

FreeCAD

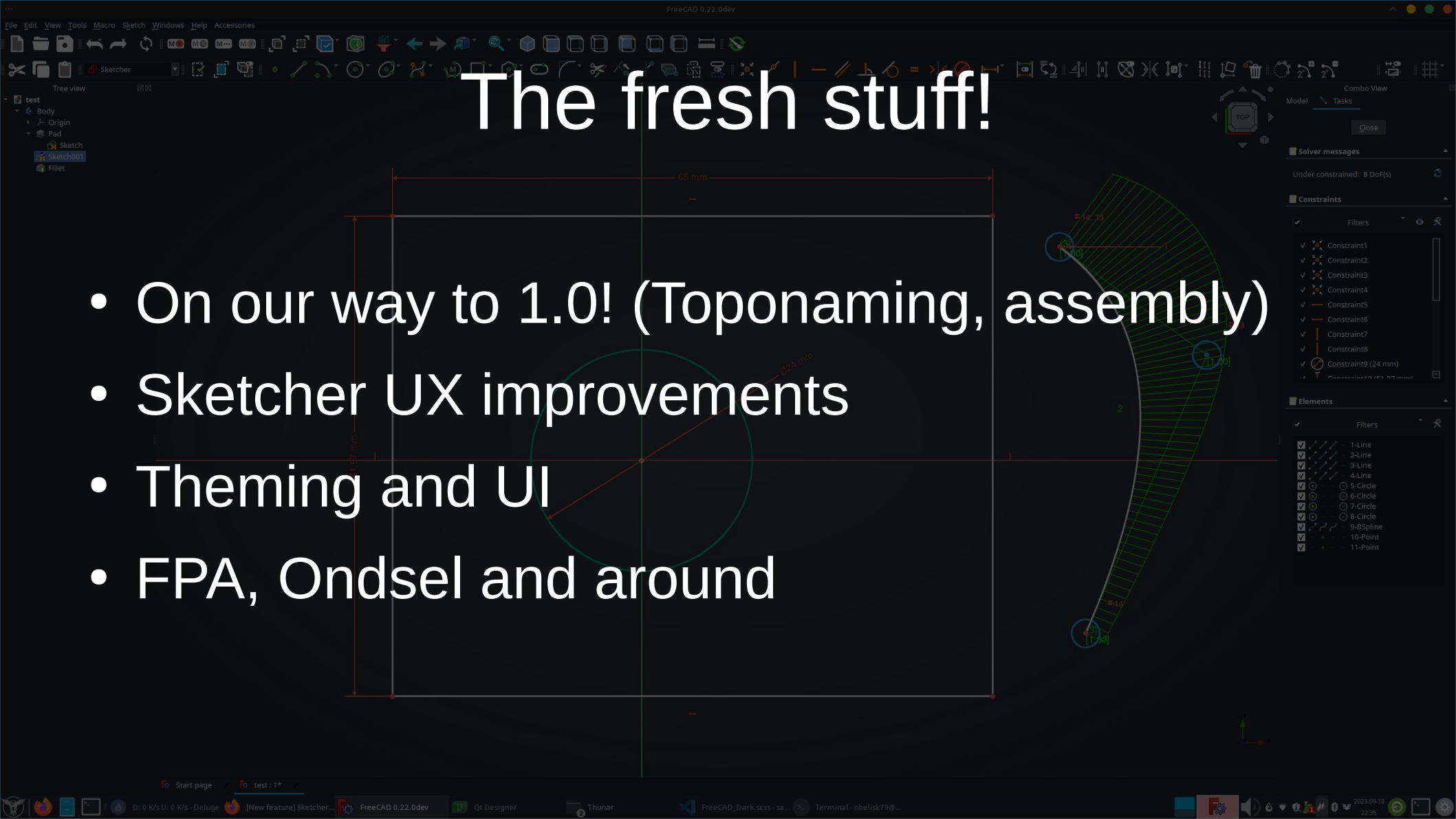
State of the union

Yorik van Havre / Aik-Siong Koh

FOSDEM 2024

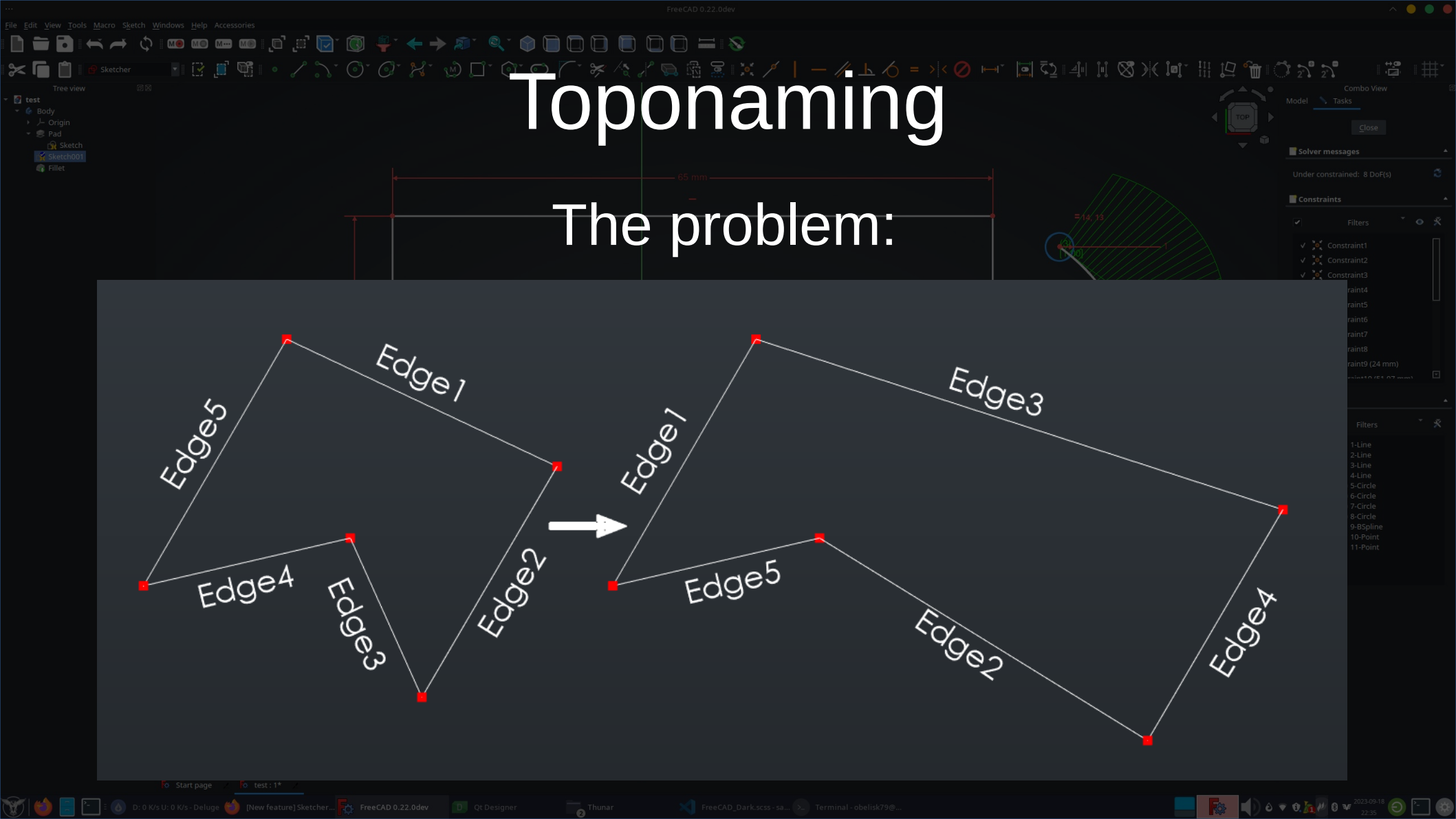
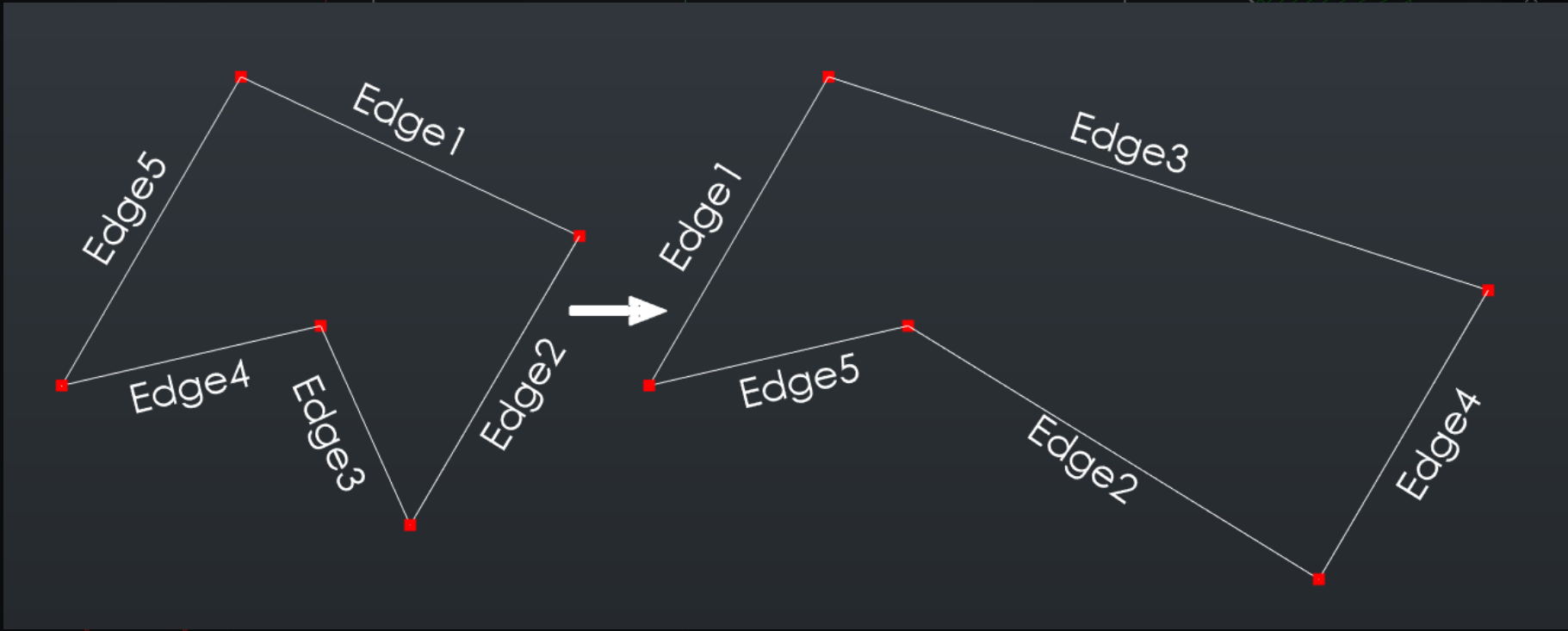
The fresh stuff!

- On our way to 1.0! (Toponaming, assembly)
- Sketcher UX improvements
- Theming and UI
- FPA, Ondsel and around



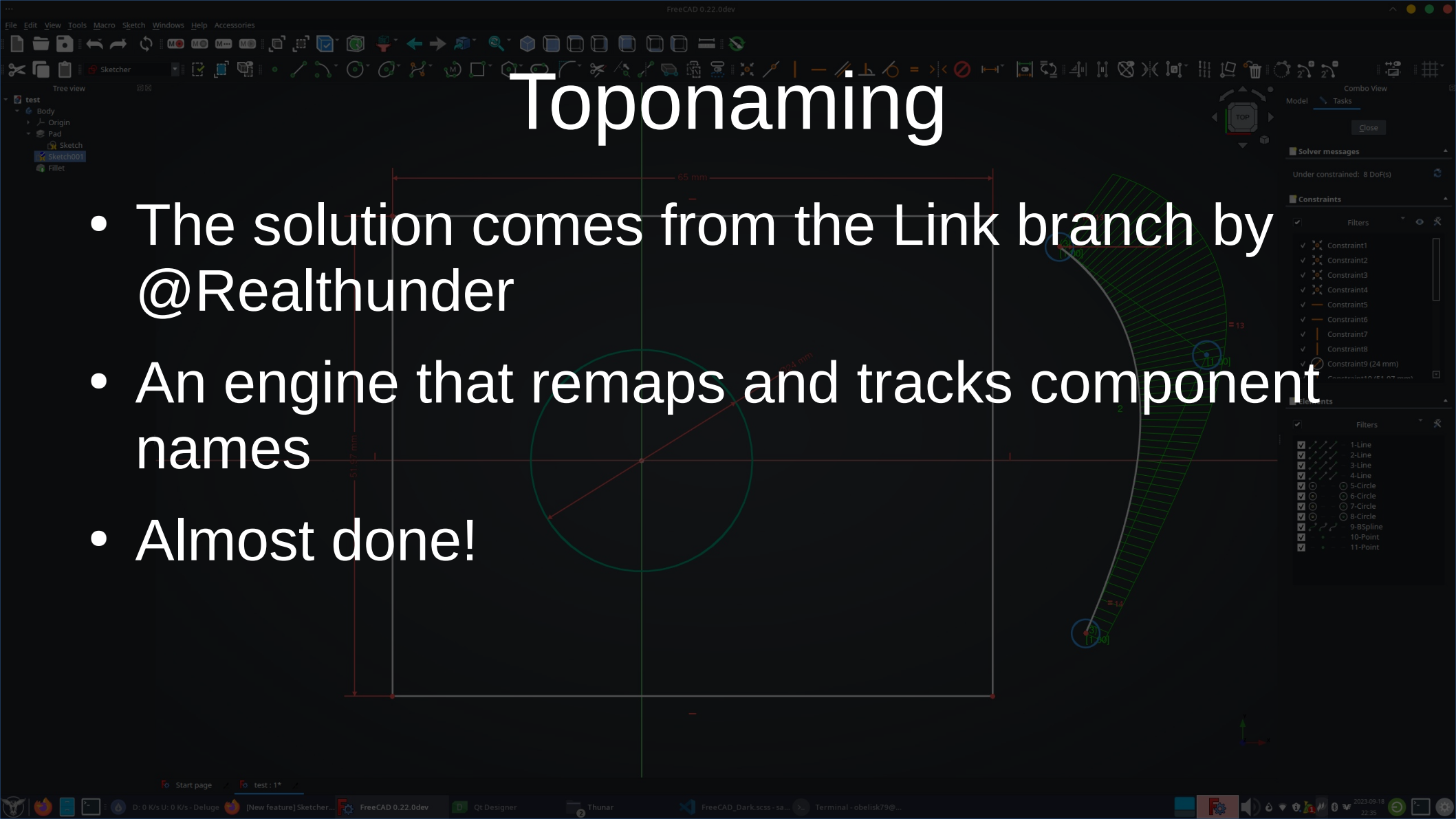
Toponaming

The problem:



Toponaming

- The solution comes from the Link branch by @Realthunder
- An engine that remaps and tracks component names
- Almost done!





- test
- Body
- Origin
- Pad
- Sketch
- Sketch001
- Fillet

Assembly

Model Tasks

Close

Solver messages

Under constrained: 8 DoF(s)

- Constraints
- Filters
- Constraint1
 - Constraint2
 - Constraint3
 - Constraint4
 - Constraint5
 - Constraint6
 - Constraint7
 - Constraint8
 - Constraint9 (24 mm)

- Elements
- Filters
- 1-Line
 - 2-Line
 - 3-Line
 - 4-Line
 - 5-Circle
 - 6-Circle
 - 7-Circle
 - 8-Circle
 - 9-Bspline
 - 10-Point
 - 11-Point

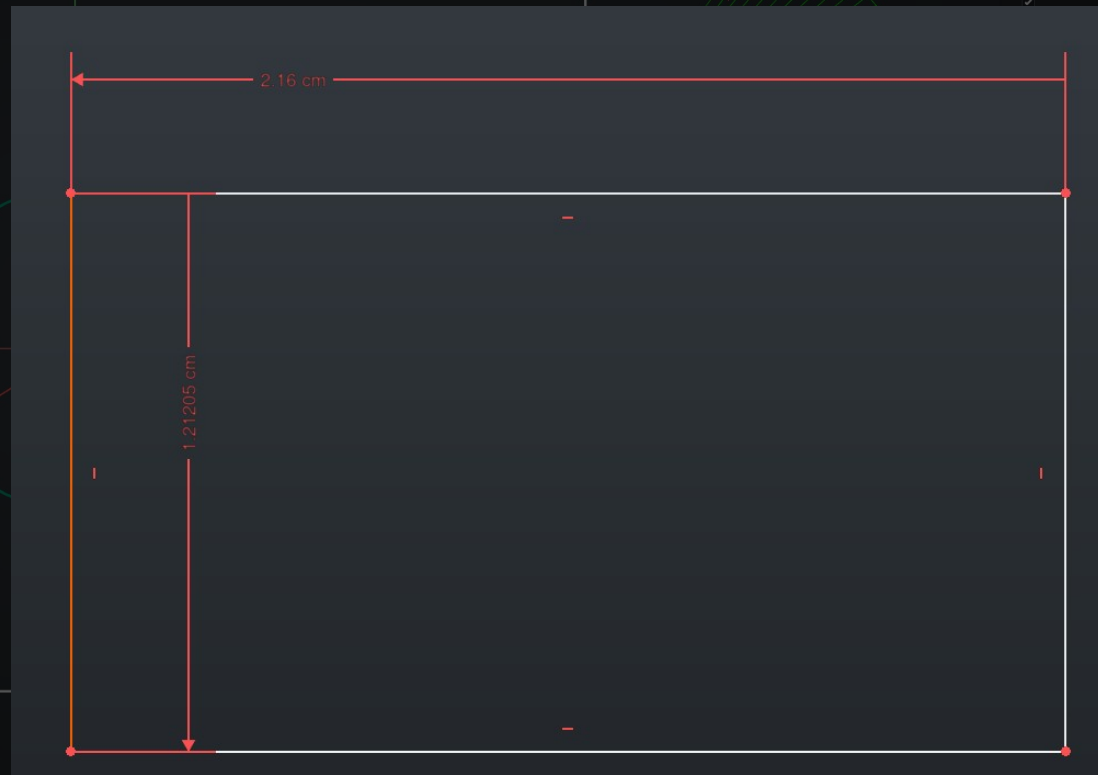


In a moment!

Sketcher UX

Auto-constraining

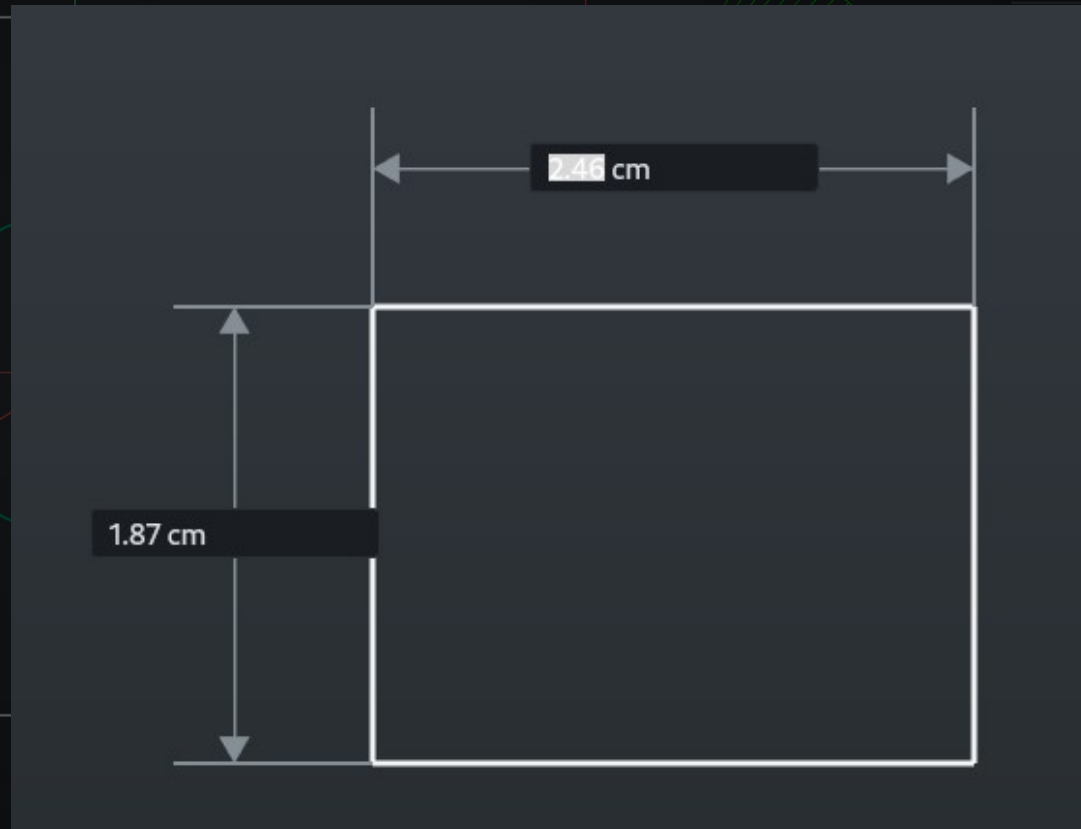
Automatically
selects
vertical/horizontal
length constraints



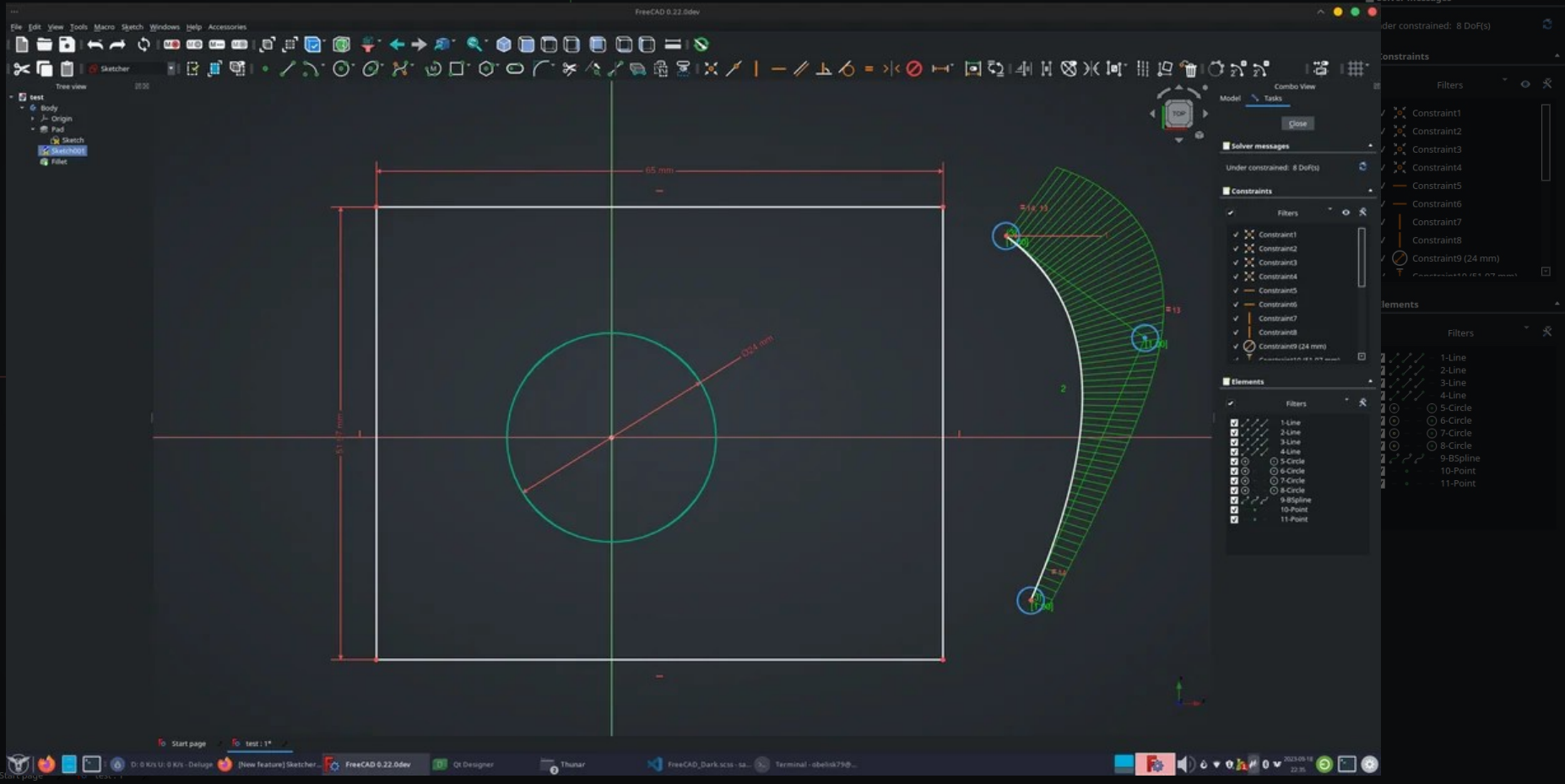
Sketcher UX

On-screen input

Allows to insert dimensions on creation



Theming and UI



Theming and UI

The screenshot displays the FreeCAD 0.22.0dev interface. The main window shows a 3D model of a mechanical part with a complex pocket. The left sidebar contains a 'Tree view' showing the object hierarchy: Application > PartDesignExample > Body > Origin > Pad > Sketch > Pocket > Sketch001 > Pocket001 > Sketch003 > Pocket002 > Sketch002. Below the tree is a 'Property view' table:

Property	Value

On the right, a 'Combo View' dialog is open for 'Pocket parameters'. The settings are:

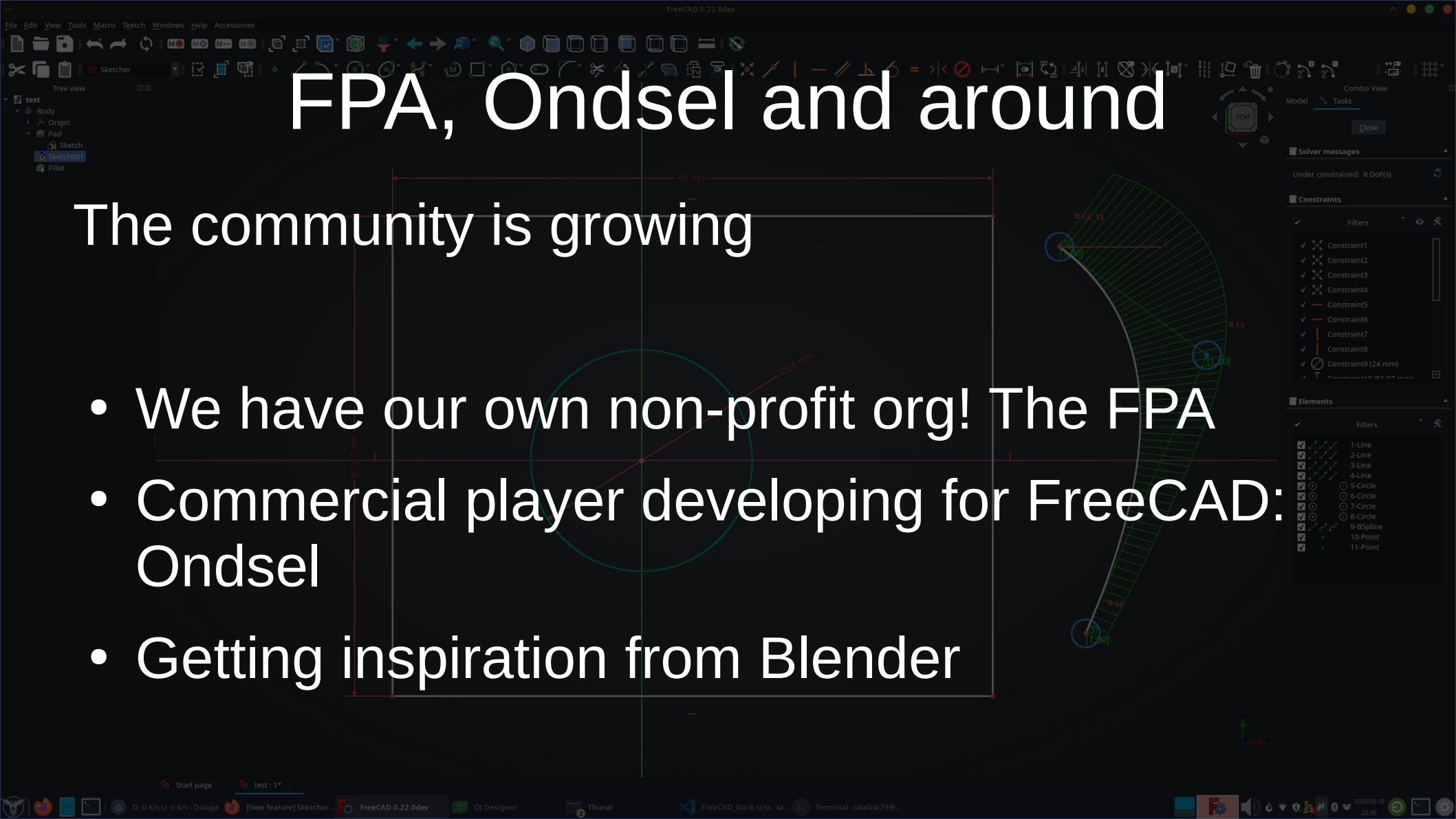
- Type: Through all
- Offset to face: 0.00 mm
- Direction: Sketch normal
- Show direction
- Symmetric to plane
- Reversed
- Select face: No face selected
- Update view

At the bottom, there are three panels: 'Report view' (empty), 'Python console' (showing Python 3.10.6 environment info), and a status bar at the very bottom indicating 'Valid, Internal name: Pocket002' and 'Blender 309,03 mm x 189,37 mm'.

FPA, Ondsel and around

The community is growing

- We have our own non-profit org! The FPA
- Commercial player developing for FreeCAD: Ondsel
- Getting inspiration from Blender



Ondsel Assembly Solver

Aik-Siong Koh

2024-02-04 Sun

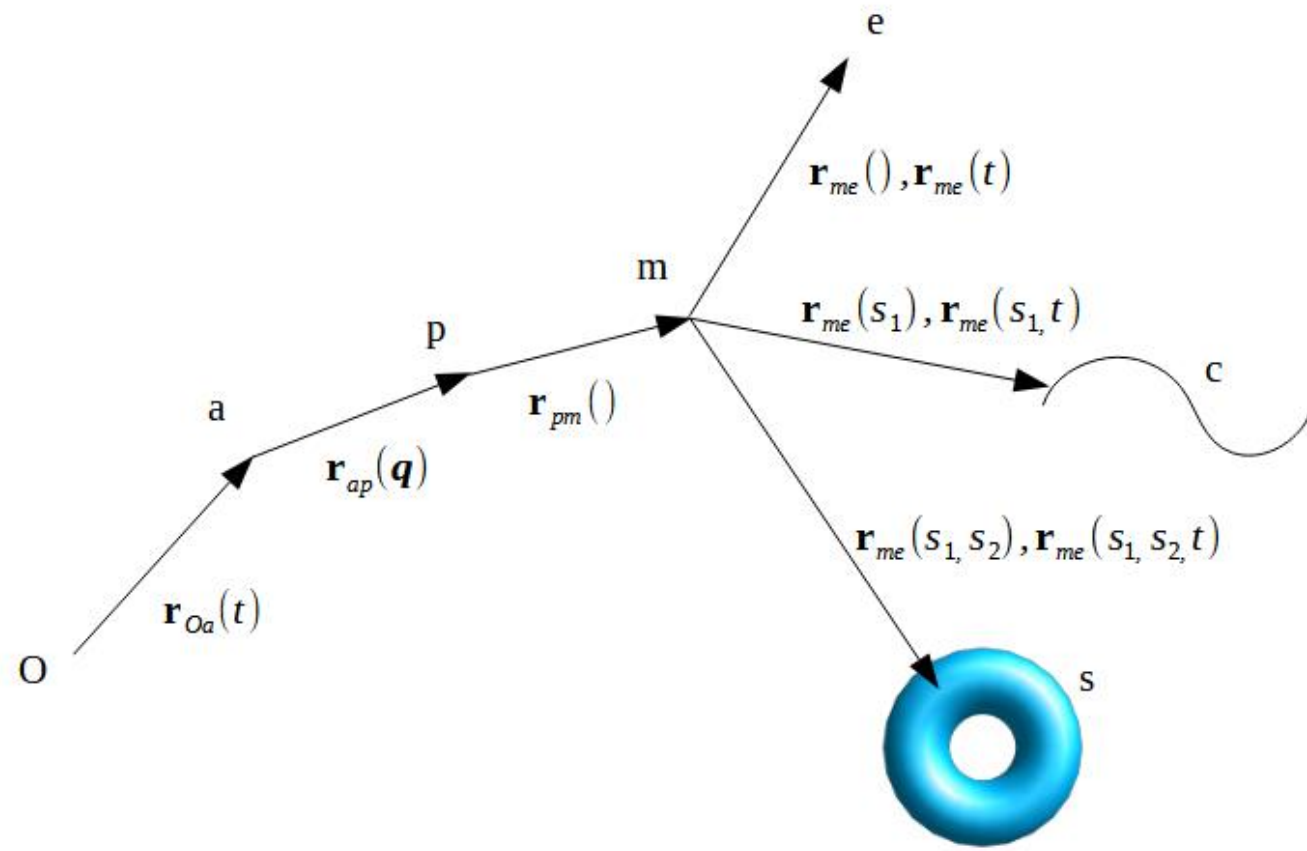
freeCAD by askoh

- Basic 3D CAD with Motion Simulation
 - <https://www.ar-cad.com/>
- Visualworks Smalltalk and OpenGL
- Used as addin in Alibre, SpaceClaim

OndselServer

- Assembly constraints for FreeCAD.org
- Smalltalk motion simulator translated to C++
- <https://ondsel.com/blog/>
- <https://github.com/Ondsel-Development/MbDTheory>
- <https://github.com/Ondsel-Development/OndselSolver>

Assembly Theory



Constraints

- Absolute
- Euler Parameter
- At Point
- In Plane
- Perpendicular
- Distance
- Constant Velocity
- Coupler

$$\mathbf{G}_{abs} = \mathbf{q}_{Ip} = \mathbf{0}$$

$$G_E = E_1^2 + E_2^2 + E_3^2 + E_4^2 - 1 = 0$$

$$\mathbf{G}_{IeJeO}(\mathbf{q}, \mathbf{s}, t) = \mathbf{r}_{IeJeO}(\mathbf{q}, \mathbf{s}, t) = \mathbf{0}$$

$$\mathbf{G}_{IeJeIe}(\mathbf{q}, \mathbf{s}, t) = \mathbf{r}_{IeJeIe}(\mathbf{q}, \mathbf{s}, t) = \mathbf{0}$$

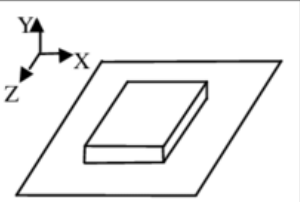
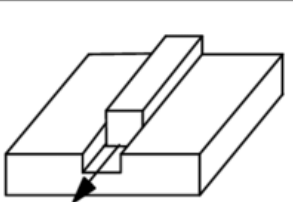
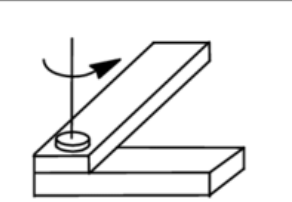
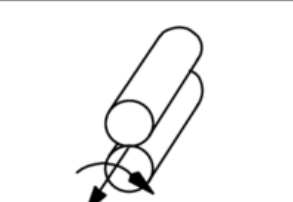
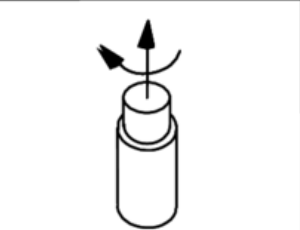
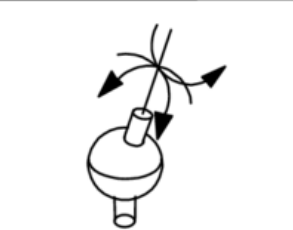
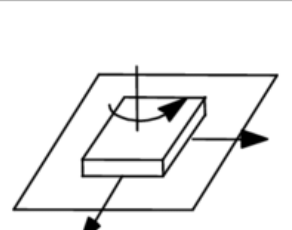
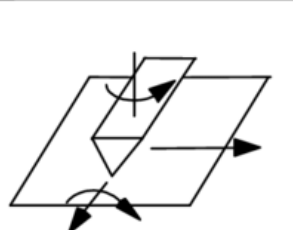
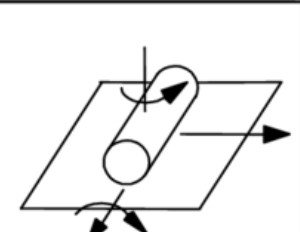
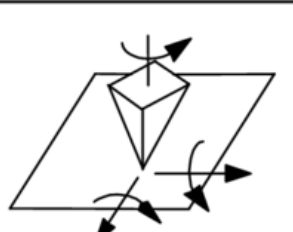
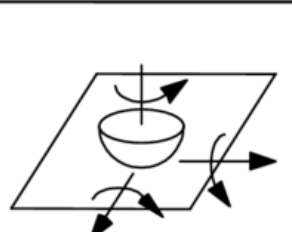
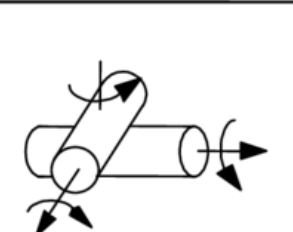
$$G_{\perp}(\mathbf{q}, \mathbf{s}, t) = \mathbf{n}_{IeO}^T \mathbf{t}_{JeO} = 0$$

$$G_{IeIe} = r_{IeIe} - f_{IeIe}(t) = 0$$

$$G_{\omega}(\mathbf{q}, \mathbf{s}, t) = \mathbf{i}_{IeO}^T \mathbf{j}_{JeO} + \mathbf{j}_{IeO}^T \mathbf{i}_{JeO} = 0$$

$$G_{nlc}(\mathbf{q}, \mathbf{s}, t) = G_{nlc}(r_{iIeJeIe}, \theta_{iIeJeIe}, \dots) = 0$$

Joints

 <p>Rigid (no motion)</p>	 <p>Prismatic (1)</p>	 <p>Revolute (1)</p>	 <p>Parallel Cylinders (2)</p>
 <p>Cylindrical (2)</p>	 <p>Spherical (3)</p>	 <p>Planar (3)</p>	 <p>Edge Slider (4)</p>
 <p>Cylindrical Slider (4)</p>	 <p>Point Slider (5)</p>	 <p>Spherical Slider (5)</p>	 <p>Crossed Cylinders (5)</p>

Smalltalk to C++ Translation

- Simplified C++
 - Very Smalltalk like
- Public and Virtual methods
- Use Smart Pointer `std::shared_ptr`
 - Pointer with reference counting
 - No memory leak worries
 - No new or delete
 - No difference in passing by value or reference
 - Need to avoid circularity

Digital Twin Concept (2002)

The Digital Twin



Engineering

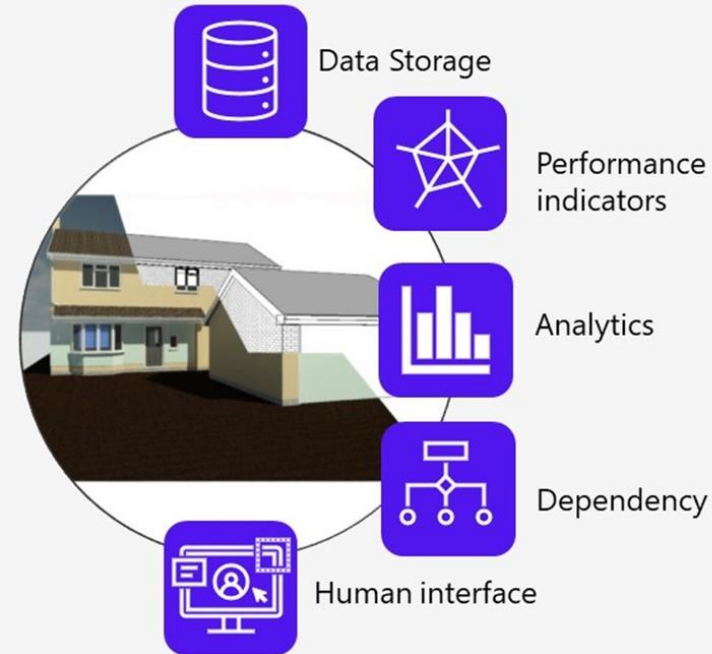
- Drawings
- Specification
- BIM Model

Operations

- IoT Feeds
- Sensors
- Smart Appliances
- Maintenance
- Occupation
- Energy

Information

- Asset Locations
- Asset Details
- Product Details
- Maintenance Regimes
- Inspections



Physical

Digital

Digital Twin Concept

Digital
Twin



Modeling
Simulation
Animation

Digital-TwinS: Digital Twin applied to Software

- Combine best of static and dynamic languages
- TIOBE Index (Dec 2022) popularity ranking
 1. Python (dynamic)
 2. C (static)
 3. C++ (static)

Why Digital-TwinS



C++ is best of FAST



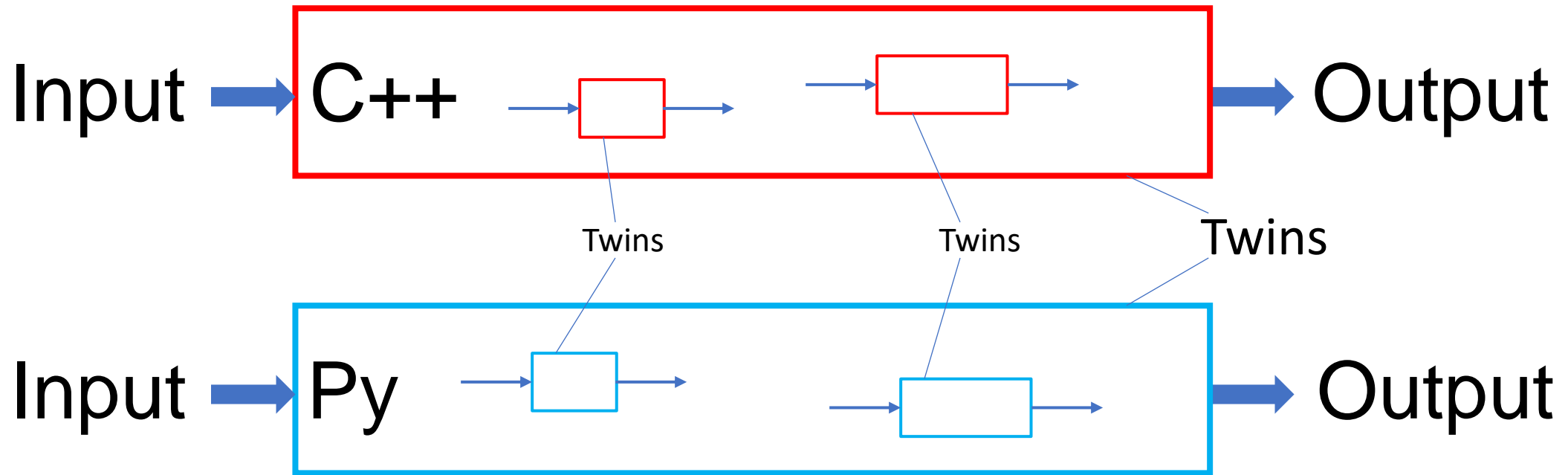
Python is very FLEXIBLE

Digital-TwinS: C++ and Python



Same Input Same Output (SISO)
Internals can be independent

Same Input Same Output (SISO)



Twins can be any size or any component
Internals can be partially dependent

C++ is FAST at all cost



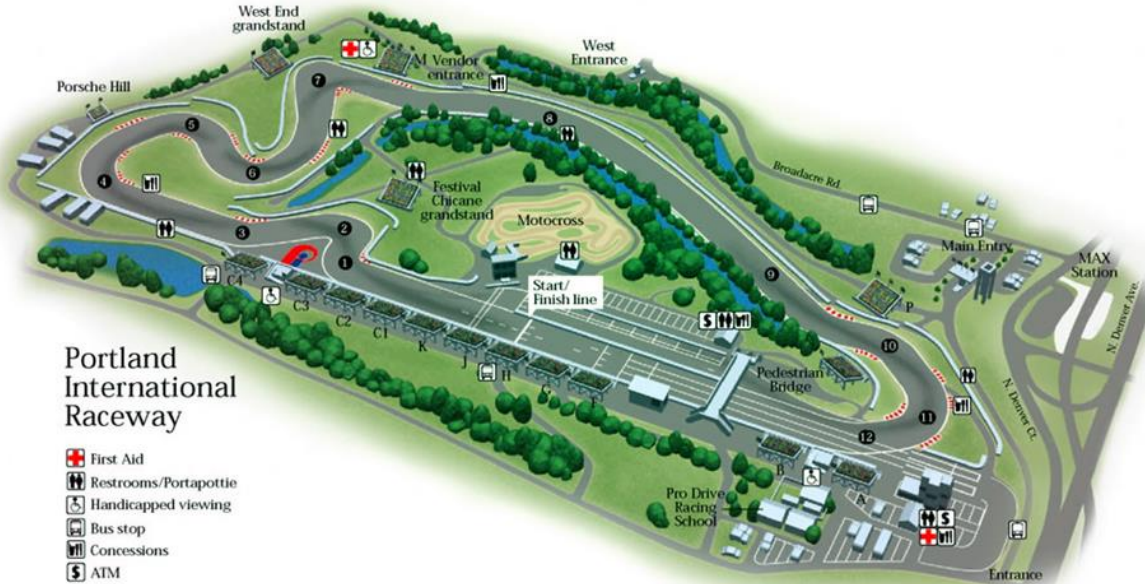
Python is NIMBLE and rugged
Low cost



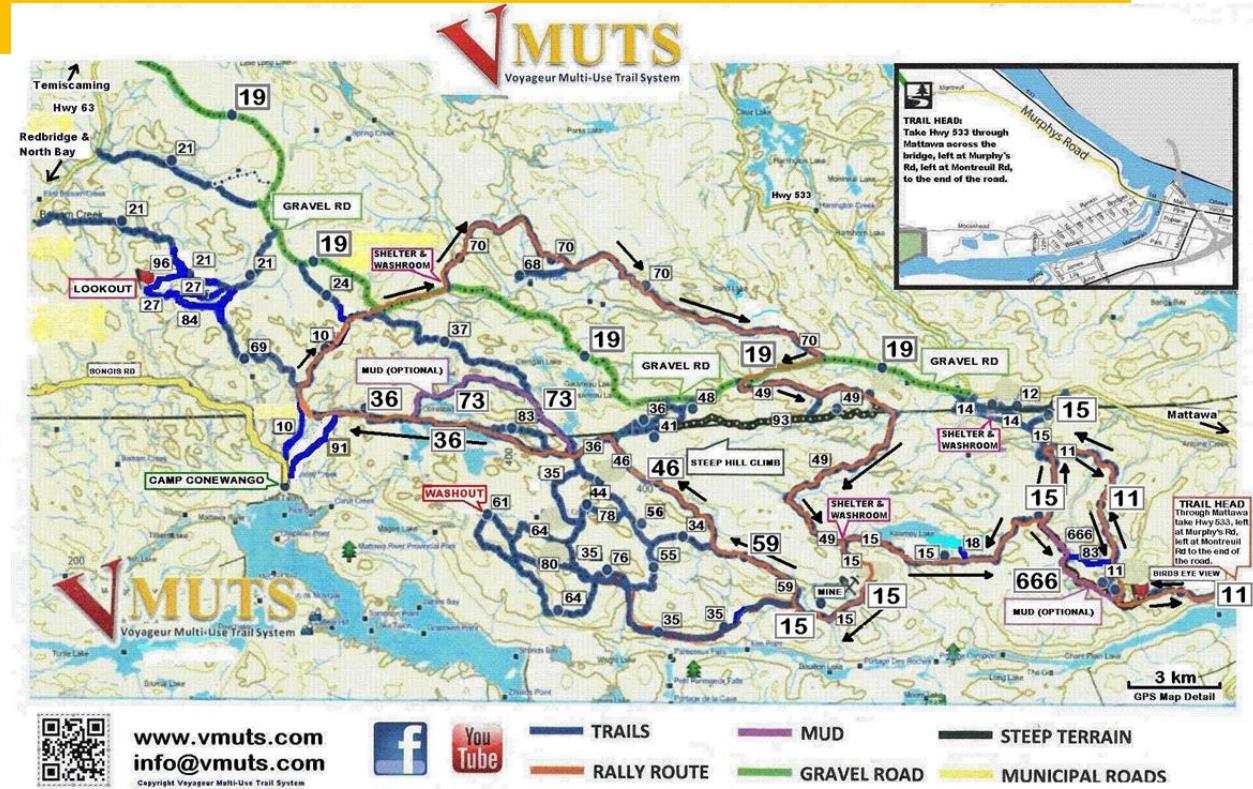
A hybrid vehicle would have compromised capabilities
Java, C#, Obj C

C++ Heavy Infrastructure
Small area

Python Light Infrastructure
Large area



Execution



Exploration

We want to win in both settings

Why Digital-TwinS cont.

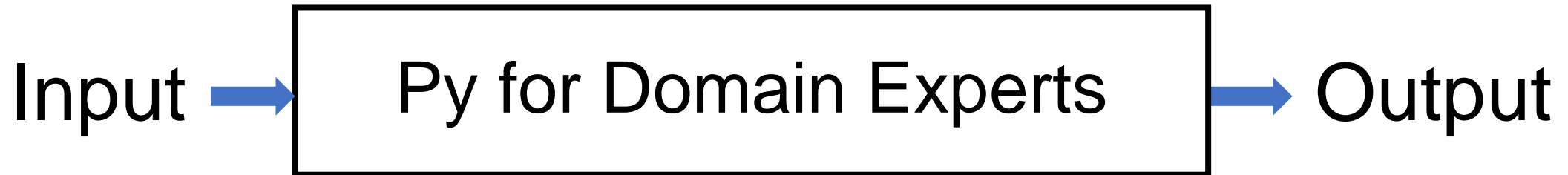
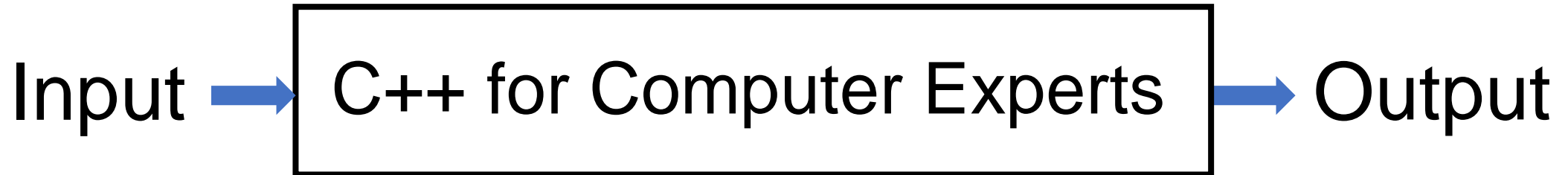
Input → C++ is Machine centric → Output

Input → Py is Programmer centric → Output

Humans think Objects

“Development at the speed of thought”

Why Digital-TwinS cont. 1



Synergy and feedback between experts

Python and FreeCAD for Brain Dump

Why Digital-TwinS cont. 2

- Assume developing a brand-new feature.
- Python alone can do it in T days. But the feature is slow.
- C++ alone can do it in $5T$ days. But the feature is fast.
- Twins can do it in $3T$ days. Python development T days. Guided port to C++ is $2T$ days. Feature is fast and development is shorter.
- Twins cross-checking each other will reduce bugs in both greatly. This is a bonus.

Strategy for Digital-TwinS

- Capture C++ algorithms in Python twin
 - Executable documentation
- Experiment in Python twin (superset program)
 - Fearless programming
- Transfer discoveries to C++ twin
 - Manually, automated or both
 - Strict testing
 - Iterate with twin
- Debug in Python twin
- Transfer fixes to C++ twin

Prototype vs Digital-TwinS vs Production

Time

Prototype Dev is Nimble and Simple in Py

Production Dev is Difficult in C++. **Runs FAST**

Prototype in Py

Digital-TwinS in Py

advanced features

Production in C++

stable features

Production in C++ Slow progress

Faster progress stable features

Digital-TwinS in Py

advanced features