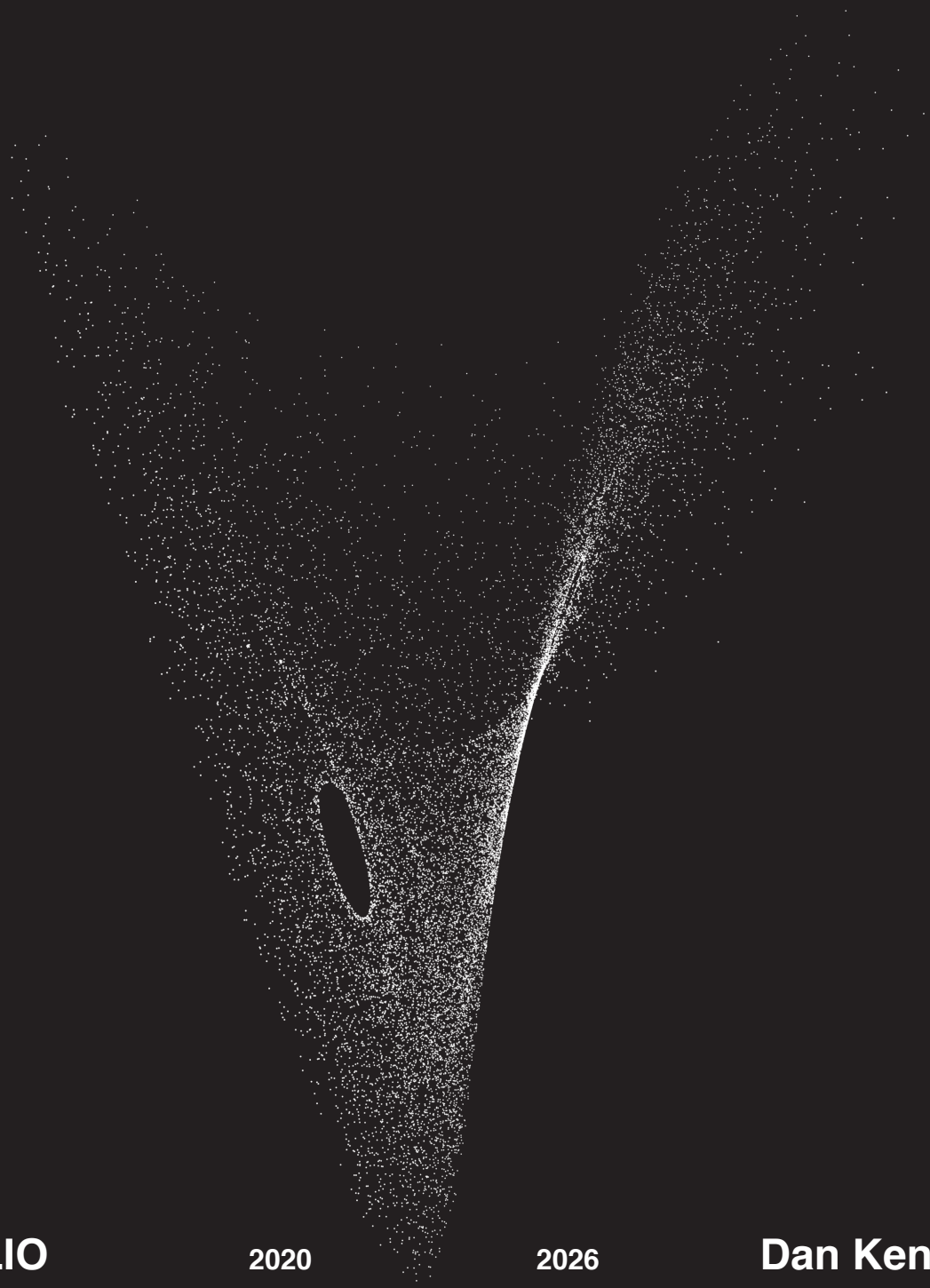


PORTFOLIO

2020 2026

Dan Kenerson

Front:
Lorenz strange attractor,
front-perspective view

Lorenz Equation:
dx = (a • (y - x)) • dt
dy = (x • (b - z) - y) • dt
dz = (x • y - c • z) • dt

On the Creative Process:

“Wisdom is not a product of schooling but of the lifelong attempt to acquire it.” – Albert Einstein

I love to create; whether through the lens of a camera, the stroke of a pen, or the clack of a keyboard. Lessons learned in the darkroom about the interplay of light and shadow find their way into pen-plotting techniques. Algorithms developed through programming breathe life into generative architecture. My creative pursuits are not only an outlet for expression but also a constant source of learning and inspiration to explore further.

As a lifelong student of design and technology, I strive to expand my understanding with every project and every medium I engage with. This portfolio highlights how my work connects across disciplines, showcasing pivotal moments where critical concepts were discovered and applied.

Contents.

01 Reimagining the Familiar

Using algorithms for conceptual iteration.

Year: 2024
Tools: Blender, Python, JavaScript, Three.js

Of the many different ways to acheive procedural generation wave function collapse has captivated me for a long time. The algorithm works by limiting possibilities within a space, determined by the spaces around it. Imagine sudoku: each cell is bound by rules based on it's row, column and square. Uncertainy narrows as the grid fills until a single correct solution is evident.

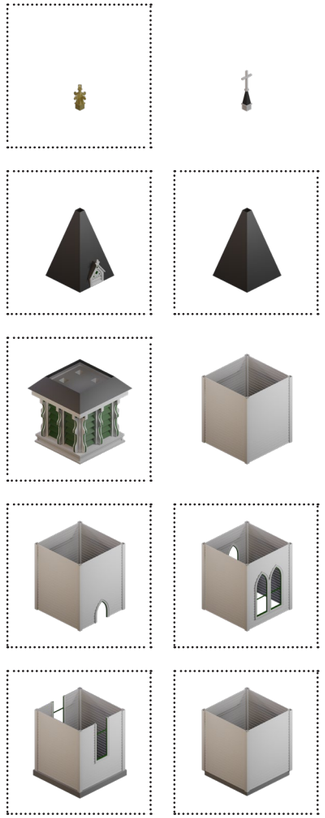
I first experimented with this idea 7 years ago—a crude terrain generator in Python that filled tiles based on a ruleset.

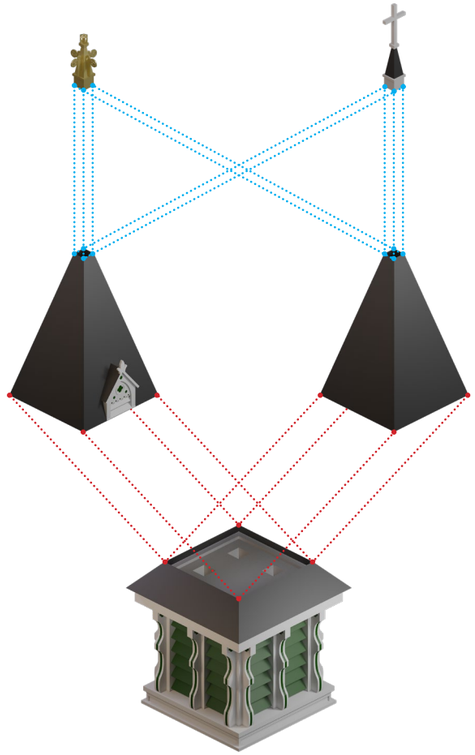
Today, I revisit this algorithm to explore it's potential in generating conceptual forms procedurally.

The Tiles.

The tiles used in making these structures are displayed below. Each is sized to work in a 3D grid of cubes.

Note: For each cube shown, there are the 3 alternate rotations of 90° which are also used in the set.





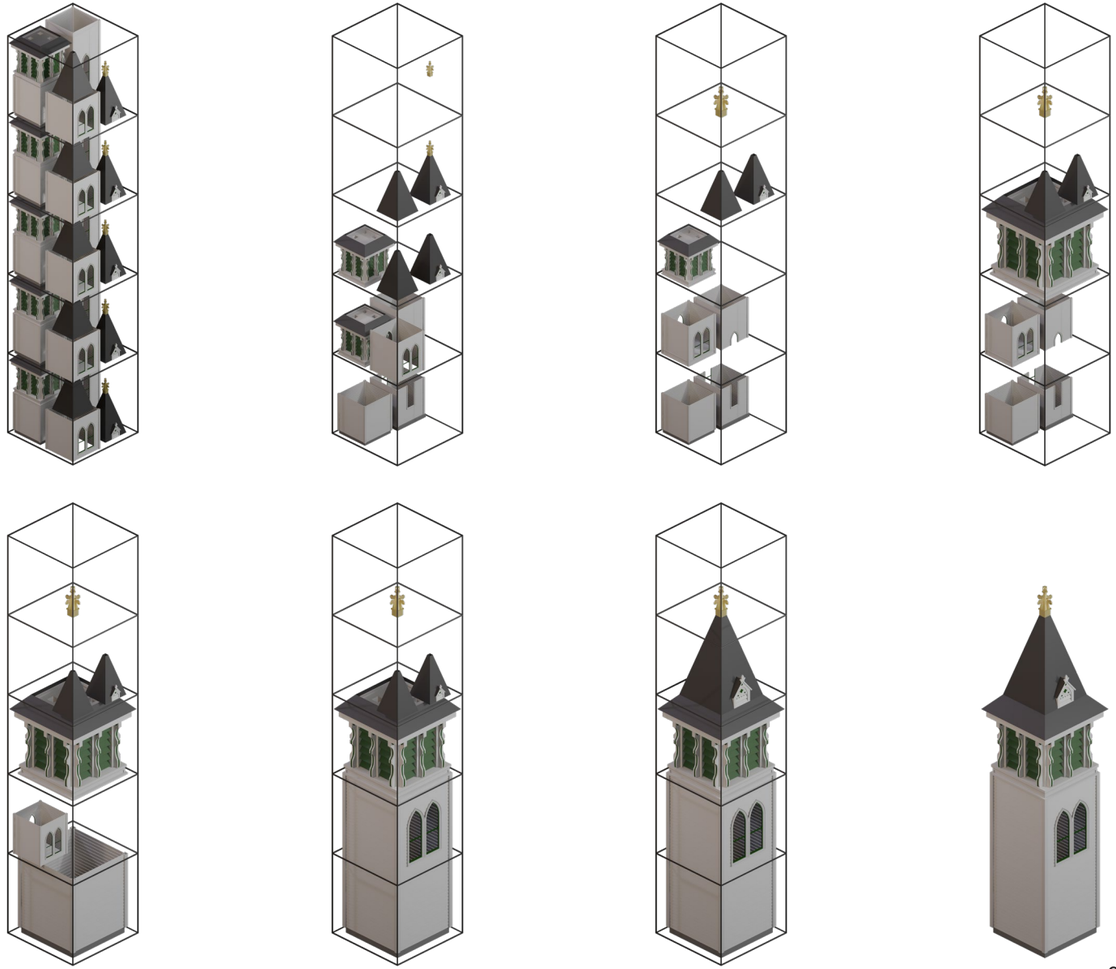
Connections.

Valid connections between pieces are determined by the vertices along a bounding face. If the vertices match then the cube is a valid match. Not all faces have rotationally symmetrical vertices, so connections can also depend on the rotation of both cubes.

Once a cube is selected for a space all invalid options are removed as neighbors, since the vertices will not match up.

An Example Collapse.

To illustrate how the wave function collapses a limited number of tiles were selected. These are circled with a dashed line on page 8, and shown in superposition to the right. At each step entropy is removed from the system as constraints narrow.



02 What the Camera Sees

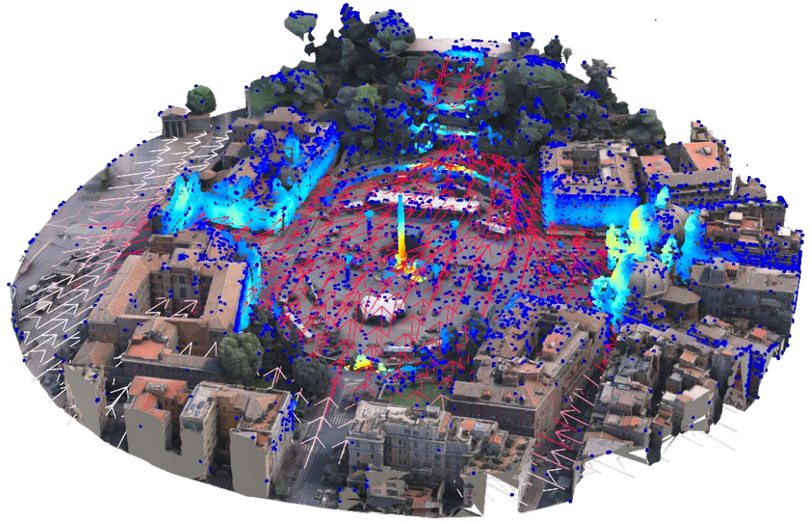
Measuring visual attention through 3D reconstruction of tourist photos.

Year: 2025
Team: Jose Venegas Barriga, Weilun Chen, Nishitha Parakullthil
Tools: COLMAP, GLOMAP, Rhino3D, Blender

As people increasingly experience places through photographs shared online, we wanted to explore how these images shape the perception of public space.

Using public image datasets, we reconstructed three urban squares through photogrammetry and analyzed the resulting camera positions, directions, and visibility patterns. By combining COLMAP and GLOMAP workflows, we generated point cloud models, camera frustums, and visibility fields for each site.

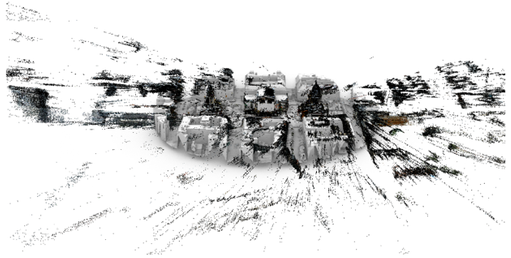
These datasets allowed us to identify both where photographs are most commonly taken and which areas of a space receive the greatest visual attention. The resulting methodology demonstrates how photogrammetry can be used not only to reconstruct space, but also to understand how people collectively document and experience it.



Making a Point Cloud; COLMAP or GLOMAP?

COLMAP excelled at accurate models, but had a much longer compute time. COLMAP displayed was a tendancy to overfit data in symmetrical scenes or when an object looks similar from multiple sides (see red points below).

GLOMAP was much faster, and was including more of the reference images, though many had unrealistic camera transforms. GLOMAP would also mix scales in a single point cloud (see red points below).

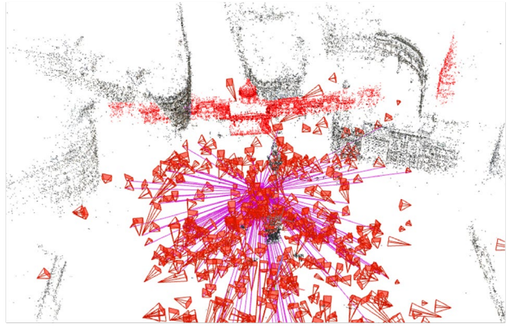


COLMAP algorithm

1077 images used / 16.5 hour compute

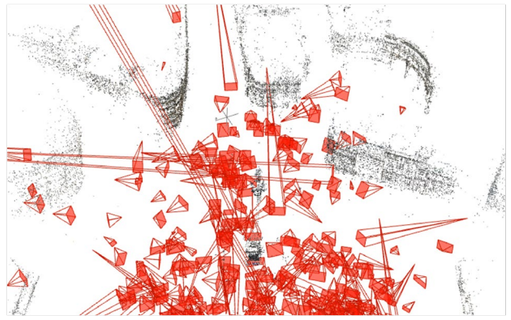
GLOMAP algorithm

1122 images used / 2.75 hour compute



COLMAP algorithm

Data Overfitting

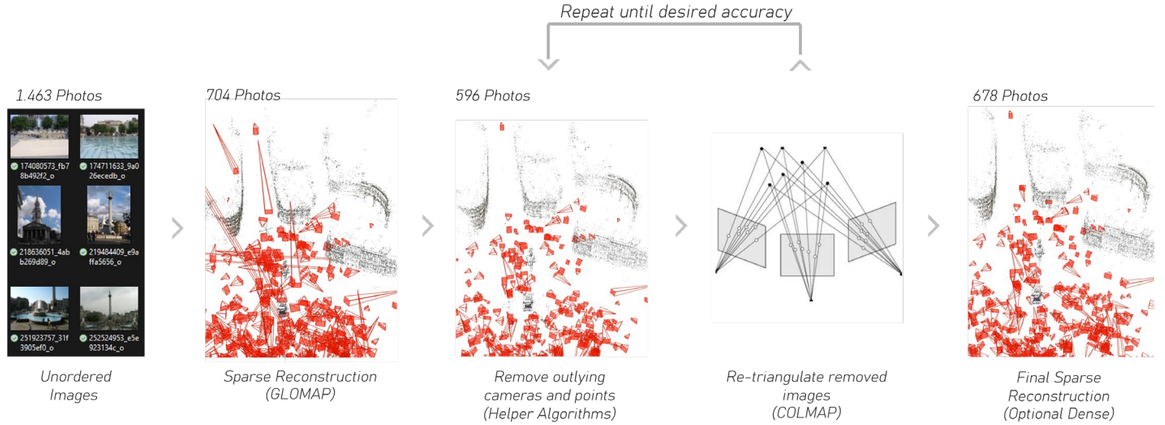


GLOMAP algorithm

Cameras Present with Unrealistic Transforms

Final Workflow.

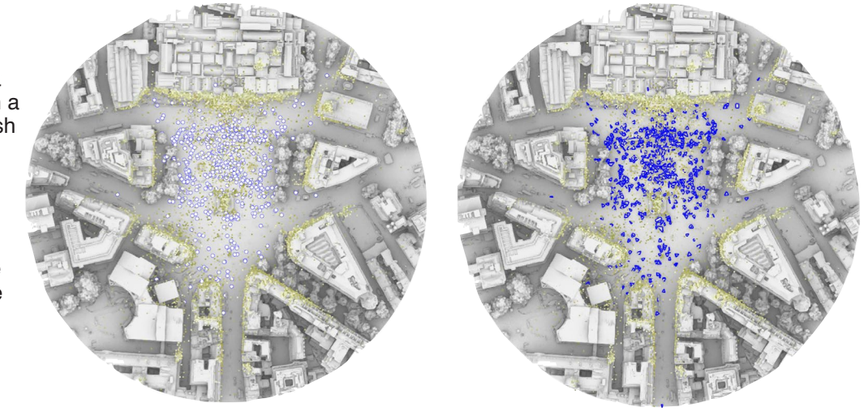
In order to capture the strengths of each algorithm we used both GLOMAP and COLMAP in the final workflow. An initial fast pass was done in GLOMAP, outlying images with high error were then removed and possibly re-placed with the accuracy of COLMAP.



Modeling.

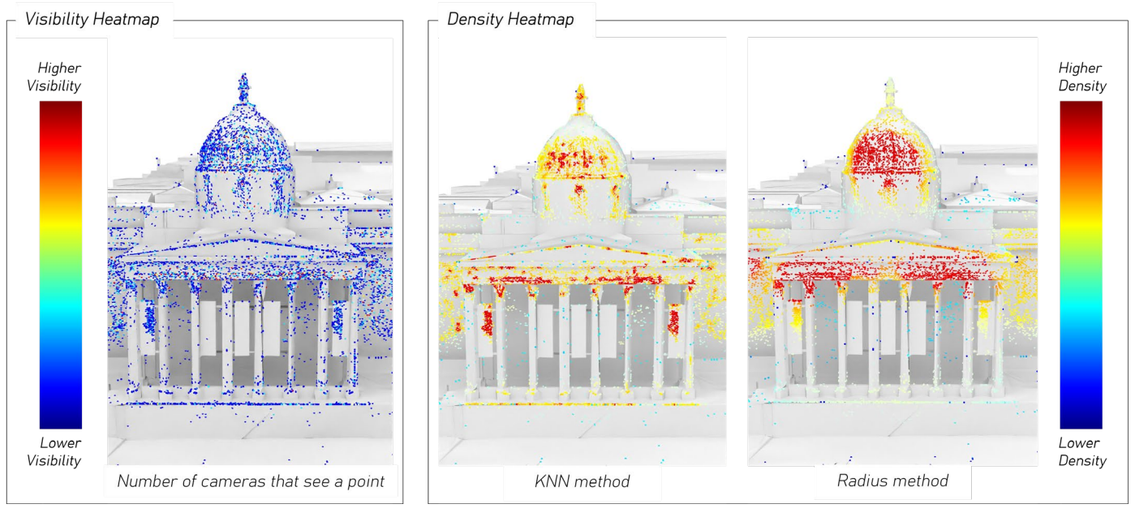
The point cloud and camera positions were overlayed on a Google Maps extracted mesh of the location.

While lower resolution than the final point cloud it provides a ground truth for the output data. This can be used to see where there are areas of hyper-attention or blindspots.



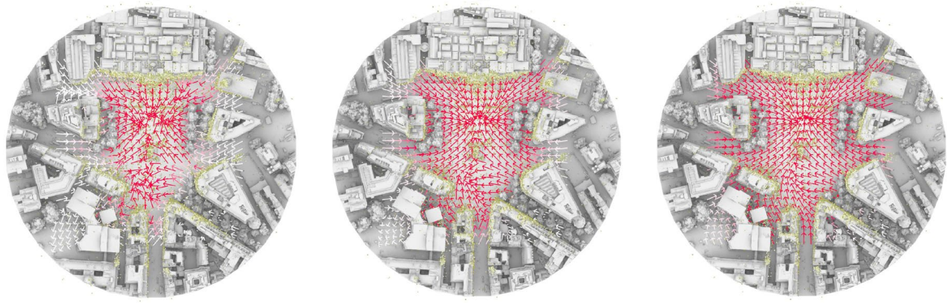
Determining the “Most Visible” Areas.

Three possible methods were used to make a heatmap of the “Most Visible” areas from the dataset. KNN and Radius methods outperformed point visibilty. Point visibilty performs worse, like noise, becasue not every lighting condition would highlight the same feature on the building. Thus, most images wouldn’t register connections to every visible point.



Camera Position and Direction.

After obtaining a point cloud model, we could make a 3-dimensional vector field representing the average camera position at any point in the space. Using different influenece parameters highlight local vs global trends.

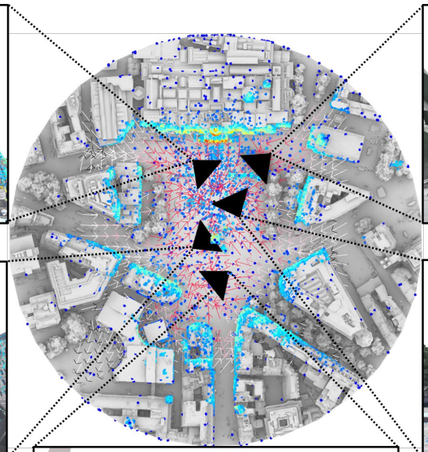


The Camera Agent.

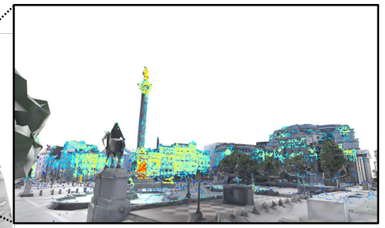
Using the calculated camera vector averages and choosing a data overlay, we can move an agent through 3D space. The below images are agent views at random points of the sqaure, with settings set by the vector map.

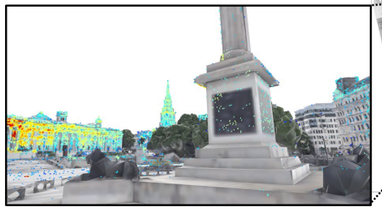
Plaza Looking SE.

Map with Camera Frustums.



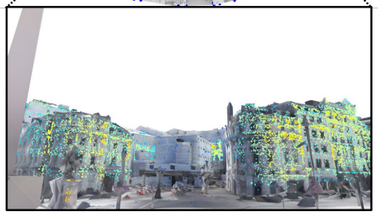
Plaza Looking SW.





Tower Base.

National Gallery from Tower.



Path Preview.

03 AutoVisualizer

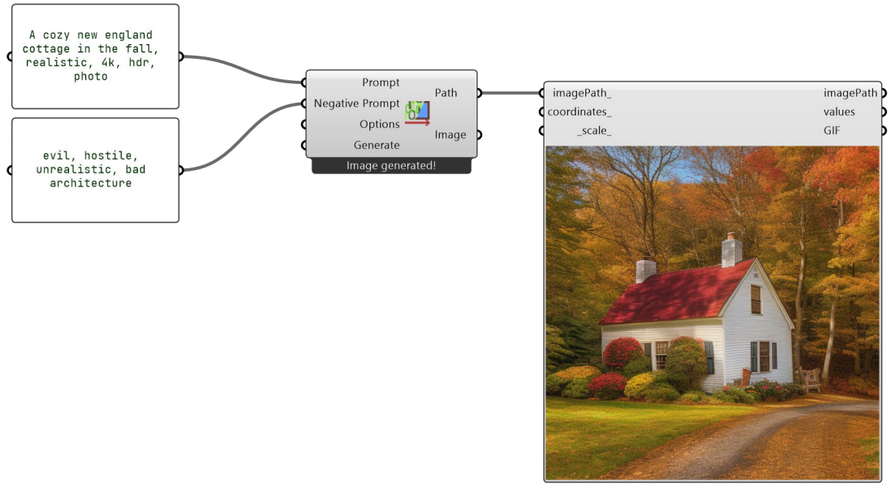
Integrating AI for Grasshopper.

Year: 2023-2024
Tools: Grasshopper, Automatic1111 API, C#

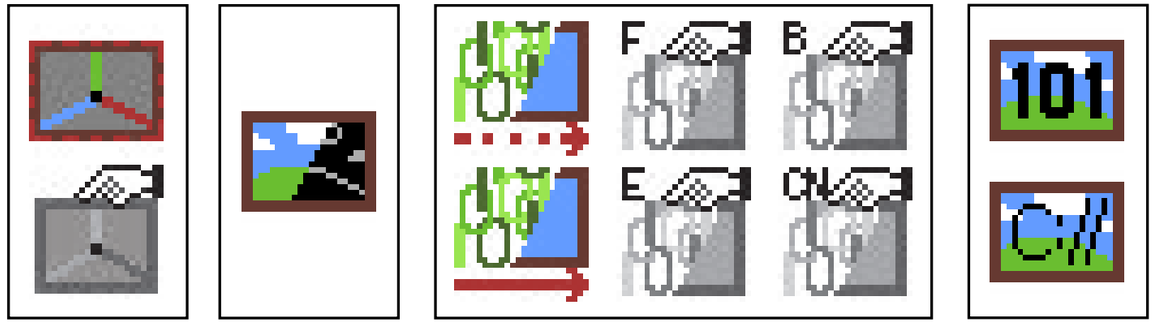
AutoVisualizer is a plugin I designed to bridge architectural design and generative AI, offering seamless integration with the Automatic1111 Stable Diffusion API. This tool allows users to generate captivating AI-driven images directly within Grasshopper, providing both ease of use and fine-grained control over the image output.

With AutoVisualizer, designers can streamline their workflow, moving effortlessly between algorithmic modeling and visualization. The plugin gathers user input, formats it for the API, and processes the results in real-time. The generated images can then be customized, manipulated, or saved for immediate use, all within the same environment.

AutoVisualizer not only simplifies the image generation process but also serves as a stepping stone for exploring the potential of AI-driven creativity in architectural design.



Current Plugin Components.



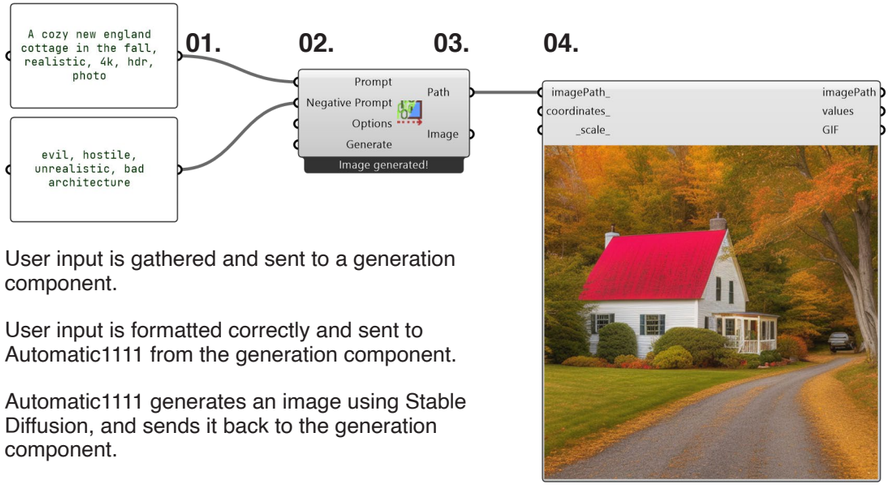
Capture View Components

ControlNet Components

Image Generation Components

Image Viewing Components

How It Works.



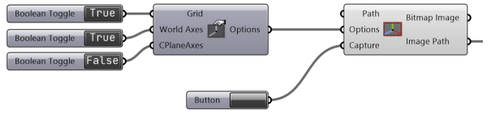
01. User input is gathered and sent to a generation component.

02. User input is formatted correctly and sent to Automatic1111 from the generation component.

03. Automatic1111 generates an image using Stable Diffusion, and sends it back to the generation component.

04. The image filepath or raw data is sent to be modified as seen fit.

Plugin Usage Examples.

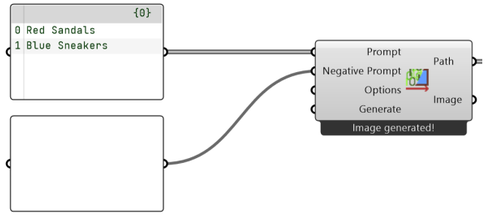


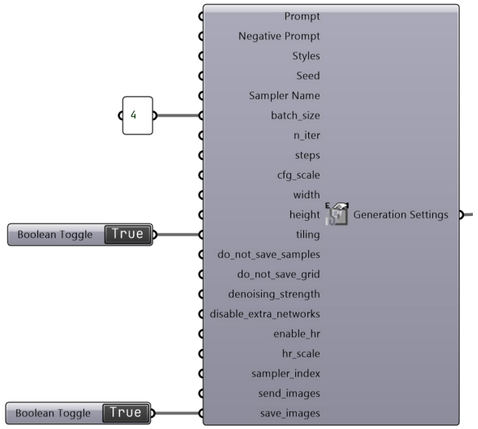
Screen Capture.

No more exporting images to use in AI generation! Capture view components make selecting your screen a breeze.

Lists.

Generation components work with multiple Grasshopper input types. If given a list it will generate images for each prompt.





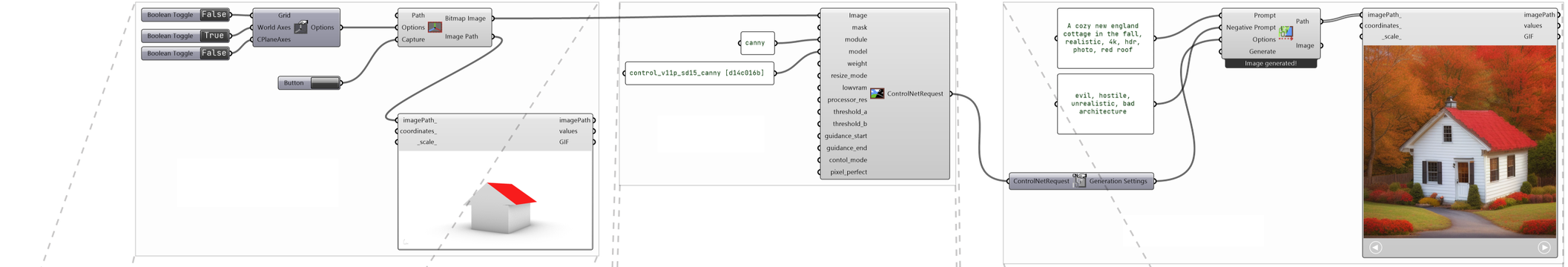
Options.

Four options components allow varying levels of customization to the generation settings.

This one, with a batch size of four, provides us with 4 images to pick between. It also makes the resulting images tileable and has Automatic1111 save them.

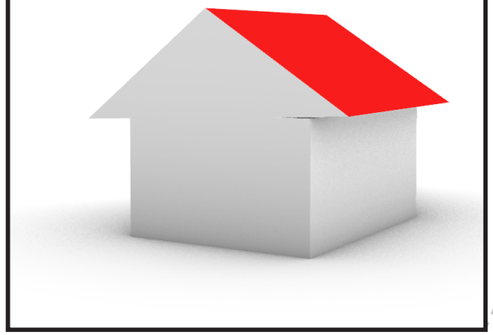
Putting It All Together.

In this example a simple model in Rhino can be visualized with only a few nodes.



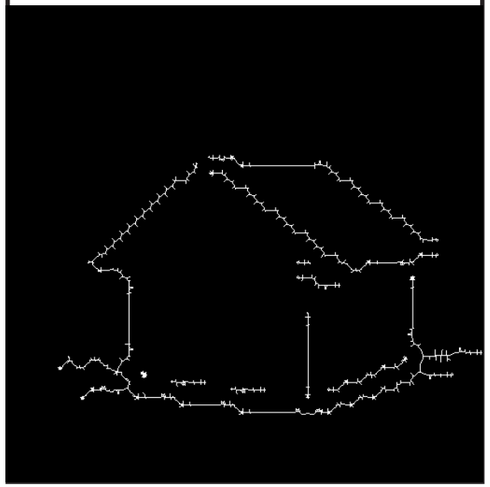
Capture View

The viewport is captured by the component, with settings for hiding the viewport grid, world axes, or CPlane axes.



ControlNet

ControlNet interprets the captured image to create an influence map for the output. Here, canny edge detection was used.



GenerateASync

Final output image, combining the text prompts and the edges defined with ControlNet.

04 Caustic Calculations

Evaluating architectural caustics through automated simulation.

Year: 2026
Team: Weilun Chen, Nishitha Parakullthil, Lucas Tanner
Tools: C++, Python, Blender, CNC Mill

Inspired by the work of Mark Pauly, but noticing few examples of caustics being used in architecture, we wanted to understand what prevents the technique from being applied at larger scales.

To investigate this, we developed an automated workflow that generated and analyzed thousands of simulated caustic configurations using Blender, LuxCore, and custom Python scripts. Variables relating to generation, fabrication, and installation were systematically adjusted to determine which had the greatest impact on the final projected image.

The resulting dataset revealed which factors most strongly influence caustic quality and which tolerances can be relaxed without significant visual degradation. Selected geometries were CNC milled and tested physically to validate the simulation results.

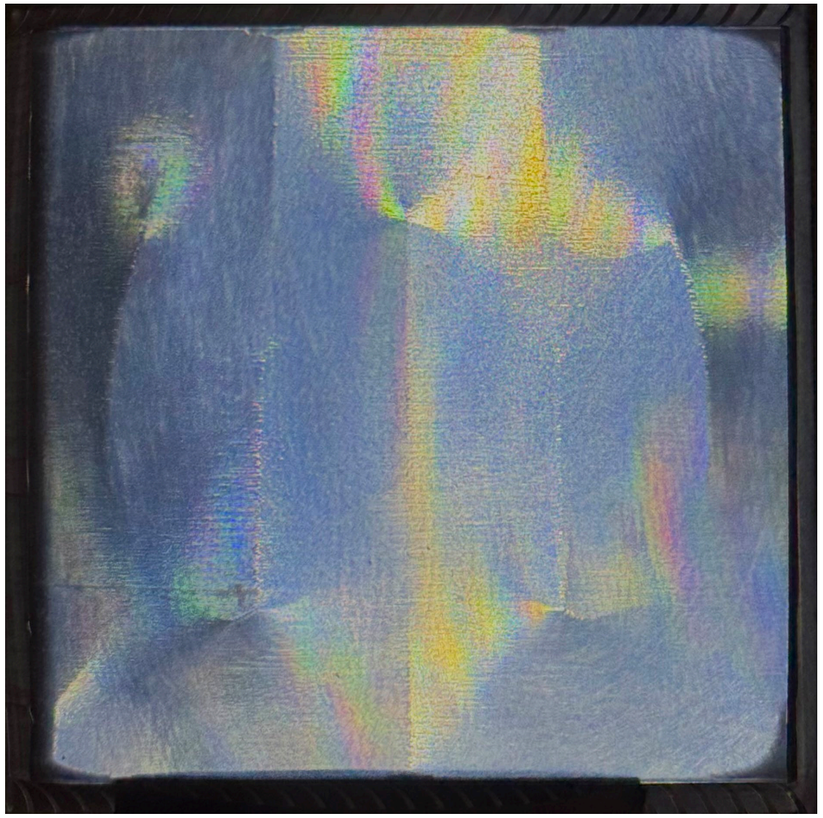
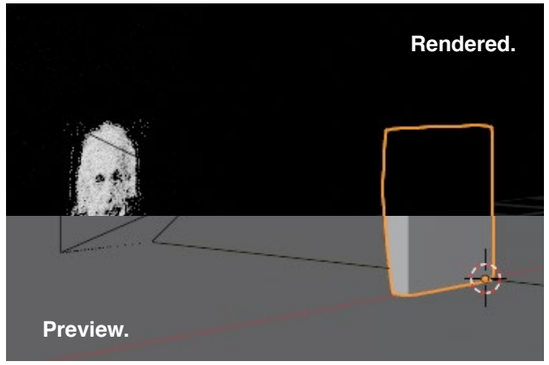


Image of caustic lens taken from focal plane.

Experimental Setup.

In order to compare the effect of certain variables on the final quality of the caustic we ran each combinaiton through a third-party renderer for Blender, Luxcore. Known for its physical based and accurate rendering we knew we could trust the results. The rendered image was then differenced against the input to determine how “different” the result is from the desired image.

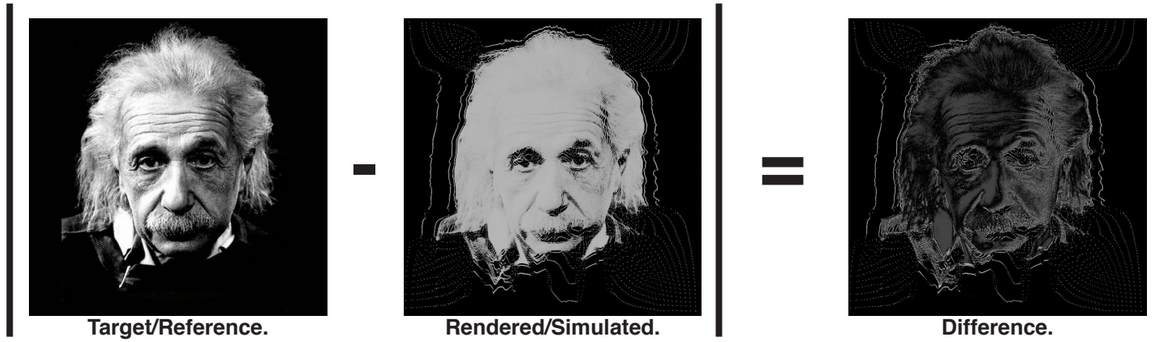
This was all done automatically by a script to ensure that for each mesh and independent variable the rest is the same.



Difference Calculation.

Each variable’s effect on the final quality of the image was determined by taking an absolute difference of pixel brightness between the original image nad the render. This was then normalized based on the maximum possible diference in pixel brightness based on the input image and used for comparison.

We calculate the Mean Pixel-wise Normalised Absolute Error (MPNAE), as seen below, in order to get a relative change in fianal image based on the input image, and a maximum possible difference. In MPNAE, N is the total number of pixels, I_(r_i) is the reference irradiance of pixel i, and I_i is the simulated irradiance.



Target/Reference. Rendered/Simulated. Difference.

$$MPNAE = \frac{1}{N}\sum_{i=1}^{N}\frac{\left|I_i - I_{r\_i}\right|}{\max\left(255 - I_{r\_i},\, I_{r\_i}\right)}$$

Variable Testing.

To determine which variable has the greatest effect on the final caustic quality we narrowed our search down to three areas, each with it’s own set of variables:

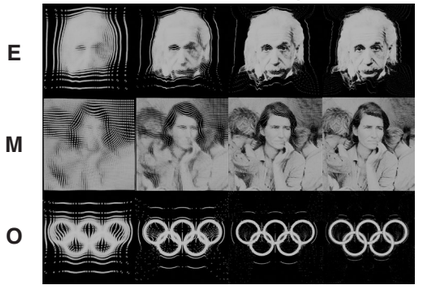
During Generation:
- Mesh Resolution
- Distance to focal plane

During Manufacture:
- ZMap tool stepover
- ZMap tool size
- Polishing pass tolerance

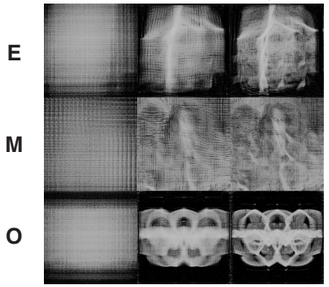
During Installation:
- Installation offset
- X & Y incident angles

Each variable was tested across three different types of images, to see if some disproportionatly affected images of different types (i.e. low contrast vs high contrast).

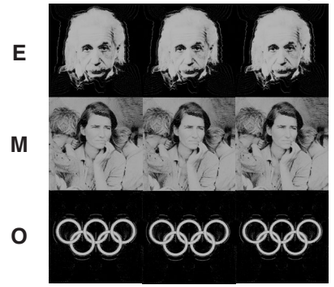
Effect of Mesh Resolution on Caustic Quality



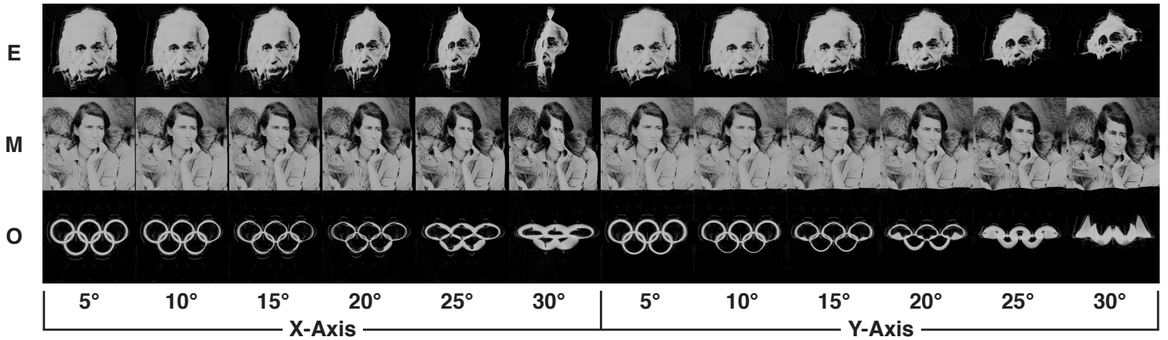
Effect of ZMap Stepover on Caustic Quality



Effect of Focal Plane Offset on Caustic Quality



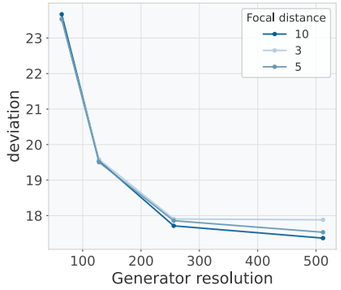
Effect of Incident Light Angle on Casutic Quality

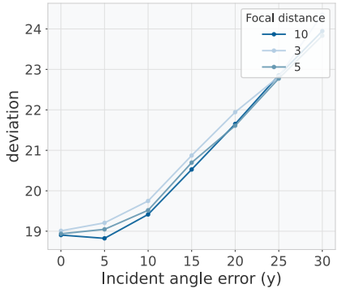


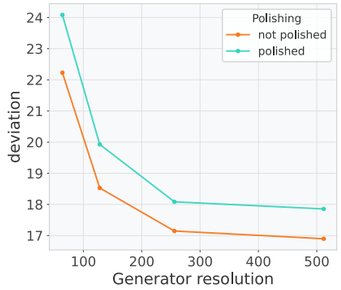
Data Analysis.

Having been able to visually see how the caustics were effected it was time to see which variables had the largest effect on caustic quality. Some results can be seen on line charts, like the influence greater resolution has on reducing deviation, with better results at longer focal lengths. The nearly linear effect of incident angle on deviation increase, or the gain from milling and then polishing (not polished is in reference to the baseline).

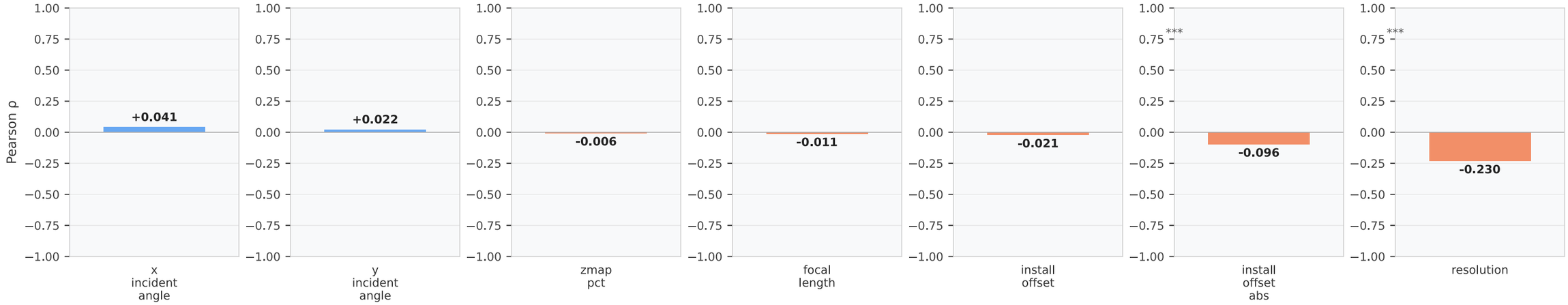
These results are also corroborated by a pearson orrelation graph, which shows positive or negative correlated effects between variables, in this case all relwating to the final simulated deviation. Our strongest result was the generator resolution size. ZMap and polising were grouped into one step which explains the near zero correlation, that after polising most were close to original quality.





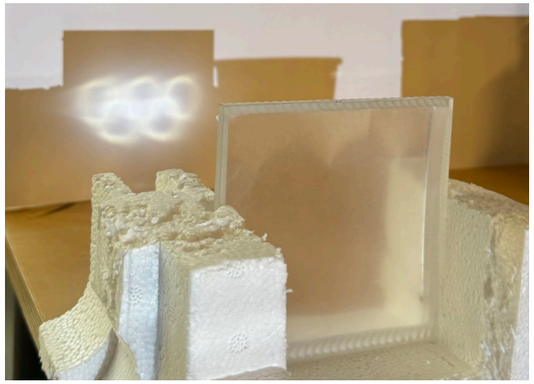


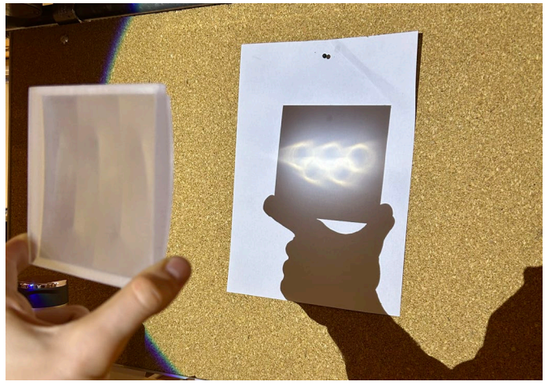
Pearson Correlation between Variable and Design.



Test Manufacture.

In the end two seperate example peices were milled from the same stock of acryllic and used as physical examples. These were not measured against the simulated results. Though in changing certain factors like offset distance or incident light angle, we observed similar results to our virtual model.





05 Redefining Home

Shaping homes through conversation.

Year: 2022-2023
Team: Lucas Kamal, Talia Mamayek, Max Wojtas
Tools: Revit, Illustrator

This project began with conversations. Our team engaged with public housing residents, college students, and local stakeholders to understand the evolving needs of modern families. These insights revealed a demand for housing that supports diverse living arrangements, fosters community, and feels like home.

The resulting design utilizes a point-block typology to maximize space efficiency while creating a warm, residential atmosphere. By incorporating flexible layouts and shared spaces, the project offers multi-generational and nontraditional families the opportunity to connect while maintaining privacy.

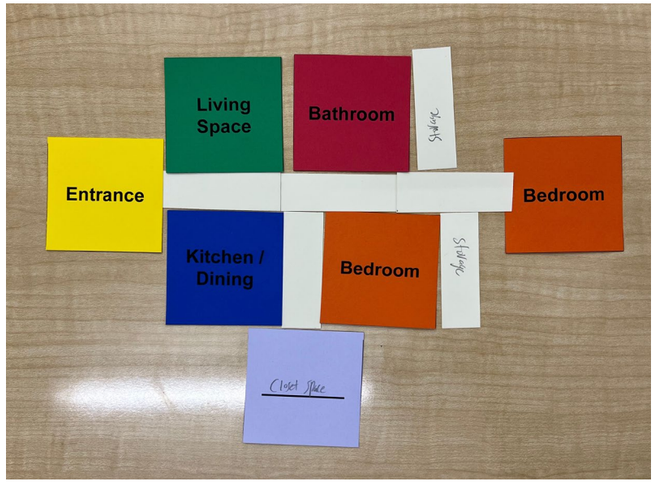
Rooted in user feedback, Redefining Home demonstrates how thoughtful, collaborative design can shape housing solutions that prioritize both individuality and community.

Render by Lucas Kamal

A Layout That Works.

Focus groups were held in order to gather data about what people like and dislike about their current living arrangements. They were then tasked with creating ideal apartement layouts for different living situations.

Time spent in each location, as well as the importance, sentiment and personal experience were all recorded during the focus groups as well.



Above: An example of one participant's ideal two-bed, one-bath apartment

Data Analysis.

Popular Connections.

The most popular connections present in the participants' layouts coincidentally align with a gradient of spaces from most public to private.

Waking Hours.

Where time is spent is important. These are the spaces with the highest average use per week among participants.

What's Important.

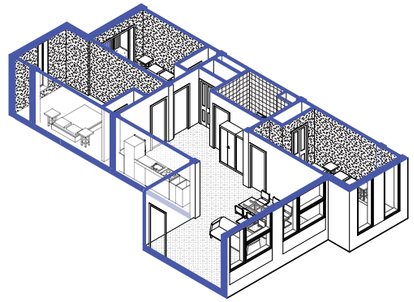
Some spaces, while used less, are valued higher. This is based on average personal perceptions, rating spaces on a scale of 1-10.

Entrance to Kitchen	Kitchen to Living	Living to Bathroom	Bathroom to Bedroom
Bedroom 19 hrs	Living 14.5 hrs	Kitchen 14 hrs	Workspace 9.75 hrs
Bedroom 9.6	Kitchen 9	Bathroom 8	Laundry & Living 6.7

Creating Typologies.

After data analysis was completed, three apartment types were created. Our building has single-bed, three-bed, and five-bed apartments. The latter is designed to be for multigeneraitonal families or co-living.

These aimed to address wants and needs mentioned in the focus groups, like private access to laundry and workspaces seperate from the normal living space.

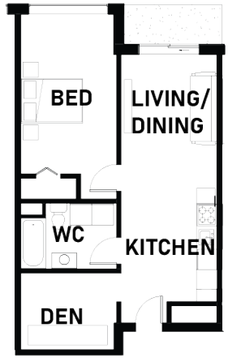


Three-Bed Apartment.

Five-Bed Apartment.

KEY PLAN

Single-Bed Apartment.



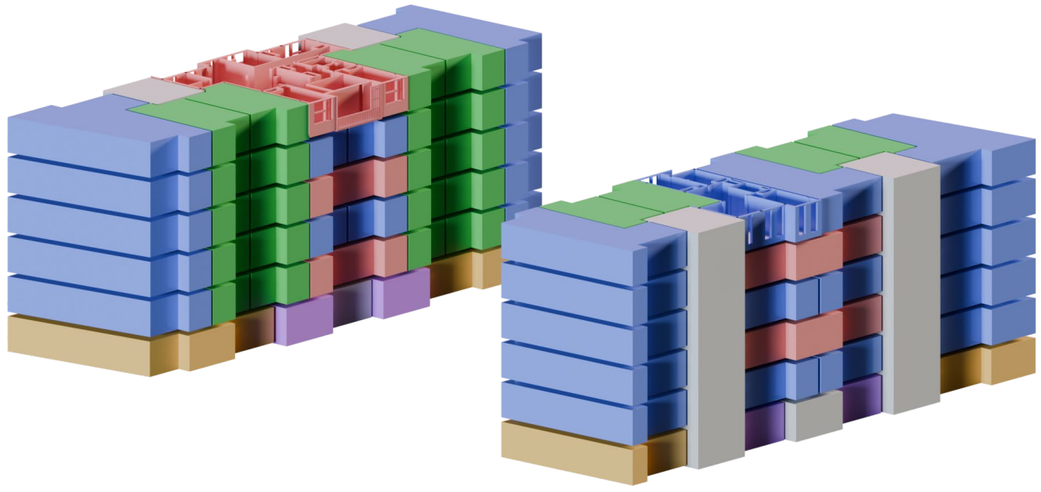
KEY PLAN

Point Access Block.

Integral to the design of our system is the use of point access block. This is a form of building access defined by a single stair servicing multiple floors of apartments. In our design, to conform with second egress laws in the U.S. the elevator is fire rated, and seperate from the exit stairs.

Typical building floorplan.

Building layout.

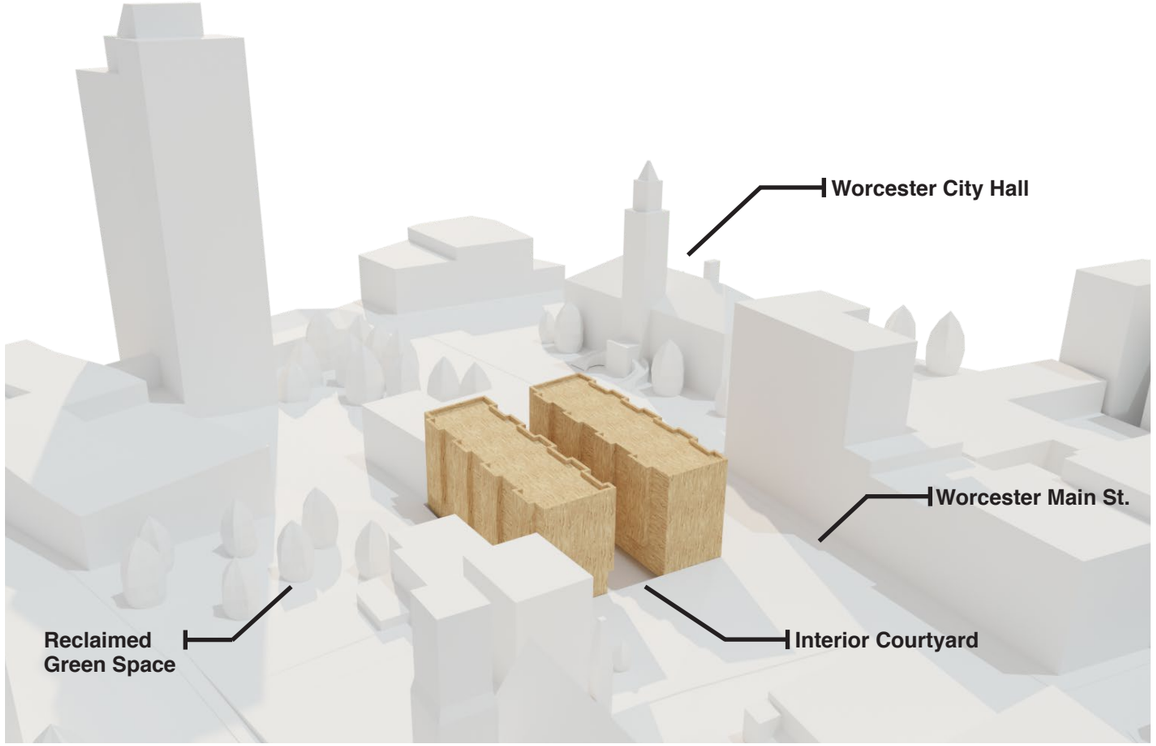


Key: Single-Bed, Three-Bed, Co-Living (5-Bed), Community Space, Retail

Scaling the Design.

Point access block signifigantly reduces the area used by circulation in the building, by not needing hallways that span the entire length. Instead, smaller, more private landings, shared by just four units create an urban “front porch.” This design choice not only fosters a sense of community, but also enables cross-ventilation in the units; both the 3-bed and 5-bed layouts span the entire floorplate for optimal airflow and natural light.

Located in Worcester, MA, the proposed building leverages data-driven design to enhance urban living. Community and retail spaces on the first floor encourages the use of third spaces and neighborhood engagement. An interior courtyard and reclaimed green space further occupant well-being and sense of place.



06 Shaping Code

Experiments in custom slicing.

Year: 2023-2024
Tools: Grasshopper

As both a makerspace enthusiast and experienced 3D printer user, I've long used GCode prepared from slicers like Cura. However, by creating custom slicing algorithms tailored to specific fabrication methods, I've been able to go beyond what traditional slicers can do.

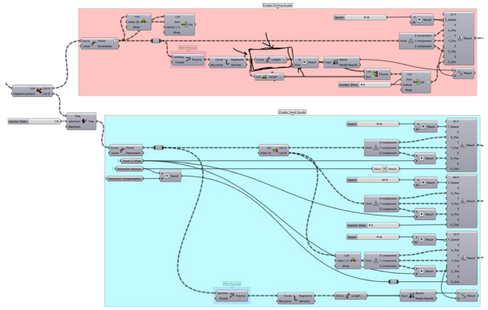
This project showcases experiments in custom slicing, spanning diverse applications from clay 3D printing to the precision of pen plotting in 2.5D. Each algorithm is designed to optimize fabrication outputs, blending computational design with the craftsmanship of digital fabrication.

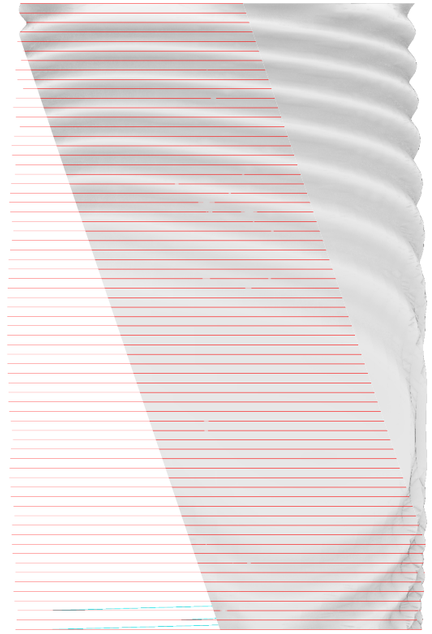
Shaping Code demonstrates how thoughtful scripting can transform digital ideas into tangible, meticulously crafted forms.

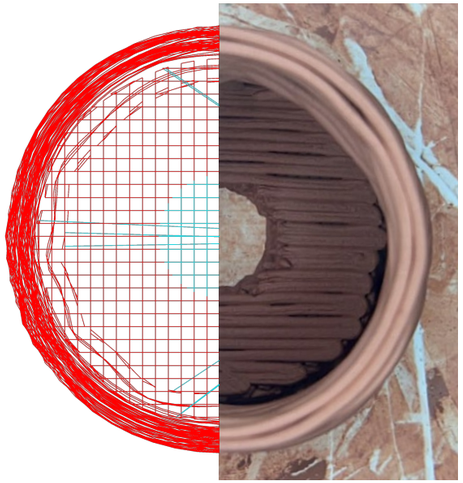
3D Printing.

Through multiple iterations of trial and error, and referencing the GCode command library, a working script was created to output GCode for a variety of 3D printers. Options for feed rates, wall thickness and whether to spiralize the print were all implemented.

The main medium used for this was clay on a delta printer, but plastic was also experimented with.



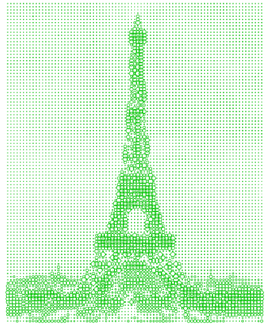




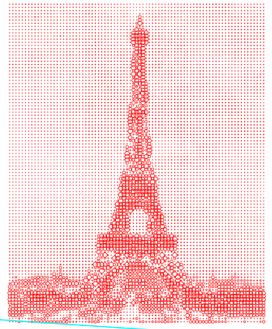
Pen Plotting.

Modifying the 3D printer GCode script for work on a 2D plane allows for pen plotting or brushwork.

Design.



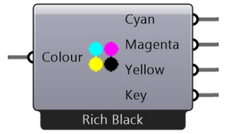
Path Preview.



Plotted.

Split CMYK.

Grasshopper does not come with a node that splits a color into CMYK values, which are often used for printing. With my experience making AutoVisualizer I created a component to get these values. It allows for easier creative color plotting.



07 Reframing Color

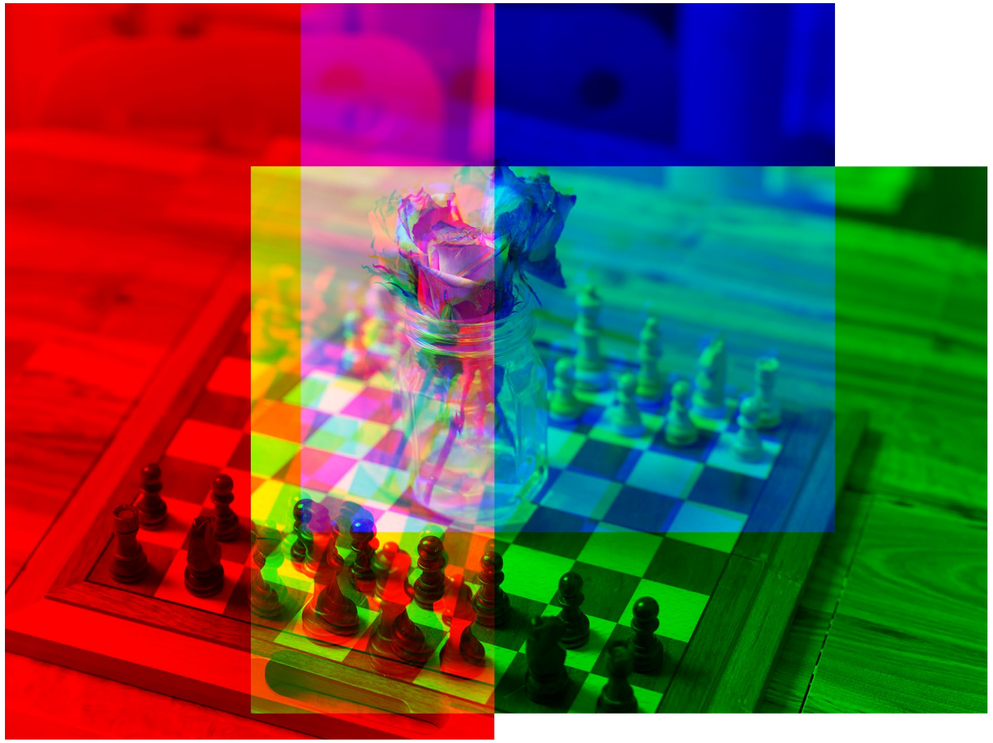
Shooting with the trichrome process.

Year: 2023
Tools: Digital Camera, Trichrome Filters, Photoshop

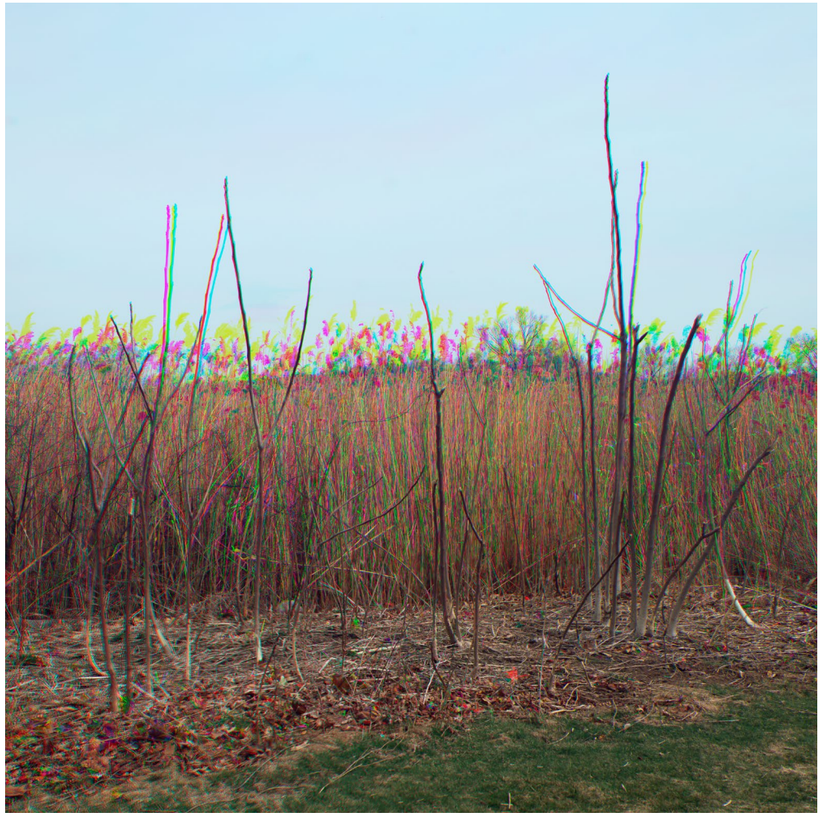
The trichrome process, rooted in early photographic methods, involves capturing three separate images using red, green, and blue filters and then combining them to create a full-color photograph. By revisiting this historical technique, I explored the interplay of color, light, and composition in a uniquely modern context.

Using a digital camera and custom filters, I embraced the challenges of aligning and merging these filtered captures into a cohesive whole. Each image became an experiment in reconstructing color perception, where subtle shifts in alignment or exposure revealed vibrant, unexpected textures and patterns.

This project bridges analog traditions and digital precision, offering a contemporary take on a technique that reframes how we perceive and compose with color.



“Three Moves Ahead”



“Crowned by Breeze”

08 Drawing Chaos

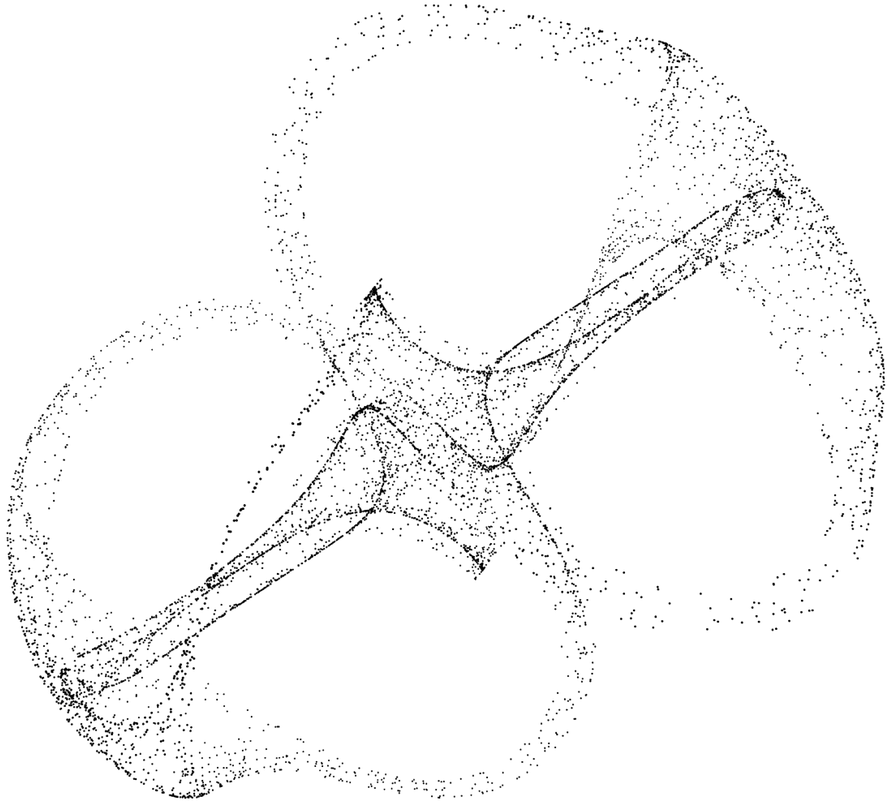
Generating complexity through strange attractors.

Year: 2023-2024
Tools: Blender

When I first encountered the term strange attractors in my senior year of college, I was intrigued by the challenge of visualizing these complex mathematical systems. At the time, I had been exploring Blender's simulation nodes for other projects, and it felt natural to adapt this tool to transform abstract equations into artistic representations.

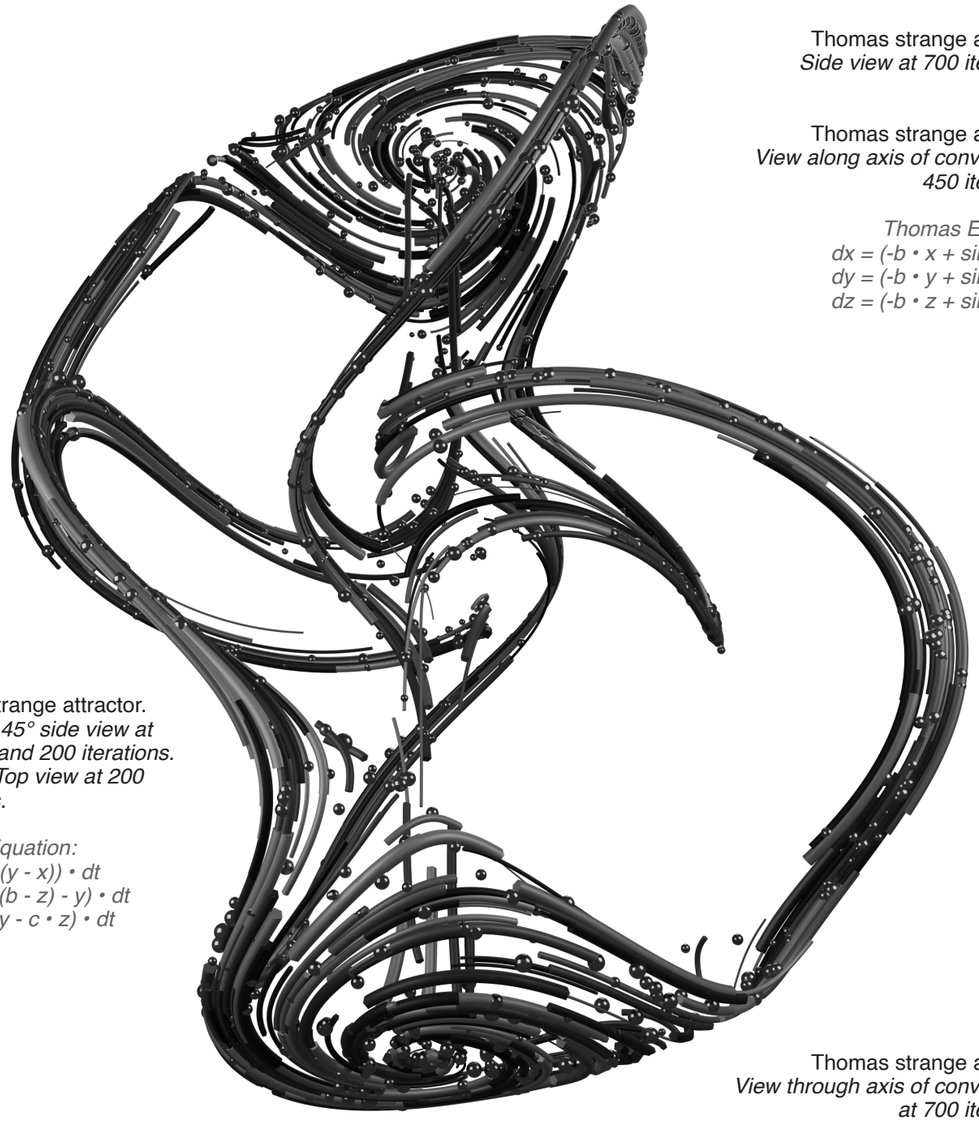
Strange attractors are more than just visually compelling—they serve as a way to represent entropy and the inherent chaos within complex systems. Through iterative exploration, I developed techniques to display these equations in dynamic and engaging ways, blending science and art to reveal the beauty hidden within mathematical complexity.

This project celebrates the intersection of computation, art, and unpredictability, offering a glimpse into the ordered chaos that underpins strange attractors.



P34:
Lorenz strange attractor.
Top row: *45° side view at 50, 100, and 200 iterations.*
Bottom: *Top view at 200 iterations.*

Lorenz Equation:
dx = (a • (y - x)) • dt
dy = (x • (b - z) - y) • dt
dz = (x • y - c • z) • dt

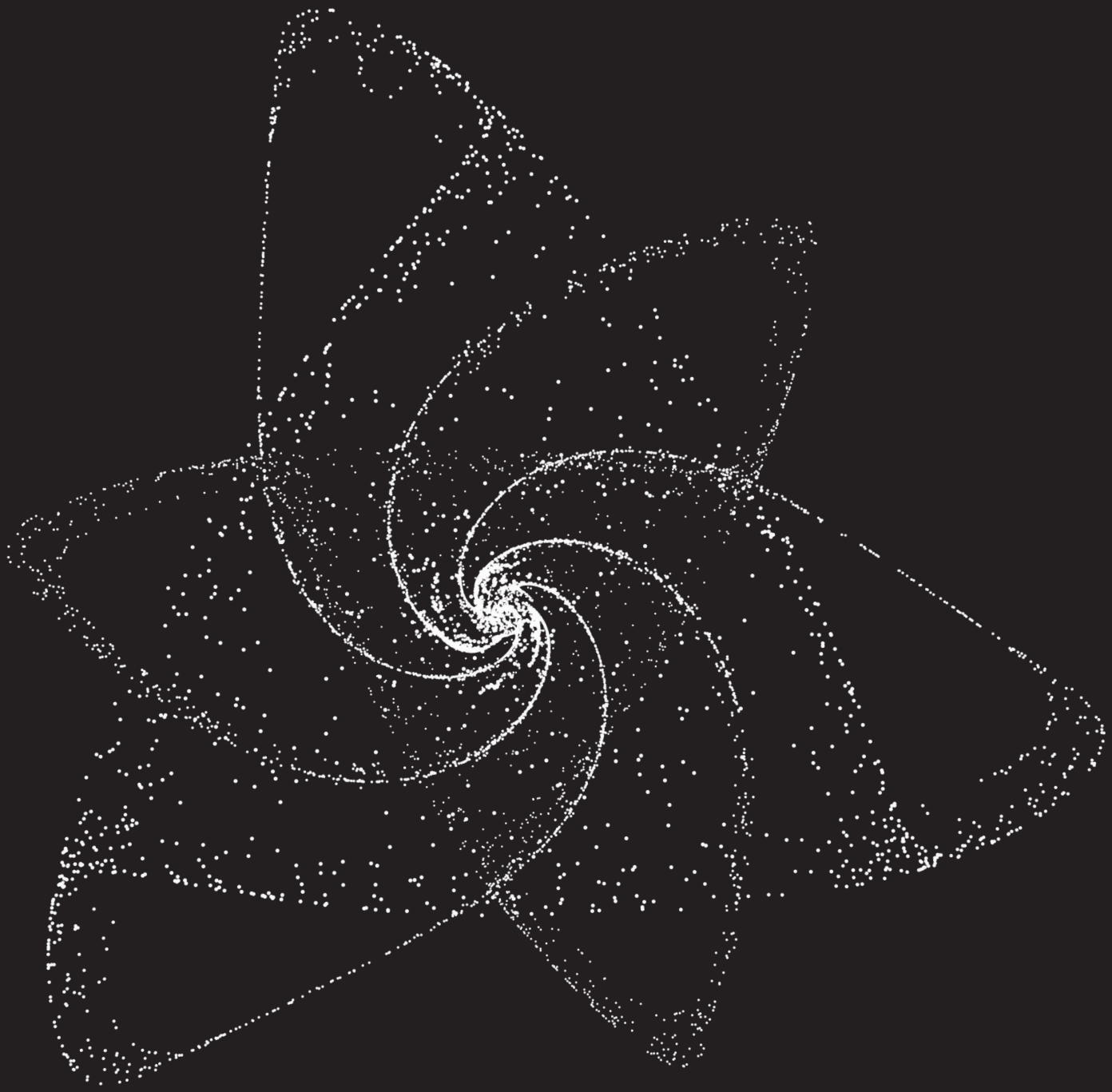


P33:
Thomas strange attractor.
Side view at 700 iterations.

P35:
Thomas strange attractor.
View along axis of convergence 450 iterations.

Thomas Equation:
dx = (-b • x + sin(y)) • dt
dy = (-b • y + sin(z)) • dt
dz = (-b • z + sin(x)) • dt

P36:
Thomas strange attractor.
View through axis of convergence at 700 iterations.



</PORTFOLIO>

Dan Kenerson