Tekla Open API

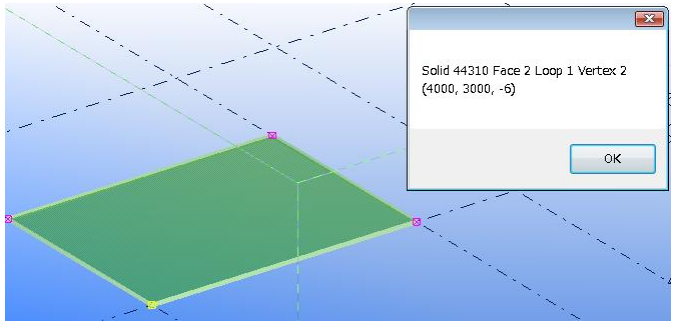
3D Geometry & Solids

Overview

- The API provides tools for working in different work planes or coordinate systems through the work plane handler and matrix transformations.
- The API supplies basic 3D geometric objects and methods for performing 3D calculations and operations.
- The API provides access to the full part solid for use in complex calculations and data extraction. It is also possible to get and work with the cuts and fittings (boolean operations) applied to a part.

Skills to learn

- How to use the work plane and matrixes
  - Work Plane Handler
  - Transformation Planes
  - Matrix Factory
- Content and overview of Geometry3D namespace
  - Geometric objects
  - Calculations and constructions
- Working with solids
  - GetSolid() and solid types
  - Enumerate the faces, loops, and vertices
  - Available solid properties and methods

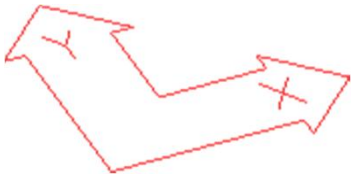


Work Planes & Matrixes

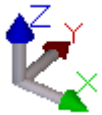
Manipulating the work plane
Effects of changing the work plane
Matrixes for translating from one work plane to another

Work planes and Matrixes

- Work plane
  - The plane that the user has chosen in a Tekla Structures model and that is currently active when working in a model view
  - The current ‘active’ coordinate system
  - Can get and set using the WorkplaneHandler

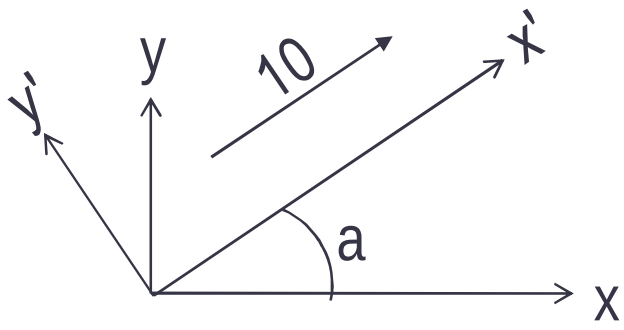


- Matrixes
  - Use to transform points from one coordinate system to another
  - Can be obtained from a TransformationPlane
  - Can be created with the MatrixFactory



Work Planes

- WorkPlaneHandler
  - use to get or set the TransformationPlane which defines the work plane
- TransformationPlane (work plane)
  - defines different coordinate systems in the model
  - provides matrixes for translating to and from global and local coordinates
- Advantages
  - changing the work plane can make it easy to work in sloped or skewed conditions

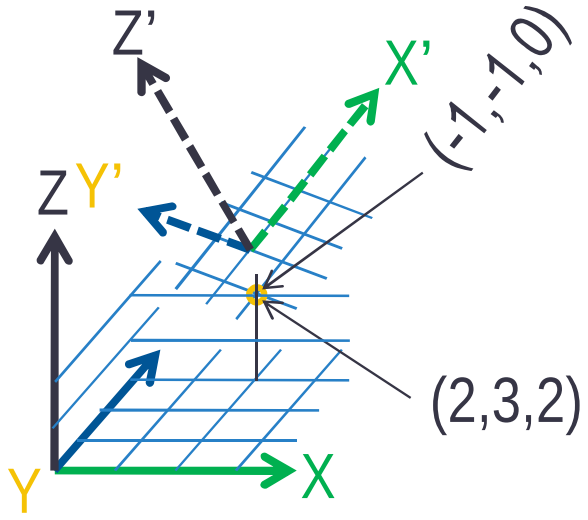


To move 10 in the x' direction move 10*cos(a) in x and 10*sin(a) in y OR change the work plane and move 10 units in x'

Work Planes

- Changing the work plane
  - Object properties (start point, position, etc.) are initialized according to the current work plane
  - If the work plane is changed, object properties can be refreshed by calling the Select() method on the object

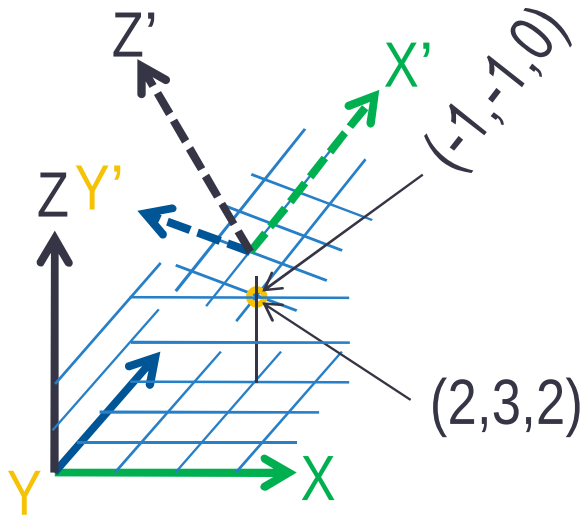
```
workplaneHandler.SetCurrentTransformationPlane(newWorkPlane);
part.Select();
```



Work Planes

- Changing the work plane
  - The global work plane can always be restored by setting it to a new transformation plane

    ```
    SetCurrentTransformationPlane(new TransformationPlane());
    ```
  - Points, vectors, etc. are not translated if the work plane changes, and cannot be updated using Select()
  - Matrixes can be used to translate points, vectors, etc. between different coordinate systems or the current coordinate system and the global coordinate system without needing to change the work plane



Work planes & Matrixes

- A TransformationPlane (work plane) provides methods to create matrixes

- TransformationMatrixToLocal()
  - Provides a matrix to translate from the global work plane to the current work plane

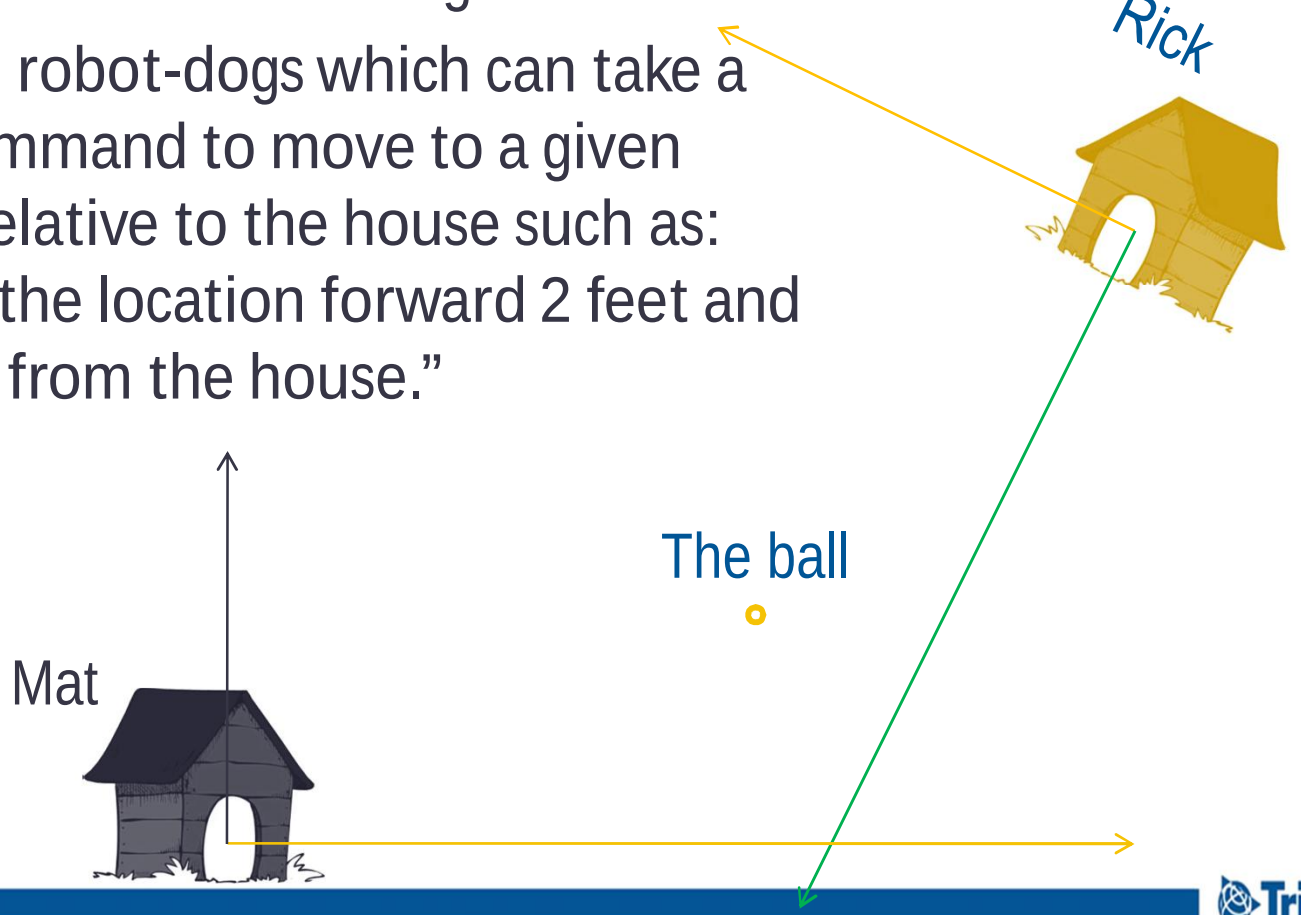
- TransformationMatrixToGlobal()
  - Provides a matrix to translate from the current work plane to the global work plane

Matrixes

- The MatrixFactory class provides methods to create various matrixes
  - FromCoordinateSystem()
    - For translating from a coordinate system to the current work plane
  - ToCoordinateSystem()
    - For translating from the current work plane to a coordinate system
  - ByCoordinateSystems()
    - For translating directly between two coordinate systems
  - Rotation()
    - For rotating around a given axis by an angle A

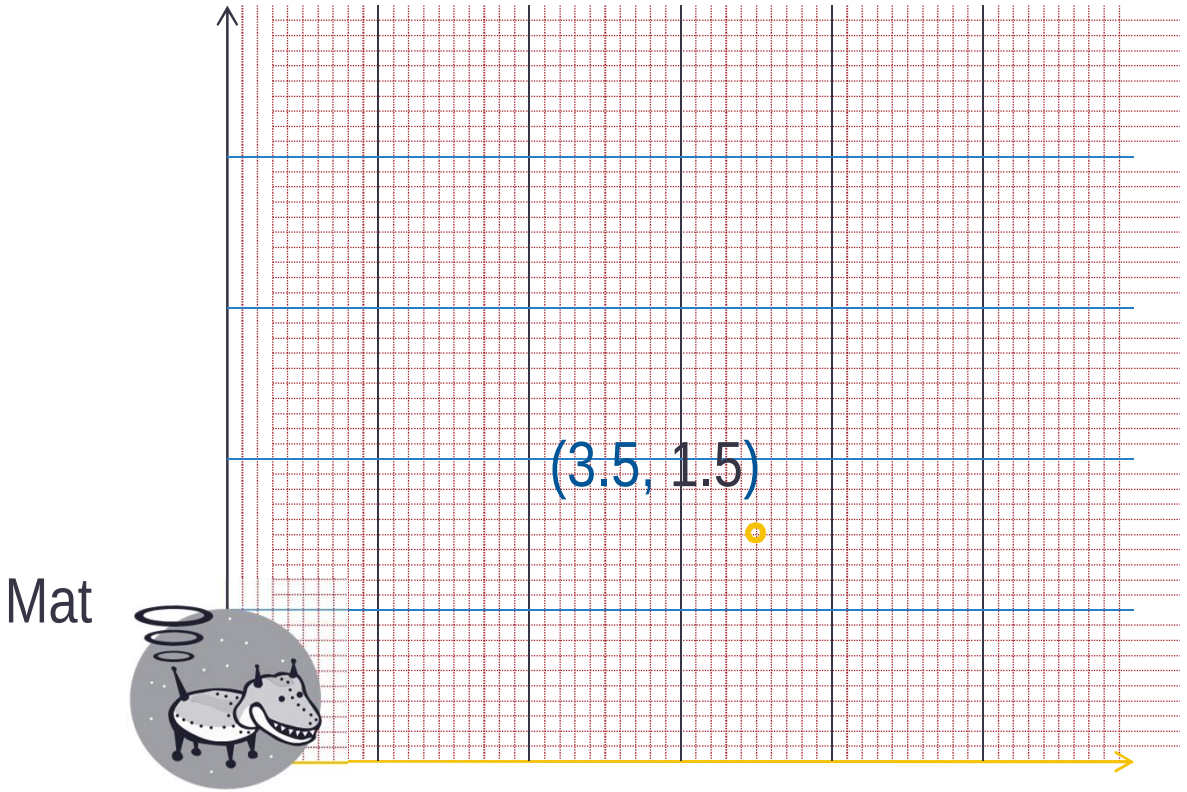
Matrixes – A simple example

- Mat and Rick share a common back lawn but their houses are not aligned.
- Both have robot-dogs which can take a simple command to move to a given location relative to the house such as: “move to the location forward 2 feet and left 3 feet from the house.”



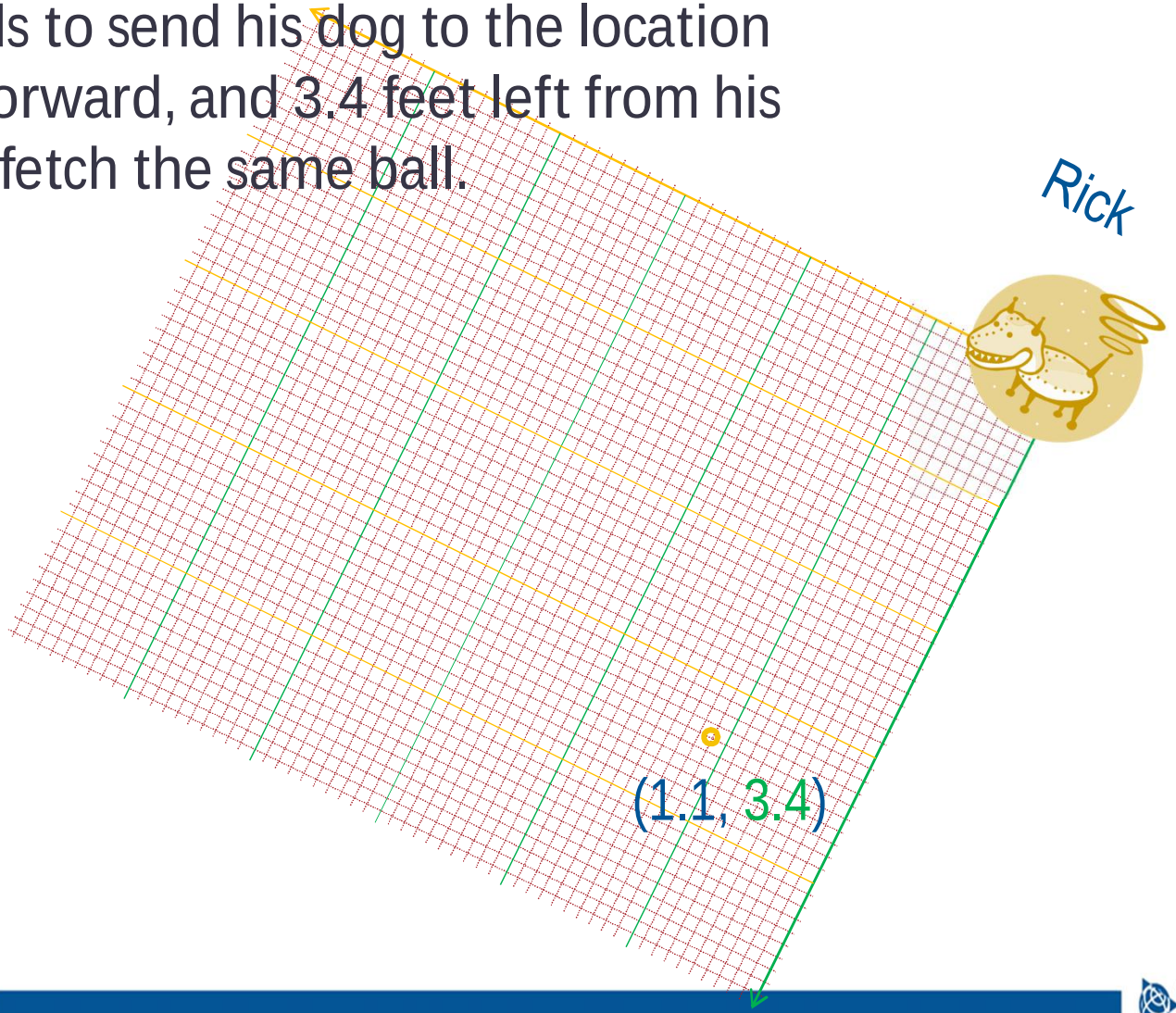
Matrixes

- To fetch the ball, Mat must send his dog to the location 3.5 feet forward, and 1.5 feet left from his house.



Matrixes

- Rick needs to send his dog to the location 1.1 feet forward, and 3.4 feet left from his house to fetch the same ball.

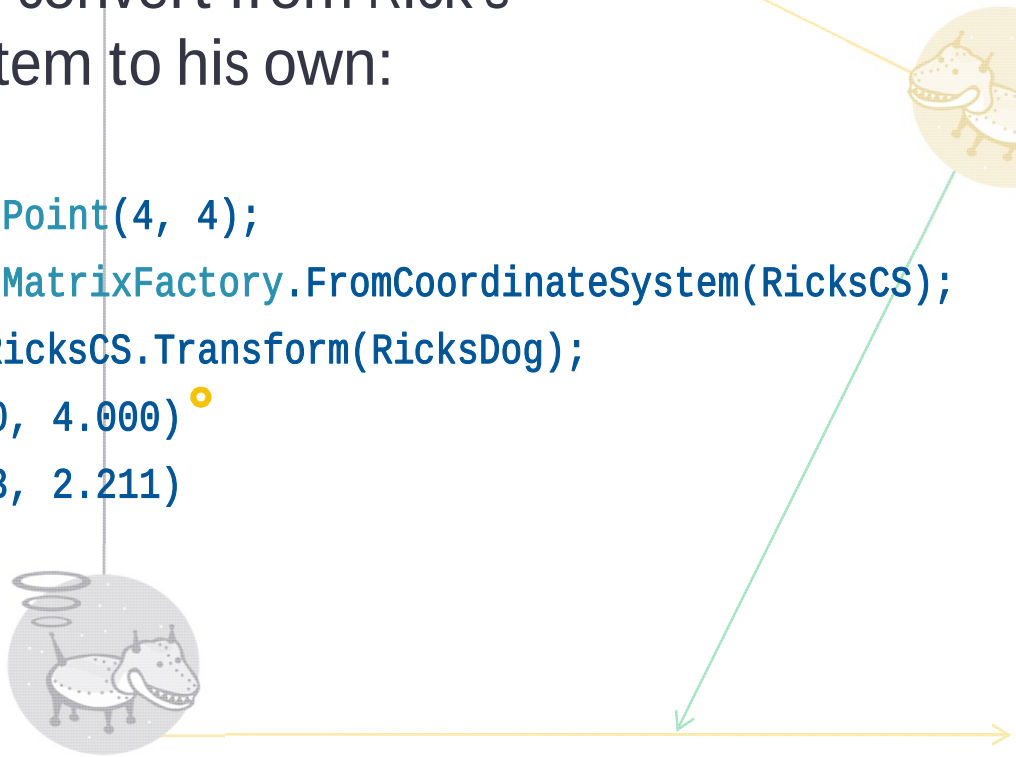


Matrixes- From

- Suppose Mat and Rick want their dogs to meet. Rick will send his dog to 4 forward and 4 left.
- For Mat to send his dog to the same place, he will need to convert from Rick's coordinate system to his own:

```
Point RicksDog = new Point(4, 4);
Matrix FromRicksCS = MatrixFactory.FromCoordinateSystem(RicksCS);
Point MatsDog = FromRicksCS.Transform(RicksDog);
```

```
    Rick's Dog: (4.000, 4.000)
    Mat's  Dog: (0.633, 2.211)
```

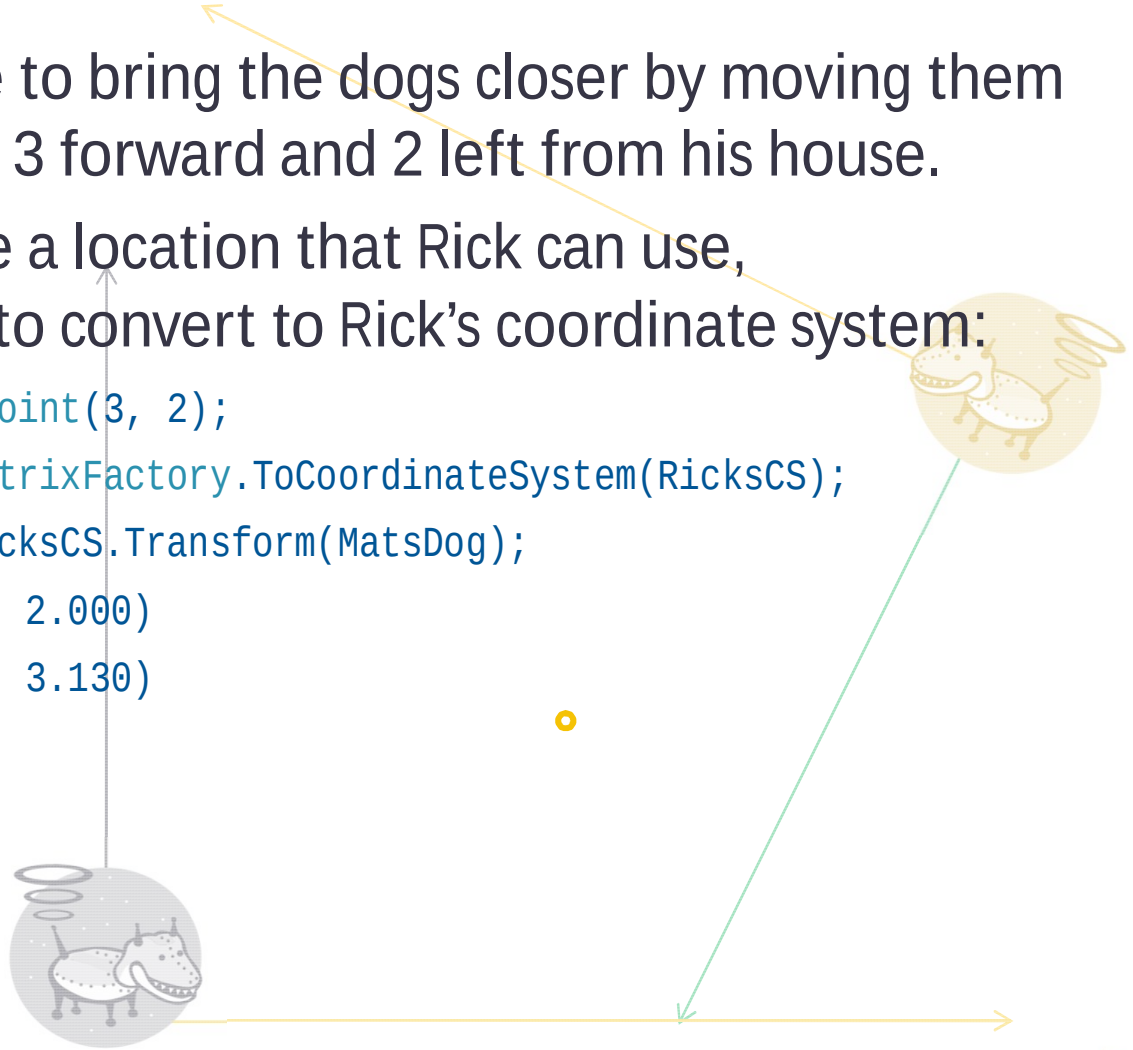


Matrixes- To

- Mat would like to bring the dogs closer by moving them to the location 3 forward and 2 left from his house.
- For Mat to give a location that Rick can use, Mat will need to convert to Rick's coordinate system:

```
Point MatsDog = new Point(3, 2);
Matrix ToRicksCS = MatrixFactory.ToCoordinateSystem(RicksCS);
Point RicksDog = ToRicksCS.Transform(MatsDog);
```

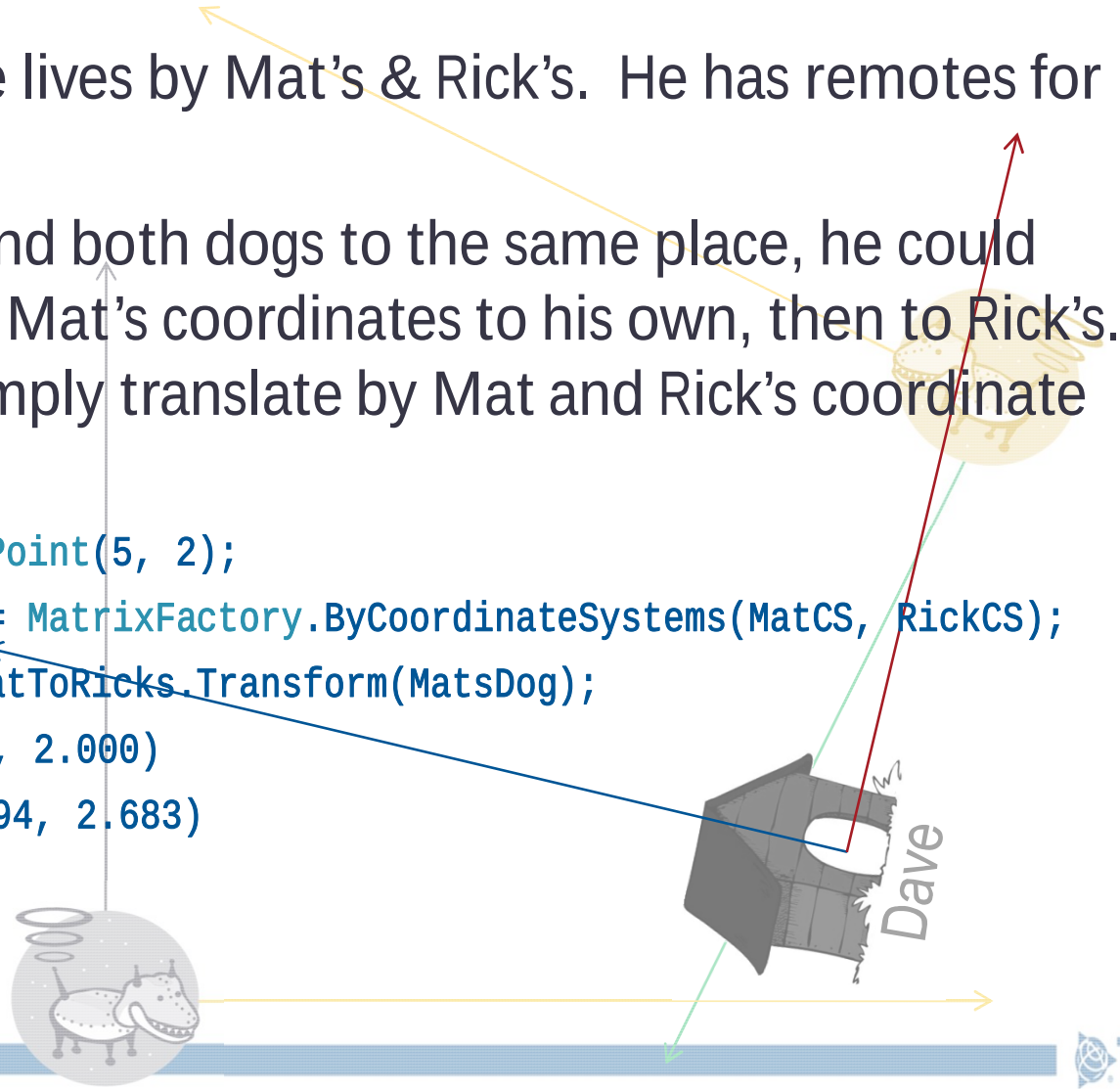
```
    Mats  Dog: (3.000, 2.000)
    Ricks Dog: (1.789, 3.130)
```



Matrixes- By

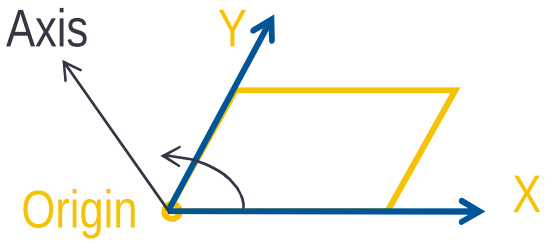
- Neighbor Dave lives by Mat's & Rick's. He has remotes for both dogs.
- For Dave to send both dogs to the same place, he could translate from Mat's coordinates to his own, then to Rick's. Or he could simply translate by Mat and Rick's coordinate systems.

```
Point MatsDog = new Point(5, 2);
Matrix ByMatToRicks = MatrixFactory.ByCoordinateSystems(MatCS, RickCS);
Point RicksDog = ByMatToRicks.Transform(MatsDog);
    Mat's Dog: (4.000, 2.000)
    Rick's  Dog: (0.894, 2.683)
```



Matrixes - Rotation

- It is also possible to create a matrix to rotate points around a given axis
  - Rotation is centered at the origin of the current work plane
  - Combine with a From and To matrix to rotate around a different origin, or use a vector to shift each point, rotate, and then shift back.



Matrixes – Notes

- Multiplication is also supported

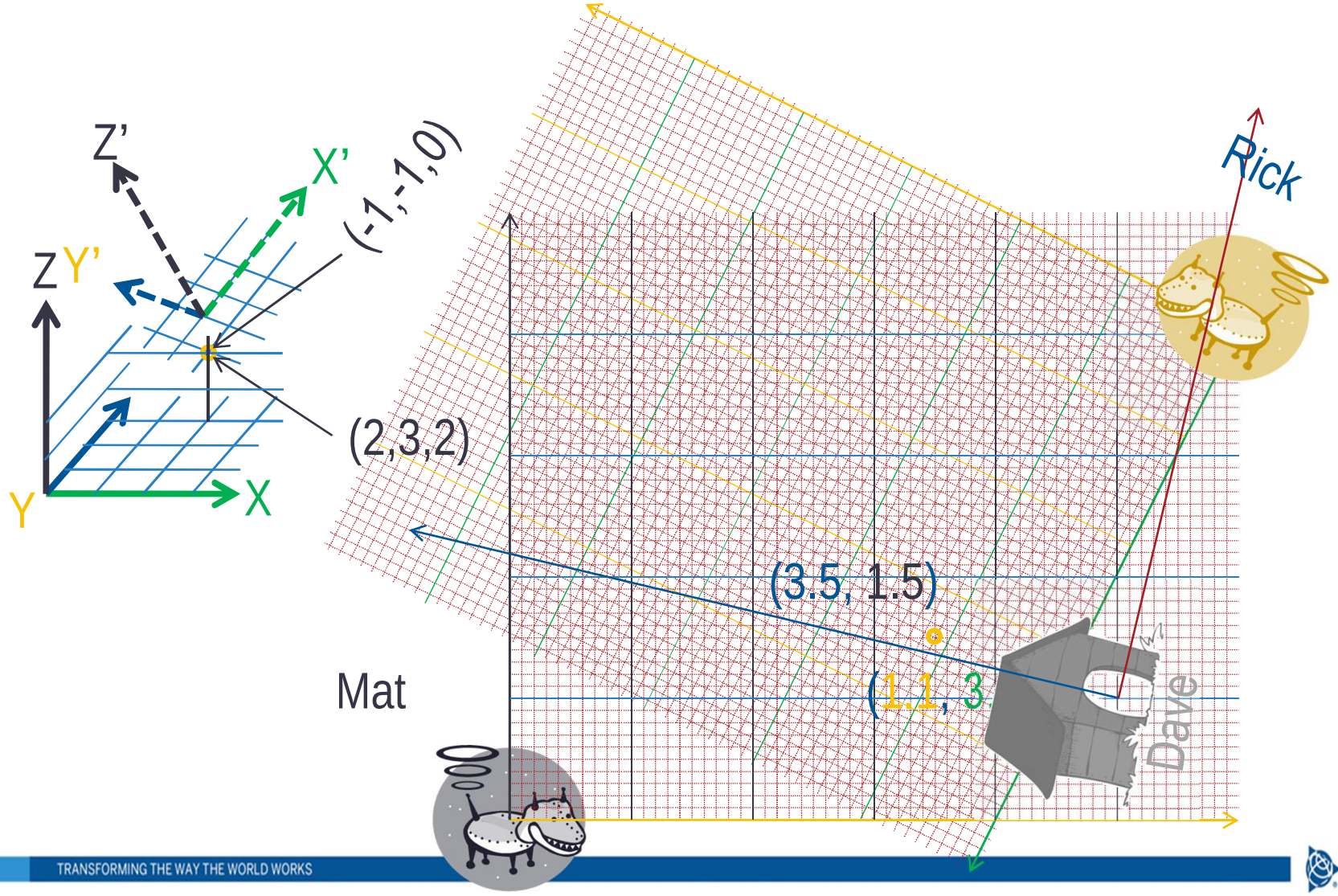
```
translatedPoint = myMatrix * myPoint;
newMatrix = myMatrix * myOtherMatrix;
```

- In addition to the MatrixFactory and TransformationPlanes, it is possible to directly modify and create a matrix.

```
Matrix myMatrix = new Matrix();
myMatrix[0, 0] = 2.0;
```

- Transforms vectors as points
  - To translate a vector, either edit the matrix to remove the origin shift or create a new vector by subtracting the origin

Matrixes



Matrix – ByCoordinateSystems

- One more example
  - Suppose we are in a component or part's local coordinate system and we measure the location of a hole and store that point.
  - Now, we go to the global coordinate system and want to create a hole in the same position on any new parts we click. Here are two approaches:
    1. Change the work plane for every new part picked and create the hole it in the same position (using the stored and un-translated point)
       OR
    2. Use ByCoordinateSystems to map the point from the original part coordinate system to the coordinate system of each successive part we pick

Workplane – Important Reminders

- Objects are ‘initialized’ in the current workplane.
- ModelObject.Select() will update an object’s properties and attributes after changing the work plane.
- Points, vectors, and non-model objects cannot be updated using Select(), but must be updated using matrixes.
- In general, report values are always returned according to the global coordinate system.

Tekla.Structures.Geometry3D

Classes and methods for 3D geometry and math

Geometric Objects

- Point

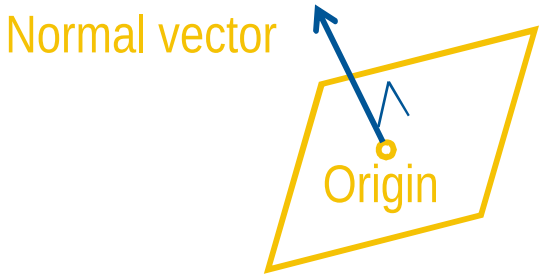
- Vector

- Line

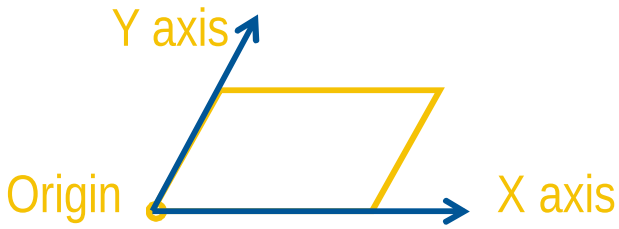
- Line Segment

Geometric Objects

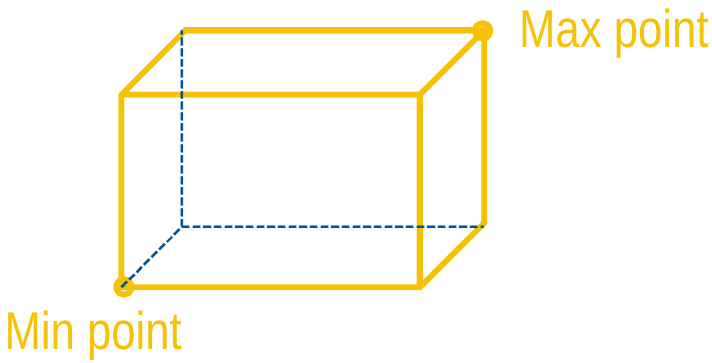
- Geometric Plane



- Coordinate System



- Axis Aligned Bounding Box (AABB)



Tekla.Structures.Geometry3D

- Geometric calculations and tests
  - Distance
    - Point to Point, Point to Line, Point to Plane
  - Parallel
    - Line to Line, Line to Plane, Plane to Plane
- Geometric constructions
  - Intersection
    - Line to Line, Line to Plane
  - Projection
    - Point to Line, Point to Plane, Line to Plane
- Matrix
    - Transformations from one coordinate system to another
    - MatrixFactory for easy creation of matrixes

Solids

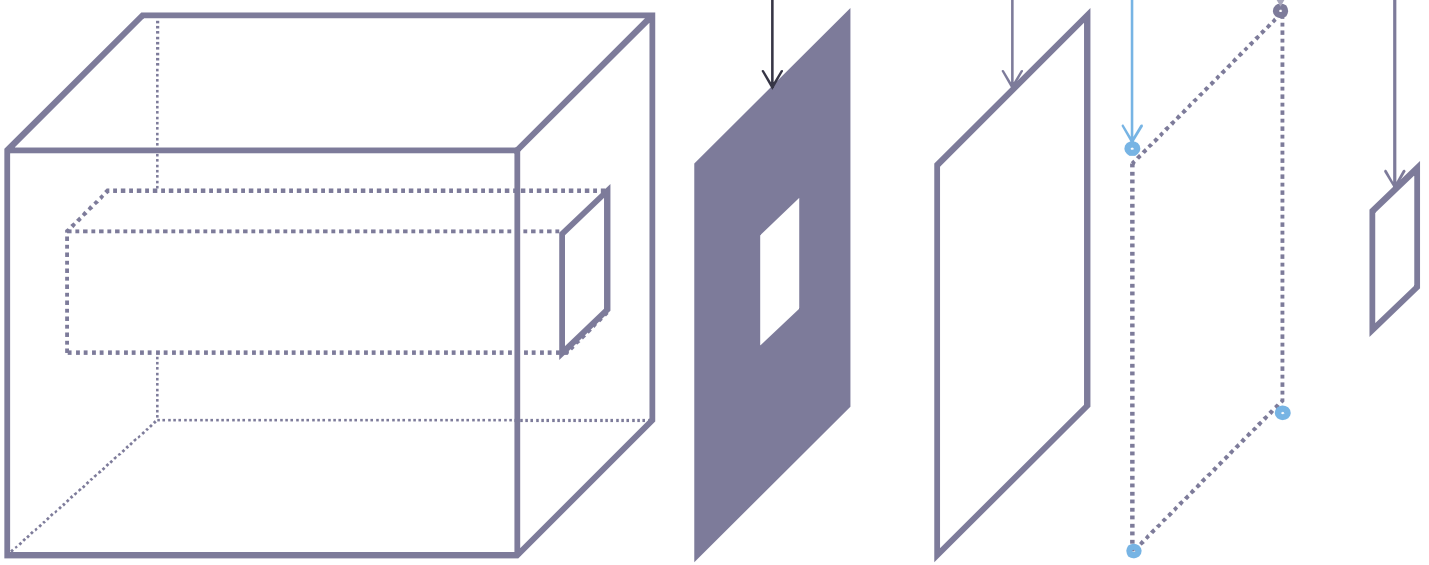
Exploring the content and structure of part solids

Content of a Solid

Object	Description
Solid	A single solid
FaceEnumerator	A list of faces in the solid
Face	A single face
LoopEnumerator	A list of loops in a face
Loop - Most faces have only one loop - If there is an opening, it is 2nd loop	A single loop
VertexEnumerator	A list of vertices in a loop
Point	A single vertex (point)

Content of a Solid

- Solid
  - Face
    - Loop (outer contour)
      - Vertex
      - Vertex...
    - Loop (inner contour – hole)
      - ...



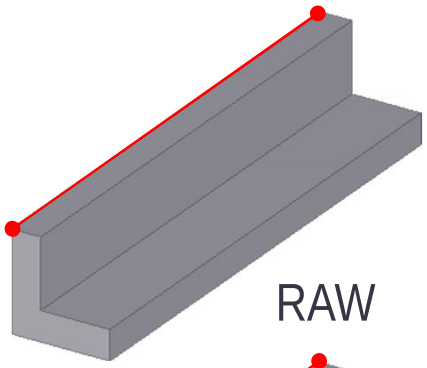
Solid properties and methods

- Some properties and calculations can be obtained directly from the solid
- Solid min and max points
  - Calculated in the current work plane
  - Can be used to create axis aligned bounding boxes
- Solid intersections can be directly obtained
  - LineSegment
    - point(s) of intersection between the solid and the line segment
  - Plane
    - Returns loop(s) of intersection between the solid and the plane

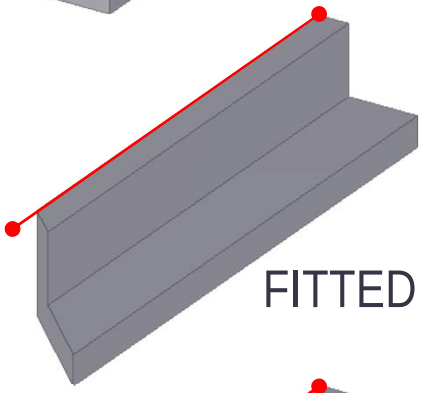
Solid options

- GetSolid() can return different solids
  - RAW
    no boolean operations
  - FITTED
    fittings only
  - PLANECUTTED
    fittings and cut planes only
  - NORMAL_WITHOUT_EDGECHAMFERS
    fittings, cut planes, part cuts, and part adds
  - NORMAL
    all cuts, fittings, edge chamfers, and part add operations
    The default solid type returned if no type is specified
  - HIGH_ACCURACY
    same as NORMAL with an exact profile cross section (e.g. fillets for hot rolled profiles)

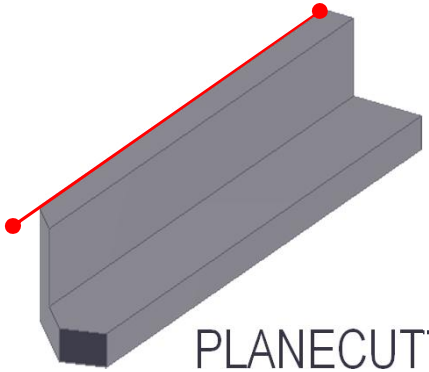
Solid options



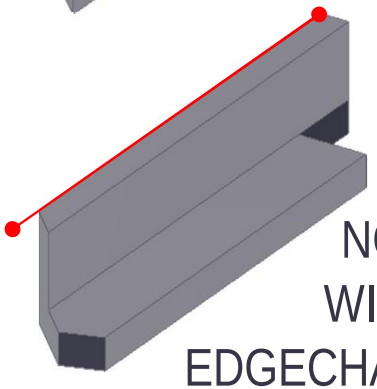
RAW



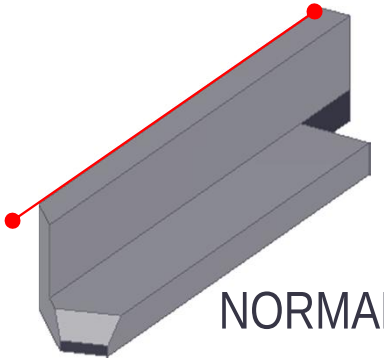
FITTED



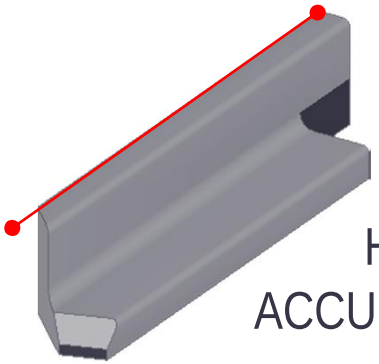
PLANECUTTED



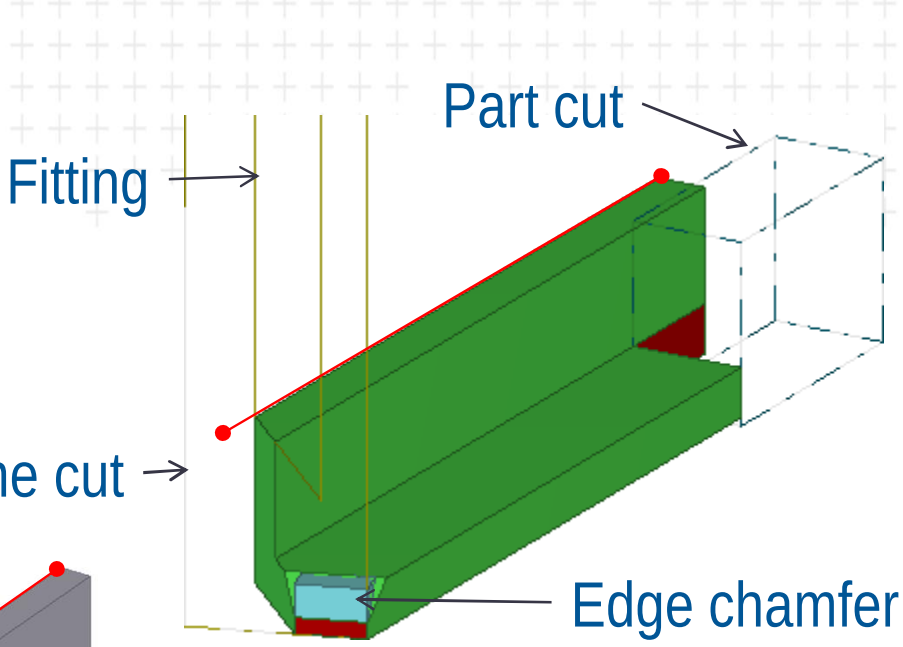
NORMAL_
WITHOUT_
EDGECHAMFERS



NORMAL



HIGH_
ACCURACY



Booleans: Cutting & Gluing

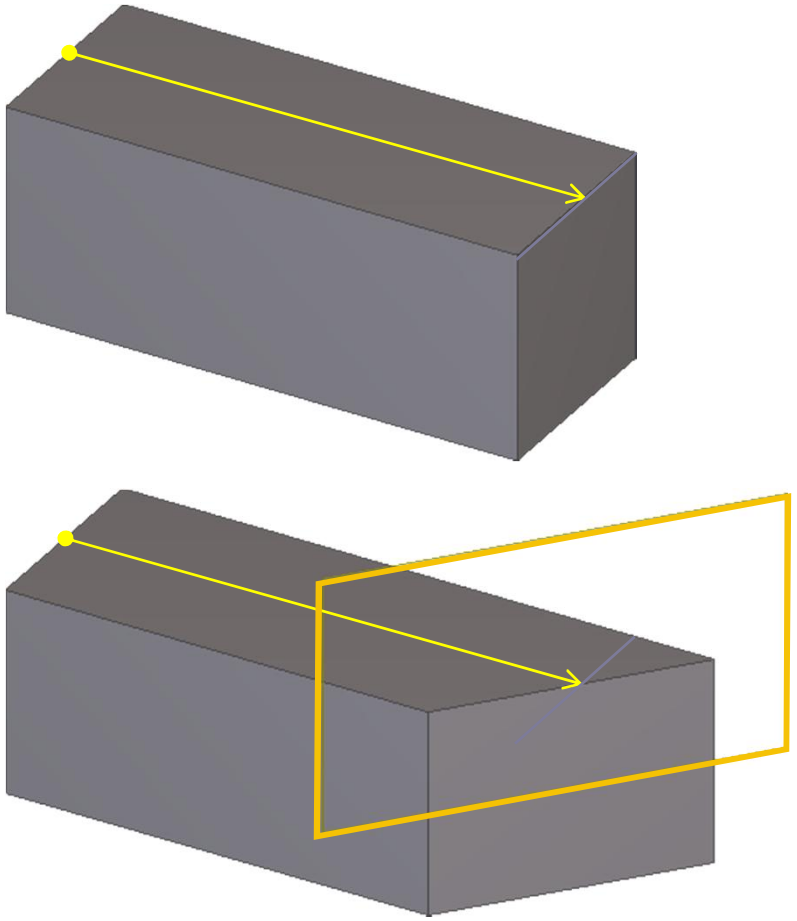
Objects which modify the raw solid

Tekla.Structures.Model

- Boolean
  - Fitting
  - CutPlane
  - BooleanPart
  - EdgeChamfer
- BooleanPart.Type (BooleanTypeEnum)
  - BOOLEAN_ADD ‘Part add’
  - BOOLEAN_CUT ‘Part cut’
- Part.GetBooleans()
  - Returns a ModelObjectEnumerator of all 1st level Boolean operations on the solid.

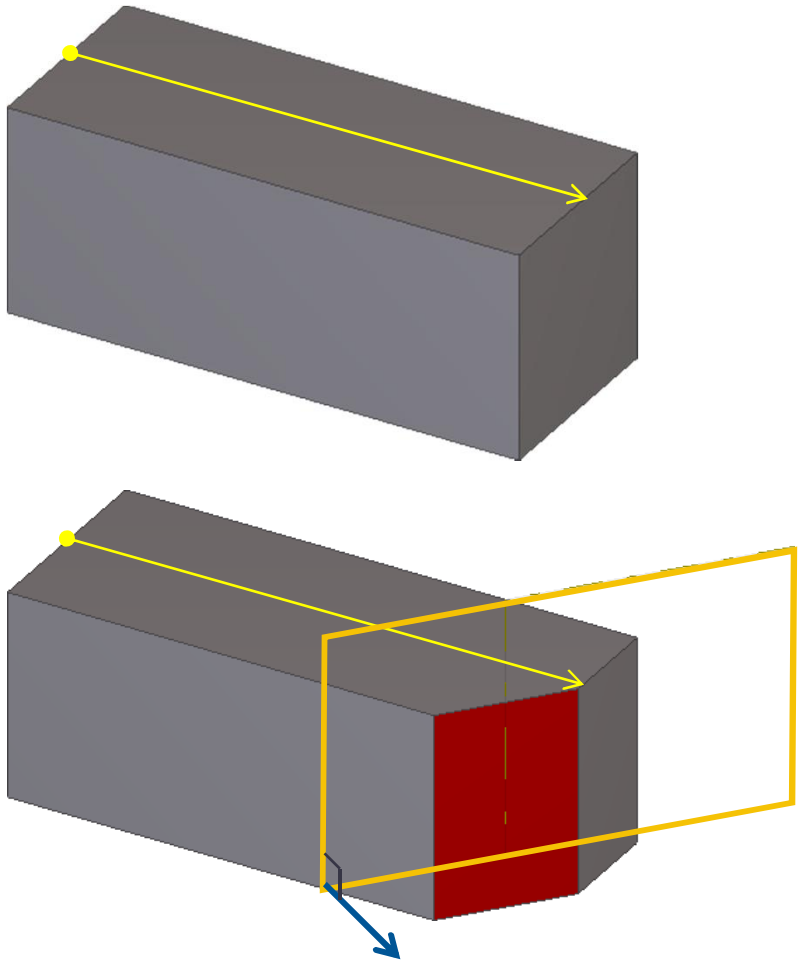
Fitting

- A fitting plane describes a plane to which the part cross section is extended, cut, or both.
- A fitting can only be applied to a Beam or PolyBeam object.



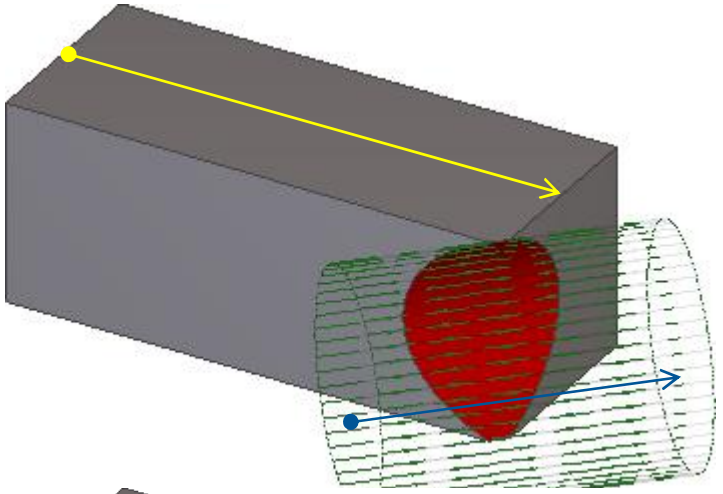
CutPlane

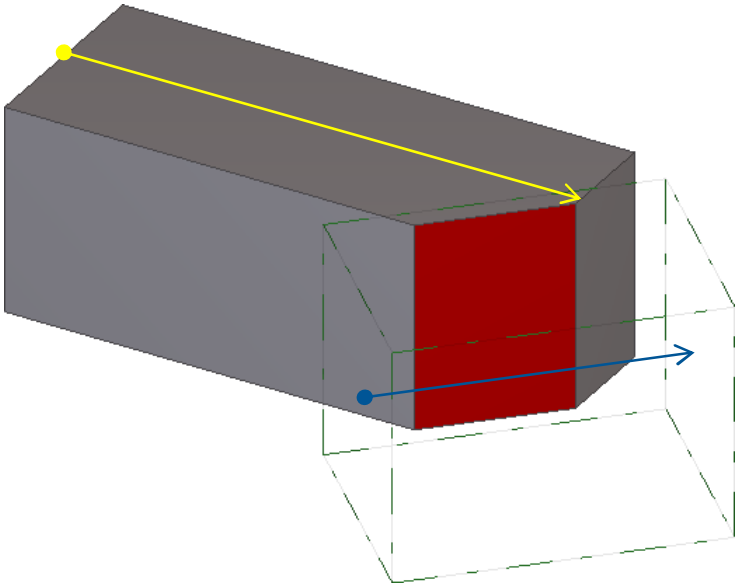
- A cut plane describes a plane beyond which the part is cut.
- A cut plane can be applied to any a Beam, PolyBeam, or ContourPart object.
- The normal or z vector of the cut plane indicates the side to be removed.

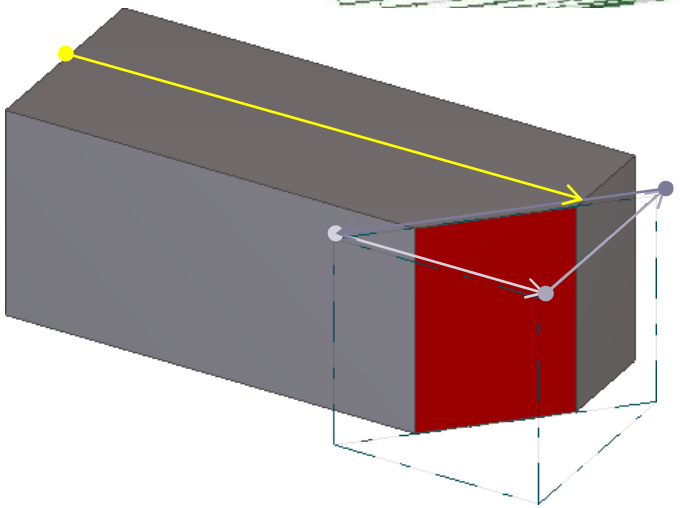


BooleanPart - Cut

- A part cut part can be a Beam, PolyBeam, or ContourPlate, and can be applied to a Beam, PolyBeam, or ContourPlate object.

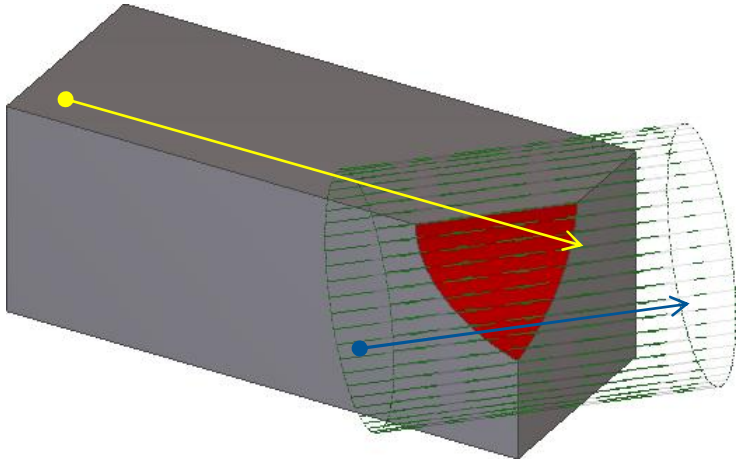


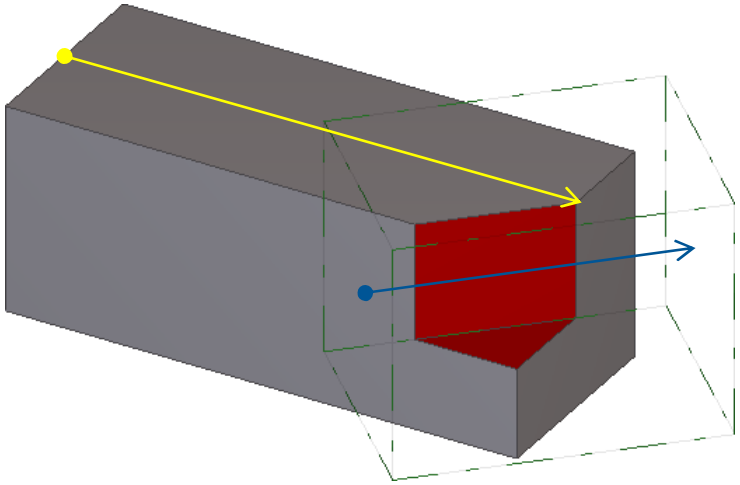


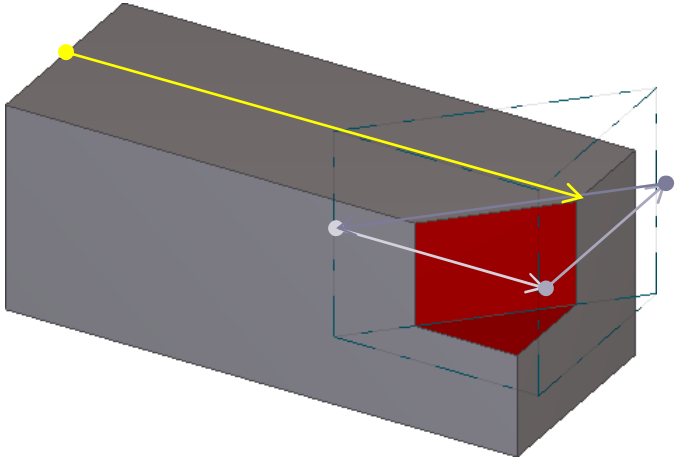


BooleanPart - Cut

- A part cut removes the solid intersection of the cutting part and the part.

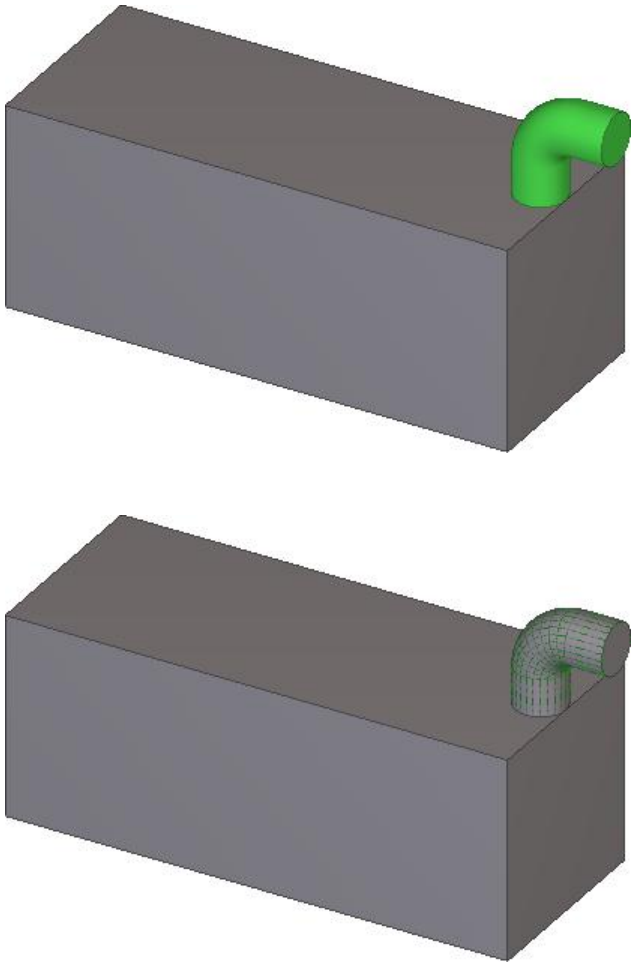






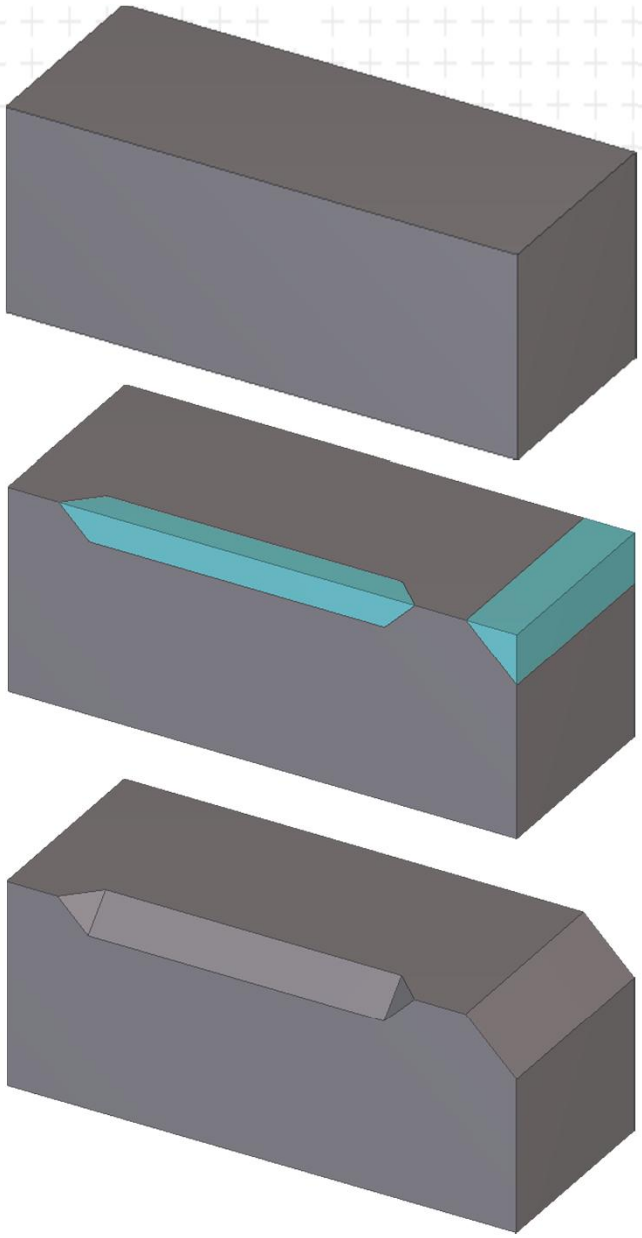
BooleanPart - Add

- Part adds are rarely used, but represent the addition of a solid to the original part solid.
- A part add can be defined by a Beam, PolyBeam, or ContourPlate



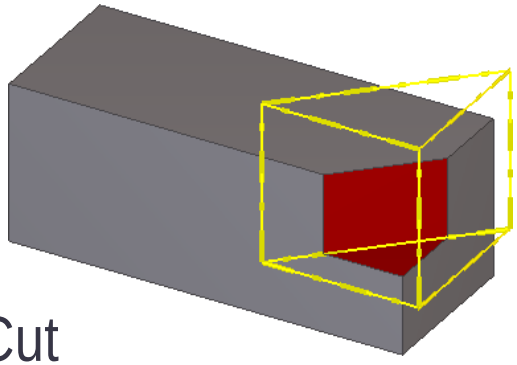
Boolean - EdgeChamfer

- An edge chamfer defines a chamfer or bevel along an edge of the part solid.
- The ends of a partial chamfer may be beveled.

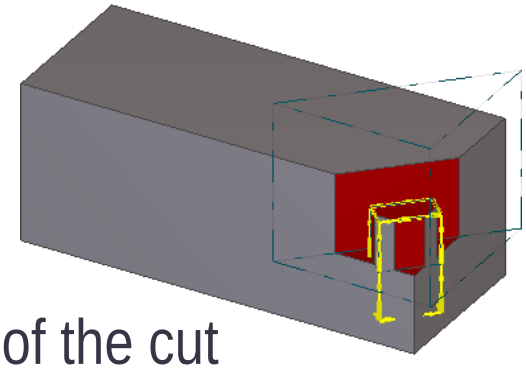


Boolean – General notes

- BooleanParts
  - Cutting parts often extend outside the solid they cut
  - In this case the cut only removes the existing solid
  - Portions of the cut outside the solid are ignored
- Booleans can be used in combination and nested
  - For example a BooleanPart may be cut by another cut to allow for complex cuts, or may cut a part add
  - BooleanPart.GetChildren() provides access to nested Boolean operations



Cut



Cut of the cut

Additional resources

- Vectors
  - http://en.wikipedia.org/wiki/Euclidean_vector
  - http://www.draftsperson.net/index.php?title=Understanding_Vectors
- Matrixes

  Technical note: the matrixes used in the Geometry3D name space represent affine transformation matrixes.
  - http://en.wikipedia.org/wiki/Transformation_matrix
  - http://en.wikipedia.org/wiki/Translation_matrix
  - http://en.wikipedia.org/wiki/Affine_transformation
  - http://en.wikipedia.org/wiki/Rotation_matrix

Trimble

Thank You