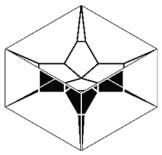


PolyFrame Beta 0.1

What is PolyFrame?

PolyFrame is a geometry-based, structural form finding plugin for Rhinoceros3d based on the principle of the equilibrium of polyhedral frames known as 3D graphic statics. In graphic statics, there are two diagrams: the form diagram that represents the geometry of the structure, and the force diagram that describes the equilibrium and the magnitude of the internal and external forces in the form. These 2D diagrams are topologically dual and geometrically dependent. In 3D graphic statics (3DGS), the equilibrium of the external forces or a single node of an equilibrated structure is represented by a closed polyhedron or a polyhedral cell with planar faces. Each face of the force polyhedron is perpendicular to an edge in the form diagram, and the magnitude of the force in the corresponding edge is equal to the area of the face in the force polyhedron. These form and force diagrams are reciprocal. i.e., topologically and geometrically dependent: any the change in one diagram results in a change in the other. Moreover, the force diagram represents the equilibrium and magnitude of the internal and external forces geometrically, a property that none of the other form finding tools possess (for further reading, please visit references section).

PolyFrame is a computational framework that allows the construction and manipulation of reciprocal polyhedrons for structural form finding. With this tool, a designer can simultaneously design and control the flow of forces in a structural form and drive the optimal structural solutions constrained to certain boundary conditions or specific geometric properties such as edge lengths, surface areas, etc. PolyFrame is based on a polyhedral construction and manipulation environment called PolyFramework.

Data structure

In order to facilitate the manipulation of polyhedral clusters, PolyFramework employs a data structure to maintain the topological and geometric relationships that define the cluster. Each complete PolyFrame object (also known as PFoam) contains elements that encode all the subparts of the cluster: cells (PFCell), faces (PFFace), edges (PFEdge) and vertices (PFVertex).

Naming convention

The majority of the PolyFrame functions are based on the relationship between two reciprocal polyhedrons one representing the force while the other representing the form.

The majority of the PolyFrame functions are based on the relationship between two reciprocal polyhedrons one representing the force while the other representing the form. In the context of PolyFrame, the two objects are topologically dual, and they would become reciprocal if and only if the edges of one are perpendicular to the faces of the other. The duality and perpendicularity are the two prerequisites of static equilibrium in the structural design process of the PolyFrame. As a result of this connected relationship between two diagrams, the naming convention in PolyFrame also follows these prerequisites. For simplicity reasons, the input diagram, including the faces of the polyhedron geometry (brep geometry), will be referred to as primal, and the diagram derived from primal will be called the dual. The primal/dual denominations are irrespective of the form/force designation of the diagrams. i.e. they can be interchanged depending on which diagram is considered as the primal.

PolyFrame and PolyFramework are developed by the Polyhedral Structures Laboratory at PennDesign, University of Pennsylvania.

Principal Investigators

Dr. Masoud Akbarzadeh (masouda@upenn.edu), Dr. Andrei Nejur (nejur@upenn.edu)

Software Developers

Dr. Andrei Nejur, Dr. Masoud Akbarzadeh

Command descriptions

Until now, PolyFrame adds 7 new commands to the Rhino environment

PFBuild 

PFBuild creates a PolyFrame data object using native Rhino brep geometry.

- The accepted **input** for the command can consist of polyhedrons represented either as individual faces or as polysurfaces. The command has three options.
- PointColapseLimit governs the minimum distance between the vertices of two breps to be considered separate. Under this limit, all vertices collapse into one single vertex. This is useful when dealing with non-perfect input where initial geometry has tiny overlaps or gaps. Setting this too low can cause geometry to remain disjointed thus resulting in unacceptable input or producing undesirable results. Setting

PointColapseLimit to high (in general higher that the shortest edge in the input) can cause edges to collapse into single points with all its topological consequences.

- PlanarityMaxDeviation sets the maximum allowable deviation, i.e., the perpendicular distances of the vertices from the average plane of the input faces. While PolyFrame can process non planar breps as input, large deformations can cause unexpected errors in the cell finding process. A very small value can cause the command to reject some of the otherwise planar input as non-planar.
- ReplaceGeo decides whether the constructed PolyFrame will replace the input geometry or it will be placed somewhere else in the document.
- Output options:
    - Type defines the type of container geometry for the PolyFrame data structure. The possible choices are Edges – for line based geometry; BrepFaces for trimmed surfaces representing each face; BrepCells for polysurfaces representing each cell; MeshFaces for individual meshes for each face; MeshCells for a mesh based representation of each PolyFrame cell.

PFDual

PFDual creates the dual of a PolyFrame structure. Each element in the input (the primal) has a corresponding member in the dual. Primal Cells map to Vertices of the Dual Vertices; Primal Faces map to Dual Edges; Primal Edges map to Dual Faces; and Primal Vertices map to Dual Cells.

- Input for the command is a PolyFrame object from the Rhino document. i.e., the user should first run the PFBuild to be able to run PFDual. Several options are available for the input stage:
    - UpdateGeo decides whether the input PolyFrame object will be updated with the recent geometric manipulations;
    - ClearConstraints specifies whether it should preserve the geometric constraints of the PolyFrame object.

- Output options
    - Type a user can choose from the following geometric types [Edges, BrepFace, BrepCells, MeshFaces, MeshCells].

PFPerp

PFPerp establishes the structural reciprocity between a primal and its dual. Using the topological relationship, the tool iteratively moves the vertices of the primal until the primal edges are perpendicular to the dual face planes. The command has layers of options for input, processing, constraints and output.

- Input
    - UpdateGeo assumes or discards the transformations of the container geometry
    - ClearConstraints allows the user to carry the previously-defined geometric constraints in the most recent PolyFrame object.

- Processing
    - MaxIterations controls the maximum allowable number of cycles the tool makes before it stops which is 1000 as a default value. For more complex models, consider increasing this value. The process will also stop if the solution reaches prescribed maximum perpendicularity deviation (see next option), where the solution stagnates (no significant vertex movement is recorded between consecutive steps). The use can also stop the command at any step by pressing <ESC> key.
    - MaxDeviation stipulates the maximum allowable angle deviation between a primal edge and the normal of its dual face. The iteration stops. If all angles between primal edges and the normal of the dual faces are smaller than this value.
    - MinEdgeLength defines the minimum edge length for the dual during the process. Please note that a large value for the MinEdgeLength will cause the structure to grow and that might require more iteration steps until the convergence. On the other hand, setting a very small value can produce very short edges or can cause some edges to flip during the process. Best value is usually determined through experimentation for each case.
    - SetEdgeLength allows the user to set target lengths for the edges of the current dual. This is a perpedincularization constraint. The option opens a new dialogue that allows for the selection of edges in the PolyFrame. The user can select one/multiple edges and set the target lengths in a dedicated pop-up dialogue.
    - SetVertPos allows the user to set the position of vertices in the dual diagram. This is a perpedincularization constraint. The option opens a new dialogue that allows for the selection of vertices and subsequent position constraints for them. The user can select either one vertex or multiple vertices in sequence. By

default, the user can click on any geometry in the document and use it as a constraint. Currently, there are two options in the command line: Outside has a meaning only for closed constrain geometry and specifies whether the point will be kept inside the geometry rather than on its surface. Fixed simply constrains the selected vertices to their current positon.

- UpdatePrimalGeo as the name implies can update the primal geometric information stored in the PolyFrame object. For this operation to succeed the dual will be perped to the updated primal.

- Output options:
  - Type [Edges, BrepFace, BrepCells, MeshFaces, MeshCells]

PFPlanarize 

PFPlanarize as the name implies, attempts to planarize all the faces of a PolyFrame or cluster of trimmed surfaces/polysurfaces regardless of the dual reciprocal relationships. The process moves vertices of the input geometry that onto their average face planes until all faces are planar within a provided tolerance. Like PFPerp the command allows the user to interact with options on multiple layers.

- Input
  - UpdateGeo assumes or discards the transformations of the container geometry
  - ClearConstraints allows the user to erase/preserve the existing transformation constraints in the current PolyFrame. See below for clarifications regarding constraints
- Processing
  - Maximum allowed vertex deviation stipulates the maximum allowed distance between a vertex and the average plane of a face containing that vertex. If all vertex distances from the face planes are smaller than this value, the iteration stops. In Rhino, surface with vertex deviations less than 1e-4 (0.0001) units are considered planar. PolyFrame can work with much larger planarity deviations. The only problem may arise if BrepFaces or BrepCells are chosen for the container geometry and faces are not planar. In such cases, the geometry will still be created, but may have small imperfections and take longer to be visualized. The default set value should work for most cases.

- MaxIterations controls the maximum number of cycles the tool is allowed to run before it stops. 5000 is a conservative value. For more complex models, consider increasing it. Even if the user sets this value too high the iteration will stop if the solution reaches maximum perpendicularity deviation (stagnant solution where no significant vertex movement is recorded between consecutive steps), or user pressed <ESC> key.
- Algorithm allows the user to choose between two planarization algorithms. SType or soft planarization allows for the use of geometric constraints. However, it is relatively a slow process and is less capable of reaching small deviations since it checks the stagnating solutions during the process. HType or hard planarization ignores any constraints and has no other automatic stopping criteria than MaxIterations or maximum allowed deviation. Nevertheless, it is faster than SType and more capable of reaching small deviations. HType is usually recommended if no geometry constraints are imposed.
- MinEdgeLength defines the minimum length an edge is allowed to shrink to during the process. Using the default value is suggested. Otherwise, edge flipping might occur during the planarization with extreme deviations from face planes.
- SetVertPos allows the user to set position in the primal. This is a geometric constraint. The option opens a new dialogue that allows for the selection of vertices and subsequent position constraints for them. The user can select either one vertex or multiple vertices in sequence. Once the selection set is confirmed another layer of options is provided for the user. By default, the user can click on any geometry in the document and use it as a constraint. There are two options in the command line at this time. Outside has meaning only for closed constrain geometry and specifies whether the point will be kept inside the geometry rather than on its surface. Fixed simply constrains the selected vertices to their present positon.
- SetEdgeLength allows the user to set target lengths for the edges in the primal. This is a constraint. The option opens a new dialogue that allows for the selection of edges in the PolyFrame. The user can select one edge at a time and set its target length in a dedicated pop-up dialogue. If multiple edges are selected in a sequence the provided target length will apply to all of them.
- SetFaceArea allows the user to set target areas for faces in the system. The option opens a new dialogue that allows for the individual selection of faces and subsequent specification of target areas. During the planarization process faces will be scaled to meet the target areas

- Output options
    - Type [Edges, BrepFace, BrepCells, MeshFaces, MeshCells]

PFSwitch 

PFSwitch allows the user to switch the representation of any PolyFrame object. The command simply loads the data from the document and saves it again replacing/duplicating the original input while allowing the user to pick another type of geometric representation.

- Input
    - UpdateGeo assumes or discards the transformations of the container geometry
    - ClearConstraints allows the user to decide whether the saved transformation constraints present in the PolyFrame data will be maintained or not.
- Output options
    - Type [Edges, BrepFace, BrepCells, MeshFaces, MeshCells]

PFPipe 

PFPipe creates a volumetric representation of a PolyFrame based on its container and type. If the selected PolyFrame represents a form diagram, the volumetric representation will use the force magnitude from its dual (the area of the dual faces) to create cylinders around its edges. If the selected PolyFrame represents a force diagram the command will extract its dual from the data structure and create the piped representation based on it. The produced pipes will have radii and colors corresponding to the magnitude of the forces in the dual.

- Input
    - UpdateGeo assumes or discards the transformations of the container geometry;
    - ClearConstraints allows the user to erase/rewrite the transformation constraints in the current PolyFrame object.
- Processing
    - MinRadius / MaxRadius control the interval the radii of the pipes. The force magnitude values read from the dual will be “normalized” between those two values while keeping the relative ratio between them.

PFTransform 

PFTransform allows for a transformation of the PolyFrame object without breaking the reciprocity, thus keeping the direction of its face normals constant.

- **Input**
  - **UpdateGeo** assumes or discards the transformations of the container geometry
  - **ClearConstraints** allows the user to erase/rewrite the geometric constraints.
- **Processing** allows for individual vertices to be picked and moved either freely or along connecting edges. The algorithm propagates the changes in the whole structure following the rules of parallelism. The command shows an interactive preview of the transformation with the color-coded edges: blue if they remain with the same direction, and red if they flip.

PolyFrame Workflows

Form finding of a compression-only/tension-only structure

The process of form finding of a compression-only/tension-only structure using PolyFrame follows three steps only. It creates an optimized structural form from a set of polyhedrons representing the forces diagram of the structure. The workflow is a linear one starting with the creation of a PolyFrame, continuing with the creation of its dual and the perpendicularization of that dual to reciprocity (equilibrium) between diagrams.

Prerequisites.

In order to begin, a start geometry representing the force diagram is needed. This must be present in the Rhino document as an aggregation of trimmed surfaces representing the faces of the force polyhedrons. Alternatively, it is possible to start from a set of polysurfaces that represent cells or any aggregation of polygon-trimmed surfaces. It is important that the faces are built and assembled together in the document without significant discontinuities or overlaps and that they are as planar as possible.

Step.1

Using the PFBuild command a PolyFrame is constructed from the input geometry. The default representation of the PolyFrame will be edges. Also by default, the original input geometry will be replaced by the PolyFrame container. The image in step one shows a BrepCells representation of the constructed PolyFrame. The reason for this is a better

contrast between the input and the result of the command. (BrepCells is available as an option for every PolyFrame command that produces or modifies a PolyFrame container)

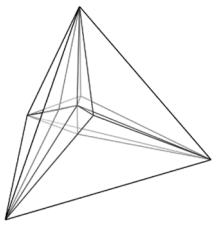
In the event that the input geometry does not meet the minimum criteria to be turned into a PolyFrame, the tool will abort and bake the problem geometry into a separate layer entitled <<Error Geometry>> in orange.

Step.2

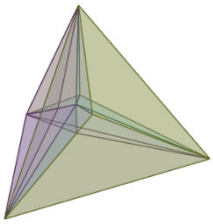
Create the dual of the built PolyFrame from the previous step using the PFDual command and the default options. If for some reason the input of the command is not an Edge container, make sure that for the output of the PFDual command the Edges option is selected.

Step.3

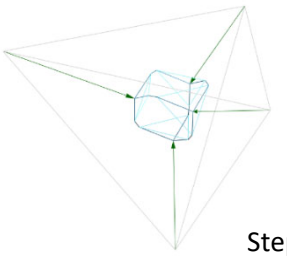
Perpendicularize the new structure with PFPerp using the default options. By default, the dual digram created at the previous step will be prependicularized (perped) in place.



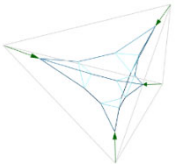
Step.



Step.



Step.



Step.

Perping a PolyFrame with constraints

This workflow will detail how to perpendicularize a PolyFrame structure with added constraints like certain positions for the vertices or certain set lengths for the edges.

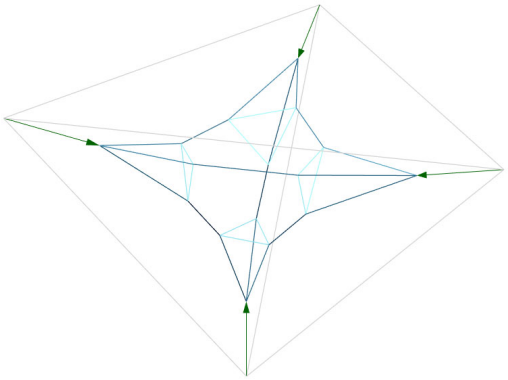
Prerequisites

In order to start this workflow, a form PolyFrame (the dual of a force) is required. Although reciprocity is not required as a starting condition it is advisable that the PolyFrame being constrained through perping starts from an equilibrium (reciprocal to force) state. It is actually advisable to use the output of the last step from the previous exercise to start this one.

If you plan to use geometric constraints for the vertices in the diagram, you need to have them ready in the Rhino document when PFPerp is started. For now, Points, Surfaces, PolySurfaces and Meshes are accepted as geometric constraints.

Step.1

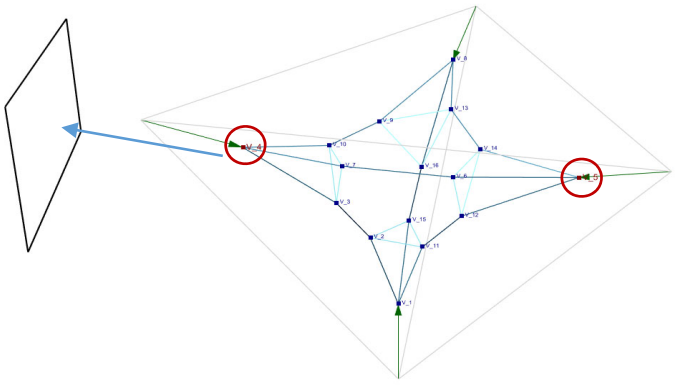
Load the form PolyFrame using the PFPerp command



Step.2

Set one vertex constraint.

For example, one vertex can be set to lay on a given surface. For this, after the work diagram was specified for PFPerp, click on the SetVertPos option. This will display all the vertices in the diagram that are available to be constrained. Click on one vertex (its color should change to red), confirm the selection with <Enter> and then click on the constraining geometry. If the operation succeeded the color of the vertex dot should now

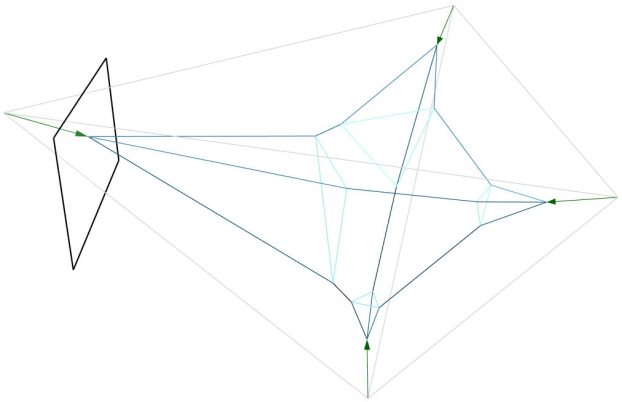


be light blue.

Step.3

Set another vertex constraint. Set this one vertex to fixed.

You should still be in vertex selection mode. If that is not the case, click on SetVertexPos again. Select the desired vertex and confirm like in the previous step, but instead of selecting a constraining geometry click on the Fixed option in the command line. If the operation succeeded, you should now have two light blue vertex dots in your diagram. Press <Enter> to exit the vertex selection section.



Step.4

If you are working on a diagram of similar complexity with the one depicted in the images, the standard options of the main PFPerp dialog should work just fine. Perform the actual perping operation by pressing <Enter>. You should now briefly see the diagram change shape on the screen.

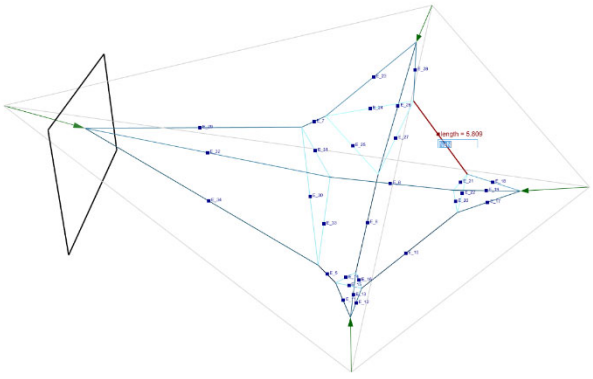
The constrained points should act according to their set constraints. Please note that the constraints are not absolute so some small deviation from the target geometry is to be expected.

Step.5

Reload the perped model into PFPerp maintaining constraints (use the default settings for loading the diagram). All constrained points will keep their previously prescribed constraints.

Step.6

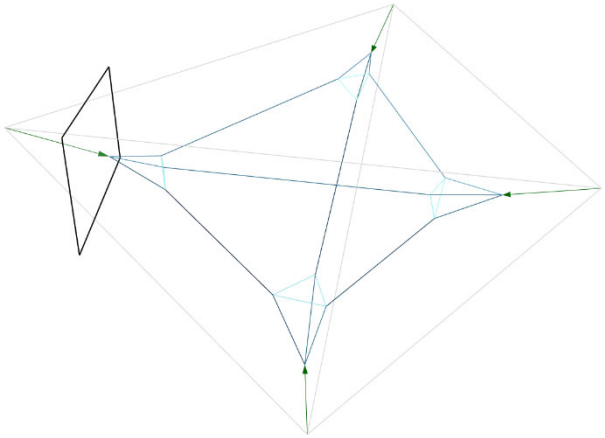
Set at least one edge length constraint. For this, after the work diagram was specified for PFPerp click on the SetEdgeLen option. This will display all the edges in the diagram that are available to be constrained. Click on one edge (its color should change to red and current length should be displayed on the screen), confirm the selection with <Enter> and a dialog will appear on the screen just below the edge midpoint asking for a number. Input the desired length in the dialog and press <Enter>. For example, set one edge to



twice its size. If the operation succeeded, the color of the edge midpoint should now be light blue. Press <Enter> to exit the edge selection section.

Step.7

Perform another perping round using the default settings similar to step 4. If the model is not fully perped at this point the operation can be repeated with a higher number of iterations. The command should show at the end, in the command line, how close to perfect perpendicularity the diagram ended up.



Please NOTE:

This manual, as the PolyFrame tool, itself are a work in progress. At times, information might be missing and the tools might misbehave, throw errors or crash. DO NOT use PolyFrame for production purposes. If you do, you are doing this at your own risk. PSL does not accept any liability for damages or losses incurred by the use of the PoyFrame tool.

As PolyFrame is updated so will be this manual. Make sure you are using the latest version of PolyFrame and Manual by downloading them for the PSL website.

References

M. Akbarzadeh, “Three Dimensional Graphical Statics using Polyhedral Reciprocal Diagrams,” *dissertation*, Zurich, 2016.

M. Akbarzadeh, T. V. Mele, and P. Block, “On the equilibrium of funicular polyhedral frames and convex polyhedral force diagrams”, *Computer-Aided Design*, vol. 63, pp. 118–128, 2015.

A. Nejur, M. Akbarzadeh, “Constrained manipulation of polyhedral systems”, *IASS 2018 : Creativity in structural design*, Boston, 2018.