Trame - the basics

Sebastien Jourdain

kitware

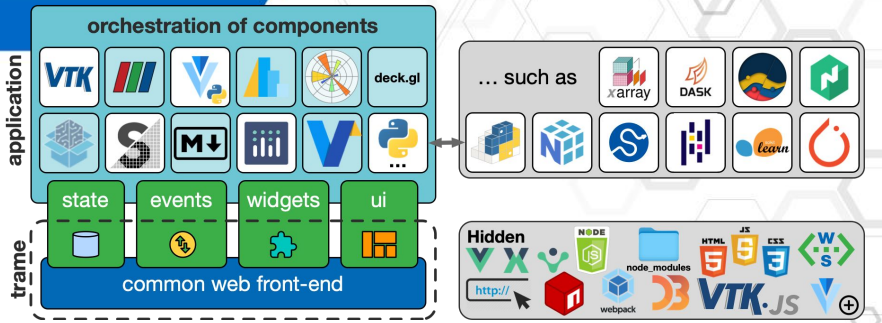
Outline (~2 hours)

- **Introduction**
  - What is trame?
  - How trame works?
  - Why is it revolutionary?
- **Getting started**
  - Documentation
  - Tutorial & examples
  - Cookie cutter
- **Trame and vtk(.js)**
  - Usage exploration
- **Cheat sheets**
  - Syntax explained
- **Cookiecutter**
  - Standardisation for python
- **ParaView gotcha**
  - pvpython is not your python
- **Deployment**
  - Desktop, jupyter and cloud

Introduction

15 minutes

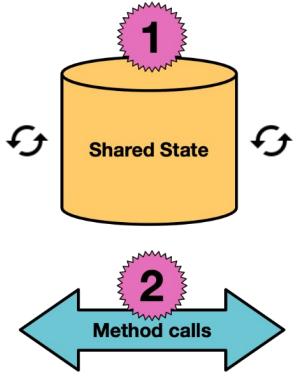
What is trame?

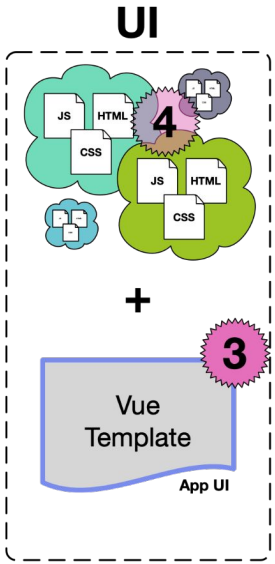


- **Simple**
  All the logic and UI definition can be done in plain Python
- **Powerful**
  Python offer scientific and information data visualization with capable data processing (numpy, Plotly, Matplotlib, VTK, ParaView…)
- **Ubiquitous**
  Runs on laptops, desktops, clusters, and the cloud while displaying everywhere (phone, tablet, laptop, workstation)

How trame works?

Server side





0 - Just a Python file
1 - Simple data exchange
2 - Simple code binding
3 - Efficient UI definition
4 - Add-on widgets

Client side

kitware

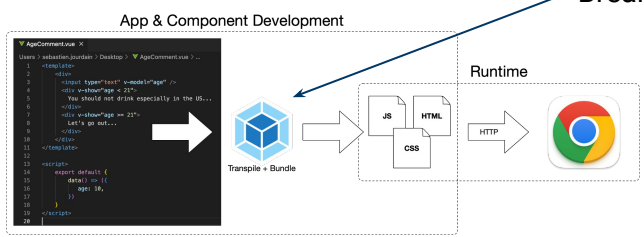
Why is it revolutionary?

New way

Just a Python application

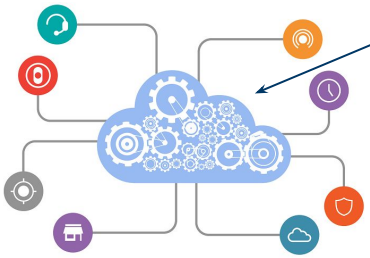
No need to fix build system every 6 months.
Only your python code to maintain and deploy.
Uniform and simple way for coding and binding UI

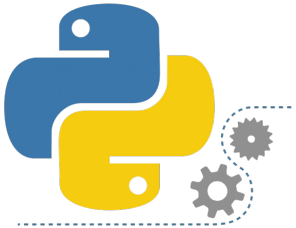
Old way



Break every 6 months

Maintenance cost





Complex client/server state management and binding

Complex process

kitware

Getting started

5 minutes

Getting started

- **Documentation**
  - https://kitware.github.io/trame/
  - https://trame.readthedocs.io/
- **Tutorial & Examples**
  - https://github.com/Kitware/trame-tutorial
  - https://github.com/Kitware/trame => ./examples
- **Cookiecutter**
  - https://github.com/Kitware/trame-cookiecutter

Trame and VTK

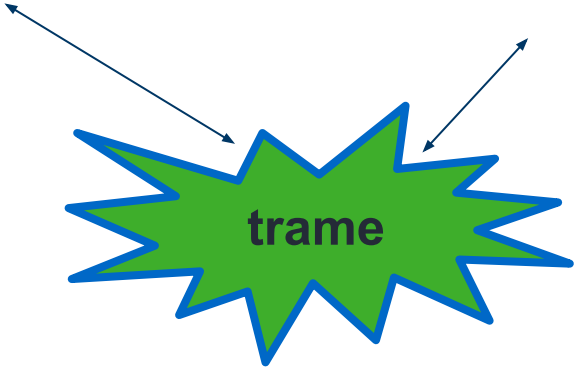
50 minutes

VTK (C++, Python)

- Server only (python)
- I/O
- Data processing
- Rendering

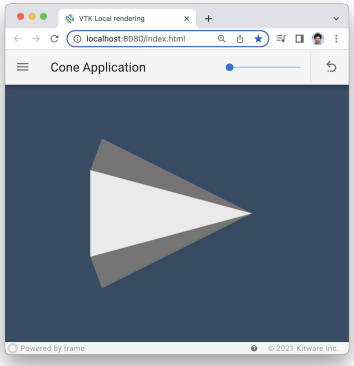
Vtk.js

- Client only (browser)
- Rendering
- Limited processing
- Limited I/O



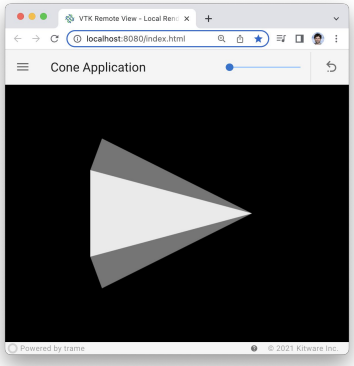
How many ways can you make the same application?

No VTK, just vtk.js



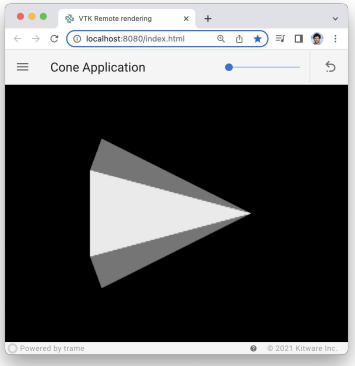
~ClientView.py

VTK for processing and vtk.js for rendering



LocalRendering.py

VTK for processing and rendering



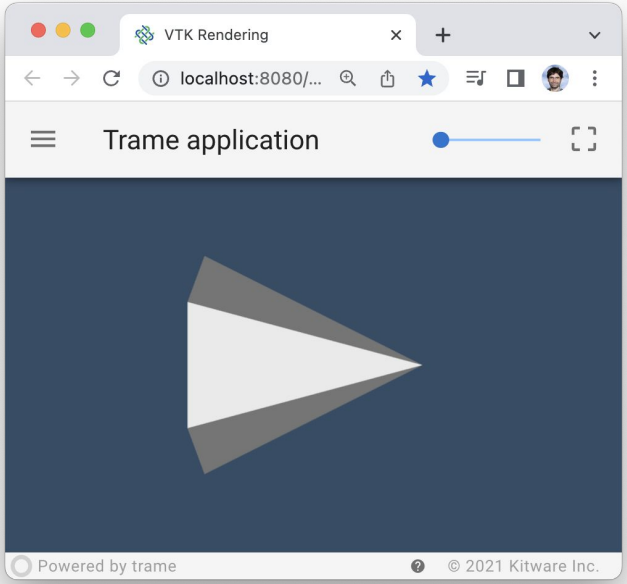
RemoteRendering.py

kitware

Let’s get started on your machine

```
mkdir trame-course && cd $_
python3.9 -m venv .venv
source ./.venv/bin/activate
python -m pip install --upgrade pip
pip install "trame"

# Just to get the examples
git clone https://github.com/Kitware/trame.git
```

VTK without VTK



client.py — LICENSE UPGRADE REQUIRED

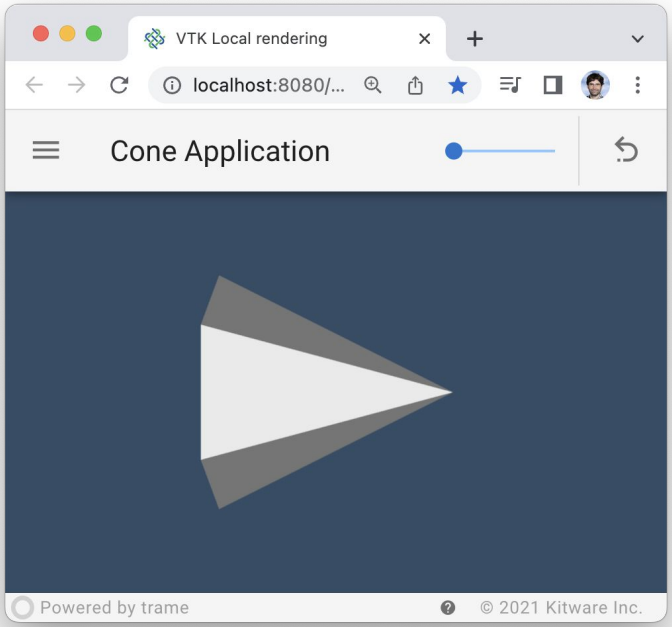
```python
from trame.app import get_server
from trame.ui.vuetify import SinglePageLayout
from trame.widgets import vuetify, vtk

server = get_server()
state, ctrl = server.state, server.controller
state.trame__title = "VTK Rendering"

with SinglePageLayout(server) as layout:
    with layout.content:
        with vuetify.VContainer(fluid=True, classes="pa-0 fill-height"):
            with vtk.VtkView(ref="view"):
                with vtk.VtkGeometryRepresentation():
                    vtk.VtkAlgorithm(
                        vtkClass="vtkConeSource", state=("{ resolution }",)
                    )

    with layout.toolbar:
        vuetify.VSpacer()
        vuetify.VSlider(
            hide_details=True,
            v_model=("resolution", 6),
            min=3, max=60, step=1,
            style="max-width: 300px;",
        )
        with vuetify.VBtn(icon=True, click="$refs.view.resetCamera()"):
            vuetify.VIcon("mdi-crop-free")


if __name__ == "__main__":
    server.start()
```

vtk_class=

Line 32, Column 1 — Spaces: 4 — Python

```
python ./trame/examples/06_vtk/00_ClientOnly/client-side-cone.py
```

VTK mesh rendering



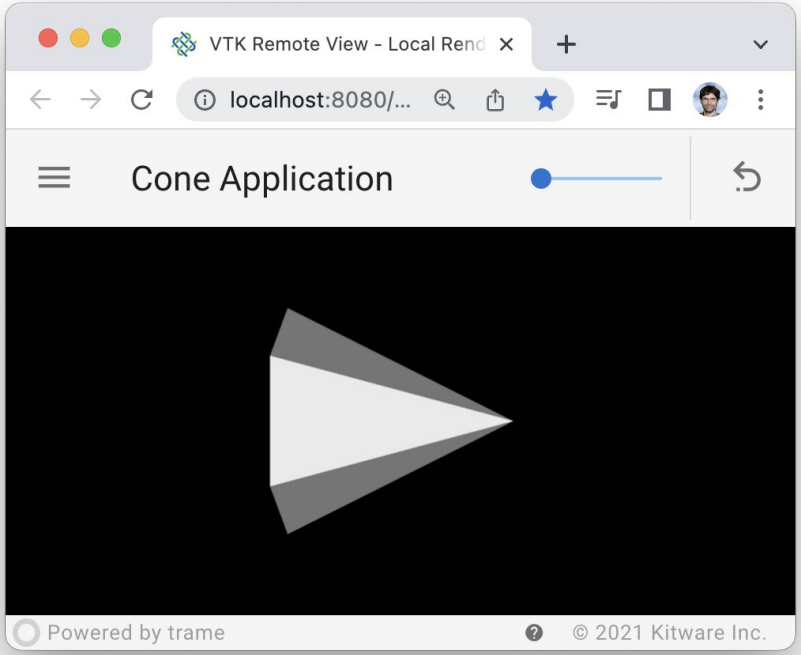
client-servermesh.py — LICENSE UPGRADE REQUIRED

```python
from trame.app import get_server
from trame.widgets import vuetify, vtk
from trame.ui.vuetify import SinglePageLayout

from vtkmodules.vtkFiltersSources import vtkConeSource

server = get_server()
state, ctrl = server.state, server.controller
state.trame__title = "VTK Local rendering"

DEFAULT_RESOLUTION = 6
cone_generator = vtkConeSource()

@state.change("resolution")
def update_cone(resolution=DEFAULT_RESOLUTION, **kwargs):
    cone_generator.SetResolution(resolution)
    ctrl.mesh_update()

def update_reset_resolution():
    state.resolution = DEFAULT_RESOLUTION

with SinglePageLayout(server) as layout:
    layout.icon.click = ctrl.view_reset_camera
    layout.title.set_text("Cone Application")

    with layout.toolbar:
        vuetify.VSpacer()
        vuetify.VSlider(
            v_model=("resolution", DEFAULT_RESOLUTION),
            min=3, max=60, step=1,
            hide_details=True, dense=True,
            style="max-width: 300px",
        )
        vuetify.VDivider(vertical=True, classes="mx-2")

        with vuetify.VBtn(icon=True, click=update_reset_resolution):
            vuetify.VIcon("mdi-undo-variant")

    with layout.content:
        with vuetify.VContainer(fluid=True, classes="pa-0 fill-height"):
            with vtk.VtkView() as view:
                ctrl.view_reset_camera = view.reset_camera
                with vtk.VtkGeometryRepresentation():
                    html_polydata = vtk.VtkPolyData("cone", dataset=cone_generator)
                    ctrl.mesh_update = html_polydata.update

if __name__ == "__main__":
    server.start()
```

Line 49, Column 1 — Spaces: 4 — Python

```
pip install vtk
python ./trame/examples/06_vtk/01_SimpleCone/ClientView.py
```

VTK rendering without rendering



vtk-localrendering.py

```python
from vtk import *
from trame.app import get_server
from trame.widgets import vuetify, vtk
from trame.ui.vuetify import SinglePageLayout

server = get_server()
state, ctrl = server.state, server.controller
DEFAULT_RESOLUTION = 6

renderer = vtkRenderer()
renderWindow = vtkRenderWindow()
renderWindow.AddRenderer(renderer)

renderWindowInteractor = vtkRenderWindowInteractor()
renderWindowInteractor.SetRenderWindow(renderWindow)
renderWindowInteractor.GetInteractorStyle().SetCurrentStyleToTrackballCamera()

cone_source = vtkConeSource()
mapper = vtkPolyDataMapper()
actor = vtkActor()
mapper.SetInputConnection(cone_source.GetOutputPort())
actor.SetMapper(mapper)
renderer.AddActor(actor)
renderer.ResetCamera()
renderWindow.Render()

@state.change("resolution")
def update_cone(resolution=DEFAULT_RESOLUTION, **kwargs):
    cone_source.SetResolution(resolution)
    ctrl.view_update()

def update_reset_resolution():
    state.resolution = DEFAULT_RESOLUTION

with SinglePageLayout(server) as layout:
    layout.icon.click = ctrl.view_reset_camera

    with layout.toolbar:
        vuetify.VSpacer()
        vuetify.VSlider(
            v_model=("resolution", DEFAULT_RESOLUTION),
            min=3, max=60, step=1,
            hide_details=True, dense=True, style="max-width: 300px",
        )
        vuetify.VDivider(vertical=True, classes="mx-2")
        with vuetify.VBtn(icon=True, click=update_reset_resolution):
            vuetify.VIcon("mdi-undo-variant")

    with layout.content:
        with vuetify.VContainer(fluid=True, classes="pa-0 fill-height"):
            view = vtk.VtkLocalView(renderWindow, ref="view")
            ctrl.view_update = view.update
            ctrl.view_reset_camera = view.reset_camera

if __name__ == "__main__":
    server.start()
```

```
python ./trame/examples/06_vtk/01_SimpleCone/LocalRendering.py
```

VTK rendering with rendering

vtk-localrendering.py -> vtk-remoterendering.py — Desktop

LICENSE UPGRADE REQUIRED

```
1       with layout.content:
2           with vuetify.VContainer(fluid=True, classes="pa-0 fill-height"):
3   -           view = vtk.VtkLocalView(renderWindow, ref="view")
4   +           view = vtk.VtkRemoteView(renderWindow, ref="view")
5               ctrl.view_update = view.update
6               ctrl.view_reset_camera = view.reset_camera
7
```

Line 7, Column 2 29% Tab Size: 4 Diff

```
python ./trame/examples/06_vtk/01_SimpleCone/RemoteRendering.py
```

What you've seen?

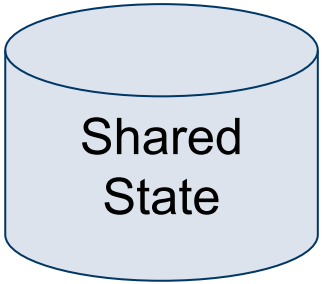
- **Layout (trame.ui.*)**
- **Widgets (trame.widgets.*)**
- **Trame server, state, controller (trame.app.get_server)**
  - State usage, initial value, monitoring changes
  - Method binding with UI and/or controller
- **VTK can be integrated with various granularity**
  - vtk.js only
  - VTK mesh and vtk.js rendering
  - VTK rendering with either local or remote rendering

Cheat sheets

10 minutes

State handling

```python
from trame.app import get_server
server = get_server()
state = server.state
```




```python
@state.change("a", "b", "c")
def change_detected(a, b, c, d, **kwargs):
    pass

state.a = 10
print(f"a={state.a}")

def update_var(name, value)
    print(f"Before: {name}={state[name]}")
    state[name] = value
    print(f"After: {name}={state[name]}")
```

JS

```
Widget(key="welcome")          : Use string 'welcome'
Widget(key=("welcome",))       : Evaluate expression 'welcome'
Widget(key=("welcome","Hello")) : Evaluate expression 'welcome' and initialize it to "Hello"
```

Method handling and controller helper

```python
from trame.app import get_server
server = get_server()
ctrl = server.controller
```

```python
# @ctrl.set("hello_alias")
def hello(*args, **kwargs):
    print(args, kwargs)

@ctrl.add("hello_alias")
def hello2(*args, **kwargs):
    print("v2:", args, kwargs)

ctrl.hello_alias = hello
ctrl.hello_alias('arg_0', key='yes')
```

JS

```
VBtn(click=fn)                                  : Call the function with no args
VBtn(click=(fn, "[1, $event, 'hello']"))        : Call the function like fn(1, event, "hello")
VBtn(click=(fn, "[$event]", "{ x: 5 }"))        : Call the function like fn(event, x=5)
```

From JavaScript to Python

```
<v-tooltip bottom>
  <template v-slot:activator="{ on, attrs }">
    <v-btn
      color="primary"
      dark
      v-bind="attrs"
      v-on="on"
    >
      Button
    </v-btn>
  </template>
  <span>Tooltip</span>
</v-tooltip>
```

@click=...

```
from trame.widgets import html, vuetify

with vuetify.VTooltip(bottom=True) as debug:
    with vuetify.Template(v_slot_activator="{on, attrs}"):
        vuetify.VBtn(
            "Button",
            color="primary",
            dark=True,
            v_bind="attrs",
            v_on="on",
        )
    html.Span("Tooltip")

    print(debug.html) # To validate that the template match
```

click=...

Any invalid Python character become “_” for any given attribute key.

https://vuetifyjs.com/en/components/tooltips/

Cookiecutter

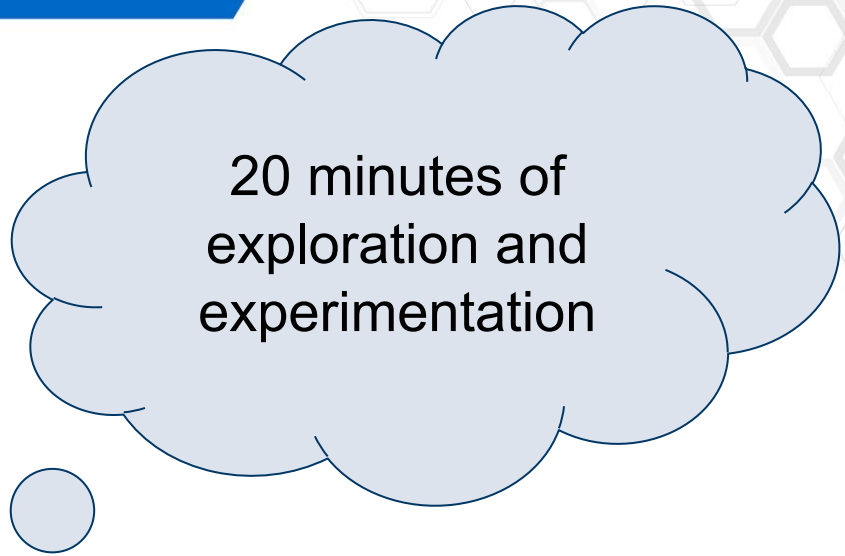
25 minutes

Making your own application (Cookiecutter)

- **Start with the cookiecutter**

```
pip install cookiecutter
cookiecutter gh:kitware/trame-cookiecutter
```

Anatomy of the cookiecutter



- **Generic trame application**
  - Code layout and split
- **Bundles**
  - Desktop
  - Service with docker
- **Jupyter integration**
  - Embed application inside a cell
- **Example on how to bring your JavaScript widgets**
  - Adding vue.js components (option at creation)

Q&A

```
project_name [Trame App]: Visualis
Select project_type:
1 - App
2 - App with Components
3 - Components
Choose from 1, 2, 3 [1]:
author [Trame Developer]: Kitware Inc.
short_description [An example Trame application]: VTK viewer for 3d stuff
Select license:
1 - BSD License
2 - MIT License
3 - ISC License (ISCL)
4 - Apache Software License
5 - GNU General Public License v3 (GPLv3)
6 - Other
Choose from 1, 2, 3, 4, 5, 6 [1]: 4
include_continuous_integration [y]: n
package_name [visualis]:
import_name [visualis]:
```

Let’s install it + various add-on…

```
cd visualis
pip install -e .
pip install pywebview jupyterlab
```

Let’s run it…

```
visualis
```

```
visualis --app
```

```
jupyter-lab
```

```
from visualis.app.jupyter import show
show()
```

```
show(height=200)
```

ParaView

10 minutes

ParaView Gotcha

- **ParaView can run VTK code too…**
  - We can use ParaView to provide VTK-EGL backend
- **ParaView has its own python but without trame…**
  - How to run your application with your venv?
    - Use “pvpython -m paraview.apps.trame …”
    - Use a custom python start file
- **ParaView is not* compatible with your python**
- **Try running paraview-visualizer**
  - https://github.com/Kitware/paraview-visualizer

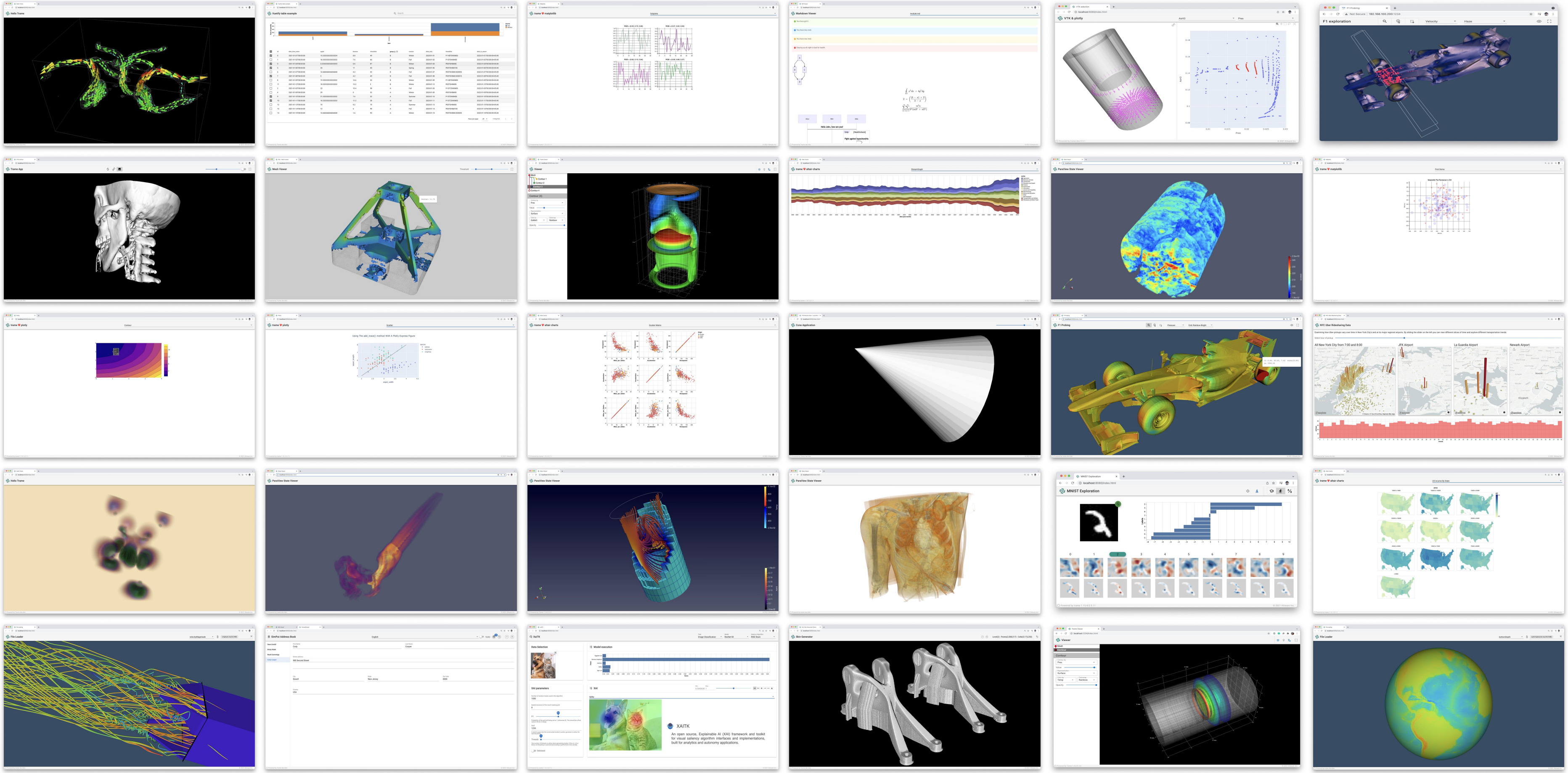
Deployment

5 minutes

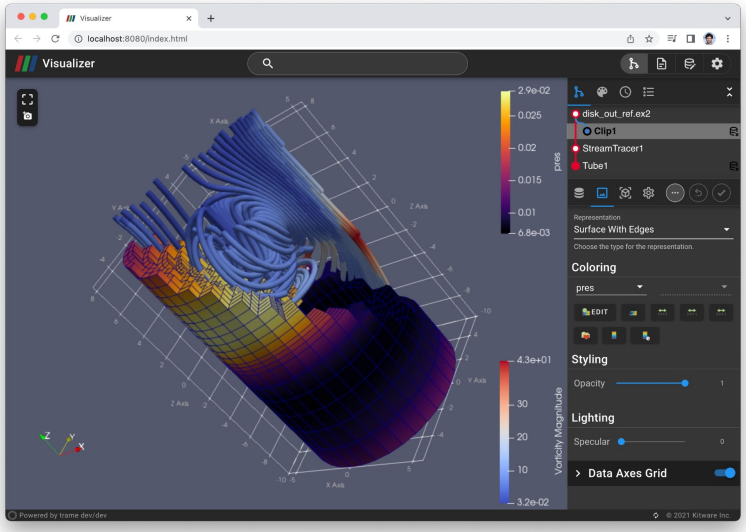
Deployment

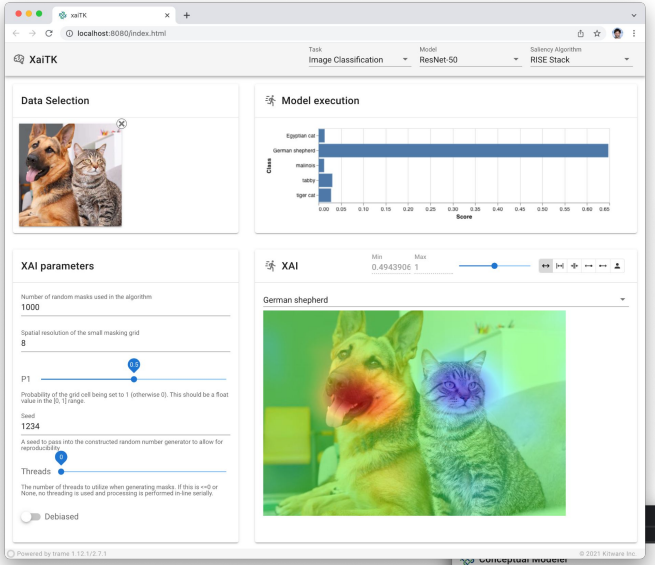
- **Desktop**
  - pywebview + pyinstaller
  - Use tk for file dialog and skip browser security limitation
- **Cloud service**
  - Docker image with multi users routing for a single node
  - Reverse connection + relay (infinity scaling)
- **Jupyter**
  - Trame provides helpers to run and display your application within your Jupyter cells

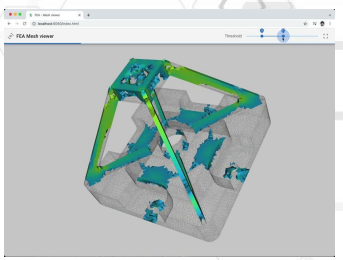
Examples

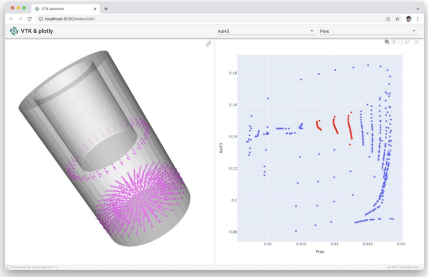

F1 exploration
MNIST Exploration
XAITK
An open source, Explainable AI (XAI) framework and toolkit for visual saliency algorithm interfaces and implementations, built for analytics and autonomy applications.

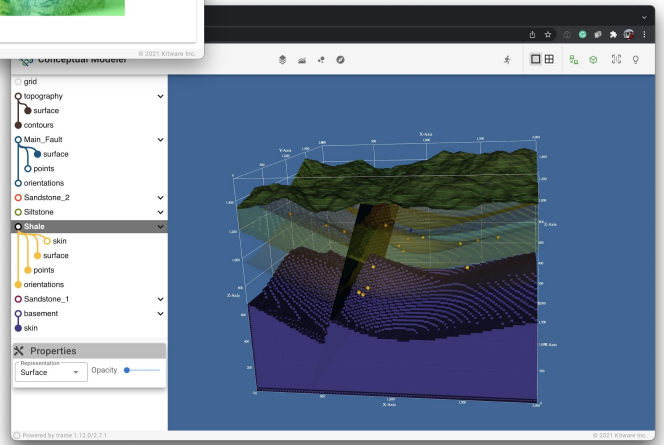
Applications











kitware

Questions / Discussions

sebastien.jourdain@kitware.com

Additional materials

- **Trame**
  - https://kitware.github.io/trame/
- **Course video**
  - https://vimeo.com/761096621/af2287747f
- **Course examples**
  - https://drive.google.com/file/d/1jU1lCi8_ts99C2U3sk_EmiTA015U84kf/view?usp=sharing