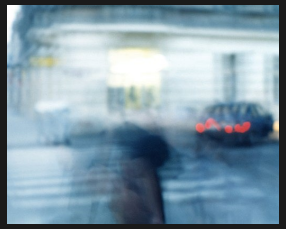
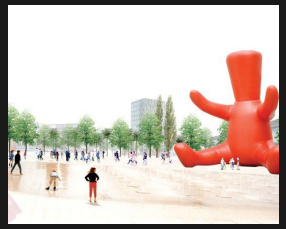
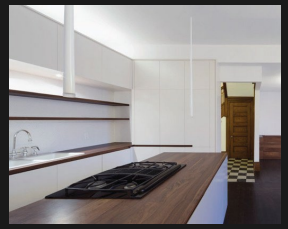
SIMO Urban Information Modelling by **Spatiomatics**
#simo #uim #cim #citybim

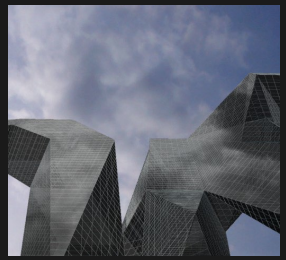
2023-06-28 – **Parametric Cafe Amsterdam**

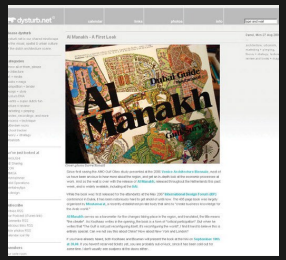
Who, Why, What & How

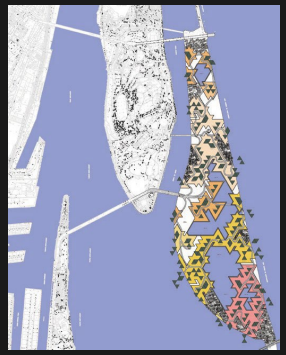












1996 – 2002
Design / Fine Arts
U of Manitoba

2003 - 2005
M.Arch
Hal Ingberg
U de Montreal

2005 - 2007
Maxwan a+u
Dysturb.Net
TU Delft

2008 - 2011
Open Form
Architecture
British School

2011 - 2019
MVRDV
KCAP
TU Delft

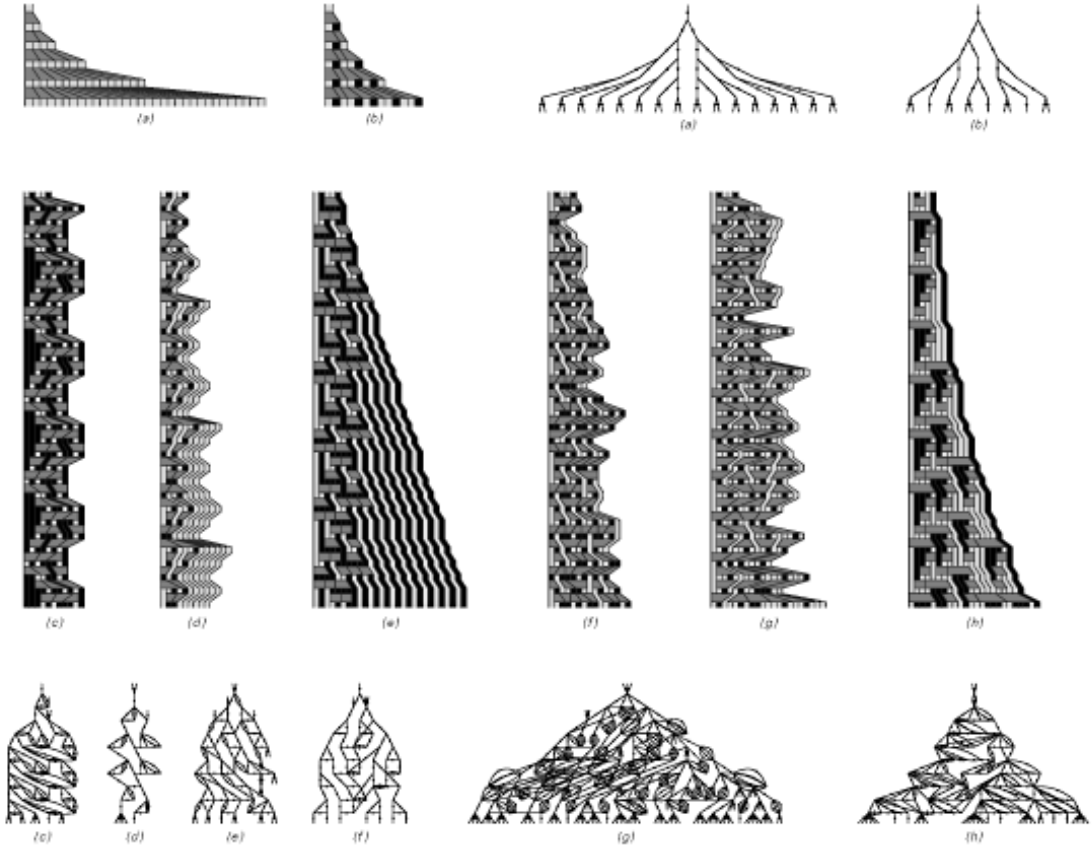
Context and Personal Journey

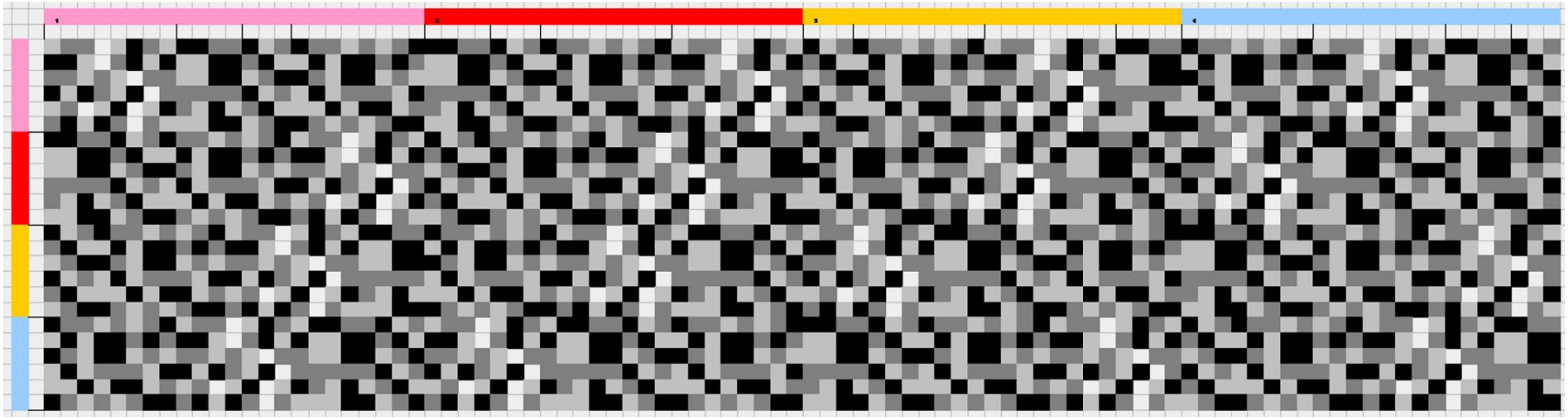
- 2016 → Accelerator GIS Collaboration
- Many years of ongoing research

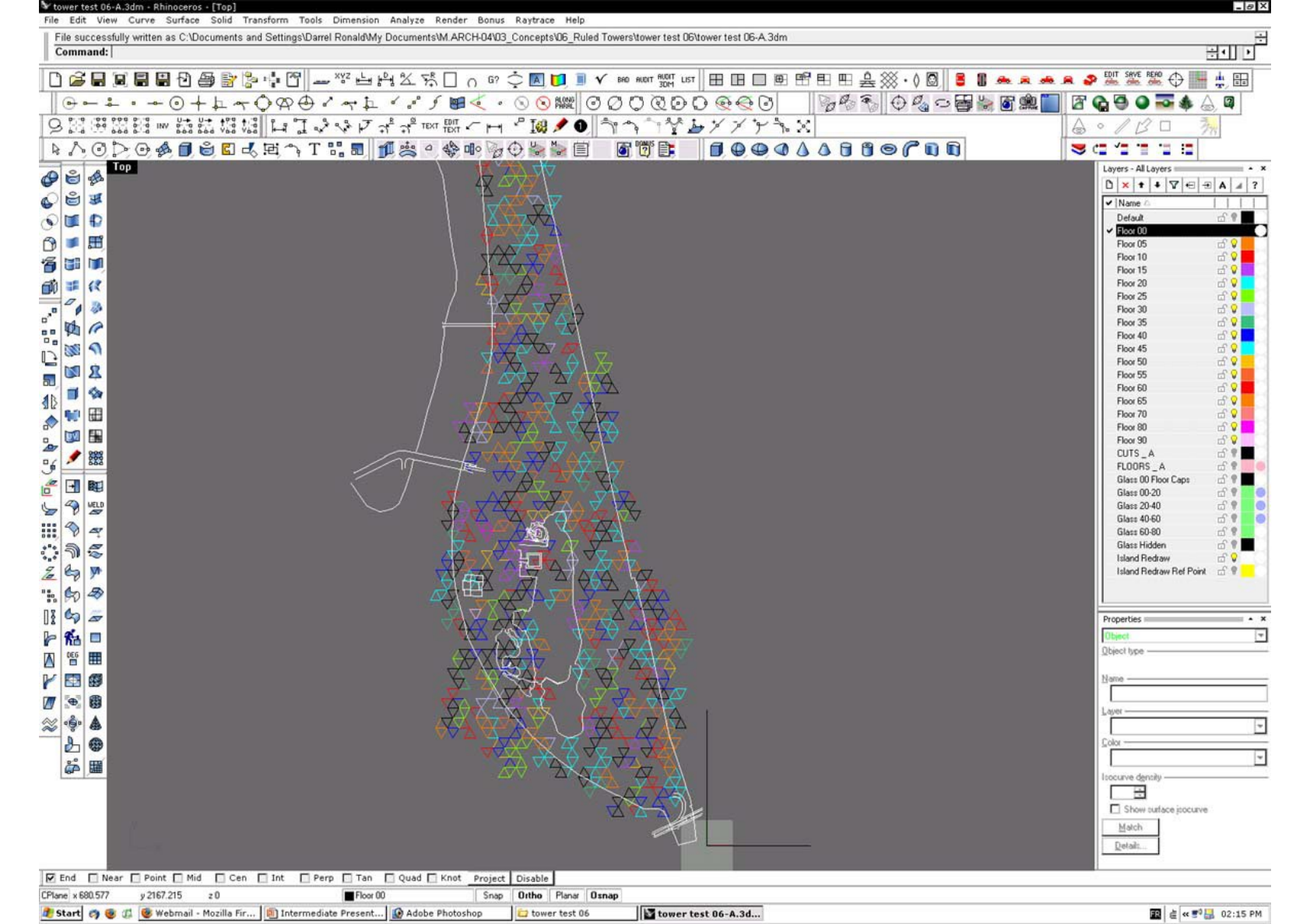
- 2017 & 2018 → Sabbaticals (Web Apps, Data Science)

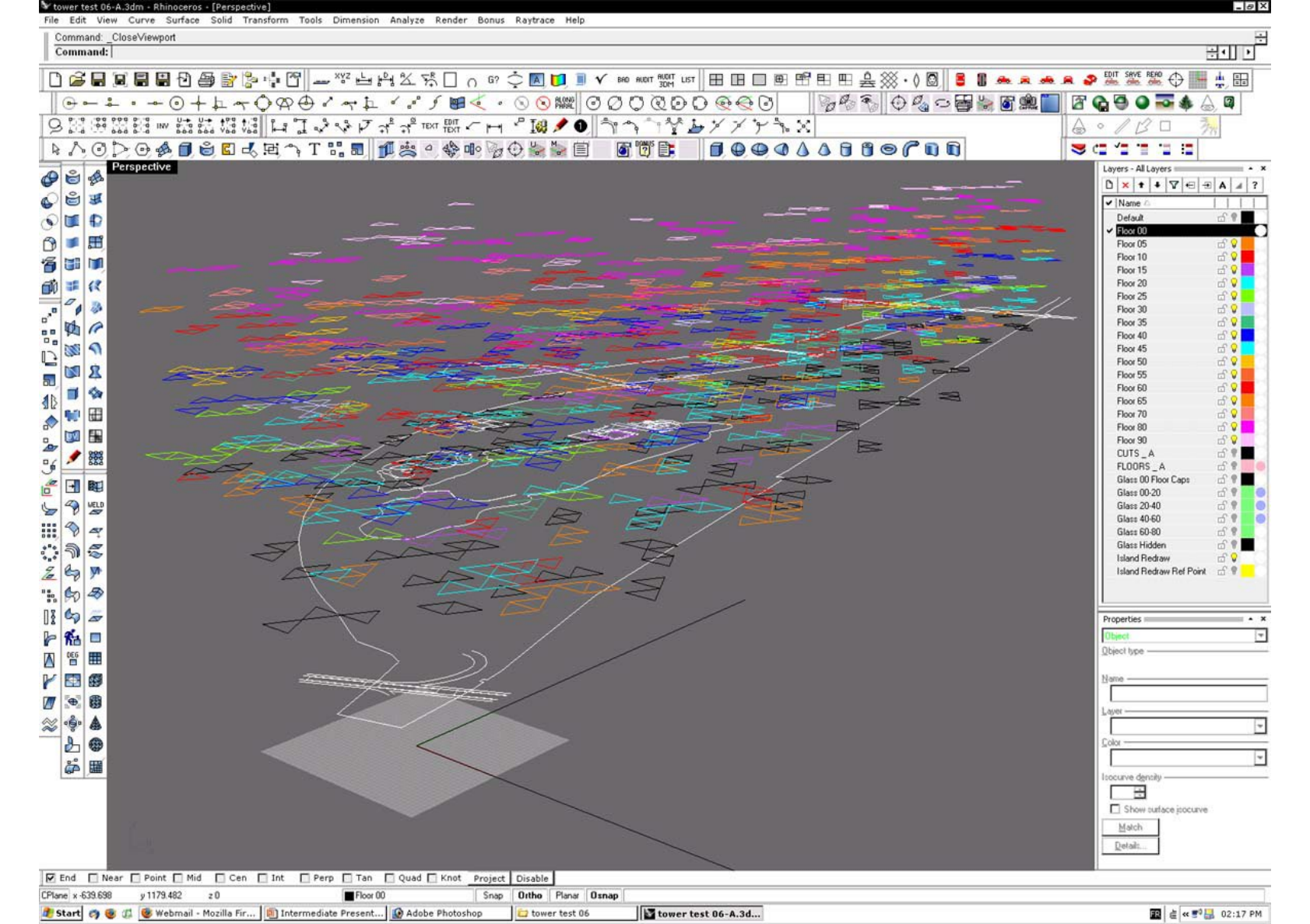
- 2019 → Computation at KCAP + leaving at end
- 2020 → Career change + Covid-19
- 2020 → SIMO v0 Prototype + customer feedback
- 2021 → SIMO v1
- 2022 → SIMO v2

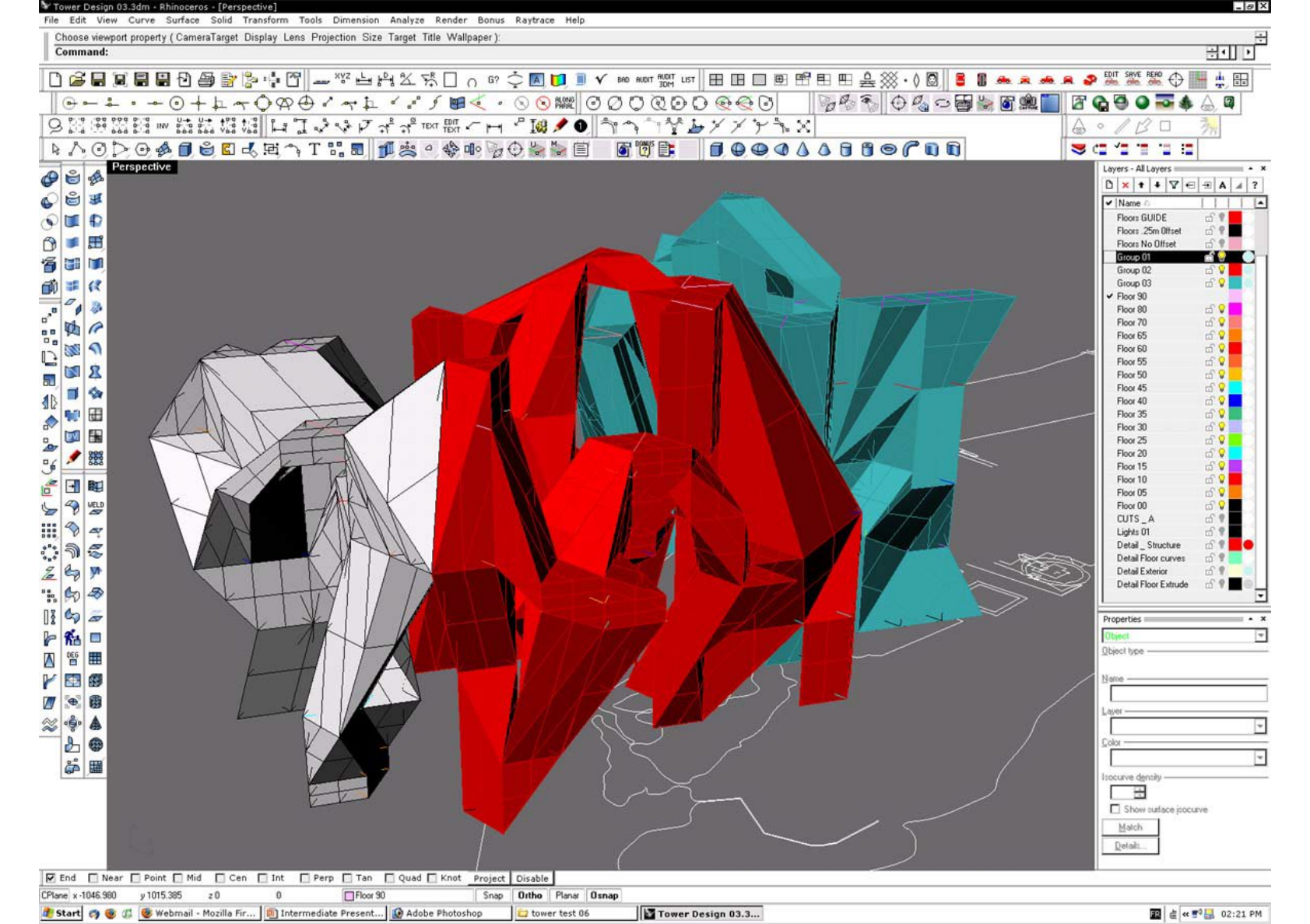
Rhino v3
2005 M.Arch.
Vertical Urbanism

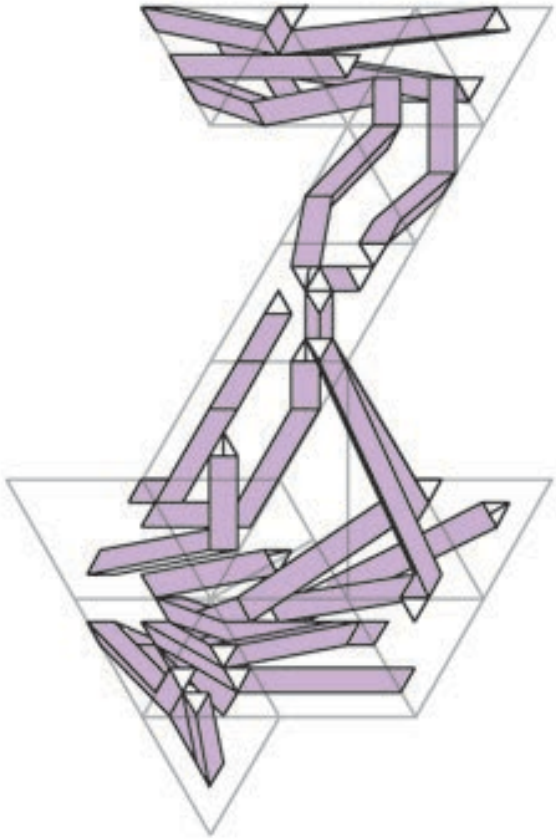

(a)
(b)
(a)
(b)
(c)
(d)
(e)
(f)
(g)
(h)
(c)
(d)
(e)
(f)
(g)
(h)
rule (f) fluctuations
80
60
40
20
0
0
500
1000
1500
2000
rule (g) fluctuations
40
20
0
-20
-40
0
500
1000
1500
2000
(a)
(b)
(c)
(d)
(e)
(f)
(g)
(h)

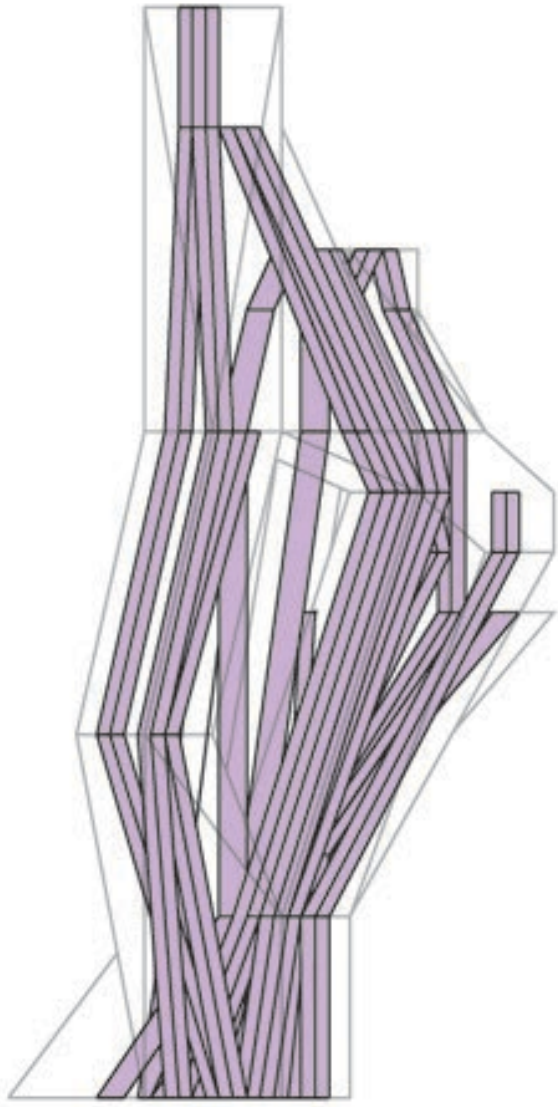


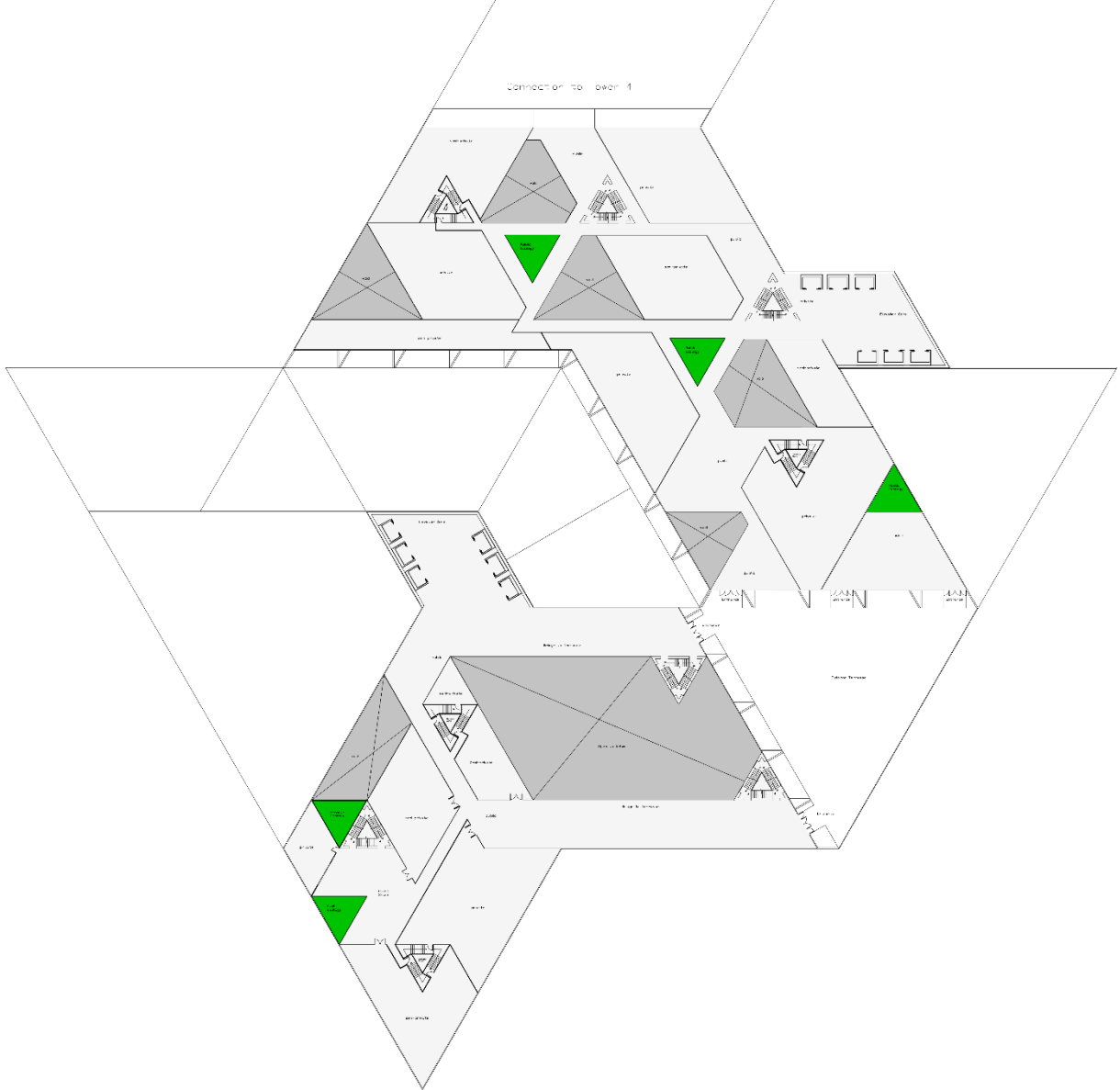

tower test 06-A.3dm - Rhinoceros - [Top]
File Edit View Curve Surface Solid Transform Tools Dimension Analyze Render Bonus Raytrace Help
File successfully written as C:\Documents and Settings\Darrel Ronald\My Documents\M.ARCH-04\03_Concepts\06_Ruled Towers\tower test 06\tower test 06-A.3dm
Command:
Top
Layers - All Layers
Name
Default
Floor 00
Floor 05
Floor 10
Floor 15
Floor 20
Floor 25
Floor 30
Floor 35
Floor 40
Floor 45
Floor 50
Floor 55
Floor 60
Floor 65
Floor 70
Floor 80
Floor 90
CUTS _ A
FLOORS _ A
Glass 00 Floor Caps
Glass 00-20
Glass 20-40
Glass 40-60
Glass 60-80
Glass Hidden
Island Redraw
Island Redraw Ref Point
Properties
Object
Object type
Name
Layer
Color
Isocurve density
Show surface isocurve
Match
Detail...
End Near Point Mid Cen Int Perp Tan Quad Knot Project Disable
CPlane x 680.577 y 2167.215 z 0
Floor 00
Snap Ortho Planar Osnap
Start
Webmail - Mozilla Fir...
Intermediate Present...
Adobe Photoshop
tower test 06
tower test 06-A.3d...
02:15 PM

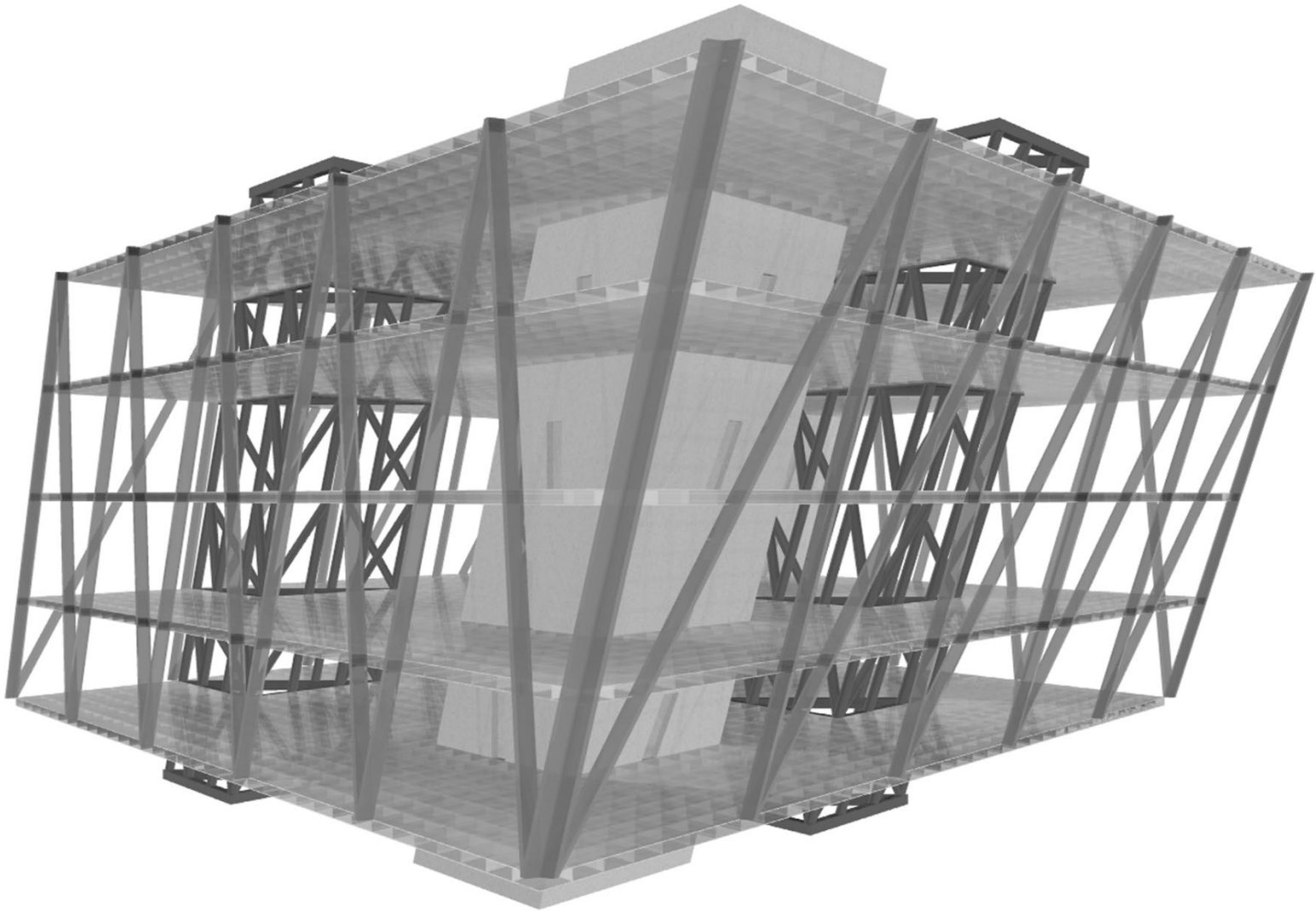

tower test 06-A.3dm - Rhinoceros - [Perspective]
File Edit View Curve Surface Solid Transform Tools Dimension Analyze Render Bonus Raytrace Help
Command: _CloseViewport
Command:
Perspective
Layers - All Layers
Name
Default
Floor 00
Floor 05
Floor 10
Floor 15
Floor 20
Floor 25
Floor 30
Floor 35
Floor 40
Floor 45
Floor 50
Floor 55
Floor 60
Floor 65
Floor 70
Floor 80
Floor 90
CUTS _ A
FLOORS _ A
Glass 00 Floor Caps
Glass 00-20
Glass 20-40
Glass 40-60
Glass 60-80
Glass Hidden
Island Redraw
Island Redraw Ref Point
Properties
Object
Object type
Name
Layer
Color
Isocurve density
Show surface isocurve
Match
Detail...
End Near Point Mid Cen Int Perp Tan Quad Knot Project Disable
CPlane x -639.698 y 1179.482 z 0
Floor 00
Snap Ortho Planar Osnap
Start
Webmail - Mozilla Fir...
Intermediate Present...
Adobe Photoshop
tower test 06
tower test 06-A.3d...
02:17 PM


Tower Design 03.3dm - Rhinoceros - [Perspective]
File Edit View Curve Surface Solid Transform Tools Dimension Analyze Render Bonus Raytrace Help
Choose viewport property (CameraTarget Display Lens Projection Size Target Title Wallpaper):
Command:
Perspective
Layers - All Layers
Name
Floors GUIDE
Floors .25m Offset
Floors No Offset
Group 01
Group 02
Group 03
Floor 90
Floor 80
Floor 70
Floor 65
Floor 60
Floor 55
Floor 50
Floor 45
Floor 40
Floor 35
Floor 30
Floor 25
Floor 20
Floor 15
Floor 10
Floor 05
Floor 00
CUTS _ A
Lights 01
Detail _ Structure
Detail Floor curves
Detail Exterior
Detail Floor Extrude
Properties
Object
Object type
Name
Layer
Color
Isocurve density
Show surface isocurve
Match
Detail...
End Near Point Mid Cen Int Perp Tan Quad Knot Project Disable
CPlane x -1046.980 y 1015.385 z 0 0 Floor 90 Snap Ortho Planar Osnap
Start
Webmail - Mozilla Fir...
Intermediate Present...
Adobe Photoshop
tower test 06
Tower Design 03.3...
02:21 PM

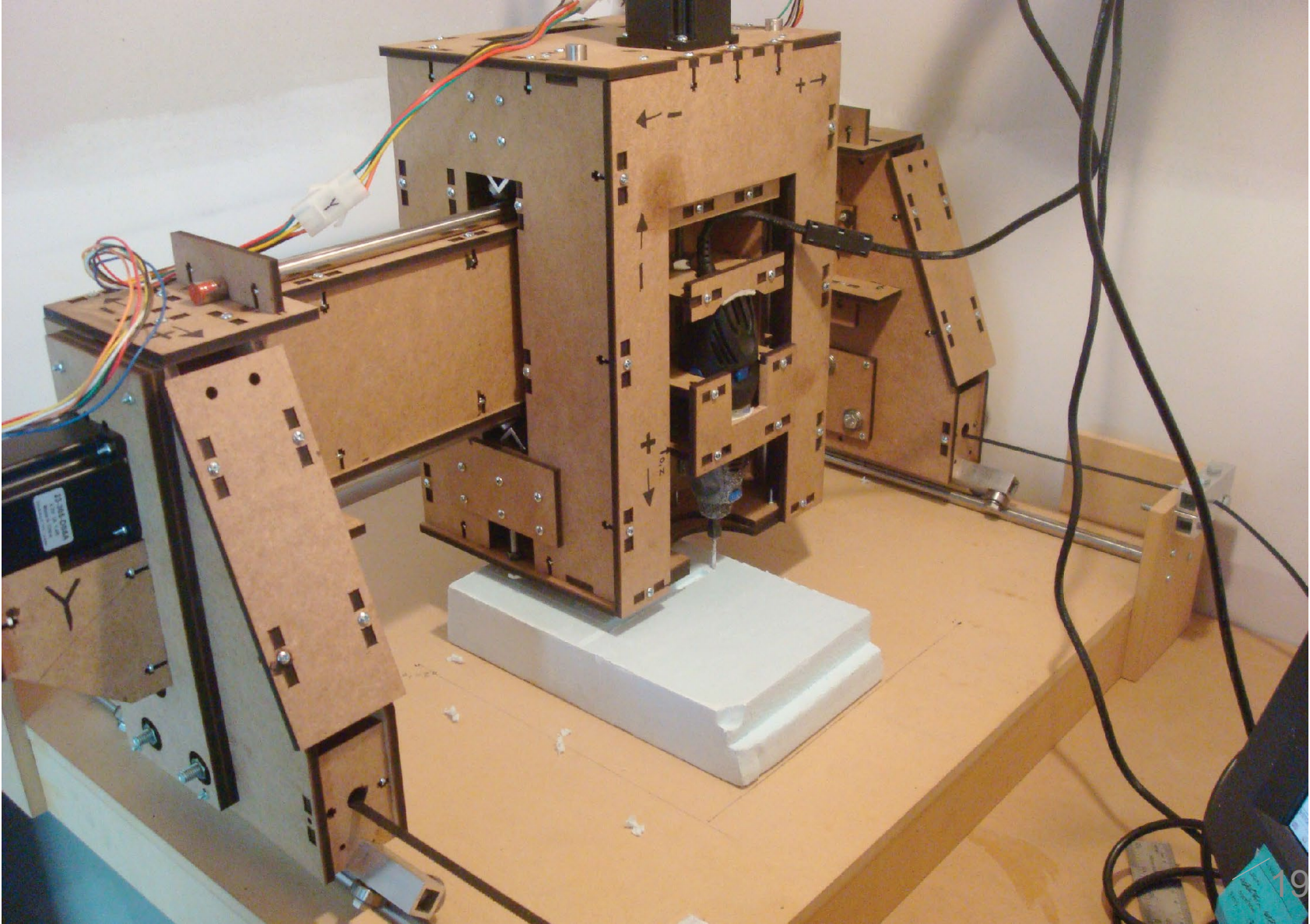


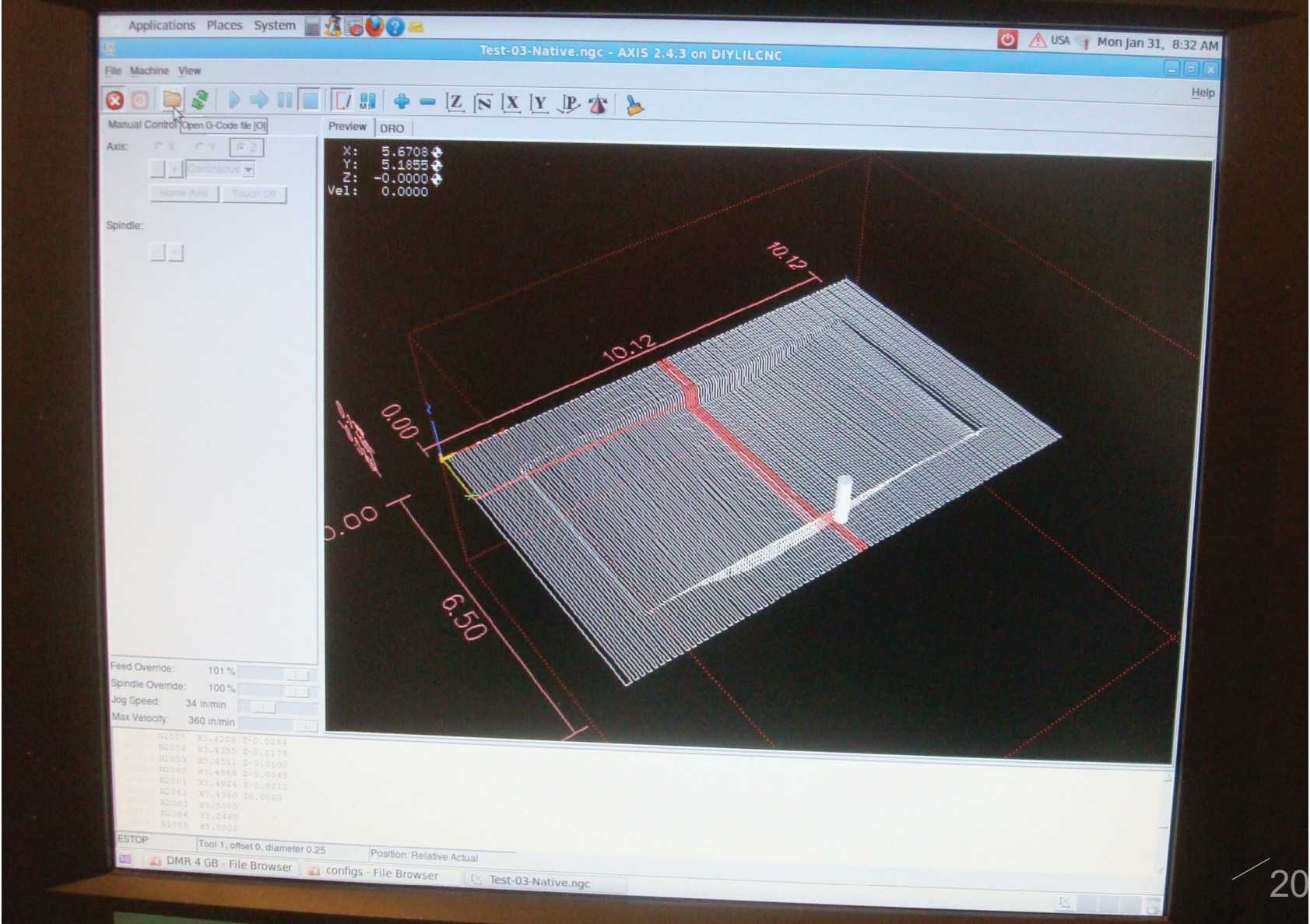


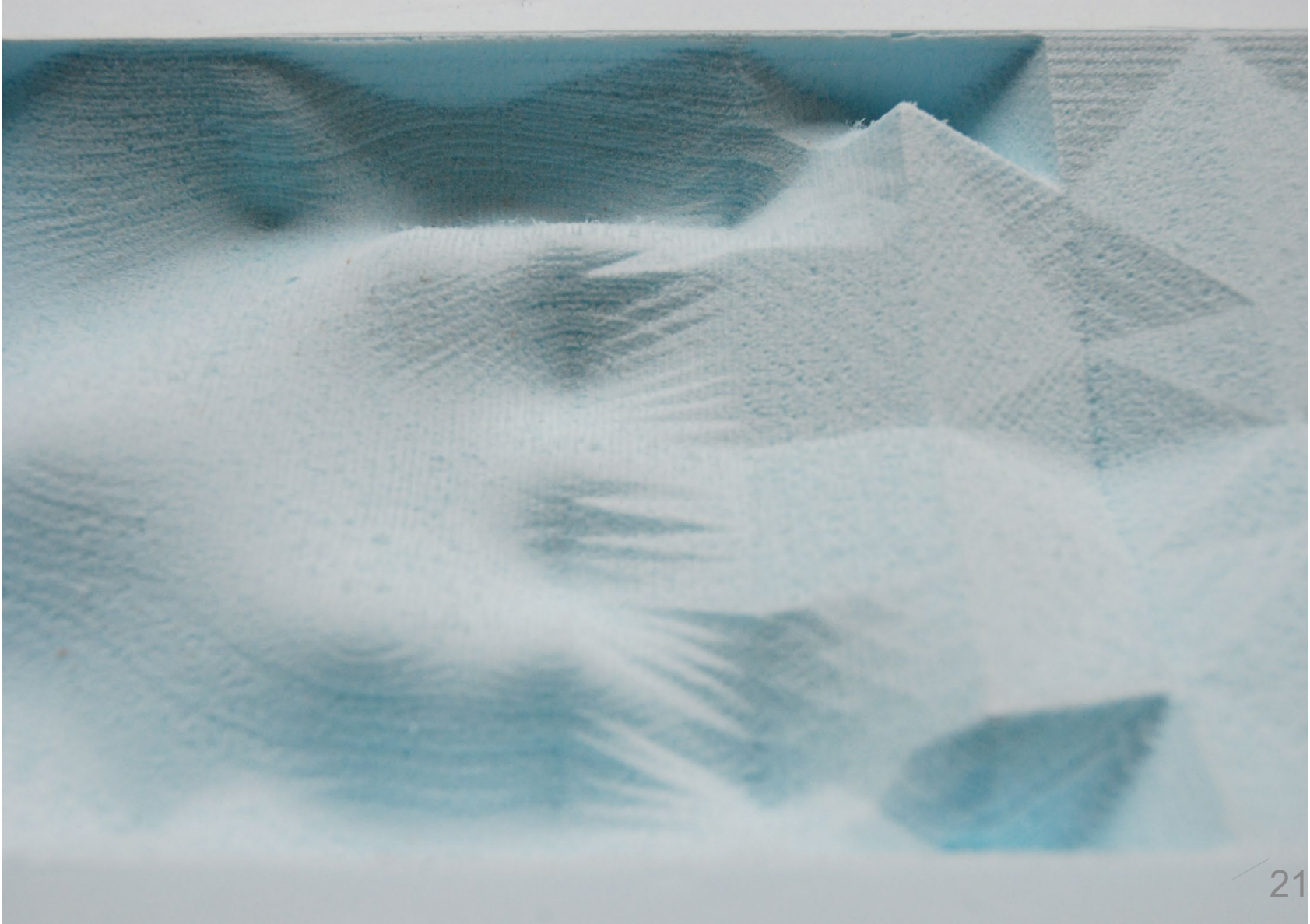


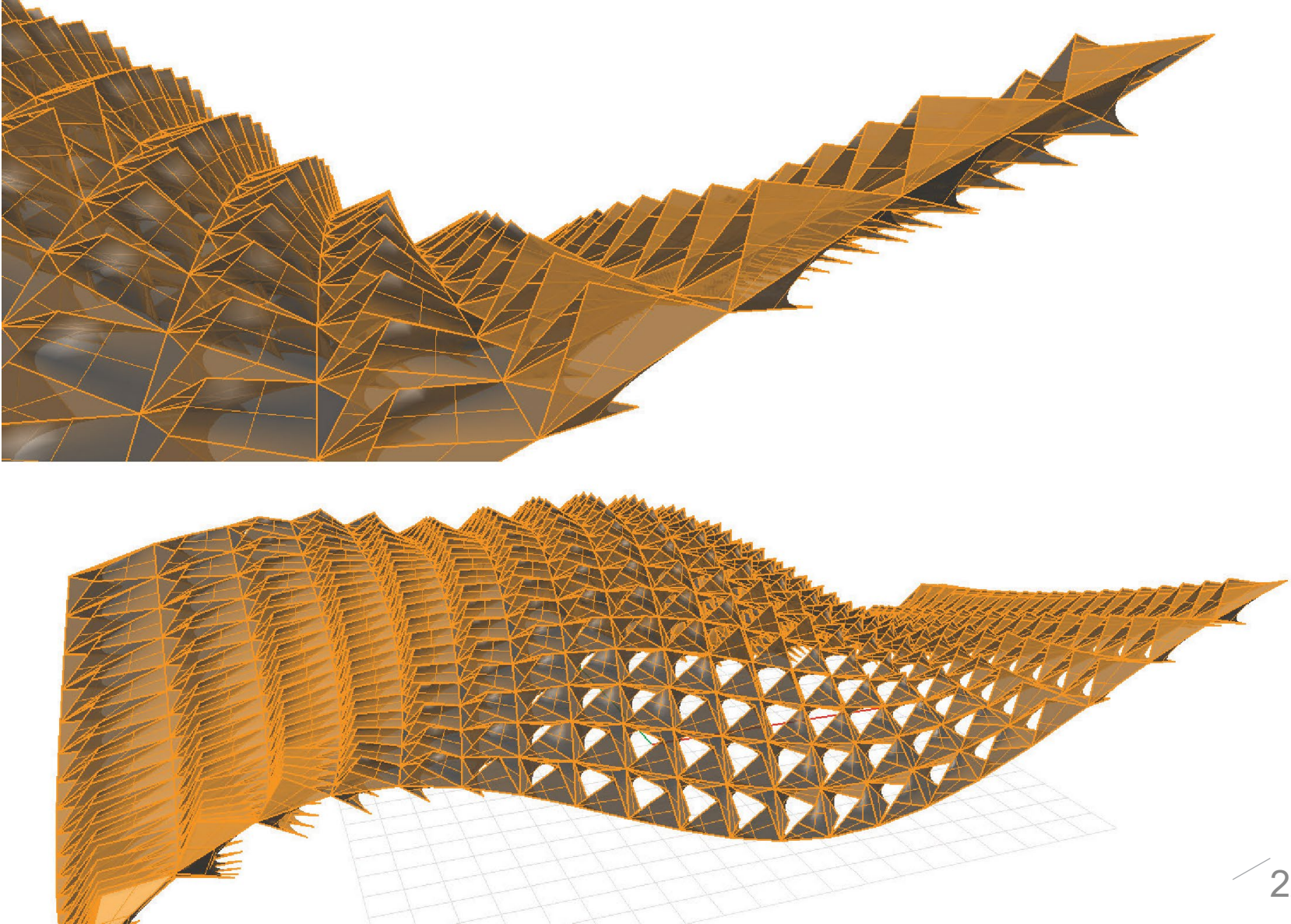


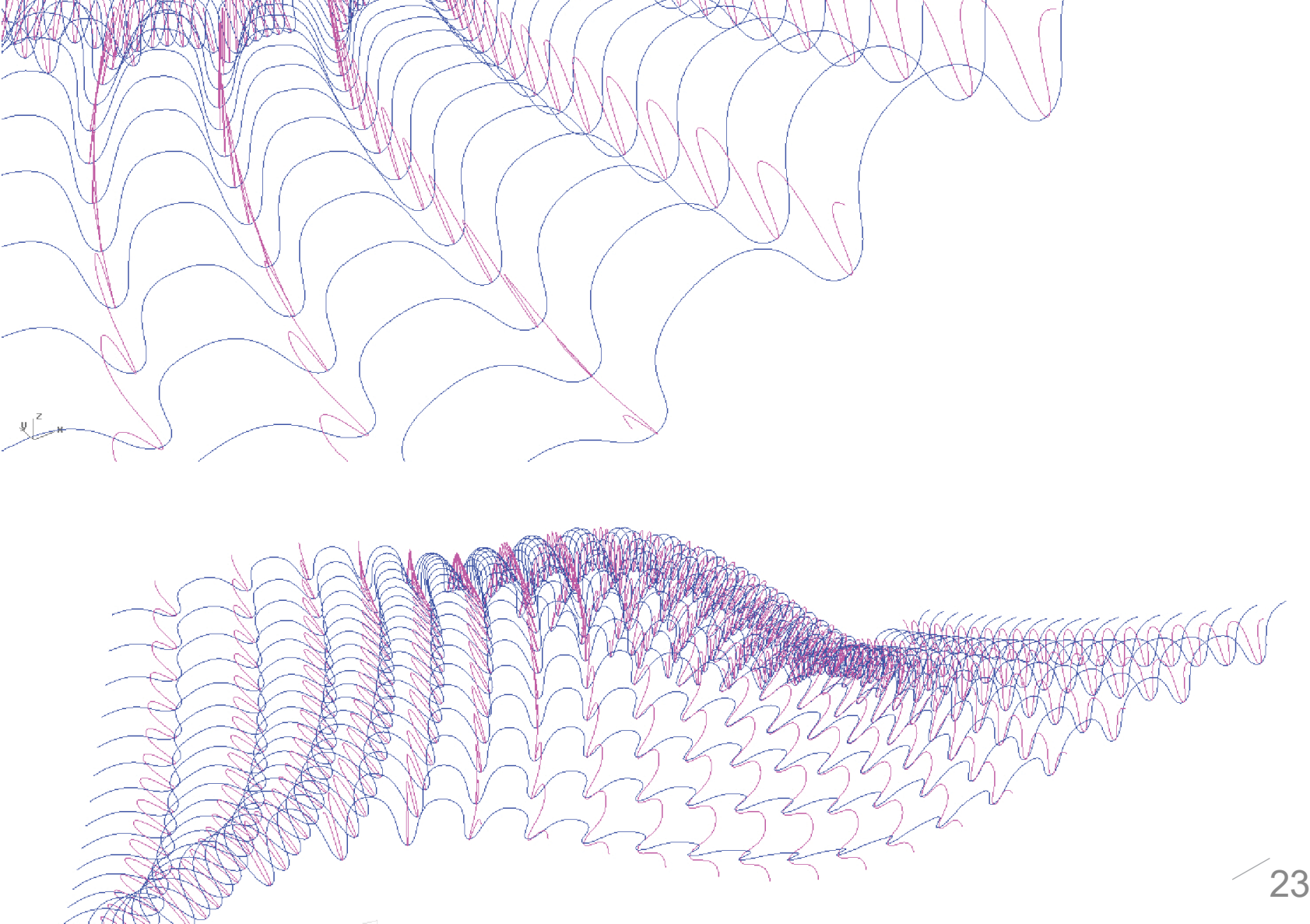
Rhino & GH
Open Form Architecture

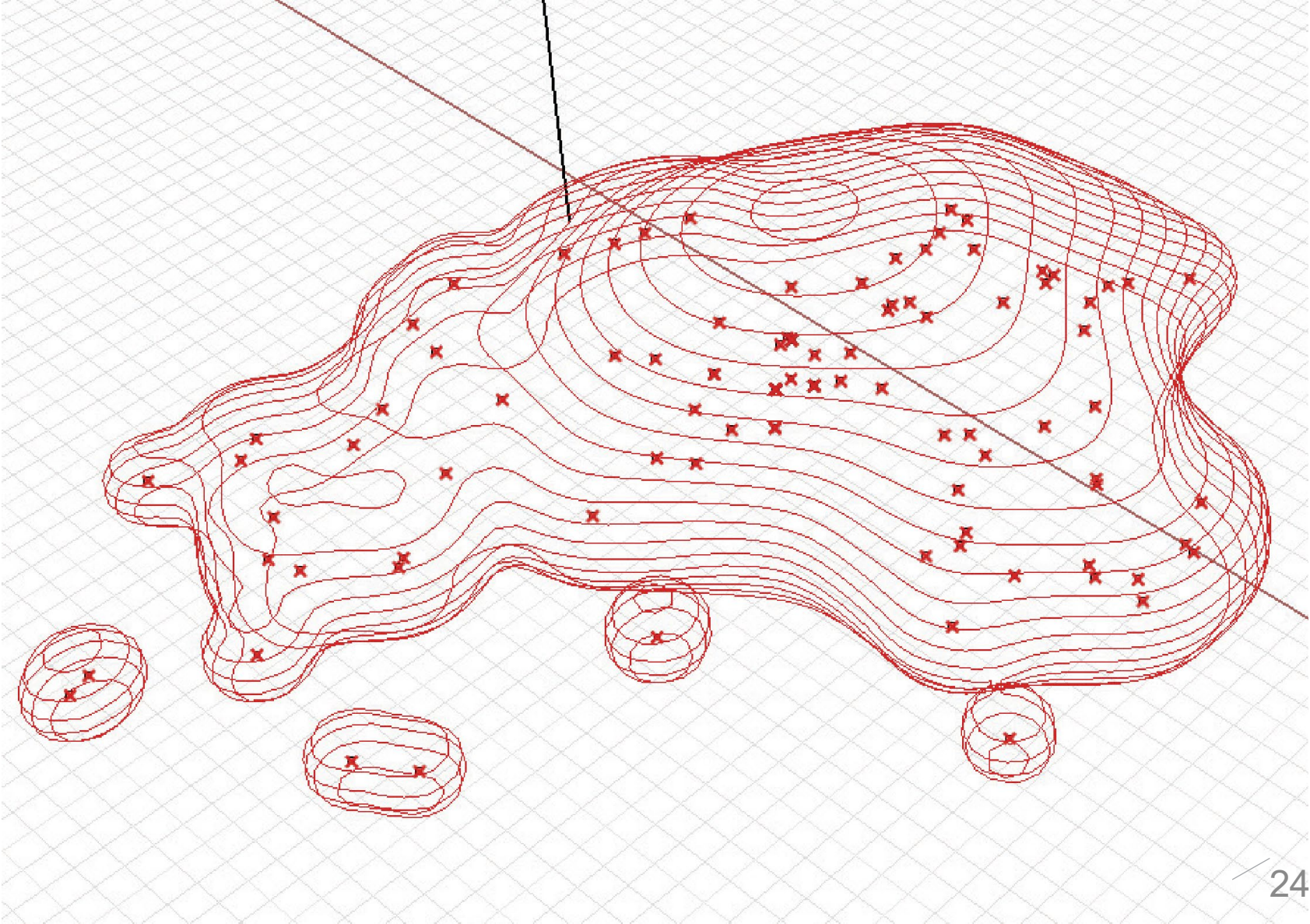

Y
Z-Axis

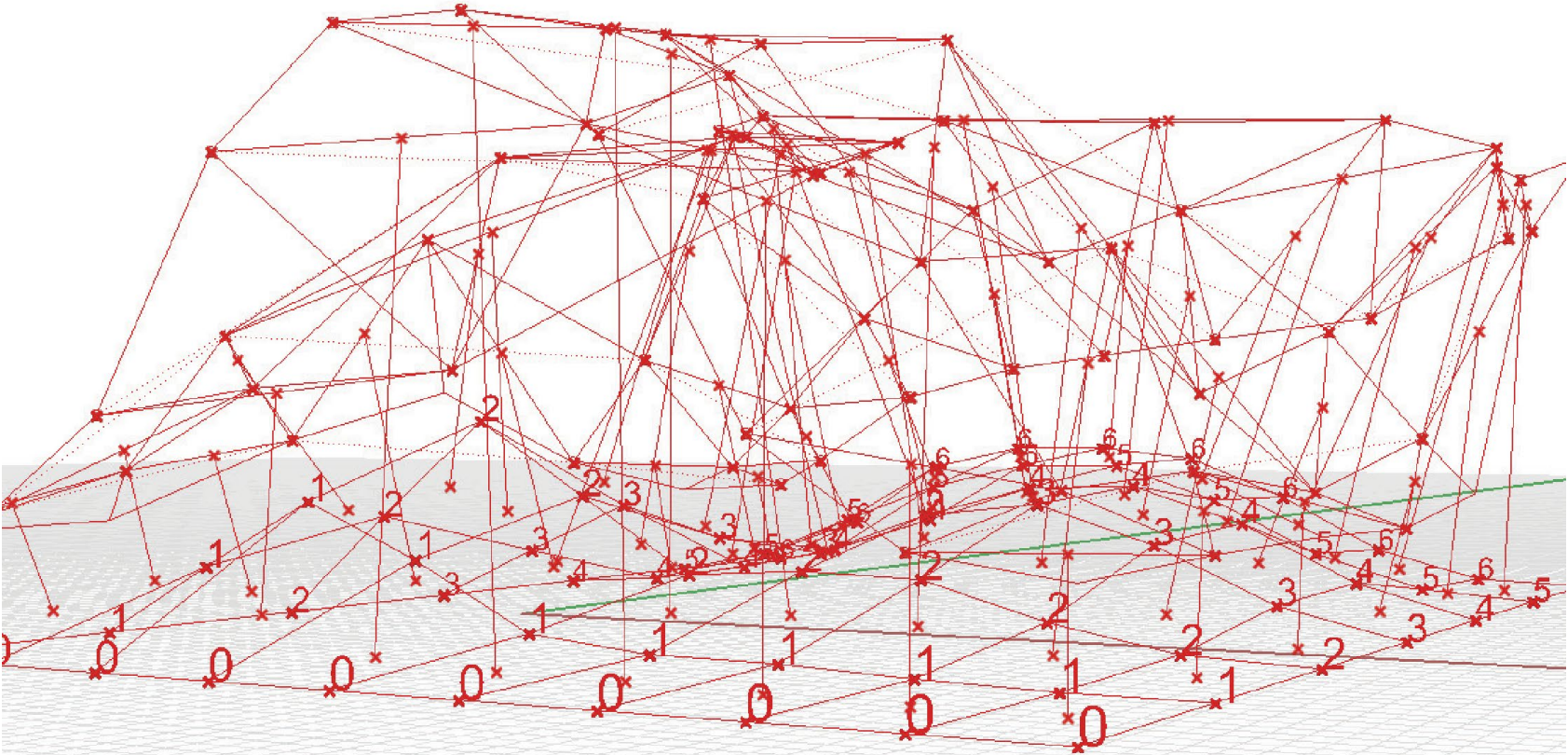

Applications Places System
Test-03-Native.ngc - AXIS 2.4.3 on DIYLILCNC
File Machine View
Help
Manual Control
Open G-Code file [O]
Axis:
Continuous
Home Axis
Touch Off
Spindle:
Preview
DRO
X: 5.6708
Y: 5.1855
Z: -0.0000
Vel: 0.0000
10.12
10.12
0.00
0.00
6.50
Feed Override: 101 %
Spindle Override: 100 %
Jog Speed: 34 in/min
Max Velocity: 360 in/min
ESTOP
Tool 1, offset 0, diameter 0.25
Position: Relative Actual
DMR 4 GB - File Browser
configs - File Browser
Test-03-Native.ngc



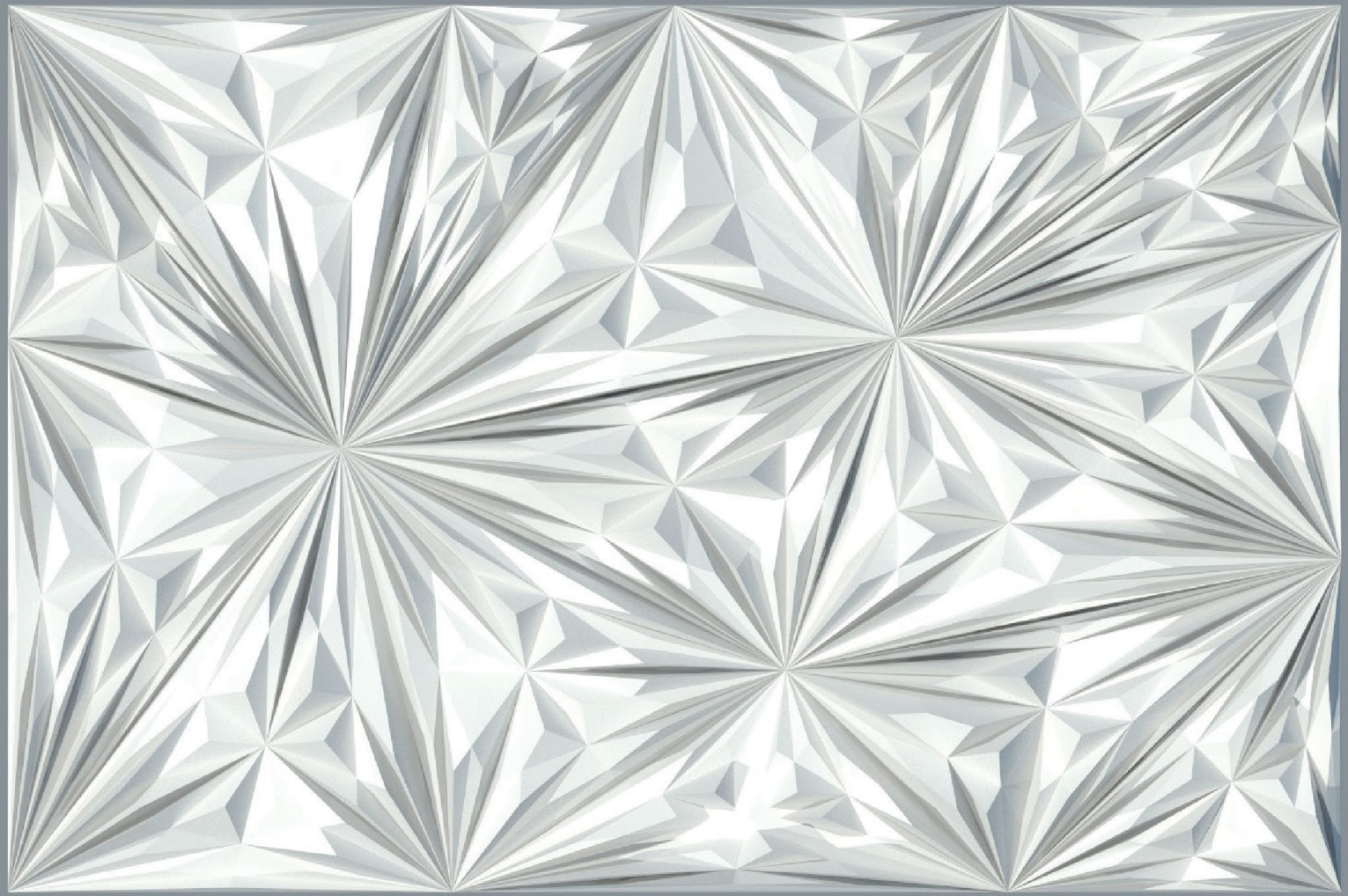


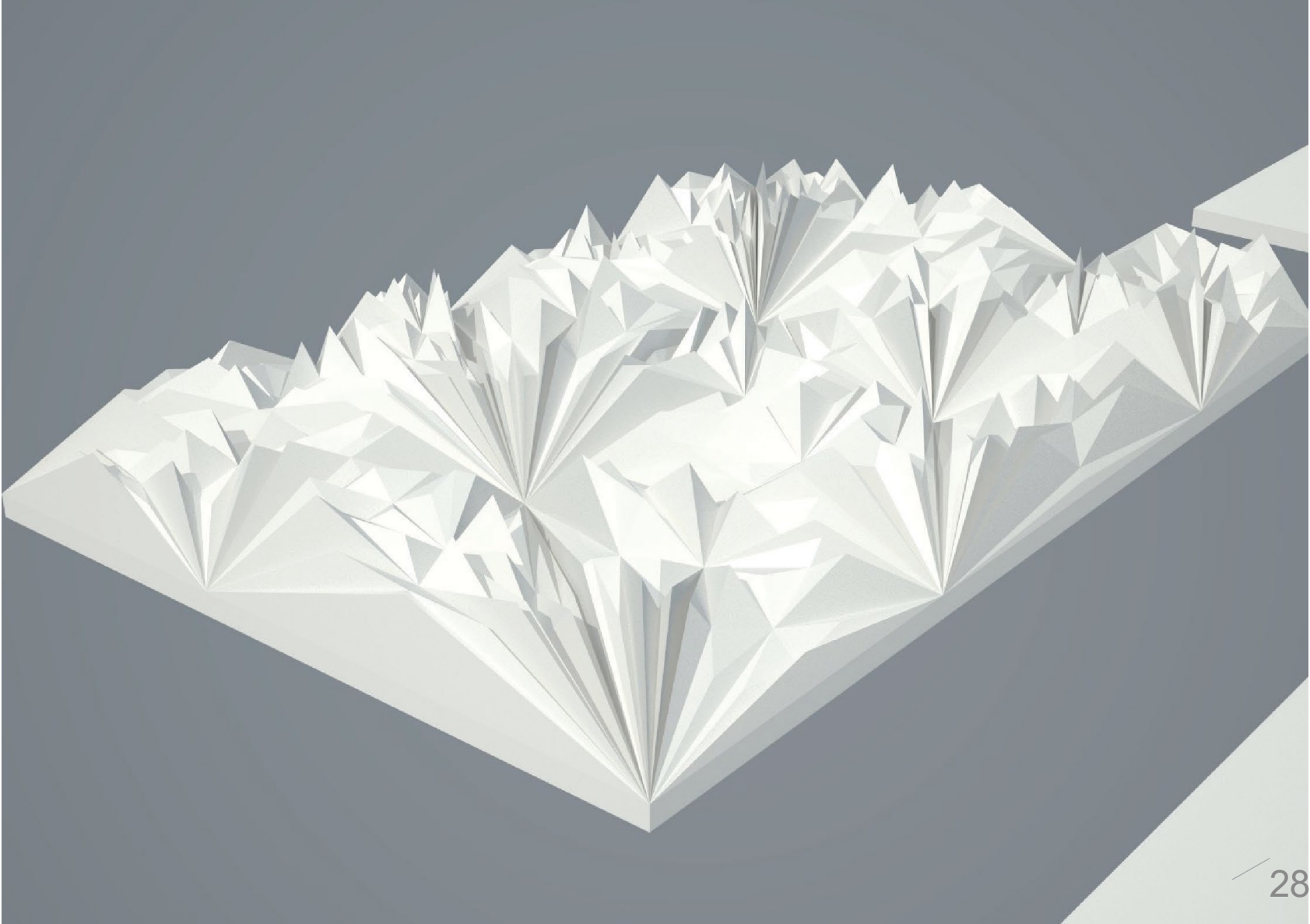


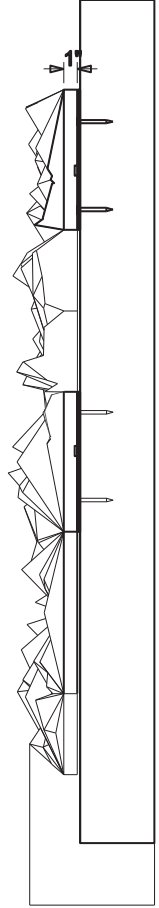




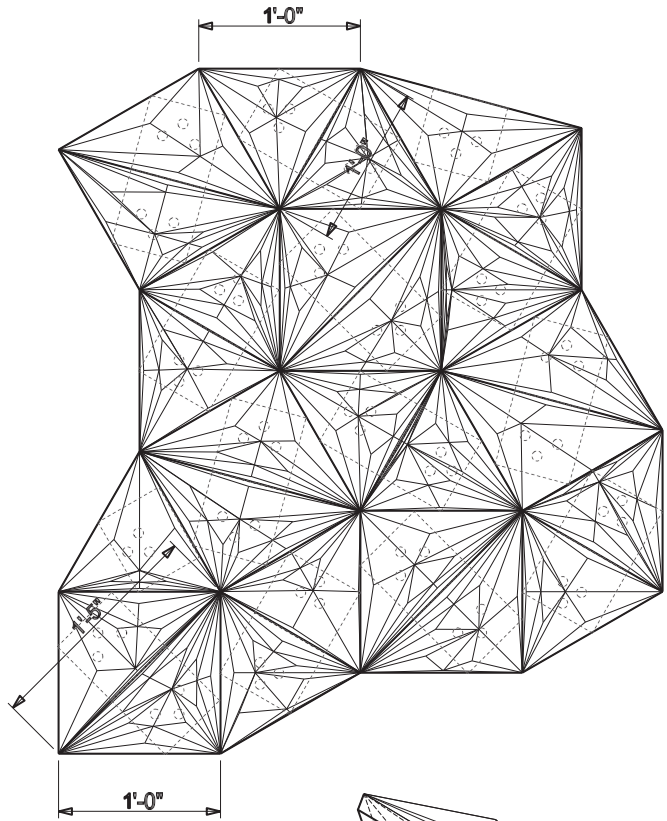
```
Option explicit
‘Script written by <Darrel Ronald>
‘Script copyrighted by <Open Form Architecture>
‘Script version wednesday, October 21, 2009 Call
Main()
Sub Main()
        Dim aP1, aP2, aP3, gen
        Call  rhino.enableRedraw(False)  ‘’
        Get Points
        aP1  =  rhino.getpoint(“pick  Pt1”)
        aP2  =  rhino.getpoint(“pick  Pt2”)
        aP3  =  rhino.getpoint(“pick  Pt3”)
        gen = 0
        Call  recursiveCrack(aP1,  aP2,  aP3,  gen)
        Call rhino.enableRedraw(True)
end Sub
Function  recursiveCrack(aP1,  aP2,  aP3,  gen)
        Dim aCentro
        Dim sl1, sl2, sl3, srf1, srf2, srf3
        Dim genmax : genMax = 3
        aCentro = midPtPonderate(aP1, aP2, aP3, 1, 1, 1)


‘       ‘’ lINeS
‘       sl1  =  rhino.Addline(ap1,aCentro)  ‘
sl2    =    rhino.Addline(ap2,aCentro)    ‘
sl3 = rhino.Addline(ap3,aCentro)
‘       Call rhino.ObjectColor(array(sl1,sl2,sl3), rgb
        (255/genmax * gen, 0, 255 - 255/genmax * gen))

        ‘’SURFACeS
        srf1 = rhino.AddSrfPt(array(ap1,ap2,aCentro)) srf2
        = rhino.AddSrfPt(array(ap2,ap3,aCentro)) srf3 =
        rhino.AddSrfPt(array(ap3,ap1,aCentro)) ‘Call
        rhino.ObjectColor(array(srf1,srf2,srf3), rgb
        (255/genmax * gen, 0, 255 - 255/genmax * gen)) If
        gen < genMax Then
                Call  recursiveCrack(aP1,  aP2,  aCentro,  gen+1)
                Call  recursiveCrack(aP2,  aP3,  aCentro,  gen+1)
                Call recursiveCrack(aP3, aP1, aCentro, gen+1)
        end    If
end Function
Function   midPtPonderate(aP1,   aP2,
aP3, load1, load2, load3)
        midPtPonderate = Null
        midPtPonderate  =  array((aP1(0)  *  load1  +  aP2(0)  *
load2 + aP3(0) * load3) / (load1 + load2 + load3),_
                (aP1(1) * load1 + aP2(1) * load2 +
aP3(1) * load3) / (load1 + load2 + load3),_
                (aP1(2) * load1 + aP2(2) * load2 + aP3(2)
* load3) / (load1 + load2 + load3) + Rnd*200)
end Function
```

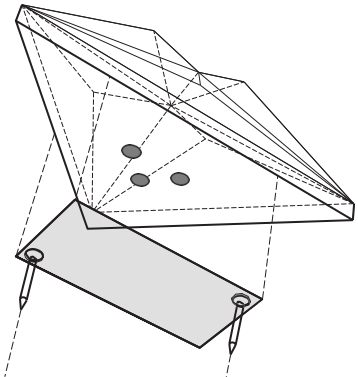


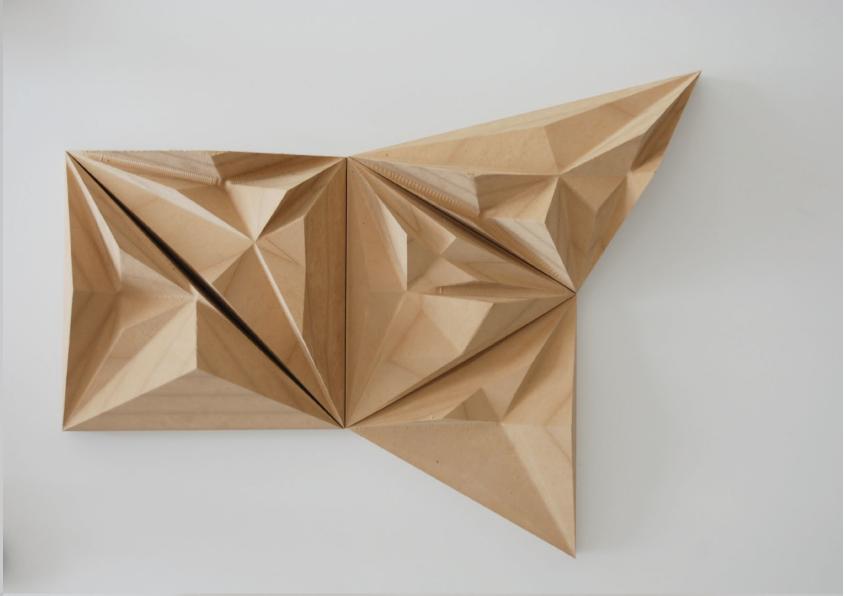




- tuiles triangulaires mdf (~5-10lbs)
- aimants enfoncés (3/4'' x 3/16'' ?)
- plaques d'acier visé
- mur de gypse







Urban Development Challenge

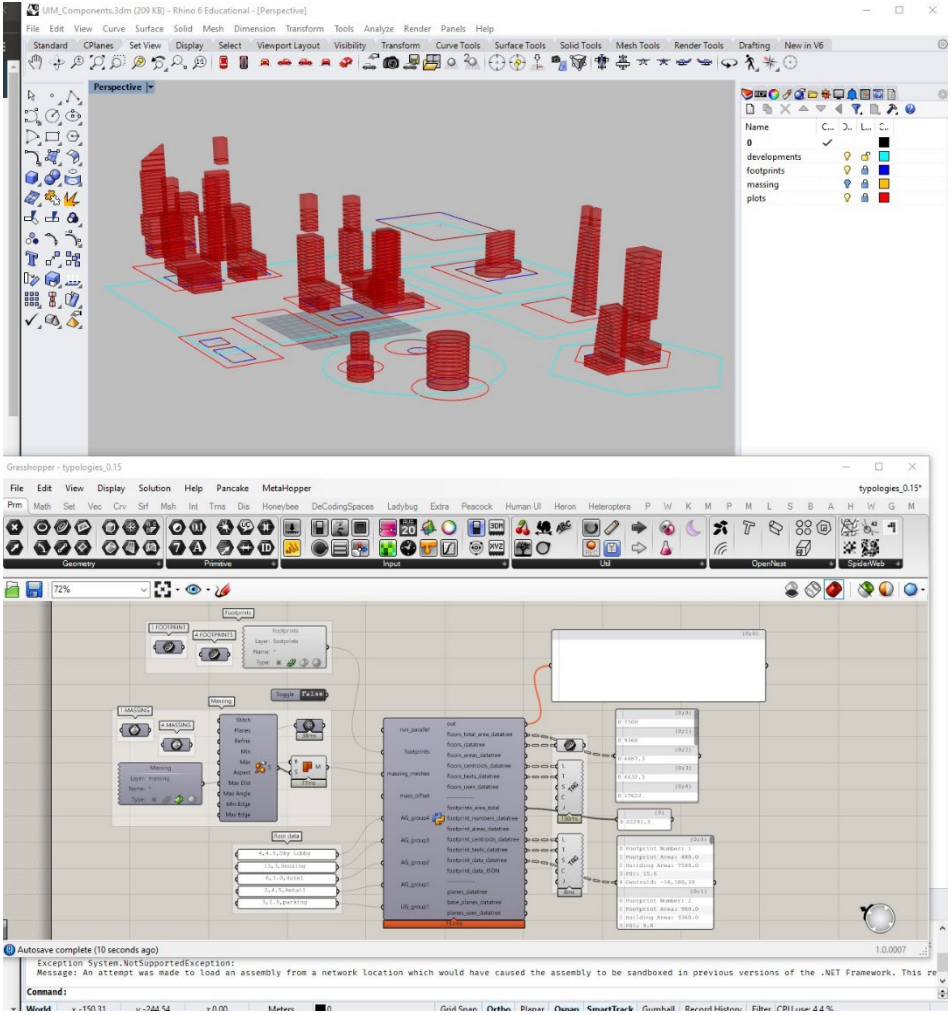
Jurong Lake District, Singapore

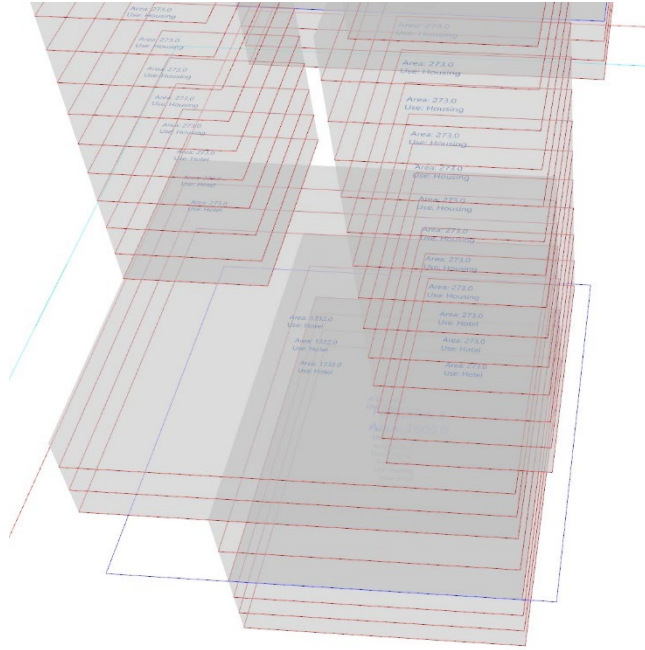
The Future of Urban Development

- Integrated Modelling and Information
- Multi-disciplinary Collaboration
- Data challenges
- Scale challenges
- Speed and experimentation
- Client requirements for BIM (without standards)

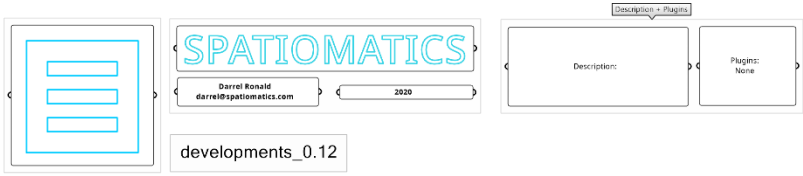
- Solving multiple crises
    - Sustainability
    - Circularity
    - Energy
    - Mobility
    - Water and Utilities

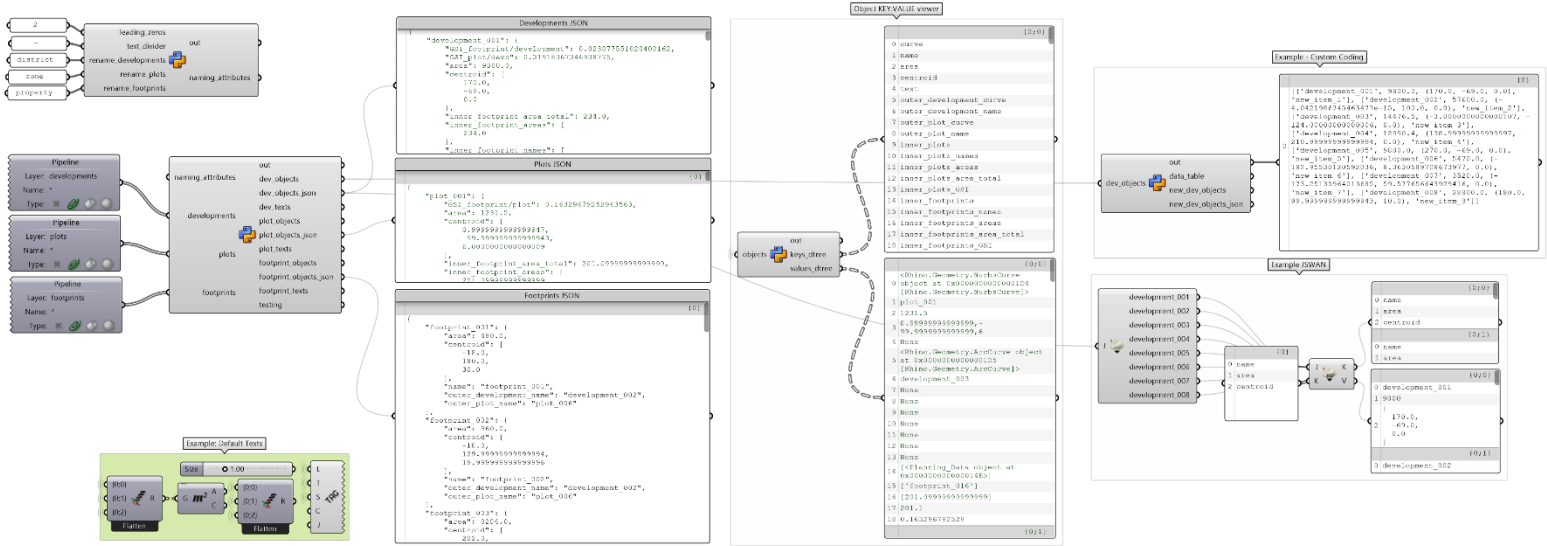
SIMO Protoype





SIMO Protoype





Do it together!

Urban Tech Stack Alliance (UTSA)

Urban Development Challenge

The planet is rapidly urbanizing, but the tools we need for urban development sit in a technology gap.

Geomatics = GIS	Urban Development = ?	Buildings = BIM
Working Group • Open Geospatial Consortium (OGC)	**Working Group** • None • Initiative "Integrated Digital Built Environment" (IDBE) by OGC + bSI • Various Urban Digital Twin initiatives	**Working Group** • Building Smart International (bSI)
Software • QGIS • ArcGIS	**Software** • No Industry Standard • CityCAD • Various Web Interfaces	**Software** • Autocad / Bricscad • Revit • Vectorworks • Rhino • ArchiCAD • ++
Data Models • CityGML + Extension • CityJSON + Extension • LandInfra • ++	**Data Models** • None? • Potentially extend CityJSON	**Data Models** • Industry Foundation Classes (IFC) only exchange of data + bSDD + MVD • BIM Collaboration Format (BCF)

Six Challenges for 3D City Modelling (GIS Perspective)

Jantien Stoter, 2020.04.09 - GIM International
The article is a collaboration of 7 TU Delft 3D Geomatics team members

1. Consistency Between Models
2. Standardization
3. Data Quality
4. Data Interoperability
5. Data Maintenance / Governance
6. From Utopian Pilots to Real-World Use Cases

Mission

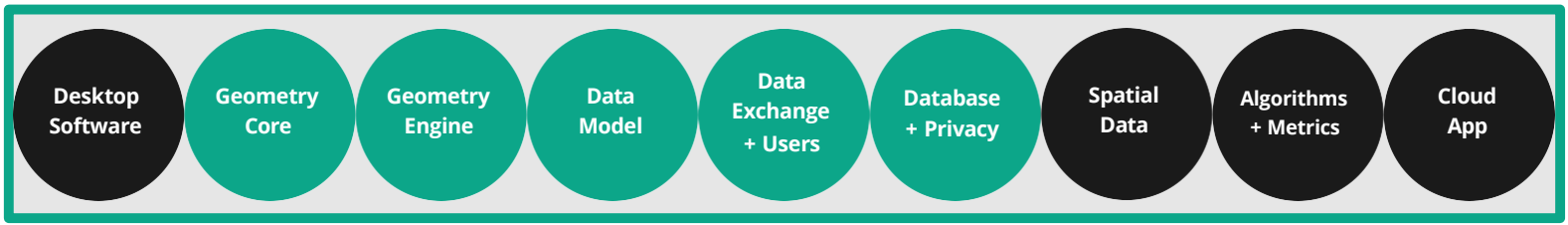
Help each other build products that solve the complex and integrated urban development challenges. Innovate through **Integration** and **standardization**. Create a **new market** and grow together with **interoperable services**.

Goals

1. Define: the Missing Technologies
2. Combine: Expertise and Solutions
3. Connect: existing Standards, Technologies and Partners
4. Build: New Partnerships
5. Assemble: a community Urban Data Model
6. Build: Foundation Technologies
7. Build: Interoperable Products

Full Stack Technologies

APIs



Tech Options
1. QT (C++, Py, JS)
2. Electron (JS)
3. .NET (C#)
4. UNO (C#)

Tech Options
1. Compas (Py)
2. Ladybug (Py)
3. Speckle 1.0 (C#)
4. Speckle 2.0 (C#,Py)
5. BHoM (C#)
6. Rhino3DM (JS, Py, C#)
7. glTF (JS)
8. openNURBS (C#)
9. ...

Tech Options
1. Compas (Py)
2. BHoM (C#) [limited] **?**
3. CGAL **?**
4. GDAL **?**
5. Rhino3DM (Py, C#, JS) [limited] **?**
6. ...

Schema Options
1. GML
2. JSON
3. CityGML
4. IFC
5. LandInfra
6. Speckle flexible
7. W3 Spatial Thing?
8. I3S
9. Custom UIM

Tech Options
1. API (REST)
2. API (GraphQL)
3. API (Websockets)
4. RPC
5. Speckle
6. database file
7. file type
8. LUCI

Generic Structures
1. Dictionary
2. Array
3. NDArray
4. Graph

Filetype Options
1. JSON
2. GML
3. uimJSON
4. CityJSON (3D)
5. CityGML (3D)
6. GeoJSON (2D)
7. i3S
8. gLTF

Tech Option
1. Graph
2. NoSQL
3. PostgreSQL
4. PostGIS
5. SQL
6. in software

Local /or/ Cloud
1. Neo4J
2. MongoDB
3. PostgreSQL
4. PostGIS
5. SQLite
6. SQL
7. GeoDjango
8. GeoServer
9. Apollo Data Graph

Data Integration
1. Airbyte
2. GeoKettle
3. Mapped (IoT)
4. Apache Spark
5. Apache Superset

Tech Option
1. Open Data
2. Government
3. Proprietary
4. Real Estate
5. Wikipedia
6. News
7. ...

Local / Cloud
1. many options...
2. OSM
3. Esri
4. Mapbox
5. GeoNode

Examples
- GIS
- IFC
- FIBREE UOI
- DiS Geo

Domain Algorithms
- Open Licensing
- Proprietary
- ...

Domain Metrics
- TOD
- LEED
- BREEAM
- WELL
- SDG
- ...

Tech Option
1. Vue
2. React
3. ...

Other Components
- Cloud Provider
- API
- Admin
- Authentication
- Mapping
- Data Storage
- Data Visualization
- Data Editing
- ML

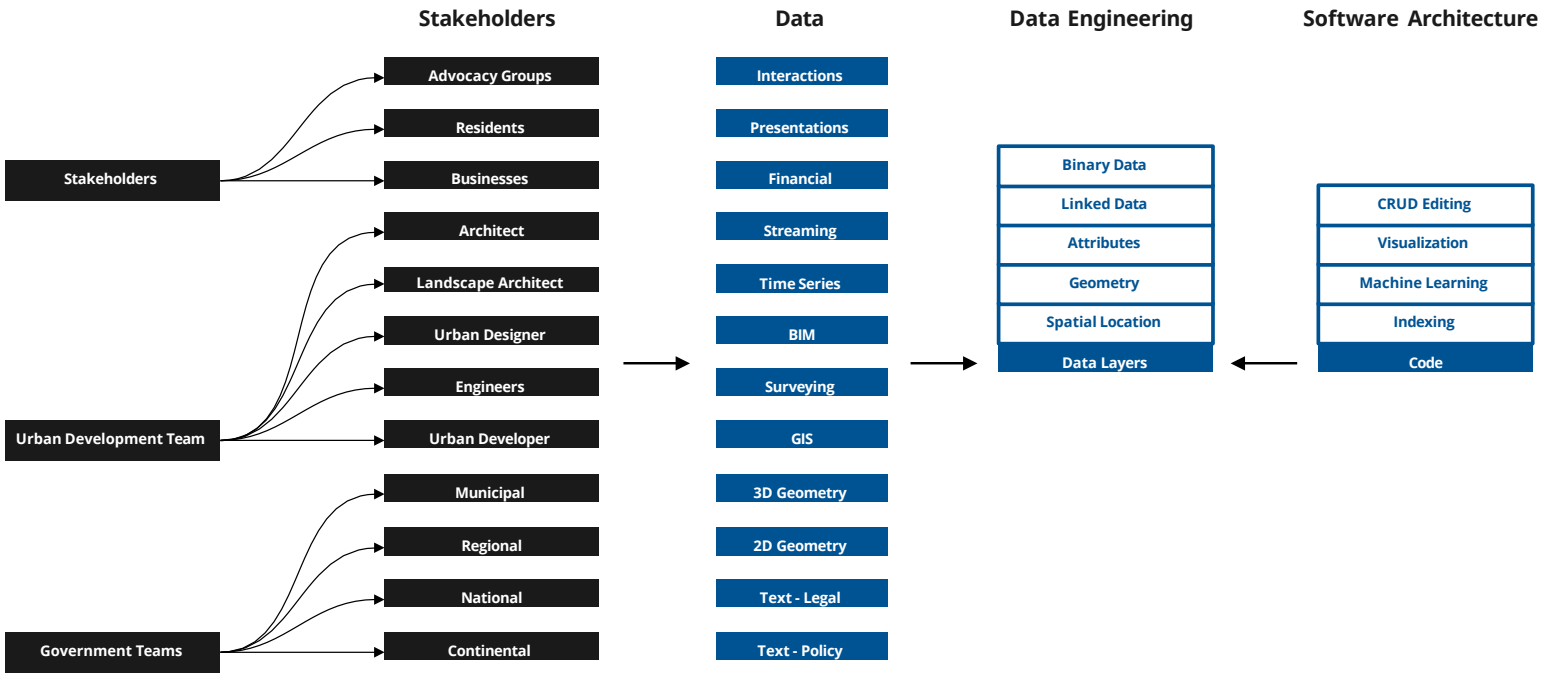
Industry Software Plugins

Tech Options
1. QGIS
2. Esri
3. Autodesk
4. McNeel
5. Bricsys
6. Trimble
7. Adobe
8. Microsoft
9. other...

Open Licensing | Mixed Licensing

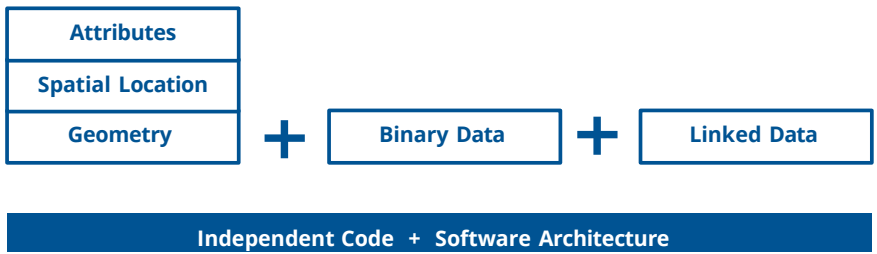
Data Infrastructure?

Urban Data Complexity

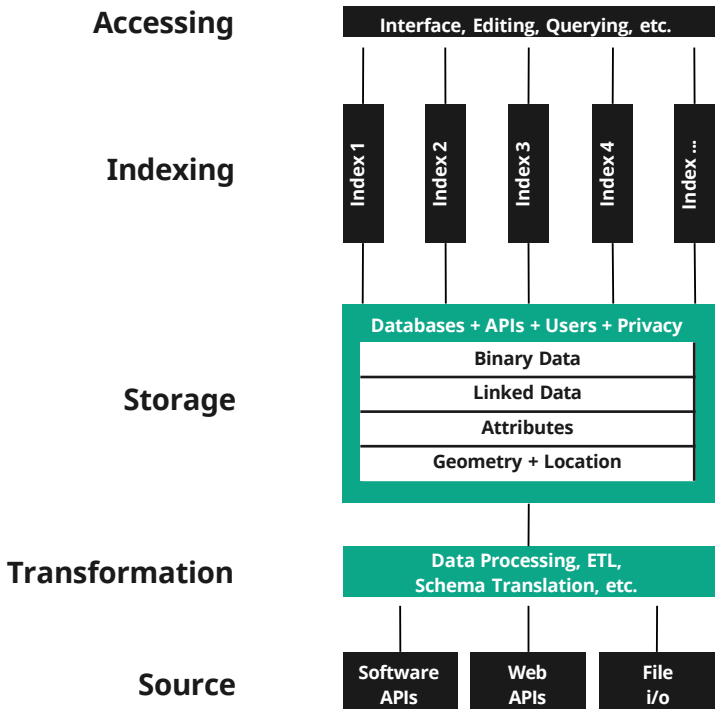


Unlimited Stakeholders and Data Types

Data Strategy - Seek Simplicity



Data Infrastructure



Urban Information Model (UIM)

Just do it yourself,
You can’t wait a decade.

Urban Tech Stack Challenges

Ask about the “Urban Tech Stack Alliance”

Built environment data standards and their integration: an analysis of IFC, CityGML and LandInfra

Image credit: Bioregional

Version 1.0 • 02 March 2020
OGC Document 19-091r1
bSI TR1012

OGC® Making location count. buildingSMART International

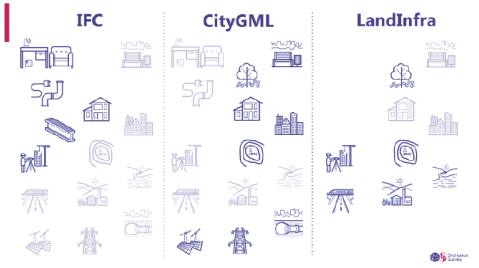


Figure 1 - Thematic coverage of the three data standards. Dark shading indicates strong coverage, light shading weaker coverage (or under development) and no shading implies no known coverage. The object icons are ordered approximately; the intention is to cluster by theme and represent finer spatial scales nearer the top.

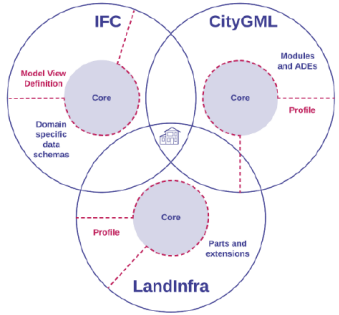


Figure 2 – The three standards differ but overlap in their thematic coverage; as an example, the concept of a building is common to all three. IFC is always subset into Model View Definitions (MVDs) for implementation, whereas the GML-based standards of CityGML and (the InfraGML implementation of) LandInfra can be subset optionally into profiles; for all three, the core of the schema must be implemented. Application Domain Extensions (ADEs) enable anyone to extend the CityGML standard to accommodate more specialist themes;

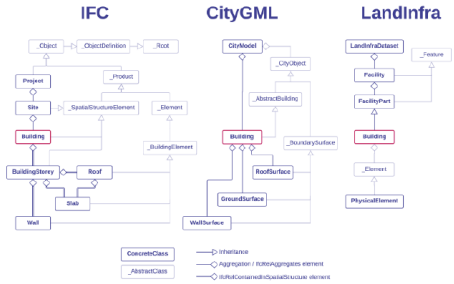


Figure 3 - The concept of a building is represented by all three standards but the detailed structure of the conceptualisation is dissimilar (geometries are not represented in this figure). The UML-like diagrams show possible representations of a very simple building; they are derived from example building instance models that are valid in each of the three standards. The IFC representation is valid for IFC4; in IFC5, a building is a predefined type of facility and a building storey is a specialisation of a facility part.

Context

- Slow to develop (ex. CityGML 3.0 is 9+ years in the making
- Huge areas of domain knowledge not included
- Difficult to extend
- Complicated to create export files

Therefore

- Why do we even need the existing schemas and file types?
- Why can't we just use an extensible JSON data object?
- Decouple Code from the Data Model - Data Oriented Design
- Radically simplify the Data Model
- Complete Data Model pulling from existing standards
- Fast development (like web) with "Worse is better" approach

OGC Paper (link)

Global to City Scale
GIS File Formats
Geometry Limits & Distortion
Non-interoperability CAD
Slow Standardization Process
Focus: Geospatial

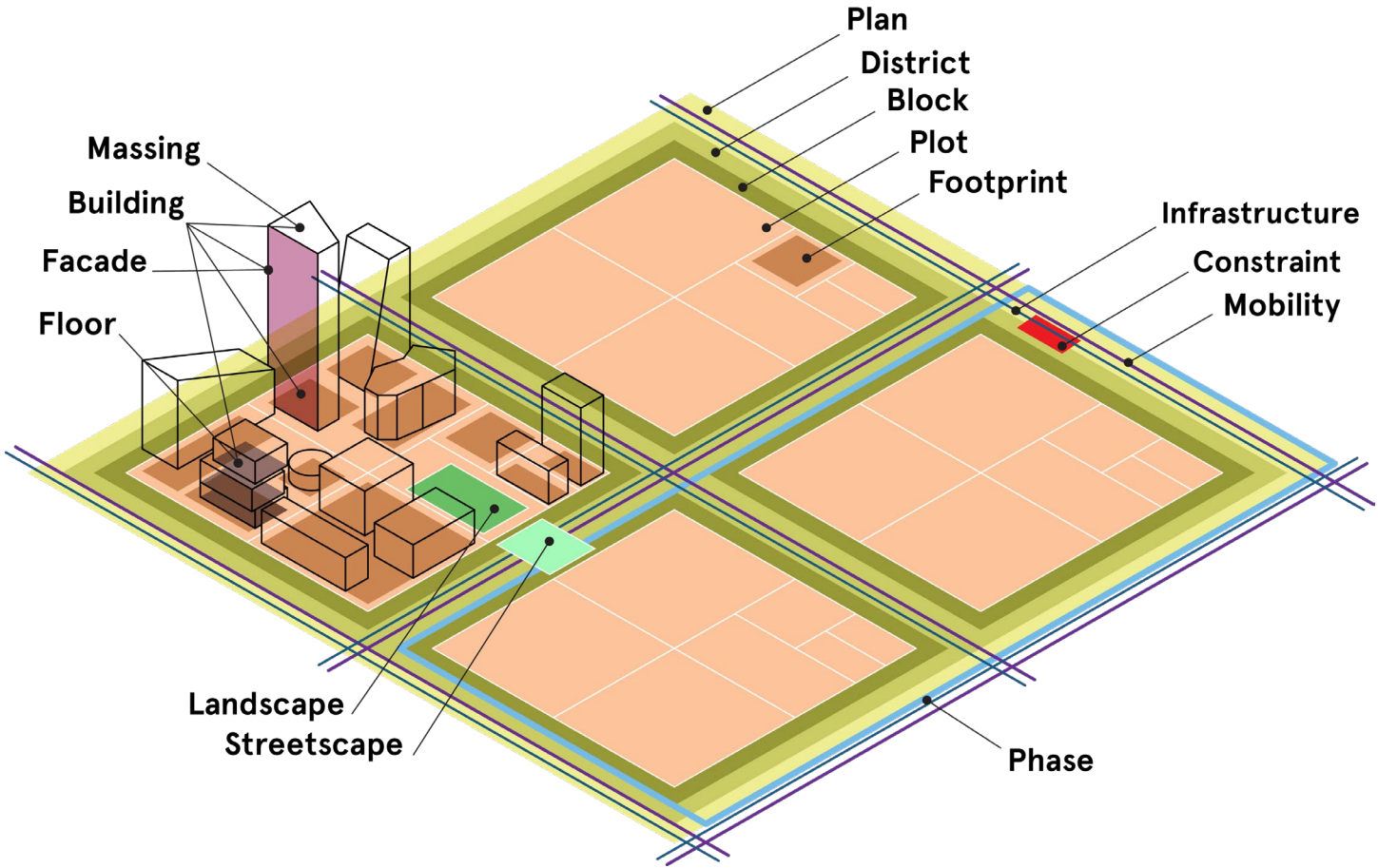
Urban Project Scale
Multiple File Formats
Community Standard
Speed to implementation
User Extensible
Focus: Urban Development

Building Scale
Difficult File Format (IFC step)
Slow Standardization Process
Limited Data Flexibility
Over complication for user
Focus: Buildings

#UIM
#CIM
#CityBIM
#NothingIsPerfect

Missing Link: 3D Urban Data & UIM.json

Custom #UIM data model to integrate with industry standards



Data Model?

Current International Standards

Many Applicable and Interlinked Standards

~ 21580	ISO, International Standards Organization
~ 275	ECMA, European Computer Manufacturers Association
~ 275	W3C, Worldwide Web Consortium (*recommendations*)
~ ?	IETF, Internet Engineering Task Force
~ 30	OGC, Open Geospatial Consortium
~ 8	bSI, Building Smart International

Challenge

- many standards criss-cross and don't interoperate
- complexity of urban development creates problems
- full stack nature of urban tools creates problems
- integrating the wide spectrum of urban domains creates problems

We must find a simple and elegant solution.

Example - Data Model Conflicts

Built environment data standards and their integration: an analysis of IFC, CityGML and LandInfra

Image credit: Bioregional

Version 1.0 • 02 March 2020
OGC Document 19-091r1
bSI TR1012

OGC® Making location count. buildingSMART International

IDBE v1.0, March 2020 Page 1 of 16
Open Geospatial Consortium & buildingSMART International

Figure 1 - Thematic coverage of the three data standards. Dark shading indicates strong coverage, light shading weaker coverage (or under development) and no shading implies no known coverage. The object icons are ordered approximately; the intention is to cluster by theme and represent finer spatial scales nearer the top.

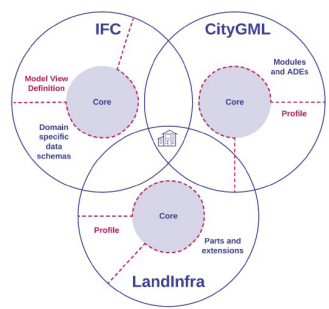


Figure 2 – The three standards differ but overlap in their thematic coverage; as an example, the concept of a building is common to all three. IFC is always subset into Model View Definitions (MVDs) for implementation, whereas the GML-based standards of CityGML and (the InfraGML implementation of) LandInfra can be subset optionally into profiles; for all three, the core of the schema must be implemented. Application Domain Extensions (ADEs) enable anyone to extend the CityGML standard to accommodate more specialist themes;

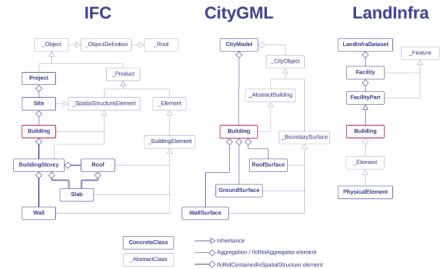


Figure 3 - The concept of a building is represented by all three standards but the detailed structure of the conceptualisation is dissimilar (geometries are not represented in this figure). The UML-like diagrams show possible representations of a very simple building; they are derived from example building instance models that are valid in each of the three standards. The IFC representation is valid for IFC4; in IFC5, a building is a predefined type of facility and a building storey is a specialisation of a facility part.

Context

- Slow to develop (ex. CityGML 3.0 is 9+ years in the making
- Huge areas of domain knowledge not included
- Difficult to extend
- Complicated to create export files

Therefore

- Why do we even need the existing schemas and file types?
- Why can't we just use an extensible JSON data object?
- Decouple Code from the Data Model - Data Oriented Design
- Radically simplify the Data Model
- Complete Data Model pulling from existing standards
- Fast development (like web) with "Worse is better" approach

Article source Link
OGC Discussion Papers Link

Example - CityGML / LandInfra / IFC

Table 1 - A Comparison of CityGML, IFC and LandInfra

#	Criterion	CityGML	LandInfra	IFC
1	Body	OGC	OGC	buildingSmart
2	Version	2.0.0	1.0.0	IFC 4 Addendum 2
3	Users	3D City Modellers	Survey Engineers & BIM	BIM & AEC
4	Encoding	GML	GML	STEP (mainly)
5	Focus	City Objects	Land and Infrastructure	BIM Models
6	Geometry	Subset of ISO 19107 / GML 3.1.1	ISO 19107 & More	ISO 10303
7	Topology	Shared surfaces only	Between facility parts	Openings, coverings & other
8	Semantics	Detailed	Not so detailed	Detailed
9	Metadata	Basic	ISO 19115 compliant	Extensively but inconsistently used
10	LODs	5 different LODs	Not supported	Not supported
11	Extensions	Generics or ADEs	Not supported	Supported
12	Appearance	Supported	Not supported	Supported
13	Software Support	Low	Almost nonexistent	Medium
14	Codelists	Supported with ISO 19103	Supported with ISO 19103	Enumerations only
15	Land Use	Simple types	Complex LADM types	Not relevant
16	File Size	Large	Large	Very Large

Reworked Table source: LandInfra to Solve GIS/BIM Quagmire Link
LandInfra Conceptual Model

Current Data Models and Encodings - Incomplete

None fit the job. Do we create our own, open source data model

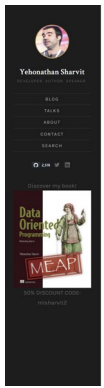
Data Models + Encoding	Owner	Detailed 3D	Data Oriented Design	Domain Complete	Extendable	Simple, Flexible	Materials	CRS Limits	Inherits	Web First	Version, Year	Good for UIM?
XML	W3C	-	-	-	yes	no	-	-	-	no	1.0, 2008	no
GML	OGC	no	no	no	yes, hard	no			XML	no	3.3, 2010	no
CityGML	OGC	yes	no	no	yes, hard	no	X3D	no	GML	no	2.0, 2012	no
JSON	ECMA	-	-	-	yes	yes		-	**-**	yes	2013	no
GeoJSON	IETF					yes		Only WGS84	JSON	yes	2016	no
GeoJSON 3D	OGC					yes		no	JSON	yes	**tbd**	possibly
CityJSON	CC	yes	no	no	yes, hard	no	X3D	no	CityGML	yes	2.0, 2012	
JSON-LD	W3C								JSON			
JSON Graph Format	CC											
LBD: Linked Building Data	W3C											
i3S	OGC	yes	no	no	?	?			JSON	yes	1.1, 2019	
Land/InfraGML	OGC	yes	no	no	no	no			GML	no	1.0, 2016	
LADM	ISO	yes		no	?	?			-	-	2012	no
IFC	bSI	yes	no	no	yes	no			-	no	4.0.2.1, 2017	no
Spatial Things	W3C	-	yes	-	-	-		-	-	-	-	-
uimJSON	**?**	**yes**	**yes**	**yes**	**yes**	**yes**	**yes, how?**	**no**	**JSON**	**yes**	**0.1, 2021**	**?**

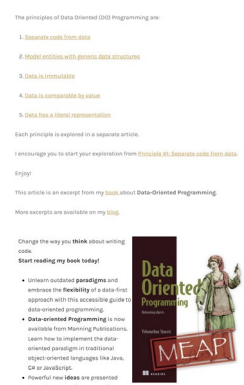
Urban Data Model Inspiration

1. Web First 'Spatial Things'

- Web of Data / Linked Data
- No Platform
- No Architecture
- Tiny Files
- Serialize easily
- REST/GraphQL

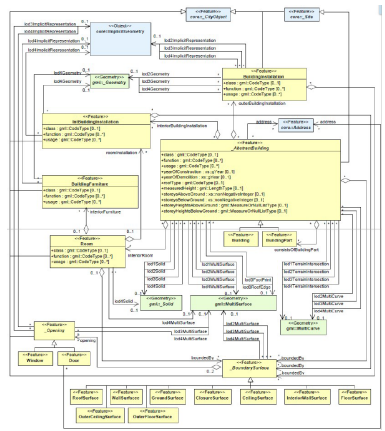
2. Data-Oriented Design





1. Separate code from data
2. Model entities with generic data structures
3. Data is immutable
4. Data is comparable by value
5. Data has a literal representation

3. No UML Hell

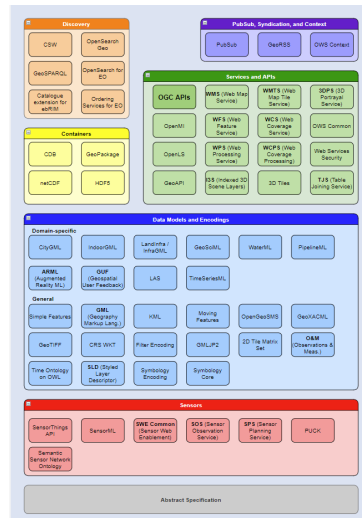


The data model should have extreme simplicity for software architecture flexibility.

There should be 3 independent elements in the encoding:

1. Geometry
2. Encoding Meta Data
3. Domain Knowledge sub-schemas

4. Copy What's Good



Use existing Nomenclature to support data mapping:

- OGC
- ISO
- bSI
- W3C
- ECMA
- OSM
- ,,,

Urban Data Model - Critical Needs and Goals

1. **Complete** urban development needs
2. **Many Schemas** that can index a single dataset
3. Consistent use of existing standards **Nomenclature** (ex. CityGML)
4. **Multi-dimensional**: **3D** (X, Y, Z) + **4D** (Time) + **5D** (NData Arrays)
5. Works with **all Geometry** Primitives, Shapes and Data Structures
6. Supports both **Cartesian Space + Geodetic Space**
7. **Loose coupling** of Geometry + Attributes
8. **Pick-and-choose freely** from Domain Knowledge Schemas
9. **Decoupled** Code and Data (Data Oriented Design)
10. **Useful Ontology:** Preserve Technical, Semantic and Descriptive details
11. Schemas don't determine **Software Architecture**
12. **Scaleless - No LOD** (Level of Detail)
13. Simple, **Maintainable, Machine Readable** (W3C Spatial Things)
14. **Web First** technologies and data exchange
15. **Generic Encoding** (JSON) and smallest file size
16. **Open Source, Community Driven**
17. **Human readable** and typeable
18. Fast, iterative development for **Business Needs** (Worse is Better)

SIMO

SIMO Signup!

www.spatiomatics.com

Complex Urban Challenges

Optimization: Rotterdam CS (Maxwan)

Smart City: Jurong Lake District Singapore (KCAP)

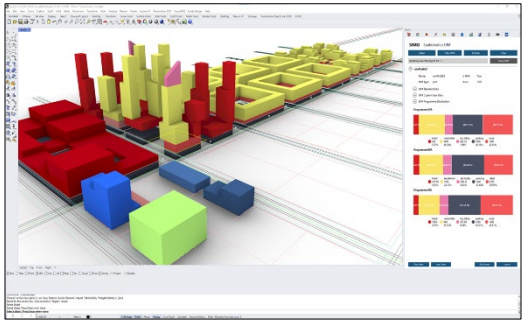
Transformation: Den Haag Binckhorst

Densification: Eindhoven Knoop XL (KCAP)

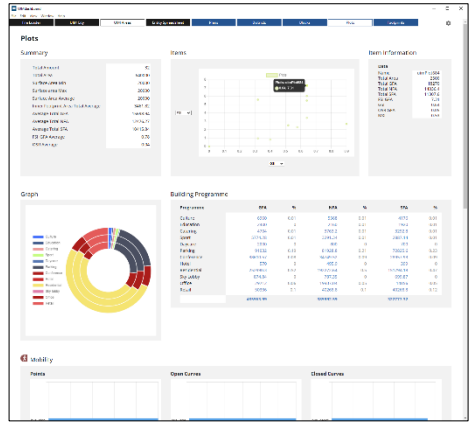
It’s not GIS, it’s not BIM…

It’s SIMO, the best of both

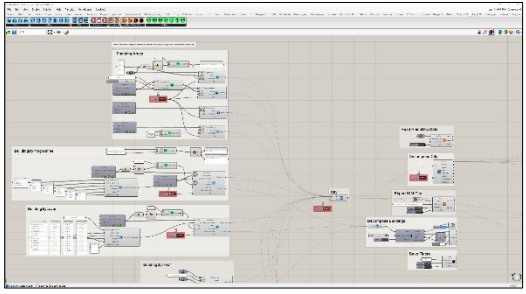
SIMO Solution



Rhino App (C#)



Data Dashboard App
JS + React + Electron



Grasshopper App (C#)

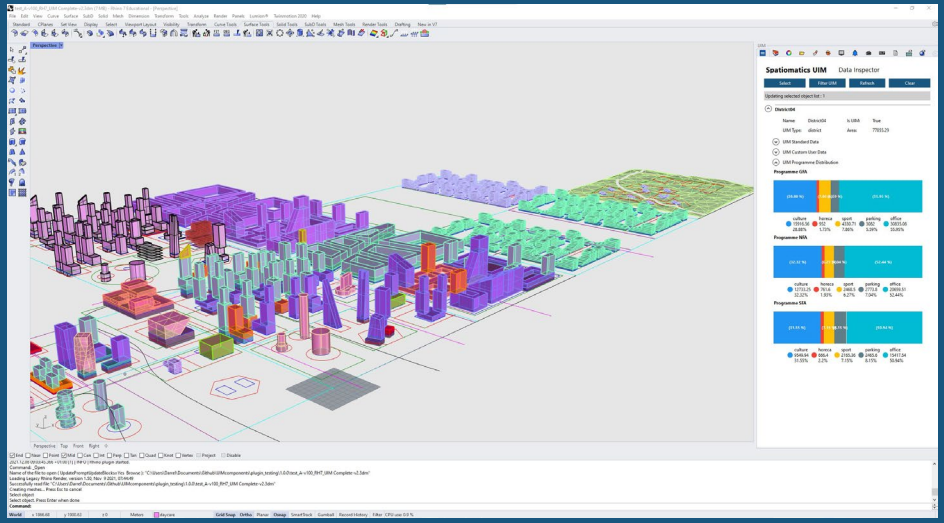
Software Development Topics

- Rhino Plugin
- Grasshopper Plugin
- Electron App
- Licensing challenges
- DevOps, Testing
- User Feedback, Telemetry
- UIM.json Schema Validation

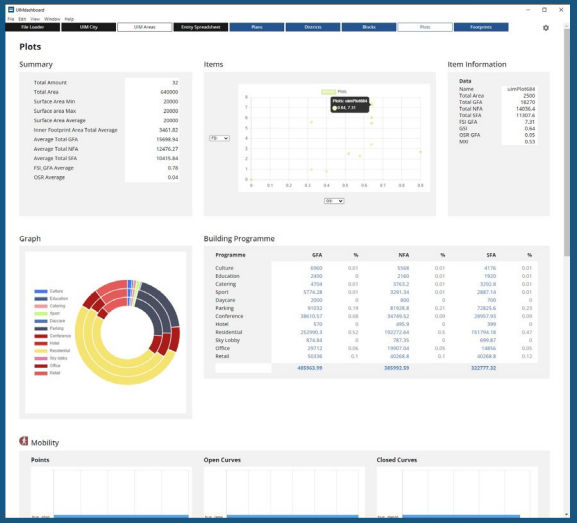
UIM.json

Intelligence 3D + Data

Create structured data for multi-disciplinary projects

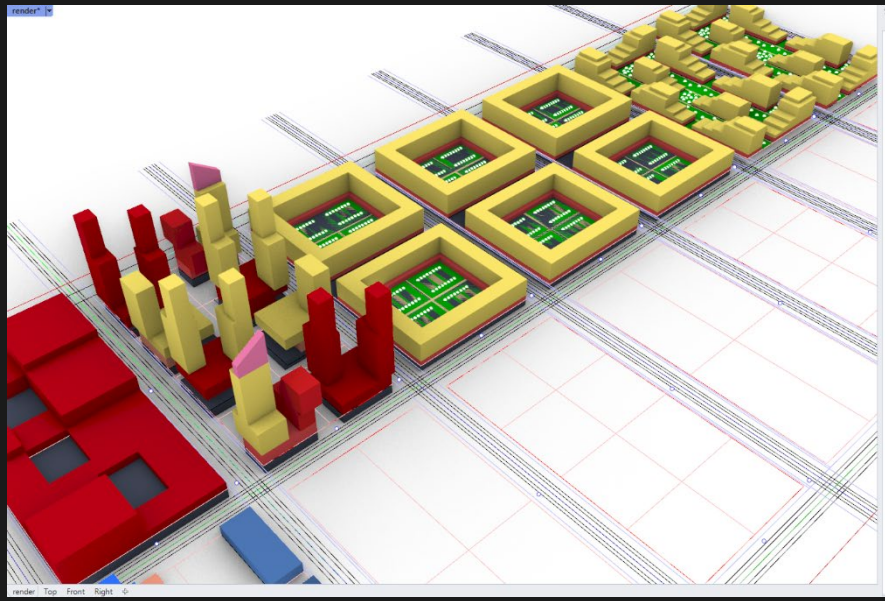


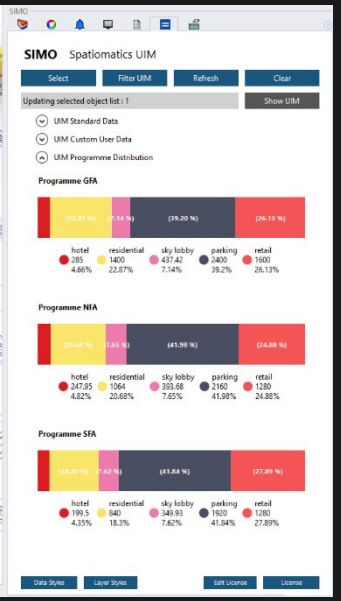
3D Model in Rhino & Grasshopper

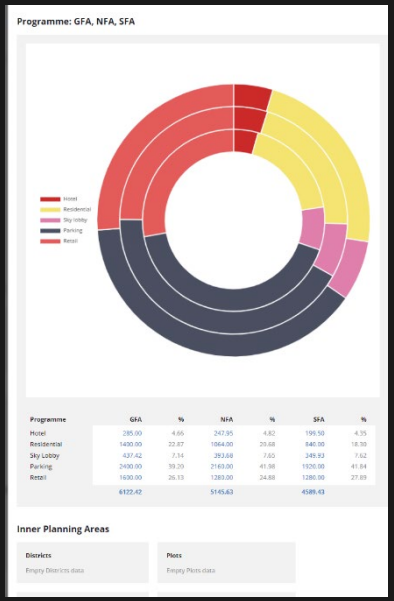


SIMO Data Dashboard

Example: District Design







Model & Visualization: SIMO

Example: Streetscape & Transport Design

Visualization: Twinmotion

Streetscapes

Type	Point #	Open Curve #	Length	Closed Curve #	Area
parking	12.00			1.00	150.00
roadway				1.00	510.00
grass				1.00	615.00
bikelane				1.00	340.00

3D Model & Data Visualization: SIMO

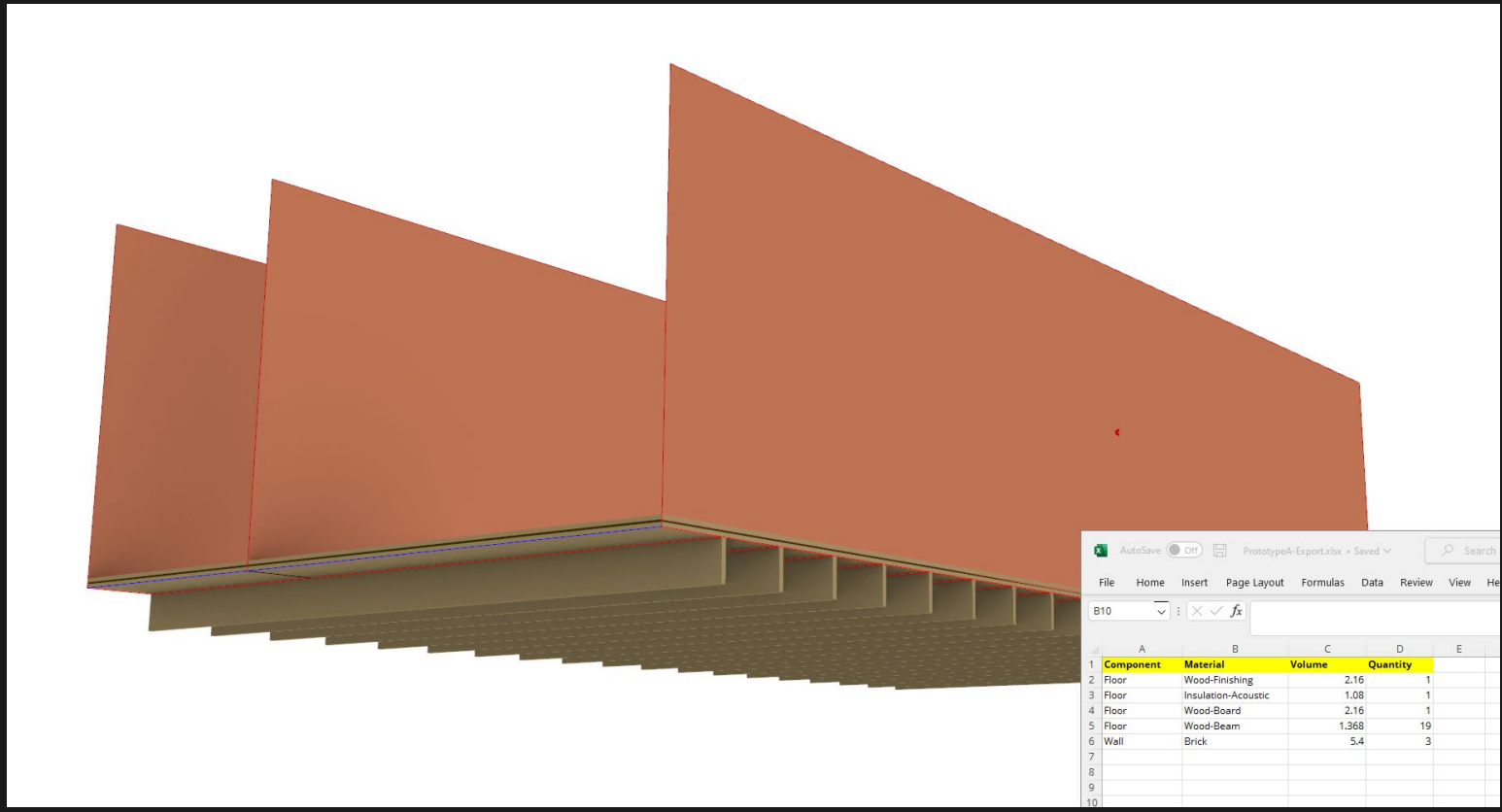
SIMO Dashboard

File Loader | City | Plan Areas | Elements | Data Tables

Plans | Streetscapes

	uim_type	uuid	name	type	centroid	buffer_distanc	in_plan	in_
1	streetscape	34e6b9b8-6b	bikelane_69380	bikelane	{"x":21.5,"y":42.5,"z":0}		6b79c3fa-a705	
2	streetscape	439c5fd3-545	bikelane_94796	bikelane	{"x":8.5,"y":42.5,"z":0}		6b79c3fa-a705	
3	streetscape	01de380c-cfa	grass_85a5b	grass	{"x":23.5,"y":42.5,"z":0}		6b79c3fa-a705	
4	streetscape	804908ef-e6e	grass_6f232	grass	{"x":10.75,"y":42.5,"z":0}		6b79c3fa-a705	
5	streetscape	4e307c7e-7fa	grass_ece59	grass	{"x":6.5,"y":42.5,"z":0}		6b79c3fa-a705	
6	streetscape	d3f6ed88-8e5	grass_745c7	grass	{"x":19.25,"y":82.5,"z":0}		6b79c3fa-a705	
7	streetscape	b2c1a065-cb7	grass_20f70	grass	{"x":19.25,"y":62.5,"z":0}		6b79c3fa-a705	
8	streetscape	413c3841-db	grass_9ee02	grass	{"x":19.25,"y":42.5,"z":0}		6b79c3fa-a705	
9	streetscape	e8fa5c7d-d90	grass_b1918	grass	{"x":19.25,"y":22.5,"z":0}		6b79c3fa-a705	
10	streetscape	bfe29489-193	grass_6fe61	grass	{"x":19.25,"y":2.5,"z":0}		6b79c3fa-a705	
11	streetscape	d1a1fd00-3bc	roadway_886e1	roadway	{"x":15,"y":42.5,"z":0}		6b79c3fa-a705	
12	streetscape	f52b4b03-82f	parking_83f7e	parking	{"x":19.25,"y":7.5,"z":0}		6b79c3fa-a705	
13	streetscape	eb563512-4fe	parking_b9d7e	parking	{"x":19.25,"y":12.5,"z":0}		6b79c3fa-a705	
14	streetscape	77e0d795-d1	parking_d0dec	parking	{"x":19.25,"y":17.5,"z":0}		6b79c3fa-a705	
15	streetscape	d2468dd0-c8	parking_7d4e5	parking	{"x":19.25,"y":32.5,"z":0}		6b79c3fa-a705	
16	streetscape	d389e504-c7	parking_62e1a	parking	{"x":19.25,"y":37.5,"z":0}		6b79c3fa-a705	
17	streetscape	c9c8d78b-01	parking_21221	parking	{"x":19.25,"y":47.5,"z":0}		6b79c3fa-a705	
18	streetscape	d65b1c98-72	parking_bc8a3	parking	{"x":19.25,"y":52.5,"z":0}		6b79c3fa-a705	
19	streetscape	1481119b-53	parking_efd04	parking	{"x":19.25,"y":57.5,"z":0}		6b79c3fa-a705	
20	streetscape	e8419710-32	parking_608d2	parking	{"x":19.25,"y":27.5,"z":0}		6b79c3fa-a705	
21	streetscape	5aed7d25-e9	parking_92561	parking	{"x":19.25,"y":67.5,"z":0}		6b79c3fa-a705	
22	streetscape	57be6bfd-34f	parking_86354	parking	{"x":19.25,"y":72.5,"z":0}		6b79c3fa-a705	
23	streetscape	d1709f0e-072	parking_0b88c	parking	{"x":19.25,"y":77.5,"z":0}		6b79c3fa-a705	
24	streetscape	53e10a90-95	parking_59038	parking	{"x":19.25,"y":7.5,"z":0}		6b79c3fa-a705	
25	streetscape	82ee7d0d-fb7	parking_57092	parking	{"x":19.25,"y":12.5,"z":0}		6b79c3fa-a705	
26	streetscape	e409be4b-c8	parking_7ec0f	parking	{"x":19.25,"y":17.5,"z":0}		6b79c3fa-a705	
27	streetscape	4ceb8bc0-89	parking_5ee97	parking	{"x":19.25,"y":32.5,"z":0}		6b79c3fa-a705	
28	streetscape	2bb786cd-a6	parking_3b7cd	parking	{"x":19.25,"y":37.5,"z":0}		6b79c3fa-a705	
29	streetscape	9a043e11-b1	parking_616a9	parking	{"x":19.25,"y":47.5,"z":0}		6b79c3fa-a705	
30	streetscape	cc3e3111-5f6	parking_c7902	parking	{"x":19.25,"y":52.5,"z":0}		6b79c3fa-a705	
31	streetscape	21a3ca6e-40	parking_29297	parking	{"x":19.25,"y":57.5,"z":0}		6b79c3fa-a705	
32	streetscape	d22ba94d-3d	parking_12ca8	parking	{"x":19.25,"y":27.5,"z":0}		6b79c3fa-a705	
33	streetscape	ba3e2a7f-de1	parking_3270a	parking	{"x":19.25,"y":67.5,"z":0}		6b79c3fa-a705	
34	streetscape	8fc5b1f0-2cc	parking_a851f	parking	{"x":19.25,"y":72.5,"z":0}		6b79c3fa-a705	
35	streetscape	23dc6182-e9	parking_f21c8	parking	{"x":19.25,"y":77.5,"z":0}		6b79c3fa-a705	

Example: Built Environment Circularity

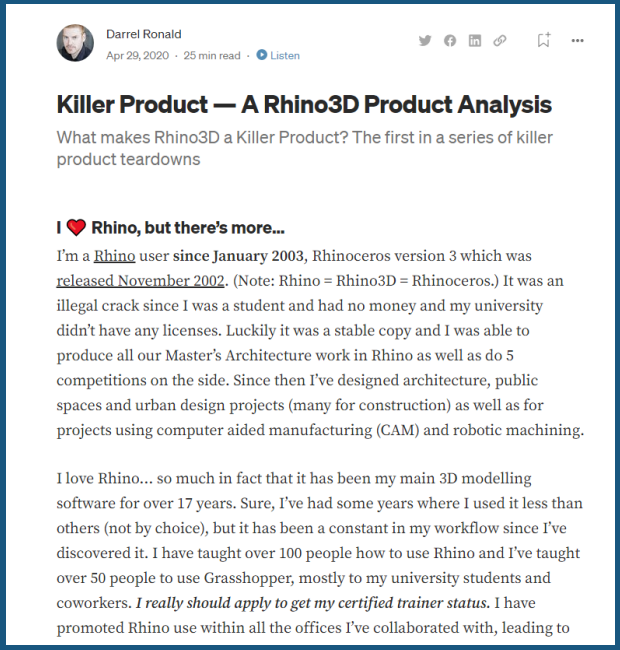


	A	B	C	D	E
1	Component	Material	Volume	Quantity	
2	Floor	Wood-Finishing	2.16	1	
3	Floor	Insulation-Acoustic	1.08	1	
4	Floor	Wood-Board	2.16	1	
5	Floor	Wood-Beam	1.368	19	
6	Wall	Brick	5.4	3	
7					
8					
9					
10					

Model & Visualization: SIMO

McNeel Rhino / Grasshopper (RH, GH)

Why Rhino & Grasshopper?



Darrel Ronald
Apr 29, 2020 · 25 min read · Listen

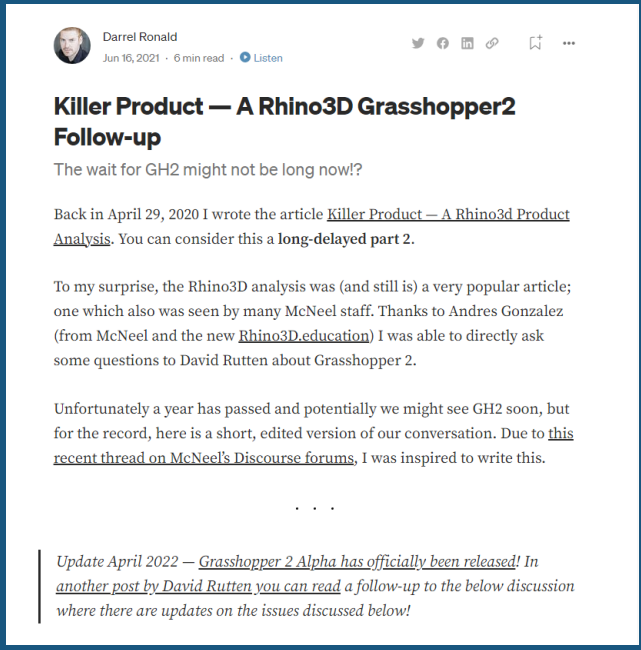
Killer Product — A Rhino3D Product Analysis

What makes Rhino3D a Killer Product? The first in a series of killer product teardowns

I ❤ Rhino, but there's more...

I'm a Rhino user **since January 2003**, Rhinoceros version 3 which was released November 2002. (Note: Rhino = Rhino3D = Rhinoceros.) It was an illegal crack since I was a student and had no money and my university didn't have any licenses. Luckily it was a stable copy and I was able to produce all our Master's Architecture work in Rhino as well as do 5 competitions on the side. Since then I've designed architecture, public spaces and urban design projects (many for construction) as well as for projects using computer aided manufacturing (CAM) and robotic machining.

I love Rhino... so much in fact that it has been my main 3D modelling software for over 17 years. Sure, I've had some years where I used it less than others (not by choice), but it has been a constant in my workflow since I've discovered it. I have taught over 100 people how to use Rhino and I've taught over 50 people to use Grasshopper, mostly to my university students and coworkers. ***I really should apply to get my certified trainer status.*** I have promoted Rhino use within all the offices I've collaborated with, leading to



Darrel Ronald
Jun 16, 2021 · 6 min read · Listen

Killer Product — A Rhino3D Grasshopper2 Follow-up

The wait for GH2 might not be long now!?

Back in April 29, 2020 I wrote the article Killer Product — A Rhino3d Product Analysis. You can consider this a **long-delayed part 2**.

To my surprise, the Rhino3D analysis was (and still is) a very popular article; one which also was seen by many McNeel staff. Thanks to Andres Gonzalez (from McNeel and the new Rhino3D.education) I was able to directly ask some questions to David Rutten about Grasshopper 2.

Unfortunately a year has passed and potentially we might see GH2 soon, but for the record, here is a short, edited version of our conversation. Due to this recent thread on McNeel's Discourse forums, I was inspired to write this.

. . .

> *Update April 2022 — Grasshopper 2 Alpha has officially been released! In another post by David Rutten you can read a follow-up to the below discussion where there are updates on the issues discussed below!*

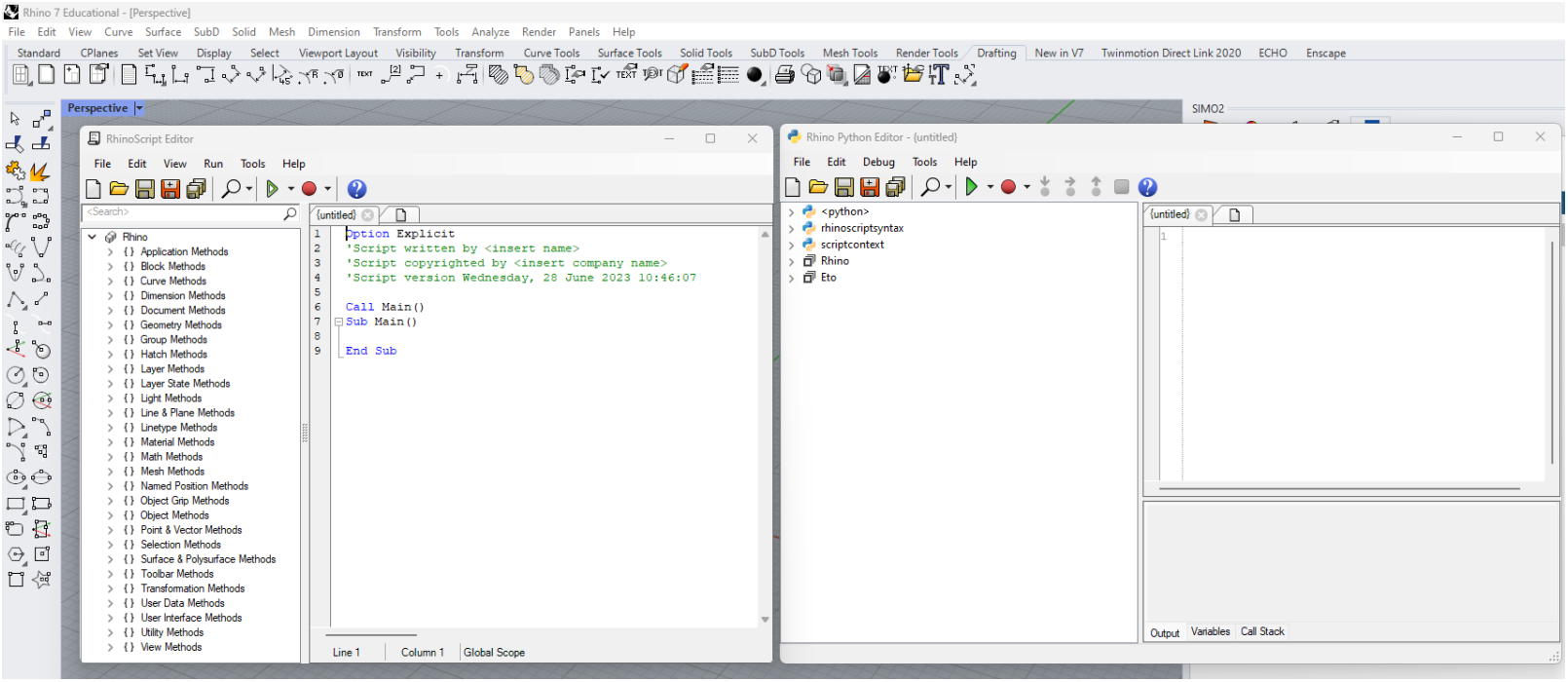
2003 – **Rhino User**
2007 – **Grasshopper User**
2020 – **SIMO Prototype GHPY**
2022 – **SIMO launch**

Articles : https://medium.com/@darrelronald

Easy to experiment as a user and developer
Python lowers the barrier to entry
C# is robust for long-term development

Not perfect, but a WIP
Lots of suggestions for improvement in the articles, some acted upon already

Scripting in Rhino 3D

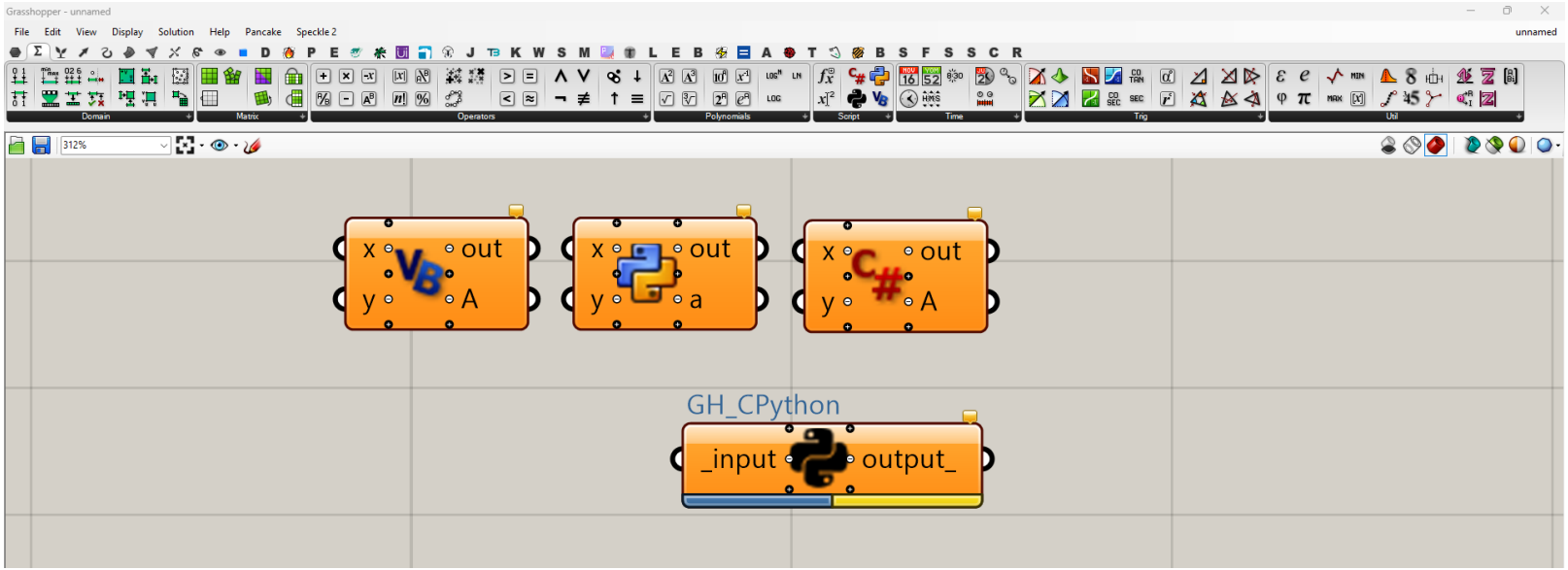


In Rhino (v6, v7) you can only script with **RhinoScript** or **RhinoPython**. You can use these editors, run script files or assign scripts to buttons.

Rhino v8 will be different.

Source : [Guide to Scripting in Rhino]

Scripting in Grasshopper 3D

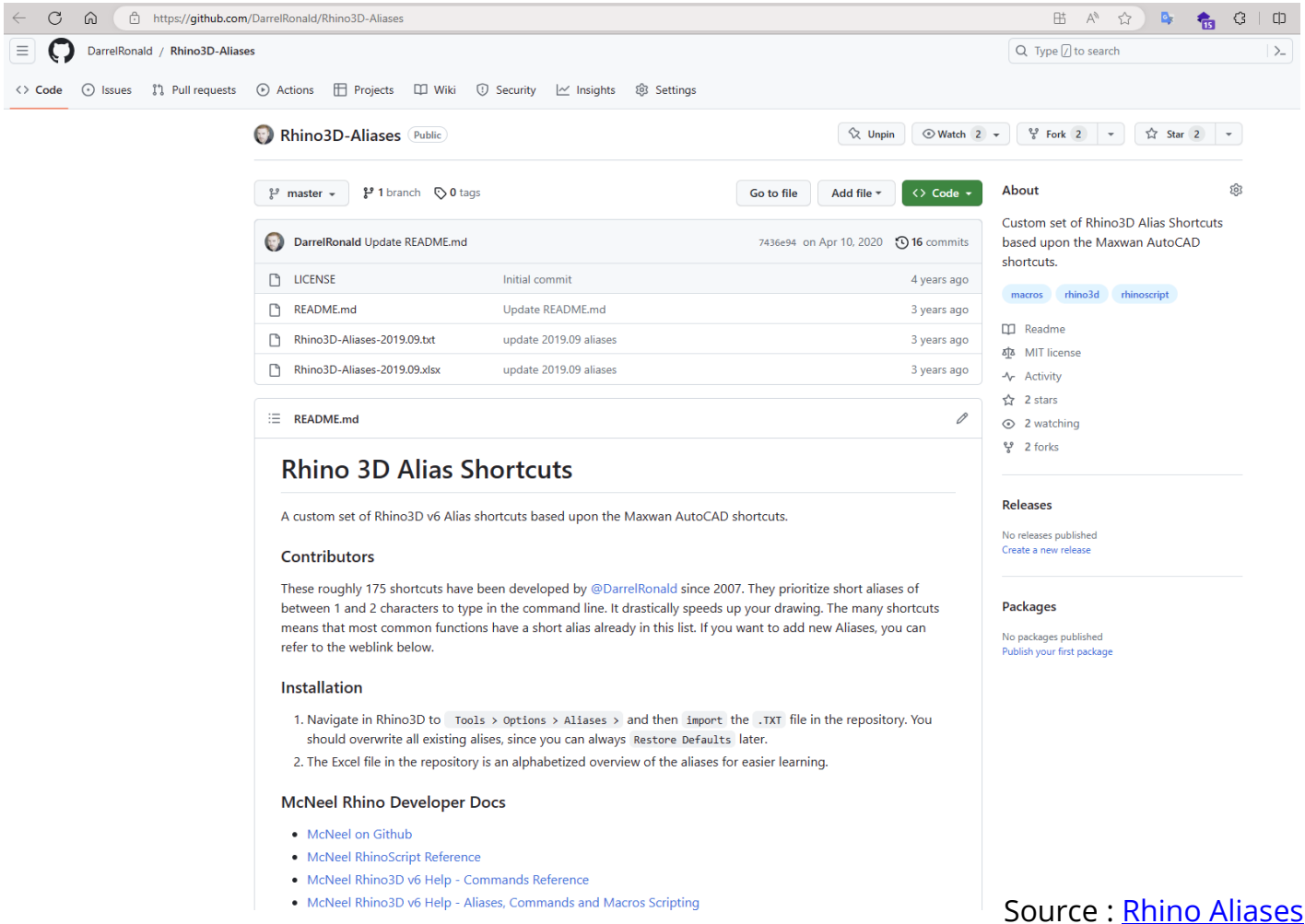


In Grasshopper (v1) you can write in scripts in 3 Languages: VBScript, RhinoPython and C# (.Net Framework 4.8).

You can also install the component “GH_Cpython” to access Python 3.*

Source : [Your first Python Component]
[Essential Guide to C# Scripting in Grasshopper]
[Guides Rhinoscript and VBScript]

Rhino (& AutoCAD) Aliases



https://github.com/DarrelRonald/Rhino3D-Aliases

DarrelRonald / Rhino3D-Aliases

Type / to search

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Rhino3D-Aliases Public

Unpin | Watch 2 | Fork 2 | Star 2

master | 1 branch | 0 tags | Go to file | Add file | Code

DarrelRonald Update README.md — 7436e94 on Apr 10, 2020 — 16 commits

LICENSE	Initial commit	4 years ago
README.md	Update README.md	3 years ago
Rhino3D-Aliases-2019.09.txt	update 2019.09 aliases	3 years ago
Rhino3D-Aliases-2019.09.xlsx	update 2019.09 aliases	3 years ago

README.md

Rhino 3D Alias Shortcuts

A custom set of Rhino3D v6 Alias shortcuts based upon the Maxwan AutoCAD shortcuts.

Contributors

These roughly 175 shortcuts have been developed by @DarrelRonald since 2007. They prioritize short aliases of between 1 and 2 characters to type in the command line. It drastically speeds up your drawing. The many shortcuts means that most common functions have a short alias already in this list. If you want to add new Aliases, you can refer to the weblink below.

Installation

1. Navigate in Rhino3D to `Tools > Options > Aliases >` and then `Import` the `.TXT` file in the repository. You should overwrite all existing alises, since you can always `Restore Defaults` later.
2. The Excel file in the repository is an alphabetized overview of the aliases for easier learning.

McNeel Rhino Developer Docs

- McNeel on Github
- McNeel RhinoScript Reference
- McNeel Rhino3D v6 Help - Commands Reference
- McNeel Rhino3D v6 Help - Aliases, Commands and Macros Scripting

About

Custom set of Rhino3D Alias Shortcuts based upon the Maxwan AutoCAD shortcuts.

macros | rhino3d | rhinoscript

Readme
MIT license
Activity
2 stars
2 watching
2 forks

Releases

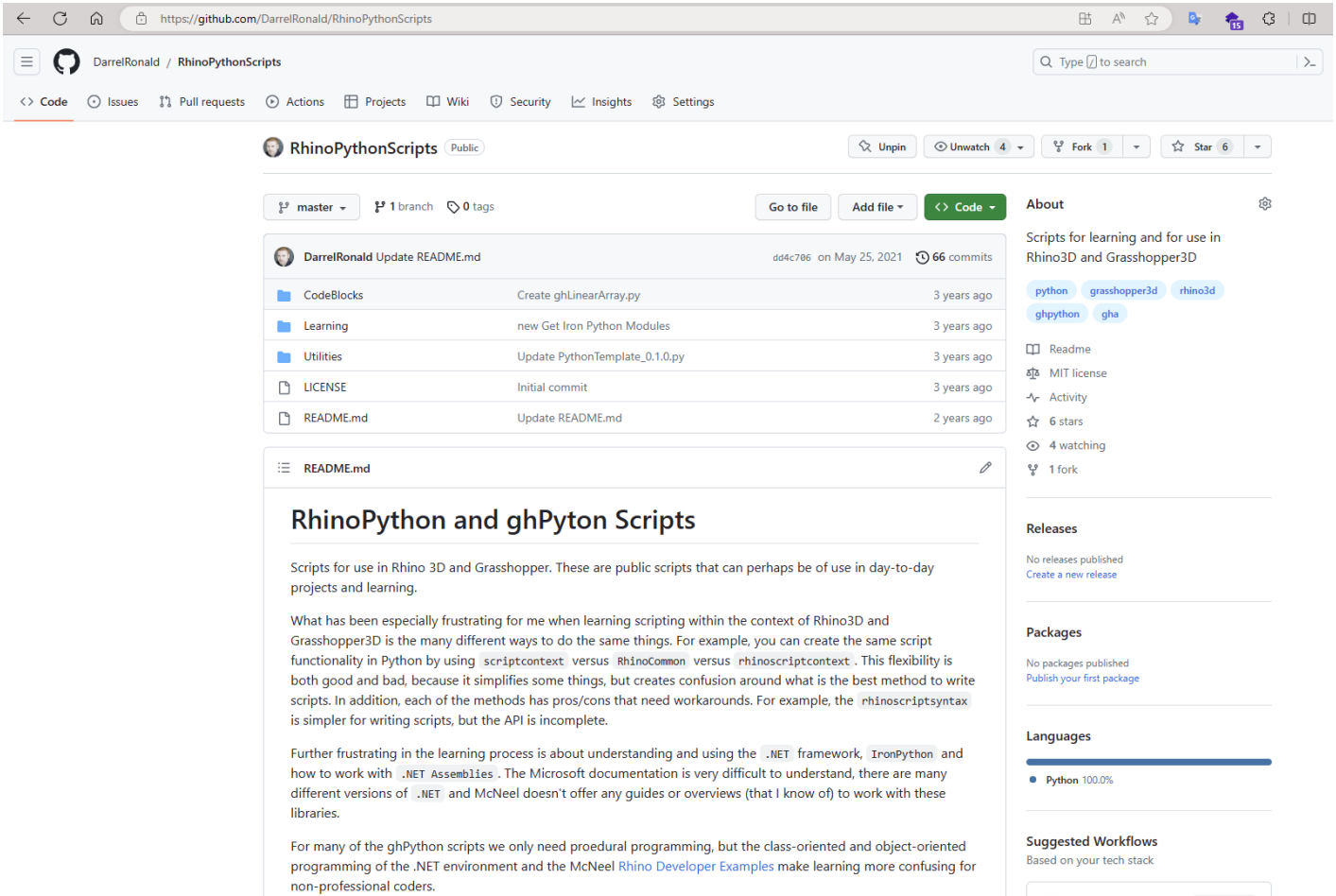
No releases published
Create a new release

Packages

No packages published
Publish your first package

Source : [Rhino Aliases](#) (RhinoScript)
[Maxwan AutoCAD Shortcuts](#) (lisp)

RhinoPythonScripts (by Darrel)



Source : [RhinoPython Scripts, Tips & Tricks]

Handy Python 2.* versus Python 3.*

Sources
- John Guttag. Intro to Programming with Python
- Sebastian Raschka Key difference 2.x 3.x
- Python 3 Readiness tracker
- Geekforgeeks main differences 2.x 3.x

__future__ module explained - explained deeply
You can import python 3.x modules into 2.x using the _future_ module

- from __future__ import division
- from __future__ import print_function
- from __future__ import with_statement
- from __future__ import generators
- from __future__ import nested_scopes
- from __future__ import unicode_literals

Print
- 2.x = print 'something'
- 3.x = print('something')

Input
- 2.x = input() and raw_input()
- 3.x = input()

Division operator
- 2.x = better to use a floating-point number to get correct division
    - print 7/5 >>> 1
    - print -7/5 >>> -2
- 3.x = works normal
    - print(7/5) >>> 1.4
    - print(-7/5 >>> -1.4

Unicode
- 2.x = implicit str type is ASCII
- 3.x = implicit str type is Unicode

XRange
- 2.x = uses xrange() to return an iterator object
- 3.x = uses range() to return an iterator object

Error Handling explained
- 2.x = different syntax
- 3.x = different syntax, needs 'as'

Transition from 2-3
- Active State Transition from Python 2 to 3

Development Challenges

User Documentation & Learning Experience

Where McNeel needs to really improve is the touch points for users across their web sites and how content is published to users and developers. There is so much non-standardized, scattered, out-of-date, incomplete documentation, resources and training material that it is really a mess of content and interface.

To get a sense of the vast selection of often conflicting, duplicate or hidden touch points for McNeel products and user experience, this is a comprehensive list (as of 2020)

- McNeel Associates — mainly redirects to other rhino3d websites
- Rhino3D — the main software product page
- Rhino3D Blog — latest news
- Rhino3D Help — great resources linked from the Rhino3D software interface
- Rhino3D Online User Guide — helpful tutorials linked from the Rhino3D software interface
- Rhino3D Tutorials —helpful tutorial links to other resources
- Rhino3D Training and Certification — Official training
- Rhino3D Resources — mix of plugins, learning, etc.
- Grasshopper3D — now closed but a valuable resource of discussions and file sharing; partially replaced by Food4Rhino and Discourse Forums
- Food4Rhino — current Grasshopper3D plugins finder and downloads
- McNeel Wiki — collection of resources
- Discourse Forums — current community forum
- Rhino3D Developer — resources for software developers
- Multiple different users accounts on YouTube and Vimeo users from McNeel and staff
- Multiple blogs by McNeel staff could be located together through a McNeel interface, such as David Rutten and Steve Baer

Articles : https://medium.com/@darrelronald

Developer Experience – Wall 1

One thing that is particularly confusing with Rhino3D is the many languages you can use to develop software and scripts within the Rhino and Grasshopper platform. Each one has its pros, cons and trade-offs. To a new or non-programmer, you hit the **first wall** of difficulty or confusion learning about these technologies and where to start.

- C# — the main language for .NET and the cross-platform **RhinoCommon** API. It is the most useful for developing Rhino software but not the easiest to learn
- IronPython — a .NET implementation limited to Python 2. **RhinoPython** has all the goodness of Python, plus you can execute within Rhino and Grasshopper
- RhinoScript — the main language for the Rhino command line, Rhino macros and Grasshopper component. **RhinoScript** is based on Microsoft's VB Script.
- VB Script —executable in a Grasshopper3D component
- JavaScript — new for use with the RhinoCompute service and Rhino3DM.js
- C++ SDK — the language of the openNURBS created by McNeel and the Windows Rhino software and plugins

Articles : https://medium.com/@darrelronald

Developer Experience – Wall 2

Once you start to digest these diverse entry points, you might hit **a second wall**. In particular, when learning Python for Rhino3D development, you will find another set of sub-choices for writing your scripts. Again, each one has its pros, cons and trade-offs. There are a set of main APIs that you should learn:

1. RhinoCommon — the complete Rhino API for cross-platform development, but more difficult to use.
2. RhinoScriptSyntax — an easy-to-use wrapper around some RhinoCommon API modules, however it doesn't include all RhinoCommon modules.
3. Scriptcontext —controls code execution in Grasshopper or Rhino. I still haven't found an API reference or learning material online, but its role is important.
4. NodeInCode — while not technically an API, it is the magical ability to call Grasshopper components from within Rhino or Grasshopper scripts.
5. Hops — Link a Grasshopper Hops component to an external Python 3.* server that runs Python 3.* scripts linked to the Hops component

Articles : https://medium.com/@darrelronald

Developer Experience – Wall 3

In the McNeel training material online, there are **many gaps of knowledge** not discussed for non-programmers. This would be a very useful set of training material, or at least links to good resources. Some of the example topics, which a computer scientist would be well aware of are:

- Integration with Microsoft's .NET Framework and libraries
- Object-oriented programming (for Rhino3D)
- Procedural programming (easier and for simple scripts)
- Functional Programming (for Grasshopper 3D)

There are however very good discussions online on both the original (now closed) Grasshopper3D website as well as the newer McNeel Discourse forums. But the best forum posts/discussions that answer questions we all face **don't get extracted out into clear summary articles** in the Rhino Developer Documents.

Further frustrating is that there are many different systems and locations for API documentation. Some API docs are found on **Github Pages** while others are found on **ReadTheDocs**. Worse is that some of these links, like Rhino Compute on ReadTheDocs isn't linked from the official RhinoCompute page.

Articles : https://medium.com/@darrelronald

Rhino and Grasshopper Versions – Wall 4

Even though Rhino is one of the best value for feature 3D software in the industry, users are still very slow to upgrade.

Developing SIMO, I encounter 4 versions:

- Rhino v5 – unsupported by SIMO
- **Rhino v6**
- **Rhino v7**
- **Rhino v8 (WIP)** – C#10, Rhino 3.*

- **Grasshopper v1**
- Grasshopper v2 (WIP) – Untested (completely new code)

Articles : https://medium.com/@darrelronald

GH Python versus C# Development – V6 and V7

Beginner

- It is much easier to just code directly within Rhino (**Script Editor**) or Grasshopper (**Script Components**).

Intermediate

- It's helpful to test ideas in Python, then convert to C# when stable. Use Visual Studio for C# debugging, intellisense, etc.

Advanced

- Directly build all Scripts and Components in C#. Use Visual Studio for C# debugging, intellisense, etc.
- You can build Hops servers and run complex Python 3.* scripts externally to the Grasshopper UI. Doesn't work with Rhino.

In V8 this will change since the editor will have access to the latest versions of C# 10* and Python 3.10.*. V8 will have VS Code built into it.

The Future

Key Lessons

The standard software route

1. Preparation
- Do the Market and Competition Research
- Analyze the Customer Segments
- Deep dive the Customer pains, gains, needs

2. Prototyping
- Prototype First
- Get Customers to Pay (= Funding)

3. Business Model Canvas
- Iterate on the Business Model + Customer Segment
- Only build and launch when paying customers

SIMO Signup!

www.spatiomatics.com

Community, Partners, Creators, Coders

Talk with us about

- Your projects & use cases
- Your Jobs & Needs
- SIMO Product Partnerships
- UIM Code Partnerships
- Consulting, Support

Darrel Ronald
https://www.linkedin.com/in/darrelronald/
www.Spatiomatics.com

Thank you!