ISBN 978-89-6211-388-4

vtkGraph : GNUPlot 그래프의 VTK 출력 라이브러리

구 기 범 (voxel@kisti.re.kr)

한 국 과 학 기 술 정 보 연 구 원
Korea Institute of Science & Technology Information

목차

1. 서론 ····· 1

2. 기초조사 ····· 2

가. Gnuplot ····· 2

1) 개요 ····· 2

2) Gnuplot의 실행 ····· 2

3) Gnuplot의 그래프 스타일 ····· 3

3. 어플리케이션 특성 분석 ····· 5

가. 로터 동역학 분야에서 사용하는 그래프 ····· 5

1) Sectional data ····· 5

2) Trim solution ····· 8

3) Pitch angle ····· 8

4) Pressure distribution ····· 9

4. 설계 ····· 11

가. 기본 기능 정의 ····· 11

나. 클래스 분할 ····· 11

다. Input / Output ····· 12

5. 구현 ····· 14

가. API 구성 ····· 14

6. 기술적인 문제점 및 해결방법 ····· 23

소스 차례

[소스 2-1] gnuplot의 문법 ····· 2
[소스 2-2] Gnuplot에서의 문자열 사용 ····· 2

[소스 5-1] mode 함수의 문법 ····· 14
[소스 5-2] mode 함수의 사용 예제 ····· 14
[소스 5-3] enable, disable, isEnabed 함수의 문법 ····· 16
[소스 5-4] enable, disable, isEnabled 함수의 사용 예 ····· 16
[소스 5-5] set 함수의 문법 ····· 17
[소스 5-6] set 함수의 사용 예 ····· 17
[소스 5-7] set 함수의 사용 7 ····· 18
[소스 5-8] set 함수의 사용 예 ····· 18
[소스 5-9] set 함수의 사용 예 ····· 19
[소스 5-10] get 함수의 문법 ····· 20
[소스 5-11] get 함수의 사용 예 ····· 20
[소스 5-12] load 함수의 문법 ····· 21
[소스 5-13] set 함수의 사용 예 ····· 21
[소스 5-14] draw 함수의 문법 ····· 22
[소스 5-15] draw 함수의 사용 예 ····· 22
[소스 5-16] reset 함수의 문법 ····· 22
[소스 5-17] reset 함수의 사용 예 ····· 22

[소스 6-1] IPC pipe를 이용한 Gnuplot 실행 ····· 24

1. 서론

KISTI Visualization & Collaborative Environment 팀에서는 가상현실 환경에서의 대용량 데이터 분석을 위한 어플리케이션인 GLOVE를 개발하고 있다.

GLOVE는 통상적인 data visualization 기법을 이용한 정성적 분석 외에도 정량적인 분석이 가능하도록 하기 위해 다양한 형태의 그래프를 보여주는 특징을 갖고 있다. GLOVE를 구성하는 소프트웨어 라이브러리는 VRJuggler와 VTK인데, 이 중 VRJuggler는 가상현실 입/출력 장치를 관장하기 위해 사용하고 VTK는 데이터 visualization의 핵심을 담당한다. 하지만 VTK가 제공하는 그래프는 응용과학자가 이용하기에 충분한 기능을 제공한다고 보기 어렵기 때문에 보다 강력한 그래프 기능을 제공하는 Gnuplot을 이용하기로 했다.

하지만 Gnuplot은 커맨드라인 방식으로 운영되는 독립 어플리케이션이기 때문에 GLOVE의 소스 코드 수준에서 이용하는 데에는 불편함이 따른다. glvGraph는 VTK와 Gnuplot을 연계하는 C++ 라이브러리로, 어플리케이션의 소스 레벨에서 Gnuplot의 그래프를 손쉽게 사용해주는 역할을 담당한다.

이 보고서에서는 glvGraph의 설계 및 구현 사례에 대해 설명한다.

2. 기초조사

가. Gnuplot

1) 개요

Gnuplot은 UNIX, MS Windows, VMS, Atari 등의 다양한 플랫폼에서 사용할 수 있는, 커맨드-라인 중심의 data/function plotting utility다. 이 소프트웨어는 과학/공학 분야에서 함수나 데이터를 visualize할 목적으로 1986년부터 개발된 툴로 여러 종류의 2D/3D 그래프를 지원한다.

2) Gnuplot의 실행

문법)

```
gnuplot {X11OPTIONS} {OPTIONS} file1 file2 ...
```

[소스 2-1] gnuplot의 문법

X11 옵션은 반드시 맨 앞에 나와야 하며, X11을 사용하지 않는 경우에는 무시할 수 있다. 파일은 나열된 순서대로 실행된다. Command-line 인자는 gnuplot 명령을 포함하는 파일의 파일명으로 간주될 수 있으며, 문자열은 반드시 인용 부호로 둘러싸야 한다.

```
load "filename"
cd 'dir'
```

[소스 2-2] Gnuplot에서의 문자열 사용

모든 파일은 load 명령으로 읽어 들이며, gnuplot은 마지막 파일이 처리된 뒤 exit 상태로 빠져나간다. Gnuplot이 실행되면 제일먼저 초기화 파일을 읽는다. 초기화 파일은 Unix 및 AmigaOS에서는 .gnuplot이고 다른 시스템에서는 GNUPLOT.INI다. 초기화 파일을 발견하면 g

nuplot은 초기화 파일에 포함된 명령을 실행하는데, 이 명령은 대부분 터미널 설정이나 자주 사용되는 함수/변수 정의에 관련된 명령이다.

3) Gnuplot의 그래프 스타일

- Boxerrorbars
- Boxes
- Boxxyerrorbars
- Candlesticks
- Circles
- Dots
- Filledcurves
- Financebars
- Fsteps
- Hsteps
- Histograms
- Image
- Impulses
- Labels
- Lines
- Linespoints
- Points
- Steps
- Rgbalpha
- Rgbimage
- Vectors
- Xerrorbars
- Yyerrorbars

- Yerrorbars
- Xerrorlines
- Xyerrorlines
- Yerrorlines
- 3D(surface plots)

3. 어플리케이션 특성 분석

가. 로터 동역학 분야에서 사용하는 그래프

1) Sectional data

가) 데이터 개요

- Sectional data에 대한 그래프는 6가지 변수에 대해 전진 비행은 2종류, 제자리 비행은 1종류의 그래프를 만들어야 한다. (총 6 * 2 + 6 = 18개)
- 원래 sectional normal force와 sectional thrust는 분리해서 생각해야 하지만 이 두 값은 거의 유사하기 때문에 이 실험에서는 두 값을 동일하다고 가정했다. 따라서 두 값의 visualization 결과 역시 동일하다.
- cts**_*******.unw 데이터에서 후처리를 통해 추출할 수 있다. 즉, blade surface에만 존재한다.
- 각 span 위치에서의 sectional normal force 등의 값을 구하려면 같은 span 위치에서 서로 다른 chord 값을 갖는 여러 값들을 모아서 적분해야 하는데, 이는 실제 데이터에서 정확한 위치의 값을 interpolation해서 찾은 뒤 합산하면 된다.

나) Graph (forward flight)

방위각(azimuth angle)의 변화에 따른 값의 변화를 보는 것이 중요하다.

- Sectional normal force (specific span position, distribution)
- Sectional thrust(specific span position, distribution) : 지금은 sectional normal force와 동일한 것으로 가정한다.
- Sectional span force (specific span position, distribution)
- Sectional chord-wise force (specific span position, distribution)
- X-moment (specific span position, distribution)

• Y-moment (specific span position, distribution)

• Z-moment (specific span position, distribution)

다) Graph(hover flight)

방위각이 변화해도 값의 변화가 미미하기 때문에 steady data로 처리한다.

• Sectional normal force (specific span position)

• Sectional thrust (specific span position) : 지금은 sectional normal force와 동일한 것으로 가정한다.

• Sectional span force (specific span position)

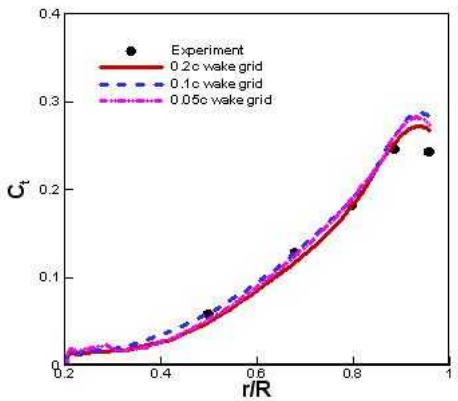
• Sectional chord-wise force (specific span position)

• X-moment (specific span position)

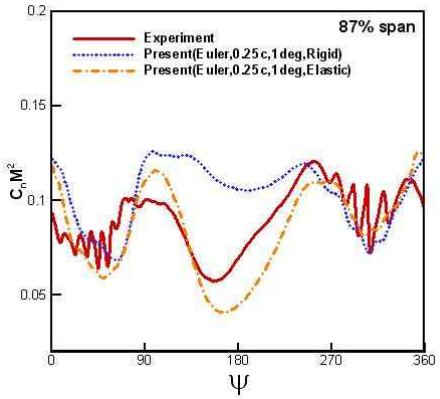
• Y-moment (specific span position)

• Z-moment (specific span position)

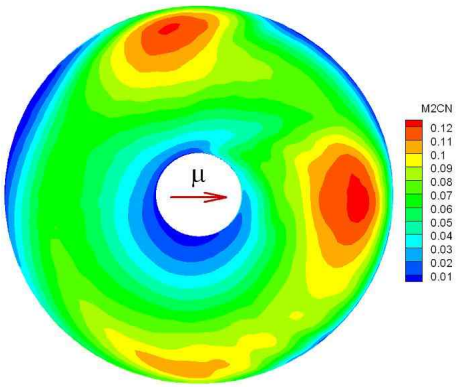
라) Graph samples



[그림 3-1] 정지비행 중의 sectional thrust



[그림 3-2] 특정 span position에서의 sectional thrust



[그림 3-3] 전진 비행 시의 sectional thrust 분포

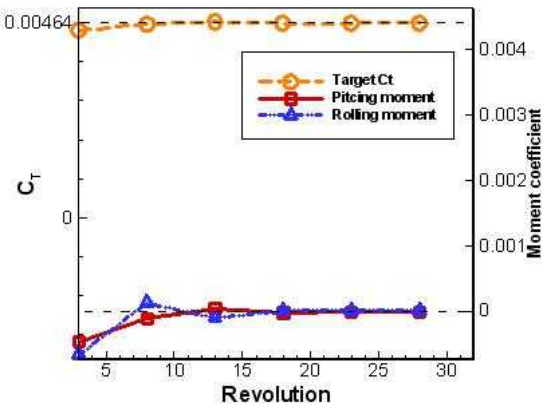
마) Important data fields

- Type of force : sectional normal / span / chord-wise force, X-moment, Y-moment, Z-moment 등
- r/R : Span position
- Azimuth angle : 정확한 위치는 angle과 distance(= span position)로 잡을 수 있겠다.
- (x, y) : Azimuth angle을 2차원 좌표(cartesian)로 변환한 결과
- Force : 실제 데이터
- Hover flight vs. forward fght : 그래프에서 레이블로 표기하면 충

분하다.

2) Trim solution

Trim solution 그래프는 회전에 따른 trim solution 변화를 보여줘야 한다. 여기에는 tangent CT, pitching moment, rolling moment 등이 있다.

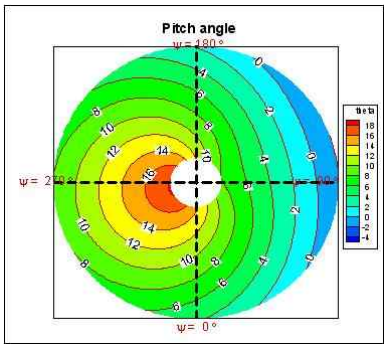


[그림 3-4] 회전에 따른 trim solution의 변화

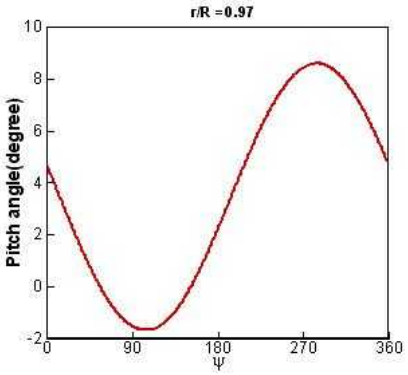
3) Pitch angle

Pitch angle 그래프는 크게 두 가지로 나눠진다.

- Pitch angle(%): 모든 방위각에 대해 특정 blade span position에서의 pitch angle의 변화(선형 그래프)
- Pitch angle distribution: 모든 방위각에 대해 전체 blade span 위치에서의 blade pitch angle의 변화(원형 그래프)



[그림 3-5] Pitch angle over whole span

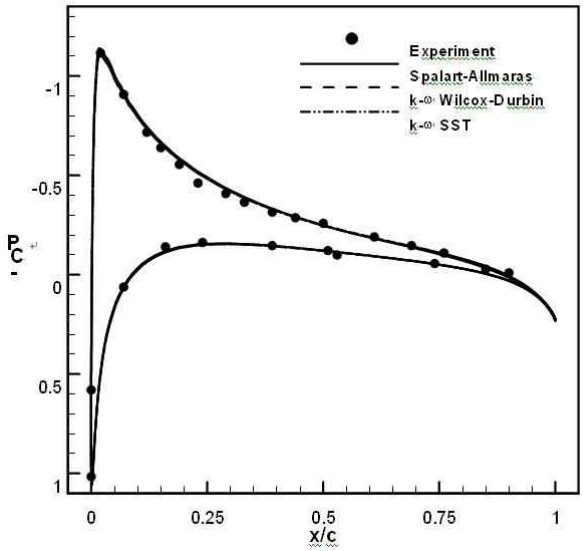


[그림 3-6] Pitch angle at a span position

여기서 중요한 데이터 필드는 회전각(azimuth angle)과 pitch angle로, 이 중 회전각은 원래 각도로 표기하지만 2차 평면상에서의 (x, y) 좌표로 변환해서 사용하고 있다.

4) Pressure distribution

압력 분포 그래프는 실제 계산에서 무차원화(non-dimensionalization)를 이용하므로 KFLOW에서 출력하는 CP값에서 $(r/R)^2$으로 나눈다. 그리고 전진비행 (forward flight)의 경우에는 무차원화 한 변수가 무엇인가에 따라 후처리할 때 전진비의 값이 변경된다. 즉, 위의 식대로 하지 않고 C,.P,,값을 $(r/R + \mu x \sin\psi)^2$으로 나눠야 한다. 이때, 무차원화 한 변수가 무엇인지에 따라 전진비 μ의 값이 바뀐다. 현재 코드에서는 tip velocity로 무차원화 했으므로 그대로 사용하면 되지만 만약 section velocity로 무차원화 했다면 section velocity 값으로 전진비를 나눠줘야 한다.



[그림 3-7] Pressure at a span position

여기서 주의해야 할 사항은 날개의 위/아랫면을 정확히 구분해야 한다는 사실이다.

4. 설계

가. 기본 기능 정의

glvGraph를 설계할 때 아래에 정의한 기능은 모두 지원하도록 했다.

- 데이터 입력 (as 2D array)
- Graph title 설정
- 그래프 종류 설정 : curved graph, disk graph
- X, Y 축 title 설정
- Curved graph의 경우
  - ☞ Curve의 개수 설정 (입력되는 데이터 개수)
  - ☞ 각 curve의 title 설정
  - ☞ 각 curve의 색깔, 라인 종류, 굵기 등 설정
- Disk graph의 경우 팔레트 설정 기능이 있어야 한다.
- 그래프 생성 실행
  - ☞ 현재 설정에 맞춰서 GNUPlot script를 생성하고 GNUPlot을 실행시킨다.
  - ☞ 그래프 이미지 리턴

나. 클래스 분할

- **glvGraph class**

glvGraph는 모든 graph 관련 작업의 핵심으로, Gnuplot의 C++ wrapper다. 이 클래스의 역할은 Gnuplot을 호출해서 그 결과를 이미지로 받으면 VTK 전용의 자료구조(data structure)로 저장해주는 것이다. Gnuplot를 호출하고 그 실행결과를 받는 부분은 IPC pipe을 이용한다.

• Application specific classes

glvGraph 클래스는 서로 다른 종류의 어플리케이션에서 공통으로 사용할 수 있기 때문에 특정 어플리케이션을 위한 클래스는 상황에 맞게 별도로 구현하는 것이 바람직하다고 생각했다. 예를 들어서 로터 동역학을 위한 어플리케이션이라면 glvGraphRotor, 천문학의 경우에는 glvGraphAstro 등등을 따로 구현해서 각 어플리케이션의 특징을 반영하는(sectional normal force …….) 함수, 변수를 구현하는 것이다.

다. Input / Output

입력 데이터로 VTK의 자체 자료구조를 사용할 때와 배열을 사용할 때의 차이점은 다음과 같다.

- VTK 데이터 : steady일 경우 vtkImageData, unsteady일 경우 vtkImageData (3D) 또는 vtkTemporalData로 처리한다.
- Array : 데이터의 종류에 따라 steady 2D/3D array 또는 unsteady 3D/4D array로 처리한다.

GLOVE data structure에서 trim 및 sectional data를 저장하고 있다가 그래프를 생성하는데 필요한 데이터를 계산 및 추출해서 이를 data analysis core로 array 형태로 넘긴다. 이때, array의 크기와 raw 및 column의 index 그리고 X와 Y축의 min/max range도 함께 전달된다. 한 개의 array는 한 개의 plot을 생성하고 다수의 array가 전달될 경우에는 다수의 plot이 생성된다. 어떤 종류의 plot이 생성될지 (curve, distribution disk)도 함께 전달된다. Array(들)과 함께 plot 형태 및 color, legend, X 및 Y 축 label 등이 함께 전달된다. 만약 disk distribution 형태의 그래프라면 color 대신 transfer function이 전달된다.

GLOVE의 렌더링 코어가 VTK를 기반으로 개발되므로 vtkImageData

형태로 출력되면 좋겠지만 Gnuplot에 이런 기능이 존재하지 않으므로 일반 pixmap으로 생성한 후 이를 GLOVE에서 알아서 vtkImageData로 변환한 후 렌더링한다.

5. 구현

가. API 구성

• **mode** : 어떤 그래프를 그릴지 정한다.

문법)

```
void mode (int mode);
```

[소스 5-1] mode 함수의 문법

사용례)

```
glvGraphRotor *graph;

   ...
   graph->mode(ROTOR_GRAPH_BASE_AZIMUTH | ROTOR_GRAPH_TYPE_SECT_N
ORM_FORCE);
```

[소스 5-2] mode 함수의 사용 예제

파라미터)

① mode : 아래에 나열한 상수에서 ROTOR_GRAPH_BASE_*와 ROTOR_GRAPH_TYPE_*를 하나씩 선택해서 bitwise OR을 수행한 결과

- ROTOR_GRAPH_BASE_UNKNOWN
- ROTOR_GRAPH_BASE_AZIMUTH
- ROTOR_GRAPH_BASE_SPAN
- ROTOR_GRAPH_BASE_DISTRIBUTION
- ROTOR_GRAPH_BASE_REVOLUTION
- ROTOR_GRAPH_BASE_CHORD

- ROTOR_GRAPH_TYPE_MASK

- ROTOR_GRAPH_SECT_NORM_FORCE
- ROTOR_GRAPH_SECT_SPAN_FORCE
- ROTOR_GRAPH_SECT_CHORD_FORCE
- ROTOR_GRAPH_X_MOMENT
- ROTOR_GRAPH_Y_MOMENT
- ROTOR_GRAPH_Z_MOMENT
- ROTOR_GRAPH_TANGENT_CT
- ROTOR_GRAPH_PITCHING_MOMENT
- ROTOR_GRAPH_ROLLING_MOMENT
- ROTOR_GRAPH_PITCH_ANGLE
- ROTOR_GRAPH_PITCH_DIST
- ROTOR_GRAPH_PRESSURE

• *enable / disable / isEnabled*

문법)

```
void enable (int dataID);
void disable (int dataID);
void isEnabled (int dataID);
```

[소스 5-3] enable, disable, isEnabed 함수의 문법

사용례 1)

```
glvGraphRotor *graph;
boot ret;

  ...
  graph->enable(GLV_DATA_0);
  graph->disable(GLV_DATA_3);
  ret = graph->isEnabled(GLV_DATA_0);
```

[소스 5-4] enable, disable, isEnabled 함수의 사용 예

파라미터)

① dataID : GLV_DATA_0 ~ GLV_DATA_7 중 하나

• *set* : 그래프의 다양한 속성을 변경한다. 여기에는 제목, x축 레이블, y축 레이블, 그래프의 너비와 높이 등이 포함된다. 이 함수는 다양한 파라미터를 가질 수 있으며, 어플리케이션에서 호출할 때 전달받는 파라미터의 데이터 타입에 따라서 그래프의 서로 다른 속성을 제어한다.

문법)

```
void set (GLV_GRAPH_MODE mode, int value);
void set (GLV_GRAPH_MODE mode, char *value);
void set (GLV_GRAPH_MODE mode, double value);
void set (GLV_GRAPH_MODE mode, double *value);
void set (GLV_GRAPH_MODE mode, int dataID, char *value);
```

[소스 5-5] set 함수의 문법

사용례 1)

```
glvGraphRotor *graph;

  ...
  graph->set(GLV_GRAPH_WIDTH, 320);
```

[소스 5-6] set 함수의 사용 예

파라미터)

① mode : 아래 상수 중 하나

- GLV_GRAPH_WIDTH
- GLV_GRAPH_HEIGHT
- GLV_GRAPH_TYPE

② value : mode가 GLV_GRAPH_WIDTH이거나 GLV_GRAPH_HEIGHT인 경우에는 양의 정수 값을 가지며, mode가 GLV_GRAPH_TYPE일 때에는 GLV_GRAPH_TYPE_DISC나 GLV_GRAPH_TYPE_LINE을 갖는다.

사용례 2)

```
glvGraphRotor *graph;

  ...
  graph->set(GLV_GRAPH_TITLE, "Test graph");
```

[소스 5-7] set 함수의 사용 7

파라미터)

① mode : 아래 상수 중 하나

- GLV_GRAPH_TITLE
- GLV_GRAPH_X_LABEL
- GLV_GRAPH_Y_LABEL
- GLV_GRAPH_Z_LABEL

② value : 문자열

사용례 3)

```
glvGraphRotor *graph;
double value[2] = { 2.0, 3.0 };

  ...
  graph->set(GLV_GRAPH_X_RANGE, value);
```

[소스 5-8] set 함수의 사용 예

파라미터)

① mode : 아래 상수 중 하나

- GLV_GRAPH_X_RANGE
- GLV_GRAPH_Y_RANGE
- GLV_GRAPH_XTICS
- GLV_GRAPH_YTICS

② value : 두 개의 항(element)을 갖는 실수(double) 배열

사용례 4)

```
glvGraphRotor *graph;

  ...
  graph->set(GLV_GRAPH_LEGEND, GLV_DATA_0, "Line graph");
```

[소스 5-9] set 함수의 사용 예

파라미터)

① mode : GLV_GRAPH_LEGEND

② dataID : GLV_DATA_0 ~ GLV_DATA_7 중 하나

③ value : 문자열

• ***get*** : class에 저장된 그래프 이미지를 vtkImageData 형태로 전달 받는다.

문법)

```
vtkImageData *get (void);
```

[소스 5-10] get 함수의 문법

사용례)

```
glvGraphRotor *graph;
vtkImageData *image;

   graph = new glvGraphRotor();

   ...

   image = graph->get();
```

[소스 5-11] get 함수의 사용 예

• *load* : class에 데이터를 저장한다. Class 내부적으로는 저장한 데이터를 STL string 형태로 바꿔서 Gnuplot에 대한 명령어로 보관한다.

문법)

```
void load (int dataID, int nCount, double *xData, double *yData);
void load (int dataID, int nCount, double *xData, double *yData,
double *zData);
```

[소스 5-12] load 함수의 문법

사용례)

```
glvGraphRotor *graph;
double *xData;   // 1D array
double *yData;   // 1D array

  ...
  graph->load(GLV_DATA_0, xData, yData);
```

[소스 5-13] set 함수의 사용 예

파라미터)

① dataID : GLV_DATA_0 ~ GLV_DATA7 중 하나의 값을 가질 수 있으며, 최대 8개의 그래프를 동시에 그릴 수 있게 된다.

② nCount : 데이터의 갯수

③ xData : X 축 데이터

④ yData : Y 축 데이터

⑤ zData : Z 축 데이터

• ***draw*** : class에 저장돼있는 데이터를 Gnuplot에 제공하고, 그래프를 그려서 이미지 형태로 보관한다.

문법)

```
void draw (void);
```

[소스 5-14] draw 함수의 문법

사용례)

```
glvGraphRotor *graph;

  ...
  graph->draw();
```

[소스 5-15] draw 함수의 사용 예

• ***reset*** : class에 저장돼있는 모든 데이터와 그림을 지우고 초기 상태로 돌아간다.

문법)

```
void reset (void);
```

[소스 5-16] reset 함수의 문법

사용례)

```
glvGraphRotor *graph;

  ...
  graph->reset();
```

[소스 5-17] reset 함수의 사용 예

6. 기술적인 문제점 및 해결방법

• *VTK에서 image를 출력할 때 위아래가 뒤집히는 문제*

⇒ vtkImageReader2에서 FileLowerLeftOn이라는 member function을 사용할 수 있다. 하지만 이 클래스에서는 virtual function으로만 구현됐고, 실제로는 이 클래스를 상속받는 클래스에 구현돼있을 것이다.

• VTK에서 image 형태의 데이터를 가장 잘 다룰 수 있는 클래스

⇒ 바로 위에 적은 문제와도 관련이 있는데, vtkImageData 클래스는 기본적으로 vtkDataObject 계열에 속하고, vtkPNMReader와 같이 실제로 file을 읽을 때 일반적으로 사용하는 클래스는 vtkAlgorithm 계열에 속한다는 차이가 있다. 이것이 문제가 되는 이유는 바로 FileLowerLeftOn를 사용할 수 있느냐 없느냐의 차이가 발생하기 때문이다. 비전문가의 또 다른 관점에서 생각해본다면 VTK 내에서도 vtkAlgorithm 계열의 클래스는 왠지 vtkDataObject 계열의 클래스를 manipulate하는 subject의 역할을 주로 수행하고, vtkDataObject는 말하자면 internal data structure 정도가 맞을 것 같다는 생각이기 때문에 vtkImageData를 직접 사용하는 것이 바람직한지의 여부에 대한 판단이 잘 서지 않는다는 문제도 있다.

• *STL의 string 클래스의 길이제한 문제*

• *vtkXYPlotActor 문제*

⇒ vtkXYPlotActor는 vtkActor2D의 파생 클래스로 GLOVE에서 사용하는 vjRenderer와 호환이 가능한지는 미지수. vjRenderer는 vtkActor를 받을 수 있는데, vtkActor와 vtkActor2D는 꽤 먼 친척

• *vtkActor2D 문제*

⇒ vtkActor2D는 rendering window를 2D 캠퍼스라고 간주하고 지정

한 위치에 그림을 그린다. 즉, GIVI에서 요구하는 것과 같은 임의의 plane에 그리도록 하고 사용자가 마음대로 3차원 공간에서 배치할 수 있게 하는 것은 안 된다. 지금 상태에서는 vjRenderer가 vtkActor2D를 지원한다고 해도 (특별한 처리를 하지 않은 한 – 그리고 특별한 처리를 했을 것 같지는 않다) 메뉴와 같이 화면의 어느 지점에 고정으로 배치할 수밖에 없다.

• *IPC pipe을 이용해서 GNUPlot의 결과를 읽어오는 방법*

```
int
Run (char *com[])
{
int p1[2], p2[2], width, height, format, size, result;
static char *command = {"set size 2.0, 2.0\nset title \"Test Plot
\"\nset xlabel \"Deflection (meters)\"\nset ylabel \"Force (kN)\"
\nset terminal pbm color\nplot \"-\" with linespoints\n0.000 0 0
\n0.001 104 51\n0.002 202 101\n0.003 298 148\n0.0031 290 149\n0.0
04 289 201\n0.0041 291 209\n0.005 310 250\n0.010 311 260\n0.020 2
80 240\ne\n"};
char *pixmap;

   /* create pipe from mother to child */
   if (pipe(p1) == -1)
     {
       fprintf(stderr, "Unable to create pipe from mother to chil
d\n");
     }

   /* create pipe from child to mother */
   if (pipe(p2) == -1)
     {
       fprintf(stderr, "Unable to create pipe from child to mothe
r\n");
     }

   switch(fork())
     {
       case 0 : /* child */
          /* connect pipe descriptor into stdin */
          dup2(p1[0], 0);
          /* connect pipe descriptor into stdout */
          dup2(p2[1], 1);

          close(p1[0]);
          close(p2[1]);

          execvp(com[0], com);
```

```
         break;

       default : /* mother */

         dup2(p2[0], 0);

         close(p2[0]);

         write(p1[1], command, strlen(command));

         close(p1[1]);

         ReadPBMHeader(&width, &height, &format);

         if (format == PBMP4)
           {
             size = width * height;
           }
         else if (format == PBMP6)
           {
             size = width * height * 3;
           }

         pixmap = (char *) calloc(size, sizeof(char));
         if (pixmap == NULL)
           {
             fprintf(stderr, "Cannot allocate memory for pixmap
\n");
             exit(0);
           }

        result = fread(pixmap, sizeof(char), size, stdin);
        fprintf(stderr, "result : %d\n", result);

        convertPixmaptoVTKImage(pixmap, width, height, format);
    }

  return 1;
}


int
main()
{
char *gnuplot[2] = {"gnuplot", NULL};
int ret;

  ret = Run(gnuplot);

  return 1;
}
```

[소스 6-1] IPC pipe를 이용한 Gnuplot 실행