
MethodInvoker upd = new
Methodinvoker(updateButtonAppearance),
//call asynchronously the function
updateBiittonAppearance() to return control to
the
//main thread and change the appearance of the
Form's controls
this.BeginInvoke(upd);
}
```

The last and simplest step in the procedure is to asynchronously call the target method *calculateBVector()* inside the handler for the event *button _CIick* of the button named *calculateField*, as shown below:

```
private void calculateField_Click(object sender,
System.EventArgs e)
}
//create an object of class calculate that
contains ihe target method calculateBVector()
calculate c = new calculate();
//create an object of class vectorData that
contains the defines the data to be processed
vectorData list = new vectorData();
//create an object of the .NET Framework delegate
class AsyncCallback that reffers to the
//callback function
AsyncCalLBack callback = new
AsyncCallback(this.calculateBVectorCallback);
this. calcuiateField.Enabled = false;
//call target method calculateBVector()
asynchronously
IAsyncResult obj = c.BeginInvoke(list, callback,
null);
}
```

When the *BeginInvoke* method was called it calls the target method *calculateBVector()* on a separate thread from the runtime's thread pool and returns immediately to the caller. The main thread, which made the asynchronous call is free to continue execution in the parallel to the target method which is running on a thread *pool thread*.

On the completion of the asynchronous operation the callback fianction is automatically called by the .NET Framework. In turn, the callback function calls *EndInvoke* method for the target method *calculateBVector()* and obtains the results. Finally the user is notified for the completion of the asynchronous operation through the updated UI: the appearance of the button calculate Field is changed.

Before updating to take place the callback function returns control to the main thread using the MethodInvoke delegate; the thread in which the *calculateBVector()* method runs asynchronously returns to the thread pool. The switch of control between threads is needed because Windows Form controls can only be safely called from the main thread.

Conclusions

The current paper illustrates the procedure for implementing the .NET Framework asynchronous programming model using only one of the computational methods as an example. All the methods in the program solution that are time consuming and play a key role in the computation and visualization process are implemented in such a way to permit asynchronous calls.

On the base of the created component without any changes in its code an ASP .NET client application was developed too. This application till now implements only small part of the functionality of the presented Windows Forms application. Its goal is to demonstrate how the change of the technology can satisfy different user requirements with minimal or no code changes.

References

1. **Brandinsky K., Georgiev G., Mladenov V., Stancheva R.** Elektrotechics, part 1. – Sofia, 2005. – P.368.
2. **Foley J. D., A. van Dam, S. Feiner, J. Hughes.** Computer Graphics: Principles and Practice. – Addison-Wesley, Reading. – MA, 1990. – P. 518.
3. **Hearn D., Baker M. P.** Computer Graphics. – Prentice-Hall, NJ. – 1997. – P. 370.
4. **Тодорова В., Малешков Ст.** Geometric Data Exchange in XML format Using .NET Environment // Proceedings of International Conference “Computer Science '2004”. – Technical University of Sofia, Bulgaria, 2004. – P. 68–74.

5. **Loop Ch.** Triangle Mesh Subdivision with Bounded Curvature and the Convex Hull Property // Proceedings of International Conference, ICEST 2006. – Technical University of Sofia, July 1–2, 2006. – P. 134–139.

6. **Feiner S., Hughes J.** Applications for Windows with Visual C# .NET. – Addison-Wesley-Reading, MA, 1998. – P. 240.

Submitted for publication 2006 10 01

D. Tz. Dimitrov, Improving the Performance of Program Package for 3D Simulation of Low Frequency Magnetic Field in Medical Therapy // Electronics and Electrical Engineering. – Kaunas: Technologija, 2007. – No. 1(72). – P. 69–72.

The purpose of this article is to present one particular application of the new .NET technologies: Windows Forms and asynchronous programming model. These technologies are used in the development of the software solution tor 3-dimensional simulation of low frequency magnetic field. The simulation requires preparation and processing of the 3-dimensional vector data which is a time consuming task and thus requires special cares for optimizing both the user interface and the organization of the calculations. Bibl. 6 (in English; summaries in English, Russian and Lithuanian).

Д. Ц. Димитров. Улучшение действия программ для 3D моделирования низкочастотного магнитного поля в медицинской терапии // Электроника и электротехника. – Каунас: Технология, 2007. – № 1(73). – С. 69–72.

Цель статьи – демонстрирование одного особого применения новой технологии .NET: Windows Forms и асинхронной программной модели. Этой технологией пользуются в процессе развития программ компьютерного 3D моделирования низкочастотного магнитного поля. Данное моделирование требует подготовления 3D вектора данных. Этот вектор со своей стороны требует запаса времени, который необходим для процесса вычислений и для организации потребительского интерфейса. Библ. 6 (на английском языке; рефераты на английском, русском и литовском яз.).

D. Tz. Dimitrov. Programų paketo, skirto medicininės terapijos žemųjų dažnių magnetiniam laukui 3D modeliuoti, veikimo pagerinimas // Elektronika ir elektrotechnika. – Kaunas: Technologija, 2007. – Nr. 1(73). – P. 69–72.

Straipsnio tikslas – parodyti, kaip naudojama nauja .NET technologija – Windows Forms ir asinchroninio programavimo modelis. Šios technologijos panaudotos žemųjų dažnių magnetinio lauko 3D kompiuteriniam modeliavimui tobulinti. Tokiam modeliavimui reikia paruošti ir apdoroti trijų matmenų duomenų vektorių. Šioms operacijoms gaištamas laikas, todėl svarbu optimizuoti tiek vartotojo sąsają, tiek skaičiavimo algoritmus. Bibl. 6 (anglų kalba; santraukos anglų, rusų ir lietuvių k.).

DOI: 10.5755/j02.eie.10342