한국정밀공학회지 제 34 권 제 11 호 pp. 797-805 November 2017 / 797
J. Korean Soc. Precis. Eng., Vol. 34, No. 11, pp. 797-805 https://doi.org/10.7736/KSPE.2017.34.11.797
ISSN 1225-9071 (Print) / 2287-8769 (Online)

Visual Fortran과 OpenGL을 이용한 윈도우용 레이저가공 통합 시뮬레이션 프로그램 개발

Development of Laser Ablation Integrated Simulation Program for Windows with Visual Fortran and OpenGL

방세윤[1,#]
Se Yoon Bang[1,#]

1 동국대학교 서울캠퍼스 기계로봇에너지공학과 (Department of Mechanical, Robotics and Energy Engineering, Dongguk University-Seoul)
Corresponding Author / E-mail: sybang@dongguk.edu, TEL: +82-2-2260-3704, FAX: +82-2-2263-9379

KEYWORDS: Visual Fortran (비주얼포트란), OpenGL (오픈지엘), Laser ablation (레이저 어블레이션), Integrated simulation program (통합 시뮬레이션 프로그램), Windows (윈도우 운영체제)

An integrated simulation program for virtual laser ablation is developed to help understand and predict the effects of machining parameters on the final machined results. The main solver of the program is based on the model for polymer ablation with short-pulse Excimer lasers. The GUI of the program is built using Visual Fortran and OpenGL so that the user can work in a visual environment, such as Windows on a PC, where the important machining parameters can be input via a dialog box and the calculated results for the machined shape can be plotted by means of a 3D graphics window using OpenGL. The developed GUI can be implemented for use with most pre-developed FORTRAN solvers for Windows application, allowing the user to control the input parameters and to see the results in a 3D plot; hence most FORTRAN users could create their own visual programs on PC Windows systems similarly, widening the range of application of FORTRAN solvers.

Manuscript received: February 20, 2017 / Revised: June 28, 2017 / Accepted: July 17, 2017

NOMENCLATURE

P = Laser beam energy per pulse
w_o = Beam radius at focal position
H = Laser irradiance on surface of the specimen
d = Etched depth per pulse
f = Frequency of pulse laser beam
Xmin, Xmax, Ymin, Ymax = Non-dimensional value of minimum and maximum x, y-ranges
a = Height of rectangular mask
b = Width of rectangular mask
h, 2θ = Shape parameters of triangular mask
w = Radius of half-circular mask

1. 서론

PC에서 실제현상의 해석을 위해 널리 이용되어 온 수치모사의 프로그래밍 언어는 거의 Fortran을 이용해 왔으나, 윈도우환경의 운영체제가 보편화됨에 따라 GUI를 통해 보다 시각적인 환경에서 사용자가 관련 변수를 입력하고 계산 결과를 그래프나 표의 형태로 출력하도록 구성해 해석하기 쉽게 보여줄 필요가 증대하고 있다. C 또는 Visual Basic과 같은 윈도우 전용 언어를 사용하면 비주얼 환경의 구축에 효과적이기는 하다. 반면에 Fortran 언어의 경우는 수치 계산의 효율성과 오랜 기간 개발되어 온 풍부한 소스의 라이브러리가 있지만 일반적으로는 비주얼 환경의 프로그램 개발에는 적절하지 않은 것으로 간주되고 있다.[1] 본 논문에서는 Fortran 프로그램을 이용해 PC의 윈도우 운영체제에서 비주얼 환경의 입력 및 3D 출력과 계산을 수행할 수 있는 통합 프로그램을

개발하고자 하며, 이를 위해 입출력을 담당하는 GUI부를 개발하고 기존의 계산 수행을 담당하는 Fortran 솔버(Solver)부와 결합하는 것을 목표로 한다.

본 논문에서는 Visual Fortran[2]과 비주얼 스튜디오[3]를 개발 환경으로 구축하고, 원래의 계산 부분인 솔버는 통상의 Fortran 프로그램으로 만들어진 루틴을 그대로 활용하였다. 입력부분은 윈도우에서 대화상자를 이용해 구축하였으며, 출력부분은 WIN32 API 함수를 이용해 OpenGL 창을 등록/실행한 후 OpenGL 함수를 불러 가공 깊이를 나타내는 3D 서피스를 화면에 보이고, 마우스로 회전 또는 자동 회전하도록 구현하였다.

폴리머의 레이저 어블레이션 가공 모델[4,5]을 대상 솔버로 활용하였으며, 기 발표된 정지 상태에서의 구멍 가공 루틴과, 마스크를 이용한 일정 속도의 이송가공[6] 상태에서의 준삼차원 미세 형상가공 루틴을 프로그램에 통합해 구축하였다. 대상 소재는 PI, 폴리우레탄, PMMA 등의 폴리머이며 (1) 정지상태에서 다양한 레이저빔 변수에 따른 어블레이션 (2) 균일한 강도분포의 평행빔이 다양한 형상의 마스크를 통과해 조사되는 조건에서 일정속도로 시편을 이송할 때의 어블레이션 가공 결과를 살펴 보고 최종 가공되는 형상을 3D 그래픽 화면에 출력하였다.

2. Visual Fortran과 WIN32 API 함수를 이용한 GUI

2.1 전반적인 접근 방법 검토

Fortran 프로그램을 활용해 비주얼 환경에서 계산과 입출력을 동시에 할 수 있는 통합 프로그램의 구축을 위해서 고려할 수 있는 방안 중에는 C 또는 C++ 과 같이 GUI를 개발하기 쉬운 언어를 활용해 메인 프로그램을 개발하고, 계산에 사용되는 Fortran 솔버 부분은 DLL (Dynamic Link Library)로 구축하고 이를 호출하는 형태, 즉 Mixed-Language Programming의 방법이 있다.[7,8] 그러나 서로 다른 프로그래밍 언어에 대해 이해하고 있어야 하며, 다소 번거로운 과정을 거쳐야 한다.

반면에 본 논문에서 고려하듯 GUI와 솔버 부분을 모두 Fortran으로 구축하면 처음부터 하나의 언어로 구축된 통합 프로그램의 개발이 가능하다. 특히 그래픽 워크스테이션이 아닌 범용의 PC 환경에서 구동되는 프로그램을 위해서는 Visual Fortran과 같은 개발 툴을 활용할 수 있으며 비주얼 스튜디오에 통합해 개발이 가능하다. 입력은 대화상자를 활용하고 계산결과의 3D 출력은 OpenGL 라이브러리를 활용한다.[9,10]

대화상자의 구축은 Visual Fortran의 QuickWin Runtime Library[9]를 이용하면 비교적 간단하지만[11], 이 방법은 MS의 WIN32 API (Application Programming Interface)함수를 활용하는 전형적인 윈도우 프로그래밍이 아닌 Visual Fortran에 특화된 라이브러리를 사용하고 있으며, 특히 OpenGL과 같이 강력한 3D 그래픽 툴을 사용할 수 없다.

OpenGL은 기종에 관계없이 작동하고 실행속도도 매우 빠른 모델링 라이브러리이지만,[12] OpenGL 자체는 윈도우(창) 관리, 사용자와의 상호 작용, 파일 입출력 등의 함수를 갖지 않으므로 윈도우 운영체제에서의 OpenGL 프로그래밍은 특정한 초기화 및 사용 순서를 필요로 한다. 또한 비주얼 스튜디오 버전2008부터는 GLAUX 라이브러리가 삭제되어 간단하게 OpenGL 창을 등록하고 열 수 있는 방법이 없으며, 오직 전형적인 WIN32 API 함수를 이용해 정식으로 윈도우 창을 등록하고 실행해야 한다.

윈도우 창을 등록하고 실행하는 부분과 대화상자를 만들고 실행하는 부분은 C나 C++ 언어로 된 자료[13-15]는 풍부하지만, Visual Fortran 환경에서 실행하기 위한 자료는 관련 언어의 오프라인 및 온라인 매뉴얼,[16,17] 윈도우 Fortran 관련 일부 저서[9,10,18-20] 및 Visual Fortran을 설치한 후 온라인 환경에서 볼 수 있는 샘플 프로그램[21] 정도이다. 이 중에서 유용한 참고 자료로 관련 샘플 소스[21]나 Lawrens의 저서[10]를 들 수 있으며, 특히 Lawrens의 저서는 Visual Fortran을 활용해 윈도우 프로그래밍을 수행하고, OpenGL 라이브러리를 응용하는 내용에 대해 체계적으로 기술하고 있으므로 필수 참고자료이다. 다만 윈도우 프로그래밍에 관해서는 내용이 충분하지 않으므로 다른 자료[13-15]를 부가적으로 참고해야 하였다.

2.2 폴리머 어블레이션 모델링

펄스레이저에 의한 폴리머재료의 어블레이션 가공 모델에 대해서는 선행 연구[4-6]에 정리되어 있으며, 이하에 관련 입력 변수의 이해를 돕기 위해 간단히 서술하였다.

가공 관련 변수는 가공 소재의 물성치, 레이저 파장, 펄스 에너지, 강도분포, 빔의 초점에서의 반경 및 대상 소재로부터의 수직 방향 초점 위치 및 빔 Quality Factor M^2 등과 같은 빔변수가 공통적이다. 이송 가공시에는 가공 속도, 이송 거리, 사용하는 마스크의 형상 변수가 추가된다. 고에너지의 레이저 조사량에서의 펄스당 에칭 깊이를 구하기 위한 해석 모델은 광화학 에칭 효과만 고려하는 JS모델과 빛에 의한 열에칭 효과를 추가적으로 고려하는 SSB 모델을 사용한다. 펄스당 에칭 깊이, d는 레이저 조사량 H (J/cm^2)의 함수 형태로 나타낼 수 있다. 모델에 사용되는 소재의 물성치 관련 변수는 실험결과를 Curve Fitting해 얻은 값을 사용한다.

해석은 빔과 소재의 상대적인 이송 속도가 0인 고정 가공, 일정한 방향으로 동일 속도로 이송하는 이송 가공의 두 가지로 나누어 진행하였다. 고정 가공시에는 해석 영역의 중앙에 레이저빔이 조사되는 것으로 간주하였으며, 이송 가공시에는 일정한 강도분포의 평행빔이 마스크를 통과해 시편에 조사되며 일정한 이송속도로 정해진 거리만큼 이송하는 것으로 간주하였다. 레이저빔의 펄스 반복율, 마스크의 형상과 이송 속도, 이송거리를 고려해 임의의 위치에서의 조사되는 빔의 펄스 수를 구할 수 있으며 가공되는 깊이를 구하게 된다. 따라서 마스크의 형상 관련 변수값이 필요하며 여기서는 직사각형, 이등변삼각형 및 반원형 마스크를 대상으로 하고 있다. 상세한 내용은 참고문헌에 정리되어 있다.

2.3 WIN32 API함수를 이용한 윈도우 프로그래밍

개발된 프로그램은 앞에서 기술한 바와 같이 비주얼 스튜디오와 Visual Fortran을 이용해 구축하였다. 비주얼 스튜디오는 2015버

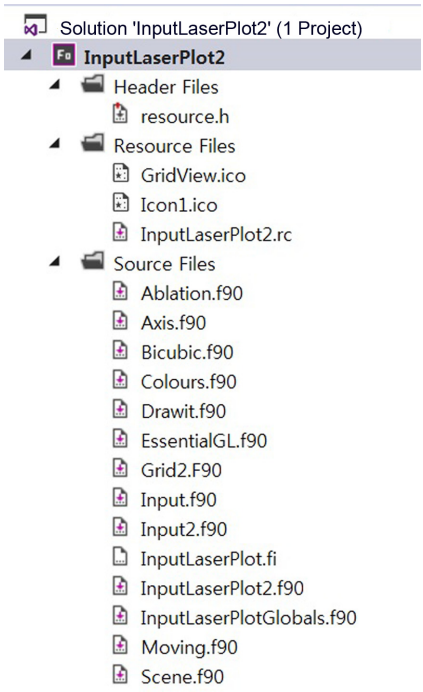

Fig. 1 Visual Studio screen showing files included

전을 사용하였으며, OpenGL창을 등록하고 실행하기 위해서 WIN32 API 함수와 Wgl 함수(윈도우 운영체제에서 OpenGL을 이용하기 위해 추가된 함수)를 사용하였다.

Fig. 1에 프로그램에 포함되어 있는 소스 파일, 리소스 파일 및 헤더 파일의 구성을 보인다. 메뉴 화면과 입력변수를 위한 대화상자는 리소스 파일(*.rc)에 구현되어 있으며 저장하면 자동으로 resource.h의 헤더 파일이 생성된다. Visual Fortran에서 리소스 파일을 사용하기 위해서는 프로그램 앞 부분에 resource.fd 파일을 포함시켜야 하는데, *.fd 파일을 생성하기 위해서는 헤더 파일의 속성창의 Custom Build Step에서 다음과 같이 입력하여야 한다.[17]

Command line option: deftofd resource.h resource.fd
Output option: resource.fd

프로그램의 소스 파일은 Fortran90형식의 프로그램이며 확장자 *.f90 의 이름을 갖는다. 데이터의 교환을 위한 전역변수(Global Variables)의 선언을 위한 InputLaserPlotGlobals.f90과 함수의 인터페이스 선언을 위한 InputLaserPlot.fi의 파일이 추가되어 있다. GUI 기본 프로그램은 InputLaserPlot2.f90으로 윈도우 관련 메인 창을 등록하고 실행하는 WinMain 함수와 메시지 처리를 담당하는 MainWndProc 함수, 프로그램의 도움말 관련 대화상자(AboutDlg)의 메시지 처리를 담당하는 AboutDlgProc함수로 구성되어 있다. 솔버에 해당하는 부분은 고정가공계산을 담당하는 Ablation.f90 과 이송가공을 담당하는 Moving.f90, 레이저빔이 조사되는 매끄러운 곡면을 표현하기 위해 3차 곡면패치(Bicubic

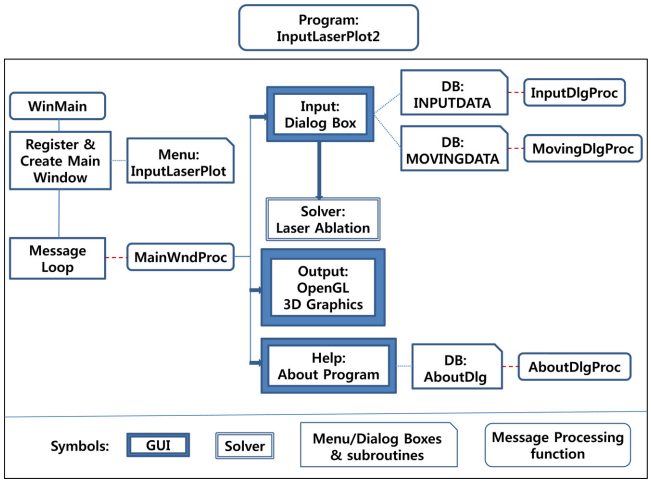

Fig. 2 Flowchart of simulation program combining GUI and solver routines

Surface Patch) 로 모델링하는 Bicubic.f90으로 이루어진다.

계산에 필요한 입력변수를 받는 부분은 고정가공입력을 위한 “INPUTDATA” 대화창을 화면에 나타내고 대화상자의 특정 버튼이나 박스에 변화가 있을 때 값을 갱신하는 등의 처리를 담당하는 Input.f90, 이송가공입력을 담당하는 “MOVINGDATA” 대화창을 나타내고 처리를 담당하는 Input2.f90으로 구성되어 있다. 소스 파일 중 나머지 파일들은 OpenGL 3D 출력과 연관된 파일이다.

개발된 프로그램의 전체적인 구성 및 흐름도는 Fig. 2와 같다. 전체 프로그램은 구조화된 개별 파일로 분리해 작성한 후 결합하였으며 이러한 모듈 방식은 부분적으로 수정하거나 디버깅할 때 매우 유용하다.

프로그램을 실행하면 다음 순서에 따라 진행된다.

(1) 리소스 파일에 구현한 내용에 따라 Main Window 창을 등록하고 실행한 후 메시지를 기다리게 된다. 메시지가 발생하면 내용에 따라 MainWndProc함수에서 적절하게 메시지를 처리하게 된다.

(2-1) 사용자가 메뉴 화면에서 ‘변수 입력(DataInput)’을 선택하면 관련 대화 상자(INPUTDATA 또는 MOVINGDATA)를 화면에 나타내고 변수값을 입력할 수 있으며, 입력이 완료되면 대화상자의 ‘InputDone/Execute’ 버튼을 클릭해 계산 솔버로 데이터를 넘겨주게 된다.

(2-2) 메뉴 화면에서 ‘Help-About Program’을 선택하면 간단한 프로그램 소개/도움말이 새로운 대화 상자 내에 출력되며, 확인 버튼을 클릭하면 다시 메인 화면으로 돌아오게 된다.

(3) 솔버에서는 고정가공 또는 이송가공 관련 계산을 수행하고 3D 형상의 데이터를 저장한 후 화면에 ‘계산종료’ 메시지를 나타내고 사용자의 지시를 기다린다.

(4) 메뉴 화면에서 ‘Plot3D Shape’를 선택하면 가공된 형상의 3D 서피스/메시 를 메인 창에 OpenGL 그래픽스 루틴을 이용해 나타낸다. 이 때 화면은 키보드의 ‘Space Bar’를 눌러 자동 회전하도록 하거나 또는 마우스 버튼을 이용해 Z축 또는 X축에 대해

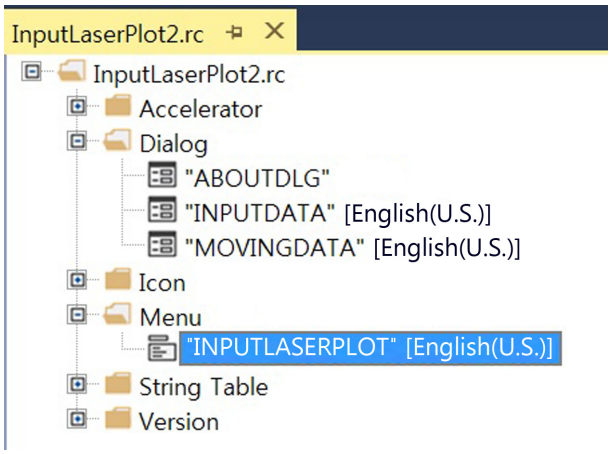

Fig. 3 Construction of program resource file

수동으로 회전시키면서 전체적인 형상을 살펴 볼 수 있다.

(5) 입력 변수의 대화상자는 별도로 'EXIT', 'CANCEL'버튼이나 종료 버튼을 실행하지 않는 한 화면에 남아 있도록 해 입력 변수와 계산 결과를 비교해 볼 수 있다. 도움말 화면을 제외한 모든 화면에서 '변수 입력' 및 '3D 형상 출력'을 다시 실행할 수 있으며 변수를 수정해 결과를 새로 확인할 수 있다.

WinMain 함수 및 MainWndProc 함수는 Lawrence의 자료[10]를 참고해 작성하였으며, MainWndProc 함수에서 실제 GUI부분에 해당하는 입력 대화상자를 제어하고 3D그래픽스 화면을 출력하는 역할을 담당한다.

리소스 파일은 Fig. 3과 같이 구성되어 있으며, 특히 메인 창의 메뉴 화면과 입력변수를 위한 대화상자가 중요하다. Fig. 3의 대화상자는 세 가지인데 하나의 모달 대화상자와 두 개의 모달리스 대화상자로 구성하였다. 메뉴-Help를 선택할 때 화면의 가운데에 간단한 도움말 대화상자를 출력하기 위한 대화상자 ABOUTDLG는 일반적인 대화상자 형식대로 모달(Modal) 대화상자로 구성해 대화상자를 닫기 전에는 다른 선택이 불가능하다.

가공 데이터의 입력을 위한 대화상자 INPUTDATA와 MOVINGDATA는 메인 창에서 입력 대화상자와 메인 창의 3D 그래픽 출력을 동시에 나타낼 수 있도록 모달리스(Modeless) 대화상자로 구성하였다. 모달 대화상자를 만들기 위해서는 DialogBoxParam함수를, 모달리스 대화상자는 CreateDialogParam 함수를 사용한다. 참고로 Help-About 메뉴를 선택할 경우 모달 대화상자를 만드는 프로그램과, AblationNoMoving 메뉴를 선택할 경우의 모달리스 대화상자를 만드는 프로그램의 해당 부분은 아래와 같다.

```
case(WM_ABOUT) !모달 대화상자
 lpszName="ABOUTDLG"
 iret=DialogBoxParam(ghInstance,&
    LOC(lpszName),hWnd,LOC(AboutDlgProc),0)
 return
```

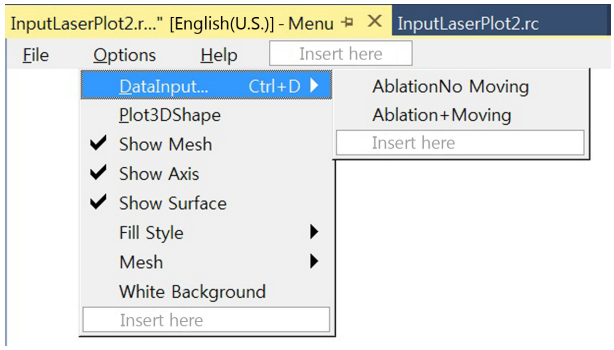

Fig. 4 Screen capture showing program menu

Output: OpenGL 3D Graphics Routines

Subroutines	contained in File	Explanation
SetDCPixelFormat	EssentialGL.f90	Set the pixel display format for OpenGL
MakeFont	Scene.f90	Create bitmap font display list
Contour_Colour	Colors.f90	Create the rainbow and grayscale colors
GridData*	Grid2.f90	Define the coordinates for a 2D X-Y grid
CreateAxis*	Axis.f90	Create a display list defining the X, Y and Z axis with appropriate labels
CreateScene*	Scene.f90	Create display lists for axis, mesh and surface
DrawScene	Drawit.f90	Draw grid using display lists according to user selections
maxmin	Drawit.f90	Obtain the maximum and minimum values for a given data set

* : should be modified with user's data

Fig. 5 Subroutines for OpenGL 3D Graphics Output

```
case(WM_IDM_INPUT) !모달리스 대화상자
 lpszName="INPUTDATA"
 ghDlgModeless=CreateDialogParam(ghInstance,&
    LOC(lpszName),hWnd,LOC(InputDlgProc),0)
 return
```

메뉴 화면은 Fig. 4와 같이 구성하였으며, 메인 창의 메뉴-Options- DataInput에서 AblationNoMoving 또는 Ablation+Moving을 선택해 고정가공 또는 이송가공에 해당하는 대화상자를 화면에 나타낼 수 있다.

계산결과를 3D 그래프로 화면에 나타내주는 후처리 담당 부분은 Lawrens[10]를 참고하였으며, Fig. 5에 보인다. Fig. 5에서 사용자의 데이터에 맞게 프로그램을 수정해야 하는 부분은 * 표시된 루틴들이며, 그 외의 루틴은 자료의 소스 코드를 그대로 활용할 수

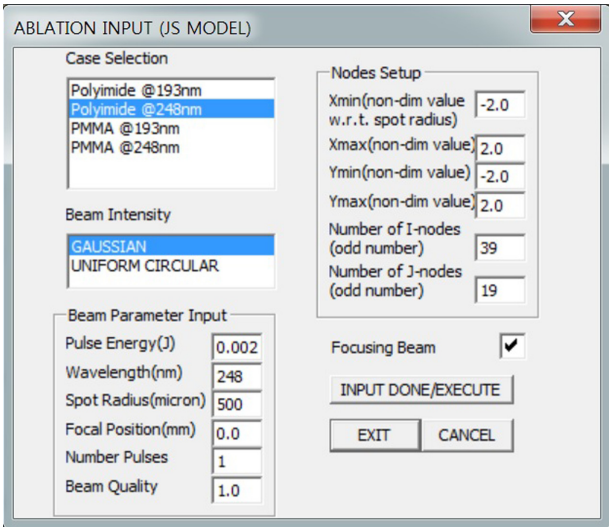

Fig. 6 Dialog box for data input (no moving case)

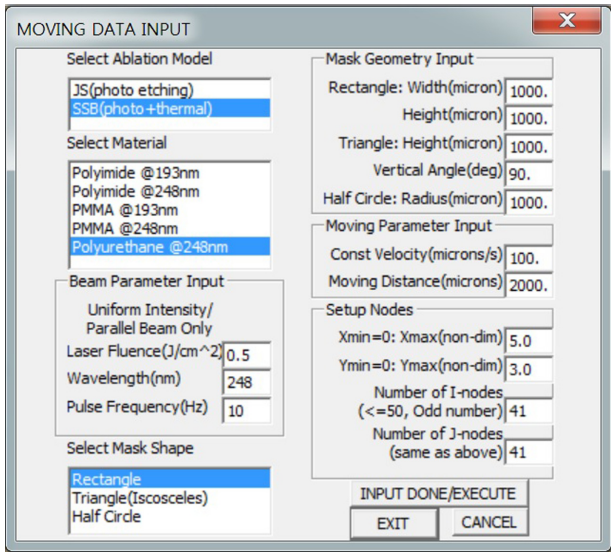

Fig. 7 Dialog box for data input (moving case)

있다. 본 연구에서는 사용자의 설정값을 반영하도록 그리드 데이터 생성부, Z-Data생성부, 축의 눈금표시부, 3D 형상의 X, Y, Z 축 표시범위를 일부 수정해 작성하였다.

2.4 입력 대화창

대화상자의 전체적인 형태를 구성하기 위해서는 Visual Fortran의 리소스 편집기에서 대화상자(Dialog) 에 새로운 이름을 추가하고 화면에서 편집 작업을 통해 필요한 리스트박스, 에디트 컨트롤, 버튼, 체크박스 등을 배치하여 완성한다. 대화상자를 포함한 전체 리소스를 프로그램에서 활용하기 위해서는 Include 'resource.fd' 명령을 입력한다. 프로그램 시작부에는 윈도우 함수를 Fortran에서 사용하도록 정의된 use IFWINA명령을 사용한다.

대화상자의 특정 위치에 초기값을 나타내거나 프로그램 실행 후 사용자가 입력한 값을 읽기 위해서 WIN32 API 함수를 활용하였다. 리스트박스를 이용하여 구성하였으며 원하는 문자열을 입력하기 위해서는 SendMessage함수에 LB_ADDSTRING, 선택된 항목의 순서를 읽어오기 위해서는 LB_GETCURSEL을 활용한다. 프로그램의 해당 부분은 다음과 같다.

```
hlist1=GetDlgItem(hDlg,IDC_LIST1)
iret=SendMessage(hlist1,LB_ADDSTRING,0,&
            LOC('GAUSSIAN'C)) !초기값 입력
...
iret=SendMessage(hlist1,LB_GETCURSEL,0,0) !선택값읽기
number=iret+1
```

한편 에디트창에 필요한 입력 데이터의 값을 설정할 때는 SetDlgItemText, 실수 데이터를 읽기 위해서는 GetDlgItemText와 Chartoreal함수를[21], 정수데이터를 읽기 위해서는 GetDlgItemInt 함수를 활용한다. 아래에 해당 프로그램을 보인다.

```
iret=SetDlgItemText(hDlg,IDC_EDIT_XMAX,'2.0'C)
iret=GetDlgItemText(hDlg,IDC_EDIT_XMAX,text,100)
XMAX=chartoreal(LOC(text)) !for real number
il=GetDlgItemInt(hDlg,IDC_EDIT_INODES,LOC(result),&
                .TRUE.) !for integer number
```

Fig. 6에 고정가공의 변수 입력을 위한 대화상자의 완성된 형태를 보인다.

현재는 광화학 에칭만을 고려한 JS모델이 적용되어 있다. 사용자는 다음 순서에 따라 변수값을 선택 및 입력한다.

(1) 대상 소재 선택
(2) 빔강도 분포 선택
(3) 빔변수 입력
(4) 계산 영역(XY 그리드) 및 절점 수 입력
(5) 집속빔 또는 평행빔 선택

입력이 완료되면 INPUT DONE/EXECUTE 버튼을 클릭해 계산 솔버로 데이터를 넘기게 된다.

한편 이송가공의 변수 입력을 위한 대화상자는 Fig. 7과 같이 구성되어 있으며, 사용자는 다음 순서에 따라 변수값을 선택 및 입력한다.

(1) 어블레이션 해석 모델 선택
(2) 대상 소재 선택
(3) 빔변수 입력
(4) 마스크의 형상 변수 입력
(5) 이송 가공 변수 (이송속도 및 이송거리) 입력
(6) 계산 영역(XY 그리드) 및 절점 수 입력

이송가공의 경우 균일강도 분포의 평행빔을 사용하는 조건을 전제로 하고 있다. 또한 보다 강한 펄스빔의 입력을 고려한 SSB 모델을 사용할 수 있도록 구성되어 있으며 따라서 해석 모델을

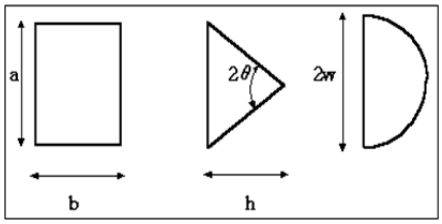

Fig. 8 Shapes and dimensions of masks (moving case)

두 가지 중에서 선택할 수 있다.[6] 준삼차원 미소형상의 가공을 위해 사용하는 마스크는 Fig. 8에 보이는 세 가지의 형상을 갖는다. 직사각형(가로b, 세로a), 이등변삼각형(가로높이h, 꼭지각 2θ), 반원형(반지름w)의 세가지 중에서 선택하며 각각의 경우에 필요한 형상 변수값도 적절하게 입력한다.

3. Visual Fortran과 OpenGL함수를 이용한 윈도우에서의 3D 그래픽 출력

3.1 OpenGL함수를 이용한 그래픽창의 설정 및 3D출력

Visual Fortran에서 OpenGL함수를 사용하기 위해서는 먼저 프로그램 시작부에 use IFOPNGL 또는 use DFOPNGL명령을 사용한다.[10,12] Fortran에서 사용하는 OpenGL 식별자(Identifiers)는 C 언어에서의 식별자와 유사하며, 접두사 gl 이 fgl 로, GL 은 FGL 로 변경된다. 다음과 같이 운영체제로부터 WM_CREATE 메시지를 받을 때 OpenGL 렌더링 컨텍스트를 활성화하도록 배치하고, WM_DESTROY 메시지를 받을 때 정리 코드를 배치한다.[10,12] 아래의 SetDCPixelFormat 함수는 픽셀포맷을 선택하고 지정한다.

```
case (WM_CREATE)
 hdc=GetDC(hwnd)!장치 컨텍스트 보관
 call SetDCPixelFormat(hdc)!픽셀포맷 선택
 hrc=fwglCreateContext(hdc)!렌더링컨텍스트 생성
 bret=fwglMakeCurrent(hdc,hrc)!활성화
 return
case(WM_DESTROY)
 bret= fwglMakeCurrent(null,null)!렌더링컨텍스트 해제
 bret=fwglDeleteContext(hrc)!제거
 bret=DeleteDC(hdc)
 callPostQuitMessage(0)!종료메시지
 return
```

일반적으로 OpenGL함수를 이용해 창에 그림을 그릴 때는 WM_PAINT메시지가 전달될 때마다 윈도우의 BeginPaint와 EndPaint함수 사이에 OpenGL함수로 구성된 서브루틴을 배치해 창이나 창의 내용물을 그리도록 한다. 즉,

```
case (WM_PAINT)
 iret=BeginPaint(hwnd,ps)
 call DrawScene(hdc,nAngleX,nAngleZ) !Plot routine
 bret=EndPaint(hwnd,ps)
 return
```

다른 메시지가 전달될 때 창을 다시 그릴 필요가 있을 경우에는 InvalidateRect함수를 사용하면 간단하다. 즉,

```
case(...)
 ...
 bret=InvaliateRect(hwnd,NULL_RECT,.FALSE.)
 return
```

OpenGL에서 트루타입 글꼴을 사용하기 위해서는 Wgl함수를 사용할 수 있다. 비트맵 형식의 2D 텍스트 출력을 위해서는 wglUseFontBitmaps, 3D 텍스트효과를 위한 3D 글꼴을 만들기 위해서는 wglUseFontOutlines 함수를 사용한다.

기본적인 설정이 완료되었으면, OpenGL함수를 이용해 다양한 3D 효과를 갖는 그래픽을 만들 수 있다. 예를 들어 메인 창의 크기가 변화하는 경우 WM_SIZE메시지가 발생하며 이때 창 크기에 맞게 다시 설정하기 위해서는 다음 순서를 따른다. 먼저 창의 폭과 높이를 새로 구하고 종횡비를 계산한다. 3차원 형상을 나타내기 위해 투영변환을 설정하고 투시투영좌표계를 다시 설정한 후에 뷰포트 창을 변화한 창 크기에 맞도록 다시 설정한다. 즉,

```
case (WM_SIZE)
 glnWidth=LOWORD(lParam)
 glnHeight=HIWORD(lParam)
 gldAspect=DBLE(DBLE(glnWidth)/DBLE(glnHeight))
 call fglMatrixMode(GL_PROJECTION) !투시투영좌표계
 call fglLoadIdentity()
 call fgluPerspective(DBLE(ViewAngle),gldAspect,&
 DBLE(1.0) &!distance to near clipping plane
 DBLE(100.0)) !distance to far clipping plane
 call fglViewPort(0,0,glnWidth,glnHeight)
 return
```

프로그램에 반영된 3D 출력 루틴은 Fig. 5에 보인 바와 같이 Lawrens의 자료를 기반으로 그리드 데이터 생성부, Z-Data생성부, 축의 눈금표시부, 3D 형상의 X, Y, Z축 표시범위를 수정해 작성하였다. 추가적으로 WM_TIMER메시지를 이용해 사용자가 원할 경우 Space Bar를 이용해 3D형상을 Z축을 중심으로 자동으로 회전/정지하도록 하였다. 즉,

```
logical::animate=.FALSE.
case (WM_CREATE)
```

```
 nTimer=SetTimer(hwnd,1,50,)
 return
case (WM_TIMER)!50밀리초마다 z축 중심 자동회전
 if(animate) nAngleZ=nAngleZ+2
 if(nAngleZ>=360) nAngleZ=0
 bret=InvalidateRect(hwnd,NULL_RECT,.FALSE.)
 return
case(WM_CHAR)!스페이스바를 이용해 회전/정지선택
 select case(wParam)
 case(ichar(' '))
 animate=.NOT.animate
 return
 end select
 return
```

메시 형태의 데이터를 3D형상으로 화면에 출력하는 부분은 세 부분으로 나누어진다.

(1) 고정된 메시형태의 2차원 배열 x(ix,iy), y(ix,iy) 그리드 데이터를 1차원 배열 xp(i), yp(i)로 읽어들이고, 직사각형 요소의 네 꼭지점 Grid(Icount, 1:4)과 연결시킨다.

(2) 위의 데이터를 이용해 fglBegin(GL_POLYGON), fglColor3f, fglVertex3f(xp,yp,zp)..., fglEnd() 명령어로 네개의 꼭지점을 연결하는 직사각형을 연속해서 그리고, fglNewList(PlotList, GL_COMPILE), ...,fglEndList() 명령어로 이미 컴파일링을 끝낸 리스트로 저장해둔다. 두 번째 단계까지는 초기에 한번만 실행하면 된다.

(3) 마지막 세 번째 부분은 매번 다시 그리는 명령을 수행하는 부분으로 이미 만들어진 리스트를 가져와 3D 공간상에 평행이동 변환 및 회전변환을 거친 형태로 출력한다. 이 세 번째 단계의 프로그램은 다음과 같다.

```
 subroutine DrawScene (hdc,nAngleX, nAngleZ)
 ...
 call fglClearColor(0.9,0.8,0.6,0.5)
 call fglClear(IOR(GL_COLOR_BUFFER_BIT,&
                   GL_DEPTH_BUFFER_Bit))
 call fglMatrixMode(GL_MODELVIEW)
 call fglLoadIdentity()
 call fglTranslatef(0.0,0.0,-25.0)!이동변환
 call fglRotatef(-90.0,1.0,0.0,0.0)!초기회전
 call fglRotatef(REAL(nAngleX),1.0,0.0,0.0)!x축중심회전
 call fglRotatef(REAL(nAngleZ),0.0,0.0,1.0)!z축중심회전
 call fglCallList(PlotList)!미리 저장해둔 리스트가져옴
 bret=SwapBuffers(hdc)!더블버퍼의 OpenGL창 사용시
 return
```

3.2 레이저 어블레이션 가공 계산결과의 3D출력

3D 형상의 수치 계산 결과는 기발표된 자료의[4-6] 결과와 동일

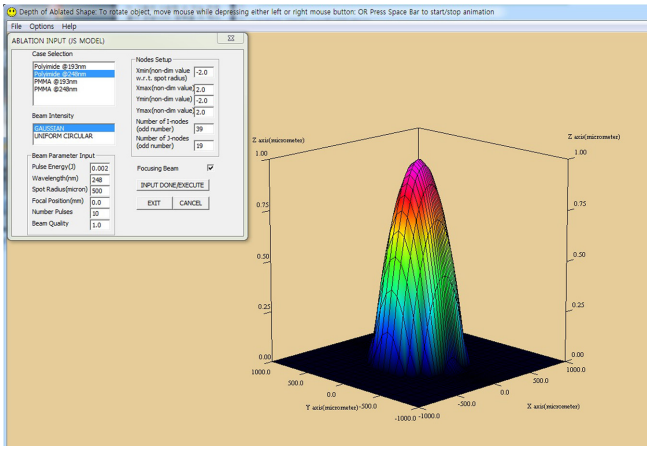

Fig. 9 Screen capture during execution of program (no moving case)

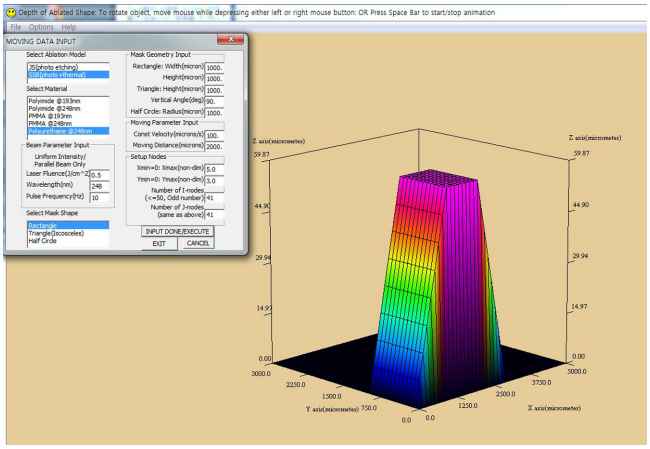

Fig. 10 Screen capture during execution of program (moving case: rectangular mask)

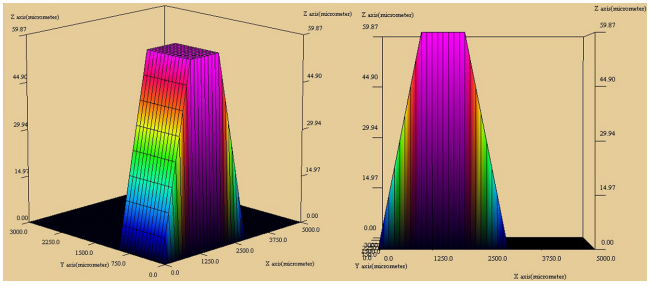

Fig. 11 Screen capture of etched shape using rectangular mask(moving case)

하다. 본 논문에서는 계산된 수치 데이터를 OpenGL 함수를 이용해 메인 윈도우에 정지 또는 회전하는 3D 그래픽 화면으로 출력하는데 주안점을 두고 있으며 아래에 몇 가지 대표적인 경우의 결과를 보인다.

Fig. 9은 고정가공의 경우 Gaussian 강도분포, 펄스 에너지 = 0.002 J, 펄스수 = 10, 초점으로부터의 거리 W = 0 mm, M^2=1,

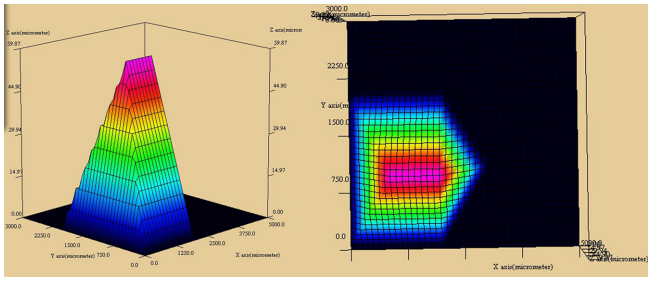

Fig. 12 Screen capture of etched shape using triangular mask (moving case)

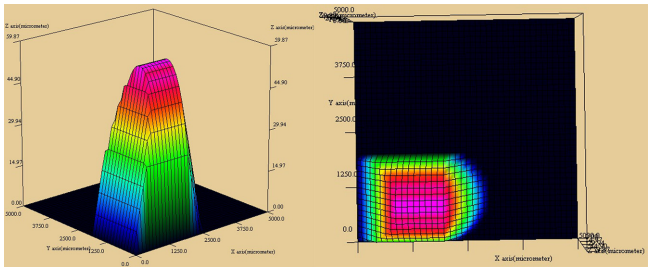

Fig. 13 Screen capture of etched shape using half-circular mask (moving case)

대상소재는 Polyimide, 파장 = 248 nm, 빔반경 = 500(μm)일 때의 실행 화면이다. 입력 대화창에서 빔변수와 소재를 변화시켜가며 결과에 미치는 영향을 쉽게 3D그래픽화면으로 파악할 수 있다.

이송가공은 대표적으로 SSB모델, 대상소재는 Polyurethane, 레이저빔 조사도 = 0.5(J/cm^2), 파장 = 248(nm), 펄스빔의 주파수 반복율 f = 10(Hz), 이송속도 = 100(μm/s), 이송거리 = 2000(μm)의 값을 사용하였다. Fig. 10에 직사각형 마스크(1000 μm × 1000 μm)를 사용할 때의 프로그램 실행 화면을 보인다.

이송 시작부위와 종료부위 형태의 비대칭 여부를 비교해 보기 위해 가공된 형상의 3D이미지와 정면에서 바라본 모습을 Fig. 11에 나타내었다.

동일한 이송가공조건에서 이등변삼각형의 마스크를 사용하고 높이 = 1000 μm, $2\theta = 90^{o}$ 인 경우의 계산결과를 Fig. 12에 3D 이미지와 위에서 내려다본 모습을 비교해 나타내었다. 시작부위와 종료부위의 가공된 형상이 비대칭적임을 확인할 수 있다. 시작부분은 균일한 경사를 가지고 기울어져 있으며 종료부분은 마스크의 삼각형상을 통과한 빔에 의해 가공된 모습을 확인할 수 있다.

동일한 이송조건하에 반경 $w = 1000$ μm의 반원형 마스크를 사용할 때의 계산결과를 각각 3D 이미지와 위에서 본 형상으로 Fig. 13에 나타내었다.

역시 시작부위와 종료부위의 가공된 형상이 비대칭적임을 확인할 수 있다. 시작부분은 일정한 각도로 경사진 형태이며 종료부분은 마스크의 반원 형상을 통과한 빔에 의해 가공된 모습을 확인할 수 있다.

3.3 Visual Fortran 통합 프로그램 구축 결과에 대한 고찰

앞 절에서는 정지 화면만을 보이고 있지만, 실제 프로그램의 실행시에는 비대칭적인 형상의 3D 가공된 결과 형상을 화면에서 마우스로 회전시키거나 또는 자동으로 회전하는 형태로 쉽게 확인 가능하다. 복잡한 형상의 계산 결과의 경우에도 X, Y, Z 메시 형태로 얻어지는 데이터는 쉽고 빠르게 3D 이미지로 화면에 구현할 수 있다.

Visual Fortran을 이용해 WIN32 API함수 및 OpenGL로 GUI를 구축한 결과와 윈도우 전용 C언어를 활용해 GUI를 구축한 결과와의 장단점을 비교해 보면 다음과 같다.

(1) C언어를 이용해 쉽게 구현되는 몇 몇 API 함수는 Visual Fortran에서 제대로 작동하지 않아 다른 함수를 이용해야 하는 경우도 있었다. 대화상자의 콤보박스 대신에 리스트박스를 이용하는 것으로 대치하였으며, 실수형 데이터를 읽기 위해서는 GetDlgItemText와 Chartoreal함수를[21] 활용하였다.

(2) 윈도우 창에 비트맵 이미지를 출력하는 기능은 C언어로 구축된 GUI 또는 Visual Fortran의 QuickWin 라이브러리 에서는 문제없이 작동하지만, WIN32 API 함수를 이용할 때는 작동하지 않았다.

(3) GUI와 계산 솔버가 동일한 프로그래밍 언어인 Fortran으로 작성되어 통합된 프로그램을 구축할 수 있는 점은 매우 큰 장점이다. 사용자의 목적에 맞게 입력부를 대화상자로 구성하고 출력부의 일부 서브루틴을 수정하면 기존의 Fortran Solver를 바로 활용해 비주얼 환경의 입출력부를 갖는 윈도우용 프로그램을 용이하게 구축할 수 있다.

4. 결론

본 논문에서는 Visual Fortran과 OpenGL을 이용해 기존의 Fortran 소스 프로그램을 이용하면서 윈도우 환경에서 유용한 통합 프로그램을 구축하였다. 결과를 요약하면 다음과 같다.

(1) WIN32 API 함수를 이용해 입력 변수를 화면에서 용이하게 입력할 수 있는 대화상자를 구축하였다.

(2) OpenGL을 이용해 계산결과를 정지 화면 또는 회전하는 3D 그래픽 화면으로 출력하도록 하였다.

(3) 이상의 윈도우용 GUI구축과정을 Fortran 계산 솔버가 만들어져 있는 레이저가공 시뮬레이션 프로그램과 결합해 하나의 통합 프로그램을 개발하였다. 사용자는 데이터 입력 창을 통해 화면에서 도움을 받아가며 편하게 설정 변수를 입력할 수 있으며, 계산 결과는 3D그래픽으로 화면에 출력되므로 이해를 도울 수 있다.

Fortran 언어에 친숙한 과학 및 공학 분야의 사용자들에게 이처럼 비주얼 환경에서 입출력을 제어할 수 있는 모듈화된 GUI를 이용하면 지금까지 개발된 많은 응용분야의 Fortran 소스 프로그램의 솔버를 윈도우 환경에서 효율적으로 사용할 수 있어 그 응용 가능성이 매우 클 것으로 생각된다.

ACKNOWLEDGEMENT

이 논문은 2006년도 동국대학교 연구년 지원에 의하여 이루어졌음.

REFERENCES

1. Visual Fortran Users Network, http://cafe.daum.net/VFUN (Accessed 16 OCT 2017)
2. Intel, “Intel Parallel Studio 2017 Composer Edition for Fortran and C++ Windows,” 2016.
3. Microsoft, “Microsoft Visual Studio 2015,” 2015.
4. Bang, S. Y., “Effects of Beam Parameters on Excimer Laser Ablation,” J. Korean Soc. Precis. Eng., Vol. 22, No. 7, pp. 38-46, 2005.
5. Bang, S. Y. and Yoon, K. K., “Modeling of Polymer Ablation with Excimer Lasers,” J. Korean Soc. Precis. Eng., Vol. 22, No. 9, pp. 60-68, 2005.
6. Bang, S. Y., Shin, K. S., Yoon, K. K., and Whang, K. H., “Modeling of Laser Micromachining of Quasi-Three-Dimensional Shapes,” J. Korean Soc. Precis. Eng., Vol. 22, No. 7, pp. 79-87, 2005.
7. Kyung, J. W. and Kwon, B. C., “Visual Fortran Application Guide - Fortran Function and Mixed-Language Programming,” MASO Campus, pp. 256-261, 2013.
8. Kyung, J. W. and Kwon, B. C., “Visual Fortran Application Guide - My Own Graphic Program,” MASO Campus, pp. 254-259, 2013.
9. Etzel, M. and Dickinson, K., “Digital Visual Fortran Programmer’s Guide,” Digital Press, 1999.
10. Lawrence, N., “Compaq Visual Fortran - A Guide to Creating Windows Applications,” Digital Press, 2002.
11. Bang, S. Y., Lee, H. Y., Lim, J. Y., Shin, K. S., Yoon, K. K., et al., “Development of a Simulation Program for Virtual Laser Machining,” J. Korean Soc. Precis. Eng., Vol. 22, No. 7, pp. 54-61, 2005.
12. Wright, R. S. and Lipchak, B., “OpenGL SuperBible,” 3rd Ed., Information Publishing Group, 2005.
13. Petzold, C., “Programming Windows,” Compeople, 5th Ed., 1999.
14. Simon, R. J., “Windows NT Win32 API SuperBible,” Information Publishing Group, 1998.
15. Microsoft, “Windows API Index,” https://msdn.microsoft.com/ko-kr/library/ff8185 (Accessed 16 OCT 2017)
16. Intel, “Intel Fortran Compiler 17.0 Developer Guide and Reference,” Off-Line Manual After Program Installation
17. Intel, “Using Intel Visual Fortran to Create and Build Windows-Based Applications,” https://software.intel.com/en-us/compiler-winapp. (Accessed 16 OCT 2017)
18. Jung, H. Y., Kyung, J. W., and Yun, B. T., “Visual Fortran Easy to Understand,” Dooyangsa, 2011.
19. Kim. D. G., “Windows Fortran,” Goomibook, 2005.
20. Kim, D. G., “Windows Fortran,” 2nd Ed., Goomibook, 2008.
21. Intel, “Intel Software Product Samples and Tutorials,” https://software.intel.com/en-us/product-code-samples (Accessed 16 OCT 2017)