

# Web3D 2026

**13-15 October 2026**

**Doha, Qatar**

# Rendering Explicit and Implicit Complex Functions in 4D:

## A WebGPU based approach

**Web3D 2026**
13–15 October 2026 · Doha, Qatar

**Admir Huseini, Art Saiti**

Ss. Cyril and Methodius University, Skopje · Ss. Cyril and Methodius University, Skopje

# Why Complex Functions

1

## Lack of robust, easy-to-use solutions

Most complex graphers are commercial and require code to work. Advanced features can require custom scripts and sometimes, the tasks are CPU bound.  Web based approaches are underdeveloped.

2

## Implicit Functions are absent

Implicit forms of complex equations, such as Riemann surfaces but necessarily limited to them, despite being a topic of wide research have very little resources for visualization



# Why WebGPU

| Factors | Desktop | WebGL | WebGPU |
|---|---|---|---|
| Portability | Generally limited to Vendor / OS | Web Support | Web Support + Desktop through Vulkan |
| Performance | High Performance | Lack of GPGPU without hacks | GPGPU through Compute Shaders |
| Complexity | Varies, but generally on the compex side | Derived from OpenGL | Moderate, uses specific shading language (WGSL) |



# Comparison Between WebGPUComplex and other Variants

| Options | Frameworks and Commercial Software | Online Tools | WebGPU Version |
|---|---|---|---|
| Robustness | Robust if individually implemented | Very Basic | Robust |
| Performance | Sometimes CPU bound | Real Time | Real Time |
| Complexity | Requires Domain Language Knowledge | Basic | Basic, although currently many UI/UX features are missing |



# Our Features

1

## Full 4D Freedom

Despite Complex Functions being inherently 4D objects, many solutions omit the 4D aspect entirely, relying primarily on colouring to convey the 4th dimension. In our case, we use both colouring and free 4D rotation

2

## Flexible Parameters

Things such as scale of the input and output, domain origin, types of rotations (Eulerian or Quaternion) and projection types, such as Riemann or other custom ones allow for a more intuitive understanding of the actual rendered objects.

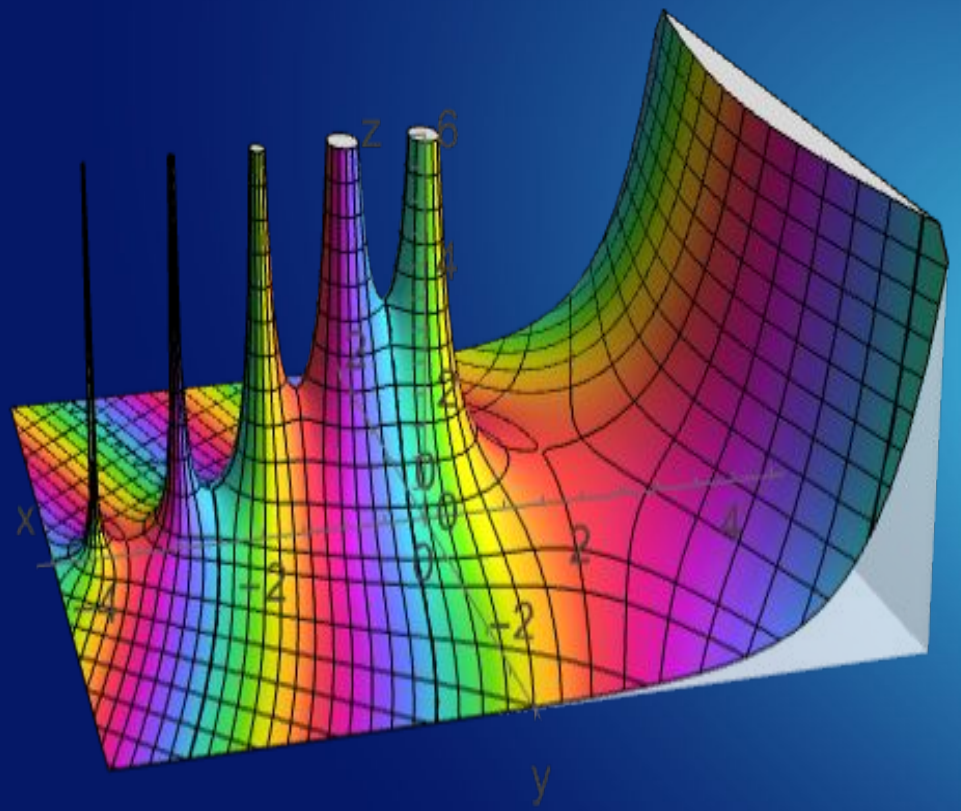
3

## Modular Architecture

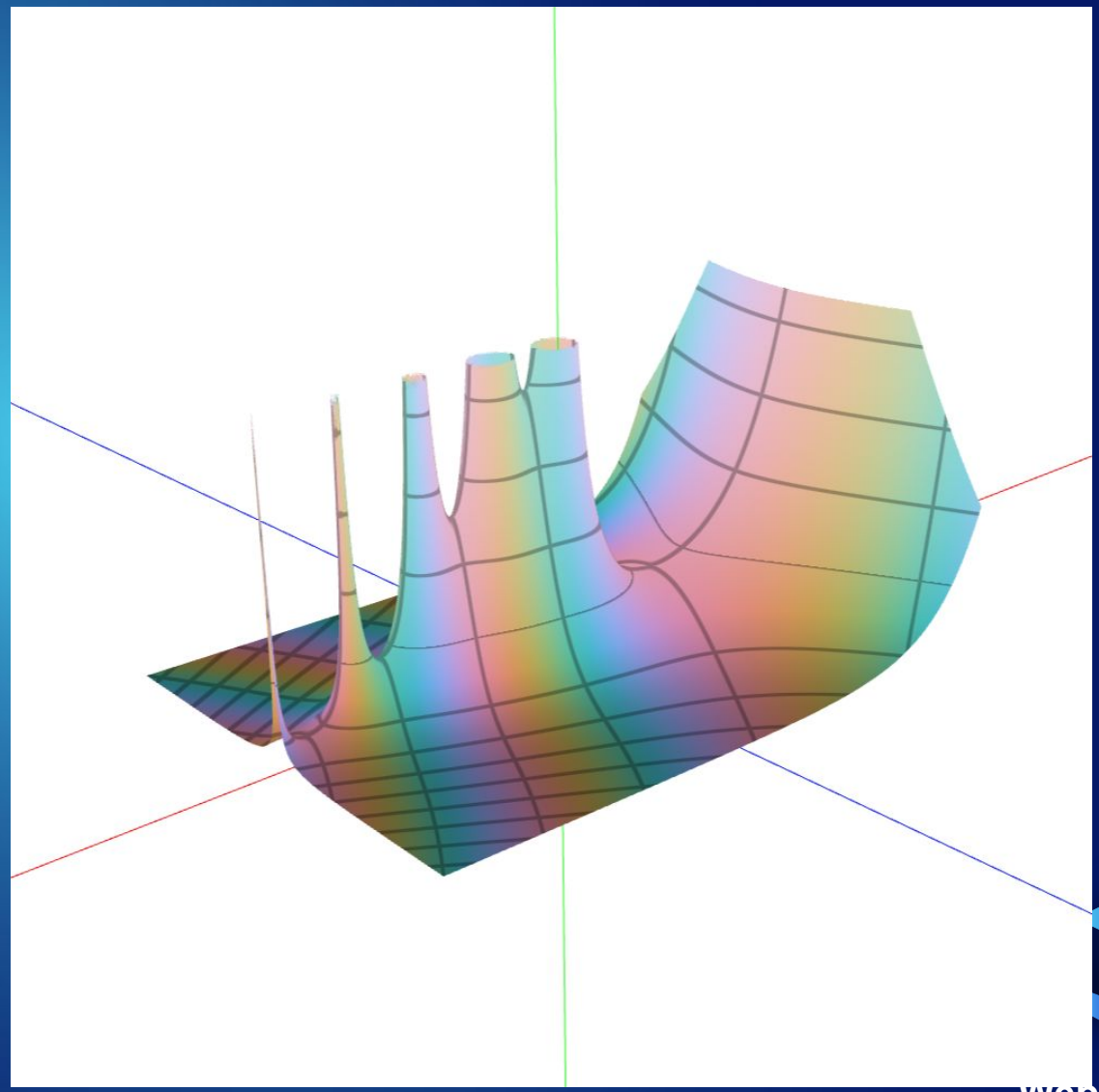
The modular architecture allows for a flexible development approach, For the explicit part, the core code is around 1000 lines, including the shaders and pipeline.



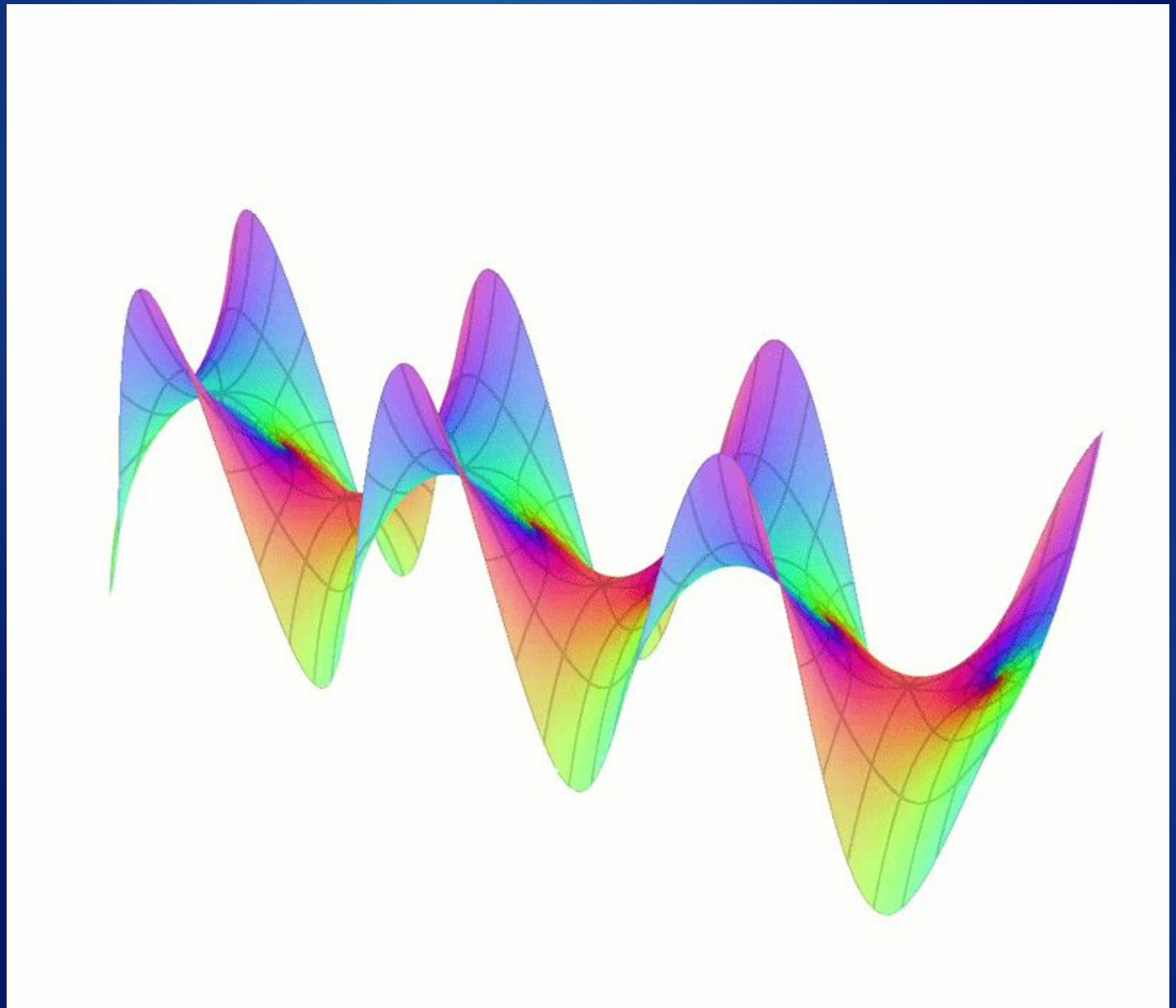
# Gamma function in Wolfram Alpha



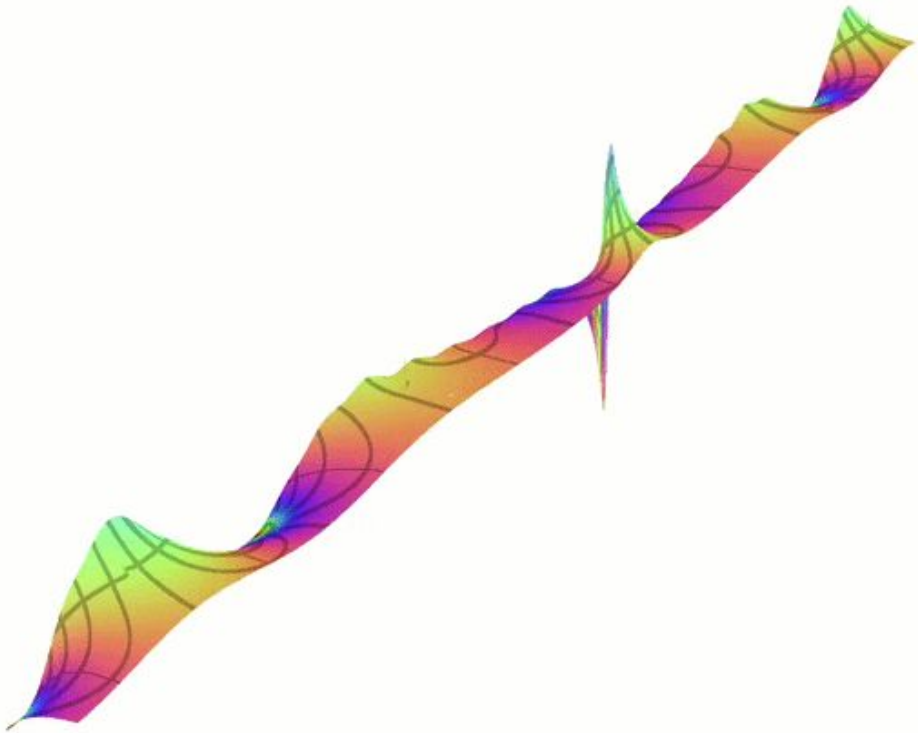

# Gamma function in Our Framework



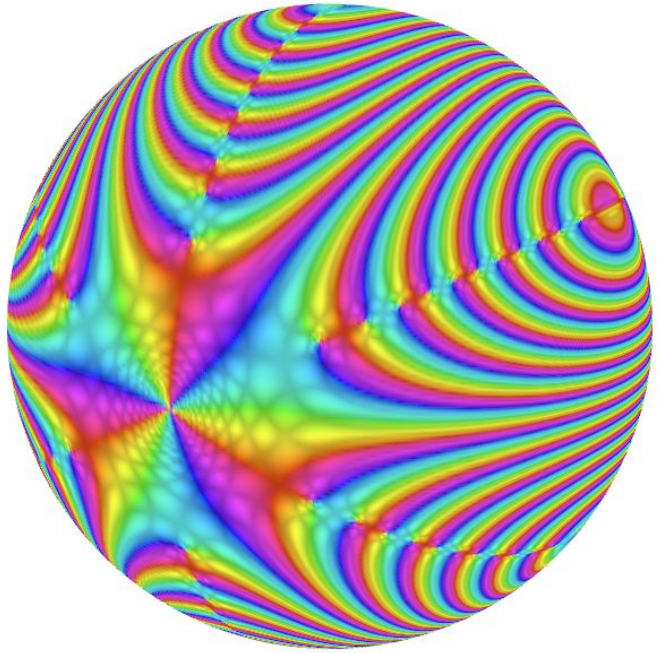
# Complex Sine with 4D Manipulations

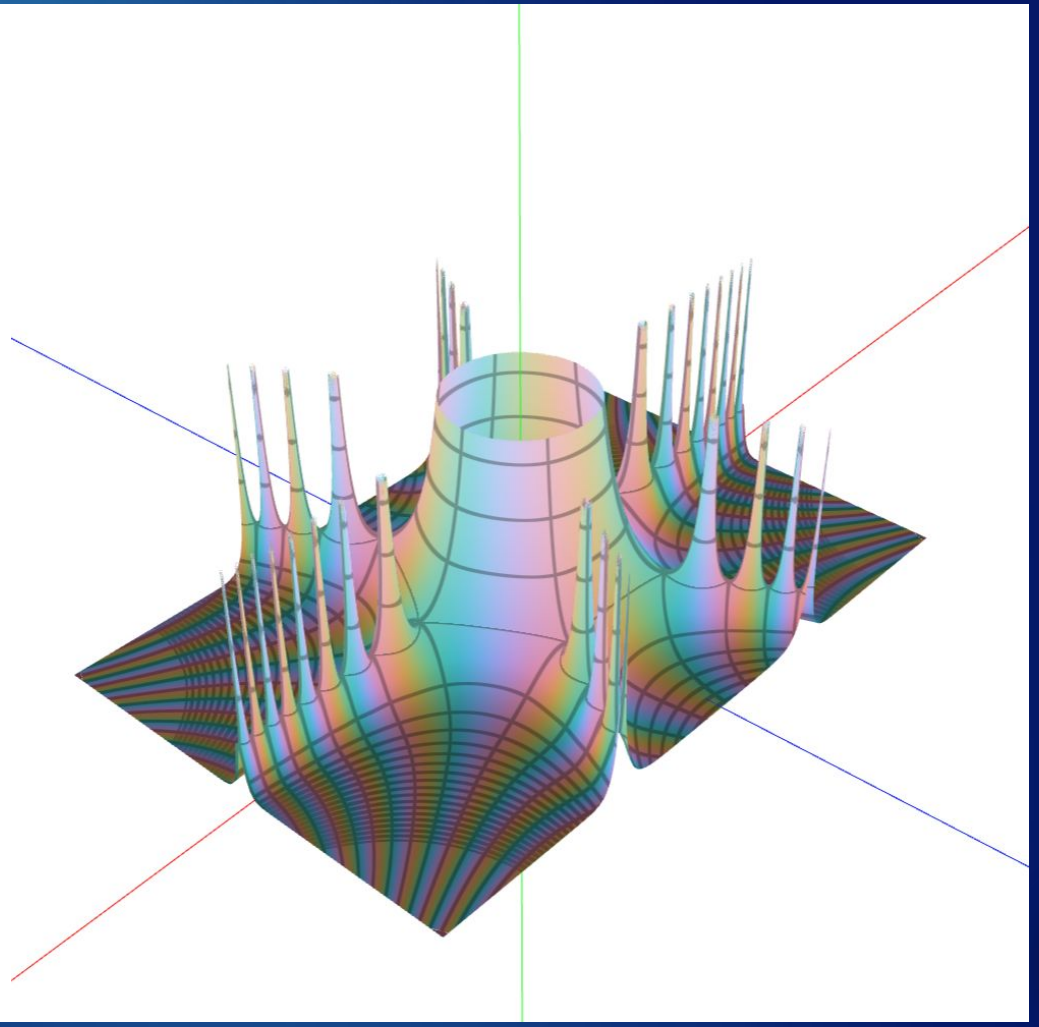


# Zeta Function Highlighting Movement

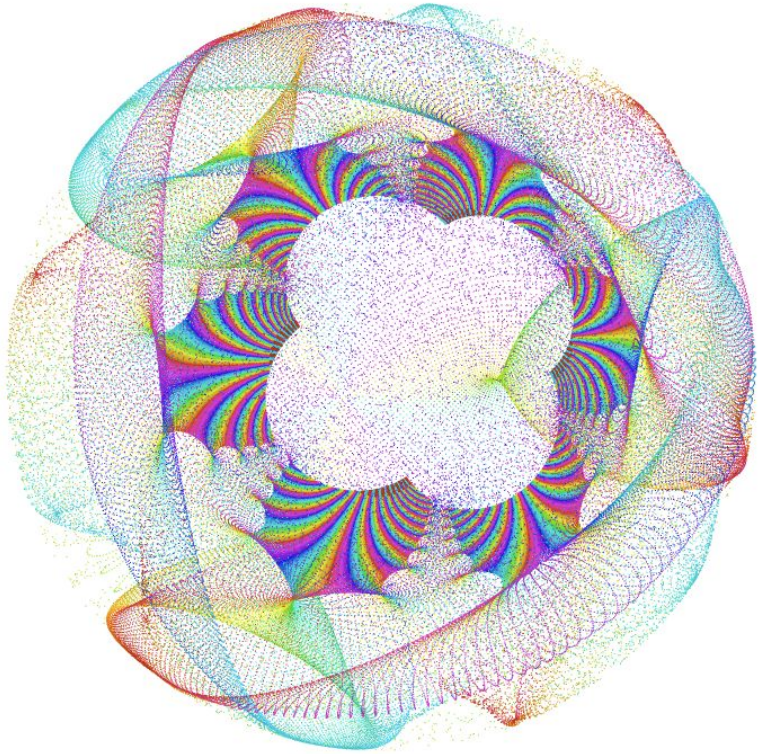


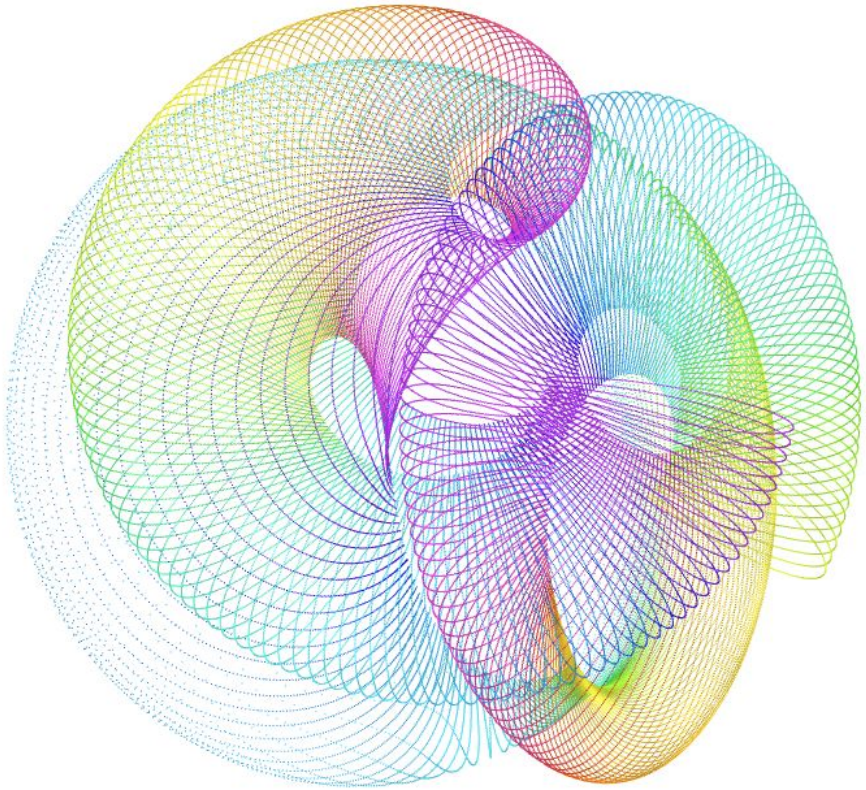
# Riemann Projection





# Custom Projection





# Why Implicit Functions

1

## Large Body of Mathematical Literature

Implicit functions are a standard topic of research, yet very little work has been done on visualizing them, even in trivial cases such as Riemann surfaces, which can easily be parametrized.

2

## Potential For Future Work

The technique we use can be generalized to any dimension, which means the 4D case can be used as a base for future work involving higher dimensional objects.



# Freudenthal-Kuhn Overview

**Steps**

**1**

### Triangulate the Mesh

Based on fixed permutations, decompose the tesseract into tetrahedrons or triangular prisms.

**2**

### Level Set

Similar to the Marching Cubes Algorithm, determine which points are inside our outside the level set, and interpolate the edges

**3**

### Iterate

This is done for all hypercubes inside the function

**Advantages**

### Seam-Free

The meshes are compact and have no irregularities, since the simplices never overlap

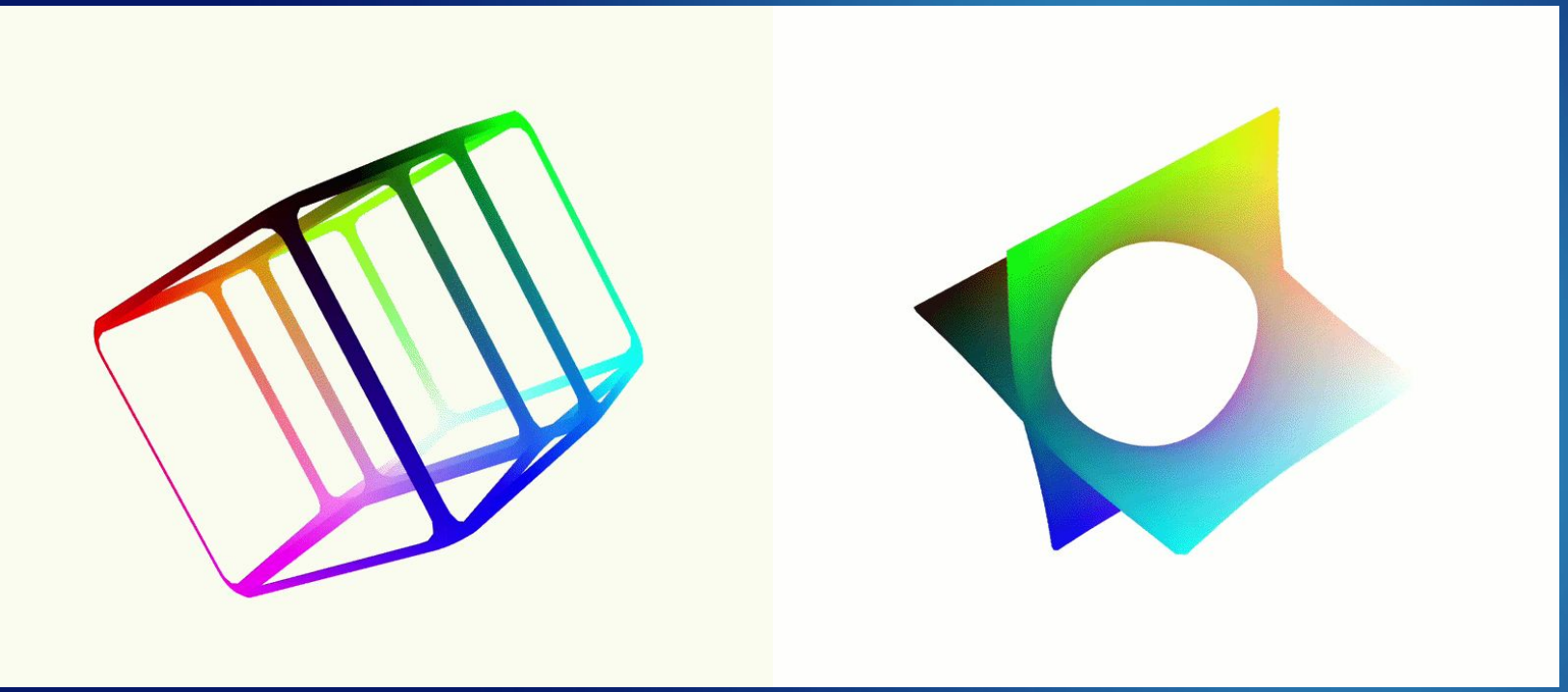
### Orientation Consistent

Primitives  are orientation consistent, since vertices are shared by faces, meaning no disagreement in terms of orientation

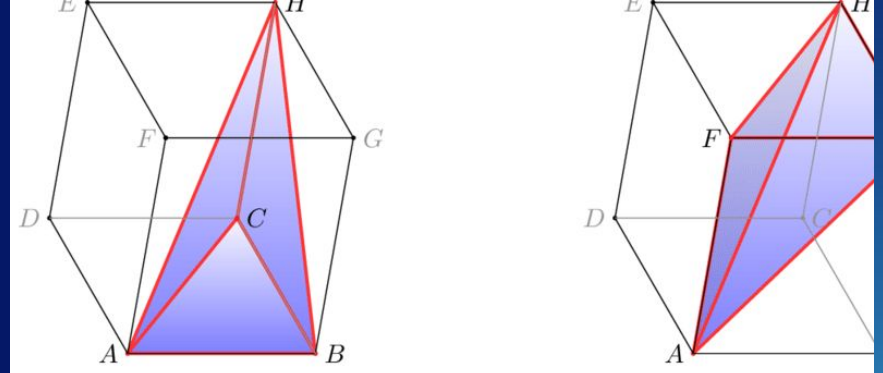
### Applicable to n-dimensions

The process is applicable to n-dimensions, making it useful for future work relating n-dimensional rendering

# Well-known Implicit Shapes



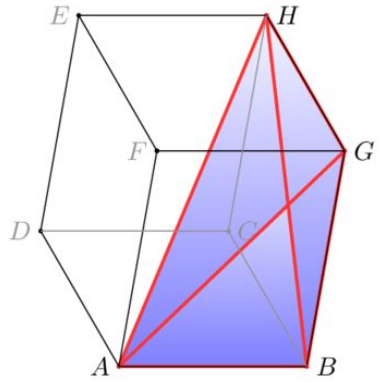
# Basic Diagram of Freudenthal-Kuhn



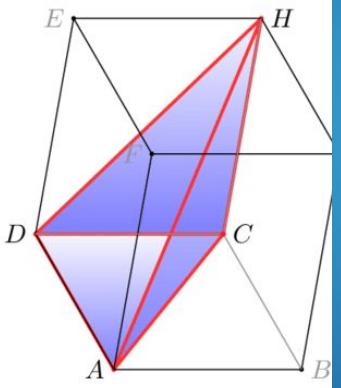

(a) The vertexes $A, B, C, H$ of $T_1$ as the corner points of the tetrahedron $C_{T_1}$.



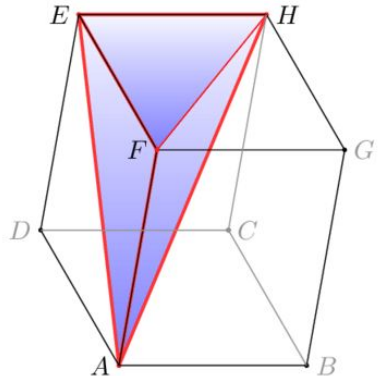

(b) The vertexes $A, F, G, H$ of $T$ corner points of the tetrahedron

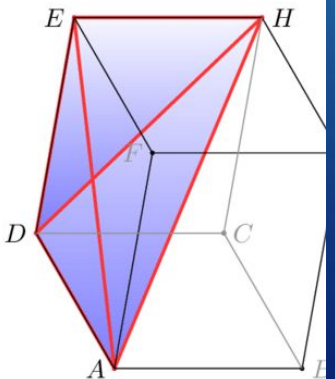


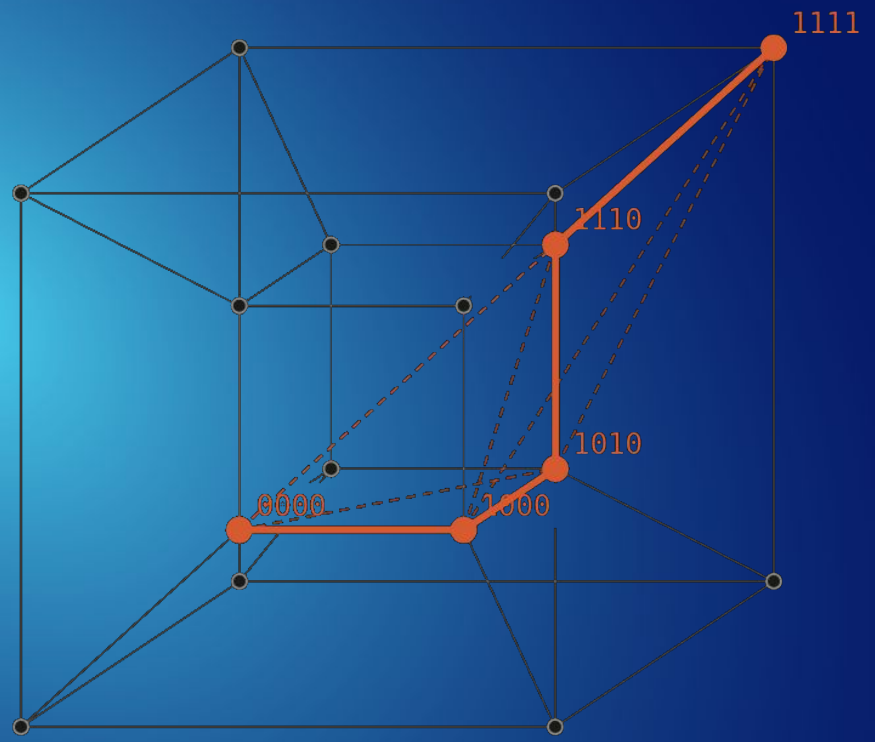

(c) The vertexes $A, B, G, H$ of $T_3$ as the corner points of the tetrahedron $C_{T_3}$.




(d) The vertexes $A, D, C, H$ of $T_4$ corner points of the tetrahedron $C_{T_4}$.

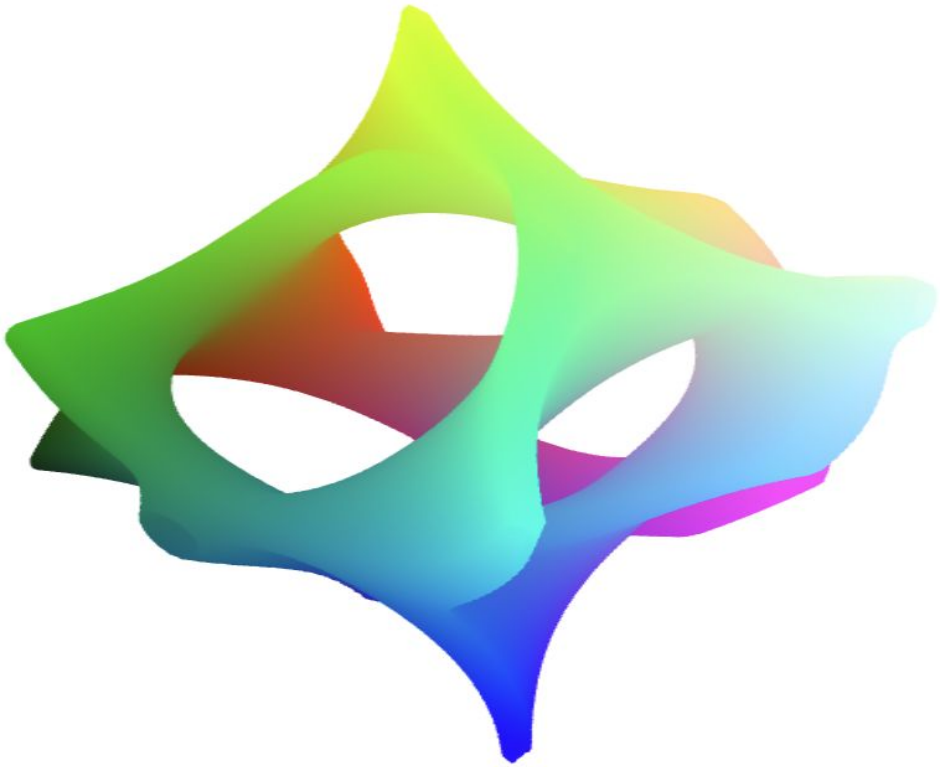


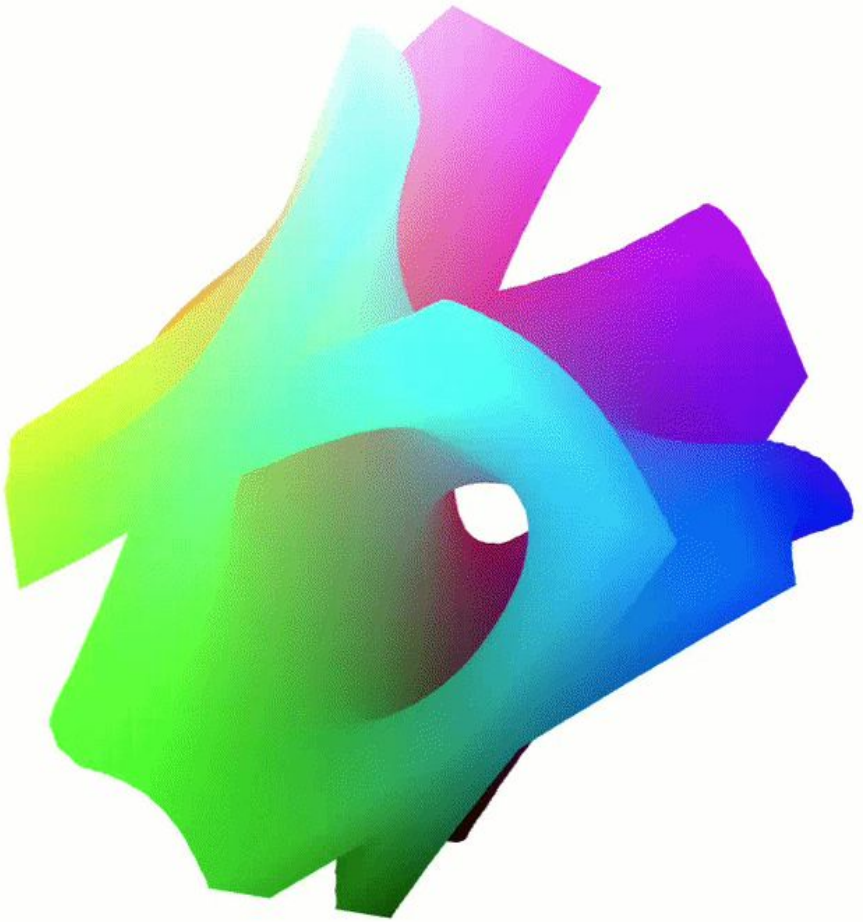



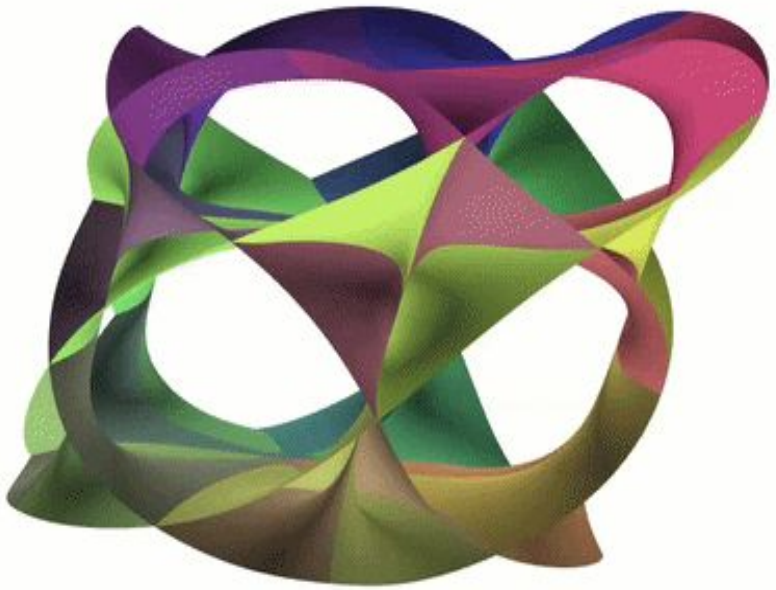





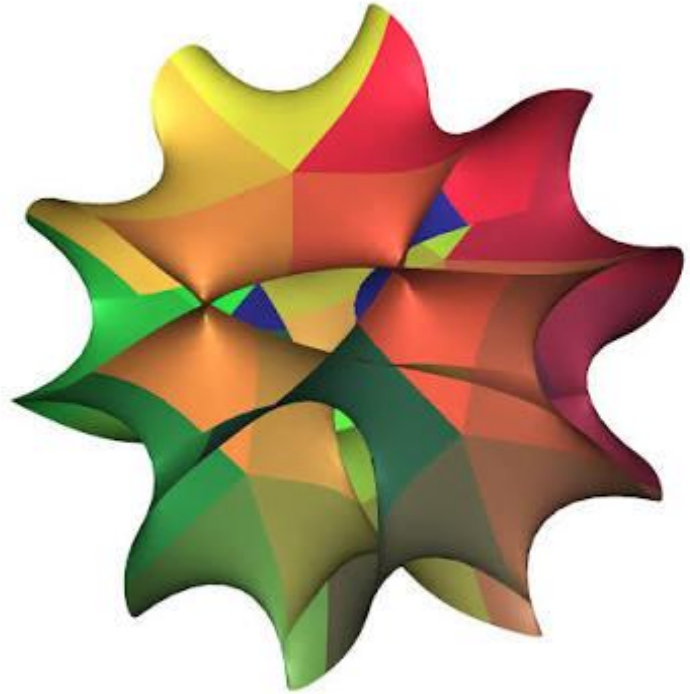

**Web3D** 2026

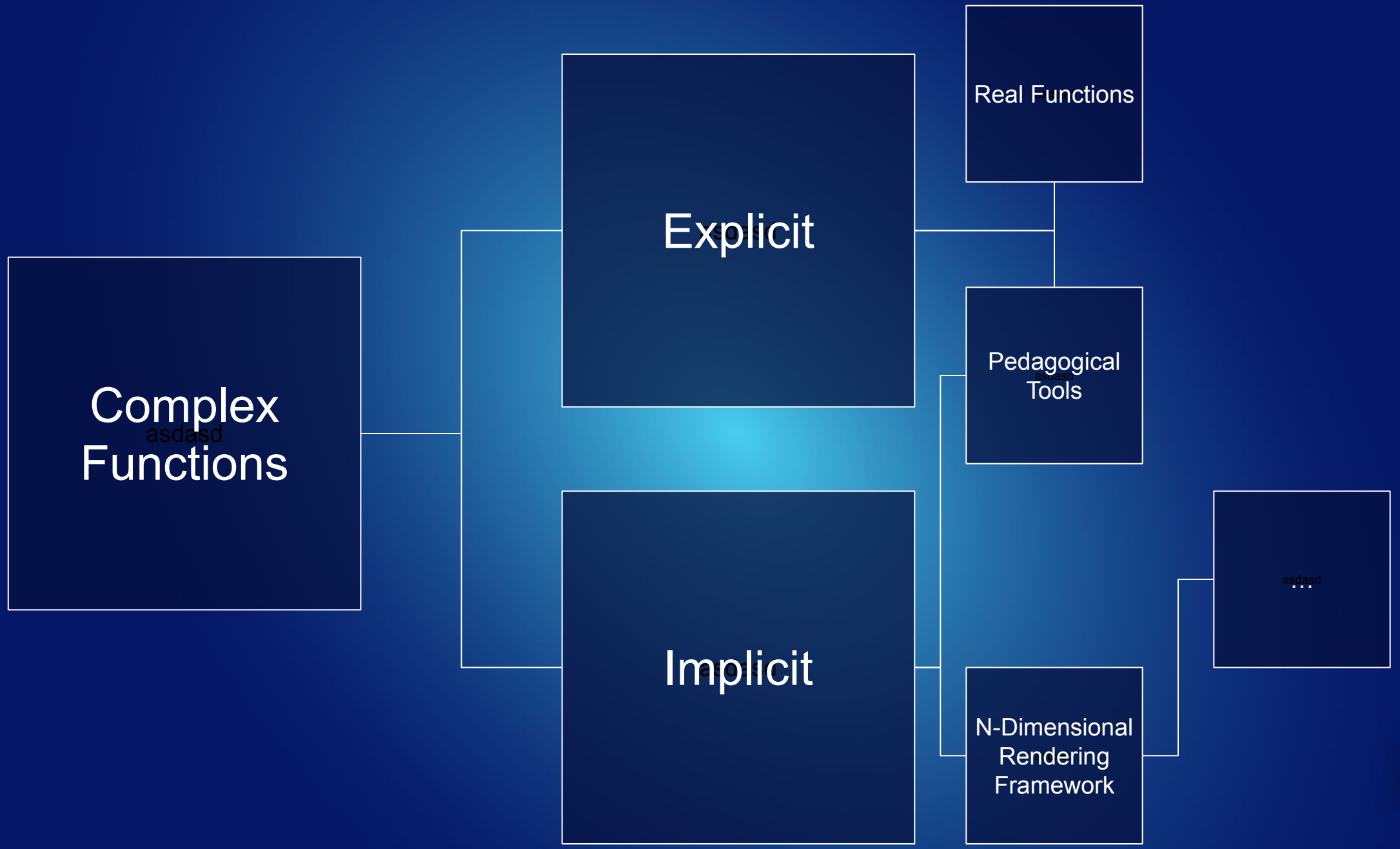


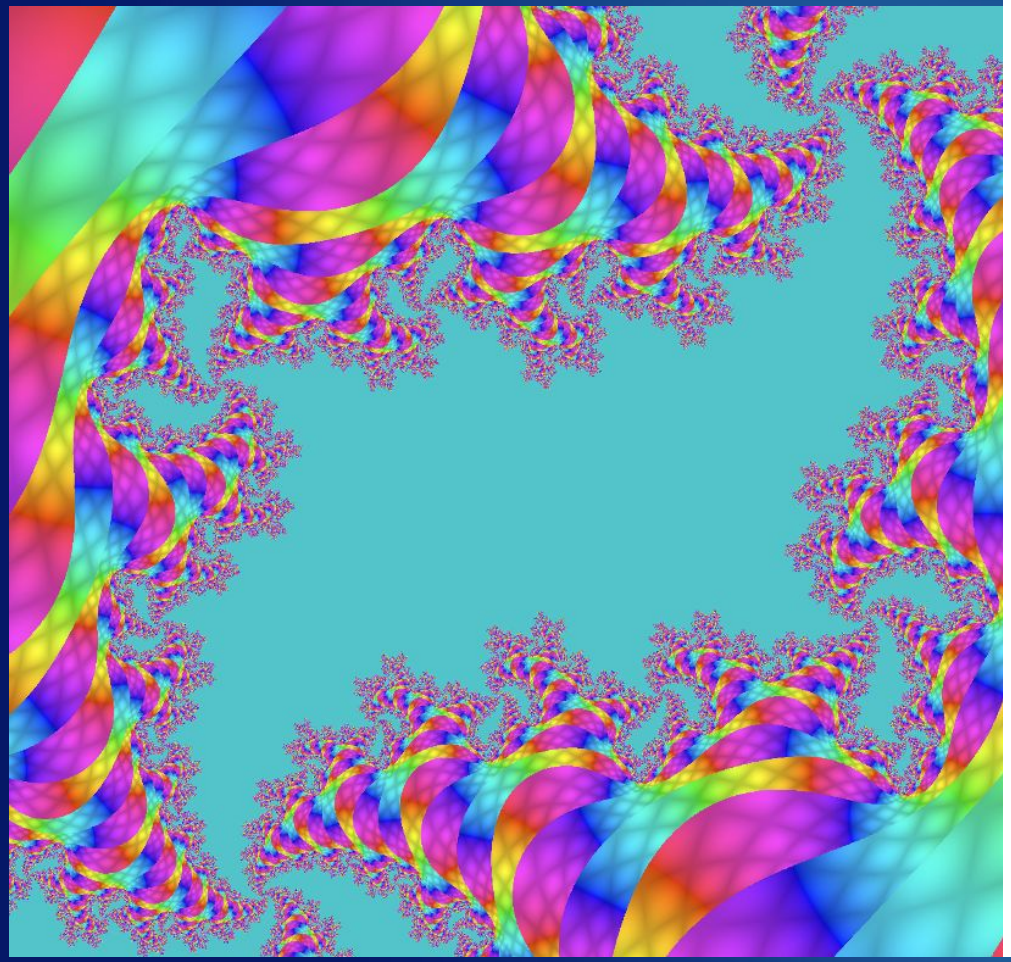


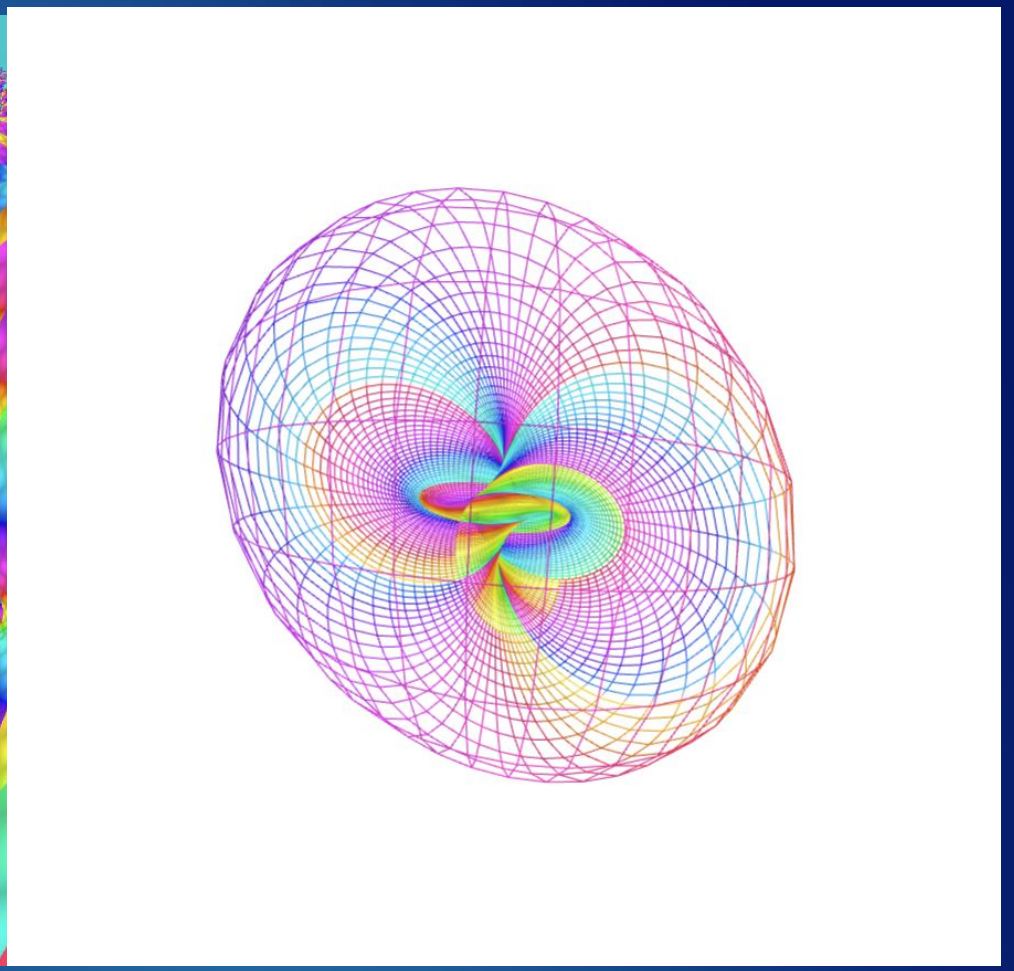


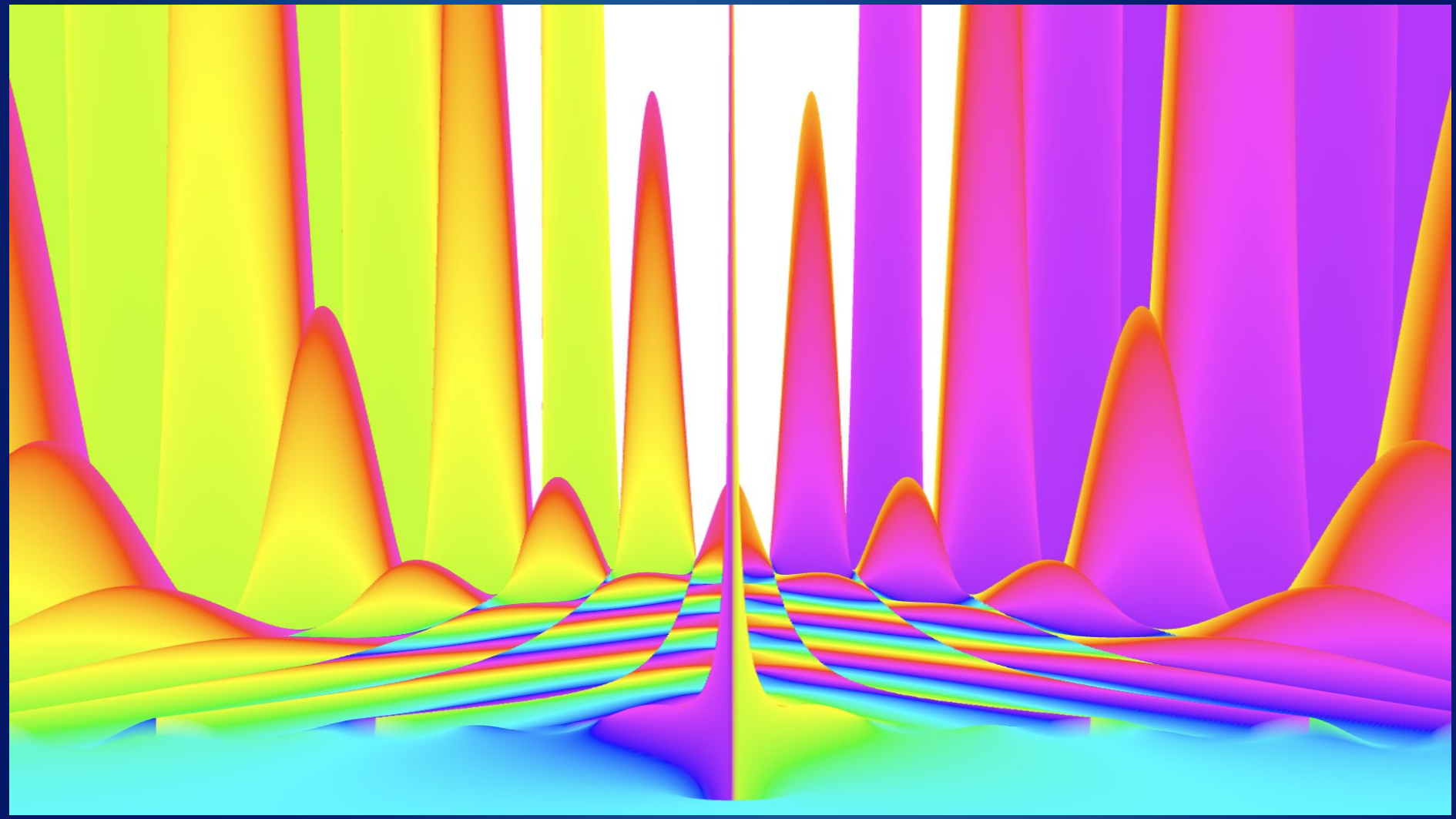

Switch Mode
4D Implicit
surfaceType
fermat quintic z^5+w
animateSpeed
0
rotationSpeed
0
4D Rotations
X-W Rotation
0
Y-W Rotation
0
Z-W Rotation
0
Parametric
March
resolution
8
isolevel
0
Smoothing passes
2
Close Controls




Complex Functions
asdasd
Explicit
asdasd
Implicit
asdasd
Real Functions
Pedagogical Tools
asdasd
N-Dimensional Rendering Framework
...
asdasd







# Future Plans

## Basic

Better UI/UX, including graph labeling

Custom Input

Side-by-side view

Picking (Interactive Information)

## Advanced

Rendering of n-dimensional objects and manifolds, using the implicit part as basis

Interactive topological simulations, including concepts such as fibrations, trivially implemented as shown in the previous slide

Generalized technique for n-dimensional rendering

Formal use of algebraic topology





# Thank You!

**Art Saiti**
artsaiti@proton.me

**Admir Huseini**
huseini@risat.org

**github.com/saitiart/complex-function-renderer**

## Web3D 2026

**Doha, Qatar**

