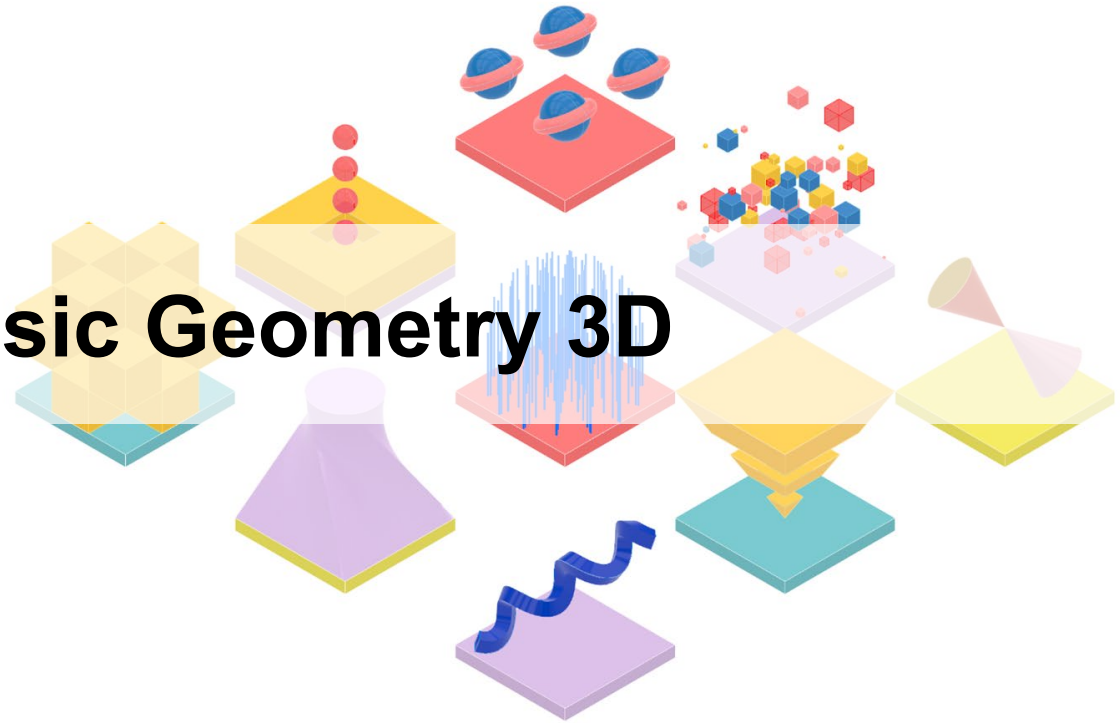
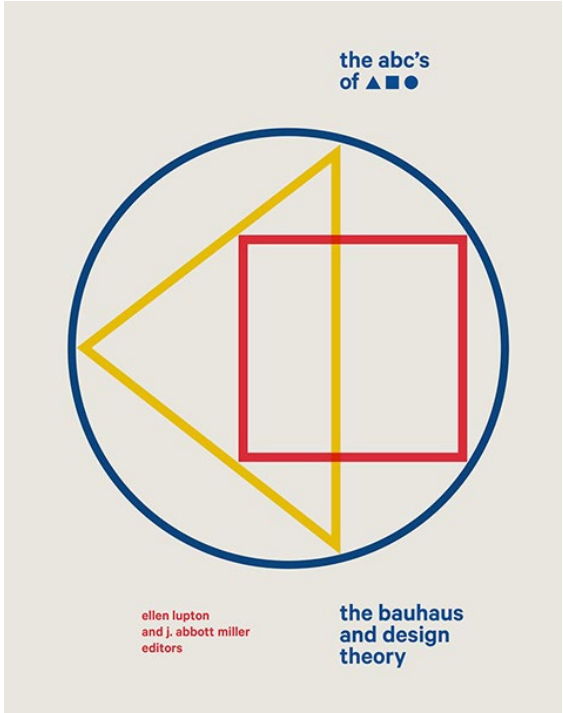
Basic Geometry 3D



Introduction

Geometry and Nature in Bauhaus Theory

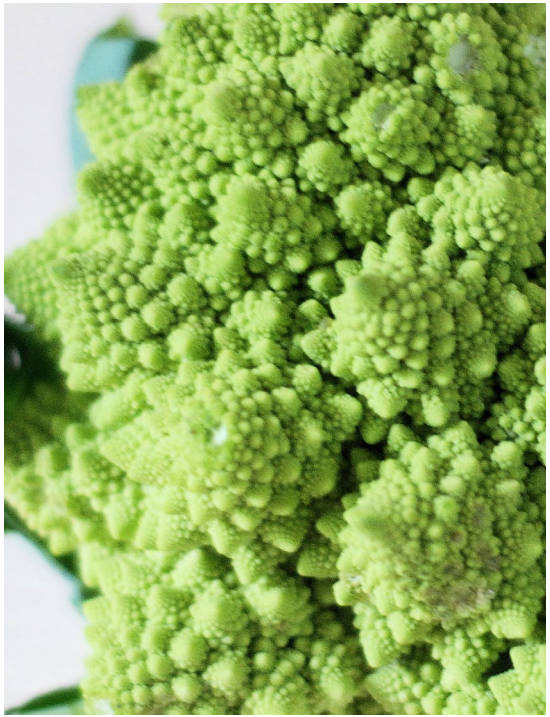


The ABC's of Triangle Square Circle /
Princeton Architectural Press

Geometric Education and Its Manifestation

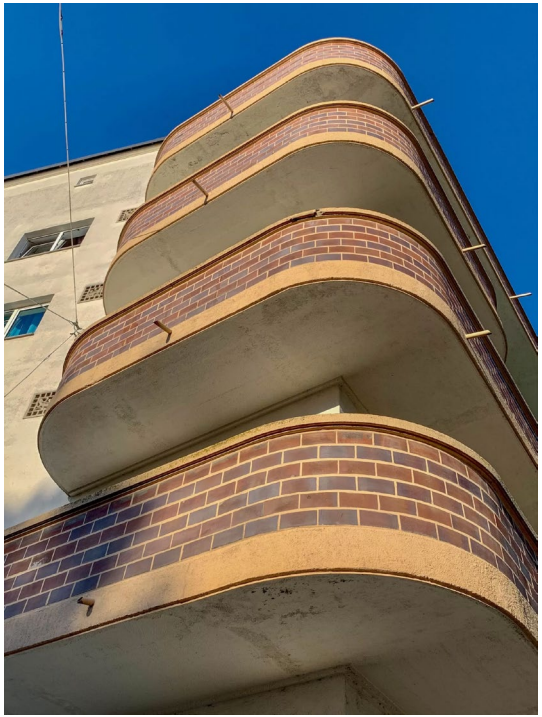
Bauhaus Campus in Dessau, Germany /
Wikipedia, M_H.DE , CC BY-SA 3.0

The Limits of Geometric Language



Romanesco broccoli approximates natural fractals / Wikipedia, Jon Sullivan, Public Domain

The Bauhaus Geometric Vocabulary



Amerikanerblock, München Neuhausen, 1930/31, Otho Orlando Kurz / https://vielfaltdermoderne.de/amerikanerblock/

Representation and Understanding

Villa of Greta and Fritz Tugendhat, Brno, Czechoslovakia (now Czechia) in 1930

Geometry as an Enabling Framework

Villa of Greta and Fritz Tugendhat, Brno, Czechoslovakia (now Czechia) in 1930

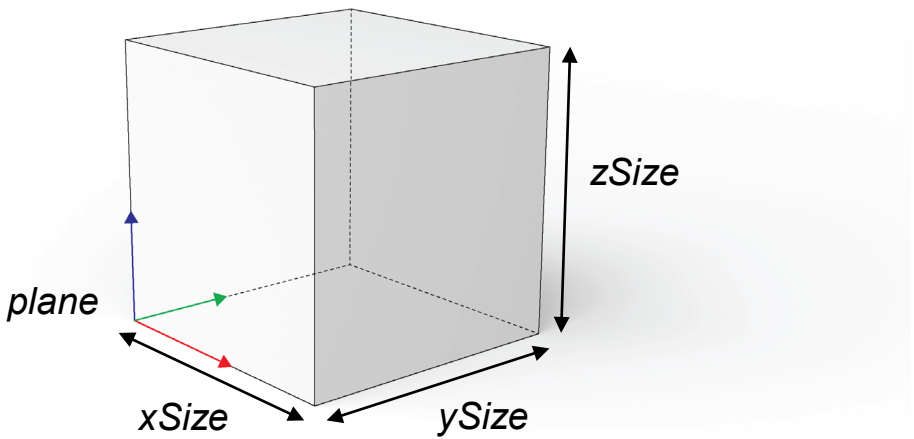
Content

1. Solid Primitives (Box, Cone, Cylinder, Sphere, Torus, etc.)
2. Surfaces (Planar, 2DNurbs, Loft, Sweep, etc.)
3. Boundary Representations (Breps)
4. Transformations
5. Boolean Operations

Solid Primitives

Solid Primitives

class **Rhino.Geometry**.**Box**(*plane: Plane, xSize: Float, ySize: Float, zSize: Float*)



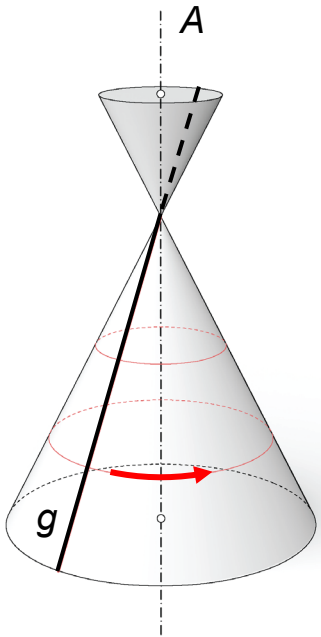
Solid Primitives

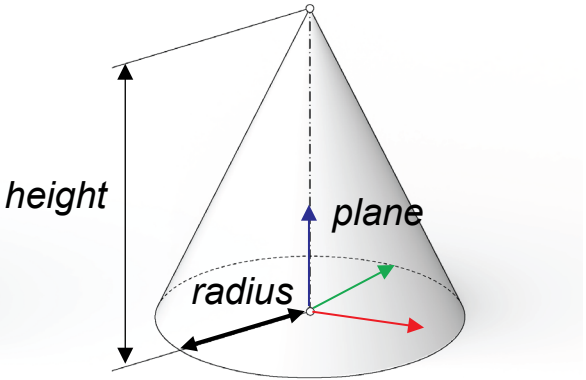
class **Rhino.Geometry**.**Cone***(plane: Plane, height: Float, radius: Float)*

A ***cone surface*** is obtained by revolving a line ***g*** around an intersecting axis ***A***

or

defined by a ***plane***, a radius ***r*** of the base circle and the height ***h***.





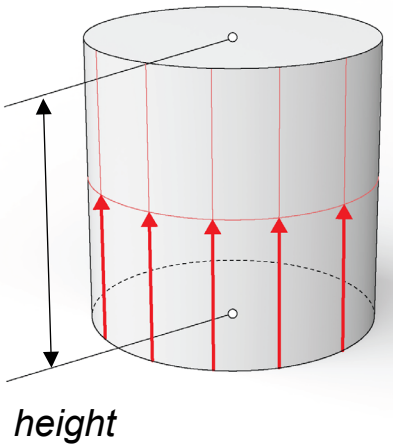
Solid Primitives

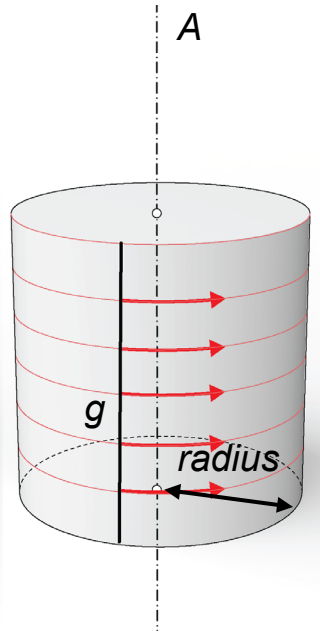
class **Rhino.Geometry**.**Cylinder**(*baseCircle: Circle, height: Float*)

A ***rotational cylinder*** can be generated by extrusion of a circle ***c*** lying in a plane ***P***

or

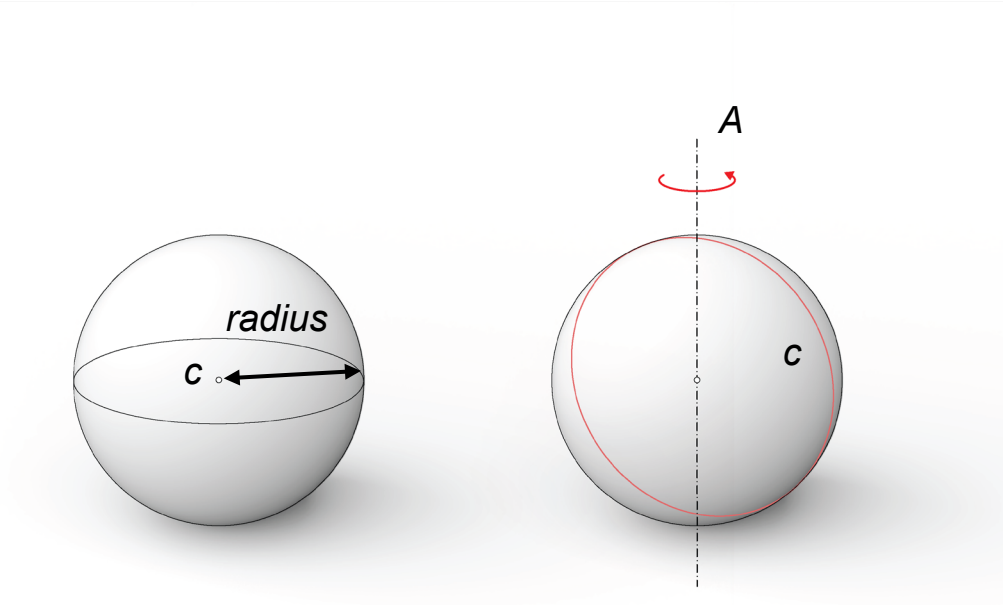
obtained by rotating a straight line ***g*** around an ***axis A*** parallel to ***g***.





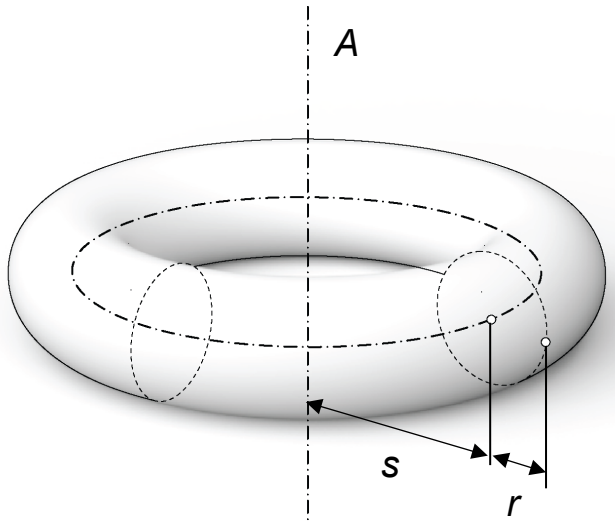
Solid Primitives

class **Rhino.Geometry**.**Sphere**(*center: Point3d, radius: Float*)



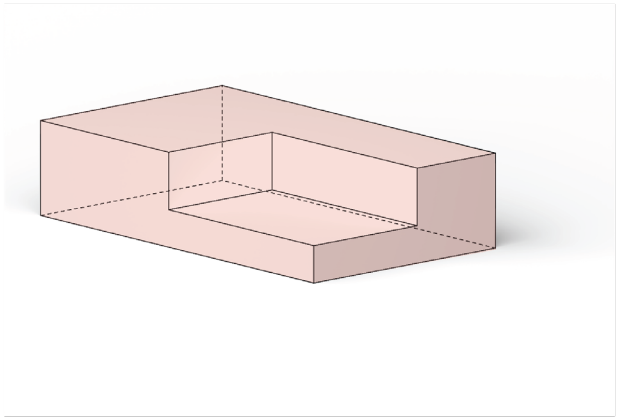
Solid Primitives

class **Rhino.Geometry**.**Torus**(*basePlane: Plane, majorRadius: Float, minorRadius: Float*)

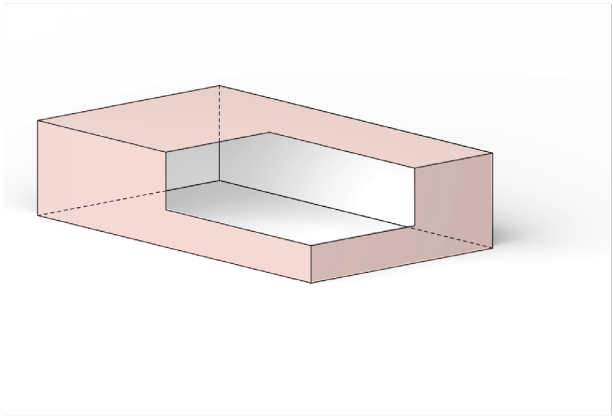


Boundary Representations (BRep)

solid model



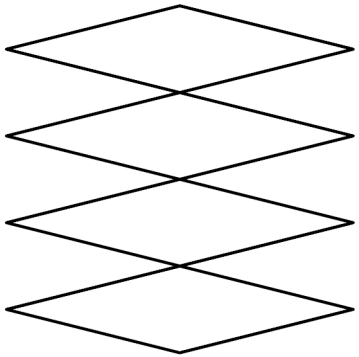
surface model



Boundary Representations (BRep)

class **Rhino.Geometry**.**Brep.CreatePlanerBreps***(crv: Curve, tolerance: Float)*

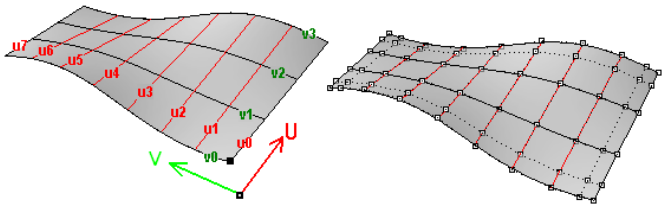
closed curves



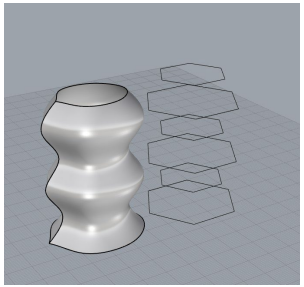
BRrep

Pre-defined Surfaces in Rhino

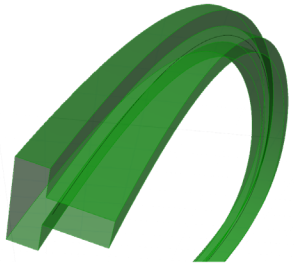
NURBS



Loft

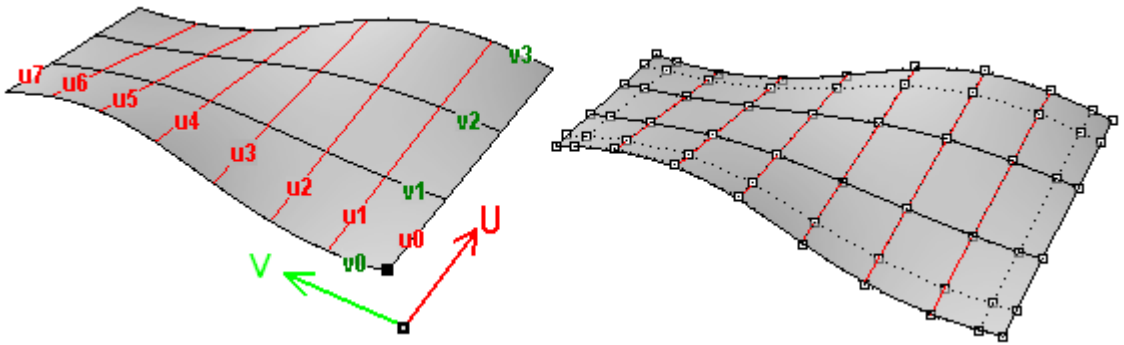


Sweep



NURBS Surfaces

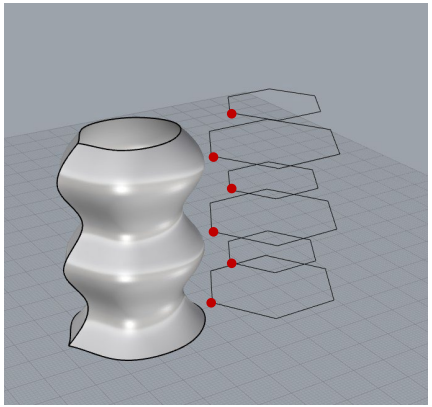
- Bilinear NURBS patch defined by two sets of splines
- Result from a control point grid



class **Rhino.Geometry.NurbsSurface**

Loft Surfaces

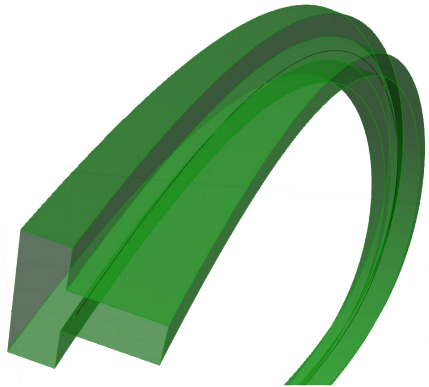
- Surface resulting from mapping corresponding points from a list of curves



class **Rhino.Geometry.Brep.CreateFromLoft**

Sweep Surfaces

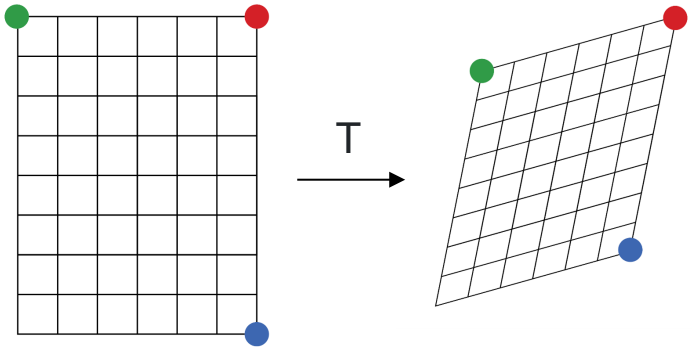
- Surface resulting by moving a profile curve along a trajectory



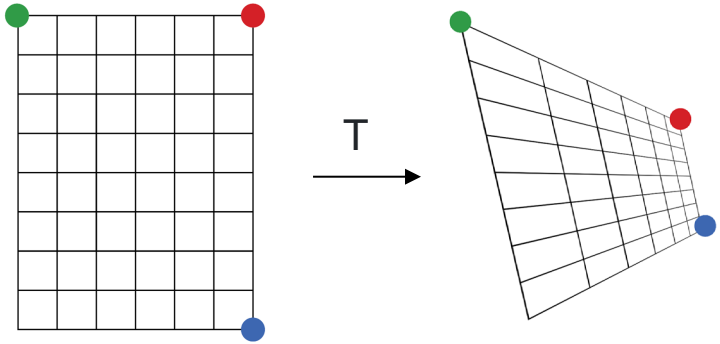
class **Rhino.Geometry.**Brep.CreateFromSweep

Transformations

Transformations

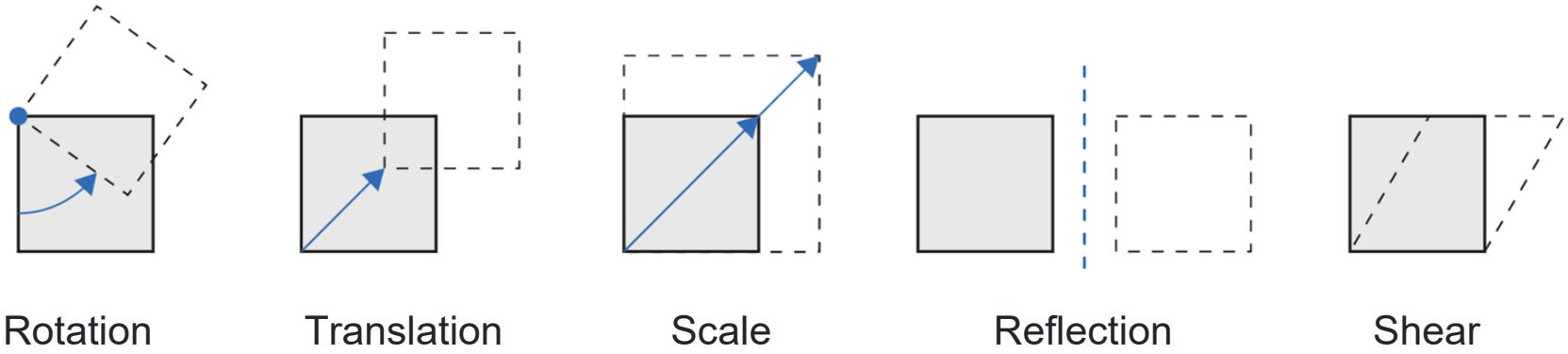


Affine



Non-affine

Affine transformations



5. Transformations

class **Rhino.Geometry.Transform.Rotation***(angleRadians: float, rotationCenter: Point3d)*

class **Rhino.Geometry.Transform.Translation***(motion: Vector3d)*

class **Rhino.Geometry.Transform.Scale***(anchor: Point3d, scaleFactor: float)*

class **Rhino.Geometry.Transform.Mirror***(mirrorPlane: Plane)*

class **Rhino.Geometry.Transform.Shear***(plane: Plane, x: Vector3d, y: Vector3d, z: Vector3d)*

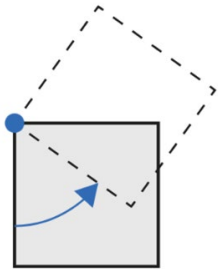
Transformation matrix

$$T = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

Affine transformation matrices

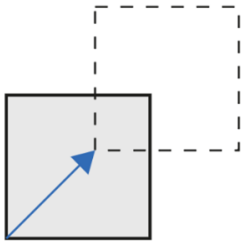
Rotation

$$T = \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 \\ a_{21} & a_{22} & a_{23} & 0 \\ a_{31} & a_{32} & a_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



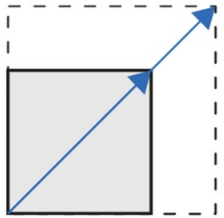
Translation

$$T = \begin{bmatrix} 1 & 0 & 0 & a_{14} \\ 0 & 1 & 0 & a_{24} \\ 0 & 0 & 1 & a_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Scale

$$T = \begin{bmatrix} a_{11} & 0 & 0 & 0 \\ 0 & a_{22} & 0 & 0 \\ 0 & 0 & a_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



But why?

Without Homogeneous Coordinates

Apply nested tranformations:

$$x_2 = A_1 x_1 + b_1$$
$$x_3 = A_2 x_2 + b_2$$

Results in:

$$x_3 = A_2(A_1 x_1 + b_1) + b_2$$
$$x_3 = A_2 A_1 x_1 + A_2 b_1 + b_2$$

→ Need to keep track of all A_1, A_2, b_1, b_2
(frustrated mathematicians)

With Homogeneous Coordinates

Group A, b in A' such that $\mathrm{y}' = A'x'$:

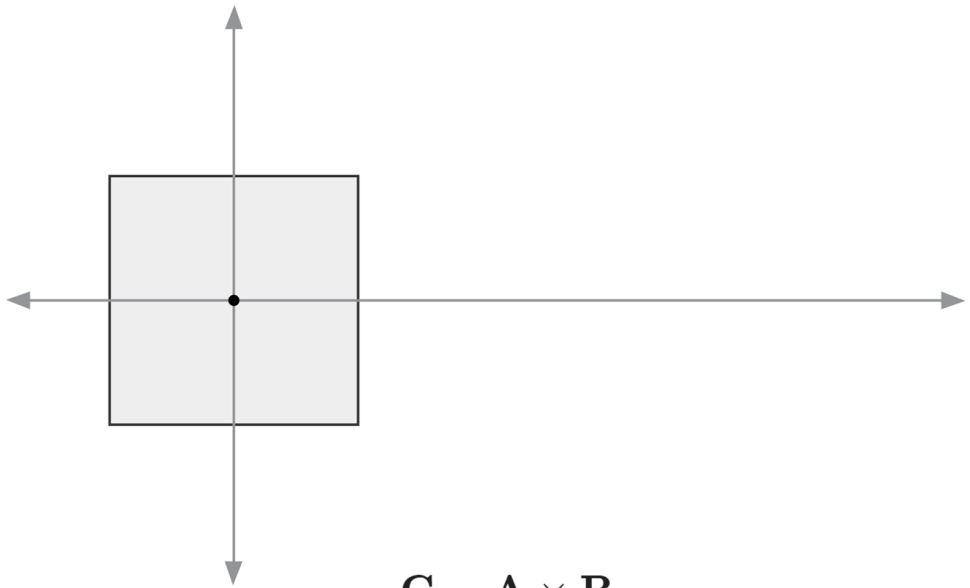
$$\begin{pmatrix} y \\ 1 \end{pmatrix} = \begin{pmatrix} A & b \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ 1 \end{pmatrix} = \begin{pmatrix} Ax + b \\ 1 \end{pmatrix}$$

Now, nested transformations simplify:

$$x_2{}' = A_1{}' x_1{}'$$
$$x_3' = A_2{}' x_2'$$

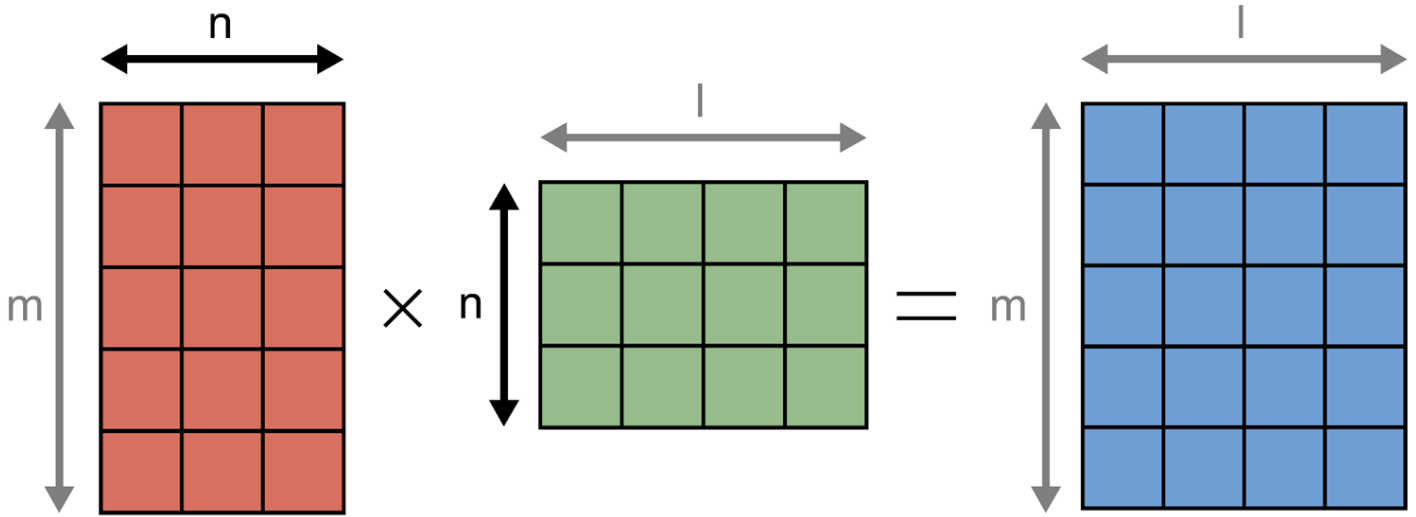
→ $x_3' = \mathrm{A}_1' \mathrm{A}_2' \mathrm{x}_1' = \mathrm{A}' \mathrm{x}_1{}'$
(happy mathematicians)

Composition



$$\mathbf{C} = \mathbf{A} \times \mathbf{B}$$

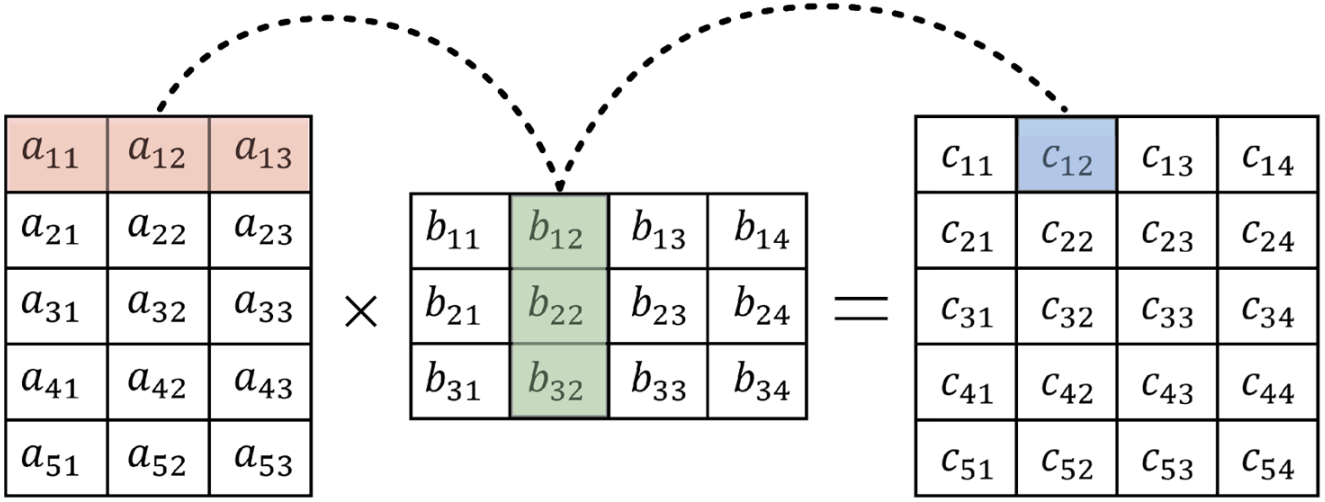
Matrix multiplication



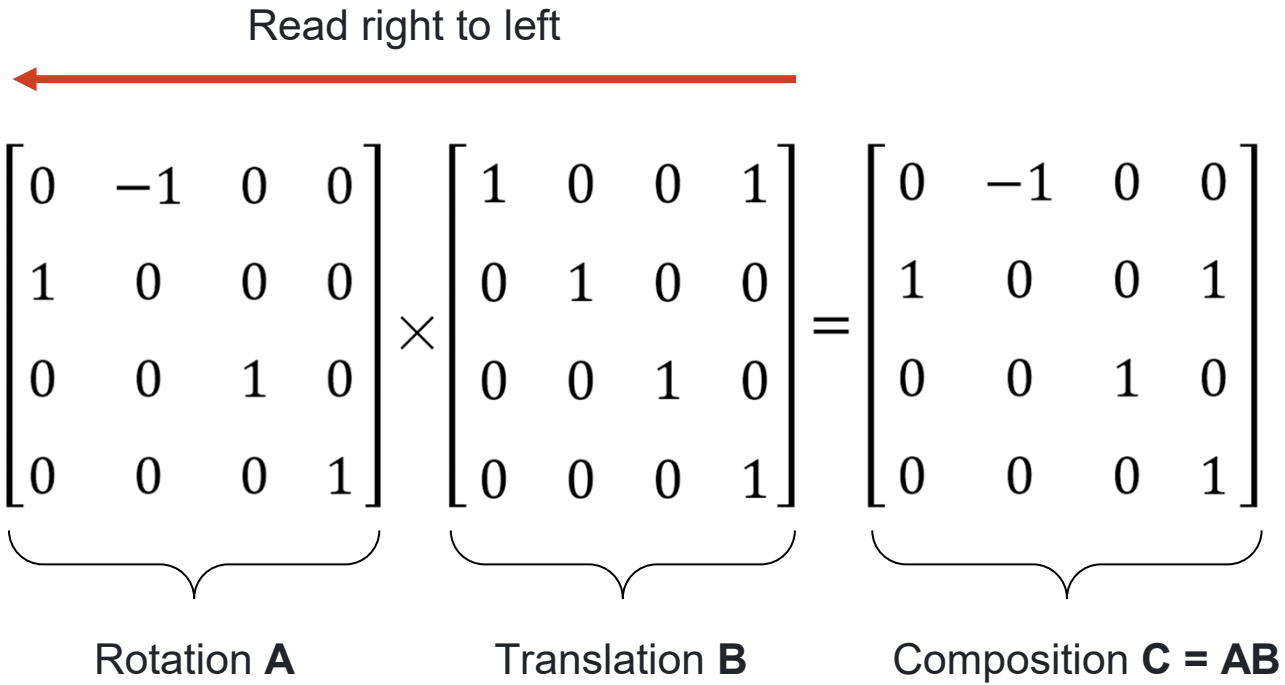
A **B** **C = A x B**

Matrix multiplication

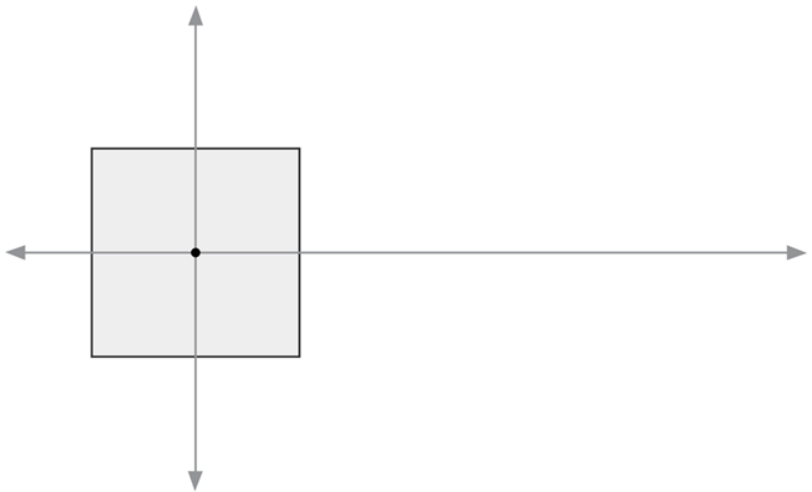
$$c_{12} = a_{11}b_{12} + a_{12}b_{22} + a_{13}b_{32}$$



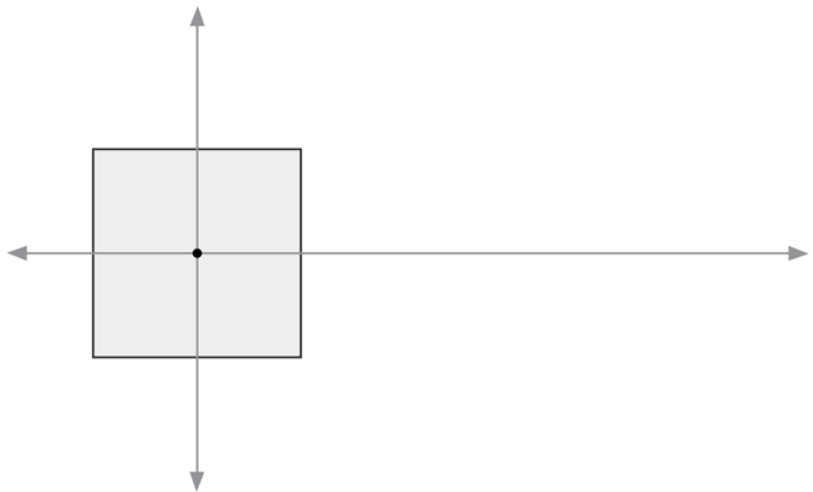
Composition



Noncommutativity $(\mathbf{A} \times \mathbf{B} \neq \mathbf{B} \times \mathbf{A})$



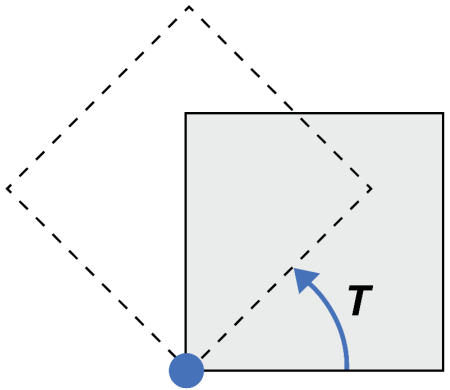
$\mathbf{C} = \mathbf{A} \times \mathbf{B}$ (apply $\mathbf{B}$ to the right of $\mathbf{A}$)



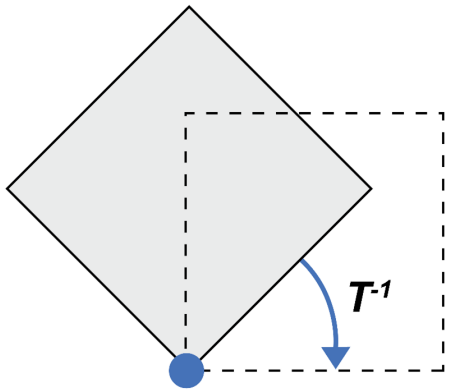
$\mathbf{C} = \mathbf{B} \times \mathbf{A}$ (apply $\mathbf{B}$ to the left of $\mathbf{A}$)

Inverse transformation

$$T \times T^{-1} = I \longrightarrow I_1 = [1],\ I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},\ I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},\ \ldots,\ I_n = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}$$



T



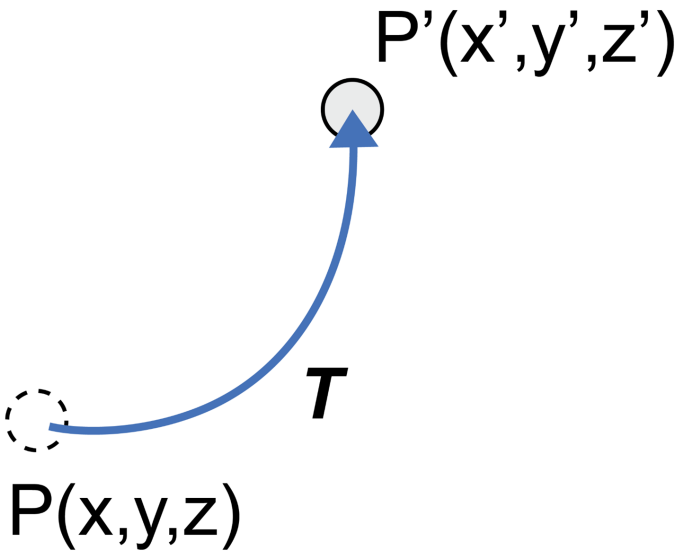
T^{-1}

Point/Vector Transformation

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} a_{11} \cdot x + a_{12} \cdot y + a_{13} \cdot z + a_{14} \cdot w \\ a_{21} \cdot x + a_{22} \cdot y + a_{23} \cdot z + a_{24} \cdot w \\ a_{31} \cdot x + a_{32} \cdot y + a_{33} \cdot z + a_{34} \cdot w \\ a_{41} \cdot x + a_{42} \cdot y + a_{43} \cdot z + a_{44} \cdot w \end{bmatrix}$$

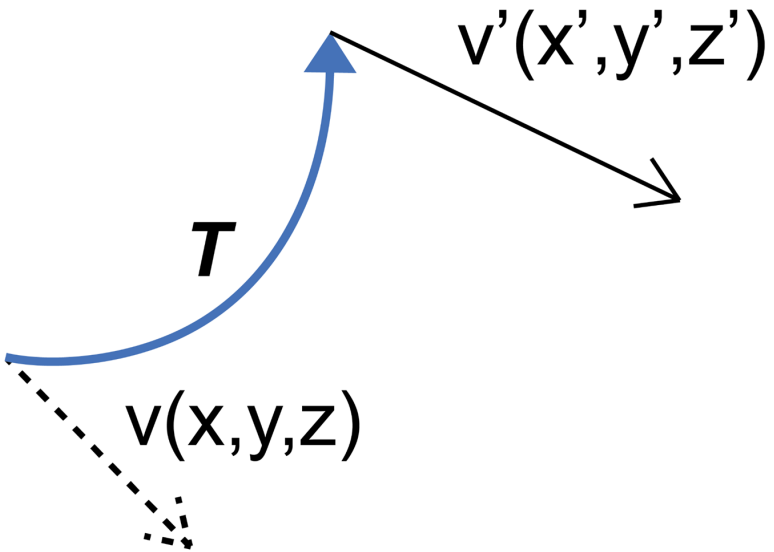
Transformation matrix Point / Vector Transformed Point / Vector

Point homogenisation



$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix}$$

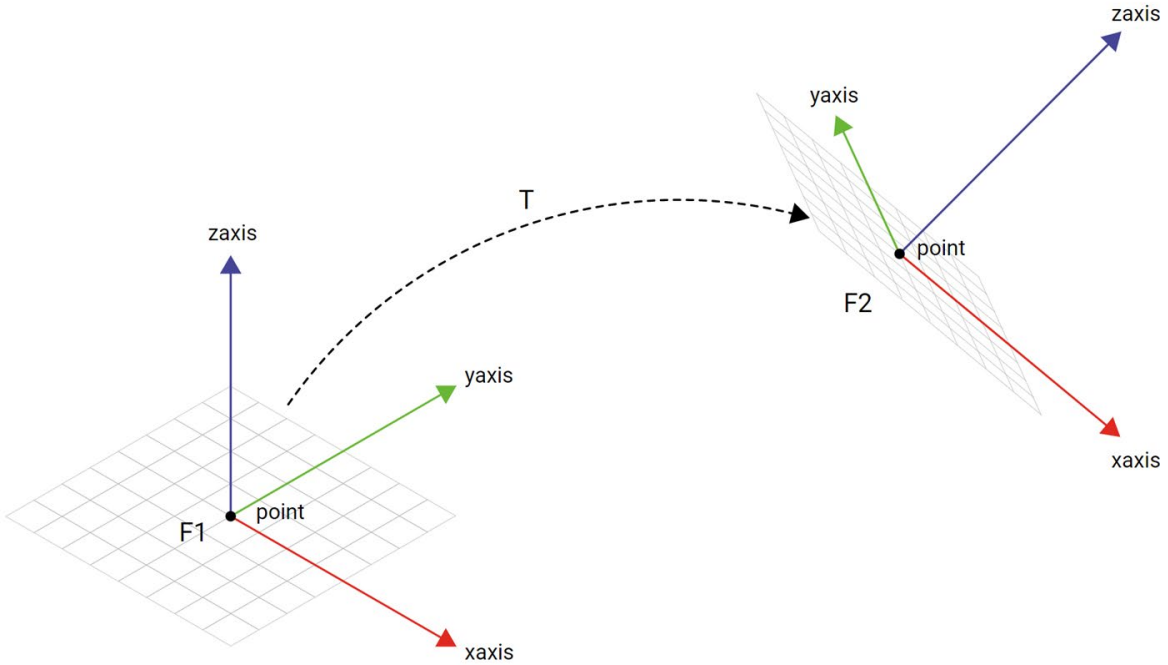
Vector homogenisation



$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \\ 0 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \\ 0 \end{bmatrix}$$

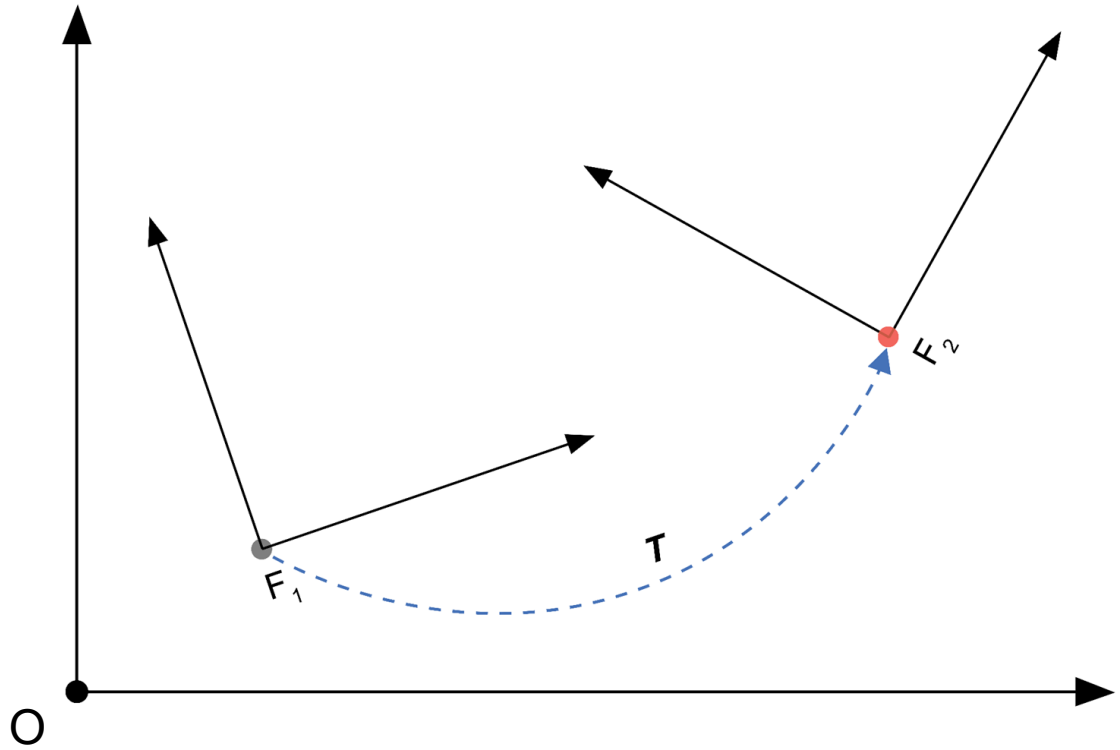
Transformation methods

- Rhino.Geometry.Transform.**PlaneToPlane***(fromPlane, toPlane)*
- Rhino.Geometry.Transform.**ChangeBasis***(fromPlane, toPlane)*



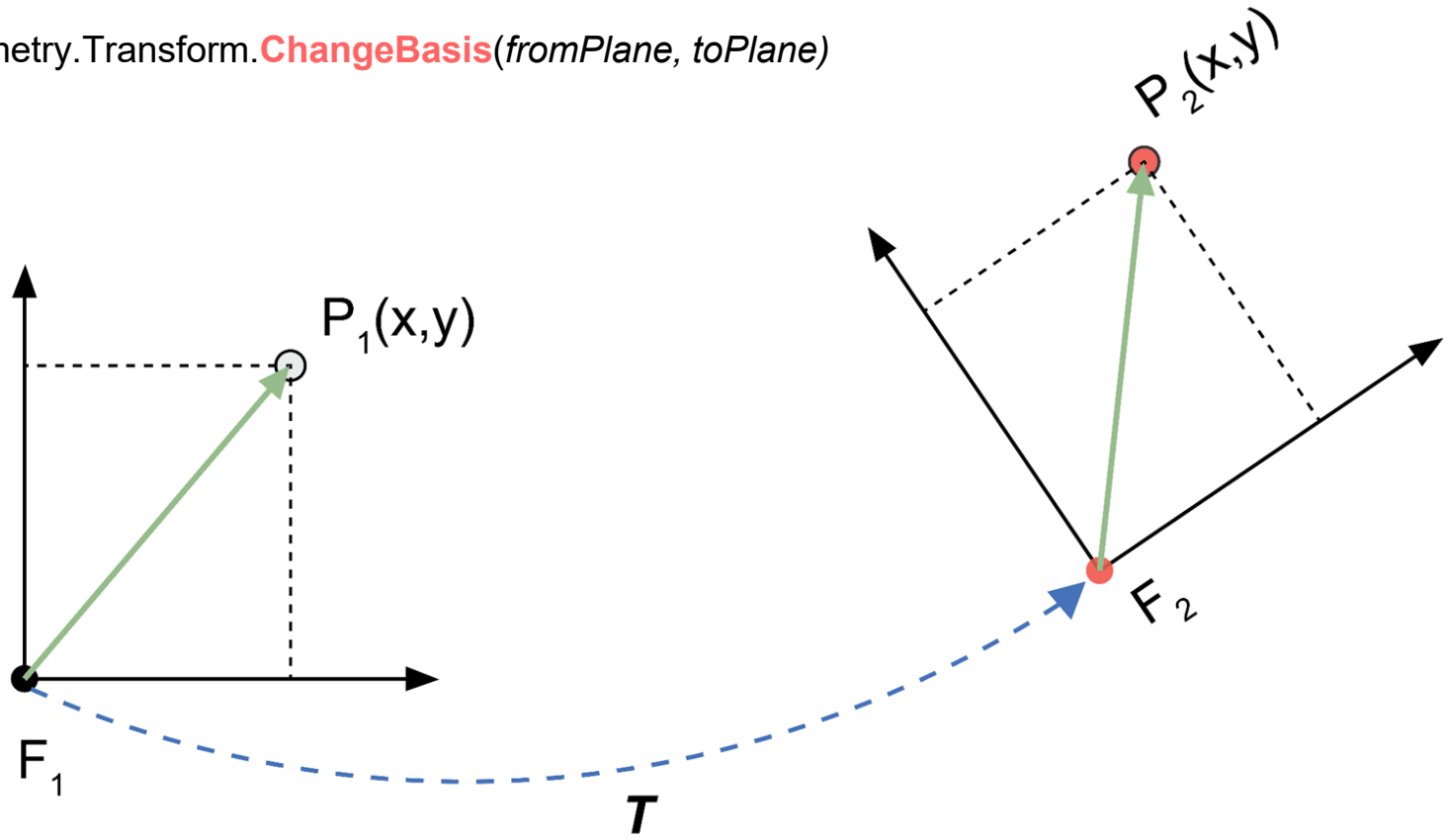
Transformation between two planes

Rhino.Geometry.Transform.**PlaneToPlane***(fromPlane, toPlane)*



ChangeBasis transformation

Rhino.Geometry.Transform.**ChangeBasis**(*fromPlane, toPlane)*

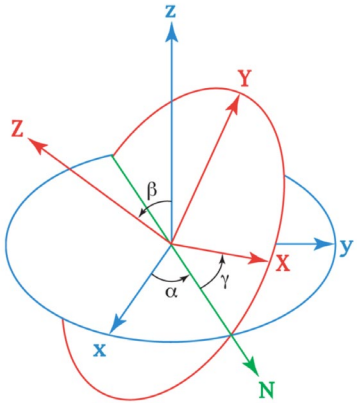


Rotation representation

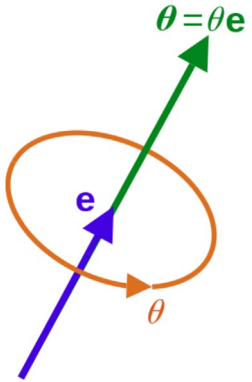
Rhino.Geometry.Transform.**Rotation**

$$T = \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 \\ a_{21} & a_{22} & a_{23} & 0 \\ a_{31} & a_{32} & a_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

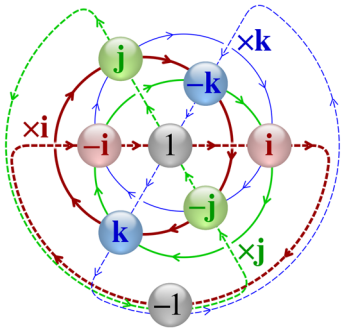
Rotation matrix



Euler angles



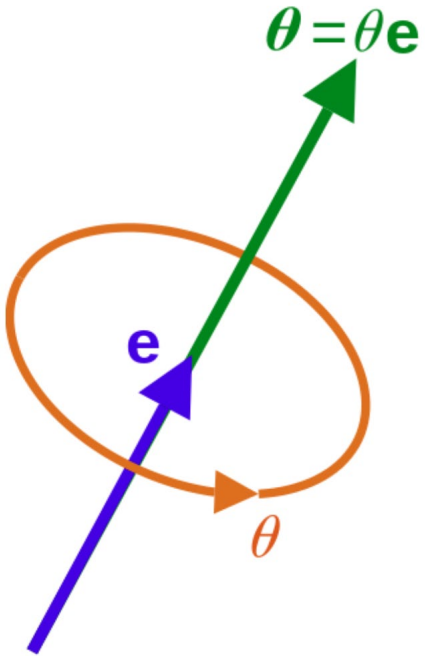
Axis-angle



Quaternions

Axis-angle representation

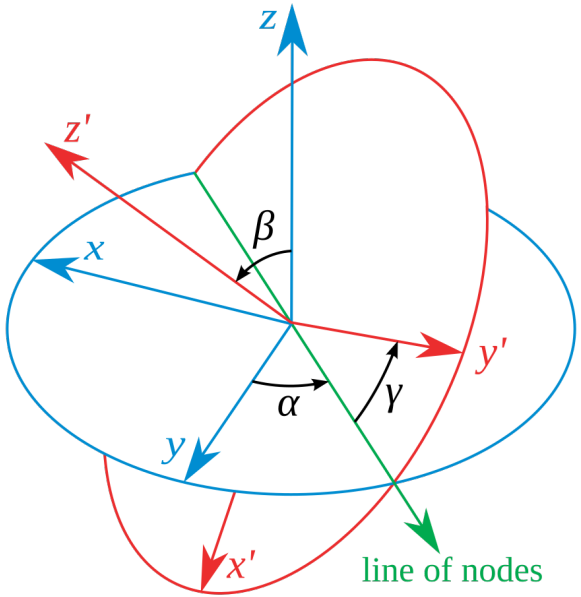
Rhino.Geometry.Transform.**Rotation***(angleRadians, rotationAxis, rotationCenter)*

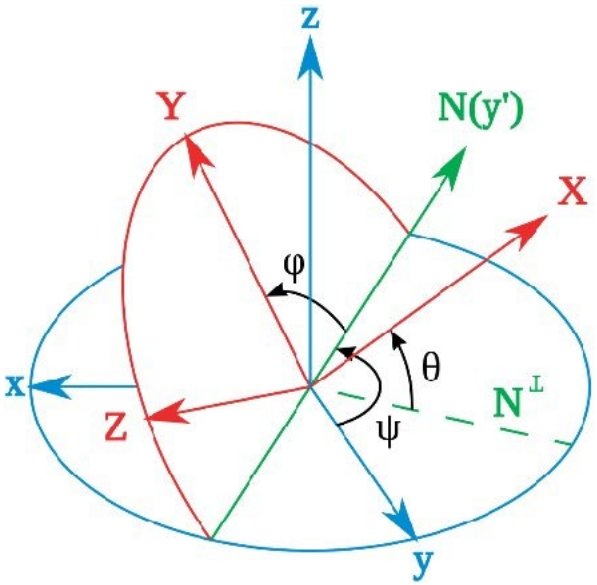


Euler angles

Rhino.Geometry.Transform.**RotationZYZ***(alpha, beta, gamma)*

Rhino.Geometry.Transform.**RotationZYX***(rot z-axis (yaw), rot y-axis (pitch), rot z-axis (roll))*





Quaternions

q = Rhino.Geometry.**Quaternion***(a, b, c, d)*

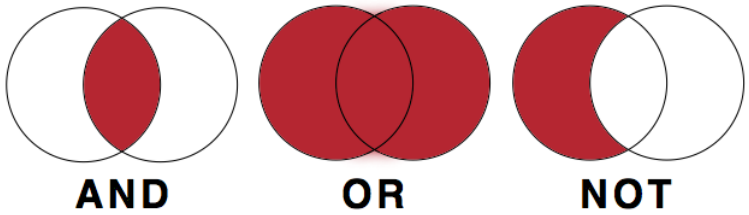
$$q = a + b\,\mathbf{i} + c\,\mathbf{j} + d\,\mathbf{k}$$

Boolean operations

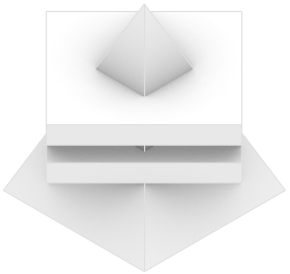
5. Boolean operations

In solid objects, a point may only be inside or outside (Boolean state)

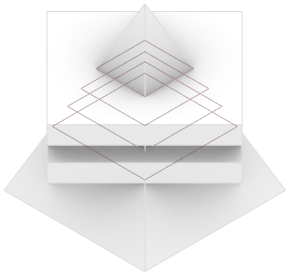
Boolean operators generalize from logic definitions to tools for manipulating solid objects



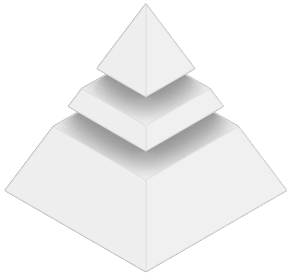
5. Boolean operations



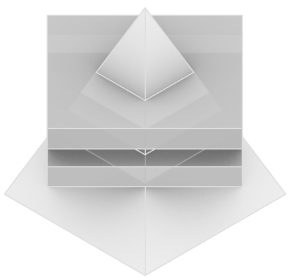
union



intersection



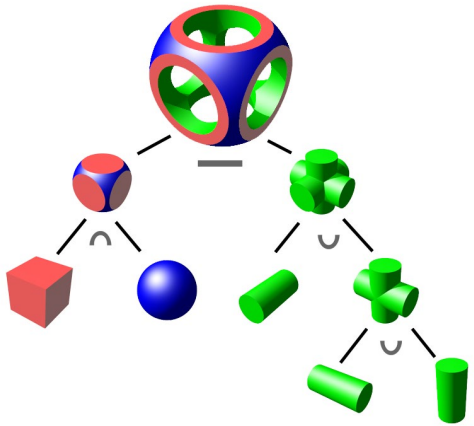
difference



split

5. Boolean operations

Boolean operations can be composed to get increasingly complex shapes:



∪ : Union
∩ : Intersection
– : Difference

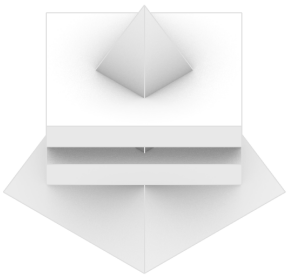
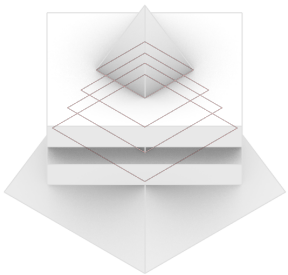
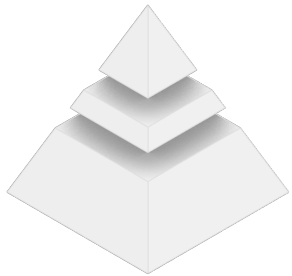
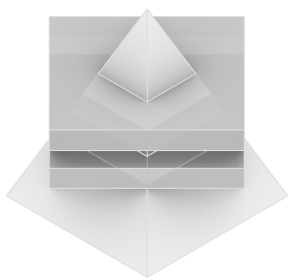
5. Boolean operations

class **Rhino.Geometry.Brep.CreateBooleanUnion***(breps: [Breps], tolerance)*

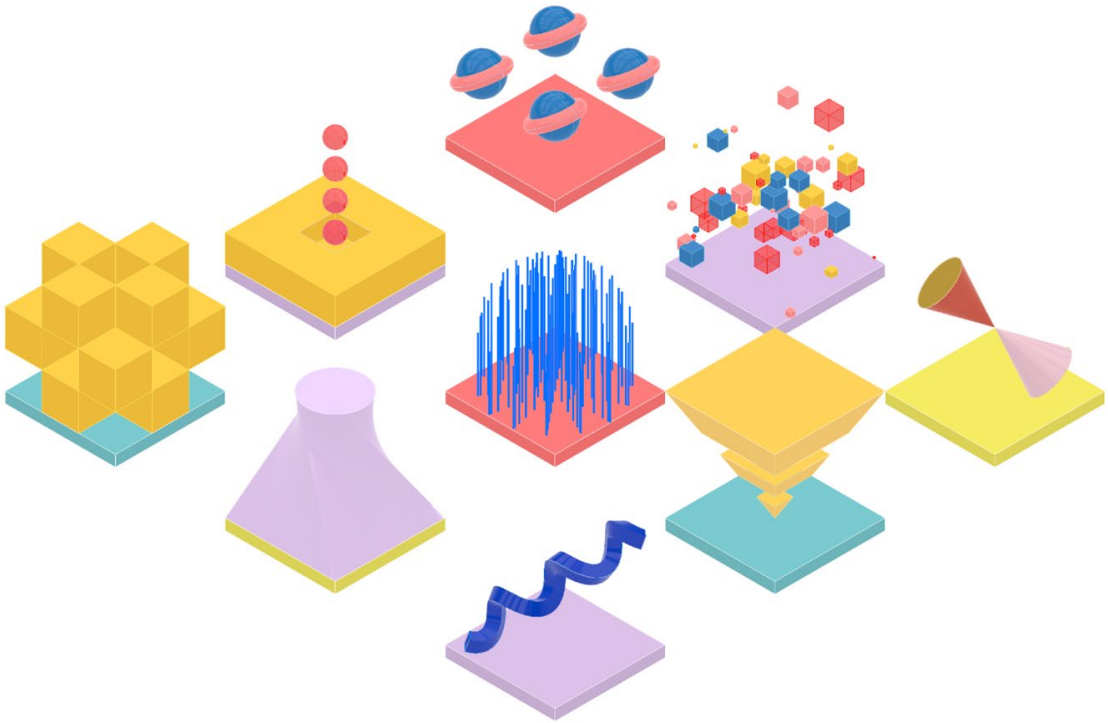
class **Rhino.Geometry.Brep.CreateBooleanIntersection***(brep1: Brep, brep2: Brep, tolerance)*

class **Rhino.Geometry.Brep.CreateBooleanDifference***(brep1: Brep, brep2: Brep, tolerance)*

class **Rhino.Geometry.Brep.CreateBooleanSplit***(brep1: Brep, brep2: Brep, tolerance)*

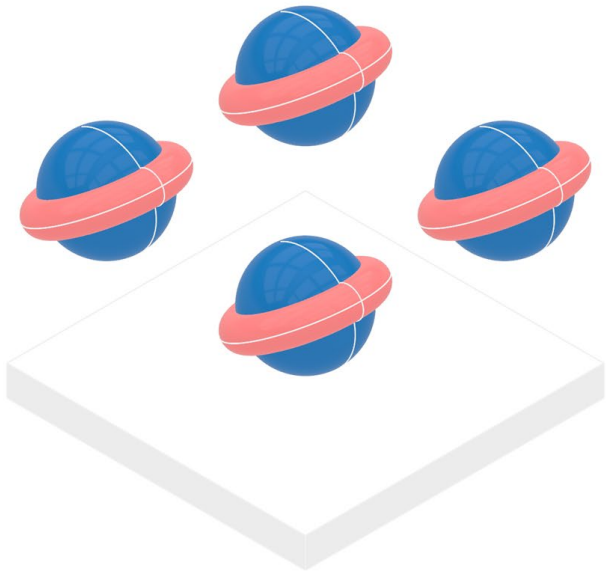
Examples



1. Spheres and toruses

Learning objectives

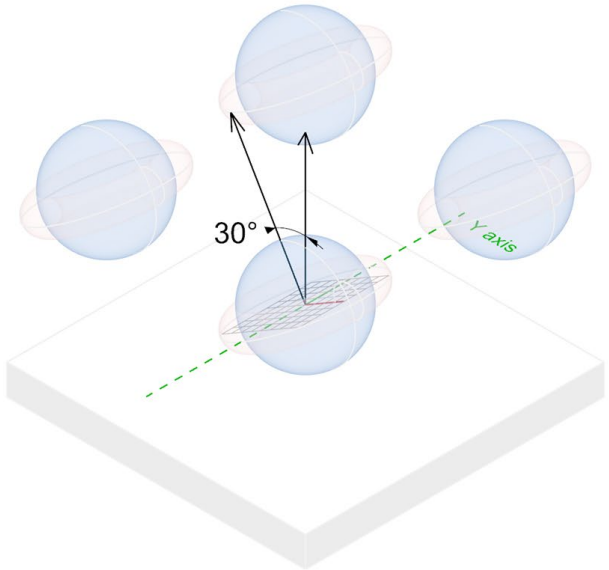
1. Constructing **spheres** and **toruses.**
2. **Rotating** a plane based on an **axis** and an **angle**.



1. Spheres and toruses

Learning objectives

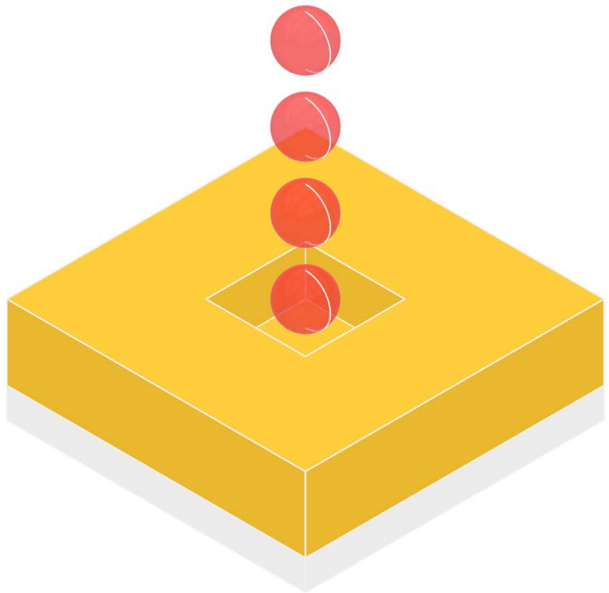
1. Constructing **spheres** and **toruses.**
2. **Rotating** a plane based on an **axis** and an **angle**.



2. Box and spheres

Learning objectives

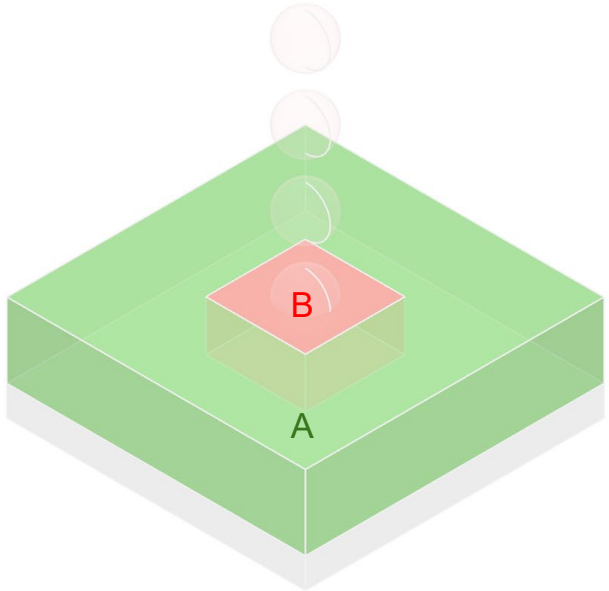
1. Boolean difference.
2. **Translation** based on vectors.



2. Box and spheres

Learning objectives

1. Boolean difference.
2. **Translation** based on vectors.

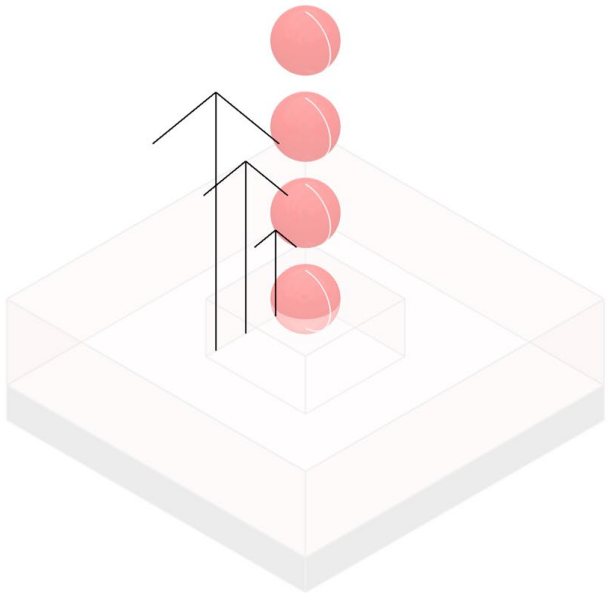


Box A - Box B

2. Box and spheres

Learning objectives

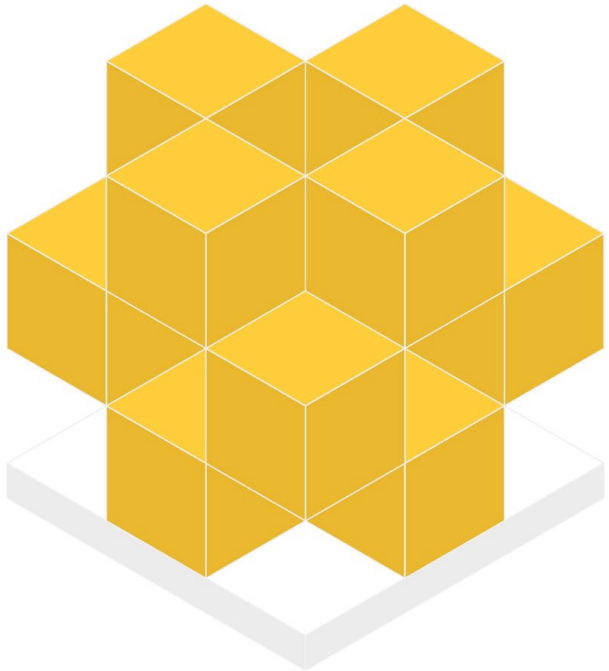
1. Boolean difference.
2. **Translation** based on vectors.



3. Chequered boxes

Learning objectives

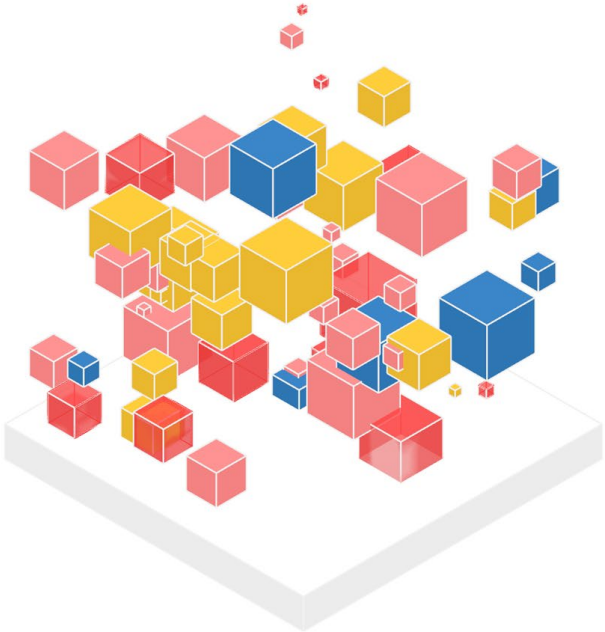
1. Extending Chequered squares in 2D to 3D.
2. Recap: Using if conditions within for loops



4. Point cloud

Learning objectives

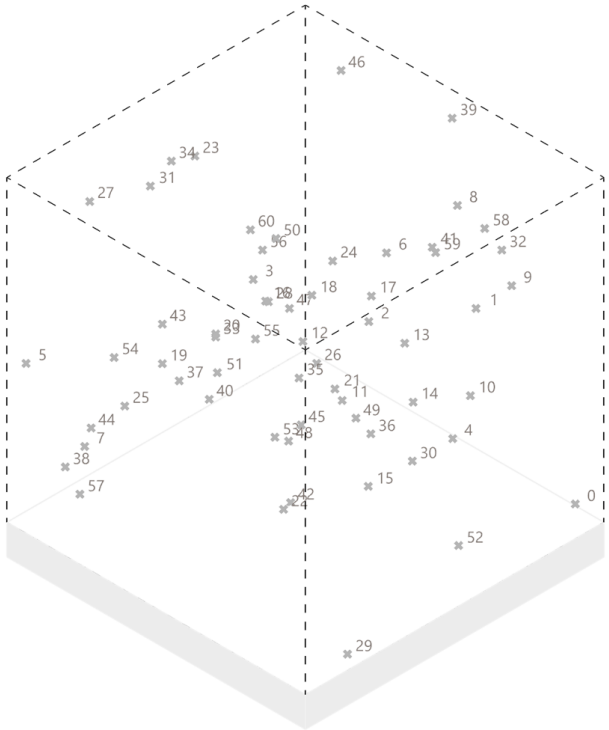
1. Creating random points within a domain.
2. Understanding scale transformation.
3. Creating random colors using script.



4. Point cloud

Learning objectives

1. Creating random points within a domain.
2. Understanding scale transformation.
3. Creating random colors using script.

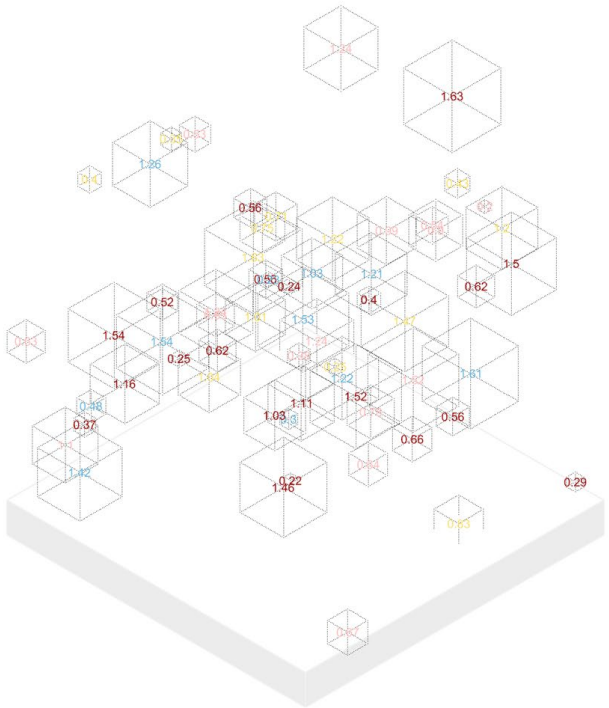


Points

4. Point cloud

Learning objectives

1. Creating random points within a domain.
2. Understanding scale transformation.
3. Creating random colors using script.

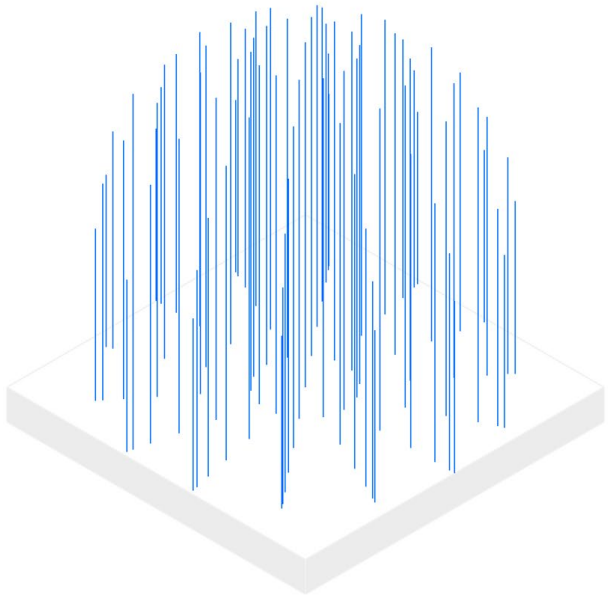


Scales and colors

5. Point traces

Learning objectives

1. **Splitting a sphere** to get the hemisphere
2. Getting **points on a surface** using **u, v** value pairs
3. **Projecting points** on a brep



5. Point traces

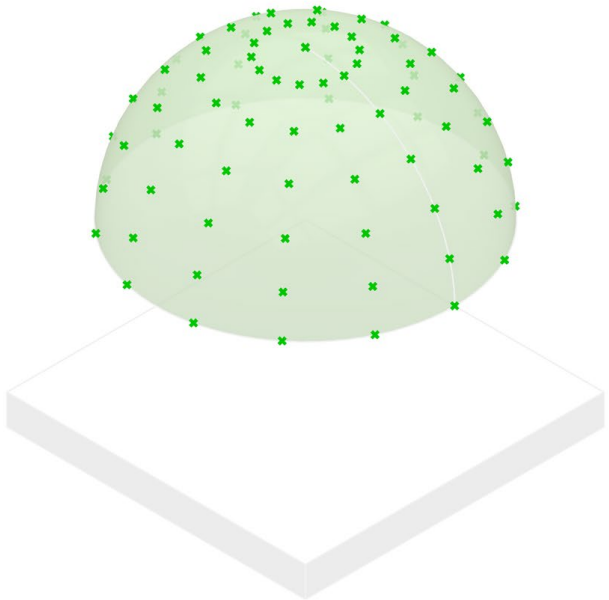
Learning objectives

1. **Splitting a sphere** to get the hemisphere
2. Getting **points on a surface** using **u, v** value pairs
3. **Projecting points** on a brep

5. Point traces

Learning objectives

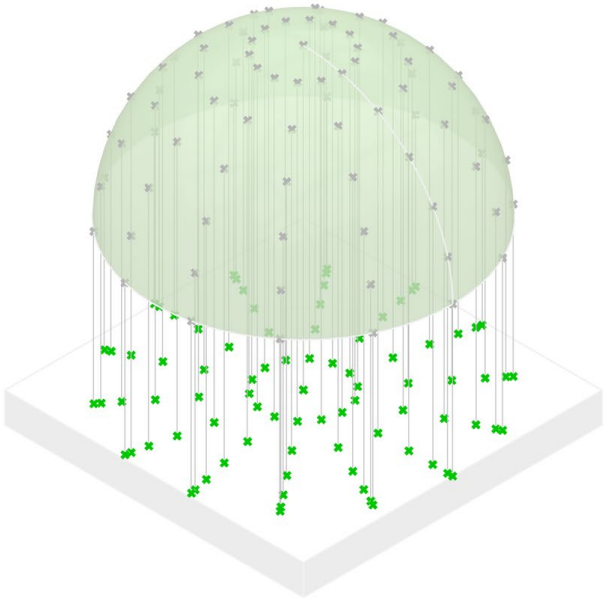
1. **Splitting a sphere** to get the hemisphere
2. Getting **points on a surface** using **u, v** value pairs
3. **Projecting points** on a brep



5. Point traces

Learning objectives

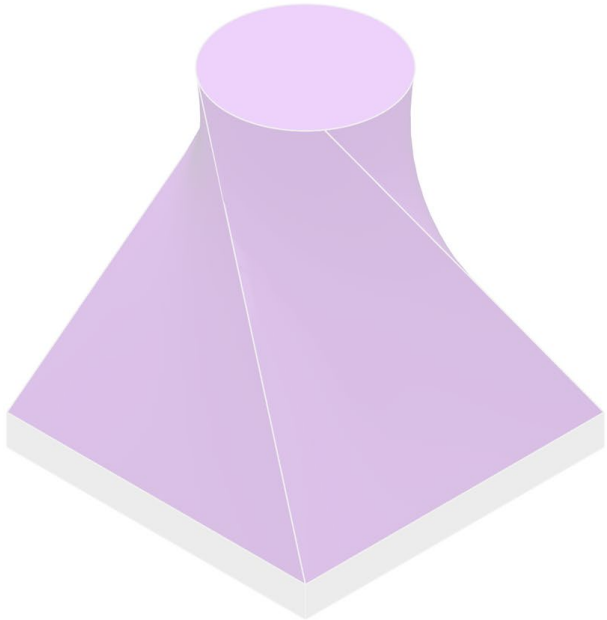
1. **Splitting a sphere** to get the hemisphere
2. Getting **points on a surface** using **u, v** value pairs
3. **Projecting points** on a brep



6. Blend surface

Learning objectives

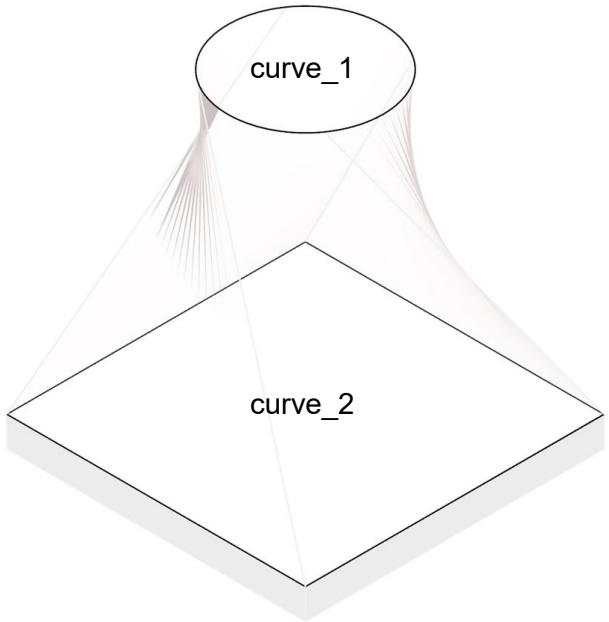
Creating a solid by blending
2 curves



6. Blend surface

Learning objectives

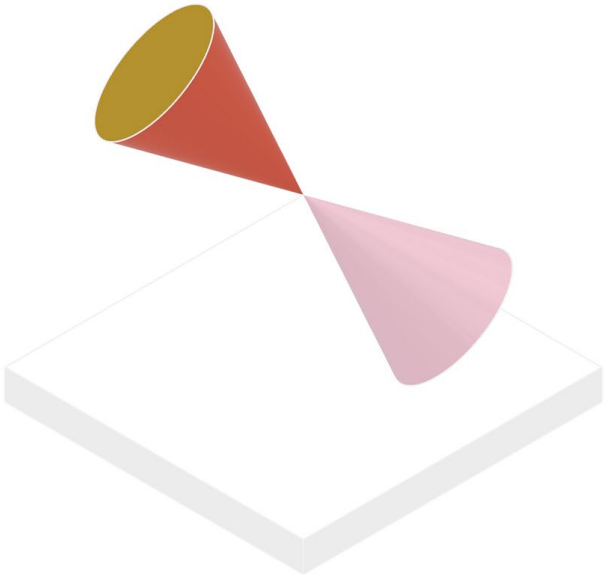
Creating a solid by blending
2 curves



7. Mirrored cones

Learning objectives

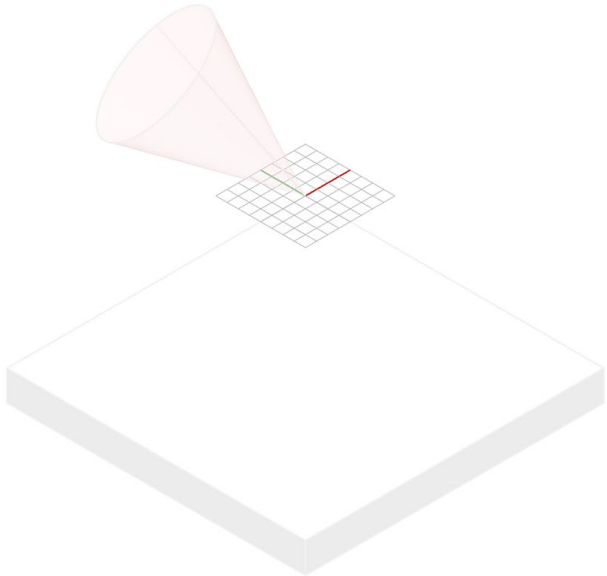
1. Constructing **cones**.
2. Using **Mirror transformation.**



7. Mirrored cones

Learning objectives

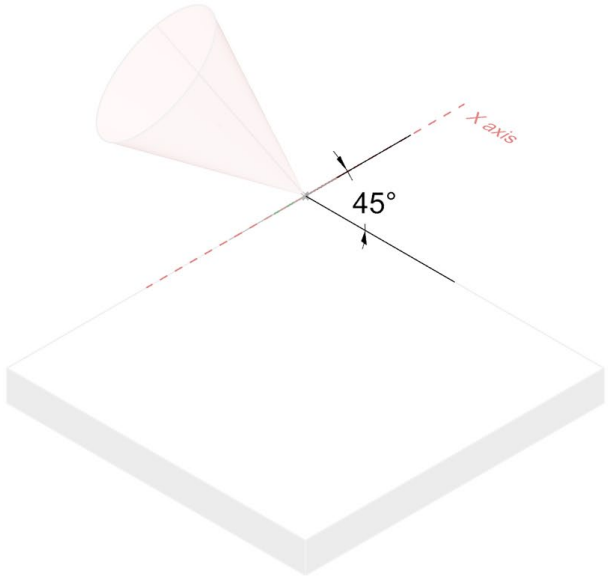
1. Constructing **cones**.
2. Using **Mirror transformation.**



7. Mirrored cones

Learning objectives

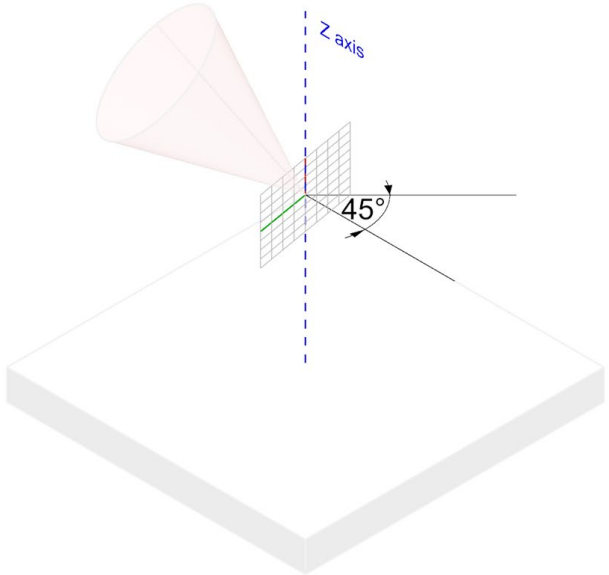
1. Constructing **cones**.
2. Using **Mirror transformation.**



7. Mirrored cones

Learning objectives

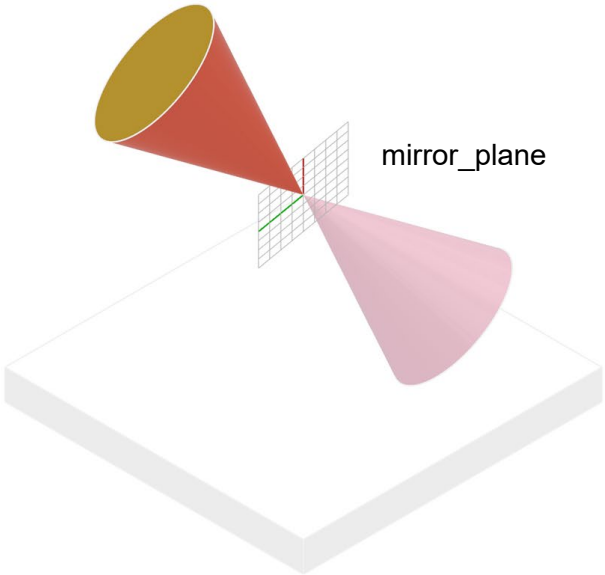
1. Constructing **cones**.
2. Using **Mirror transformation.**



7. Mirrored cones

Learning objectives

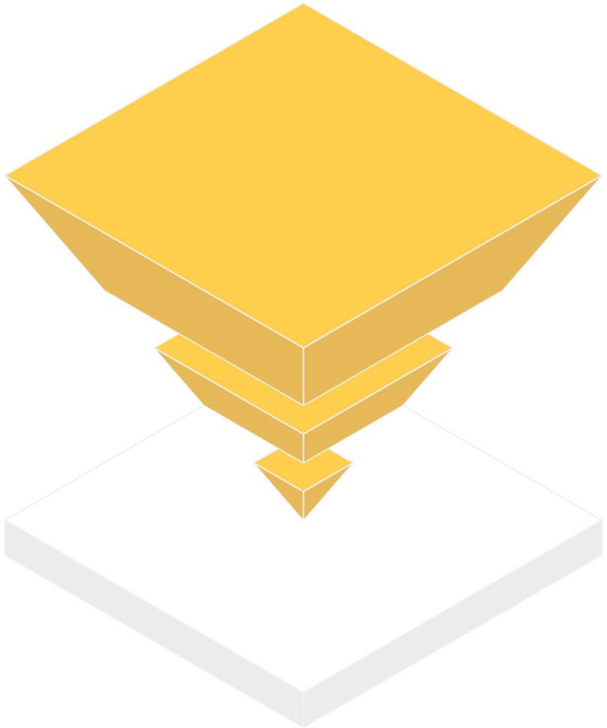
1. Constructing **cones**.
2. Using **Mirror transformation.**



8. Pyramid parts

Learning objectives

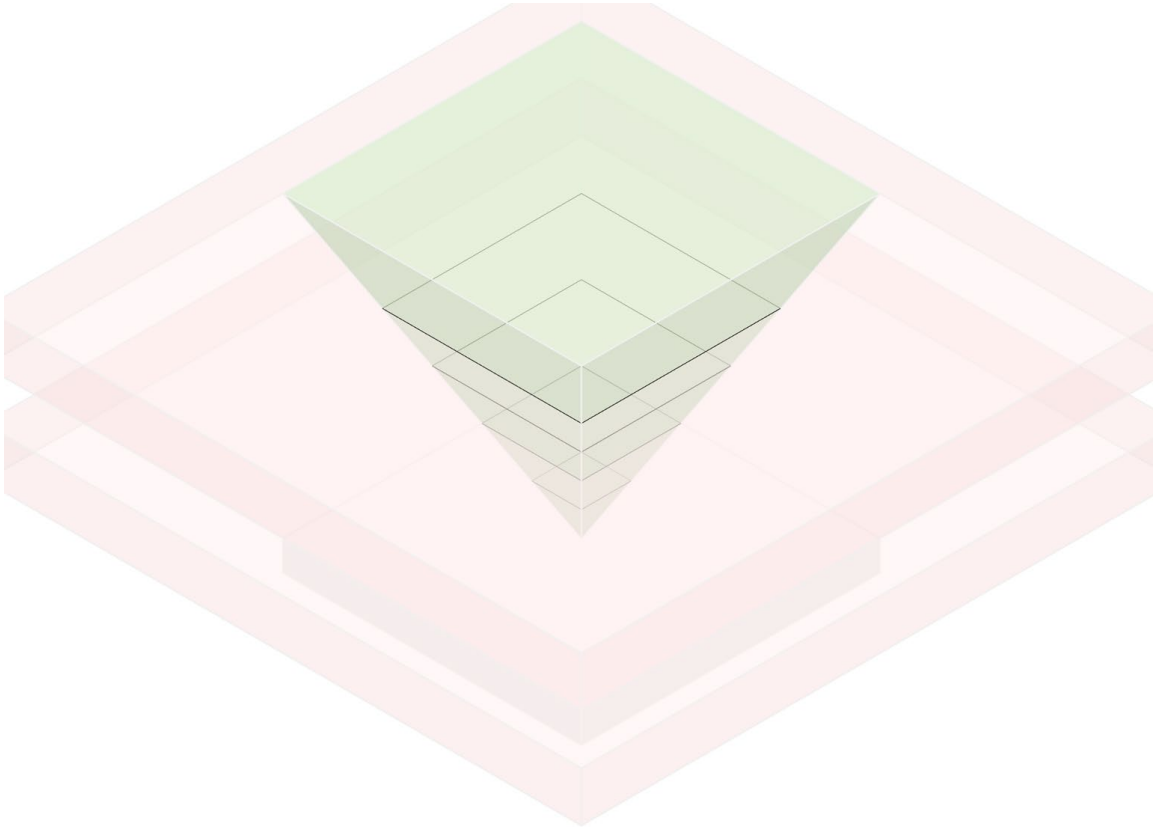
Recap: Boolean difference



8. Pyramid parts

Learning objectives

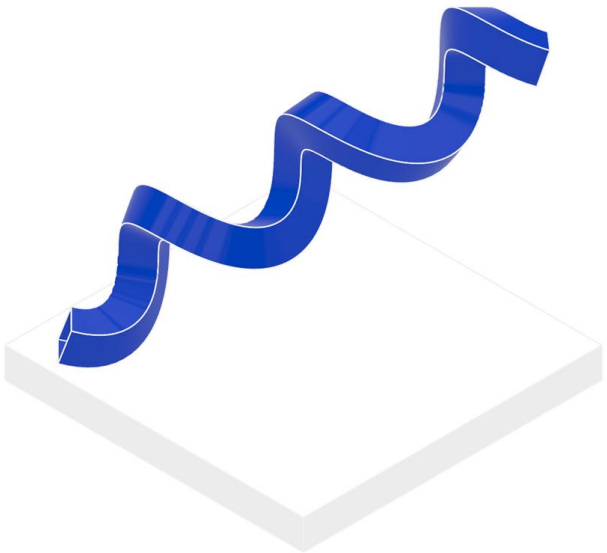
Recap: Boolean difference



9. Sweep

Learning objectives

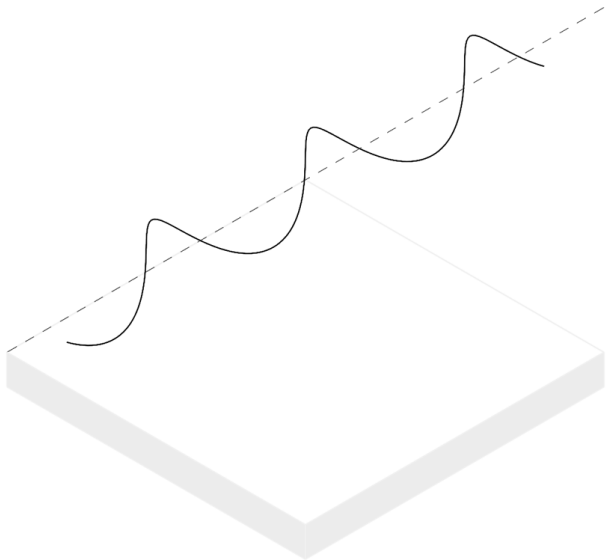
1. Construct a spiral
2. Construct a sweep



9. Sweep

Learning objectives

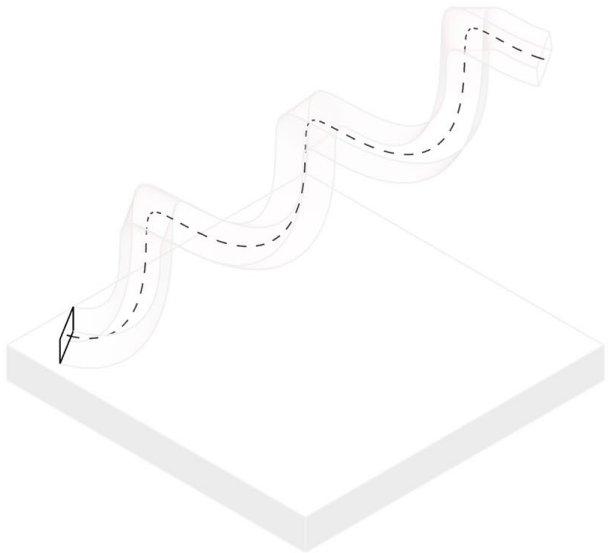
1. Construct a spiral
2. Construct a sweep



9. Sweep

Learning objectives

1. Construct a spiral
2. Construct a sweep



Open up your Grasshopper/Rhino environment and try out Rhino geometries!

Look into a RhinoCommon API for further geometries and functions.

Enjoy creating!