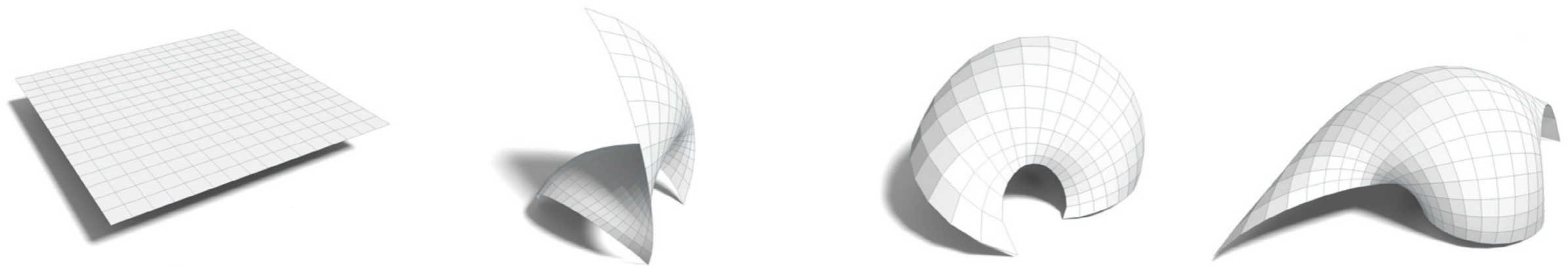


# Constrained Geometry Processing Using *ShapeOp*

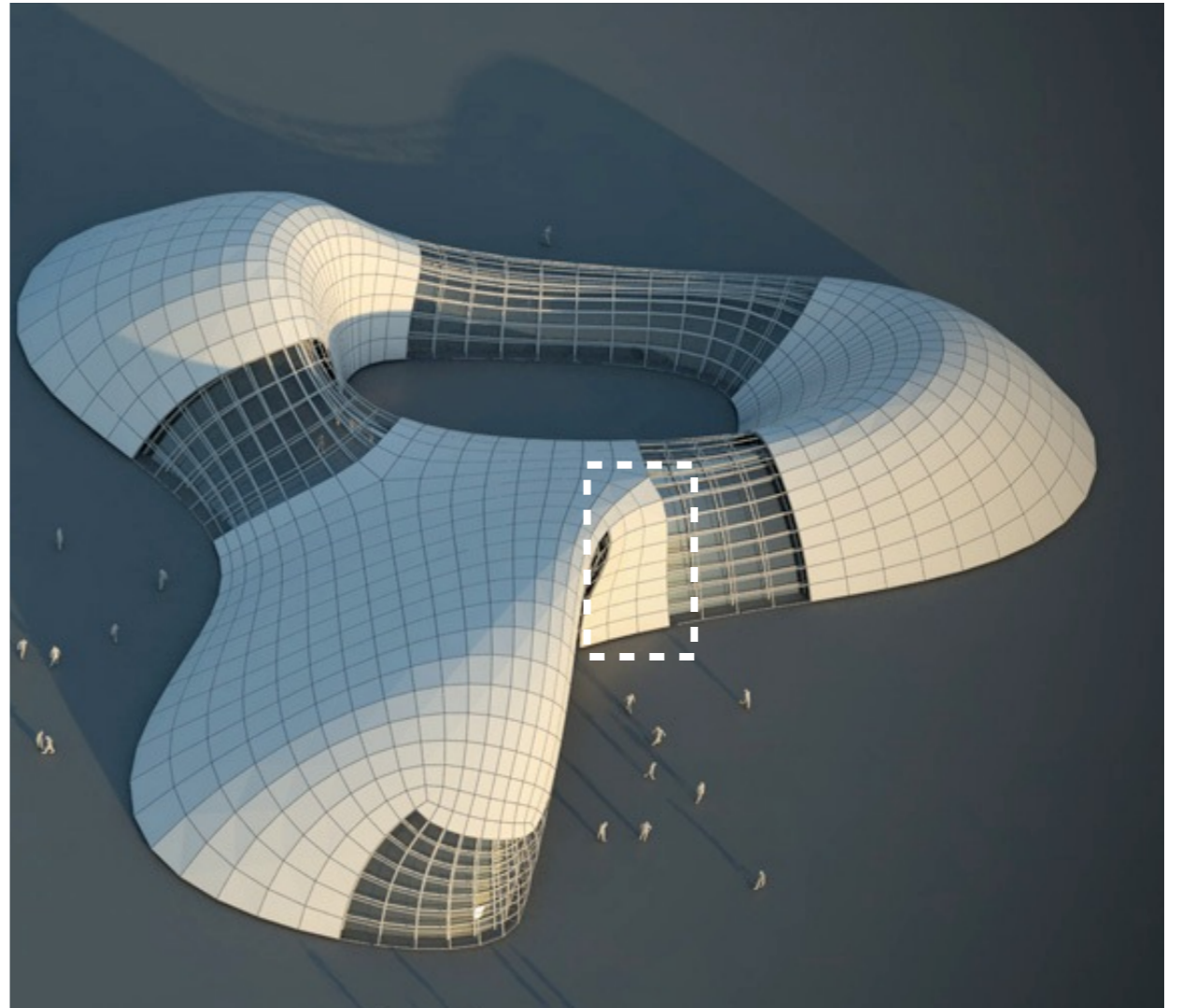


**Bailin Deng**

Department of Computer Science  
University of Hull

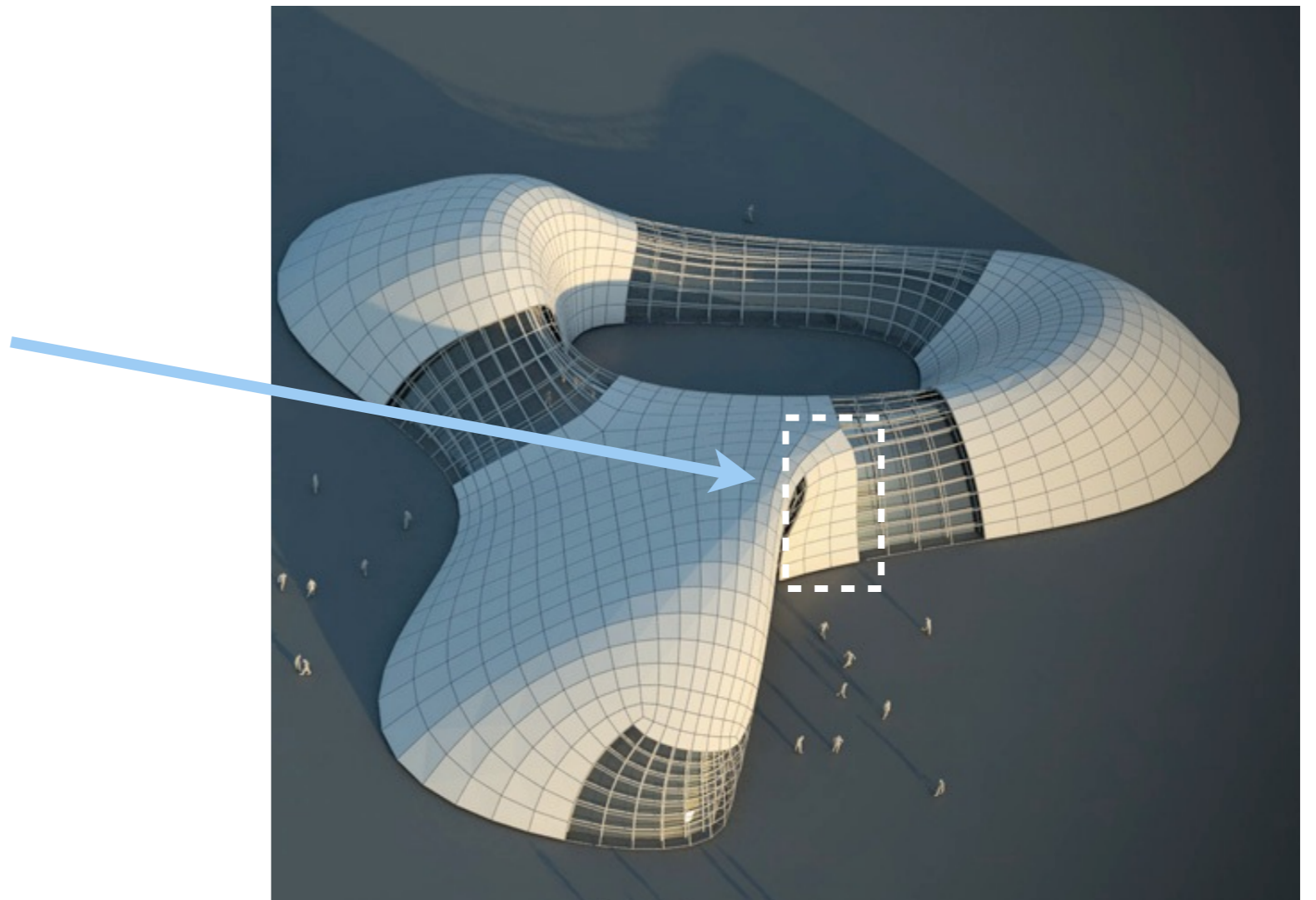
[b.deng@hull.ac.uk](mailto:b.deng@hull.ac.uk)

# Constraint-based Design



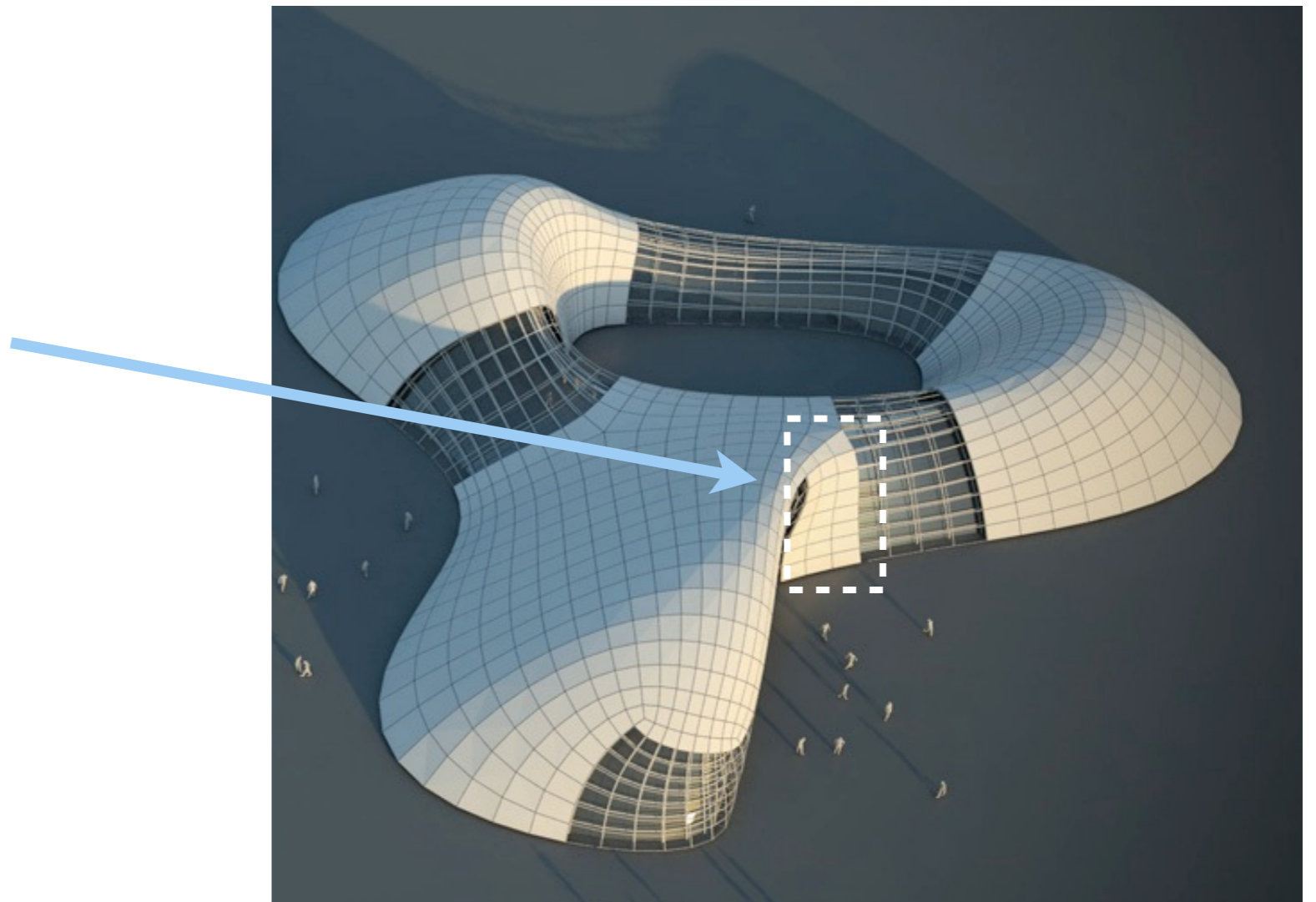
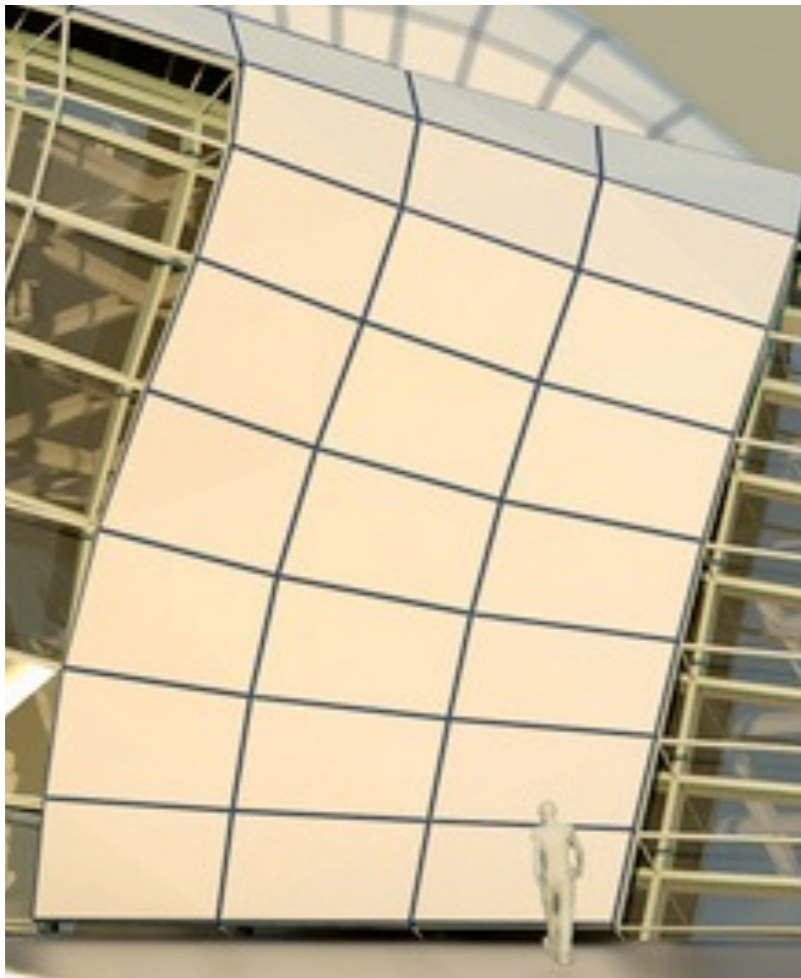
# Constraint-based Design

- Fabrication requirement: planar panels



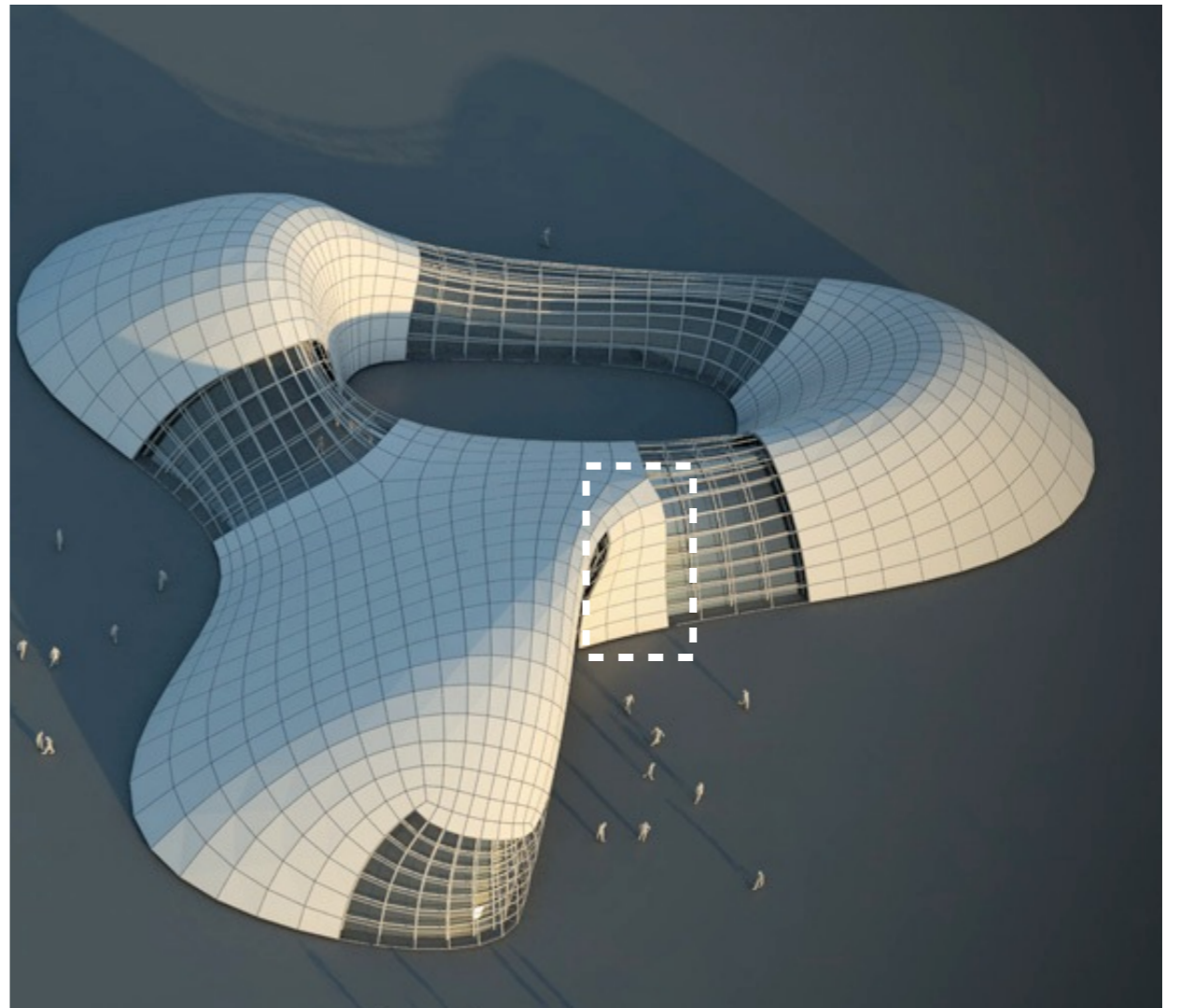
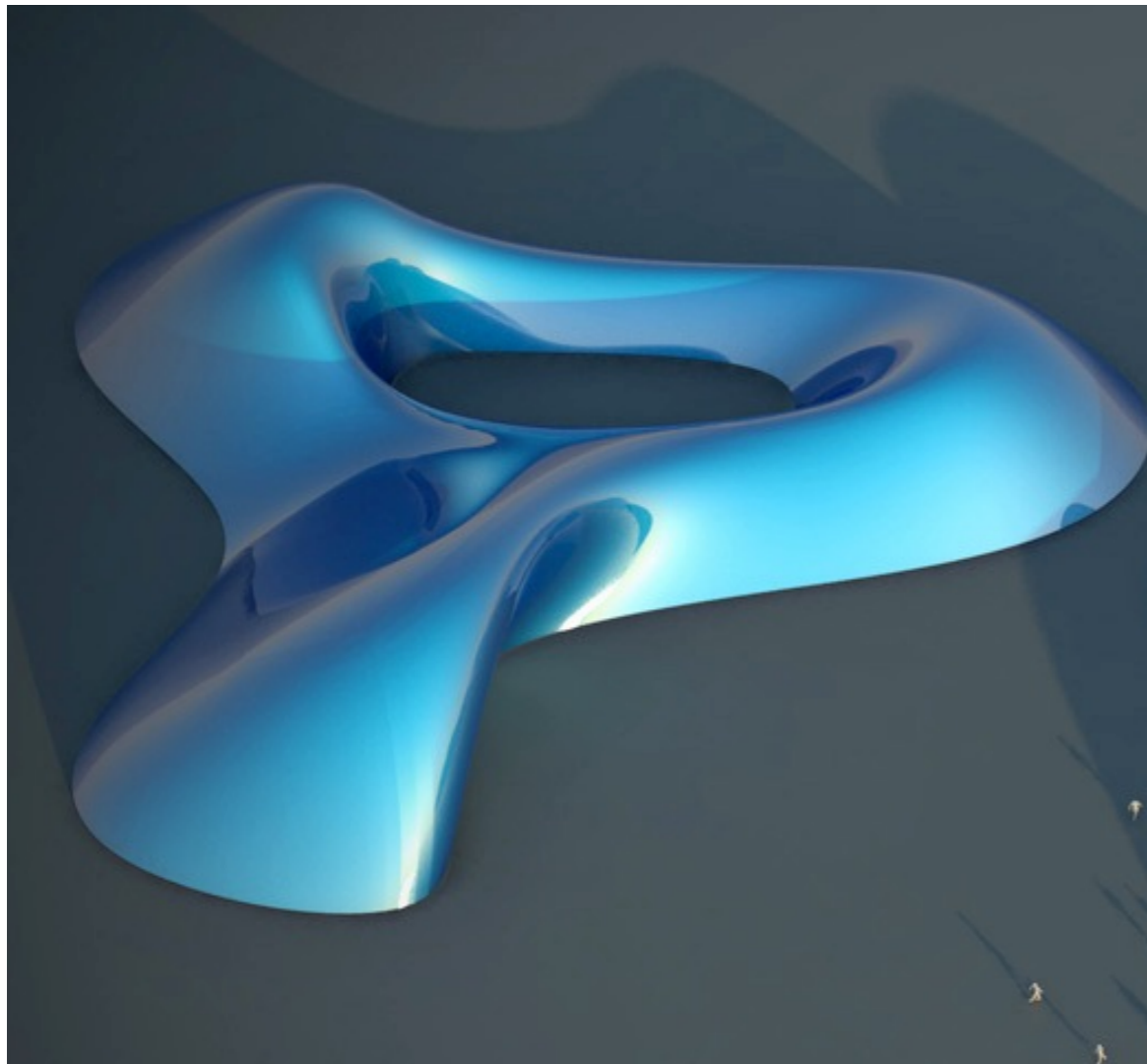
# Constraint-based Design

- Fabrication requirement: planar panels
- Aesthetics requirement: smooth patterns



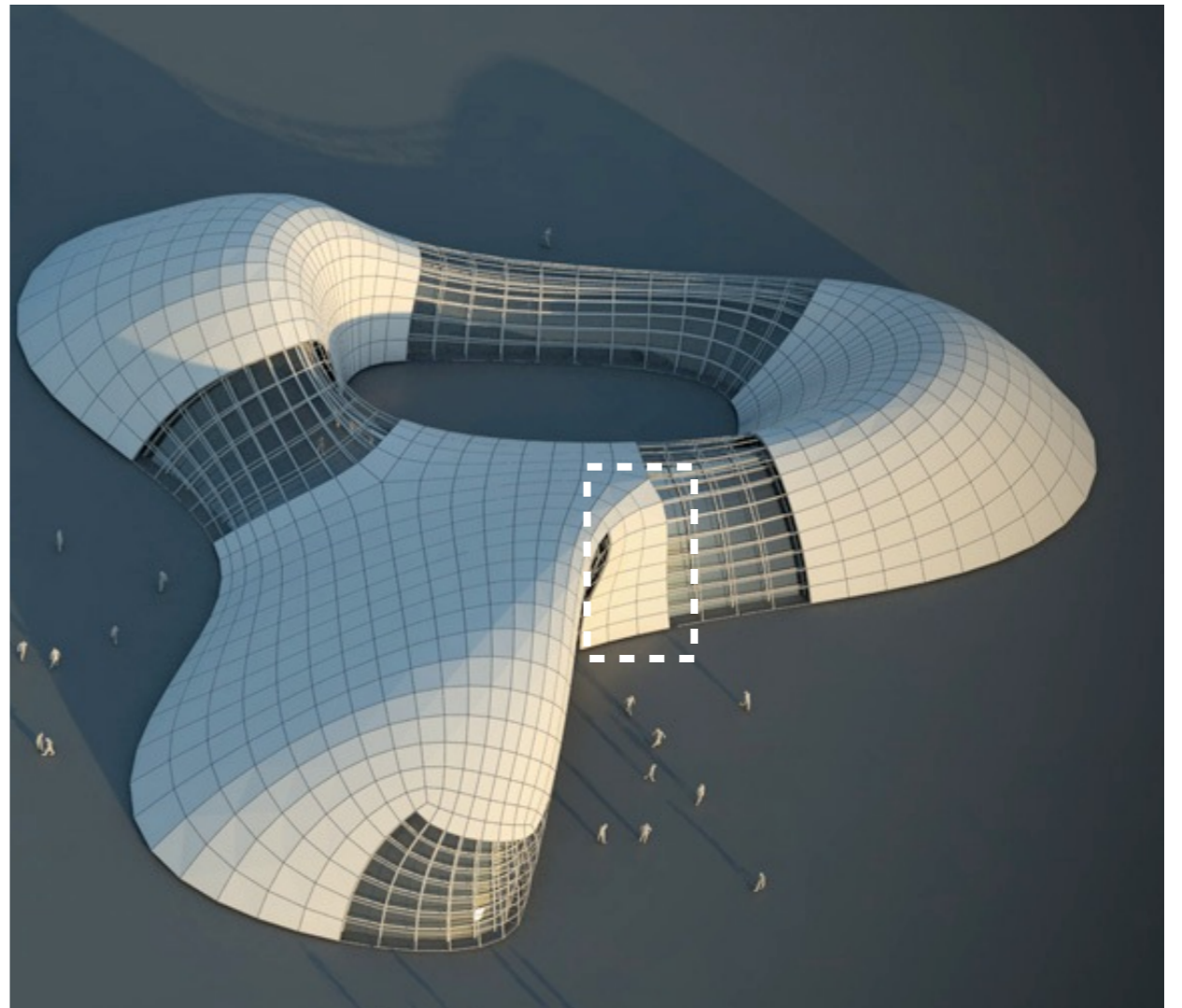
# Constraint-based Design

- Coherence with design intent



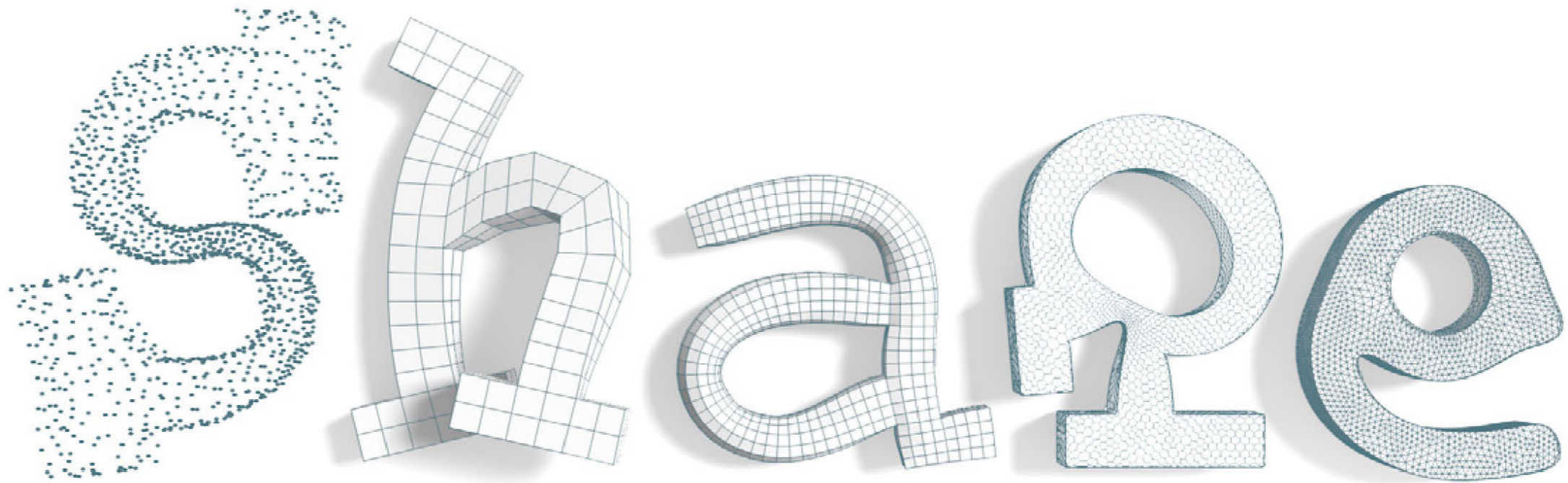
# Constraint-based Design

Geometric shapes  
subject to constraints



# ShapeOp

- C++ library for constrained geometry processing



<http://shapeop.org>

# Contributors

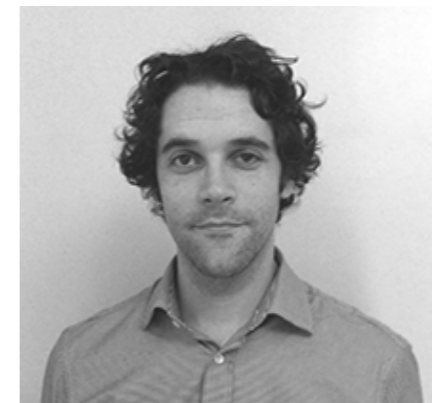
- Computer Graphics and Geometry Lab, EPFL



- CITA, Royal Danish Academy of Fine Arts



- Robert McNeel & Associates

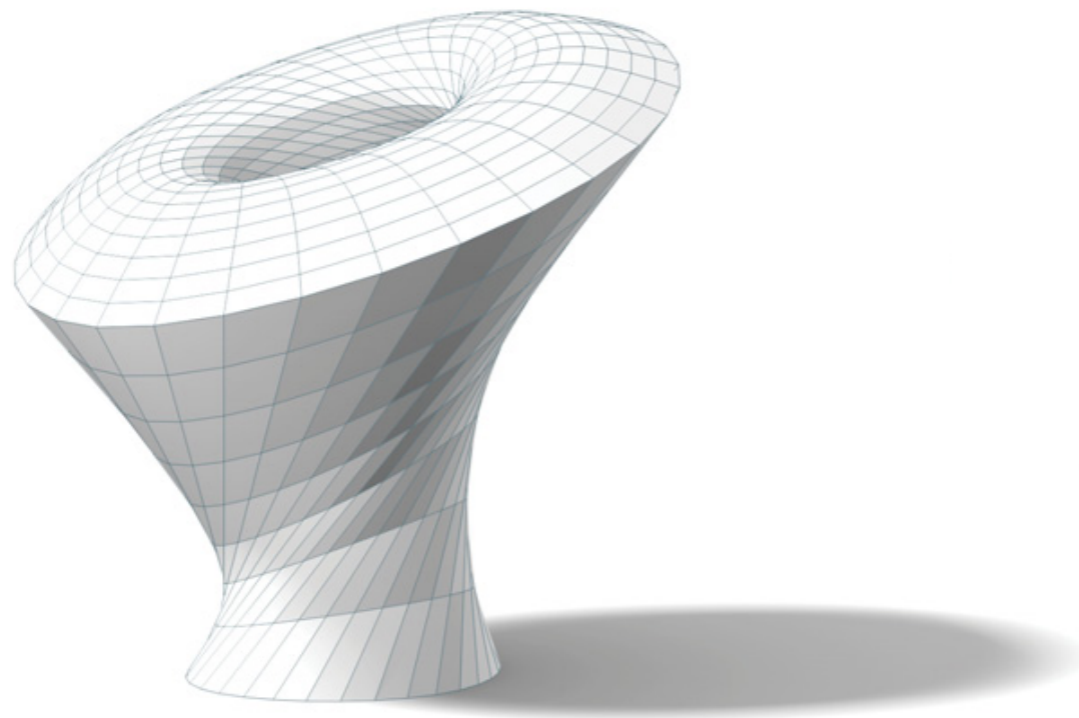


# Installation

1. Download the data files and unzip
2. Copy all library files to C:\Users\%USERNAME%\AppData\Roaming\Grasshopper\Libraries, where %USERNAME% is the windows user name. For 64-bit Rhino, choose the files in the 64bit folder. Otherwise, use the 32bit folder.
3. Unblock library files
4. Install Visual C++ Redistributable Packages for Visual Studio 2013 (<https://www.microsoft.com/en-gb/download/details.aspx?id=40784>). For 64-bit Rhino, choose vc\_redist\_x64.exe. Otherwise, choose vc\_redist\_x86.exe.
5. Add the ShapeOp.dll folder to PATH system variable
6. Open Rhino and Grasshopper, load the file ShapeOp\_Sheffield.gh.  
Specify the library files in your AppData\Roaming\Grasshopper\Libraries folder if you see error messages about not finding those files.

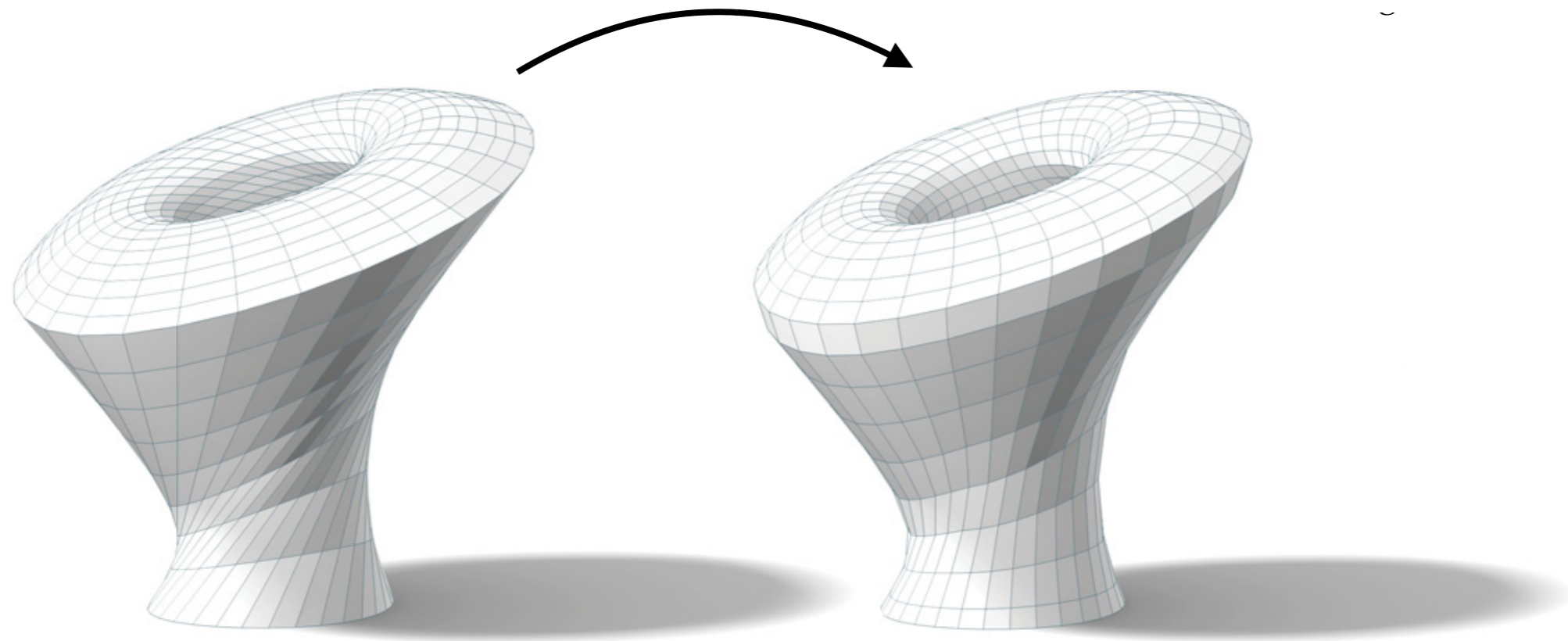
# LibShapeOp Solver

- Input
  - initial mesh
  - constraints for vertices



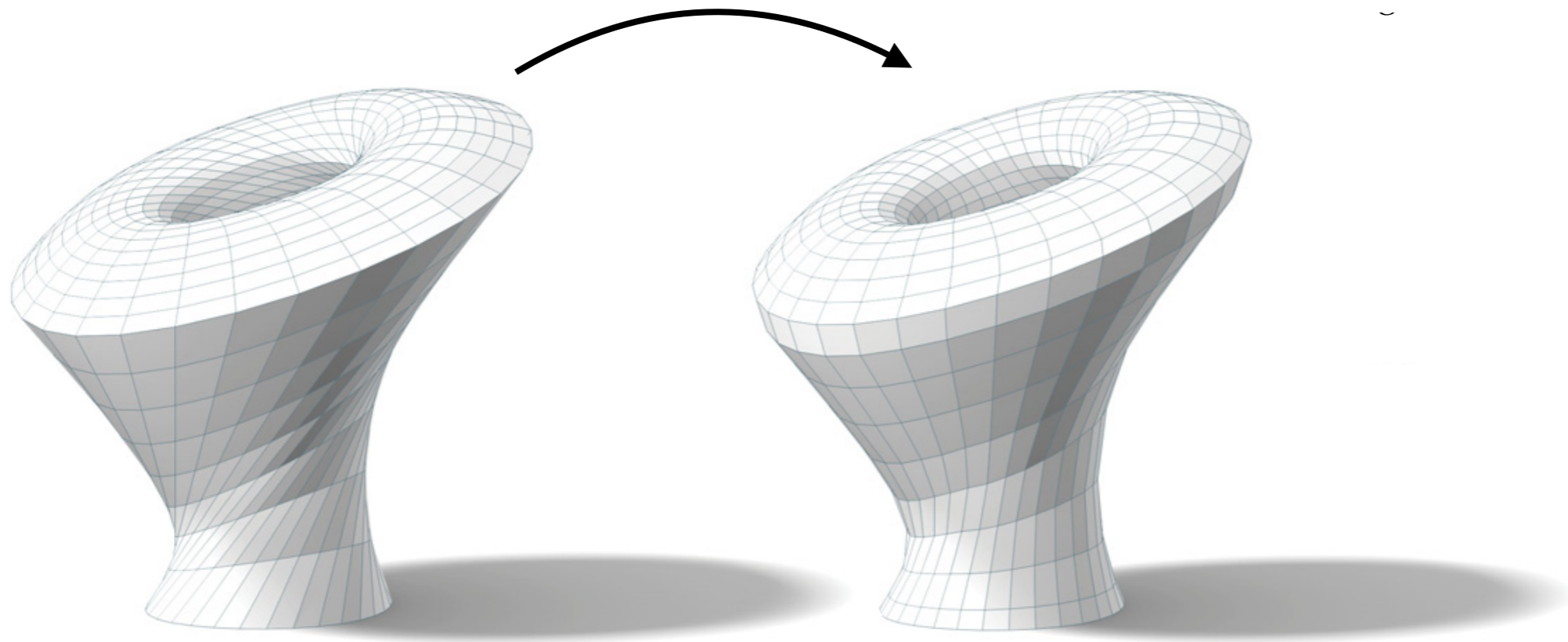
# LibShapeOp Solver

- Output: a new mesh satisfying the constraints



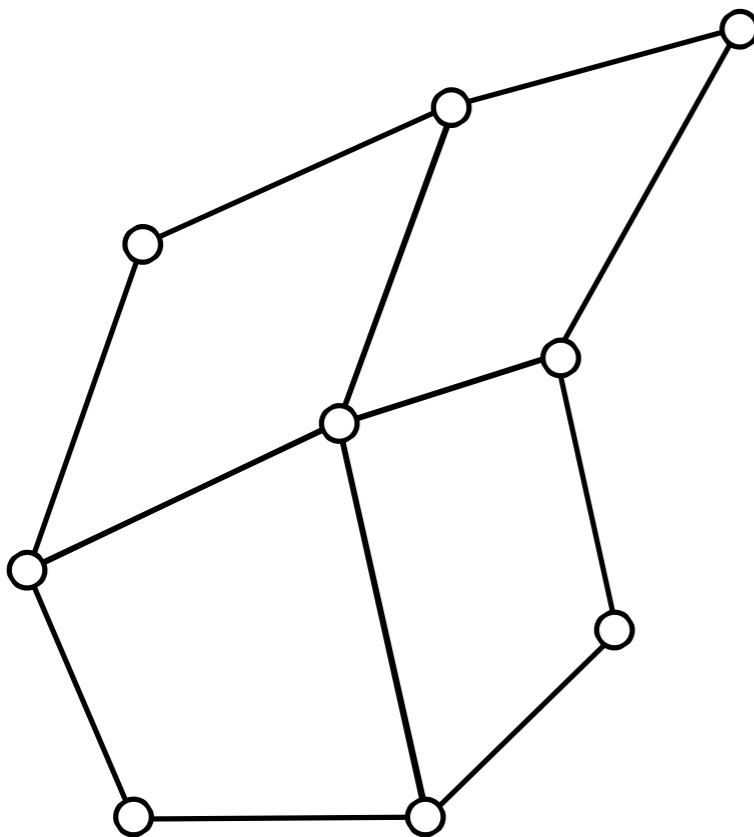
# LibShapeOp Solver

- modified vertex positions
- fixed connectivity



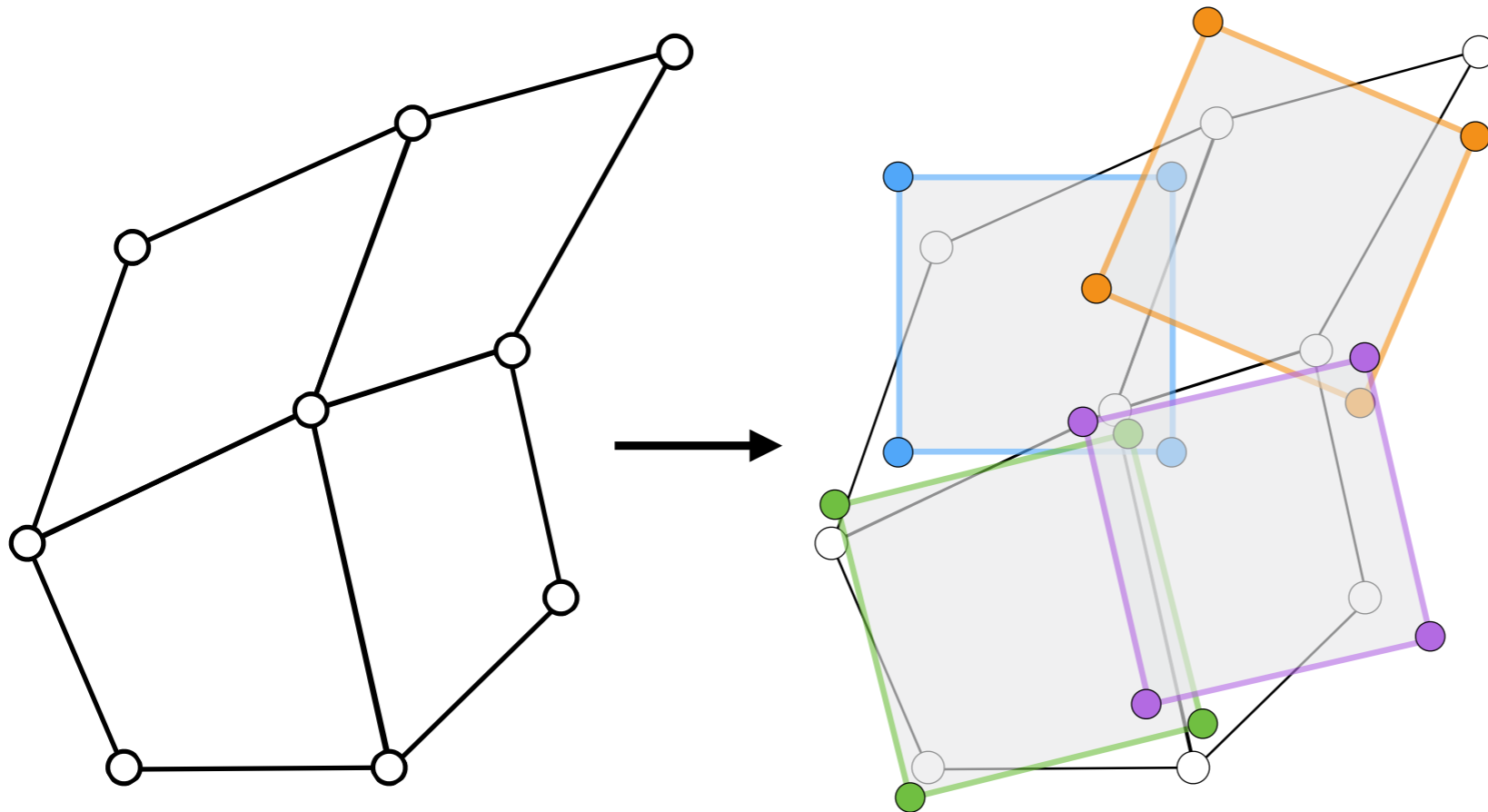
# Local-global Solver

- Constraint: planar faces



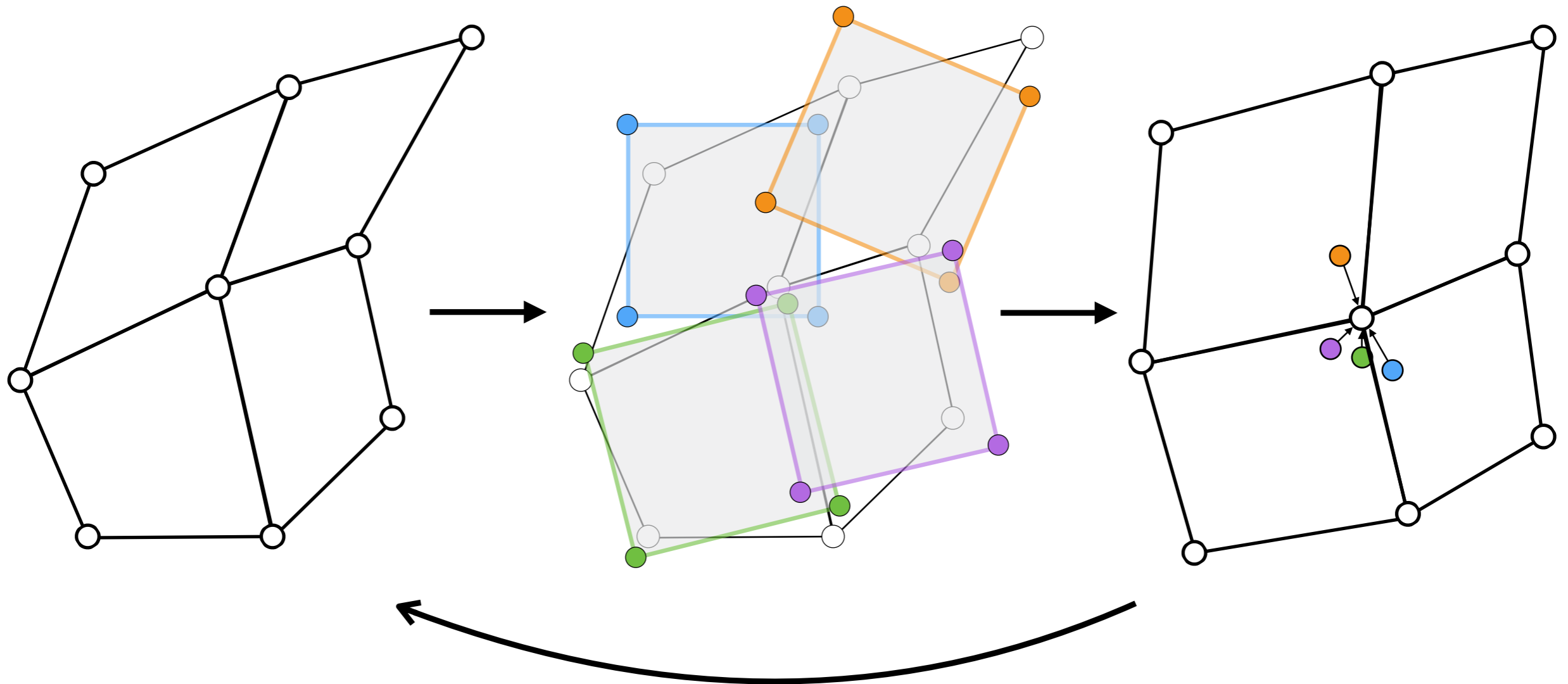
# Local-global Solver

- Local step: projection for each constraint

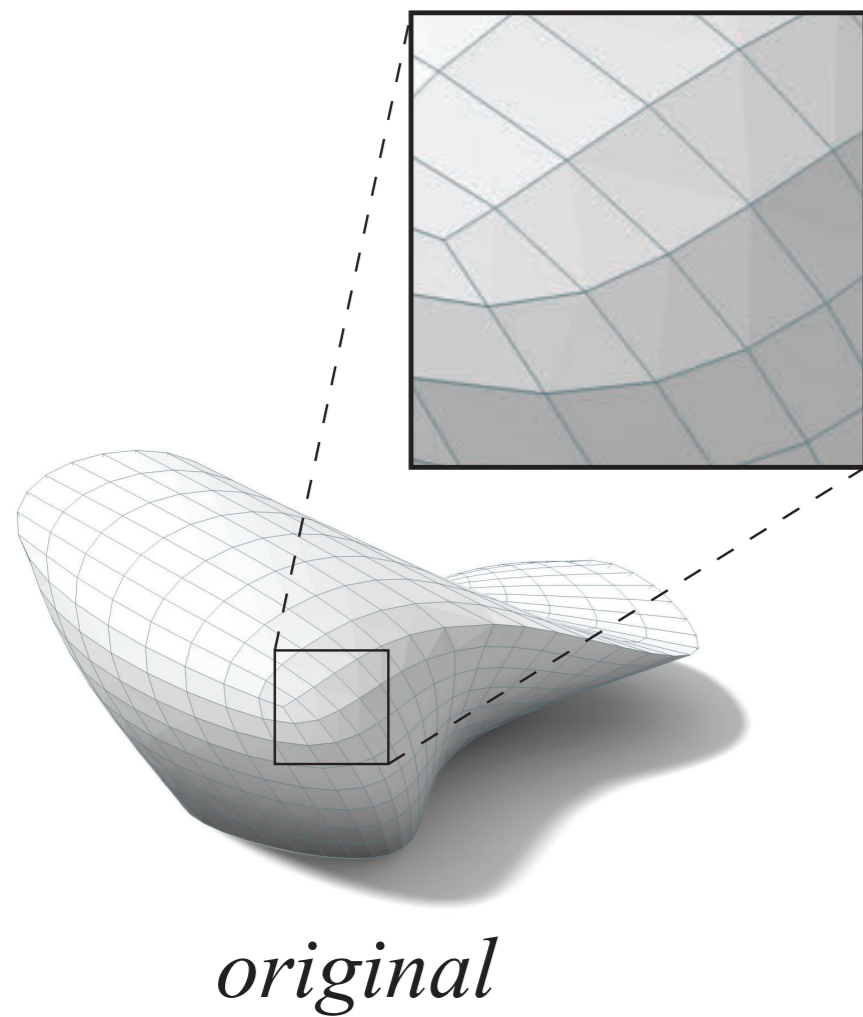


# Local-global Solver

- Global step: update vertices using local projections

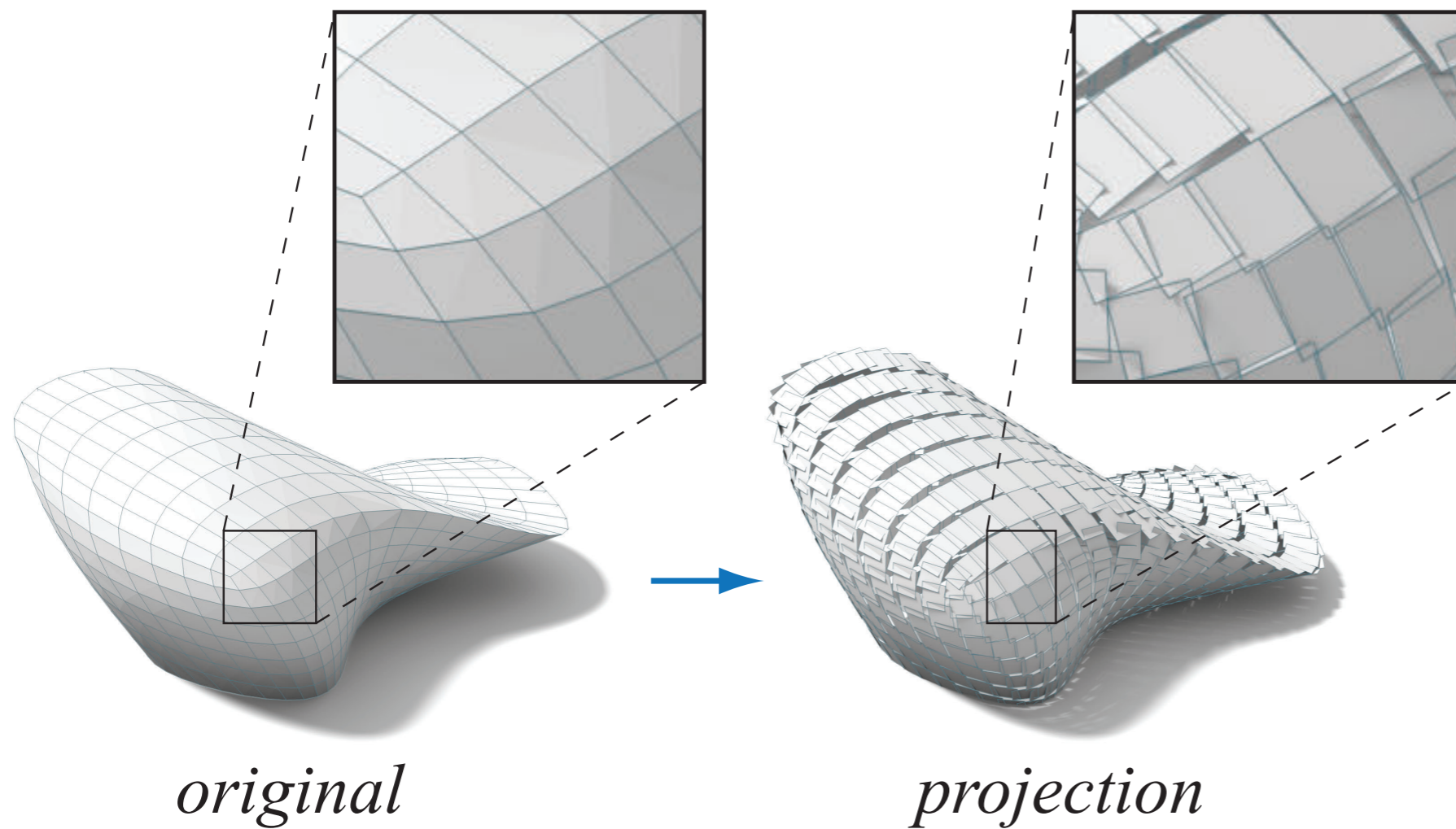


# Example



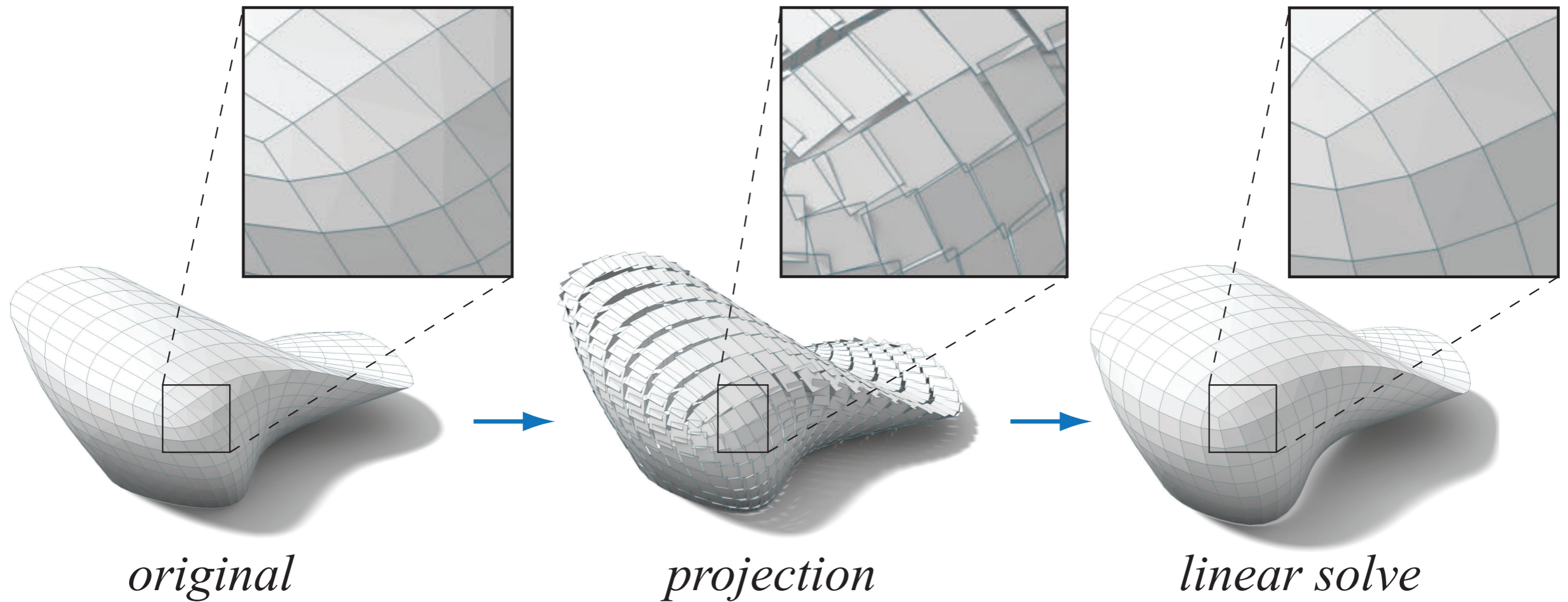
*Constraint: square faces*

# Example



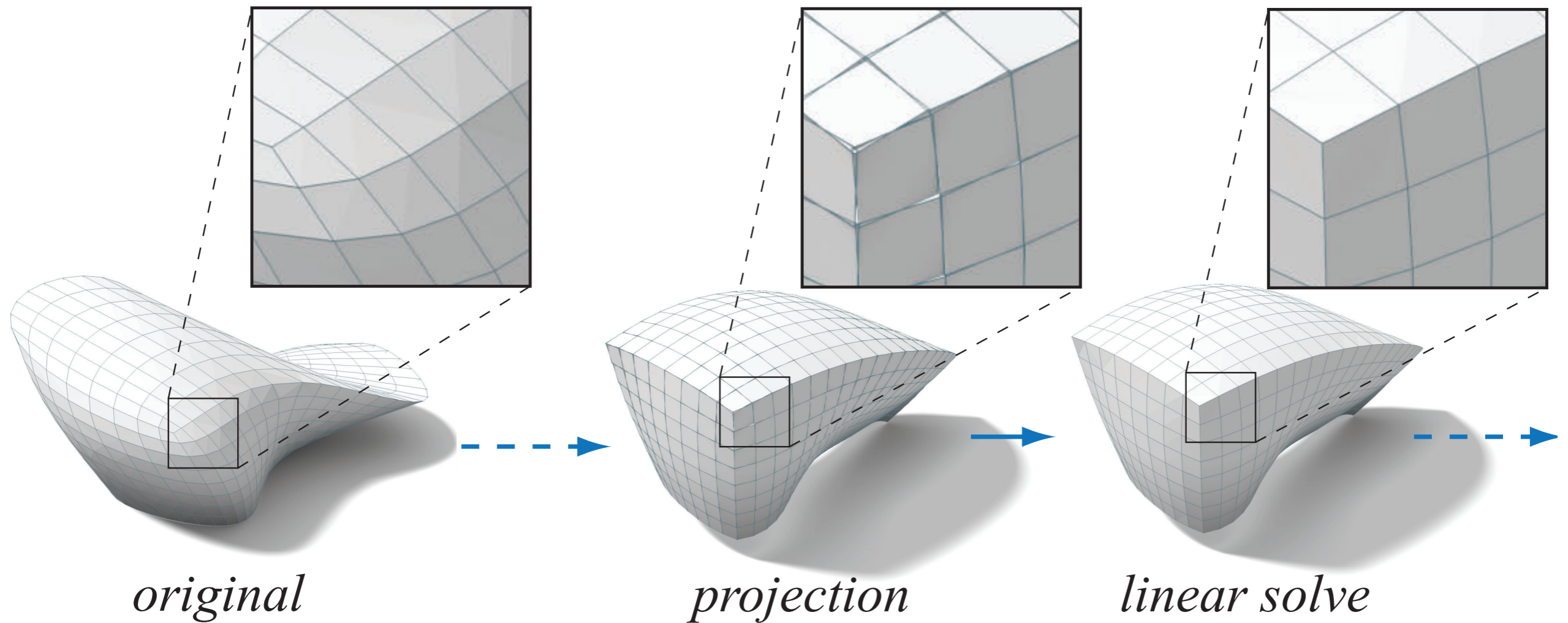
*Constraint: square faces*

# Example



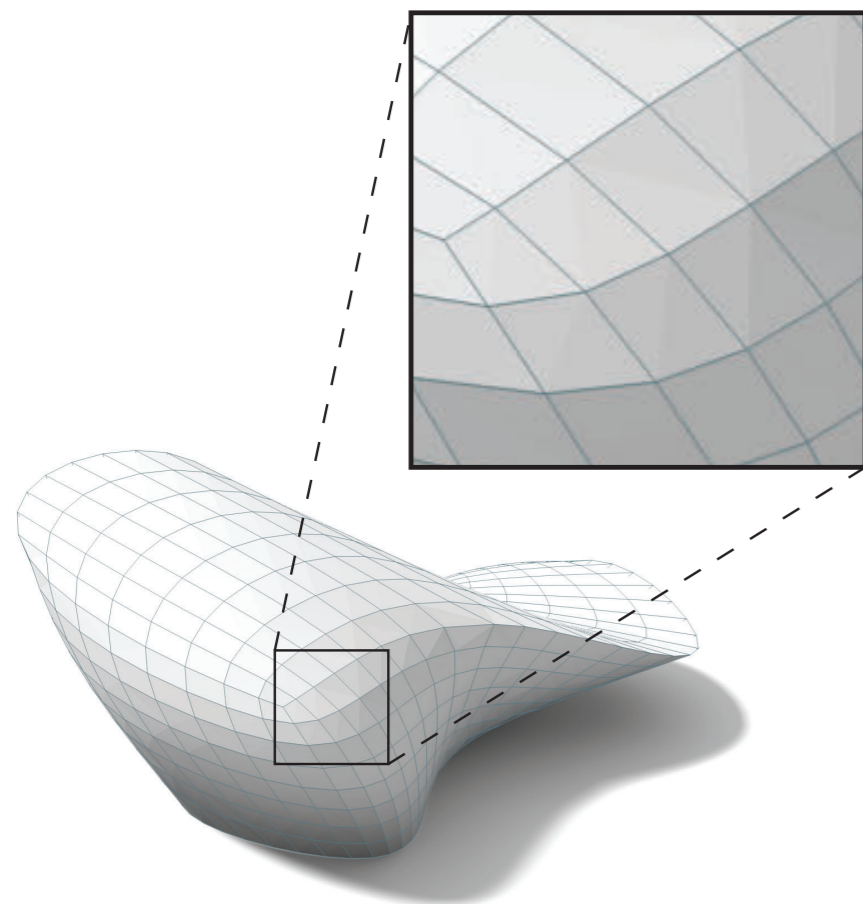
*Constraint: square faces*

# Example

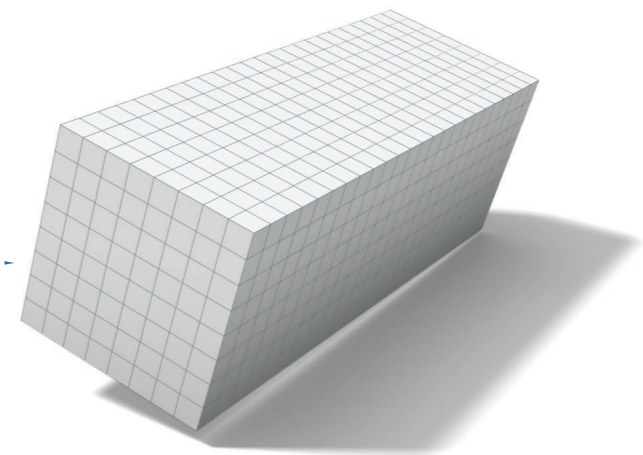


*Constraint: square faces*

# Example



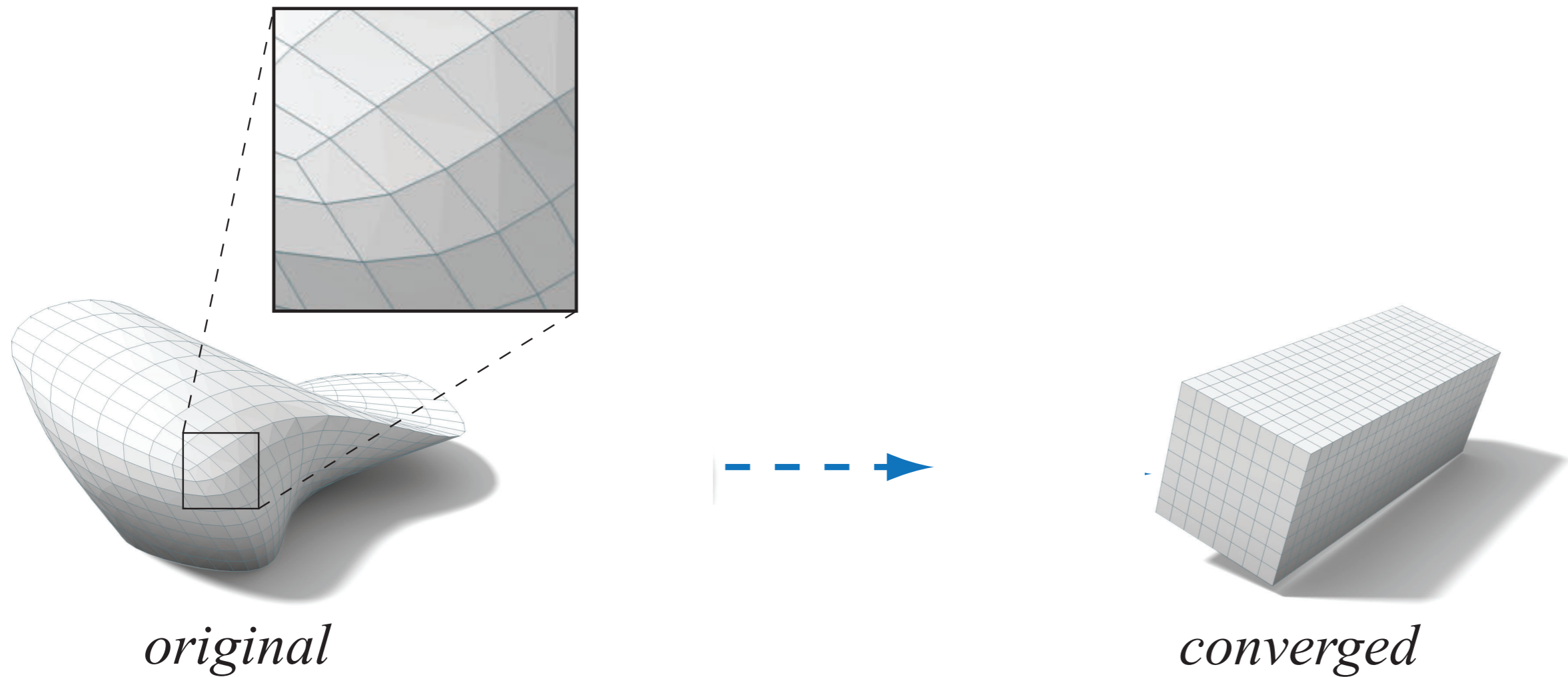
*original*



*converged*

*Constraint: square faces*

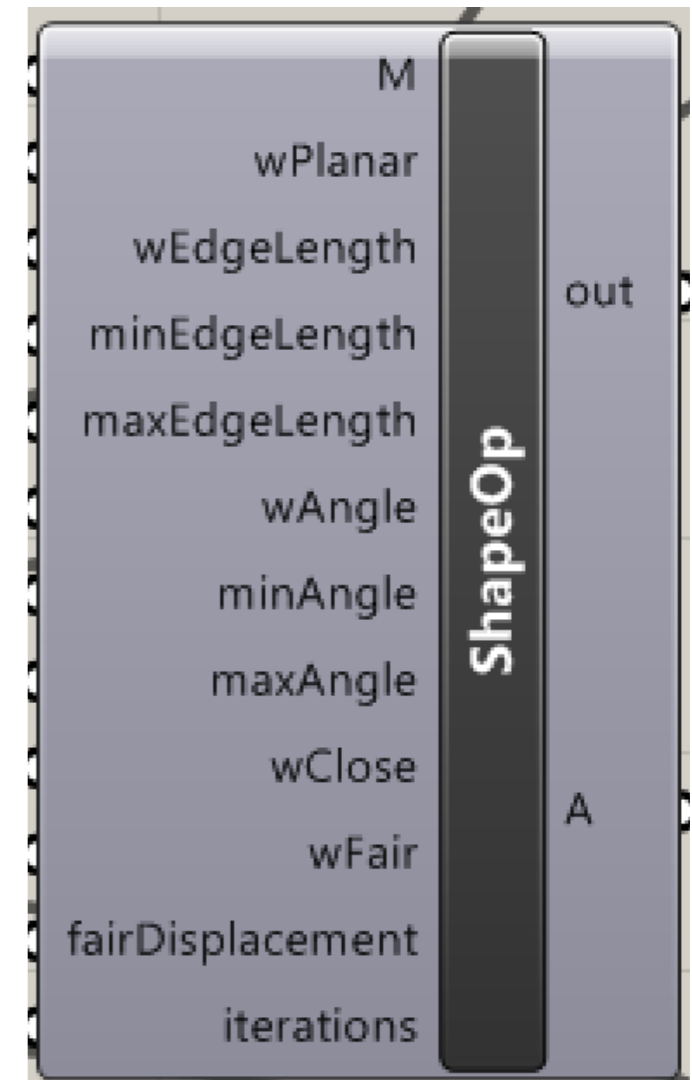
# Constraints



*Constraint: square faces*

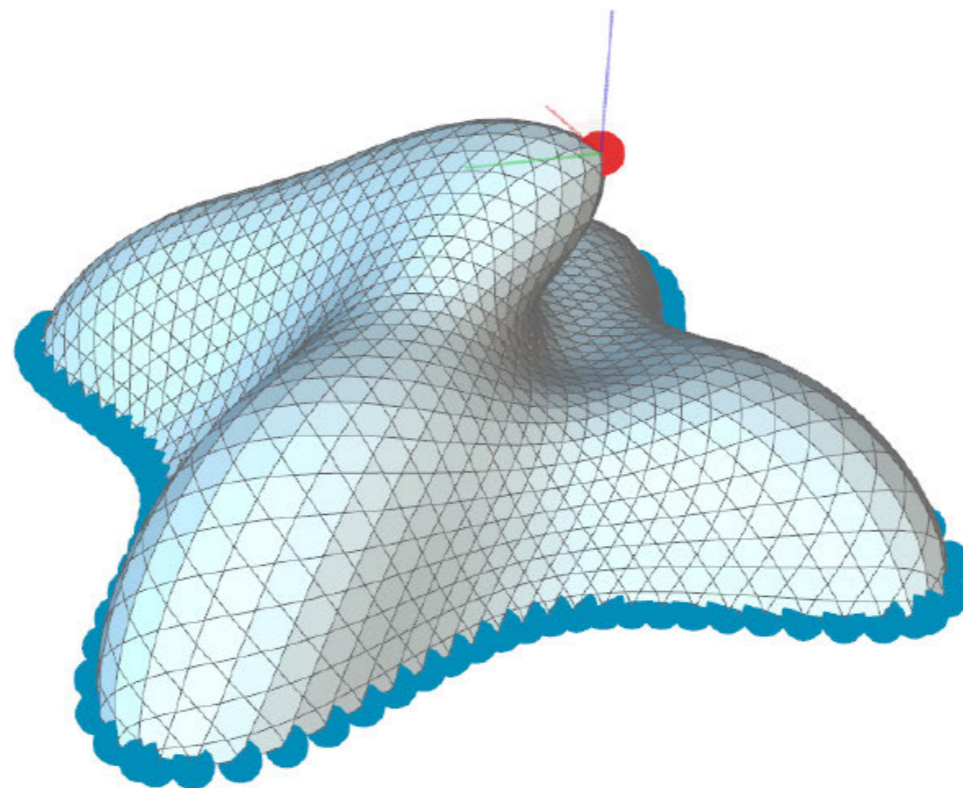
# First Taste of ShapeOp

- Supported constraints
  - planar faces
  - edge length
  - angles
  - closeness to original mesh
  - fairing



# Hard Constraints

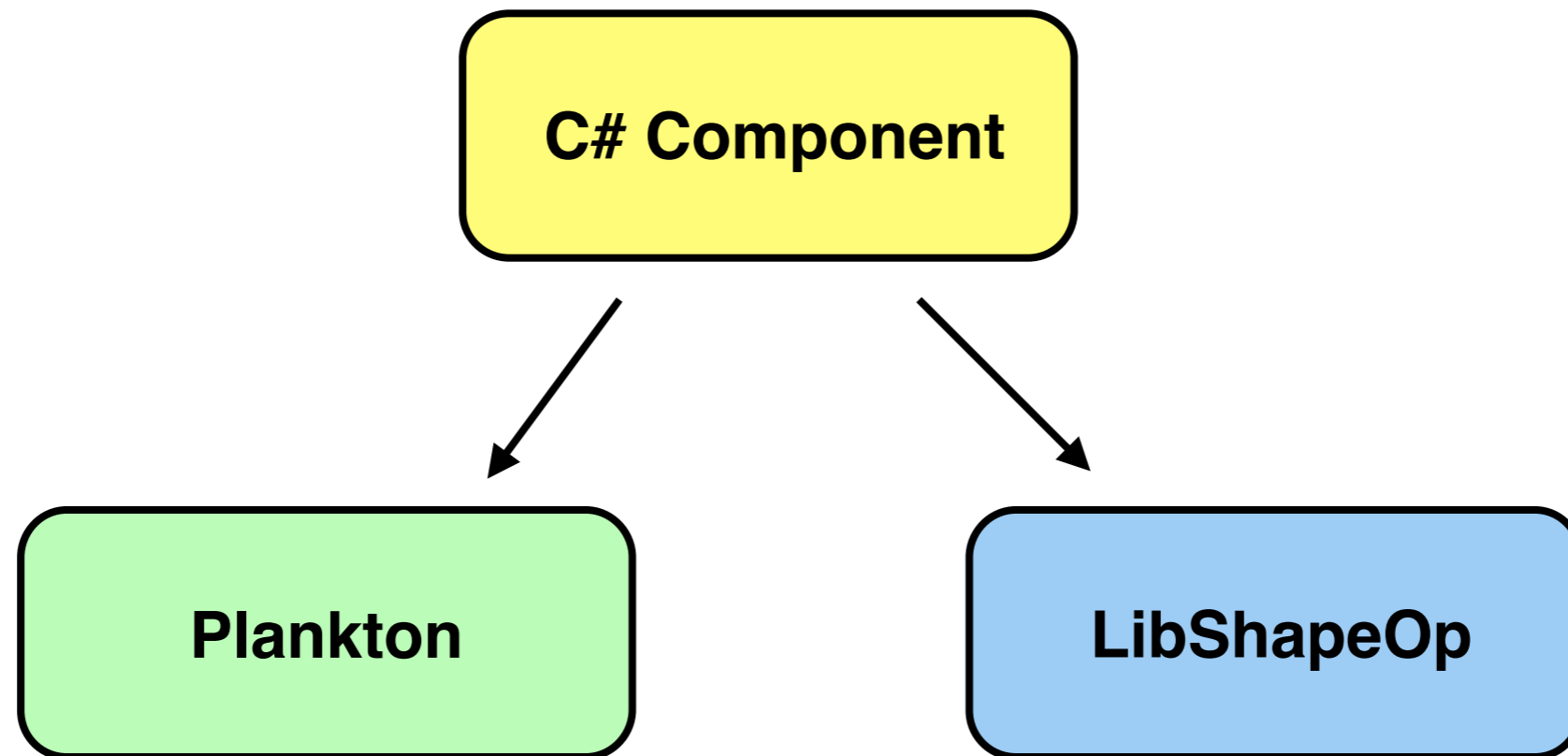
B. Deng, S. Bouaziz, M. Deuss, A. Kaspar, Y. Schwartzburg, M. Pauly. *Interactive Design Exploration for Constrained Meshes*. Computer-Aided Design, 2014.



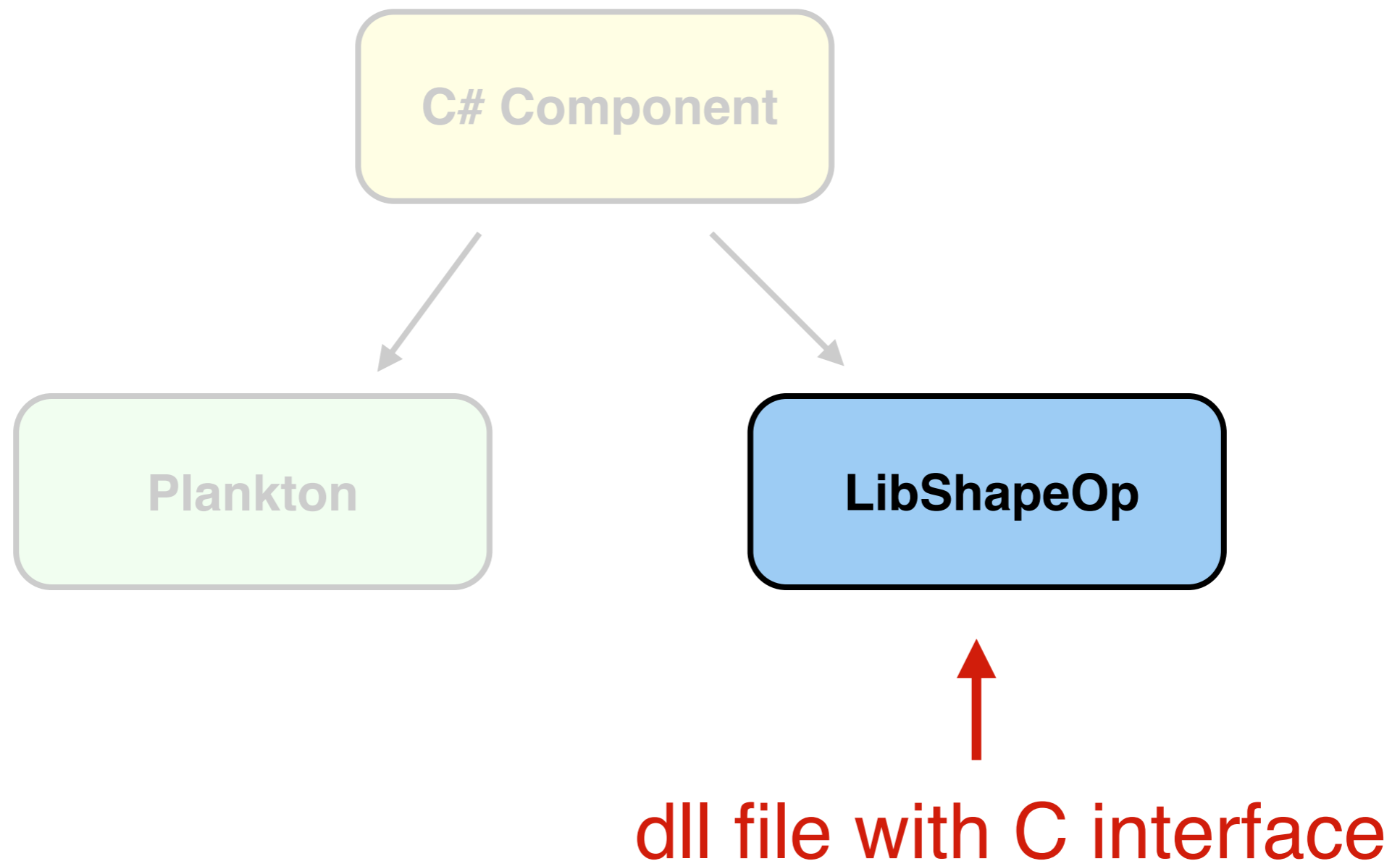
<https://www.youtube.com/watch?v=gh-OAmWPaps>

# C# Scripting with ShapeOp

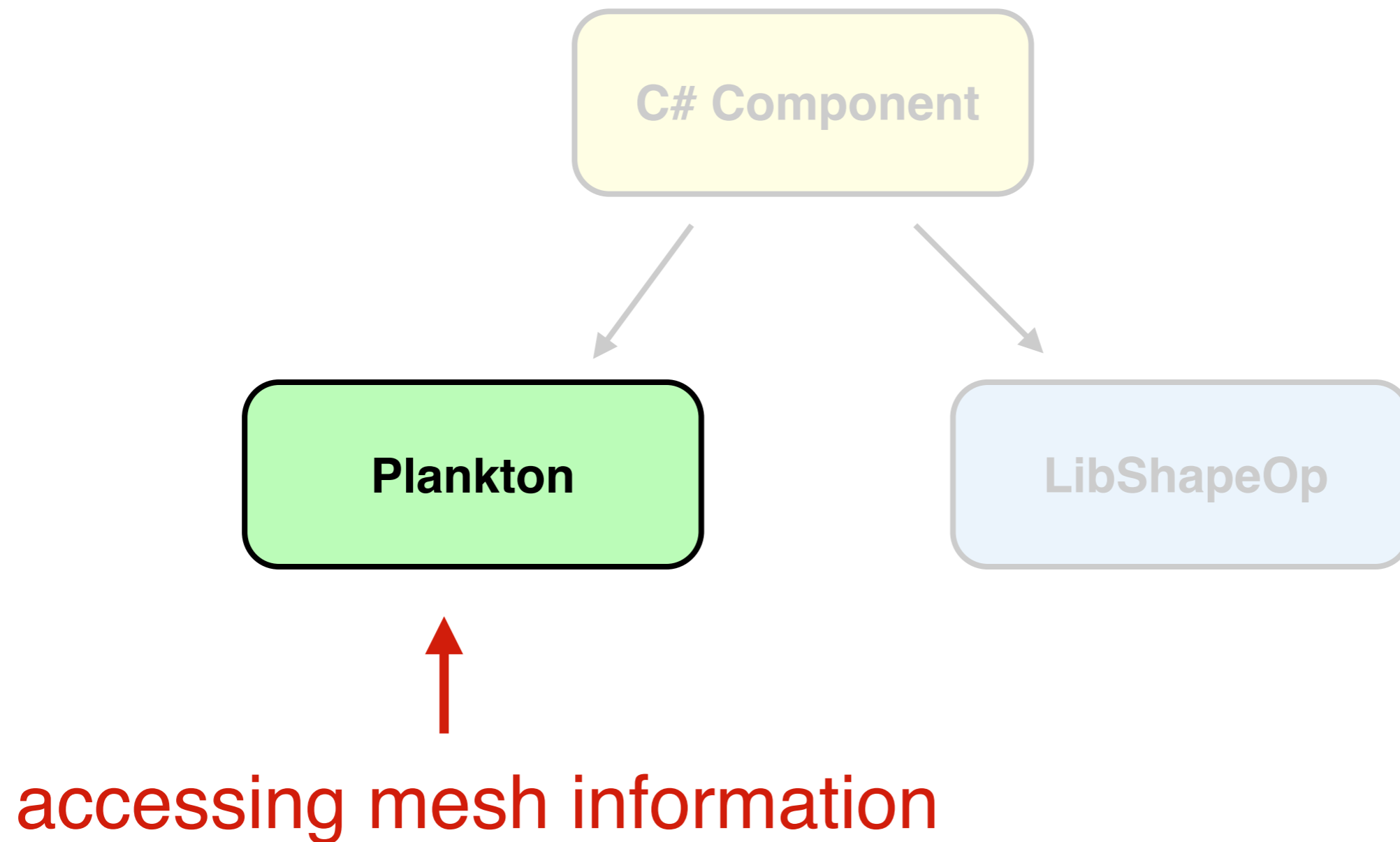
# C# Scripting with ShapeOp



# C# Scripting with ShapeOp



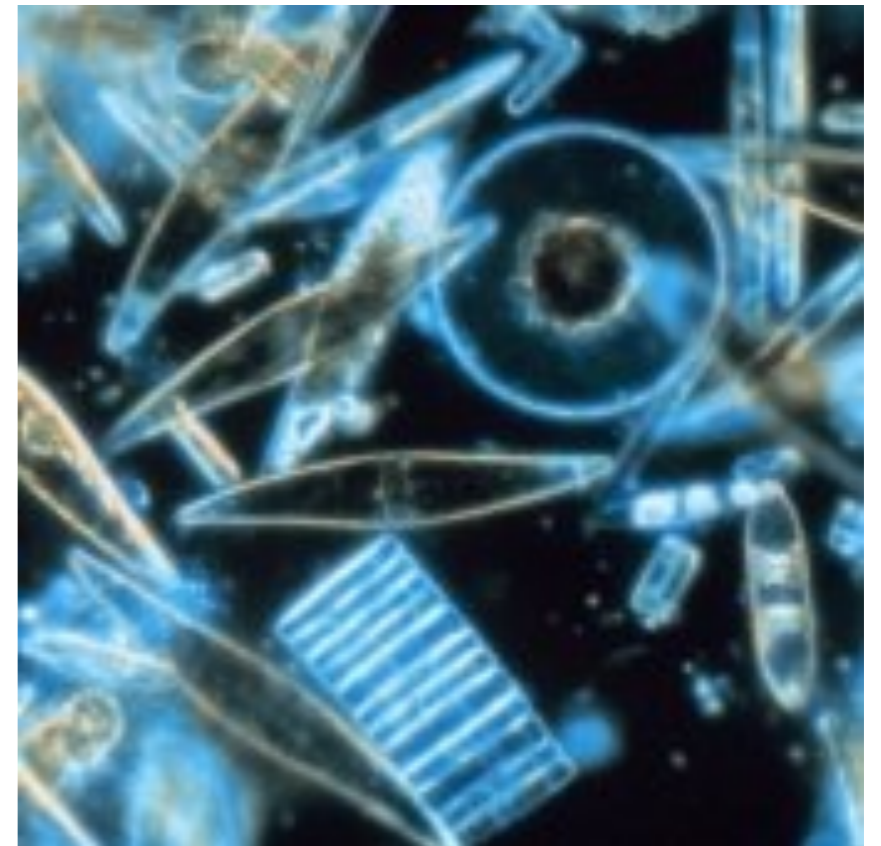
# C# Scripting with ShapeOp



# Plankton

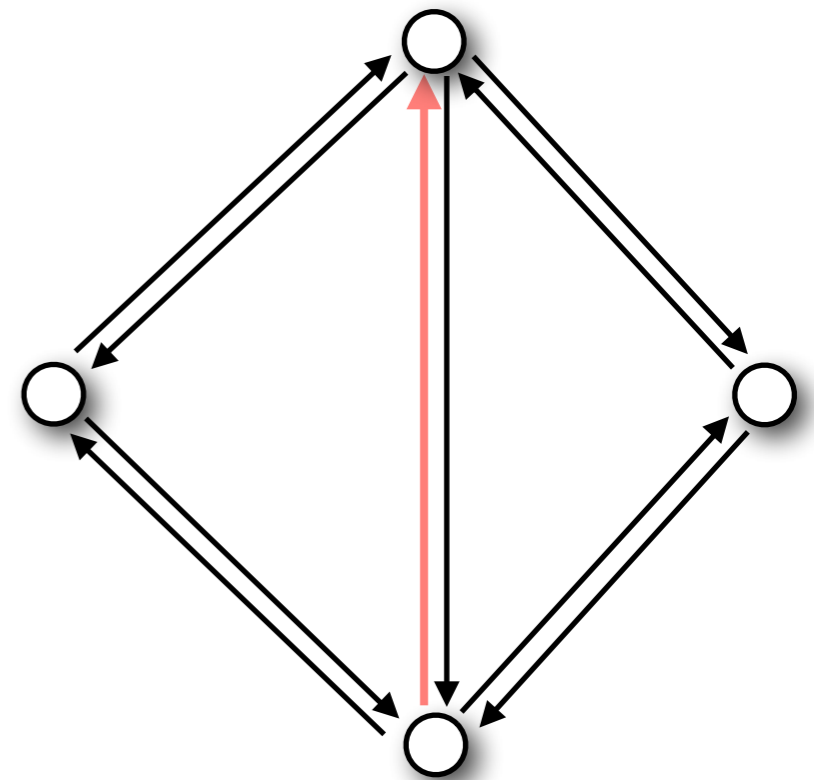
- Open-source C# library implementing half-edge data structure for polygonal meshes

<https://github.com/Dan-Piker/Plankton>



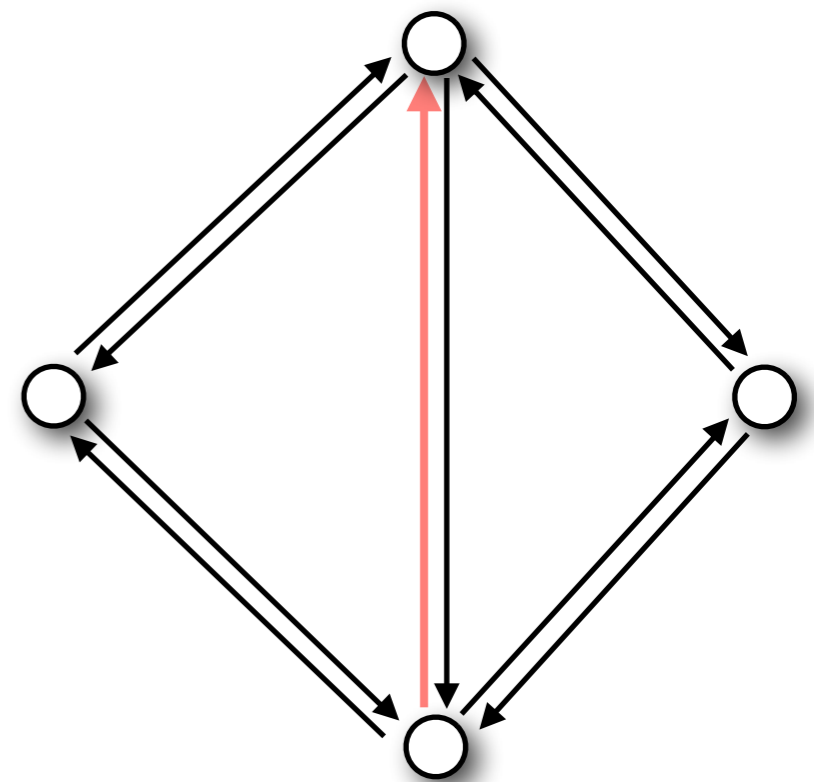
# Half-edge Data Structure

- Efficient query of mesh connectivity
  - e.g., “find all neighboring vertices of a vertex”



# Half-edge Data Structure

- Efficient query of mesh connectivity
  - e.g., “find all neighboring vertices of a vertex”



*For more information:*

<http://doc.cgal.org/latest/HalfedgeDS/index.html>

# Scripting with Plankton

- Create a PlanktonMesh:

```
PlanktonMesh P = M.ToPlanktonMesh();
```

# Scripting with Plankton

- Create a PlanktonMesh:

```
PlanktonMesh P = M.ToPlanktonMesh();
```

- Access the list of all vertices:

```
PlanktonVertexList vtxList = P.Vertices;
```

# Scripting with Plankton

- Create a PlanktonMesh:

```
PlanktonMesh P = M.ToPlanktonMesh();
```

- Access the list of all vertices:

```
PlanktonVertexList vtxList = P.Vertices;
```

- Access vertex coordinates:

```
int i;  
Point3d pt = vtxList[i].ToPoint3d();  
double pt_x = vtxList[i].X;  
double pt_y = vtxList[i].Y;
```

# Scripting with Plankton

## Exercise 1:

Output all vertex positions using `List<Point3d>`

# Scripting with Plankton

- Access the list of all faces:

```
PlanktonMesh P = M.ToPlanktonMesh();
```

```
...
```

```
PlanktonFaceList faceList = P.Faces;
```

# Scripting with Plankton

- Access the list of all faces:

```
PlanktonMesh P = M.ToPlanktonMesh();
```

```
...
```

```
PlanktonFaceList faceList = P.Faces;
```

- Access vertices inside a face:

```
int i;
```

```
...
```

```
int[] faceVtxList = faceList.GetFaceVertices(i);
```

# Scripting with Plankton

## Exercise 2:

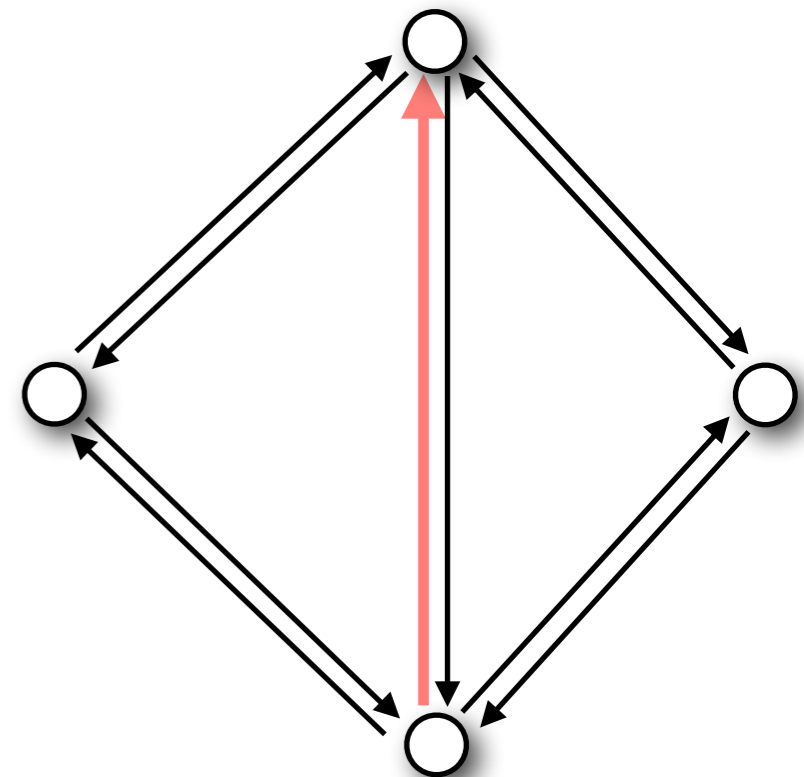
Output face centroids using `List<Point3d>`

# Scripting with Plankton

- Access the list of all half-edges:

```
PlanktonMesh P = M.ToPlanktonMesh();
```

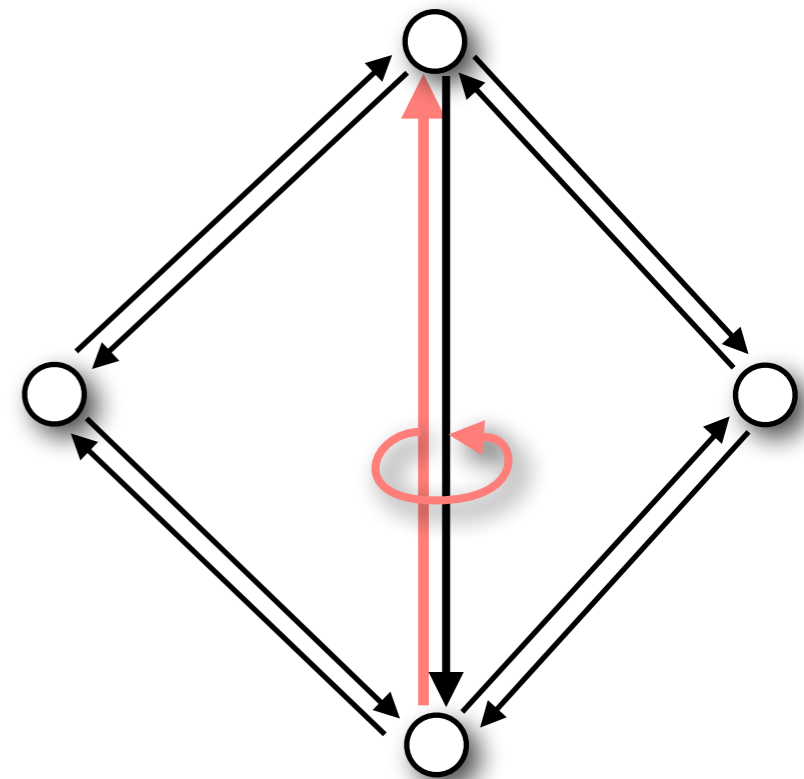
```
PlanktonHalfEdgeList halfEdgeList = P.Halfedges;
```



# Scripting with Plankton

- Opposite half-edge:

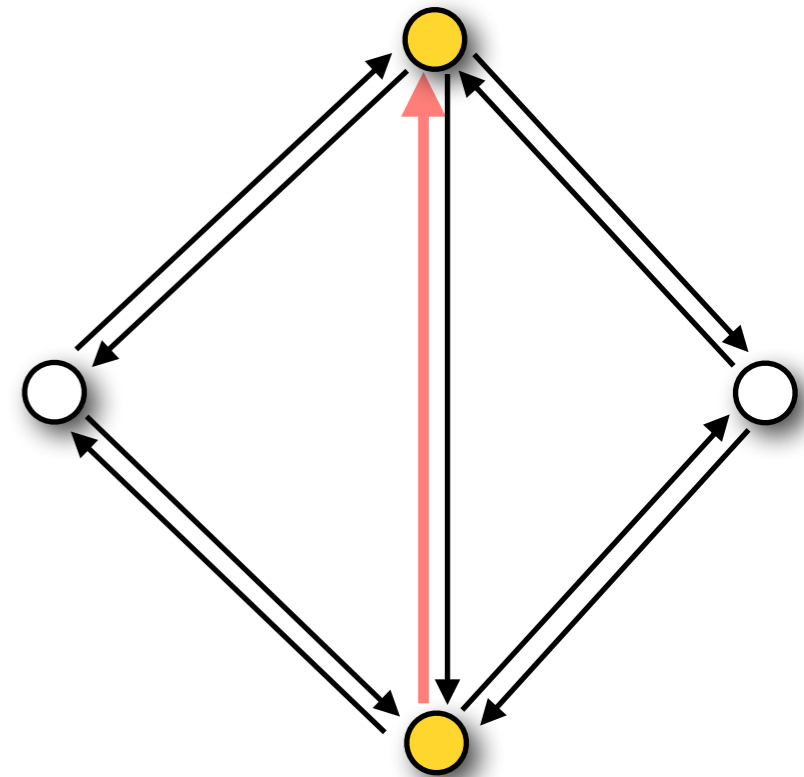
```
int j = halfEdgeList.GetPairHalfedge(i);
```



# Scripting with Plankton

- Vertices of a half-edge:

```
int[] vtxIdx = halfEdgeList.GetVertices(i);
```



# Scripting with Plankton

## Exercise 3:

Display edge mid-points

# Scripting with Plankton

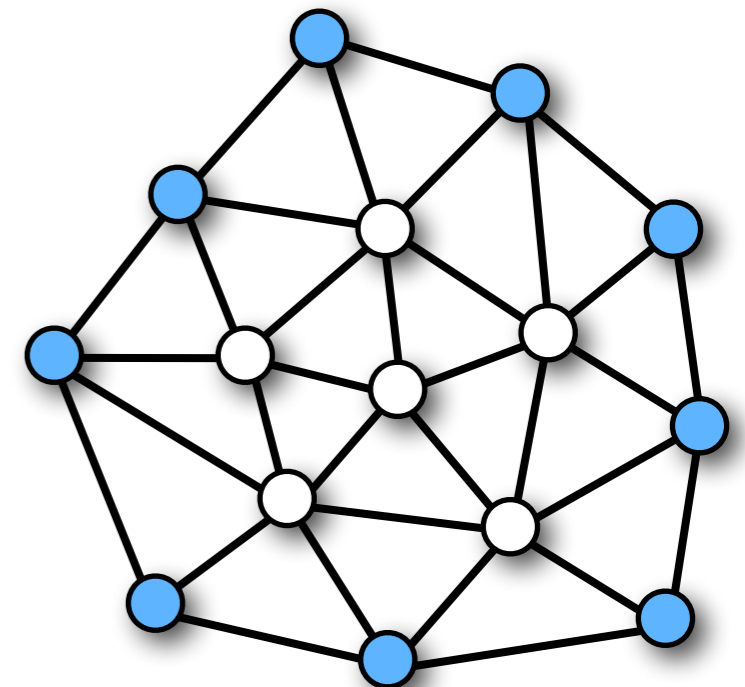
## Exercise 3:

Display edge mid-points

# Scripting with Plankton

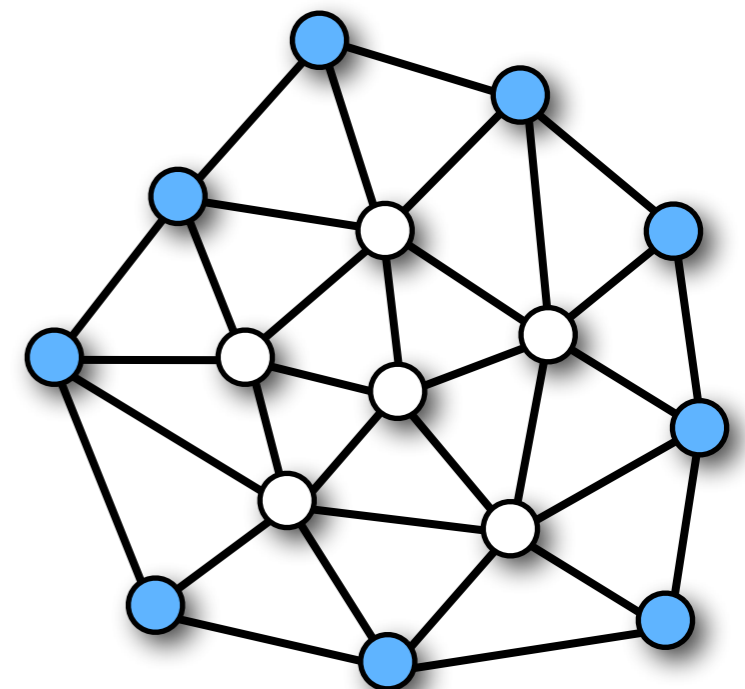
- Check if a vertex is on boundary

```
PlanktonMesh P = M.ToPlanktonMesh();  
PlanktonVertexList vtxList = P.Vertices;  
bool boundaryVtx = vtxList.IsBoundary(i);
```



# Scripting with Plankton

Exercise 4:  
Output boundary vertex positions



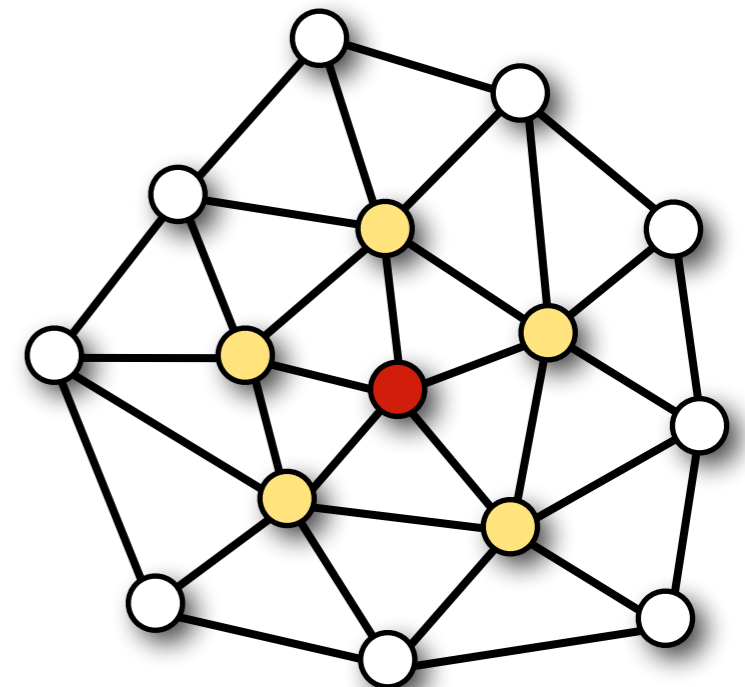
# Plankton

- Access neighbor vertices of a vertex:

```
PlanktonMesh P = M.ToPlanktonMesh();
```

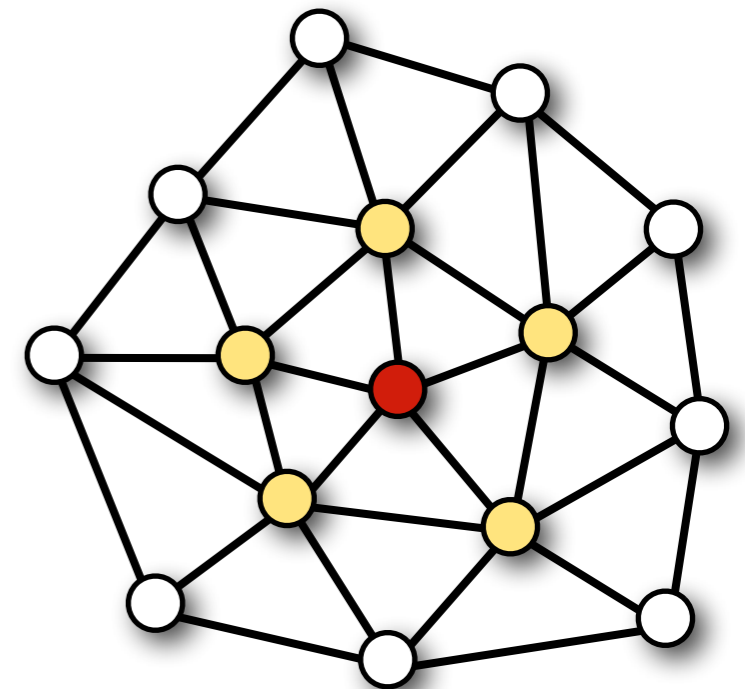
```
PlanktonVertexList vtxList = P.Vertices;
```

```
int[] neighborVtx = vtxList.GetVertexNeighbours(i);
```

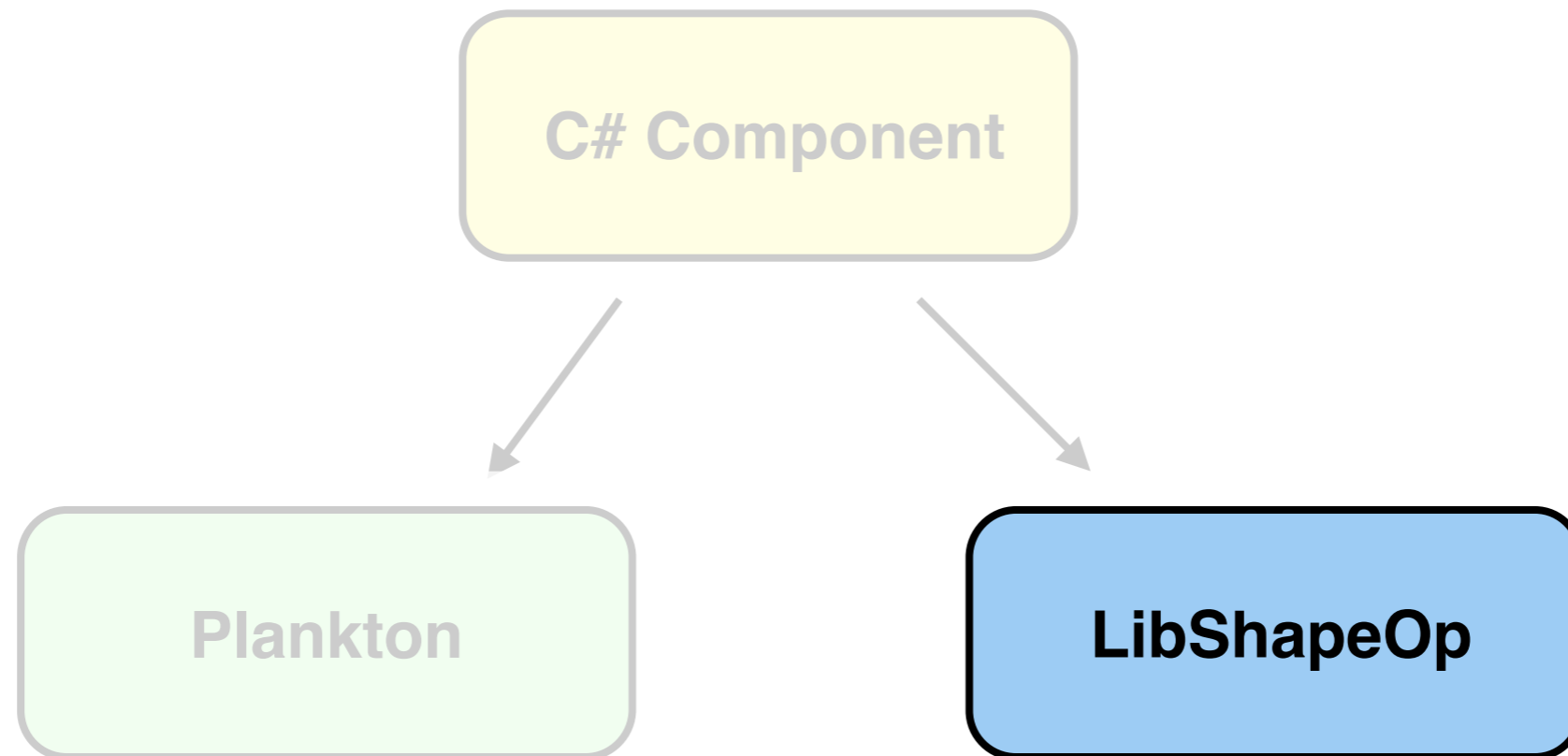


# Plankton

Exercise 5:  
Output centroids of neighboring vertices

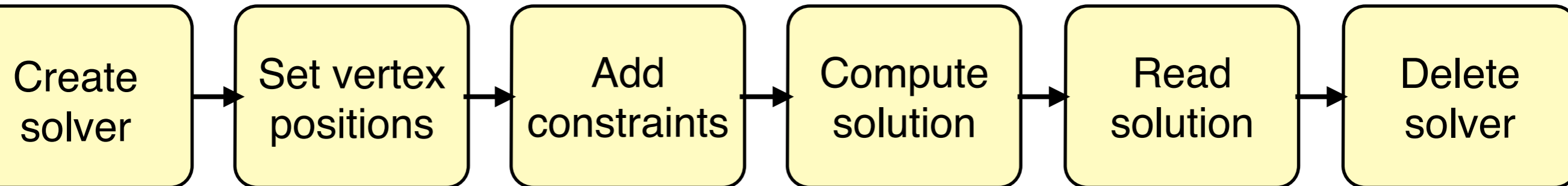


# C# Scripting with ShapeOp



# LibShapeOp

- Overview:



# LibShapeOp

Create  
solver

Set vertex  
positions

Add  
constraints

Compute  
solution

Read  
solution

Delete  
solver

```
[DllImport("ShapeOp.dll", CallingConvention=CallingConvention.Cdecl)]  
static extern IntPtr shapeop_create();
```

For more information:

[http://msdn.microsoft.com/en-us/library/aa288468\(v=vs.71\).aspx](http://msdn.microsoft.com/en-us/library/aa288468(v=vs.71).aspx)

# LibShapeOp

Create  
solver

Set vertex  
positions

Add  
constraints

Compute  
solution

Read  
solution

Delete  
solver

```
[DllImport("ShapeOp.dll", CallingConvention=CallingConvention.Cdecl)]  
static extern void shapeop_delete(IntPtr op);
```

# LibShapeOp

Create  
solver

Set vertex  
positions

Add  
constraints

Compute  
solution

Read  
solution

Delete  
solver

```
[DllImport("ShapeOp.dll", CallingConvention=CallingConvention.Cdecl)]
static extern void shapeop_setPoints(IntPtr op, double[] points, int nb_points);

[DllImport("ShapeOp.dll", CallingConvention=CallingConvention.Cdecl)]
static extern void shapeop_getPoints(IntPtr op, double[] points, int nb_points);
```

# LibShapeOp

Create  
solver

Set vertex  
positions

Add  
constraints

Compute  
solution

Read  
solution

Delete  
solver

```
[DllImport("ShapeOp.dll", CallingConvention=CallingConvention.Cdecl)]  
static extern void shapeop_setPoints(IntPtr op, double[] points, int nb_points);  
  
[DllImport("ShapeOp.dll", CallingConvention=CallingConvention.Cdecl)]  
static extern void shapeop_getPoints(IntPtr op, double[] points, int nb_points);
```

Coordinates array:  $(x_1, y_1, z_1, \dots, x_n, y_n, z_n)$

# LibShapeOp

Create  
solver

Set vertex  
positions

Add  
constraints

Compute  
solution

Read  
solution

Delete  
solver

```
[DllImport("ShapeOp.dll", CallingConvention=CallingConvention.Cdecl)]  
static extern int shapeop_init(IntPtr op);
```

```
[DllImport("ShapeOp.dll", CallingConvention=CallingConvention.Cdecl)]  
static extern int shapeop_solve(IntPtr op, int iteration);
```

# LibShapeOp

Create  
solver

Set vertex  
positions

Add  
constraints

Compute  
solution

Read  
solution

Delete  
solver

```
[DllImport("ShapeOp.dll", CallingConvention=CallingConvention.Cdecl)]  
static extern int shapeop_init(IntPtr op);
```

```
[DllImport("ShapeOp.dll", CallingConvention=CallingConvention.Cdecl)]  
static extern int shapeop_solve(IntPtr op, int iteration);
```

Num. of iterations

# LibShapeOp

Create  
solver

Set vertex  
positions

Add  
constraints

Compute  
solution

Read  
solution

Delete  
solver

# LibShapeOp

Create  
solver

Set vertex  
positions

Add  
constraints

Compute  
solution

Read  
solution

Delete  
solver

```
[DllImport("ShapeOp.dll", CallingConvention=CallingConvention.Cdecl)]
static extern int shapeop_addConstraint(IntPtr op, IntPtr constraintType,
int[] ids, int nb_ids, double weight);

[DllImport("ShapeOp.dll", CallingConvention=CallingConvention.Cdecl)]
static extern int shapeop_editConstraint(IntPtr op, IntPtr constraintType,
int constraint_id, double[] scalar_params, int nb_param);
```

# LibShapeOp

Create  
solver

Set vertex  
positions

Add  
constraints

Compute  
solution

Read  
solution

Delete  
solver

```
[DllImport("ShapeOp.dll", CallingConvention=CallingConvention.Cdecl)]
static extern int shapeop_addConstraint(IntPtr op, IntPtr constraintType,
int[] ids, int nb_ids, double weight);

[DllImport("ShapeOp.dll", CallingConvention=CallingConvention.Cdecl)]
static extern int shapeop_editConstraint(IntPtr op, IntPtr constraintType,
int constraint_id, double[] scalar_params, int nb_param);
```

# For the Passionate

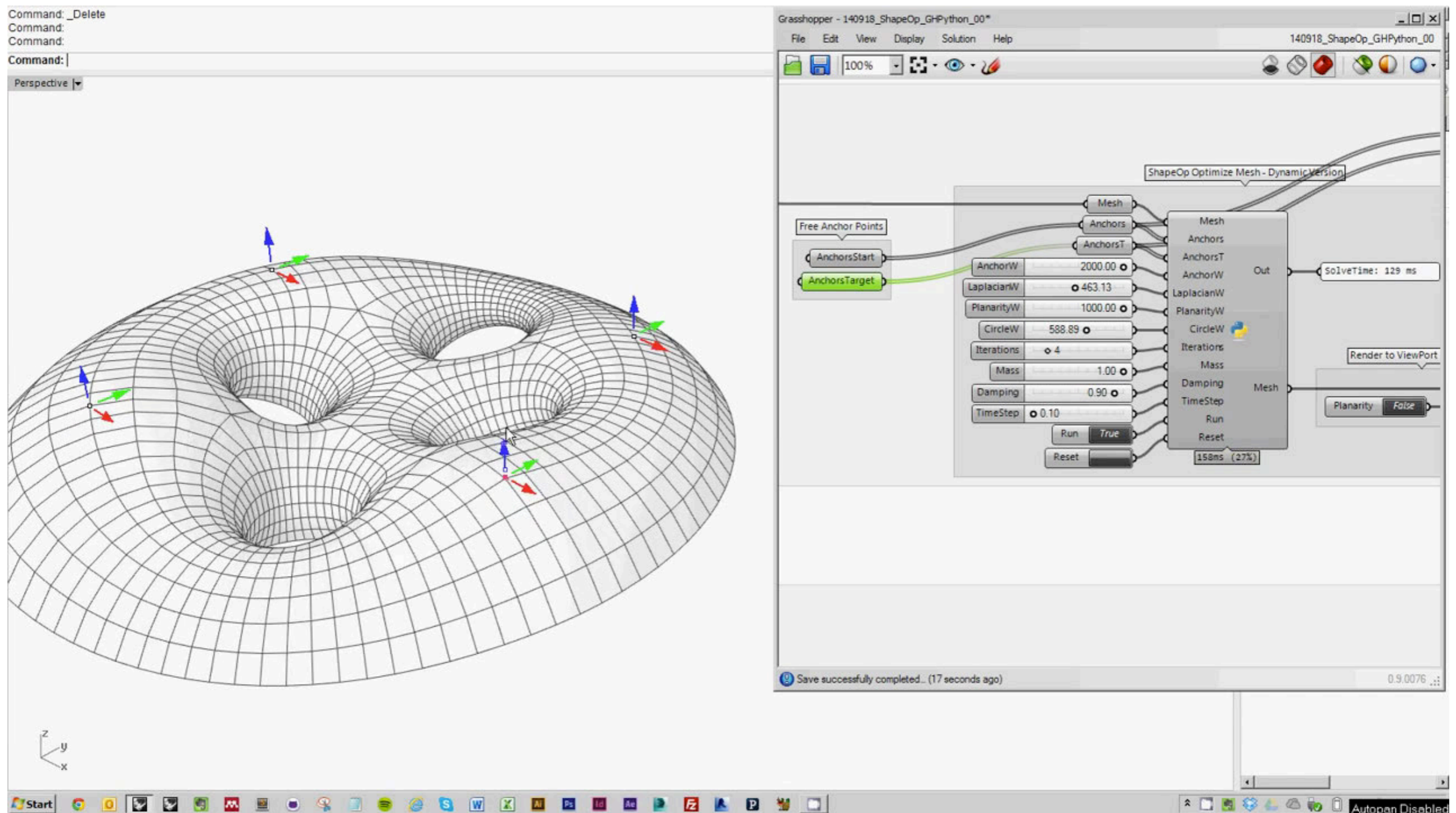
- Adding new constraints
  - implement projection operators

## For more information:

Sofien Bouaziz, Mario Deuss, Yuliy Schwartzburg, Thibaut Weise, and Mark Pauly.  
Shape-Up: Shaping Discrete Geometry with Projections. *Computer Graphics Forum*  
31(5): 1657-1667. 2012.

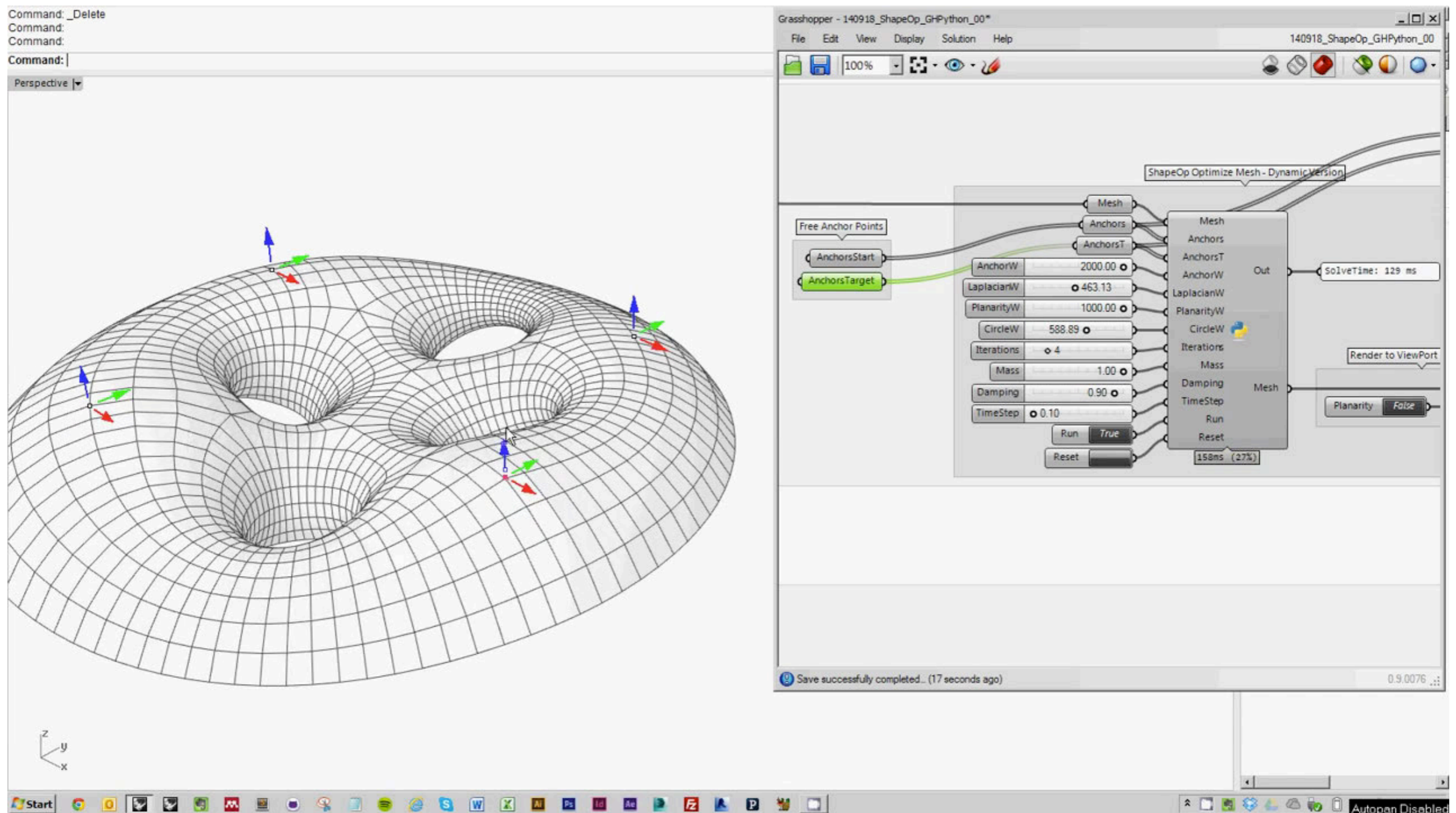
# GhPython Interface

- Interaction, dynamic simulation, etc.



# GhPython Interface

- Interaction, dynamic simulation, etc.



# Thank you!