ISBN 978-89-6211-276-4

Virtual Reality 구현을 위한 ParaView 분석

(The Analysis of ParaView for Implementing Virtual Reality System)

금 복 희 (Bokhee Keum)
bhkeum@kisti.re.kr
Visualization Team, Supercomputing Center

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

제목 차례

표 차례

그림 차례

1. 서론

Computer simulation은 computation 기술이 발달 할수록 많이 사용되는 연구방법으로, simulation 결과를 해석하기 위해서 scientific visualization 기법이 이용되고 있다. Scientific visualization은 텍스트 혹은 바이너리 형태의 데이터를 사용자가 이해하기 쉬운 그래픽 형태로 변환하여 보여주는 기법을 연구하는 분야이다. 사용자들이 많이 사용하고 있는 scientific visualization software 중에는 AVS(Advanced Visual Systems)[22]와 같은 상용 소프트웨어, COVISE, VisIT, OpenDX, ParaView과 같은 free software가 있다. 이런 visualization software는 적용 가능한 platform의 종류, parallel processing/remote visualization/collaboration 기능 지원여부, workflows 그리고 multi-view 기능의 지원, Virtual Reality(가상현실. 이하 VR)과 같은 특징에 의해 각 시스템의 장단점이 구별된다.

- **Workflows(or Data Flows)** : 복잡한 계산 과정을 표현하고 관리하는 패러다임(paradigm)으로 주어진 data set에 대해서 어떠한 visualization operation을 적용할지 계산의 내용을 정하고 계산이 적용되는 순서를 정할 수 있는 user interface 방식이다. Workflows는 visualization system에 따라 사용하는 용어와 제공하는 인터페이스 방식이 차이가 있지만 적용되는 개념은 거의 동일 하고 information analysis process를 가속화하고 변환하는데 장점이 있으므로 기존에 사용하던 shell scripts 방식에서 이 방법으로 빠르게 전환이 이루어지고 있다.
- **Remote Visualization** : Supercomputer와 같은 HPC(High Performance Computer)를 사용해야 하는 대부분의 computer simulation 결과는 데이터의 크기가 상당히 크기 때문에 빠른 visualization 결과를 얻기 위해서는 simulation을 한 computer site 즉, remote site에서 visualization을 위한 계산을 수행하고 local site에서는 그 결과를 보는 viewer의 실행만 요구된다. 이와 같이 client viewer와 server visualization engine이 서로 다른 site에서 실행되는 것을 remote visualization이라 한다.
- **Multi-View Visualization** : 하나의 data set에 대해서 동시에 여러 가지 visualization을 수행하여 각각의 결과가 서로 다른 view에 display 되거나, 서로 다른 data set에 대해서 수행한 visualization 결과가 각각의 view를 통해

하나의 display상에 나타나는 기능을 말한다.

COVISE[7]는 독일 HLRS에서 개발하고 VISENSO에서 판매하는 가시화 소프트웨어로 가상환경과 협업 네트워크 기능을 지원하는 모듈화 된 가시화 프로그램이다. 슈퍼컴퓨터를 기반으로 하는 시뮬레이션, 후처리 및 가시화 기능을 통합 환경에서 제공하며 데이터에 대한 가시화 처리를 모듈 별 그래프 연결을 지원하는 데이터 플로우 (data flow) 프로그래밍 모델을 지원하며, 다양한 형태의 데이터 파일을 지원한다. COVISE의 VR 프로그램인 COVER(COVISE Virtual Environment Renderer)[8]는 COVISE의 3D Renderer Module로 사용될 수 있을 뿐만 아니라 COVISE와는 독립적으로 사용될 수 있다.

VisIT[9]은 tera-scale data sets에 대한 visualization을 위하여 distributed visualization을 제공하고, OepnDX[10]는 IBM Visualization Data Explorer의 open source 버전으로 parallel visualization이 가능하다.

ParaView[11]는 parallel processing, large data set 처리를 위한 distributed computation 기능을 지원하는 open source visualization software이다. 데이터에 대한 visualization filter 적용 방법으로 workflow를 지원하며 하나의 data set에 대하여 여러 개의 visualization view를 생성할 수 있을 뿐만 아니라 서로 다른 data set에 대한 독립적인 visualization view를 생성할 수 있다.

VisTrails[12]은 앞에서 설명한 visualization software들에 비해서 multi-view visualization 지원이 강화되어 있는 software로[13] visualization pipeline의 생성과 관리를 쉽게 할 수 있는 기능과 함께 기존에 있는 visualization system이나 library와 연결하여 사용할 수 있는 infrastructure를 제공한다. VisTrails은 하나의 pipeline을 뜻하는 VisTrail을 key component로 하며 사용자가 VisTrail의 파라미터를 변경하더라도 이전의 결과를 history로 가지고 있어서 변화된 결과와 비교가 가능할 뿐만 아니라 이전의 pipeline과 중첩되는 부분의 결과는 caching이 되어 새로운 파라미터에서 활용함으로써 계산 속도를 높이는 optimization 기능을 가지고 있다.

SCIRun[14]은 University of Utah의 Scientific Computing and Imaging Institute에서 simulation, modelling, visualization을 위해 개발한 핵심적인 Problem Solving Environment(PSE)로 BioPSE와 함께 사용되며 application에 따라 서로 다른 customized interface를 지원하는 PowerApp를 제공한다.

Name	COVISE	VisIT	OpenDX	ParaView	VisTrails	SCIRun
Organization	HLRS	LLNL	IBM	Kitware & LANL	Utah	Utah
Since	1993	2000	1991	2000	2005	1992
VR Support	Yes	No	No	No	No	No
Multi-View visualization	No	Yes	Not Known	Yes	Yes	No
Based on VTK	No	Yes	No	Yes	Yes	Yes
Interfaces	C++	C++, Python, Java	C, Python, Tk/Tcl	C++	Python	C++
Dynamic Loading Plug-ins	Yes	Yes	No but Add-Ons 추가 가능	Yes	Not Known	Yes
Open Source	No	Yes	Yes	Yes	Yes	Yes
Parallel Processing	Yes	Yes	Yes	Yes	Not known	Not known
Remote Visuzliation	Yes	Yes	Yes	Yes	Not known	Not known
Workflows	Yes	No	Yes	Yes	Yes	Yes

[표 1] Visualization Software 비교

한편, 기하급수적으로 발전하는 computation 기술에 비하여 visualization 기술의 발전은 이를 따라가지 못하고 있으며 simulation scientist가 graphic workstation이나 PC를 이용한 기존의 방식으로 대용량의 복잡한 결과 데이터를 이해하기가 어려워지고 있다. 이에 사용자들은 2D display에서의 단순한 interface 보다 더 발전된 방식을 요구하고 있으며 이러한 사용자 요구에 대한 해결방법으로 VR 기술이 적용되고 있으며 VR 적용 시 장점이 많이 보고되고 있다[5,6].

따라서 위에서 소개한 visualization software 중에서 VR 기능을 제공하고 있지 않은 ParaView를 대상으로 VR 기능을 구현하기 위하여 필요한 기술적 분석 내용을 이 보고서에서 다루고자 한다.

2. ParaView 프로그램의 특징

가. Server Manager 구조

- Server Manager는 ParaView client의 visualization request를 처리하는 engine 부분을 말하며 distributed client-server application 구성이 용이하도록 하기 위한 구조이다.
- client는 VTK object를 직접 access하지 않고 "vtkSM***"이라는 이름을 가진 class들을 통하여 access된다.

나. Proxy 사용

- ParaView client의 GUI 구성 및 request 처리를 대신하는 Client-side Proxy(pqProxy, etc.)와 server의 기능 및 구성을 처리하는 Server-side Proxy (vtkSMProxy, etc.)를 두고 있음. 참고 : ParaView Proxy List
- Proxy는 helper class인 ProxyManager(vtkSMProxyManager)에 의해 생성되고 관리됨.

다. Client Object의 종류

ParaView가 제공하는 기본적인 object와 Plugin을 통하여 사용자가 추가할 수 있는 object의 종류는 다음과 같다.

- Source
  - Pipeline object(vtkAlgorithm)를 source라 하며 visualization의 대상이 되는 data는 모두 source가 됨.
  - Proxy group 중에서 "sources"에 속하며 SourceProxy로 관리됨(vtkSMSourceProxy).
  - ParaView에서 기본으로 제공하는 source와 Plugin으로 추가되는 source가 있을 수 있음.
  - 데이터 파일을 새로 open하여 loading할 경우 그 결과도 source가 됨
  - Source object는 pqPipelineSource class의 instance

- Filter
  - visualization 처리 module이 되는 object를 말함.
  - ParaView에서 기본으로 제공하는 filter와 Plugin으로 추가되는 filter가 있을 수 있음.
  - Filter object는 pqPipelineSource class의 instance
- Reader
  - Data file를 open하는 module를 말함.
  - ParaView에서 기본으로 제공하는 reader와 Plugin으로 추가되는 reader가 있을 수 있음.
  - Reader object는 pqPipelineSource class의 instance
  - ParaView에서 지원하는 file format
  - 참고 : paraview/Qt/Core/pqReaderFactory
- View
  - Pipeline 처리결과(visualization 처리결과)가 그려지는 window를 말하며 하나의 window에 여러 개의 representation이 함께 display될 수 있음.
  - ParaView에서 기본으로 제공하는 view와 Plugin으로 추가되는 view가 있을 수 있음.
  - View object는 pqView class의 instance
- Representation
  - Pipeline에 있는 object들을 통틀어 representation이라 함. 하나의 pipeline은 하나의 representation이 됨.
  - 따라서 representation은 data object를 가져가서 rendering이 되게 변환을 하고 view에 결과를 그려주는 기능을 한다. 예를 들어, view가 VTK view(RenderView)일 때, geometry filters, level-of-detail algorithm, vtkMappers, vtkActors를 합쳐서 하나의 representation이라 한다.
  - ParaView에서 기본으로 제공하는 representation와 Plugin으로 추가되는 view가 있을 수 있음.
  - Representation object는 pqDataRepresentation class의 instance
- Writers

- ParaView에서 기본으로 제공하는 Writer와 Plugin으로 추가되는 writer가 있을 수 있음.

3. ParaView 프로젝트 구조 분석

가. 디렉토리 구성

ParaView Project는 다음과 같은 디렉토리로 구성이 되어 있으며 각 디렉토리에서 다루고 있는 내용은 다음과 같다.

- **Applications**

ParaView의 실제 구동 프로그램이 있는 곳으로 paraview라는 Client 프로그램이 존재하는 곳임. 세가지 형태의 Client 프로그램(Client, CustomServer, DobranoViz)을 생성할 수 있도록 되어 있음.

- Client : paraview client 구동 프로그램 관리
- CustomServer
- DobranoViz

- **CMake**

ParaView의 컴파일에 필요한 cmake configuration file 제공

- **Common**

Kitware에서 작성한 ParaView용 VTK derived class 제공

- **Documentation**

help html 파일 제공

- **Examples**

Custom menu, filter, reader 등을 Plugin으로 추가하는 examples 제공

- Plugins/GUIObjectPanel: Object Inspector panel에 Plugin으로 새로운 기능(widget)을 추가하는 방법을 보여줌
- Plugins/GUIToolBar: 사용자가 Plugin 형태로 Menu bar 혹은 Tool bar에 새로운 메뉴를 추가하는 방법을 보여줌
- Plugins/GUIView: 새로운 형태의 view와 representation을 추가하는 방법을

보여줌

- Plugins/ParametricSource: VTK에서 제공하고 있는 source(vtkParametricBoy, vtkParametricConicSpiral, vtkParametricEllipsoid)를 ParaView의 "sources" proxy group에 추가하는 방법을 보여줌
- Plugins/Reader: VTK에서 제공하고 있는 data reader(vtkPNGReader)를 "sources" proxy group에 추가하는 방법을 보여줌
- Plugins/SMFilter: 새로운 filter를 "filters" proxy group에 추가하는 방법을 보여줌

- **Plugins**

Plugin 예제 프로그램 제공

- **Qt**

Qt Designer로 작성한 GUI 파일과 관련 header, source 파일 관리

- Chart : XYPlot View와 BarChart View Handling하기 위한 모듈 제공
- Components : Qt GUI resource 파일과 관련 header file, source file 제공 (주로 Interface 구성 관련 파일)
- Core : User Interface 처리와 client-side event들을 server-side event handling과 연결시키는 모듈 제공
- Python : ParaView client 프로그램에서 제공하는 Python Shell(tools menu에 있음) 생성을 위한 모듈 제공
- Testing
- Widgets : User Interface 구성에 필요한 추가적인 사용자 GUI 화면을 생성하는 모듈 제공

- **Servers**

paraview server side 프로그램 제공

- Common : !Paraview에 필요한 VTK derived class 제공
- Executables : pvserver, pvdataserver, pvrenderserver 프로그램 관리
- Filters : Paraview server의 filter 모듈관련 프로그램 관리
- ServerManager : Client의 request를 server에서 처리할 수 있도록 중간 역할을 하는 server manager 관련 모듈 제공

- Testing
- Wrapping : Python script를 실행할 수 있도록 제공되는 Python wrapper

- **Utilities**

ParaView에서 사용하는 utility 프로그램(Doxygen, hdf5, IceT, Xdmf2) 관리

- **VTK**

VTK library 제공

나. ParaView 프로세스

1) paraview

ParaView client 프로세스로 사용자에게 GUI 인터페이스를 제공하고 server로의 connection을 설정하지 않는 한 server의 모든 기능을 이 프로세스가 담당함(stand-alone 시스템으로 동작).

2) pvserver

- ParaView의 server 프로세스로 data processing과 render processing을 하나의 job으로 수행.
- pvdataserver와 pvrenderserver의 기능을 모두 가지고 있으면서 pvdataserver와 pvrenderserver간의 데이터 전달이 없으므로 data server와 render server를 따로 실행하는 것보다 성능이 우수함.
- 병렬실행 모드 지원(>mpi -np 4 pvserver).

3) pvdataserver

- ParaView의 server 프로세스로 data processing만 수행.
- pvrenderserver도 함께 실행을 시켜야 함.
- 고성능 CPU cluster와 GPU cluster를 함께 활용하는 경우 적당.
- 병렬실행 모드 지원.
- 데이터 전달에 따른 오버헤드 발생.

4) pvrenderserver

- ParaView의 server 프로세스로 render processiong만 수행.
- pvdataserver도 함께 실행을 시켜야 함.
- 고성능 CPU cluster와 GPU cluster를 함께 활용하는 경우 적당.
- 병렬실행 모드 지원.
- 데이터 전달에 따른 오버헤드 발생.

5) pvbatch

Distributed server에 batch script를 실행시키기 위한 ParaView application. ParaView가 MPI mode로 컴파일이 되면 pvbatch는 MPI 프로그램으로 실행됨. pvbatch는 interactive mode로 실행되지 않음.

6) pvpython

ParaView의 Python client 프로그램으로 paraview와 마찬가지로 server에 connection을 형성하여 batch mode 혹은 interactive mode로 실행됨.

4. ParaView Visualization Filter

참조: paraview/Servers/ServerManager/Resources/filters.xml

가. Filter List

	Function	VTK or ParaView Class
1	Probe Location over Time	vtkPExtractArraysOverTime
2	Plot Field Variable over Time	vtkExtractTemporalFieldData
3	Image Shrink	vtkImageShrink3D
4	Surface Vectors	vtkSurfaceVectors
5	Integrate Variables	vtkIntegrateAttributes
6	IntegrateFlowThroughSurface	vtkIntegrateFlowThroughSurface
7	All to N	vtkAllToNRedistributePolyData
8	AppendAttributes	vtkMergeArrays
9	Balance	vtkBalancedRedistributePolyData
10	Append Geometry	vtkAppendPolyData

11	Append Datasets	vtkAppendFilter
12	Cell Centers	vtkCellCenters
13	Cell Data to Point Data	vtkPCellDataToPointData
14	CleanPolyData	vtkCleanPolyData
15	CleanUnstructuredGrid	vtkCleanUnstructuredGrid
16	Delaunay 2D	vtkDelaunay2D
17	Connectivity	vtkPVConnectivityFilter
18	ImageClip	vtkImageClip
19	Curvature	vtkCurvatures
20	Decimate	vtkDecimatePro
21	D3	vtkDistributedDataFilter
22	Elevation	vtkElevationFilter
23	Extract CTH Parts	vtkExtractCTHPart
24	ExtractEdges	vtkExtractEdges
25	Extract Surface	vtkDataSetSurfaceFilter
26	Calculator	vtkArrayCalculator
27	FeatureEdges	vtkFeatureEdges
28	Gradient	vtkImageGradient
29	Gradient (Unstructured)	vtkGradientFilter
30	Gradient Magnitude	vtkImageGradientMagnitude
31	Linear Extrusion	vtkPVLinearExtrusionFilter
32	Loop Subdivision	vtkLoopSubdivisionFilter
33	Mask Points	vtkMaskPoints
34	Median	vtkImageMedian3D
35	Mesh Quality	vtkMeshQuality
36	Normals Generation	vtkPPolyDataNormals
37	Outline	vtkPOutlineFilter
38	Outline Corners	vtkPOutlineCornerFilter
39	Octree Depth Scalars	vtkHyperOctreeDepth
40	Process Id Scalars	vtkProcessIdScalars
41	Point Data to Cell Data	vtkPointDataToCellData
42	Quadric Clustering	vtkQuadricClustering
43	Random Vectors	vtkBrownianPoints
44	Reflect	vtkReflectionFilter
45	Ribbon	vtkRibbonFilter
46	Rotational Extrusion	vtkRotationalExtrusionFilter
47	Shrink	vtkShrinkFilter
48	Smooth	vtkSmoothPolyDataFilter
49	Triangle Strips	vtkStripper

50	Subdivide	vtkLinearSubdivisionFilter
51	Tessellate	vtkTessellatorFilter
52	Tetrahedralize	vtkDataSetTriangleFilter
53	Transform	vtkTransformFilter
54	Triangulate	vtkTriangleFilter
55	Tube	vtkTubeFilter
56	Warp (scalar)	vtkWarpScalar
57	Warp (vector)	vtkWarpVector
58	Slice	vtkCutter
59	Extract Cells by Region	vtkExtractGeometry
60	Clip	vtkPVClipDataSet
61	Threshold	vtkThreshold
62	Contour	vtkContourFilter
63	Glyph	vtkPVGlyphFilter
64	Glyph (Custom Source)	vtkPVGlyphFilter
65	Extract Subset	vtkPVExtractVOI
66	Probe Location	vtkPProbeFilter
67	Plot Over Line	vtkPProbeFilter
68	Resample with dataset	vtkPProbeFilter
69	Stream Tracer (Custom Source)	vtkDistributedStreamTracer
70	Temporal Cache	vtkTemporalDataSetCache
71	Temporal Interpolator	vtkTemporalInterpolator
72	TemporalSnapToTimeStep	vtkTemporalSnapToTimeStep
73	Temporal Shift Scale	vtkTemporalShiftScale
74	ParticleTracer	vtkTemporalStreamTracer
75	Outline (curvilinear)	vtkStructuredGridOutlineFilter
76	Clip (Generic Datasets)	vtkGenericClip
77	Contour (Generic Datasets)	vtkGenericContourFilter
78	Slice (Generic Datasets)	vtkGenericCutter
79	Extract Surface (Generic Datasets)	vtkGenericGeometryFilter
80	Outline (Generic DataSets)	vtkGenericOutlineFilter
81	Stream Tracer (Generic Datasets	vtkGenericStreamTracer
82	Tessellator (Generic Datasets)	vtkGenericDataSetTessellator
83	Group Datasets	vtkMultiGroupDataGroupFilter
84	Level Scalars	vtkMultiGroupDataGroupIdScalars
85	GeometryFilter	vtkPVGeometryFilter
86	OrderedCompositeDistributor	vtkOrderedCompositeDistributor
87	MPIMoveData	vtkMPIMoveData
88	ClientServerMoveData	vtkClientServerMoveData

89	ReductionFilter	vtkReductionFilter
90	ExtractLevel	vtkMultiGroupDataExtractGroup
91	Extract Datasets	vtkMultiGroupDataExtractDataSets
92	Histogram	vtkPExtractHistogram
93	RectilinearGridGeometryFilter	vtkRectilinearGridGeometryFilter
94	Texture Map to Plane	vtkTextureMapToPlane
95	Texture Map to Sphere	vtkTextureMapToSphere
96	Texture Map to Cylinder	vtkTextureMapToCylinder
97	Polyline to Rectilinear Grid	vtkPolyLineToRectilinearGridFilter
98	MinMax	vtkMinMax
99	Extract Selection	vtkPVExtractSelection
100	Annotate Time	vtkTimeToTextConvertor
101	Select Through	vtkExtractSelection
102	Extract Thresholds	vtkExtractSelection
103	Extract Selections	vtkExtractSelection
104	ConvertSelection	vtkPConvertSelection

[표 2] ParaView Filter List

나. Default Filter 생성 코드

```
pqPipelineSource* pqObjectBuilder::createFilter(
  const QString& group, const QString& name, pqPipelineSource* input)
{
  QMap<QString, QList<pqOutputPort*> > namedInputs;
  QList<pqOutputPort*> inputs;
  inputs.push_back(input->getOutputPort(0));
  namedInputs["Input"] = inputs;

  return this->createFilter(group, name, namedInputs, input->getServer());
}



//-----------------------------------------------------------
pqPipelineSource* pqObjectBuilder::createFilter(
  const QString& group, const QString& name,
  QMap<QString, QList<pqOutputPort*> > namedInputs,
  pqServer* server)
{
```

```
  vtkSMProxy* proxy =
    this->createProxyInternal(group, name, server, "sources");
  if (!proxy)
    {
    return 0;
    }

  pqPipelineSource* filter = pqApplicationCore::instance()->getServerManagerModel()
->
    findItem<pqPipelineSource*>(proxy);
  if (!filter)
    {
    qDebug() << "Failed to locate pqPipelineSource for the created proxy "
      << group << ", " << name;
    return 0;
    }

  // Now for every input port, connect the inputs.
  QMap<QString, QList<pqOutputPort*> >::iterator mapIter;
  for (mapIter = namedInputs.begin(); mapIter != namedInputs.end(); ++mapIter)
    {
    QString input_port_name = mapIter.key();
    QList<pqOutputPort*> &inputs = mapIter.value();

    vtkSMProperty* prop = proxy->GetProperty(input_port_name.toAscii().data());
    if (!prop)
      {
      qCritical() << "Failed to locate input property "<< input_port_name;
      continue;
      }

    foreach (pqOutputPort* opPort, inputs)
      {
      pqSMAdaptor::addInputProperty(prop, opPort->getSource()->getProxy(),
        opPort->getPortNumber());
      }

    proxy->UpdateVTKObjects();
    prop->UpdateDependentDomains();
    }
```

```
  // Set default property values.
  filter->setDefaultPropertyValues();
  filter->setModifiedState(pqProxy::UNINITIALIZED);

  emit this->filterCreated(filter);
  emit this->proxyCreated(filter);
  return filter;
}
```

다. Custom Filter 생성 코드

```
pqPipelineSource* pqObjectBuilder::createCustomFilter(const QString& sm_name,
    pqServer* server, pqPipelineSource* input/*=0*/)
{
  vtkSMProxy* proxy =
    this->createProxyInternal(QString(), sm_name, server, "sources");
  if (!proxy)
    {
    return 0;
    }

  pqPipelineSource* filter = pqApplicationCore::instance()->
    getServerManagerModel()->findItem<pqPipelineSource*>(proxy);
  if (!filter)
    {
    qDebug() << "Failed to locate pqPipelineSource for the created custom filter prox
y "
      << sm_name;
    return 0;
    }

  vtkSMProperty* inputProperty = proxy->GetProperty("Input");
  if (inputProperty && input)
    {
    pqSMAdaptor::setProxyProperty(inputProperty, input->getProxy());
    proxy->UpdateVTKObjects();
    inputProperty->UpdateDependentDomains();
    }
```

```
  // Set default property values.
  filter->setDefaultPropertyValues();
  filter->setModifiedState(pqProxy::UNINITIALIZED);
  emit this->customFilterCreated(filter);
  emit this->proxyCreated(filter);
  return filter;
}
```

5. Writer의 종류

	Function	VTK or ParaView Class
1	XMLPVDWriter	vtkXMLPVDWriter
2	XMLPolyDataWriter	vtkXMLPolyDataWriter
3	XMLUnstructuredGridWriter	vtkXMLUnstructuredGridWriter
4	XMLStructuredGridWriter	vtkXMLStructuredGridWriter
5	XMLRectilinearGridWriter	vtkXMLRectilinearGridWriter
6	XMLImageDataWriter	vtkXMLImageDataWriter
7	XMLMultiGroupDataWriter	vtkXMLMultiGroupDataWriter
8	XMLPPolyDataWriter	vtkXMLPPolyDataWriter
9	XMLPUnstructuredGridWriter	vtkXMLPUnstructuredGridWriter
10	XMLPStructuredGridWriter	vtkXMLPStructuredGridWriter
11	XMLPRectilinearGridWriter	vtkXMLPRectilinearGridWriter
12	XMLPImageDataWriter	vtkXMLPImageDataWriter
13	XMLPMultiGroupDataWriter	vtkXMLPMultiGroupDataWriter
14	DataSetWriter	vtkGenericDataObjectWriter
15	PLYWriter	vtkPLYWriter
16	MetaImageWriter	vtkMetaImageWriter
17	PNGWriter	vtkPNGWriter
18	XdmfWriter	vtkXdmfWriter
19	ExodusIIWriter	vtkExodusIIWriter
20	EnSightWriter	vtkEnSightWriter
21	AnimationSceneImageWriter	vtkSMAnimationSceneImageWriter
22	XMLPVAnimationWriter	vtkXMLPVAnimationWriter
23	SummaryHelper	vtkPVSummaryHelper
24	CSVWriter	vtkCSVWriter

[표 3] Writer의 종류

6. Plugin 추가 하기

다음과 같은 순서로 사용자는 원하는 plugin을 추가할 수 있다.

가. Plugins 디렉토리 추가

Source code 디렉토리에서 CMakeLists.txt 파일에 Plugin으로 생성할 코드가 있는 디렉토리(예. "./Plugins")를 다음과 같이 추가한다.

```
IF(BUILD_SHARED_LIBS)
    ADD_SUBDIRECTORY(Plugins)
ENDIF(BUILD_SHARED_LIBS)
```

나. CMakeLists.txt 수정

- Plugin 디렉토리의 CMakeLists.txt의 Optional Plugin 생성 규칙을 사용하고 default값은 ON으로 설정한다.
- /Plugin/CMakeLists.txt 에서 ...

```
INCLUDE(${ParaView_SOURCE_DIR}/CMake/ParaViewPlugins.cmake)

  MACRO(paraview_build_optional_plugin name comment subdirectory default)
   OPTION(PARAVIEW_BUILD_PLUGIN_${name} "Build ${comment}" ${default})
    MARK_AS_ADVANCED(PARAVIEW_BUILD_PLUGIN_${name})
    IF(PARAVIEW_BUILD_PLUGIN_${name})
      MESSAGE(STATUS "Plugin: ${comment} enabled")
      ADD_SUBDIRECTORY("${subdirectory}")
    ELSE(PARAVIEW_BUILD_PLUGIN_${name})
      MESSAGE(STATUS "Plugin: ${comment} disabled")
    ENDIF(PARAVIEW_BUILD_PLUGIN_${name})
  ENDMACRO(paraview_build_optional_plugin)

  paraview_build_optional_plugin(
```

```
    Prism "Prism" PrismPlugins ON)
```

다. Run cmake

- PARAVIEW_BUILD_SHARED_LIBS (default : OFF)를 ON으로 설정.
- ParaView build 시 이렇게 생성된 Plugin 라이브러리(*.so)는 binary directory의 bin 밑에 생성되고 ParaView 설치시 lib 디렉토리에 다른 라이브러리와 함께 설치되지 않음. 수동으로 깔아주거나 따로 Plugin 라이브러리를 두어야 함.

라. Plugin을 ParaView에 올리는 방법

- 방법1 : 환경변수 PV_PLUGIN_PATH를 사용하여 자동으로 로딩하기. 예를 들어 Plugin 라이브러리가 {user-home}/pvcmake/bin에 있을 경우, 다음과 같이 함

```
>export PV_PLUGIN_PATH=~/pvcmake/bin
>paraview
```

- 방법2 : 먼저, paraview를 실행하고 "Tools" 메뉴에서 "Manage Plugins..." 선택한다음 client, server 각 각의 라이브러리 파일을 선택하여 로딩.

7. pvserver Configuration 방법

가. Local Host connection 설정시

- paraview 실행 (>paraview)
- server configuration file (*.pvsc) 작성: configuration file 위치 : user_home/.config/ParaView (default file : servers.pvsc)

```
<Servers>
<Server name="localhost (forward)" resource="cs://localhost" owner="site">
  <CommandStartup>
   <Command exec="pvserver" timeout="120" delay="3">
    <Arguments>
     <Argument value="--server-port=$PV_SERVER_PORT$" />
    </Arguments>
   </Command>
  </CommandStartup>
 </Server>
</Servers>
```

- ParaView의 메뉴바에서 File -> Connect -> Add Server (*.pvsc 파일 선택) ->서버 선택 -> Connect 선택
- pvserver connection 시, 다음과 같은 형태로 실행됨.

```
>pvserver --server-port=$PV_SERVER_PORT$
```

- 성공적으로 connection이 이루어진 경우 Pipeline Brower의 root node 명이 “localhost"로 바뀜
- pvserver가 connection을 실행하기 전에 이미 실행되어 있는 경우에는 이 프로세스에 연결을 시도함.

나. Remote host의 pvserver connection 설정하기

stero라는 remote host의 IP address가 150.183.235.224일 때 로그인은 guest로 하는 경우, 다음과 같이 *.pvsc 파일을 구성한다.

```
<Servers>
<Server name="stero" resource="cs://stereo" owner="site">
  <CommandStartup>
   <Command exec="ssh" timeout="120" delay="3">
    <Arguments>
     <Argument value="-l"/>
     <Argument value="guest"/>
     <Argument value="150.183.235.224"/>
     <Argument value="pvserver"/>
    </Arguments>
   </Command>
  </CommandStartup>
 </Server>
</Servers>
```

다. Single Server에서 여러 개의 ParaView server 사용하기

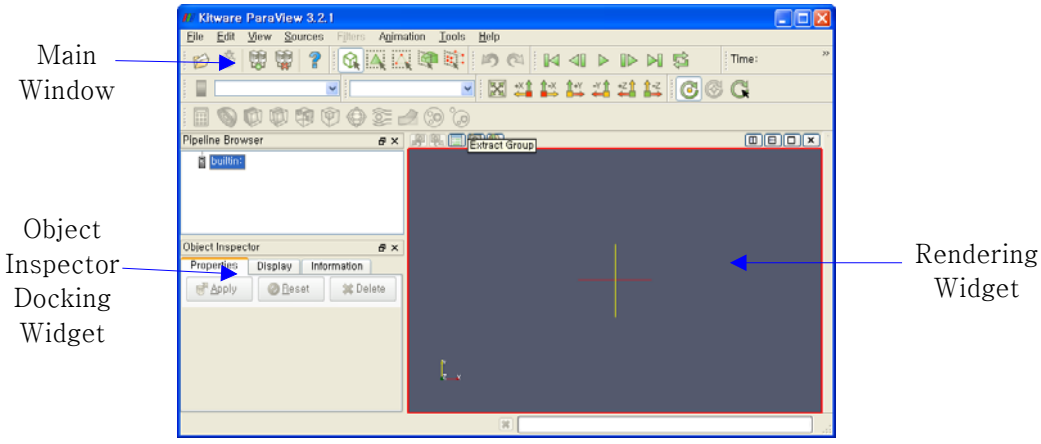
client host에서 다음과 같은 server configuration file을 작성하면 연결 호스트에서 사용이 가능한 포트번호를 탐색하여 그 번호로 연결이 가능하다.

selective_port.pvsc

```
<Servers>
<Server name="localhost (forward with port selection)" resource="cs://localhost" owner="site">
  <CommandStartup>
    <Options>
     <Option name="PV_SERVER_PORT" label="Port">
       <Range type="int" min="1" max="65535" step="1" default="11111"/>
     </Option>
    </Options>
   <Command exec="pvserver" timeout="120" delay="3">
```

```
    <Arguments>
     <Argument value="--server-port=$PV_SERVER_PORT$" />
    </Arguments>
   </Command>
  </CommandStartup>
 </Server>
</Servers>
```

8. User Interface



[그림 2] ParaView 주화면

ParaView의 User interface window 구현에 사용된 프로그램 내용은 다음과 같다.

가. Main Window

ParaView의 주 화면은 Qt 프로그램을 이용하여 만들어 졌으며 이 프로그램에서 생성된 resource 내용은 MainWindow.ui와 MainWindow.h에 있다. Qt에서 생성한 이들 resource는 주 화면을 그려줄 때 로딩되어 화면에 나타나게 되는데 이와 같이 주 화면을 생성하는 모듈은 ProcessModuleGUIHelper.cxx이다.

1) Qt UI Resource

- Resource File : paraview/Applications/Client/MainWindow.ui
- ui header file : MainWindow.h
- Qt로 디자인된 인터페이스는 .ui 확장자를 가진 파일 이름으로 저장된다. ui 파일을 이용해서 응용프로그램에서 사용할 수 있는 header 파일을 만드는 방법은 Qt 개발툴에서 제공되는 'uic' 명령어를 사용하는 것이다. 예를 들면, Qt로 디자인한 인터페이스의 파일이름이 'gui.ui'라 하고 생성하고자 하는 헤더 파일의 이름으로 gui.h를 사용하고자 할 때, '>uic -o gui.h gui.ui'와 같이 실행한다. (주: ParaView Client의 MainWindow.h는 MainWindow.ui로부터 uic를 사용하여 자동으로 생성한 파일은 아님)

2) MainWindow Creator

paraview/Applications/Client/ProcessModuleGUIHelper.cxx (-> 원도 생성과정에 대한 자세한 내용은 CreateMainWindow() 함수 참조)

나. Rendering Widget

- multiViewManager() -> MultiViewManager (pqViewManager type) : pqMainWindowCore.cxx/h -> pqView.cxx/h
- pqViewManager <- pqMultiView

다. Object Inspector Docking Widget

- Tab Creation

- pqProxyTabWidget.cxx/h

- Creates three tabs : "Properties", "Display", "Information"

- pqProxyTabWidget is initialized in MainWindow.cxx(paraview/Applications/Client)

- Properties Tab

- Name : "Properties"

- Class name and code : pqObjectInspectorWidget, (pqObjectInspectorWidget.cxx/h)

- Display Tab

- Name : "Display"

- Class name and code : pqDisplayProxyEditorWidget.cxx/h

- UI file : paraview/Qt/Components/pqDisplayProxyEditor.ui

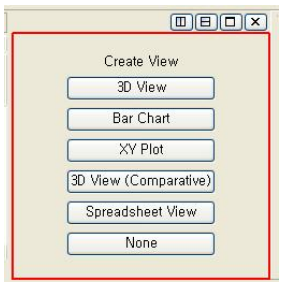
- Information Tab

- Name : "Information"

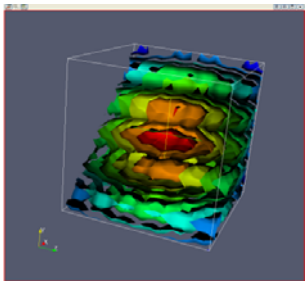
- Class name and code : pqProxyInformationWidget.cxx/h

- UI file : paraview/Qt/Components/pqProxyInformationWidget.ui

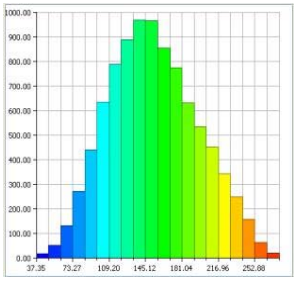
9. View의 종류와 생성



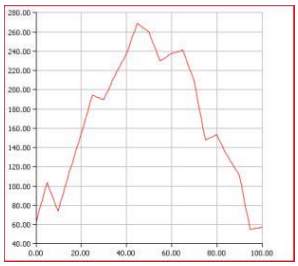
View 선택 Window



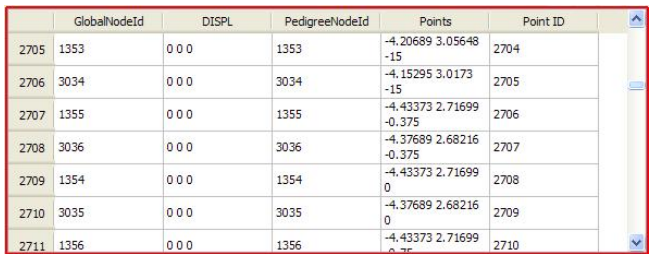
3D View



Bar Chart



XY Plot



	GlobalNodeId	DISPL	PedigreeNodeId	Points	Point ID
2705	1353	0 0 0	1353	-4.20689 3.05648 -15	2704
2706	3034	0 0 0	3034	-4.15295 3.0173 -15	2705
2707	1355	0 0 0	1355	-4.43373 2.71699 -0.375	2706
2708	3036	0 0 0	3036	-4.37689 2.68216 -0.375	2707
2709	1354	0 0 0	1354	-4.43373 2.71699 0	2708
2710	3035	0 0 0	3035	-4.37689 2.68216 0	2709
2711	1356	0 0 0	1356	-4.43373 2.71699	2710

Spread Sheet

[그림 3] ParaView의 View 유형

가. View Types

1) 3D View

ParaView의 visualization 결과를 보여주는 3D Rendering View.

- proxygroup : "views"
- view type : "RenderView"
- 관련 class : pqRenderView
- 위치 : paraview/Qt/Core

2) Bar Chart

ParaView의 pipeline 처리결과를 Bar Chart 형태로 보여주는 View.

- proxygroup : "views"
- view type : "BarChartView"

- 관련 class : pqBarChartRepresentation, pqLineChartRepresentation, classes under paraview/Qt/Chart

- 위치 : paraview/Qt/Core, paraview/Qt/Chart

3) XY Plot

ParaView의 pipeline 처리결과를 XY Plot 형태로 보여주는 View.

- proxygroup : "views"

- view type : "XY Plot"

- 관련 class : pqPlotView, pqPlotViewHistogram, pqPlotViewLineChart, classes under paraview/Qt/Chart

- 위치 : paraview/Qt/Core, paraview/Qt/Chart

4) 3D View(Comparative)

ParaView의 visualization 결과를 동시에 비교분석 할 수 있도록 한번에 4개의 Rendering View를 생성하여 결과를 보여줌.

- 각 각의 View는 RenderView(3D View)를 기본으로 함.

- 현재는 네개의 View가 모두 동일한 결과를 보여주고 있으며, Camera의 특성도 동일하게 적용되고 있음.

- proxygroup : "views"

- view type : "ComparativeRenderView"

- 관련 class : pqComparativeRenderView

- 위치 : paraview/Qt/Core

5) Spreadsheet View

ParaView의 data 처리 결과를 Spread sheet 형태로 보여주는 View.

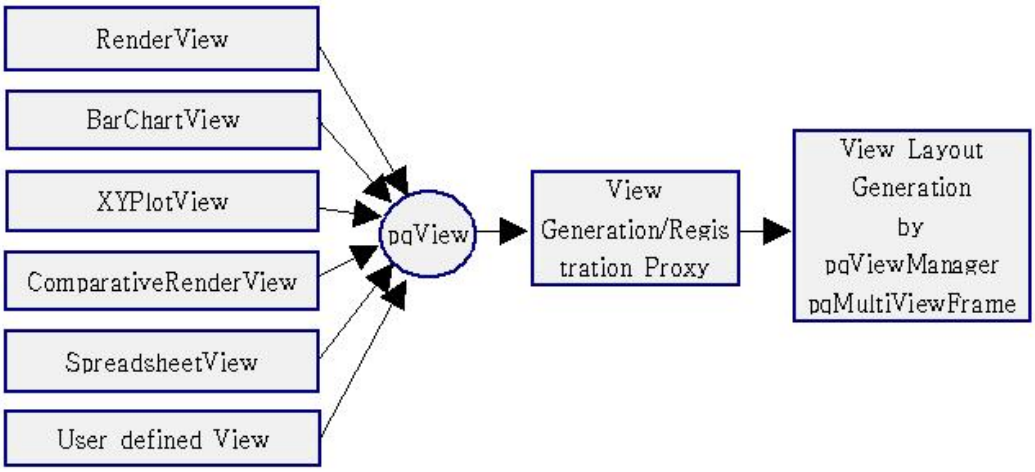
- proxygroup : "views"

- view type : "SpreadsheetView"

- 관련 class : pqSpreadSheetView, pqSpreadSheetViewModel, pqSpreadSheetViewSelectionModel

- 위치 : paraview/Qt/Core

나. View 생성



[그림 4] View 생성 과정

1) View 생성 과정

다음은 새로운 RenderView를 생성하는 과정임.

1. pqObjectBuilder::createView()
   - An object, which type is view, is created.
2. pqStandardViewModules::createView()
   - RenderView is one of the standard views,
     so pqRenderView will be created.
3. pqRenderView::pqRenderView()
   - new pqRenderView object is created
4. pqServerManagerModel::onProxyRegistered()
   - emit pqServerManagerModel: preViewAdded(view)
5. pqServerManagerModel::onProxyRegistered()
   - emit pqServerManagerModel: viewAdded(view)
6. pqViewManager::onViewAdded()
   - received viewAdded signal
7. pqViewManager::assignFrame()
   - A frame for the view is assigned to the view
8. pqViewManager::connect()
   - connects the frame and the view. The frame becomes the view's parent
9. pqRenderView::createWidget()
   - new QVTKWidget (QWidget) created.
10. pqViewManager::onActivate()
    - the frame of the created view is activated.
11. pqObjectBuilder::createView()

```
   - A new proxy for the created view is registered
     by (vtkSMProxyManager *)pxm->RegisterProxy()
12. pqObjectBuilder::createView()
   - For the created view, view->setDefaultPropertyValues()
13. pqObjectBuilder::createView()
   - viewCreated() and proxyCreated() emitted
```

- ParaView에서 이미 제공되고 있는 View와 사용자가 새로 생성하는 View는 pqView의 derived class 형태로 생성이 되며 Proxy를 통해서 view가 생성될 뿐만아니라 Server Manager와의 request/response 전달을 위하여 Proxy를 통하여 등록이 된다.
- 새로운 View 생성시 고려할 사항

View 생성 Proxy는 view의 layout을 만드는 과정에서 pqViewManager class의 assignFrame()을 호출하게되는데, 이 함수의 역할은 paraview client window에서 중앙에 위치해 있는 View Frame을 검사하여 이미 다른 view가 있는 경우, view frame을 수직/수평으로 split하여 새로운 view를 생성하고, View Frame이 Empty인 경우 View Frame의 크기를 가진 view를 생성한다.

따라서 사용자가 ParaView의 Proxy와 Server Manager의 구조를 그대로 활용할 경우(활용해야만 한다!) paraview client의 MainWindow와는 별개의 Pop-up window형태의 Rendering View를 생성하기 어렵다. 이를 위해서는 View 생성과 관련된 모듈들을 수정해야 한다(특히 User Defined View에 대해서 다른 형태의 view 생성이 가능하도록.)

2) View 생성 코드

```
<pqObjectBuilder>

pqView* pqObjectBuilder::createView(const QString& type,
  pqServer* server)
{
  if (!server)
    {
    return NULL;
    }

  vtkSMProxyManager* pxm = vtkSMProxyManager::GetProxyManager();
  vtkSMProxy* proxy= 0;

  if(type == pqRenderView::renderViewType())
    {
```

```
    proxy = server->newRenderView();
    }
  else if (type == pqComparativeRenderView::comparativeRenderViewType())
    {
    QString xmlname = server->getRenderViewXMLName();
    xmlname = "Comparative" + xmlname;
    proxy = pxm->NewProxy("views", xmlname.toAscii().data());
    }
  else  // !!! For the user-defined view
    {
    QObjectList ifaces =
      pqApplicationCore::instance()->getPluginManager()->interfaces();
    foreach(QObject* iface, ifaces)
      {
      pqViewModuleInterface* vmi = qobject_cast<pqViewModuleInterface*>(iface);
      if(vmi && vmi->viewTypes().contains(type))
        {
            proxy = vmi->createViewProxy(type);
        break;
        }
      }
    }

  if (!proxy)
    {
    qDebug() << "Failed to create a proxy for the requested view type.";
    return NULL;
    }

  proxy->SetConnectionID(server->GetConnectionID());

  QString name = QString("%1%2").arg(proxy->GetXMLName()).arg(
    this->NameGenerator->GetCountAndIncrement(proxy->GetXMLName()));


  pxm->RegisterProxy("views", name.toAscii().data(), proxy);
  proxy->Delete();

  pqServerManagerModel* model =
    pqApplicationCore::instance()->getServerManagerModel();

  pqView* view = model->findItem<pqView*>(proxy);
  if (view)
    {
    view->setDefaultPropertyValues();
    emit this->viewCreated(view);
    emit this->proxyCreated(view);
    qDebug("pqObjectBuilder::createView() - viewCreated() and proxyCreated() emitt
ed");
    }
  else
    {
    qDebug() << "Cannot locate the pqView for the "
      << "view module proxy.";
    }

  return view;
}
```

```
< pqStandardViewModule >

pqView* pqStandardViewModules::createView(const QString& viewtype,
```

```
                                                const QString& group,
                                                const QString& viewname,
                                                vtkSMViewProxy* viewmodule,
                                                pqServer* server,
                                                QObject* p)
{

  if(viewtype == pqPlotView::barChartType())
    {
    return new pqPlotView(pqPlotView::barChartType(),
                          group, viewname, viewmodule, server, p);
    }
  else if(viewtype == pqPlotView::XYPlotType())
    {
    return new pqPlotView(pqPlotView::XYPlotType(),
                          group, viewname, viewmodule, server, p);
    }
  else if(viewtype == "TableView")
    {
    // return new pqTableView(group, viewname, viewmodule, server, p);
    }

  else if (viewtype == pqSpreadSheetView::spreadsheetViewType())
    {
    return new pqSpreadSheetView(
      group, viewname, viewmodule, server, p);
    }
  else if (viewmodule->IsA("vtkSMRenderViewProxy"))
    {
    return new pqRenderView(group, viewname, viewmodule, server, p);
    }
  else if (viewmodule->IsA("vtkSMComparativeViewProxy"))
    {
    // Currently we only handle comparative render views.
    return new pqComparativeRenderView(
      group, viewname, viewmodule, server, p);
    }
  return NULL;
}
```

3) View class hierarchy

- Client-side RenderView

```
pqRenerView <- PqView <- pqProxy <- pqServerManagerModelItem <-
QObject
```

- Server-side RenderView

```
vtkSMRenderViewProxy <- vtkSMViewProxy <- vtkSMProxy <- vtkSMOb
ject <- vtkObject
```

- VTK's Qt GUI support class - VTK/GUISupport/Qt/QVTKWidget

10. ParaView Proxy List

가. Qt/Core Proxy

	Proxy	Parent Class
1	pqProxy	pqServerManagerModelItem
2	pqSMProxy	pqProxy

[표 4] Qt/Core Proxy List

나. Servers/ServerManager Proxy

	Proxy	Parent Class
1	vtkSMAnimationCueManipulatorProxy	vtkSMProxy
2	vtkSMAnimationCueProxy	vtkSMProxy
3	vtkSMAnimationCueProxy	vtkSMProxy
4	vtkSMAnimationSceneProxy	vtkSMAnimationCueProxy
5	vtkSMAxesRepresentationProxy	vtkSMRepresentationProxy
6	vtkSMBooleanKeyFrameProxy	vtkSMKeyFrameProxy
7	vtkSMBoxProxy	vtkSMProxy
8	vtkSMCameraKeyFrameProxy	vtkSMKeyFrameProxy
9	vtkSMCameraManipulatorProxy	vtkSMKeyFrameAnimationCueManipulatorProxy
10	vtkSMCameraProxy	vtkSMProxy
11	vtkSMCaveRenderModuleProxy	vtkSMCompositeRenderModuleProxy
12	VtkSMClientDeliveryStrategyProxy	vtkSMSimpleStrategy
13	VtkSMClientServerRenderViewProxy	vtkSMMultiProcessRenderView
14	VtkSMComparativeViewProxy	vtkSMViewProxy
15	VtkSMCompositeKeyFrameProxy	vtkSMKeyFrameProxy
16	VtkSMCompoundProxy	vtkSMProxy
17	VtkSMConnectionCleanerProxy	vtkSMProxy
18	VtkSMDataLabelRepresentationProxy	vtkSMDataRepresentationProxy
19	vtkSMDataRepresentationProxy	vtkSMRepresentationProxy
20	VtkSMExponentialKeyFrameProxy	vtkSMKeyFrameProxy
21	vtkSMExtractFrustumProxy	vtkSMSourceProxy
22	vtkSMExtractLocationsProxy	vtkSMSourceProxy
23	vtkSMExtractSelectionProxy	vtkSMSourceProxy
24	vtkSMExtractSelectionsProxy	vtkSMSourceProxy

25	vtkSMExtractThresholdsProxy	vtkSMSourceProxy
26	vtkSMFileSeriesReaderProxy	vtkSMSourceProxy
27	vtkSMIceTCompositeViewProxy	vtkSMMultiProcessRenderView
28	vtkSMIceTDesktopRenderViewProxy	vtkSMIceTCompositeViewProxy
29	VtkSMIceTMultiDisplayRenderViewProxy	vtkSMIceTDesktopRenderViewProxy
30	VtkSMImplicitPlaneProxy	vtkSMProxy
31	VtkSMKeyFrameProxy	vtkSMProxy
32	VtkSMLookupTableProxy	vtkSMProxy
33	VtkSMMaterialLoaderProxy	vtkSMProxy
34	VtkSMMultiDisplayProxy	vtkSMCompositeDisplayProxy
35	VtkSMNewWidgetRepresentationProxy	vtkSMRepresentationProxy
36	VtkSMNullProxy	vtkSMProxy
37	vtkSMPropRepresentationProxy	vtkSMDataRepresentationProxy
38	vtkSMProxy	vtkSMObject
39	vtkSMPVDWriterProxy	vtkSMWriterProxy
40	vtkSMPVLookupTableProxy	vtkSMProxy
41	vtkSMPVRepresentationProxy	vtkSMProxy
42	VtkSMPWriterProxy	vtkSMWriterProxy
43	VtkSMRampKeyFrameProxy	vtkSMKeyFrameProxy
44	vtkSMRenderViewProxy	vtkSMViewProxy
45	VtkSMRepresentationProxy	vtkSMProxy
46	VtkSMScalarBarActorProxy	vtkSMProxy
47	VtkSMSelectionLinkProxy	vtkSMSourceProxy
48	vtkSMSelectionRepresentationProxy	vtkSMDataRepresentationProxy
49	VtkSMServerFileListingProxy	vtkSMProxy
50	VtkSMSinusoidKeyFrameProxy	vtkSMKeyFrameProxy
51	VtkSMSourceProxy	vtkSMProxy
52	vtkSMSpreadSheetRepresentationProxy	vtkSMBlockDeliveryRepresentationProxy
53	VtkSMSummaryHelperProxy	vtkSMProxy
54	vtkSMSurfaceRepresentationProxy	vtkSMPropRepresentationProxy
55	VtkSMTextSourceRepresentationProxy	vtkSMDataRepresentationProxy
56	vtkSMTextWidgetRepresentationProxy	vtkSMNewWidgetRepresentationProxy
57	vtkSMTimeAnimationCueProxy	vtkSMAnimationCueProxy
58	VtkSMTimeKeeperProxy	vtkSMProxy
59	VtkSMTransformProxy	vtkSMProxy
60	VtkSMUpdateSuppressorProxy	vtkSMSourceProxy

61	VtkSMViewProxy	vtkSMProxy
62	VtkSMWidgetRepresentationProxy	vtkSMProxy
63	vtkSMWriterProxy	vtkSMSourceProxy
64	VtkSMXMLPVAnimationWriterProxy	vtkSMSourceProxy
65	VtkSMXYPlotRepresentationProxy	vtkSMClientDeliveryRepresentation Proxy

[표 5] ServerManager Proxy List

다. Servers/Filters Proxy

- vtkPVRenderViewProxy <- vtkObject

- client GUI 코드와 server의 코드를 분리하기 위해 사용하며 이 class가 수신한 call은 모두 vtkPVRenderView 클래스로 전달됨.

11. ParaView Source

가. Source List

참고 : paraview/Servers/ServerManager/Resources/sources.xml

	Function	VTK or ParaView Class
1	2D Glyph	vtkGlyphSource2D
2	3D Text	vtkVectorText
3	Arrow	vtkPVArrowSource
4	Axes	vtkAxes
5	Outline	vtkOutlineSource
6	Box	vtkCubeSource
7	Cone	vtkConeSource
8	Hierarchical Fractal	vtkHierarchicalFractal
9	Cylinder	vtkCylinderSource
0	Line	vtkLineSource
11	Mandelbrot	vtkImageMandelbrotSource
12	Octree Fractal	vtkHyperOctreeFractalSource
13	Plane	vtkPlaneSource
14	Point Source	vtkPointSource
15	Sphere	vtkSphereSource
16	Superquadric	vtkSuperquadricSource
17	Wavelet	vtkRTAnalyticSource
18	Test3DWidget	vtkConeSource
19	NetworkImageSource	vtkNetworkImageSource
20	TextSource	vtkPVTextSource
21	SelectionLink	vtkSelectionLink
22	SelectionSource	vtkSelectionSource
23	Annotate Time	vtkTimeToTextConvertor
24	Time Source	vtkTimeSourceExample

[표 6] ParaView Source List

나. Source Creation Module Snippet

```
pqPipelineSource* pqObjectBuilder::createSource(const QString& sm_group,
    const QString& sm_name, pqServer* server)
{
```

```
  vtkSMProxy* proxy =
    this->createProxyInternal(sm_group, sm_name, server, "sources");
  if (proxy)
    {
    pqPipelineSource* source = pqApplicationCore::instance()->
      getServerManagerModel()->findItem<pqPipelineSource*>(proxy);

    // initialize the source.
    source->setDefaultPropertyValues();
    source->setModifiedState(pqProxy::UNINITIALIZED);

    emit this->sourceCreated(source);
    emit this->proxyCreated(source);
    return source;
    }
  return 0;
}
```

12. ParaView Readers

가. Reader List

참고 : paraview/Servers/ServerManager/Resources/readers.xml

	Function	VTK or ParaView Class
1	Legacy VTK reader	vtkPDataSetReader
2	DEM reader	vtkDEMReader
3	XML Multi-Block Data reader	vtkXMLMultiBlockDataReader
4	XML Multi-Group Data reader	vtkXMLMultiGroupDataReader
5	XML Hierarchical Data reader	vtkXMLHierarchicalDataReader
6	XML Hierarchical Box Data reader	vtkXMLHierarchicalBoxDataReader
7	PVD reader	vtkPVDReader
8	XML Polydata reader	vtkXMLPolyDataReader
9	RTXMLPolyDataReader	vtkRTXMLPolyDataReader
0	XML Unstructured Grid reader	vtkXMLUnstructuredGridReader
11	XML Unstructured Grid reader	vtkXMLImageDataReader
12	XML Image Data reader	vtkXMLImageDataReader
13	XML Structured Grid reader	vtkXMLStructuredGridReader
14	XML Rectilinear Grid reader	vtkXMLRectilinearGridReader
15	XML Partitioned Polydata reader	vtkXMLPPolyDataReader

16	XML Partitioned Unstructured Grid reader	vtkXMLPUnstructuredGridReader
17	XML Partitioned Image Data reader	vtkXMLPImageDataReader
18	XML Partitioned Structured Grid reader	vtkXMLPStructuredGridReader
19	XML Partitioned Rectilinear Grid reader	vtkXMLPRectilinearGridReader
20	EnSight reader	vtkGenericEnSightReader
21	Spy Plot reader	vtkSpyPlotReader
22	VRML reader	vtkVRMLSource
23	PLY reader	vtkPLYReader
24	PDB reader	vtkPDBReader
25	XYZ reader	vtkXYZMolReader
26	PLOT3D reader	vtkMultiBlockPLOT3DReader
27	Phasta reader	vtkPPhastaReader
28	STL reader	vtkSTLReader
29	BYU reader	vtkBYUReader
30	Gaussian Cube reader	vtkGaussianCubeReader
31	POP reader	vtkPOPReader
32	EnSight Master Server reader	vtkPVEnSightMasterServerReader
33	Image reader	vtkImageReader
34	XDMF reader	vtkXdmfReader
35	Exodus reader	vtkPExodusReader
36	ExodusIIReader	vtkPExodusIIReader
37	AVS UCD reader	vtkAVSucdReader
38	Facet Reader	vtkFacetReader
39	Meta Image Reader	vtkMetaImageReader
40	PNG reader	vtkPNGReader
41	SESAME reader	vtkSESAMEReader
42	OpenFOAMReader	vtkOpenFOAMReader
43	MFIXReader	vtkMFIXReader
44	FLUENTReader	vtkFLUENTReader
45	LSDynaReader	vtkLSDynaReader
46	CSV reader	vtkCSVReader

[표 7] Reader List

나. Reader Creation Module Snippet

```
pqPipelineSource* pqObjectBuilder::createReader(const QString& sm_group,
    const QString& sm_name, const QStringList& files, pqServer* server)
{
```

```
if (files.empty())
  {
  return 0;
  }

QFileInfo fileInfo(files[0]);

vtkSMProxy* proxy =
  this->createProxyInternal(sm_group, sm_name, server, "sources",
    fileInfo.fileName());
if (!proxy)
  {
  return 0;
  }

pqPipelineSource* reader = pqApplicationCore::instance()->
  getServerManagerModel()->findItem<pqPipelineSource*>(proxy);
if (!reader)
  {
  qDebug() << "Failed to locate pqPipelineSource for the created proxy "
    << sm_group << ", " << sm_name;
  return 0;
  }

QString pname = this->getFileNamePropertyName(proxy);
if (!pname.isEmpty())
  {
  vtkSMStringVectorProperty* prop =
    vtkSMStringVectorProperty::SafeDownCast(
      proxy->GetProperty(pname.toAscii().data()));
  if (!prop)
    {
    return 0;
    }

  unsigned int numElems = files.size();
  if (numElems == 1 || !prop->GetRepeatCommand())
    {
    pqSMAdaptor::setElementProperty(prop, files[0]);
    }
  else
    {
    QList<QVariant> values;
    foreach (QString file, files)
      {
      values.push_back(file);
```

```
        }
      pqSMAdaptor::setMultipleElementProperty(prop, values);
      }
    proxy->UpdateVTKObjects();
    prop->UpdateDependentDomains();
    }
  reader->setDefaultPropertyValues();
  reader->setModifiedState(pqProxy::UNINITIALIZED);

  emit this->readerCreated(reader, files[0]);
  emit this->proxyCreated(reader);
  return reader;
}
```

다. 지원하는 파일 포맷

1. Exodus(*.ex_old)
2. ExodusII(*.g *.e *.ex2 *.ex2v2 *.exo *.gen *.exoII *.0 *.00 *.000 *.0000 *.exii)
3. LSDyna(*.d3plot *.k *.lsdyna)
4. ParaView Data Files(*.pvd)
5. VTK PolyData Files(*.vtp)
6. VTK UnstructuredGrid Files(*.vtu)
7. VTK ImageData Files(*.vti)
8. VTK StructuredGrid Files(*.vts)
9. VTK RectilinearGrid Files(*.vtr)
10. VTK PolyData Files (partitioned)(*.pvtp)
11. VTK UnstructuredGrid Files (partitioned)(*.pvtu)
12. VTK ImageData Files (partitioned)(*.pvti)
13. VTK StructuredGrid Files (partitioned)(*.pvts)
14. VTK RectilinearGrid Files (partitioned)(*.pvtr)
15. VTK MultiBlock Data Files(*.vtm *.vtmb)
16. VTK MultiGroup Data Files(*.vtm *.vtmg)
17. VTK Hierarchical Data Files(*.vtm *.vthd)
18. VTK Hierarchical Box Data Files(*.vtm *.vthb)

19. Legacy VTK files(*.vtk)
20. Legacy VTK Files (partitioned)(*.pvtk)
21. EnSight Files(*.case *.CASE *.Case)
22. EnSight Master Server Files(*.sos *.SOS)
23. BYU Files(*.g)
24. Xdmf Reader(*.xmf *.xdmf)
25. Protein Data Bank Files(*.pdb)
26. XMol Molecule Files(*.xyz)
27. PLOT3D Files(*.xyz)
28. SpyPlot CTH dataset(*.spcth *.0)
29. Digital Elevation Map Files(*.dem)
30. VRML 2 Files(*.wrl)
31. PLY Polygonal File Format(*.ply)
32. Stereo Lithography(*.stl)
33. Gaussian Cube Files(*.cube)
34. Raw (binary) Files(*.raw)
35. POP Ocean Files(*.pop)
36. AVS UCD Binary/ASCII Files(*.inp)
37. Meta Image Data Files(*.mhd *.mha)
38. Facet Polygonal Data Files(*.facet)
39. PNG Image Files(*.png)
40. Phasta Files(*.pht)
41. SESAME(*.sesame)
42. Comma-separated-values(*.csv)

13. ParaView Visualization State File 분석

ParaView는 visualization 처리 결과를 XML 포맷으로 *.pvsm 파일에 저장한다. ParaView에서 처리한 결과를 그대로 Virutal Environment에서 display할 수 있도록 하기 위하여 PVSM(ParaView State Manager) 파일을 분석한 결과는 다음과 같다.

가. PVSM File 분석

"ServerManagerState" 라는 element안에 여러 개의 "Proxy group" element와 "ProxyCollection" element가 존재.

예) state.pvsm

	"Proxy group" element의 종류	"ProxyCollection" element의 종류
1	animation	animation
2	cameramanipulators	pq_helper_proxies
3	piecewise_functions	representations
4	representations	sources
5	misc	views
6	views	none

[표 8] PVSM 파일의 Proxy group의 종류

나. Source types

1) Reader

- visualization data가 file에 저장이 되어 있는 형태 : type attribute의 값에 data file의 reader name이 명시되어 있음. 아래 예의 경우에는 reader의 name은 "PVDReader"임.
- <Property name="FileName"> element의 nested element에 full filename이 명시되어 있음.
- 예

```
<Proxy group="sources" type="PVDReader" id="80" servers="1">
 <Property name="FileName" id="80.FileName" number_of_elements="1">
   <Element index="0" value="/home/bhkeum/Desktop/ParaViewData/made_can.pvd"/>
    <Domain name="files" id="80.FileName.files"/>
  </Property>
</Proxy>
```

2) Source

- Visualization data가 Cone, Cylinder, Cube와 같은 VTK source object인 경우
- 데이터가 파일에 따로 저장이 되어 있지 않고 object의 geometry information이 명시되어 있음

● 예)

```
<Proxy group="sources" type="CubeSource" id="84" servers="1">
  <Property name="Center" id="84.Center" number_of_elements="3">
    <Element index="0" value="0"/>
    <Element index="1" value="0"/>
    <Element index="2" value="0"/>
    <Domain name="range" id="84.Center.range"/>
  </Property>
  <Property name="XLength" id="84.XLength" number_of_elements="1">
     <Element index="0" value="1"/>
     <Domain name="range" id="84.XLength.range">
       <Min index="0" value="0"/>
     </Domain>
  </Property>
  <Property name="YLength" id="84.YLength" number_of_elements="1">
    <Element index="0" value="1"/>
    <Domain name="range" id="84.YLength.range">
      <Min index="0" value="0"/>
    </Domain>
  </Property>
  <Property name="ZLength" id="84.ZLength" number_of_elements="1">
    <Element index="0" value="1"/>
    <Domain name="range" id="84.ZLength.range">
      <Min index="0" value="0"/>
    </Domain>
  </Property>
</Proxy>
```

다. VR Object Information in PVSM

1) Source Type이 Reader인 경우

<Proxy group="sources"> element의 정보를 분석한다.

Property name	기능	VR 적용	관련 VTK class 혹은 function
FileName	visualization data가 저장이 되어 있는 filename 명시	Yes	Reader name에 따라 해당 VTK reader를 mapping 해주거나 VTK에서 지원하지 않는 경우 class를 새로 작성해야함
Timestep Values	time step 값이 있는 경우 time step값 명시	-	-

2) Source Type이 VTK Source인 경우

<Proxy group="sources"> element의 정보를 분석한다.

Property name	기능	VR 적용	관련 VTK class 혹은 function
Center	object의 center 명시	Yes	-
XLength	object의 x축 길이	-	-
YLength	object의 y축 길이	-	-
ZLength	object의 z축 길이	-	-

3) View 정보 분석

<Proxy group="views"> element의 정보를 분석한다.

Property name	기능	VR 적용	관련 VTK class 혹은 function
GUISize	view window의 크기	No	-
LODResolution	-	No	-
LODThreshold	-	No	-
RenderInterruptsEnabled	-	No	-
Representations	-	No	-
UseImmediateMode	-	No	-
UseLight	Light 사용여부. 아래에 있는 LightSwitchOn, Off에서 설정	No	-
UseTriangleStrips	Triangle Strip의 사용여부	No	-
ViewPosition	view position 값	Yes	-
ViewTime	-	No	-
BackLightAzimuth	back light에 대한 방위(longitude) 값	Yes	vtkLightKit, SetBackLightAzimuth()
BackLightElevation	back light에 대한 높이 값	Yes	vtkLightKit, SetBackLightElevation()
BackLightK:B Ratio	key light에 대한 back light의 밝기값 비율	Yes	vtkLightKit, SetKeyToBackRatio()
BackLightWarmth	back light에 대한 warmth color 값	Yes	vtkLightKit, SetBackLightWarmth()
Background	background 색상	-	-
CameraClippingRange	가까운 clipping plane과 먼	Yes	vtkCamera,

	clipping plane의 거리 값을 정의		SetClippingRange(near,far)
CameraClippingRangeInfo	위와 동일	No	–
CameraFocalPoint	Camera의 초점 위치	Yes	vtkCamera, SetFocalPoint(x,y,z)
CameraFocalPointInfo	위와 동일	No	–
CameraManipulators	–	–	–
CameraParallelProjection	Camera Parallel Projection 여부	–	–
CameraParallelScale	–	–	–
CameraPosition	Camera의 위치	–	vtkCamera, SetPosition(x,y,z)
CameraPositionInfo	위와 동일	No	–
CameraViewAngle	degree로 표현되는 Camera view angle 값 정의	Yes	vtkCamera, SetViewAngle(angle)
CameraViewUp	카메라 위치를 기준으로 하는 위치벡터값 정의	Yes	vtkCamera, SetViewUp(vx,vy,vz)
CameraViewUpInfo	위와 동일	No	–
CenterOfRotation	object의 회전 중심 좌표	Yes	–
DepthPeeling	DepthPeeling 여부	No	–
FillLightAzimuth	Fill Light에 대한 방위(longitude) 값	Yes	vtkLightKit, SetFillLightAzimuth()
FillLightElevation	Fill Light에 대한 높이 값	Yes	vtkLightKit, SetFillLightElevation()
FillLightK:F Ratio	key light에 대한 fill light의 밝기 값 비율	Yes	vtkLightKit, SetKeyToFillRatio()
FillLightWarmth	fill light의 warmth color 값	Yes	vtkLightKit, SetFillLightWarmth()
HeadLightK:H Ratio	key light에 대한 headlight의 밝기 값 비율	Yes	vtkLightKit, SetKeyToHeadRatio()
HeadLightWarmth	head light의 warmth color 값	Yes	vtkLightKit, SetHeadlightWarmth()
KeyLightAzimuth	key light의 방위(longitude) 값	Yes	vtkLightKit, SetKeyLightAzimuth()
KeyLightElevation	key light의 고도 값	Yes	vtkLightKit, SetKeyLightElevation()
KeyLightIntensity	key light의 밝기 값	Yes	vtkLightKit, SetKeyLightIntensity()
KeyLightWarmth	key light의 warmth color 값	Yes	vtkLightKit, SetKeyLightWarmth()
LightAmbientColor	ambient surface color 값	Yes	vtkProperty, SetAmbientColor()
LightDiffuseColor	diffuse surface color 값	Yes	vtkProperty, SetDiffuseColor()
LightIntensity	위에 있는 key light intensity와 동일한 것 같음	–	–
LightSpecularColor	specular surface color 값	Yes	vtkProperty, SetSpecularColor()
LightSwitch	Light Switch on/off 여부	Yes	vtkLight, SwitchOn(), SwitchOff(),

			SetSwitch()
MaintainLuminance	If MaintainLuminance is set, the LightKit will attempt to maintain the apparent intensity of lights based on their perceptual brightnesses. By default, MaintainLuminance is off	No	vtkLightKit, SetMaintainLuminance()
RenderWindowSizeInfo	view window의 크기. GUISize와 중복	No	-
ViewSize	위와 동일	No	-

[표 9] "views" proxy group의 element 정보 분석

라. ParaView Light(조명) 설정 분석

1) Light 종류

- key light
  - main light
  - 해나 천장에 달려있는 조명과 같이 머리위에 위치하는 라이트를 말함
  - 주로 머리위에서 45도 각도로 scene을 비추고 scene에 있는 다른 light를 모두 합한 빛의 세기의 2배 이상의 밝기를 가진다.

- fill light
  - 주로 key light의 반대편에 위치

- headlight
  - 현재 카메라의 위치와 동일한 위치에 있는 라이트로 카메라의 초첨 위치에 빛을 비추는 라이트를 말한다.
  - key light와 fill light사이의 영역의 contrast를 줄이는 역할을 한다.
  - vtkLight 혹은 vtkLightKit class를 사용하여 선언한다.

- backlight

- 두 개의 backlight가 있으며 관찰자 위치로부터 object의 왼쪽에 하나 오른쪽에 하나가 위치한다.
- 주로 object 뒤쪽 영역의 high-contrast를 보완하는데 사용한다.

2) Light Handling

- 서로 다른 종류의 light들을 사용할 때 이들간의 관계를 만들어주기 위하여 fill light, back light, headlight간의 intensity를 key light 밝기와의 비율로 설정을 할 수 있다.
- key light 밝기와의 비율로 설정할 경우 key light의 밝기를 변경하면 다른 light들도 따라서 바뀌는 장점이 있다.
- light들은 카메라가 움직임에 따라 같이 움직이고 카메라의 초점위치로 빛을 비춘다.
- light의 위치는 카메라에 대한 elevation(latitude)와 azimuth(longitude) 값으로 설정된다.
- light가 (elevation=0, azimuth=0)으로 설정되면 카메라의 위치에 있게되며 이것은 headlight가 된다.
- light가 (elevation=90, azimuth=0)으로 설정되면 카메라가 보는 위치의 위에 있게 되며 위에서 비추는 key light가 된다.
- azimuth값이 0보다 작으면 위에서 봤을 때 시계 방향으로 위치한 light를 의미한다.