Novel visualization techniques from the *Visualize This!* competition

Alex Razoumov
alex.razoumov@westgrid.ca

WestGrid / Compute Canada

Competition goals

Since 2016

- Draw researchers' attention to popular 3D open-source tools and workflows for scientific visualization
  - ParaView
  - VisIt
  - other Python libraries and toolkits (VTK/MayaVi/etc.)
  - perhaps, domain-specific tools
  - custom C++ OpenGL/VTK code
  - anything else open-source
- Find new innovative visualization techniques + make them accessible to all Canadian researchers
  - participants have to submit not only their final visualization, but also scripts or state files so that we could *reproduce their workflow*
  - crowdsourcing solutions to complex visualization problems

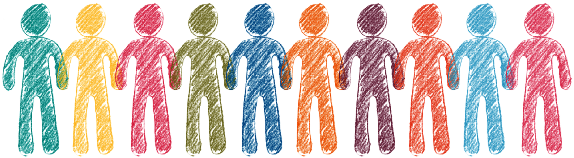
2017 Challenge Highlights

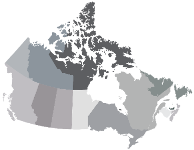
88 participants **89%** first-time participants **11%** returned from 2016 Challenge

80% students
(undergrad, grad, masters, PhD, postdoc)

12% 'other'
(developer, non-research staff, research assistant, research programmer, etc.)

8% faculty





Participant Locations
37% Western Canada
29% Ontario
18% Quebec
9% Atlantic
7% International

Top 5 disciplines represented
Computer Science
Physics
Civil Engineering
Mechanical Engineering
Earth & Ocean Science

Unexpected: all submissions were done with ParaView!

Our special thanks go to

DELL EMC intel

for providing the prizes

Also thanks to Artem Korobenko$^{(*)}$ for providing this year's dataset

(*) Mechanical and Manufacturing Engineering
Schulich School of Engineering
University of Calgary

- Time-dependent simulation of airflow around counter-rotating wind turbines
- Single time step

Competition results

Top three winning visualizations at
`https://www.westgrid.ca/visualizethis_challenge`

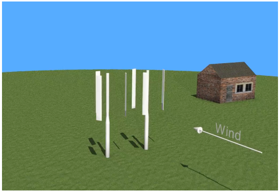
- First place: **Jarno van der Kolk**
  Postdoctoral Researcher, Department of Physics, University of Ottawa
  Winner of the Dell EMC 43" 4K Multi-Client Monitor

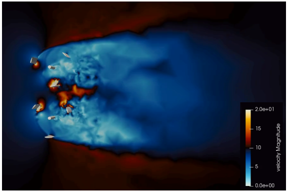
- Second place: **Nadya Moisseeva**
  PhD student, Department of Earth, Ocean & Atmospheric Sciences, UBC
  Winner of two Intel SSD drives

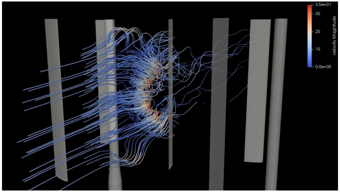
- Third place: **Thangam Natarajan, Dan MacDonald, Richard Windeyer, Peter Coppin, and David Steinman**
  Biomedical Simulation Laboratory of the University of Toronto
  Perceptual Artifacts Laboratory of OCAD University
  Winners of two Intel SSD drives

First place: Jarno van der Kolk, UofOttawa

Fantastic overall presentation, informative voice-over, extra research on windmills







(1) Toy 3D conceptual animation of rotating blades done entirely in ParaView

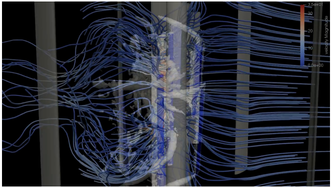
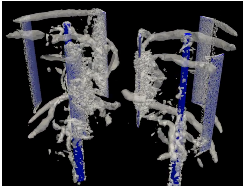
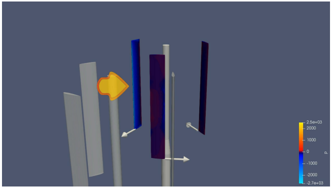
- grass, house, roof are repeated more than once $\Rightarrow$ implemented as Programmable Sources outputting vtkUnstructuredGrid or vtkPolyData with texture set to a PNG image
- door, window appear only once $\Rightarrow$ implemented as Sources $\rightarrow$ Plane with texture set to a PNG image
- OSPRay ray tracing for shadows
- toy animation fades nicely into the scientific visualization

(2) Variation of cross-section along the vertical direction + nice colour scheme for showing the wind speed

- faster than incoming in red
- slower than incoming in blue

First place: Jarno van der Kolk, UofOttawa (cont.)

Fantastic overall presentation, very informative voice-over, research on windmills

(3) Velocity streamlines with colour showing the air speed

(4) Q-criterion isosurfaces for vorticity

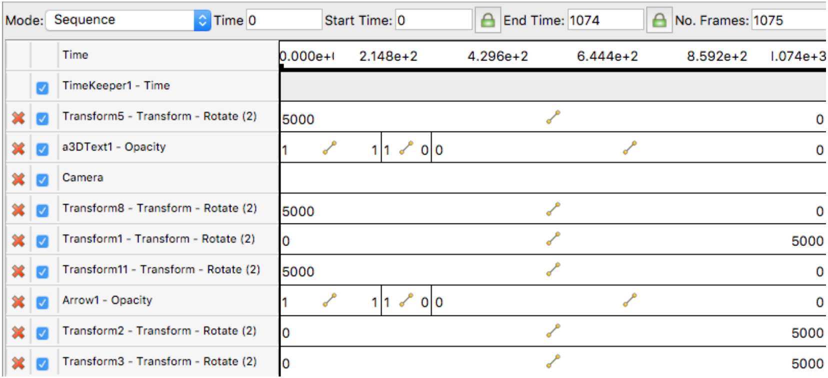
(5) Pressure field on the blades

Toy 3D conceptual animation

```
# create 1075 animation PNG frames
for i in {0..1074..100}; do     # loops 0..99, 100..199, ... (11 times)
  echo pvbatch --use-offscreen-rendering animate.py $i $(expr $i + 99)
done
# merge frames into a movie
ffmpeg -r 25 -i frame.%04d.png -c:v libx264 -crf 0 -preset ultrafast windmill.mp4 -y
```

- each `animate.py`
  - processes command-line arguments startTime/endTime
  - loads the state file `animatedBlades.pvsm`
  - goes through all scenes startTime..endTime in the animation timeline and writes corresponding PNG frames
- the state file `animatedBlades.pvsm`
  - loads the blades
  - applies zero rotation to each blade (to be used in animation)
  - creates Grass / House Base/ House Roof Programmable Sources
  - creates arrow and 3D text
  - creates Door / Window Planes
  - creates animation timeline (next page)

Conceptual animation timeline



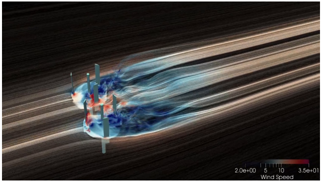
Grass Programmable Source

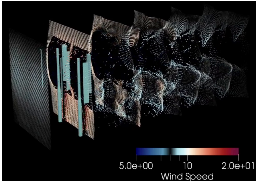
Output Data Set Type = vtkUnstructuredGrid
Texture = grass_rough2.png

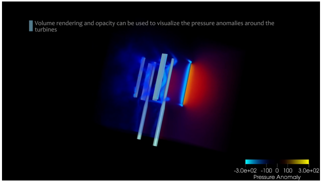
```
numQuadsX, numQuadsY = 400, 400
dx, dy = 0.25, 0.25
random.seed(1234)
pts = vtk.vtkPoints()
pts.SetNumberOfPoints(4)
for i in xrange(numQuadsX*numQuadsY):
    pts.InsertPoint(i, dx*(i % numQuadsX - numQuadsX/2), dy*(i / numQuadsX - numQuadsY/2), -3+.05*random.rand())

output.Allocate(numQuadsX, numQuadsX)
tc = vtk.vtkFloatArray()
tc.SetNumberOfComponents(2)
tc.SetNumberOfTuples(numQuadsX*numQuadsY)
tc.SetName("TextureCoordinates")

for iy in xrange(numQuadsY-1):
    for ix in xrange(numQuadsX-1):
        if((ix-numQuadsX/2)*(ix-numQuadsX/2)+(iy-numQuadsY/2)*(iy-numQuadsY/2) <= numQuadsX*numQuadsX/4):
            aQuad = vtk.vtkQuad()
            aQuad.GetPointIds().SetId(0, ix+numQuadsX*iy)
            aQuad.GetPointIds().SetId(1, ix+numQuadsX*iy+1)
            aQuad.GetPointIds().SetId(2, ix+numQuadsX*iy+numQuadsX+1)
            aQuad.GetPointIds().SetId(3, ix+numQuadsX*iy+numQuadsX)
            output.InsertNextCell(aQuad.GetCellType(), aQuad.GetPointIds())

for iy in xrange(numQuadsY):
    for ix in xrange(numQuadsX):
        tc.SetTuple2(ix+numQuadsX*iy, ix%2, iy%2)

output.SetPoints(pts)
output.GetPointData().AddArray(tc)
output.GetPointData().SetTCoords(tc)   # set texture coordinate data
```

Second place: Nadya Moisseeva, UBC

Five separate animations, impressive visualization techniques







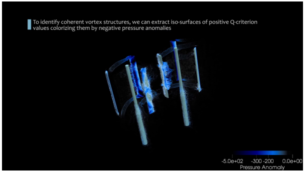
(1) "Dynamic streamlines" with a vertical swipe: using Stream Tracer With Custom Source on the velocity with a 2D grid (slice) for seed points, and then animating the slice position *forcing the streamlines to be redrawn at each height*

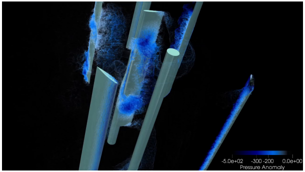
(2) Animation of wind flow deformation with integration time contours

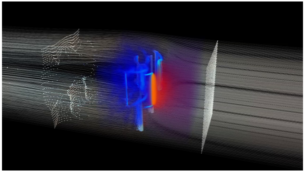
(3) Nice colours for volumetric plots of *regions of high/low pressure around the blades* and *of vorticity*

Second place: Nadya Moisseeva, UBC (cont.)

Five separate animations, impressive visualization techniques







(4) Q-criterion isosurfaces for vorticity

(5) Final multi-layer animation combining 11 timelines (*combination of previous techniques*)

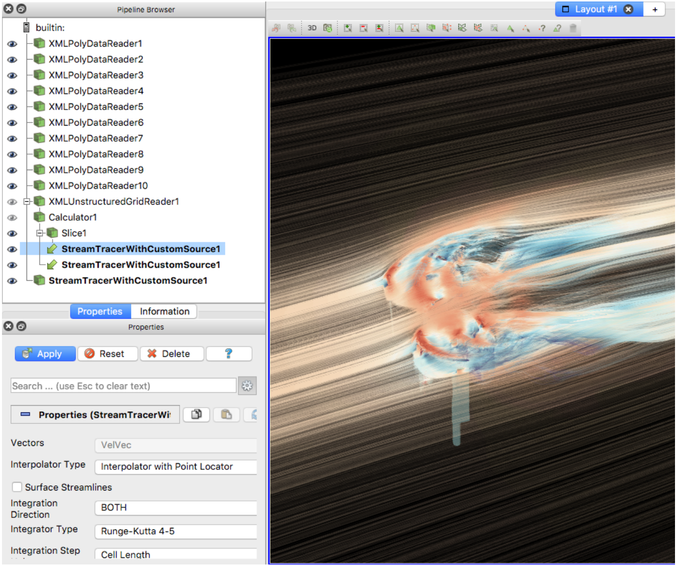
(6) Several rotation and displacement motions

(7) Smooth continuous transitions between all five animations

(8) Informative burned-in captions

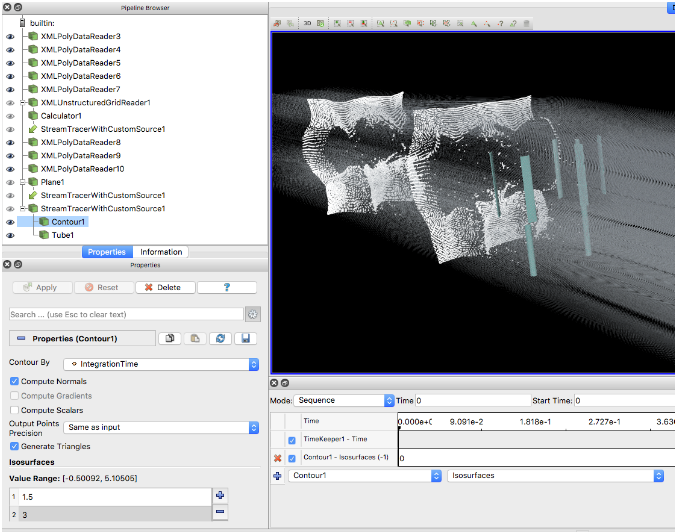
Stream Tracer With Custom Source

To animate streamlines



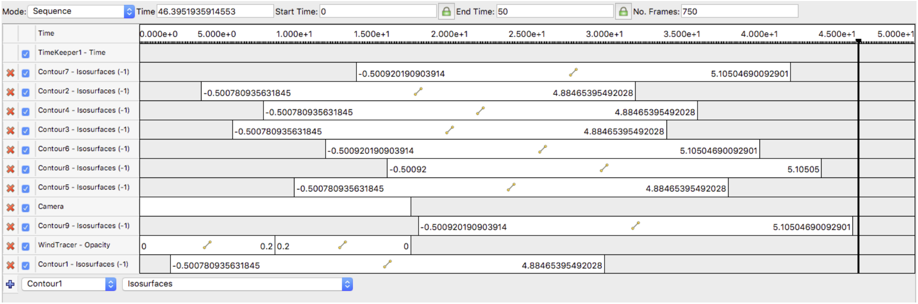
- Apply Stream Tracer With Custom Source to the output of Slice
  - *input* = 3D data
  - *seed source* = slice
- Animation View: animate the slice position from top to bottom

Animating integration time contours



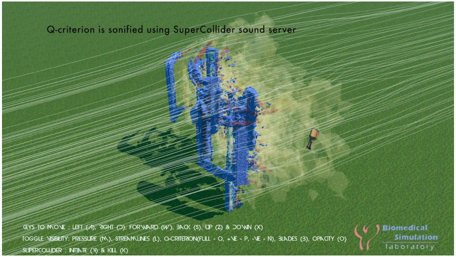
- Start with the streamtracer lines, however drawn
- Apply a Countour filter to the output of Streamtracer
  - contour by Integration Time
  - probe the range of values that works best
- *If static view:* multiple values in a single Contour
- *If animation:* multiple Contour filters
- Animation View: animate Contour - Isosurfaces for each Contour

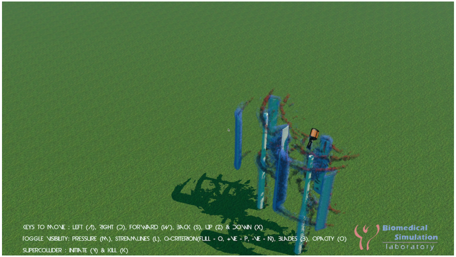
Many timelines/variables in a single animation



Third place: Thangam Natarajan, Dan MacDonald, Richard Windeyer, Peter Coppin, and David Steinman, UofT/OCAD

Exploring ParaView scenes in Blender Game Engine, sonification (on-the-fly audio)





- Rendered isosurfaces and streamlines in ParaView, exported them as X3D scenes to Blender, improved the 3D model aesthetics in Blender, created a 3D Blender Game Engine environment
- Output packaged as a Mac app: a user can move through the scene, toggle the visibility of various components
- Sonification: using the SuperCollider synthesizer server to produce on-the-fly audio from the Q-criterion under the microphone in the game engine

Summary

- Thank you to all who submitted their entries: many of you have put a lot of effort and time into your visualizations!
- All 2017 submissions used ParaView, but you can do similar renderings in VisIt
- All ParaView animations in this presentation could be done with either GUI or scripting
  - in these slides focused on GUI workflows for clarity
  - can animate any property of any pipeline object, camera variables, combine multiple timelines in a single animation
- If you want scripts for a specific visualization technique, let me know, and I will send you a simplified version
- We are looking for a great dataset for this fall's competition

Questions?

- Webstream viewers: email `info@westgrid.ca`
- Vidyo viewers: unmute & ask question or use Vidyo Chat (chat bubble icon in Vidyo menu)
- Support email `support@computecanada.ca`
- Email me anytime `alex.razoumov@westgrid.ca`