



ParaView 활용: 렌더링과 보고서 자동화

2023.10.19

NextFOAM. Co., Ltd.

박현강, 이현웅

서론

1. CFD 수요 증가

- 전통적인 조선해양, 공력해석 뿐 아니라 소 축사 내부 온도 해석, 실리콘 제조, 심지어 비닐 하우스 내부 온도 분포까지 해석을 원하는 수요가 증가함.

2. 직관적인 후처리 결과 필요

- 자연스럽게 CFD를 잘모르는 사람들도 CFD관련 결과보고서를 접함.
- 따라서 엔지니어들만 알아볼 수 있는 단순숫자표기 보다 이미지나 동영상의 형태로 결과를 보여주는 작업이 필요함. 그러나 일반적으로 CFD 결과를 보기 좋게 만드는 일은 많은 수고와 시간이 소요됨.

3. 너무 많은 후처리 작업량

- 컴퓨터 성능이 올라가면서 최적화, 민감도 테스트 등에 더 많은 케이스를 집어 넣을 수 있게 됨.
- 그러나 그 만큼 단순 노동인 후처리 양도 늘어남.

멋지고 한눈에 들어오는 결과물을 편하게 나오게 하고 싶다!

→ **ParaView를 활용한 렌더링 테크닉과 보고서 자동화**

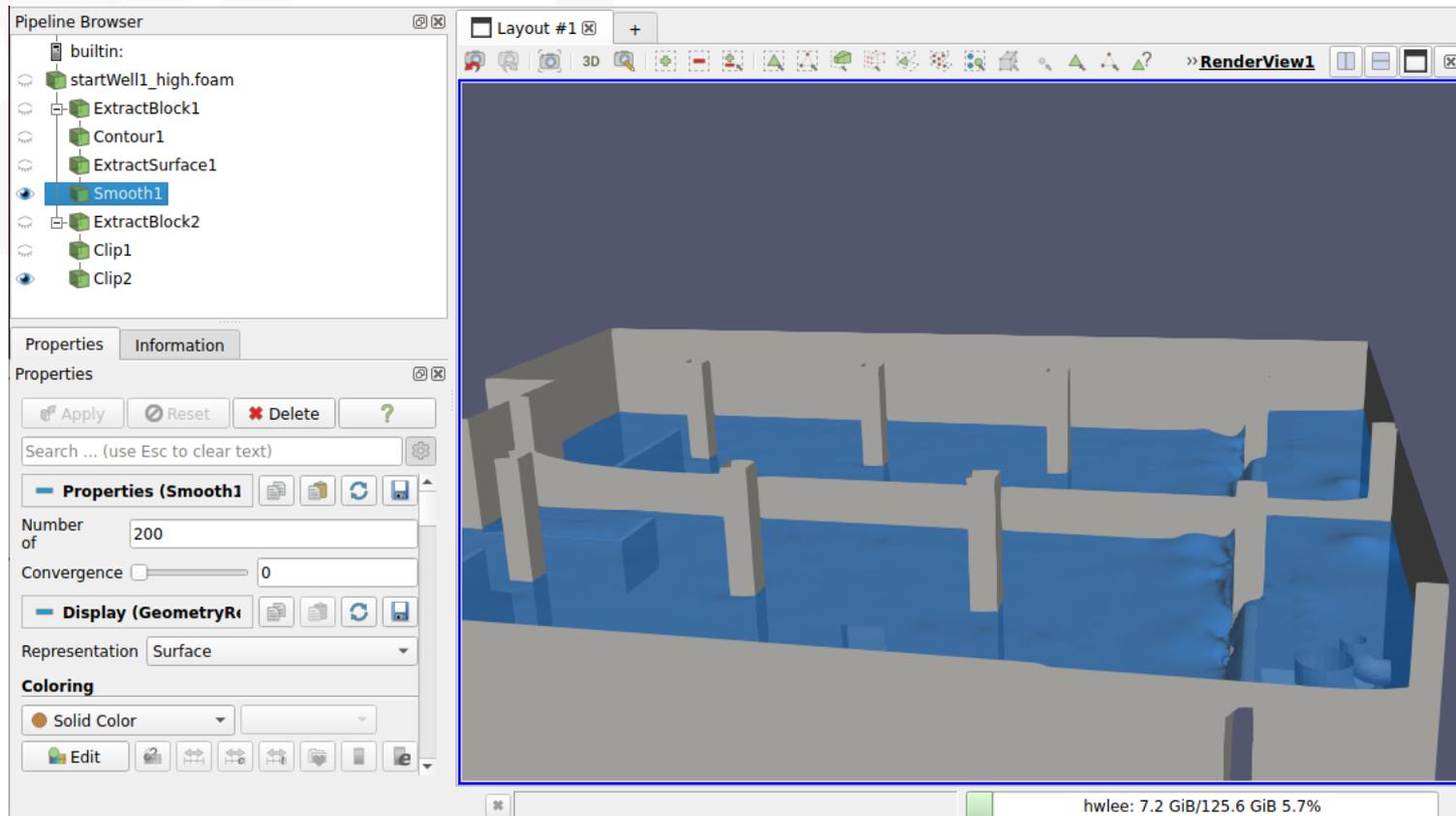
목차

1. ParaView 렌더링
2. ParaView 데이터 처리
3. 보고서 자동화

1. ParaView 렌더링

1. 레이 트레이싱

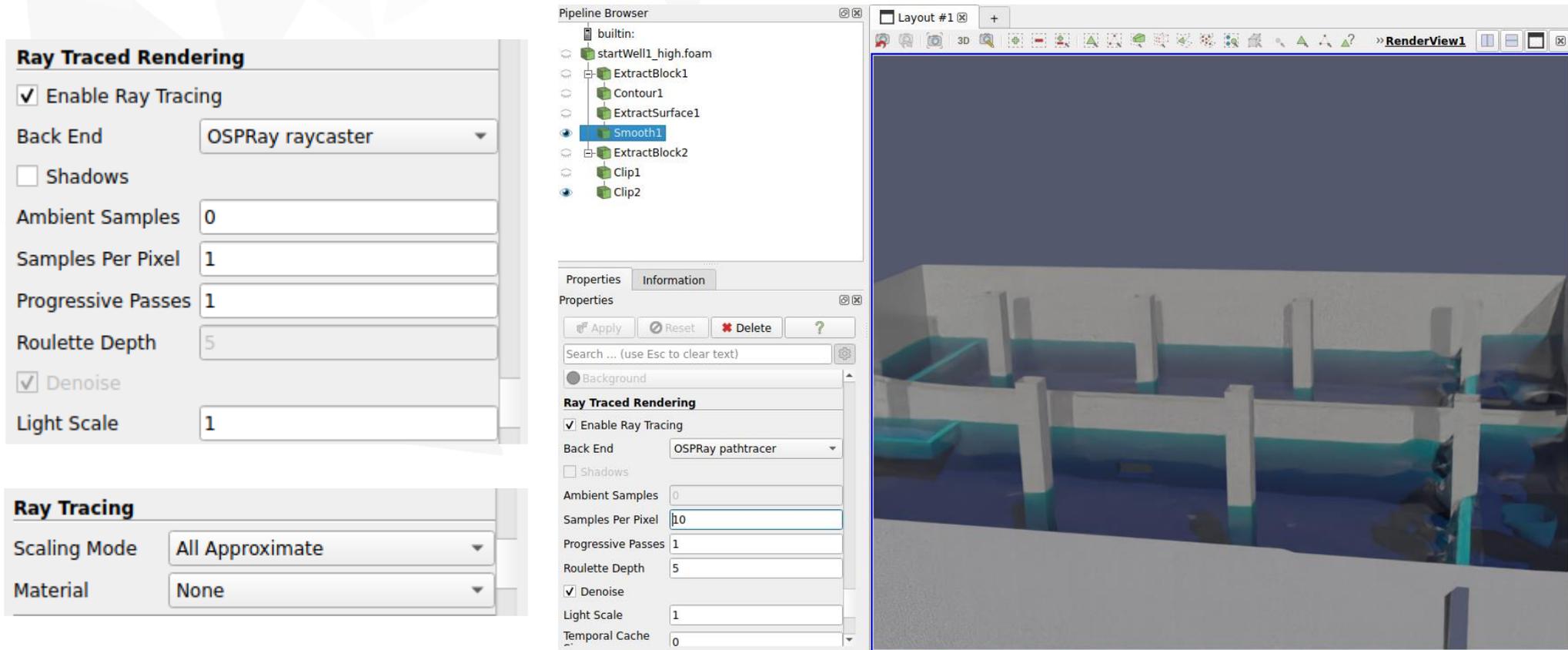
- 모 정수장의 형상의 일부
- 벽 형상에 Clip도 하고 Water Contour에 smooth도 사용해봤지만 그닥 이쁘지 않다.



1. ParaView 렌더링

1. 레이 트레이싱

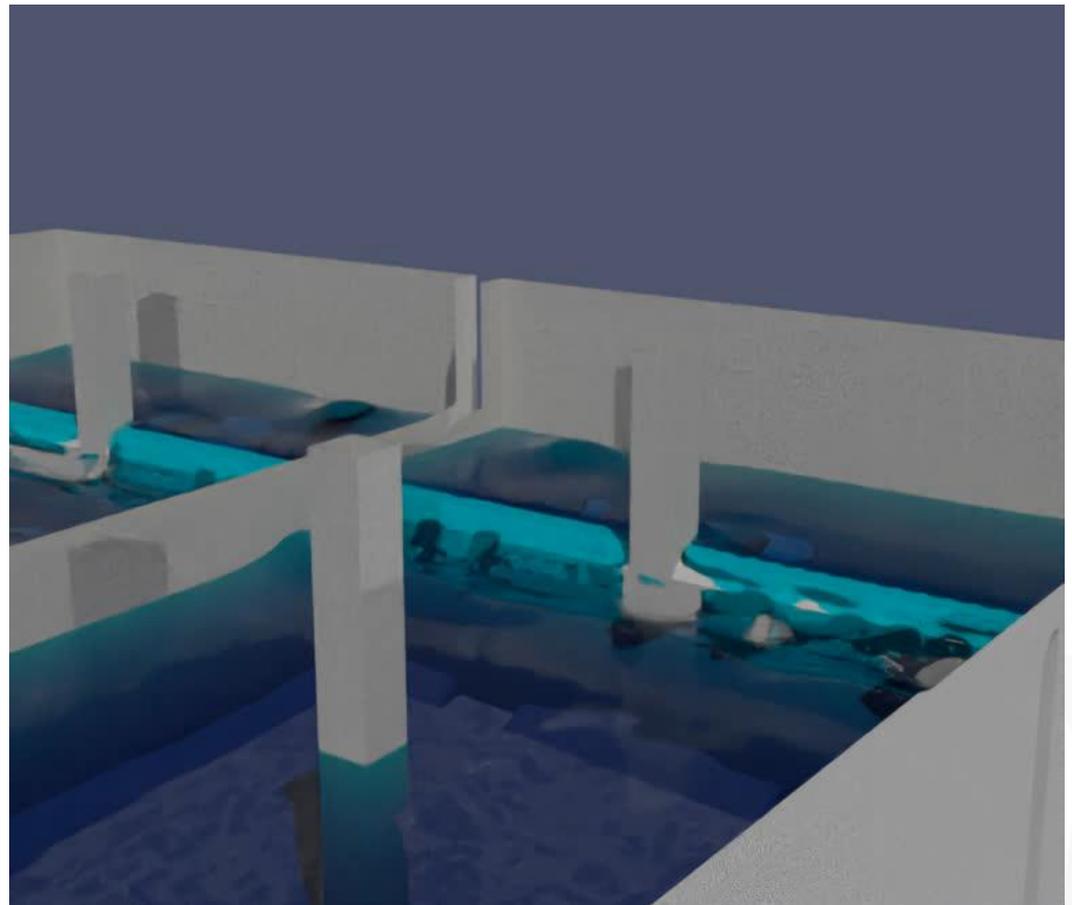
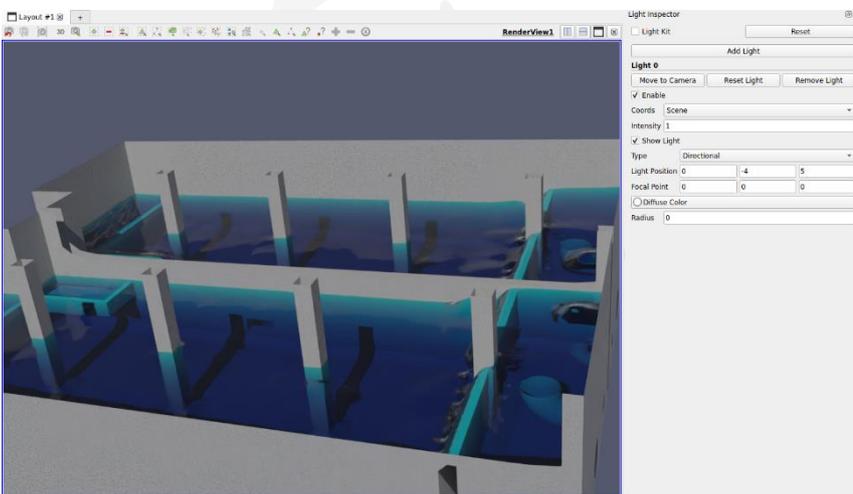
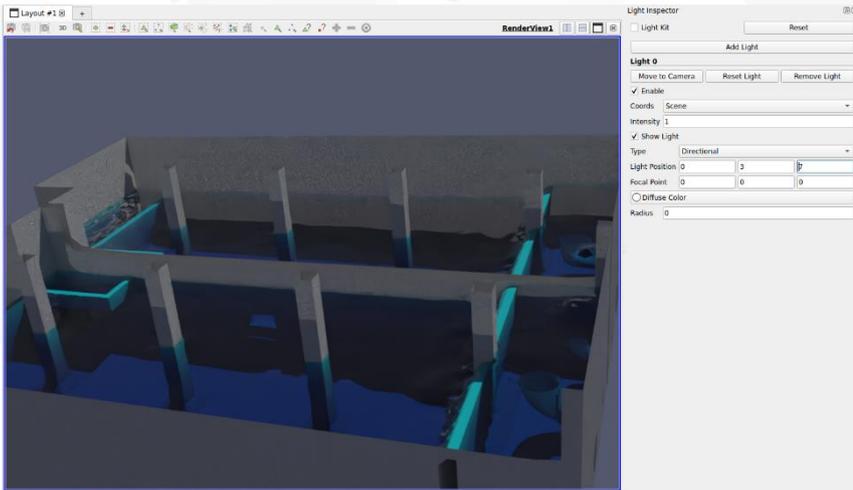
- ParaView Render Properties 가장 하단에 Ray Tracing을 적용
- Material을 적용하면 단숨에 그럴 듯 해진다.



1. ParaView 렌더링

1. 레이 트레이싱

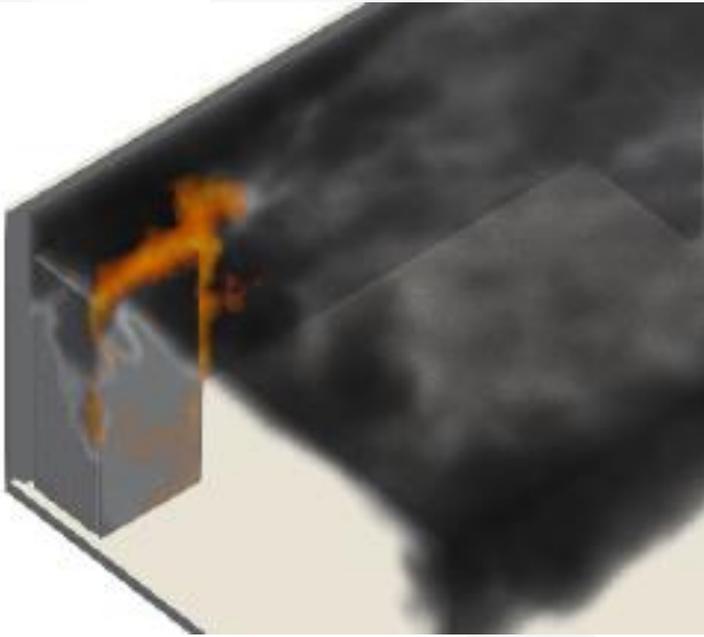
- Light Inspector를 사용해 조명을 원하는 대로 조절할 수 있다.



1. ParaView 렌더링

1. 레이 트레이싱

- 실내 화재 해석에서의 후처리 사용례



Time: 0.000000

Time: 0.000000



1. ParaView 렌더링

1. 레이 트레이싱

- Material 설정법
- 설정 파일 위치 => [paraview경로] / share / [paraview버전] / materials / ospray_mats.json
- 공식 깃랩 참조 https://gitlab.kitware.com/paraview/materials/-/blob/master/ospray_mats.json
- 인텔 OSPRay 참조 <https://www.ospray.org/documentation.html#materials>

• 지원되는 Material Type

- 1) Principled
- 2) CarPaint
- 3) Metal
- 4) Glass
- 5) Luminous
- 6) ...

```
"Glass_water": {
  "type": "Glass",
  "doubles": {
    "attenuationColor": [0.00, 0.467, 0.745],
    "etaInside": [1.33]
  },
}

"Fire1": {
  "type": "Luminous",
  "doubles": {
    "color": [0.8, 0.2, 0.0],
    "intensity": [0.3],
    "transparency": [0.01]
  }
}
```

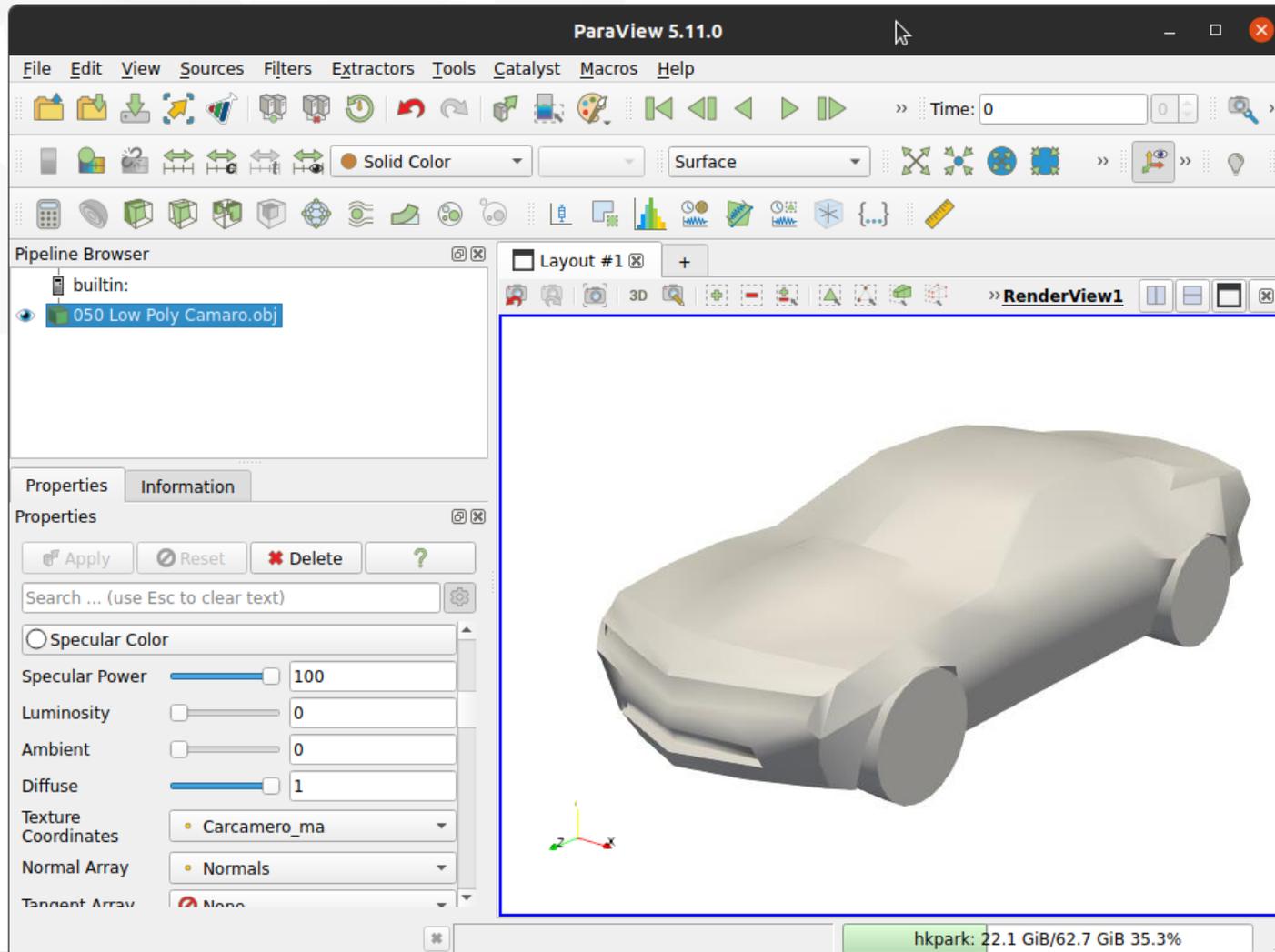
1. ParaView 렌더링

[모델링 파일 출처](https://www.sketchoverflow.com/product/low-poly-camaro-car-3d-model/)

<https://www.sketchoverflow.com/product/low-poly-camaro-car-3d-model/>

2. 텍스처

- STL파일이나 OBJ파일에 텍스처를 입히기 위한 좌표값이 있다면, 텍스처를 불러올 수 있다.



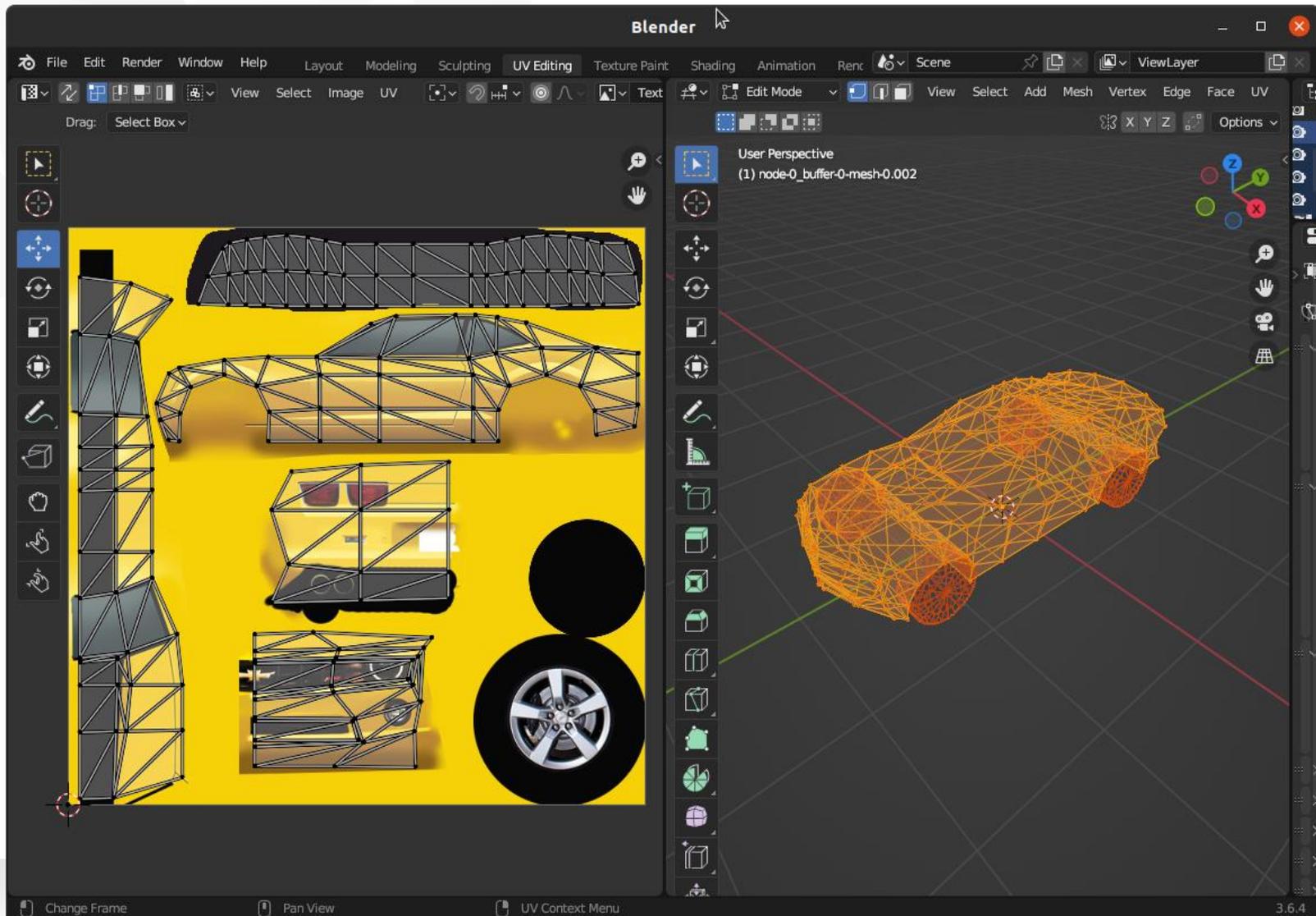
1. ParaView 렌더링

모델링 파일 출처

<https://www.sketchoverflow.com/product/low-poly-camaro-car-3d-model/>

2. 텍스처

- OBJ 파일에 좌표값을 넣는것은 Blender를 이용해 작업할 수 있다.

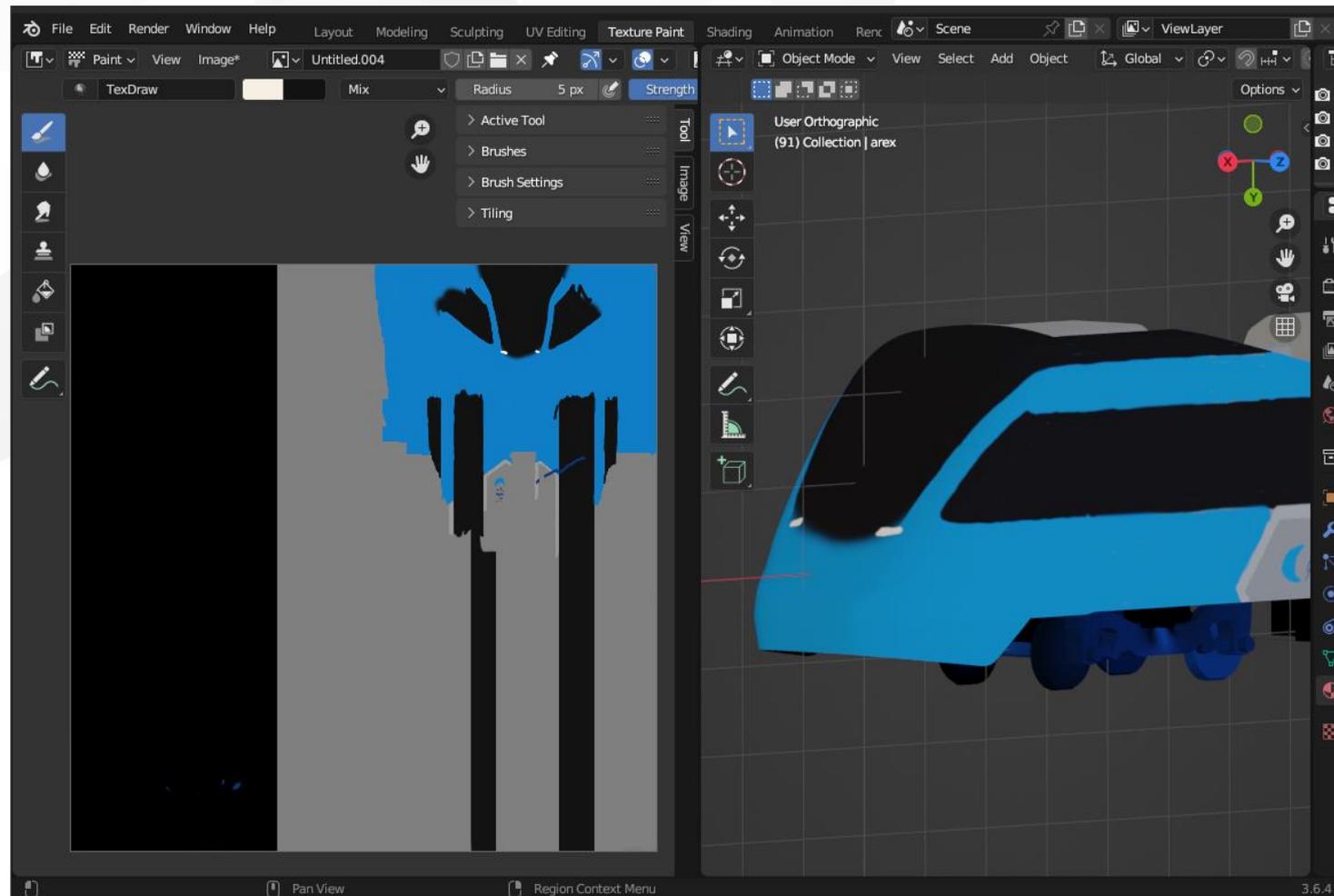


1. ParaView 렌더링

우측 사진 출처: 포커스 인천 뉴스
<http://www.focusincheon.com/news/articleView.html?idxno=1767>

2. 텍스처

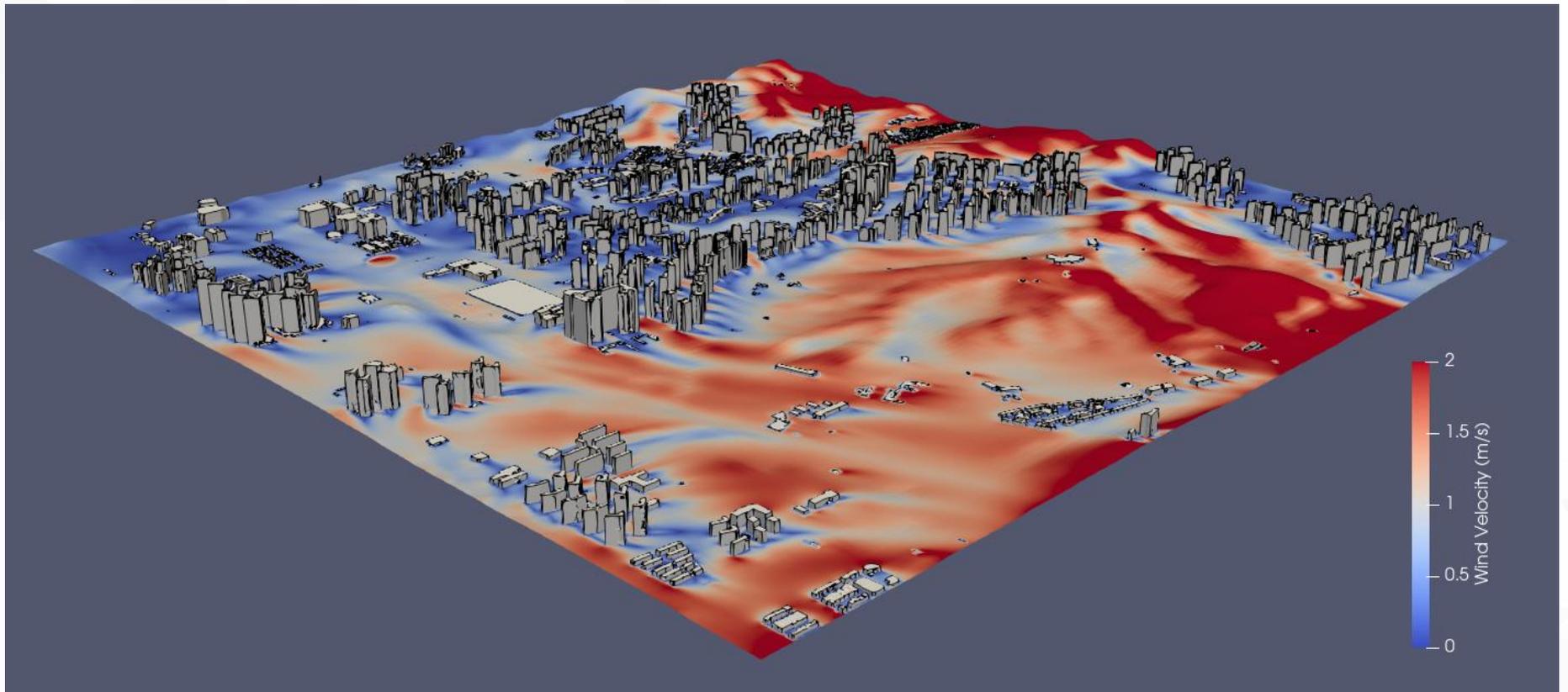
- 주의! 금손만 작업하세요...



1. ParaView 렌더링

2. 텍스처

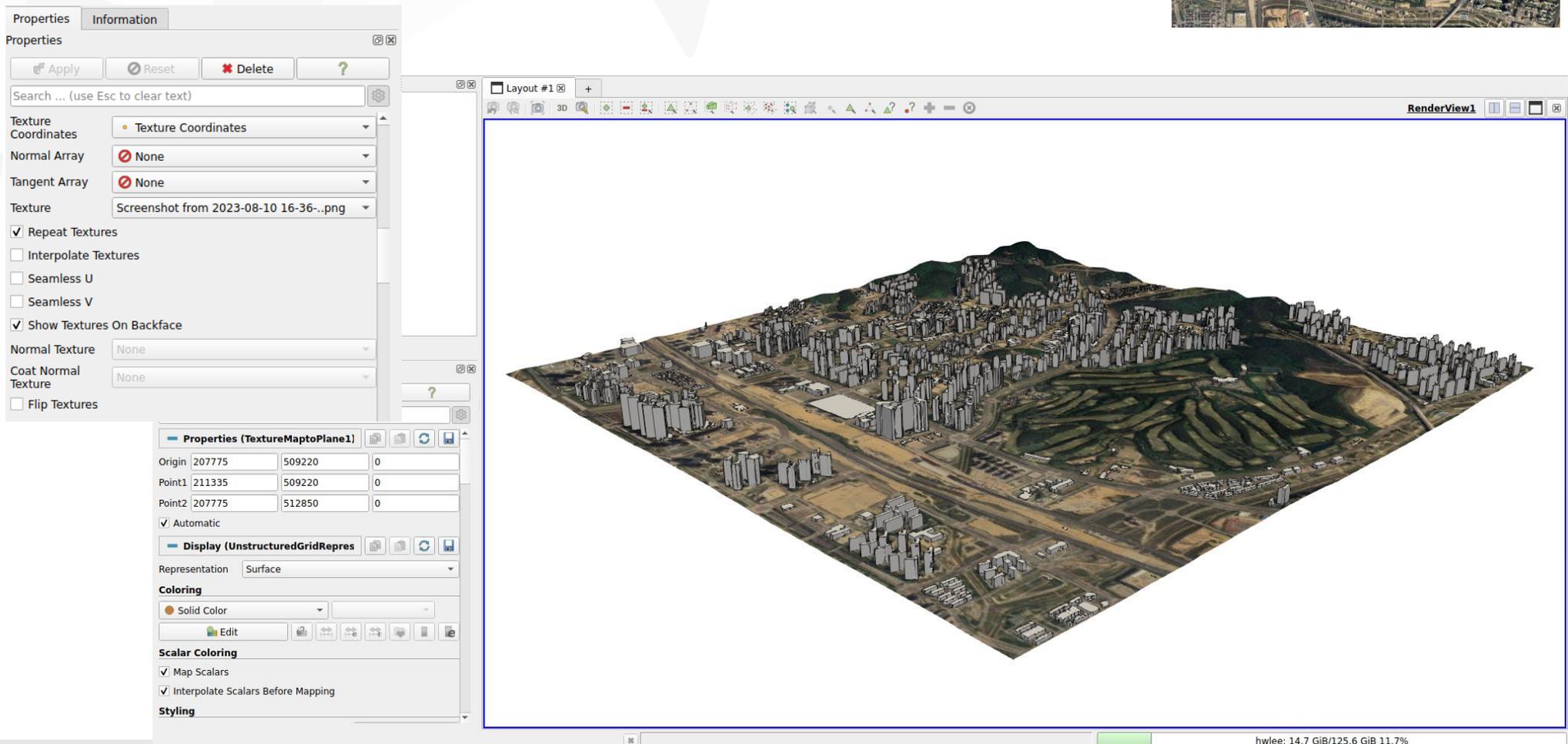
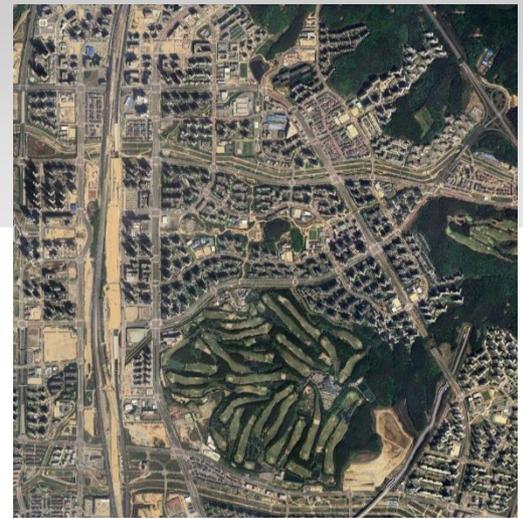
- 동탄지역의 주변 건물과 지형으로 인한 빌딩풍을 모사한 표고 2m 바람의 속도장이다.
- 지형과 건물을 전부 따와 모사했지만 그에 비해 지형이 와닿지 않는다.



1. ParaView 렌더링

2. 텍스처

- TextureMaptoPlane() 필터를 이용해 면에 텍스처를 입힐 수 있다.
- 구글 맵에서 따온 이미지를 우리가 생성한 3D 지형에 정사영하여 입힌다.

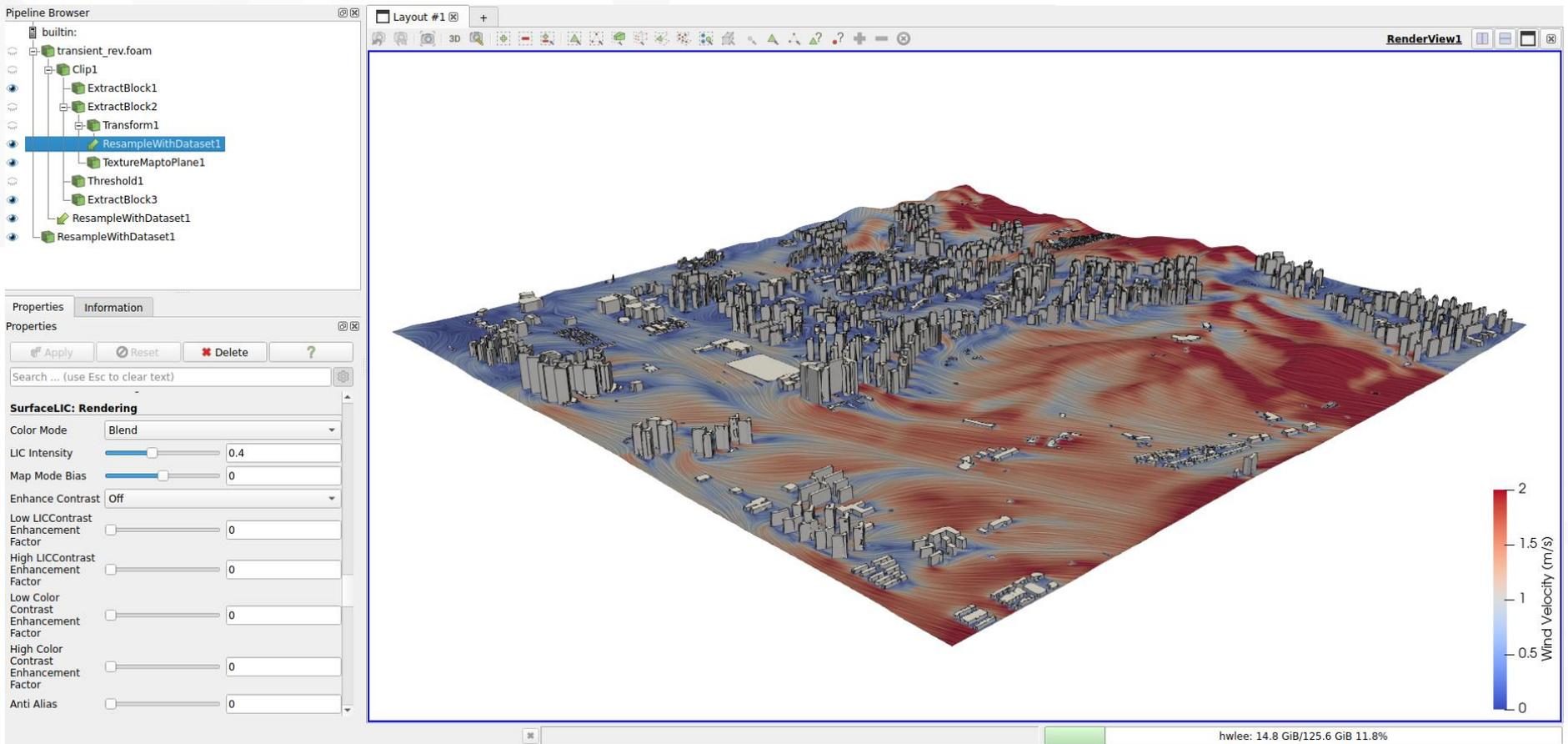
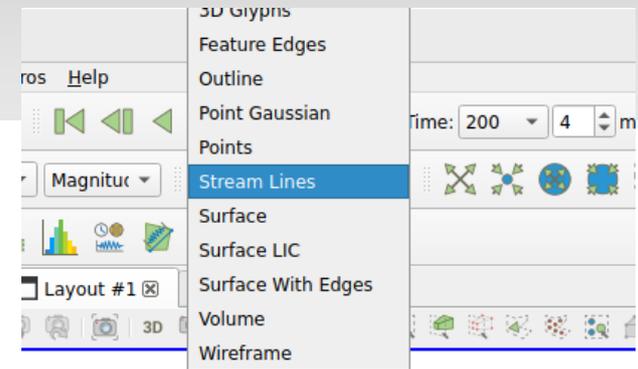


The screenshot displays the ParaView software interface. On the left, the Properties panel is open, showing the settings for the 'TextureMaptoPlane1' filter. The 'Texture' is set to 'Screenshot from 2023-08-10 16-36-...png'. The 'Repeat Textures' checkbox is checked. The 'Display (UnstructuredGridRepres)' filter is also visible, with 'Representation' set to 'Surface' and 'Coloring' set to 'Solid Color'. The 'Map Scalars' and 'Interpolate Scalars Before Mapping' checkboxes are checked. The main window shows a 3D perspective view of a city model with a texture applied, showing buildings and a golf course. The status bar at the bottom right indicates 'hwlee: 14.7 GiB/125.6 GiB 11.7%'.

1. ParaView 렌더링

3. Representation

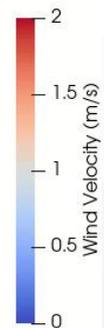
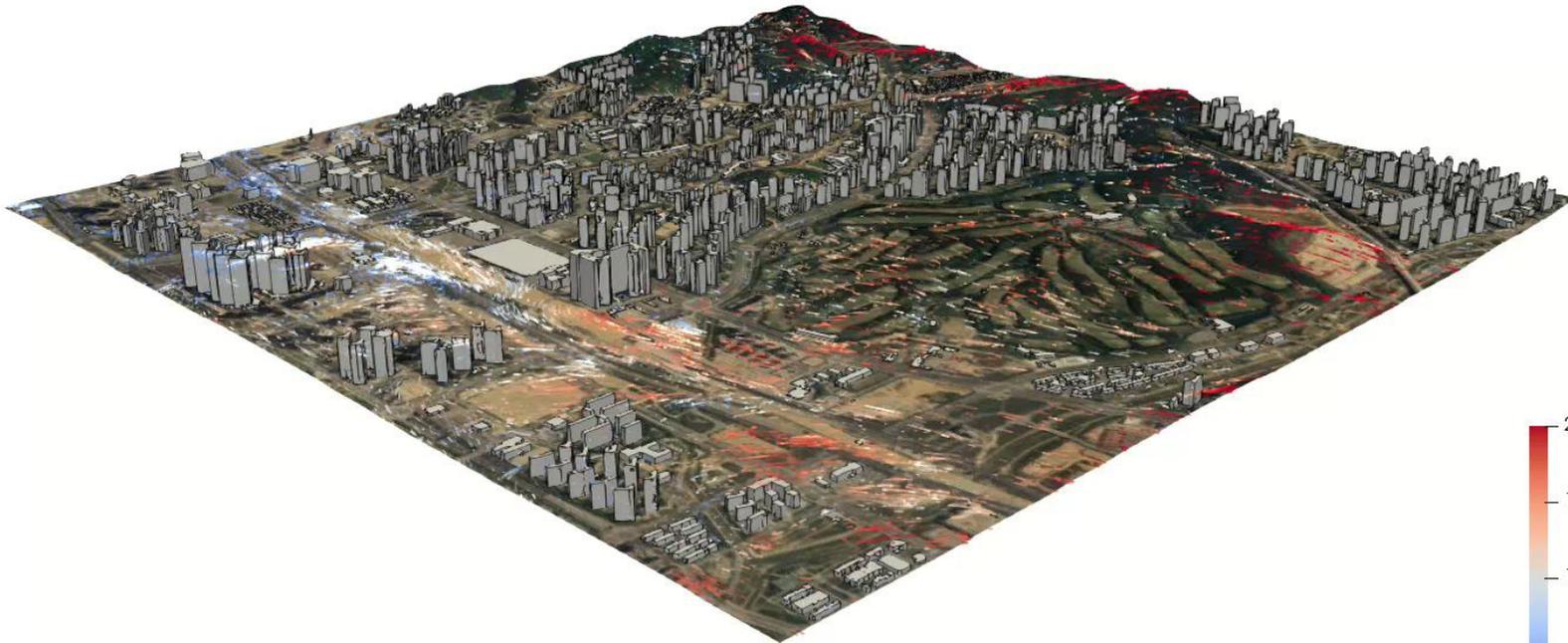
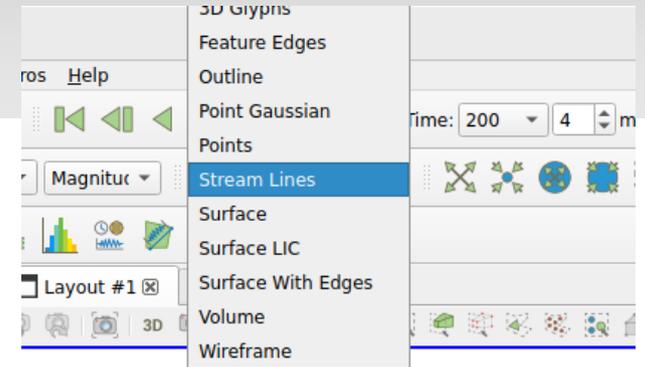
- Surface LIC Rendering (plugin)
- 표면에 라인을 추가해 유선을 모사하여 볼 수 있게 해줌



1. ParaView 렌더링

3. Representation

- Stream Line Rendering (plugin)
- 움직이는 스트림 라인을 추가해줌

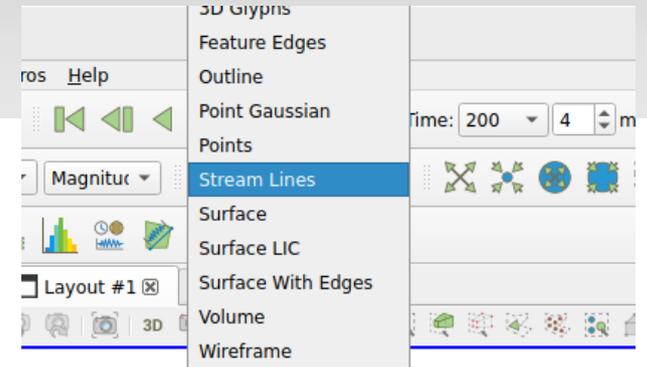


1. ParaView 렌더링

3. Representation

- Volume Rendering
- 가스나 특정 소스 등이 어디에 얼마나 뭉쳐 있는지 가시화 해줌

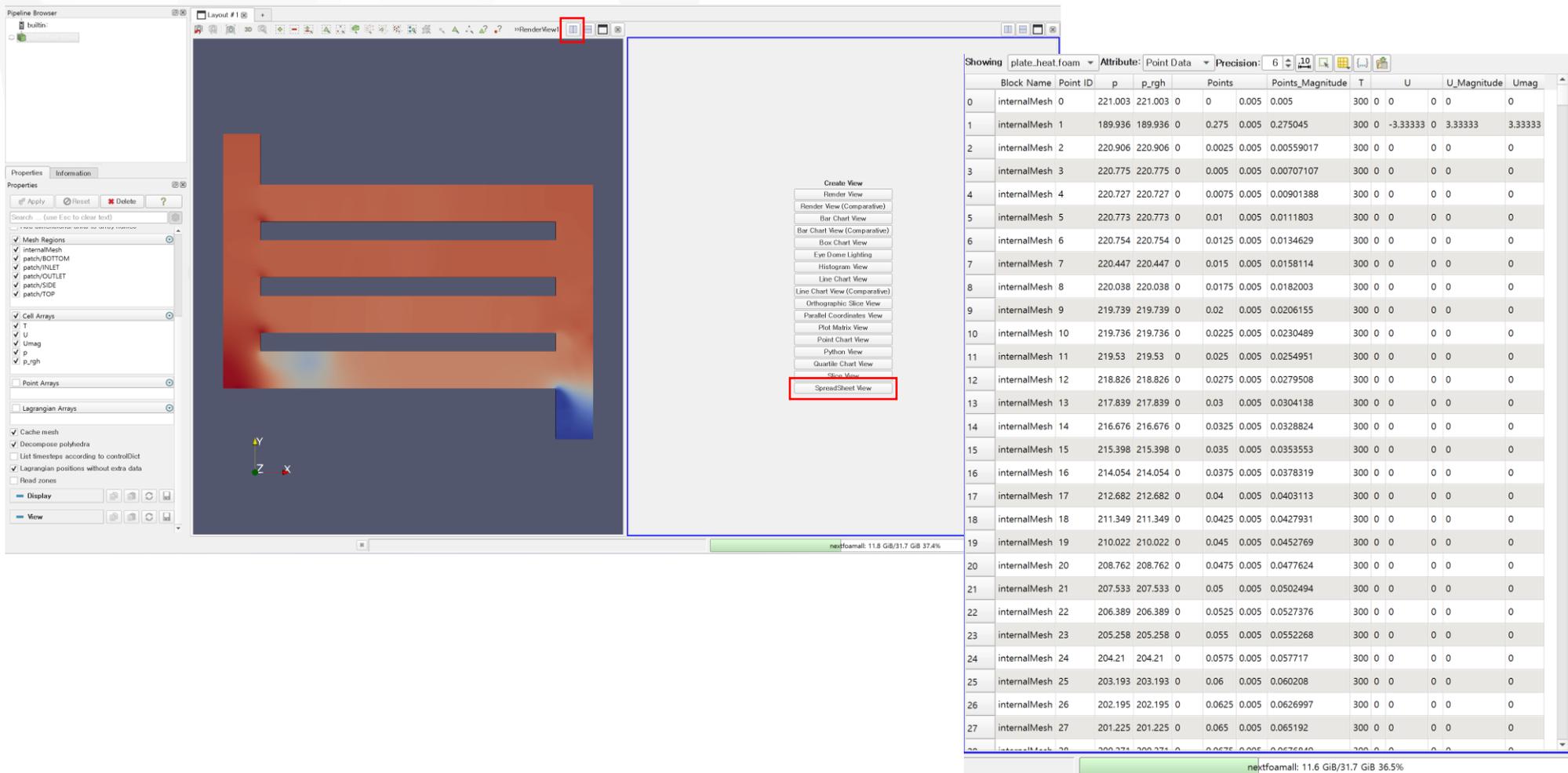
Time: 0 s



2. ParaView 데이터 처리

1. Spread sheet

- ParaView의 View 중 Spreadsheet View를 통해 데이터를 직접적으로 확인할 수 있다.



The screenshot displays the ParaView software interface. On the left, the Pipeline Browser and Properties panels are visible. The main window shows a 3D visualization of a mesh with a color gradient. A 'Create View' menu is open, with 'SpreadSheet View' highlighted. On the right, the Spreadsheet View is active, showing a table of data for the 'plate_heat_foam' case. The table columns include Block Name, Point ID, p, p_rgh, Points, Points_Magnitude, T, U, U_Magnitude, and Umag.

Block Name	Point ID	p	p_rgh	Points	Points_Magnitude	T	U	U_Magnitude	Umag
internalMesh 0	0	221.003	221.003	0	0.005	0.005	300 0 0	0 0 0	0
internalMesh 1	1	189.936	189.936	0	0.275	0.005	0.00559017	300 0 -3.33333 0	3.33333 3.33333
internalMesh 2	2	220.906	220.906	0	0.0025	0.005	0.00707107	300 0 0	0 0 0
internalMesh 3	3	220.775	220.775	0	0.005	0.005	0.00707107	300 0 0	0 0 0
internalMesh 4	4	220.727	220.727	0	0.0075	0.005	0.00901388	300 0 0	0 0 0
internalMesh 5	5	220.773	220.773	0	0.01	0.005	0.0111803	300 0 0	0 0 0
internalMesh 6	6	220.754	220.754	0	0.0125	0.005	0.0134629	300 0 0	0 0 0
internalMesh 7	7	220.447	220.447	0	0.015	0.005	0.0158114	300 0 0	0 0 0
internalMesh 8	8	220.038	220.038	0	0.0175	0.005	0.0182003	300 0 0	0 0 0
internalMesh 9	9	219.739	219.739	0	0.02	0.005	0.0206155	300 0 0	0 0 0
internalMesh 10	10	219.736	219.736	0	0.0225	0.005	0.0230489	300 0 0	0 0 0
internalMesh 11	11	219.53	219.53	0	0.025	0.005	0.0254951	300 0 0	0 0 0
internalMesh 12	12	218.826	218.826	0	0.0275	0.005	0.0279508	300 0 0	0 0 0
internalMesh 13	13	217.839	217.839	0	0.03	0.005	0.0304138	300 0 0	0 0 0
internalMesh 14	14	216.676	216.676	0	0.0325	0.005	0.0328824	300 0 0	0 0 0
internalMesh 15	15	215.398	215.398	0	0.035	0.005	0.0353553	300 0 0	0 0 0
internalMesh 16	16	214.054	214.054	0	0.0375	0.005	0.0378319	300 0 0	0 0 0
internalMesh 17	17	212.682	212.682	0	0.04	0.005	0.0403113	300 0 0	0 0 0
internalMesh 18	18	211.349	211.349	0	0.0425	0.005	0.0427931	300 0 0	0 0 0
internalMesh 19	19	210.022	210.022	0	0.045	0.005	0.0452769	300 0 0	0 0 0
internalMesh 20	20	208.762	208.762	0	0.0475	0.005	0.0477624	300 0 0	0 0 0
internalMesh 21	21	207.533	207.533	0	0.05	0.005	0.0502494	300 0 0	0 0 0
internalMesh 22	22	206.389	206.389	0	0.0525	0.005	0.0527376	300 0 0	0 0 0
internalMesh 23	23	205.258	205.258	0	0.055	0.005	0.0552268	300 0 0	0 0 0
internalMesh 24	24	204.21	204.21	0	0.0575	0.005	0.057717	300 0 0	0 0 0
internalMesh 25	25	203.193	203.193	0	0.06	0.005	0.060208	300 0 0	0 0 0
internalMesh 26	26	202.195	202.195	0	0.0625	0.005	0.0626997	300 0 0	0 0 0
internalMesh 27	27	201.225	201.225	0	0.065	0.005	0.065192	300 0 0	0 0 0

2. ParaView 데이터 처리

2. Find data

- ParaView 5.11 부터 추가된 기능
- 상단 툴바에서 Find Data 탭을 활성화 시키면 내가 원하는 셀의 데이터를 확인할 수 있다.

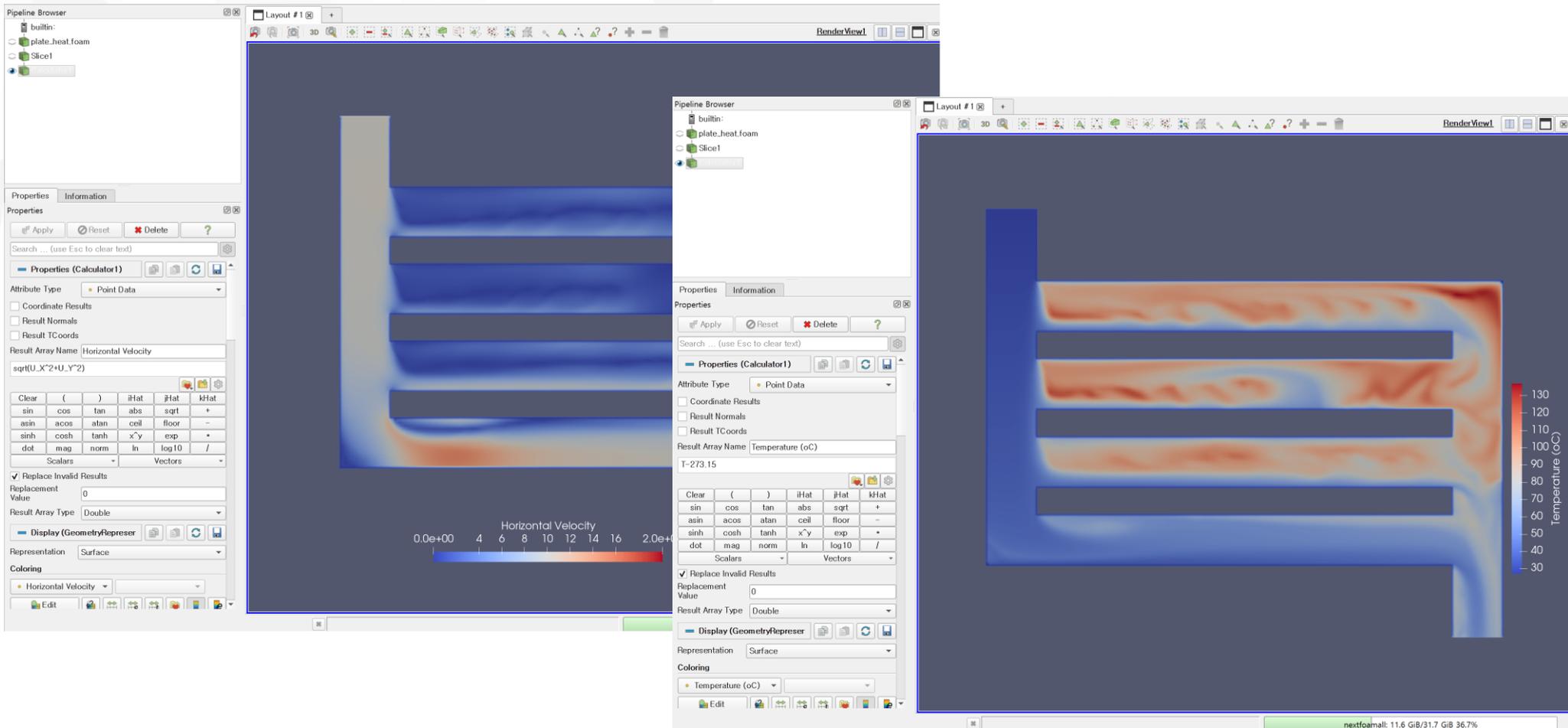
The screenshot displays the ParaView 5.11 interface. The top toolbar has a red box around the Find Data icon. The 'Find Data' panel on the right shows a table of selected data points. The table has the following columns: Block Name, Point ID, p, p_rgh, Points, Points_Magnitude, and T. The data is as follows:

Block Name	Point ID	p	p_rgh	Points	Points_Magnitude	T		
0 internalMesh	534	180.696	180.696	0.1125	0.22	0.005	0.247146	300
1 internalMesh	535	180.686	180.686	0.11	0.22	0.005	0.246018	300
2 internalMesh	1655	180.693	180.693	0.111257	0.217843	0.005	0.24466	300
3 internalMesh	2350	180.703	180.703	0.113745	0.217812	0.005	0.245774	300
4 internalMesh	3066	180.69	180.69	0.10997	0.215571	0.005	0.242052	300
5 internalMesh	4052	180.705	180.705	0.112491	0.215557	0.005	0.243195	300
6 internalMesh	13836	180.696	180.696	0.1125	0.22	0.0025	0.247108	300
7 internalMesh	13837	180.685	180.685	0.11	0.22	0.0025	0.24598	300
8 BOTTOM	773	180.696	180.696	0.1125	0.22	0	0.247096	500
9 BOTTOM	775	180.685	180.685	0.11	0.22	0	0.245967	500

2. ParaView 데이터 처리

3. Calculator

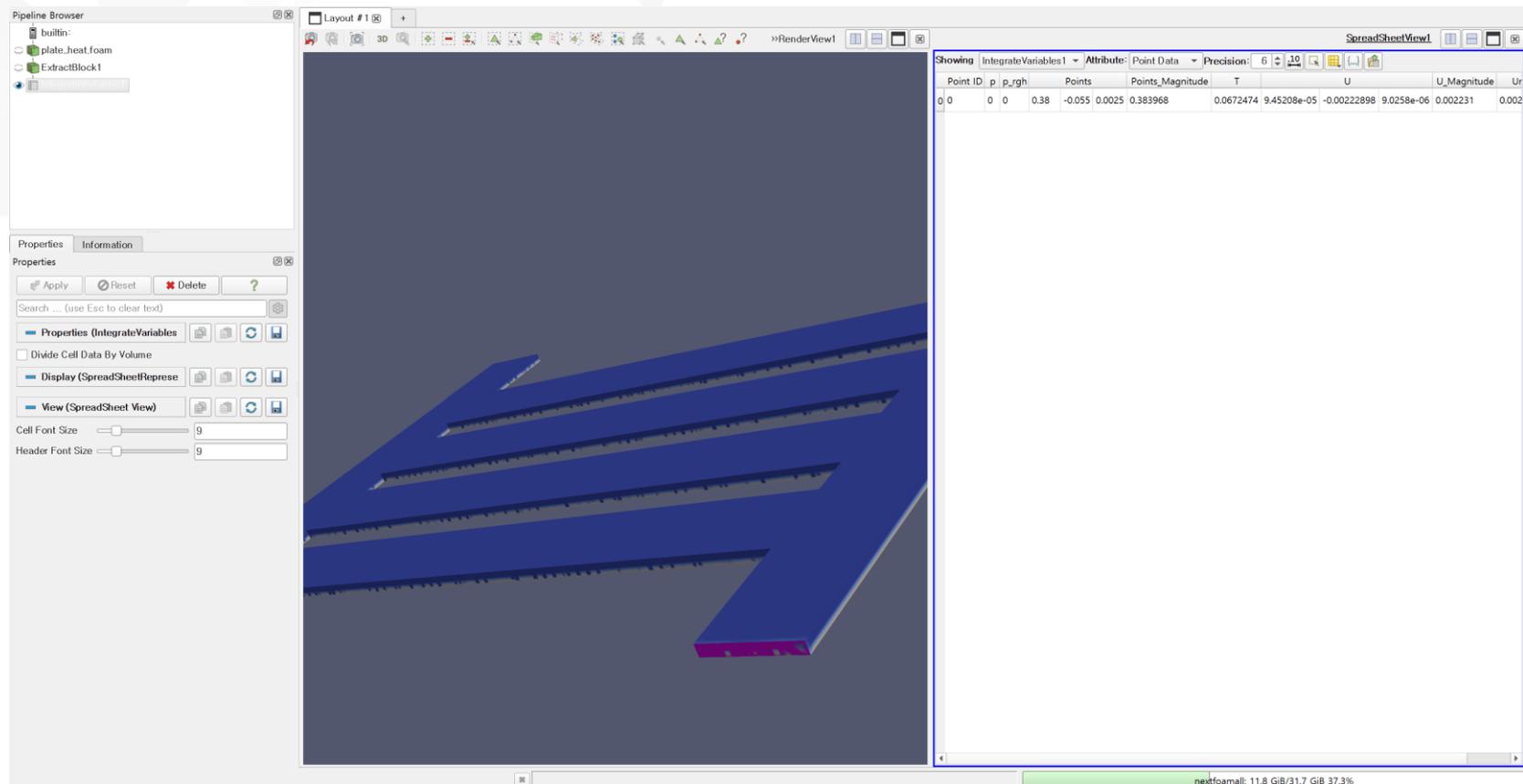
- Cell Value를 가공하여 새로운 데이터로 만들 수 있는 필터
- 절대 온도에 273.15를 감산해 섭씨로 만들거나 벡터의 원하는 성분만 뽑아서 확인할 수 있다.



2. ParaView 데이터 처리

4. integrateVariable

- 해당 Field의 값을 Area나 Volume으로 적분하여 하나의 값으로 보여주는 필터
- 원하는 patch (outlet, inlet 등) 에서 부피 유량을 찾아낼 수 있다.
- 따라서 해당 케이스에서 유량은 $0.002231 \text{ m}^3/\text{s}$



2. ParaView 데이터 처리

5. Pvpython과 Fetch()

- ParaView의 여러 기능들은 pvpython 모듈을 이용해 python을 통해 제어할 수 있다.
- Pvpython 내에서 정보들은 객체화된 형태로 저장된다. 따라서 얻은 정보에 바로 접근할 수 없다.
- Fetch() 함수를 이용해 vtk 정보를 python에서 제어할 수 있는 int나 double 형태로 추출할 수 있다.

```
outletVariable = IntegrateVariables(Input=outlet_block)
outlet_data = paraview.servermanager.Fetch(outletVariable)
outlet_pressure_integ = outlet_data.GetCellData().GetArray('p').GetValue(0)
outlet_area = outlet_data.GetCellData().GetArray('Area').GetValue(0)
outlet_pressure = outlet_pressure_integ / outlet_area
```

목표 : Outlet의 평균 압력 추출

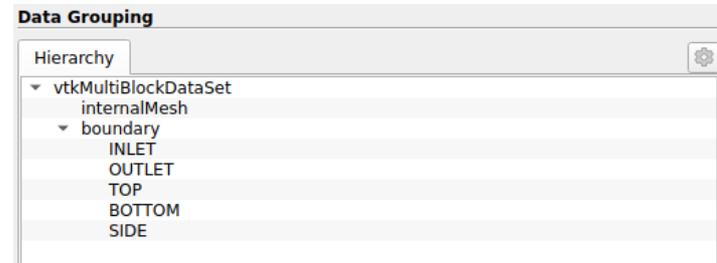
1. IntegrateVariables 필터를 이용해 outlet patch의 면적 적분 값을 생성한다.
2. Servermanager.Fetch() 함수를 이용해 '적분된 data'를 사용할 수 있는 '변수'로 변환한다.
3. GetCellData().GetArray('p').GetValue(0) 함수를 사용해 대상 객체의 행렬 중 첫번째 셀의 'p' 스칼라 값을 반환한다.
4. GetCellData().GetArray('Area').GetValue(0) 함수를 사용해 대상 객체의 행렬 중 첫번째 셀의 넓이 값을 반환한다.
* (IntegrateVariables 필터를 사용했기 때문에 첫번째 셀에 적분 값이 저장된다.)
5. 적분된 압력을 넓이로 나누어 평균 압력을 저장한다.

3. 보고서 자동화

1. 객체 제어

- 자동화를 위해 임의의 입력 파일에 대한 데이터 구조를 스크립트 스스로 판단할 수 있는 알고리즘 필요
- 대상의 기하 구조 판단을 위해 케이스의 Hierachy 정보를 추출
→ GetHeirachy() 함수 사용해 Multi-block 내 각 block의 이름을 저장

```
# case data를 불러옴.  
foam_case_data = import_foam_case(str(self.input_dict['filename']))  
  
node_path_list = []  
hierarchy = foam_case_data.GetDataInformation().DataInformation.GetHierarchy()  
node_path_list.append(hierarchy.GetNodePath(0))  
for i in hierarchy.GetChildNodes(0):  
    node_path_list.append(hierarchy.GetNodePath(i))
```



- 조건문을 사용해 inlet, outlet 과 같은 특정 patch의 주소를 딕셔너리 형태로 저장 가능
- ExtractBlock 함수를 사용해 block의 데이터를 사용하기 편하게 분리하여 객체 형태로 저장

```
def find_index(self, node_path_list):  
    index_dict = {"internal": [], "inlet": [], "outlet": [], "Patches": []}  
    for node_path in node_path_list:  
        if 'internalMesh' in node_path:  
            index_dict["internal"] = [node_path]  
        elif 'boundary' == node_path:  
            if 'inlet' in node_path.lower():  
                index_dict["inlet"] = [node_path]  
            elif 'outlet' in node_path.lower():  
                index_dict["outlet"] = [node_path]  
  
    return index_dict
```

```
def extractBlock(foam_case_data, node_path):  
  
    extractblock1 = ExtractBlock(Input=foam_case_data)  
    extractblock1.Selectors = node_path  
    extractblock1.UpdatePipeline()  
  
    return extractblock1
```

3. 보고서 자동화

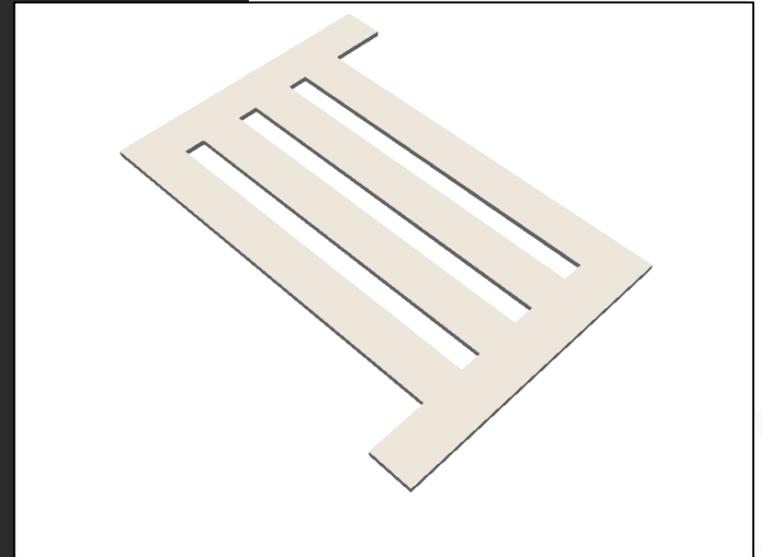
2. 렌더 뷰 컨트롤

- 자동으로 카메라의 시점을 조절하기 위해서는 대상 객체의 기하정보 필요
→ GetDataInformation().GetBounds() 함수 사용

```
>>> geo_minmax_list = list(foam_case_data.GetDataInformation().GetBounds())  
>>> geo_minmax_list  
[0.0, 0.4000000059604645, -0.054999999701976776, 0.2750000059604645, 0.0, 0.004999999888241291]
```

- 기하 정보를 이용하여 임의의 객체에 대한 isometric 시점의 RenderView를 생성가능

```
def create_isometric_view(geo_minmax_list, type='forward', name='Isometric view'):  
  
    x_length = geo_minmax_list[1] - geo_minmax_list[0]  
    y_length = geo_minmax_list[3] - geo_minmax_list[2]  
    z_length = geo_minmax_list[5] - geo_minmax_list[4]  
  
    ref_length = (x_length ** 2 + y_length ** 2 + z_length ** 2) ** 0.5  
  
    center_x = (geo_minmax_list[0] + geo_minmax_list[1]) / 2  
    center_y = (geo_minmax_list[2] + geo_minmax_list[3]) / 2  
    center_z = (geo_minmax_list[4] + geo_minmax_list[5]) / 2  
  
    # Create a new 'Render View'  
    isometricview1 = CreateView('RenderView')  
    isometricview1.ViewSize = [800, 600]  
    isometricview1.CenterOfRotation = [center_x, center_y, center_z]  
    isometricview1.StereoType = 'Crystal Eyes'  
    isometricview1.CameraPosition = [center_x - 0.8 * ref_length,  
                                     center_y + 0.8 * ref_length, center_z + 1.2 * ref_length]  
    isometricview1.CameraViewUp = [0.5, -0.5, 0.5]  
    isometricview1.CameraFocalPoint = [center_x - 0.15 * x_length, center_y + y_length * 0.15, center_z]  
    isometricview1.Background = [1.0, 1.0, 1.0]
```



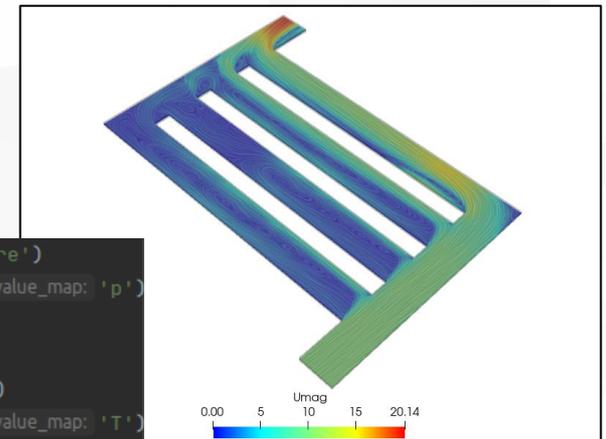
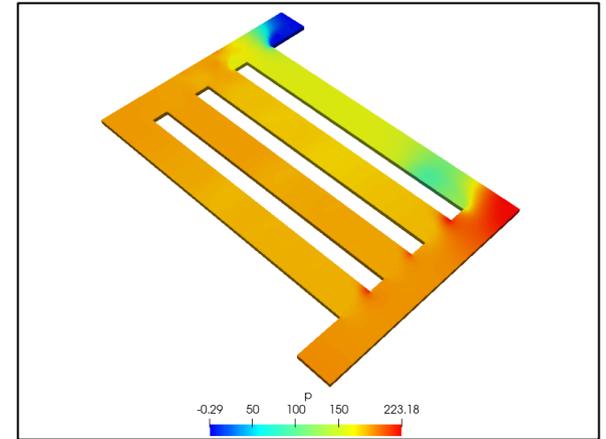
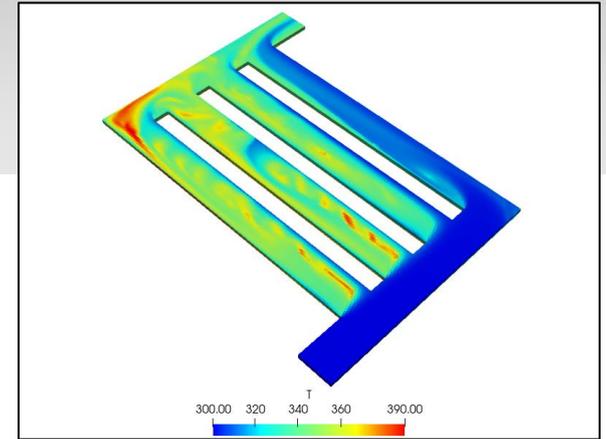
3. 보고서 자동화

3. 벨류 렌더링

- 원하는 스칼라와 객체를 렌더뷰에 투영
- GetArray()와 GetRange() 함수를 이용해 최대 최소 값과 RGB 값 조절
- Python의 적절한 활용으로 여러 이미지를 자동으로 출력

```
def solid_view(contour_object, renderview1, value_map='nan'):  
    # show data in view  
    solid_display = Show(contour_object, renderview1, 'GeometryRepresentation')  
    solid_display.Representation = 'Surface'  
  
    ColorBy(solid_display, ('CELLS', value_map, 'Magnitude'))  
  
    LUT = GetColorTransferFunction(value_map)  
    scalar_map = contour_object.PointData.GetArray(value_map).GetRange()  
    uLUT_min = scalar_map[0]  
    uLUT_max = scalar_map[1]  
  
    LUT.ApplyPreset('Jet', True)  
    LUT.RGBPoints = [uLUT_min, 0.0, 0.0, 1.0,  
                    uLUT_min + (uLUT_max - uLUT_min) * 0.25, 0.0, 1.0, 1.0,  
                    uLUT_min + (uLUT_max - uLUT_min) * 0.5, 0.5, 1.0, 0.5,  
                    uLUT_min + (uLUT_max - uLUT_min) * 0.75, 1.0, 1.0, 0.0,  
                    uLUT_max, 1.0, 0.0, 0.0]  
    LUT.NanColor = [1.0, 0.0, 0.0]  
    LUT.ScalarRangeInitialized = 1.0
```

```
isometricview1, layout1 = create_isometric_view(self.geo_minmax_dict['internal'], name='Total Pressure')  
[pressure_display, total_pressure_view1] = solid_view(self.block_dict['internal'], isometricview1, value_map: 'p')  
SaveScreenshot(str(self.png_dir / 'isometric_total_pressure.png'), view=total_pressure_view1)  
  
isometricview2, layout2 = create_isometric_view(self.geo_minmax_dict['internal'], name='Temperature')  
[temperature_display, temperature_view1] = solid_view(self.block_dict['internal'], isometricview2, value_map: 'T')  
SaveScreenshot(str(self.png_dir / 'isometric_temperature.png'), view=temperature_view1)
```



3. 보고서 자동화

4. py-pptx

- Python 패키지 py-pptx를 이용해 pptx 파일을 자동 생성
- 슬라이드를 생성하고 텍스트 박스, 그림, 표를 만들수 있음

```
# ppt 객체 생성
presentation = pptx.Presentation()
# 새 slide 생성
slide_layout = presentation.slide_layouts[6]
slide1 = presentation.slides.add_slide(slide_layout)

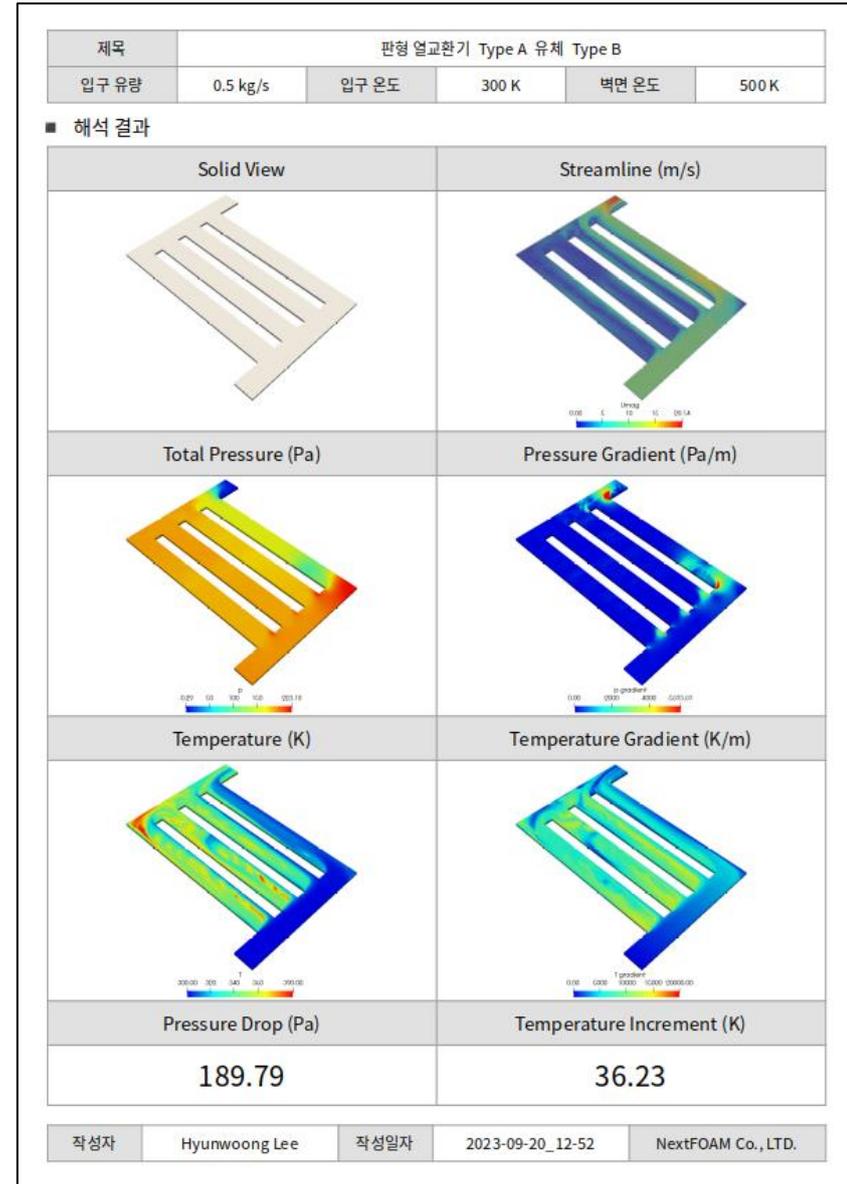
# slide 제목 생성
textBox = slide1.shapes.add_textbox(Cm(0.5), Cm(0.5), Cm(8), Cm(3))
tf = textBox.text_frame
p = tf.paragraphs[0]
p.text = "* Pressure Drop"
p.alignment = PP_ALIGN.LEFT
p.font.size = Pt(12)
p.font.name = 'Times New Roman'

def make_table_slide(slide, cols, rows, data, title_left, title_top):
    # 셀 크기 설정
    first_col_width = Cm(2)
    col_width = Cm(3.5)
    row_height = Cm(1)

    # table 위치 및 크기 설정
    table_width = cols * col_width + first_col_width
    table_height = rows * row_height
    left = title_left
    top = title_top + p.font.size + row_height / 2

    # 행렬 생성
    shape = slide.shapes.add_table(rows, cols, left, top, table_width, table_height)
    table = shape.table
```

```
#이미지 불러넣기
solid_image = os.path.join(address, 'solid_isometricview.png')
slide1.shapes.add_picture(solid_image, Cm(0), Cm(2), Cm(10), Cm(7.5))
```



요약 및 결론

1. 오픈 소스 소프트웨어인 ParaView의 다양한 **렌더링 방식**에 대해 알아보았다.
2. ParaView로 읽은 case **데이터를 활용하는 방식**에 대해 알아보았다.
3. 미리 정해놓은 **렌더링 뷰**를 사용해 **이미지나 영상을 자동으로 출력**할 수 있었다.
4. python-pptx 라이브러리를 연동해 **자동으로 보고서**를 출력했다.

다수의 케이스를 이용해 다량의 보고서를 만들어야 할 때, 한 번 스크립트를 만들어 두면 효과적으로 보고서를 자동화 할 수 있다!

중요한 장면들은 여러 렌더링 기법을 통해 강조하면 좋다.

감사합니다.