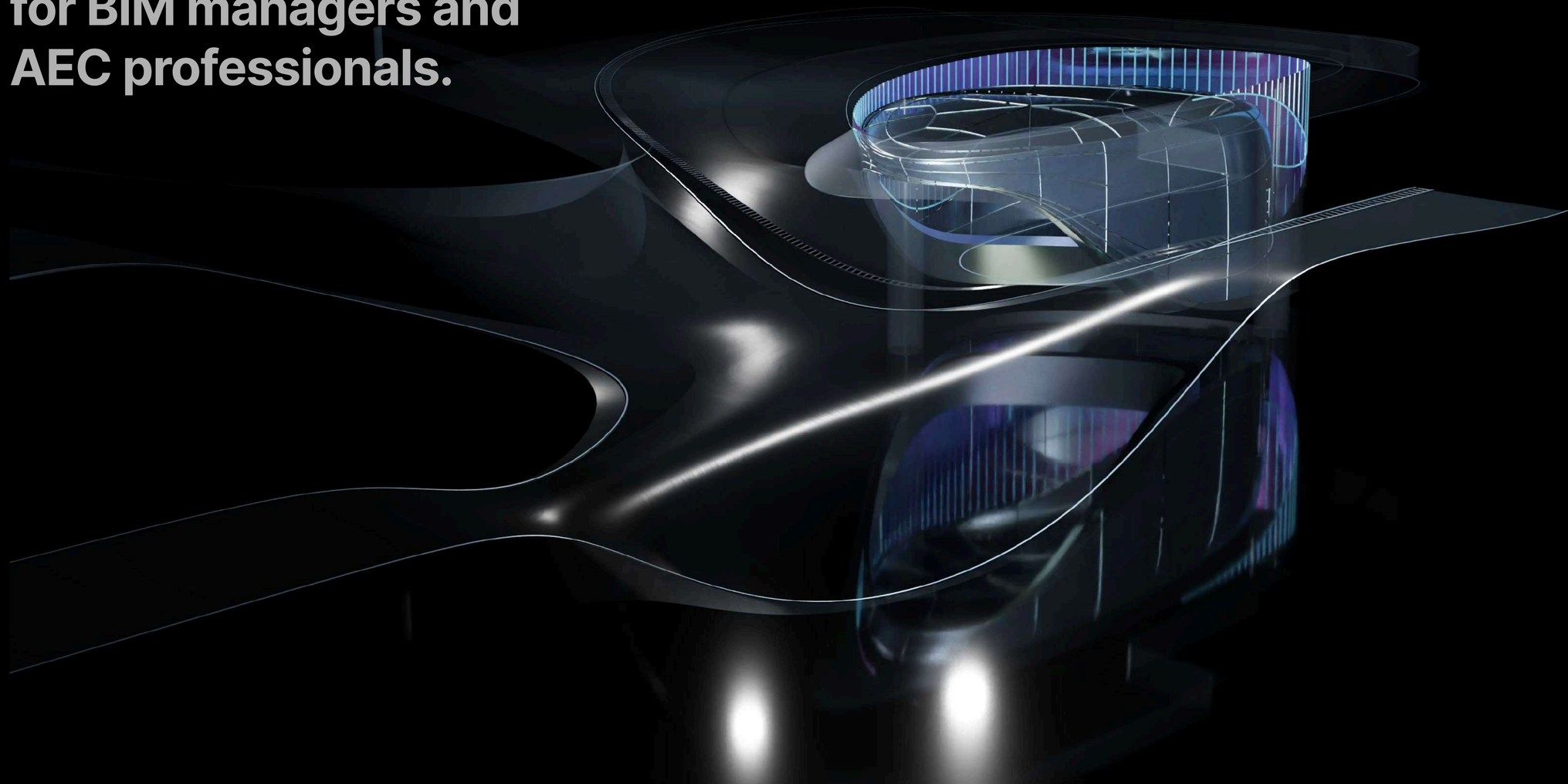
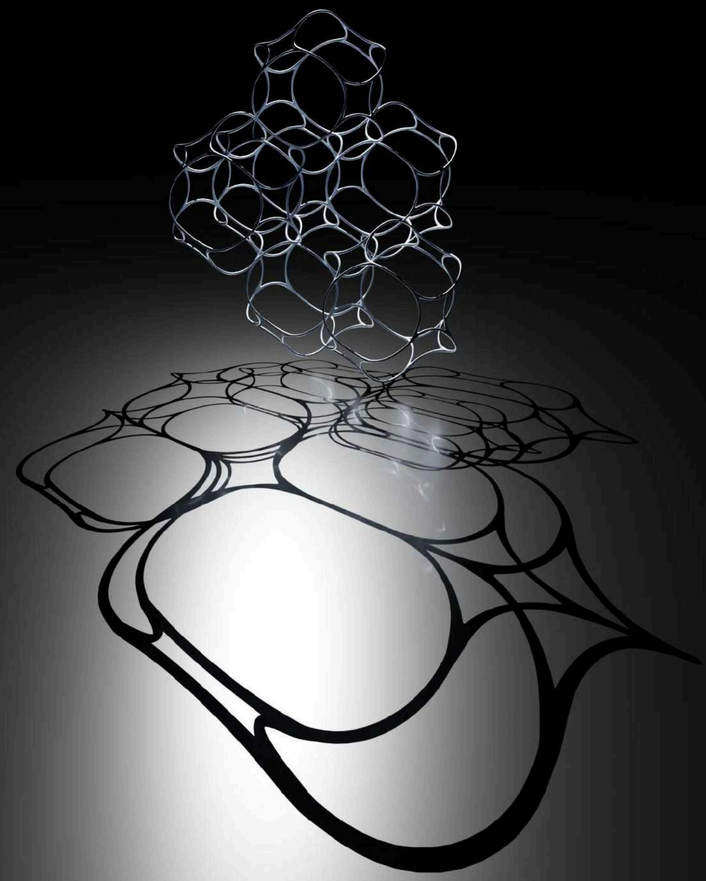
FUZES/ARCH

Detailed technical stack for BIM managers and AEC professionals.



.NET C# development

Object oriented software modules
for Rhino and Revit environments



FUZES.STUDIO.CODE Private

Watch 0 | Fork 0 | Star 0

master | 1 Branch | 0 Tags | Go to file t | Add file | Code

fuzesStudio Update README.md — c1433bc · now — 15 Commits

Components	MID Offset and collision bugfixes	2 days ago
Main	MID Offset and collision bugfixes	2 days ago
Projects/CTP	MID Offset and collision bugfixes	2 days ago
Properties	Add project files.	2 weeks ago
.gitattributes	Add .gitattributes, .gitignore, and LICENSE.txt.	2 weeks ago
.gitignore	Add .gitattributes, .gitignore, and LICENSE.txt.	2 weeks ago
FUZES.STUDIO.CODE.csproj	LARGE Foldering	last week
FUZES.STUDIO.CODE.sln	SMALL AsGraft addition, Helps namespace change	last week
FUZES.STUDIO.CODEInfo.cs	SMALL AsGraft addition, Helps namespace change	last week
LICENSE.txt	Add .gitattributes, .gitignore, and LICENSE.txt.	2 weeks ago
README.md	Update README.md	now

README | MIT license

About

Core module library of FuzesStudio.

- Readme
- MIT license
- Activity
- 0 stars
- 0 watching
- 0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

C# 100.0%

Suggested workflows

Rhino plugins.

Grasshopper components designed for absolute clarity, robustness, and repeatable use.

Easily activated nodes based on the Rhino Kernel that extend in-house scripts with validated logic and predictable behavior.

Revit commands.

Custom BIM logic and tools for reliable model operations, validation, and controlled data handling.

Family-based workflows, transactions, automated checks, error messaging and queries - based on the Revit API.

```
2 references
public class FitnessSettingsValidator : FlatMixCoreBaseValidator<FitnessSettings>

private static readonly FitnessSettingsValidator instance =
    new FitnessSettingsValidator();
1 reference
public static FitnessSettingsValidator Instance { get; } = instance;

1 reference
private FitnessSettingsValidator()
{
    RuleFor(x => x.BestPopulationCount)
        .Must(value => 0 <= value)
        .WithMessage(nameof(FitnessSettings.BestPopulationCount) +
        " must have a non-negative value.");

    RuleFor(x => new List<double>() {
        x.AreaWeight,
        x.OrderWeight,
        x.CountWeight,
    })
        .Must(value => value.All(x => 0 <= x && x <= 10))
        .WithMessage(
            nameof(FitnessSettings.AreaWeight) + ", " +
            nameof(FitnessSettings.OrderWeight) +
            nameof(FitnessSettings.CountWeight) +
```

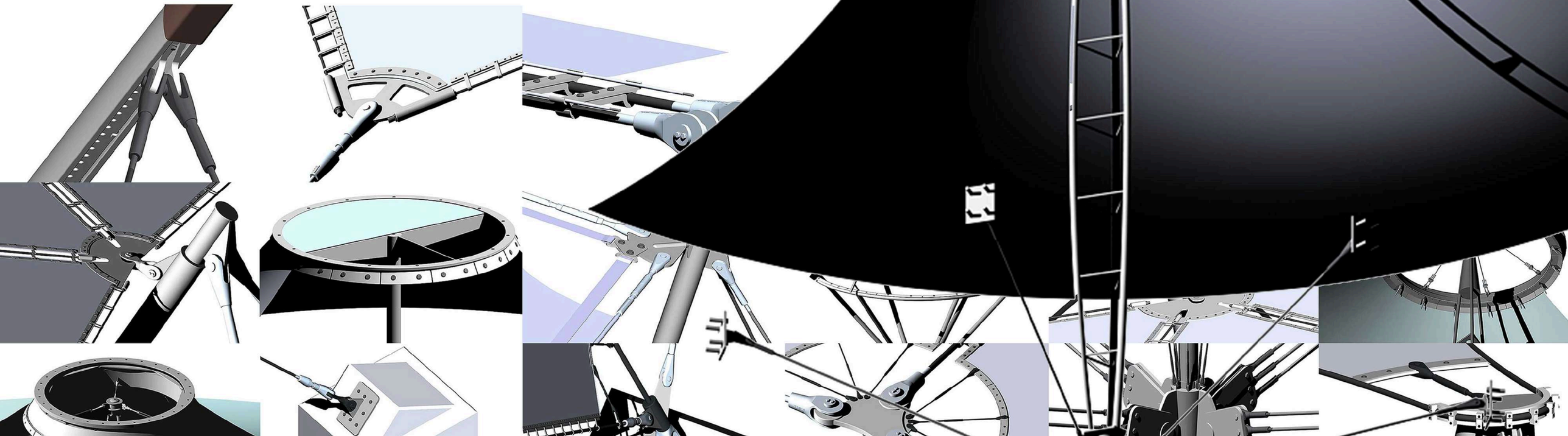
```
2 references
public FitnessSettings(
    double areaWeight,
    double orderWeight,
    double countWeight,
    int bestPopulationCount)
{
    AreaWeight = areaWeight;
    OrderWeight = orderWeight;
    CountWeight = countWeight;
    BestPopulationCount = bestPopulationCount;

    Validate();
}

2 references
public void Validate() => FitnessSettingsValidator.Instance.Validate(this);

2 references
public static FitnessSettings Default()
{
    return new FitnessSettings(
        DefWeight,
        DefWeight,
        DefWeight,
        DefBestPopulationCount);
```

Precision detailing

Scripted modeling workflows

for millimeter-accurate items.

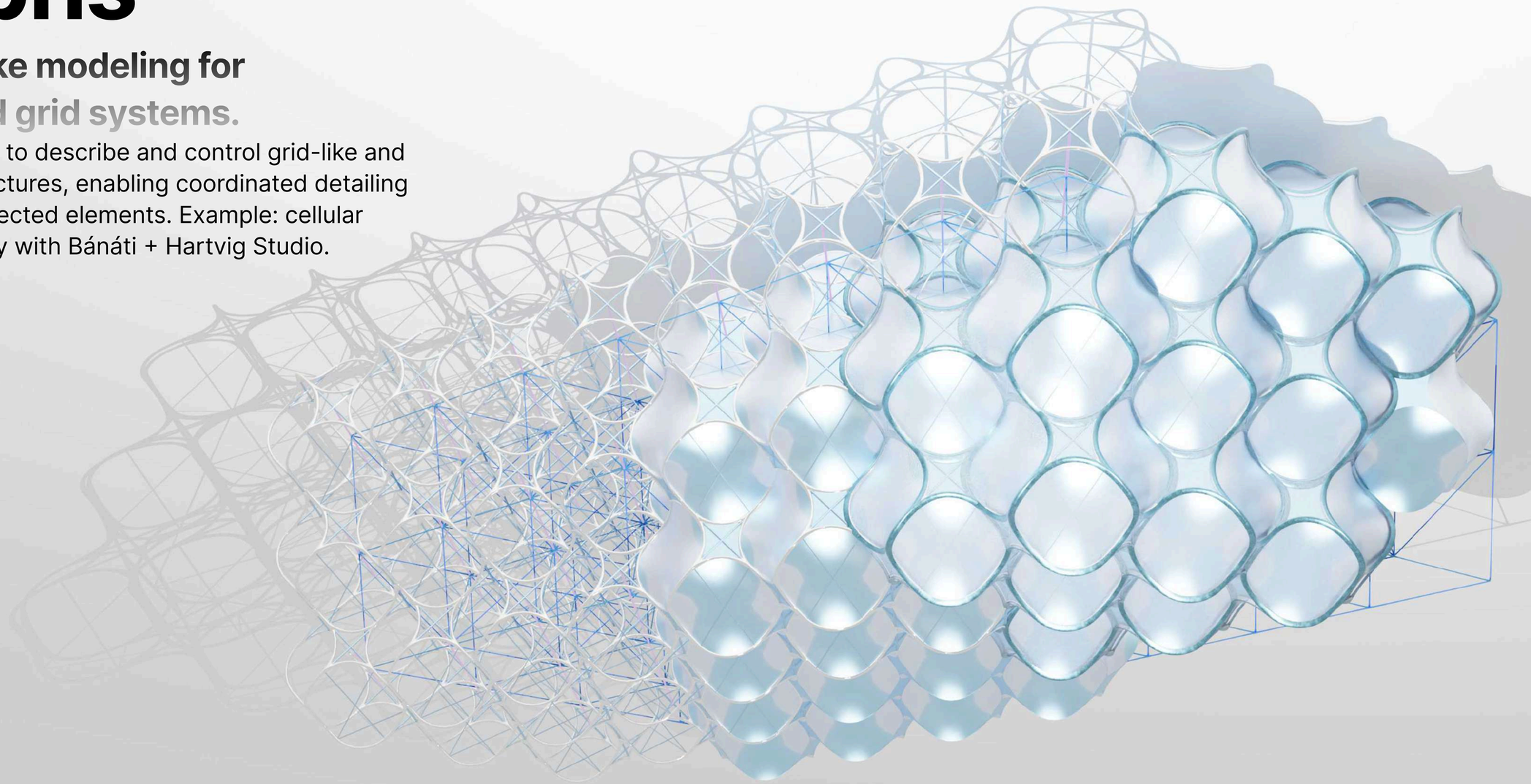
High-resolution geometric definitions combining manual detailing and automation, suitable for fabrication-level coordination and verification.



Graphs

Network-like modeling for interrelated grid systems.

Graph logic used to describe and control grid-like and background structures, enabling coordinated detailing across interconnected elements. Example: cellular competition entry with Bánáti + Hartvig Studio.

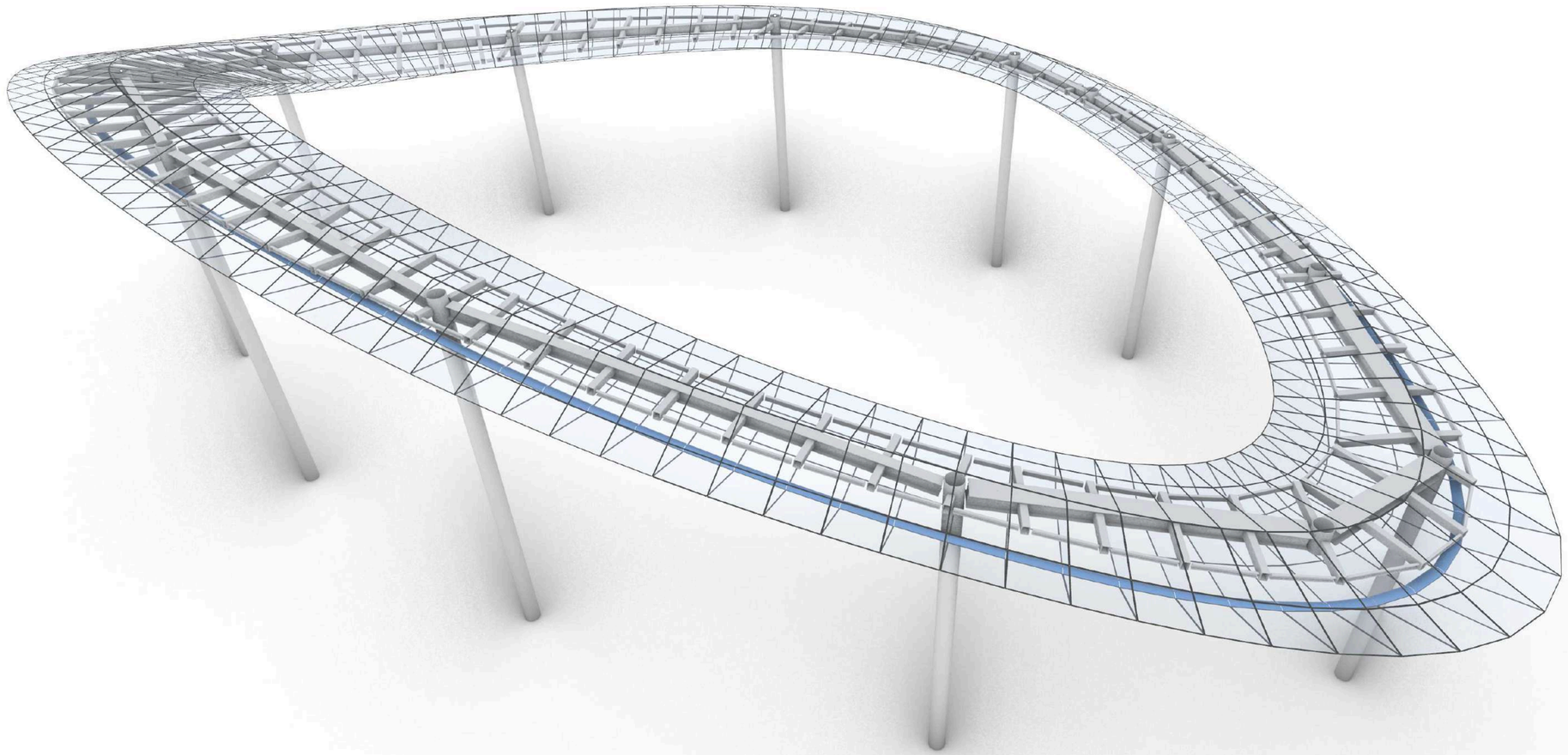


Optimization

Extending existing parametric workflows

with targeted optimum searches.

Genetic search and evaluation loops applied to real project constraints to minimize, maximize, or balance defined *performance values*. Example: panel unification logic for floating canopy.

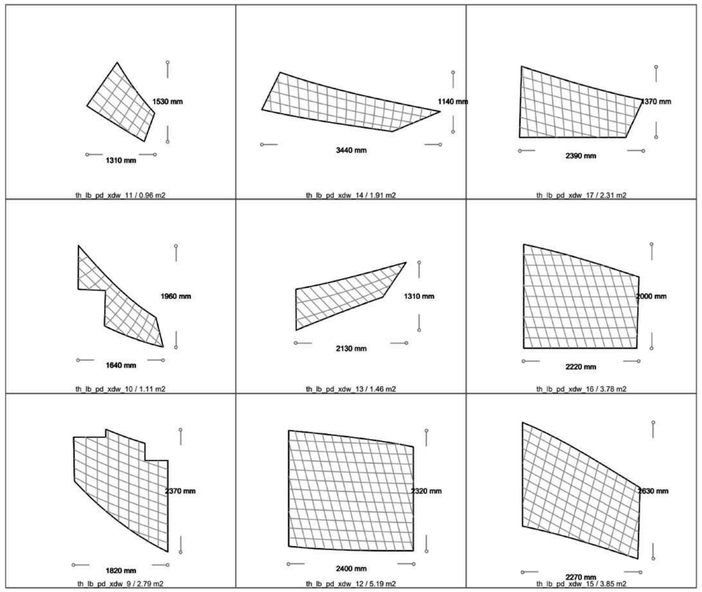


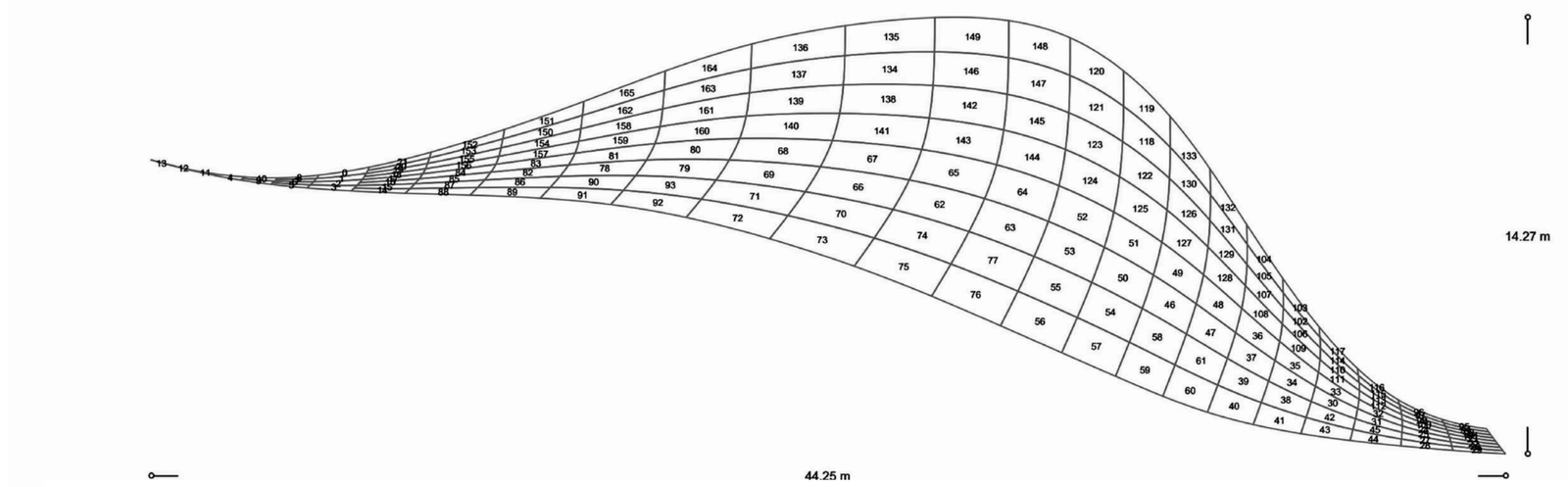
AutoDocs

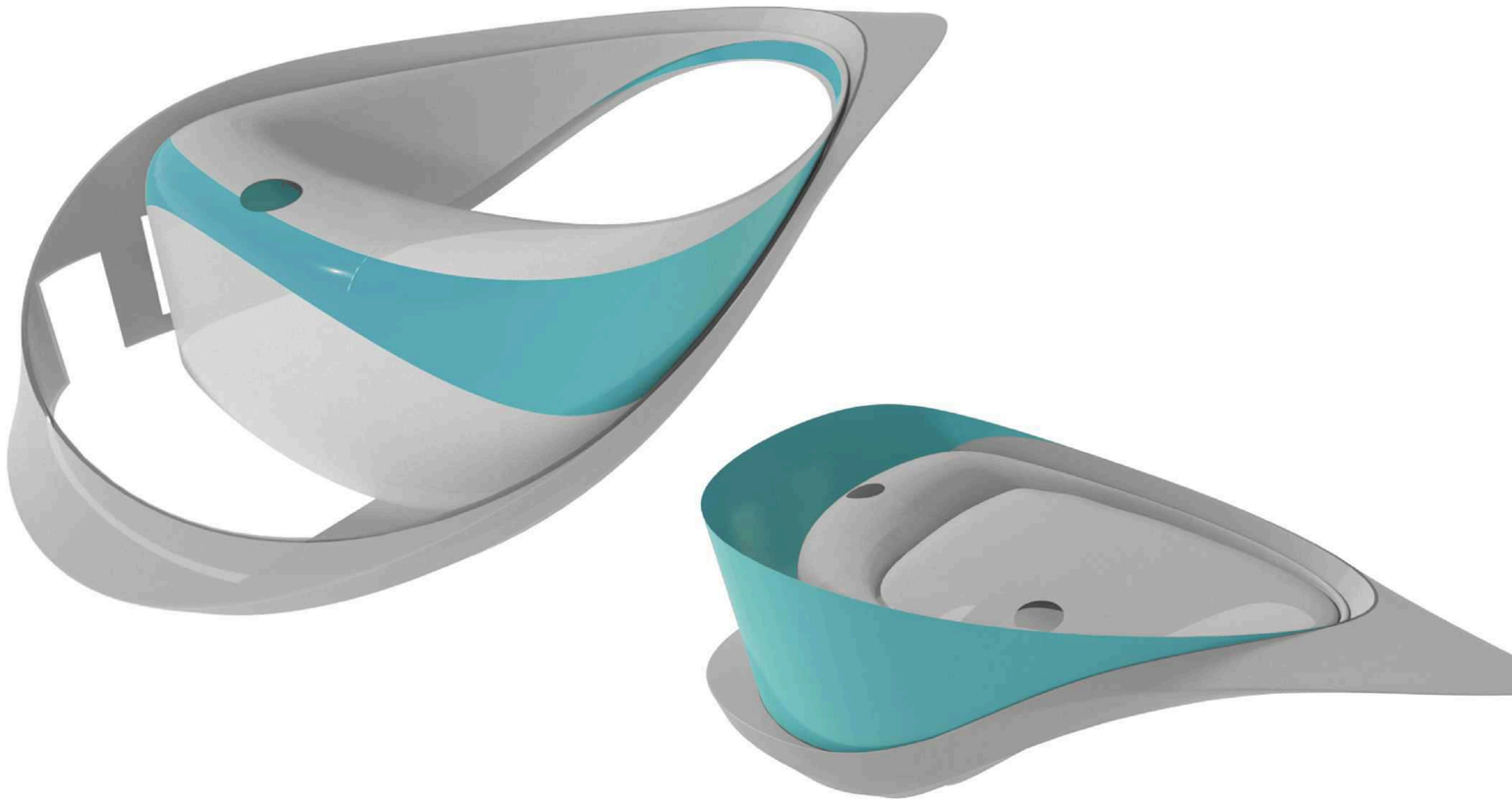
Dynamic viewport export

and sheet organization.

BIM-based tools helping teams produce drawings faster at scale, with structured item and group indexing, dynamic schedule and view creation and real-time file outputs.







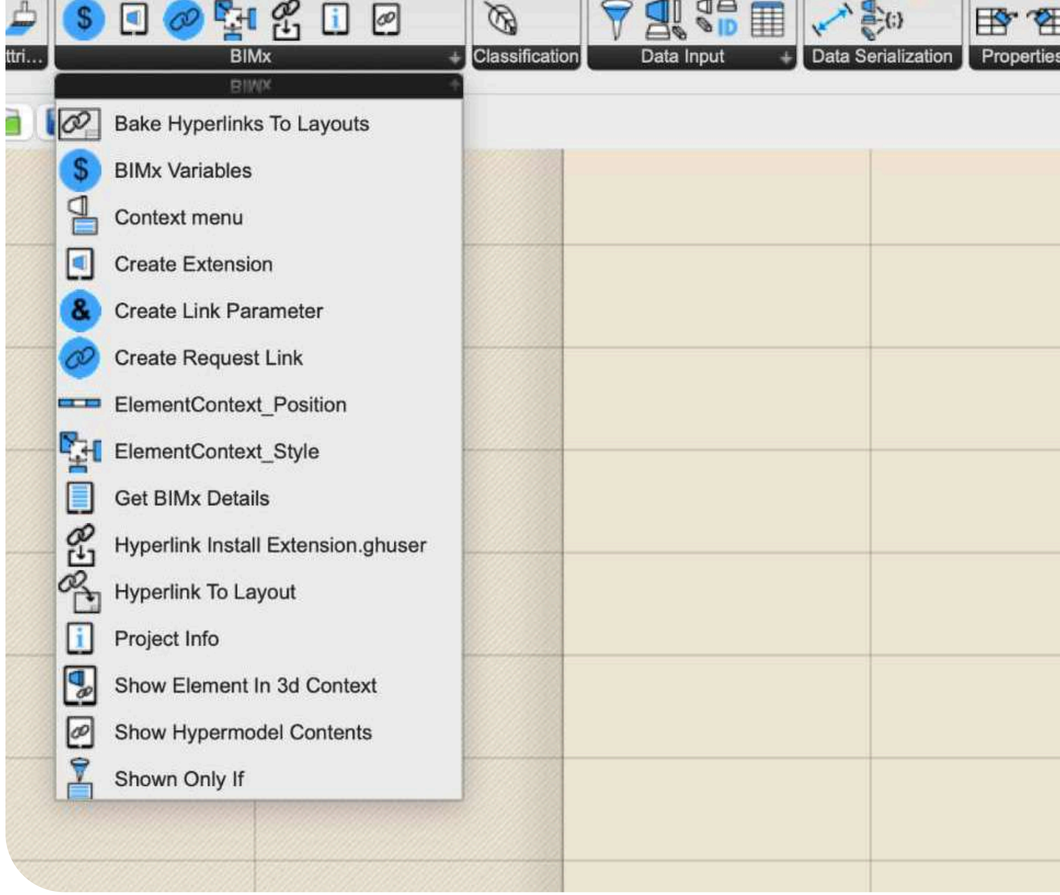
Rhino ↔ Archicad interoperability

Structured data exchange

between environments.

Grasshopper-side development and cleanup of the *tAPIr* plugin, introducing clearer logic, improved commands, and more predictable data flow.

```
2 references
protected override void AddOutputs()
{
    OutTexts("Ids");
    OutTexts("Names");
    OutTexts("Values");
}

2 references
protected override void Solve(
    IGH_DataAccess da)
{
    if (!TryGetConvertedCadValues(
            CommandName,
            null,
            ToAddOn,
            JHelp.Deserialize<ProjectInfoFields>,
            out ProjectInfoFields response))
    {
        return;
    }

    da.SetDataList(
        0,
        response.Fields.Select(x => x.ProjectInfoId));

    da.SetDataList(
        1,
        response.Fields.Select(x => x.ProjectInfoName));

    da.SetDataList(
        2,
        response.Fields.Select(x => x.ProjectInfoValue));
```



tapir