FreeCAD Assembly Without Solver / Assembly 4

FreeCAD add-on for a bare-bone assembly structure, using App::Link

Welcome to the FreeCAD_aws wiki! This page tries to explain how to make assemblies using these tools. This project the result of discussions on the FreeCAD forum.

Installation

Download and extract the archive, and then move (or link) the sub-directory `Mod_Asm4` (which contains all the actual code) into the `~/.FreeCDA/Mod` folder, containing all additional modules.
Please bear in mind that Assembly 4 is not compatible with stock FreeCAD v0.18. Pre-built binaries of realthunder's branch can be downloaded from his GitHub repository (they are called FreeCAD-asm3-xxx but are compatible with Asm4).
You can use the example assemblies to experiment with this workbench's features.

Principle

The point is that each part is a mostly standard FreeCAD `App::Part` object, and these are assembled using the `App::Link` framework found in the fork of FreeCAD (https://github.com/realthunder/FreeCAD/tree/LinkStage3, pre-built binaries on the realthunder's FreeCAD_assembly3 release page

The particularities of the `App::Part` used here are the following:

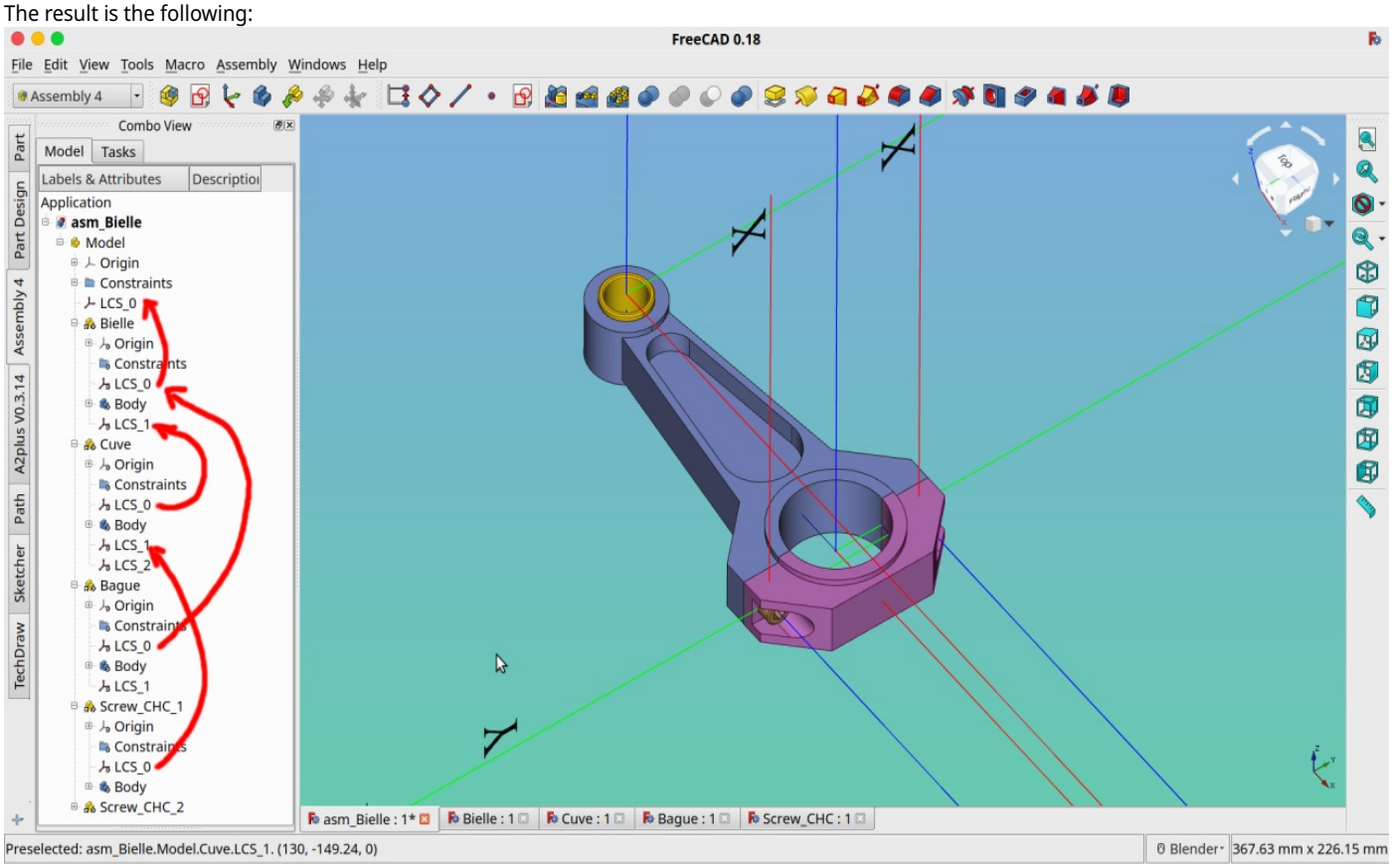
- they are called 'Model' at creation time
- they contain a group called 'Constraints' at the root
- they contain a Datum Coordinate System called LCS_0 at the root

The purpose of this is to avoid burdensome error checking: it is supposed that no other usecase would create an `App::Part` called 'Model', *i.e.* that if an `App::Part` is called 'Model' it will conform to these characteristics.

Any Model can contain (by `App::Link`) any other Model, and they are placed to each-other by matchings their Datum Coordinate Systems (`PartDesign::CoordinateSystem`, called here-after LCS (Local Coordinate System)). There is no need for any geometry to be present to place and constrain parts relative to each other. LCS are used because they are both mechanical objects, since they fix all 6 degrees of freedom in an isostatic way, and mathematical objects that can be easily manipulated by rigorous methods (mainly combination and inversion).

To actually include some geometry, a body needs to be created, and designed using the PartDesign workbench. To be linked with the previously created model, this body needs to be inside the `App::Part container` called 'Model'.

The result is the following:



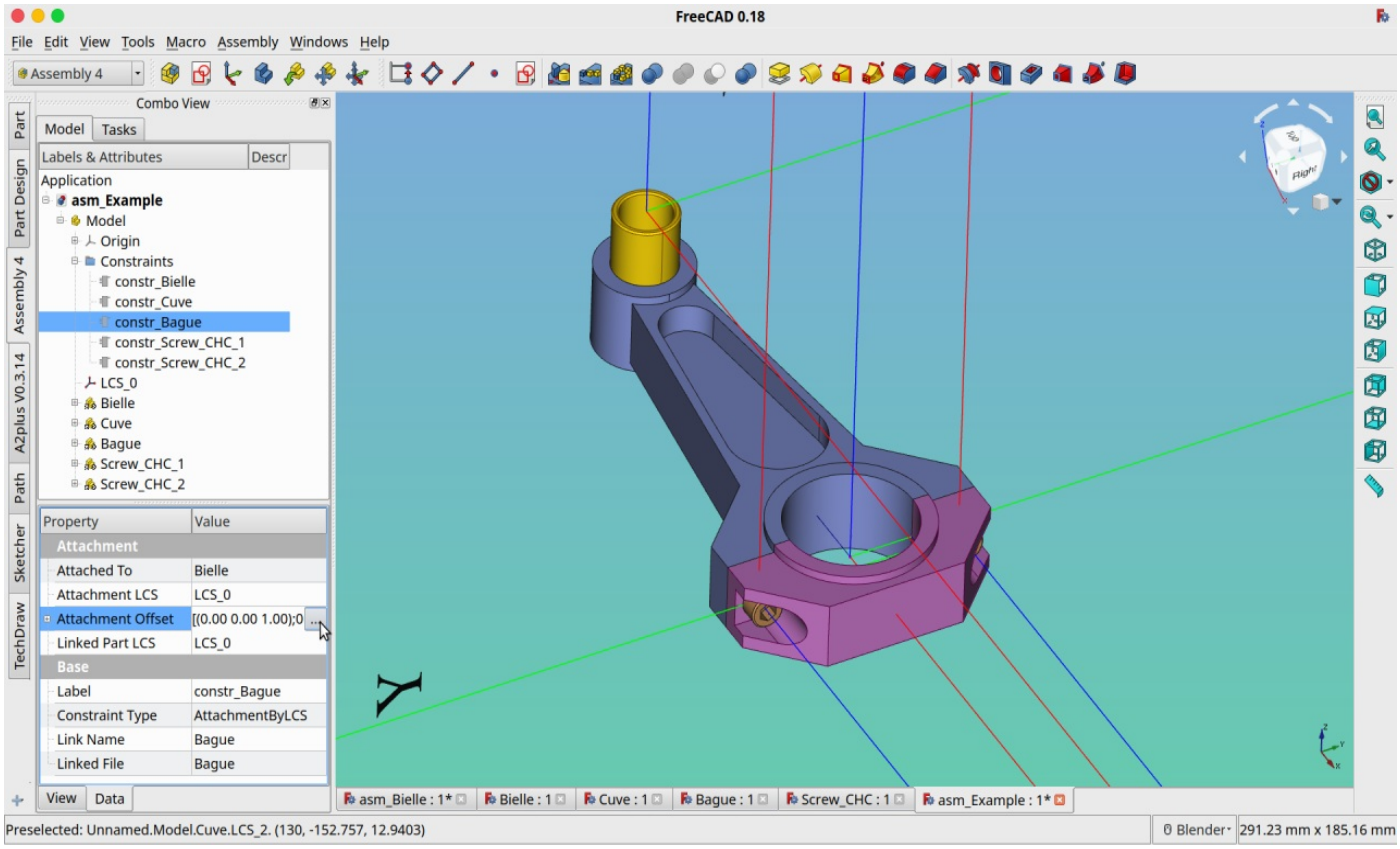
- the part *Bielle* is placed in the assembly by attaching it's *LCS_0* to the *LCS_0* of the parent assembly.

- the part *Cuve* is placed in the assembly by placing its *LCS_0* on the *LCS_1* of the part *Bielle*
- the part *Bague* is placed in the assembly by placing its *LCS_0* on the *LCS_0* of the part *Bielle*
- the parts *Screw_CHC_1* and *Screw_CHC_2* are placed in the assembly by placing their *LCS_0* on the *LCS_1* and *LCS_2* of the part *Cuve*

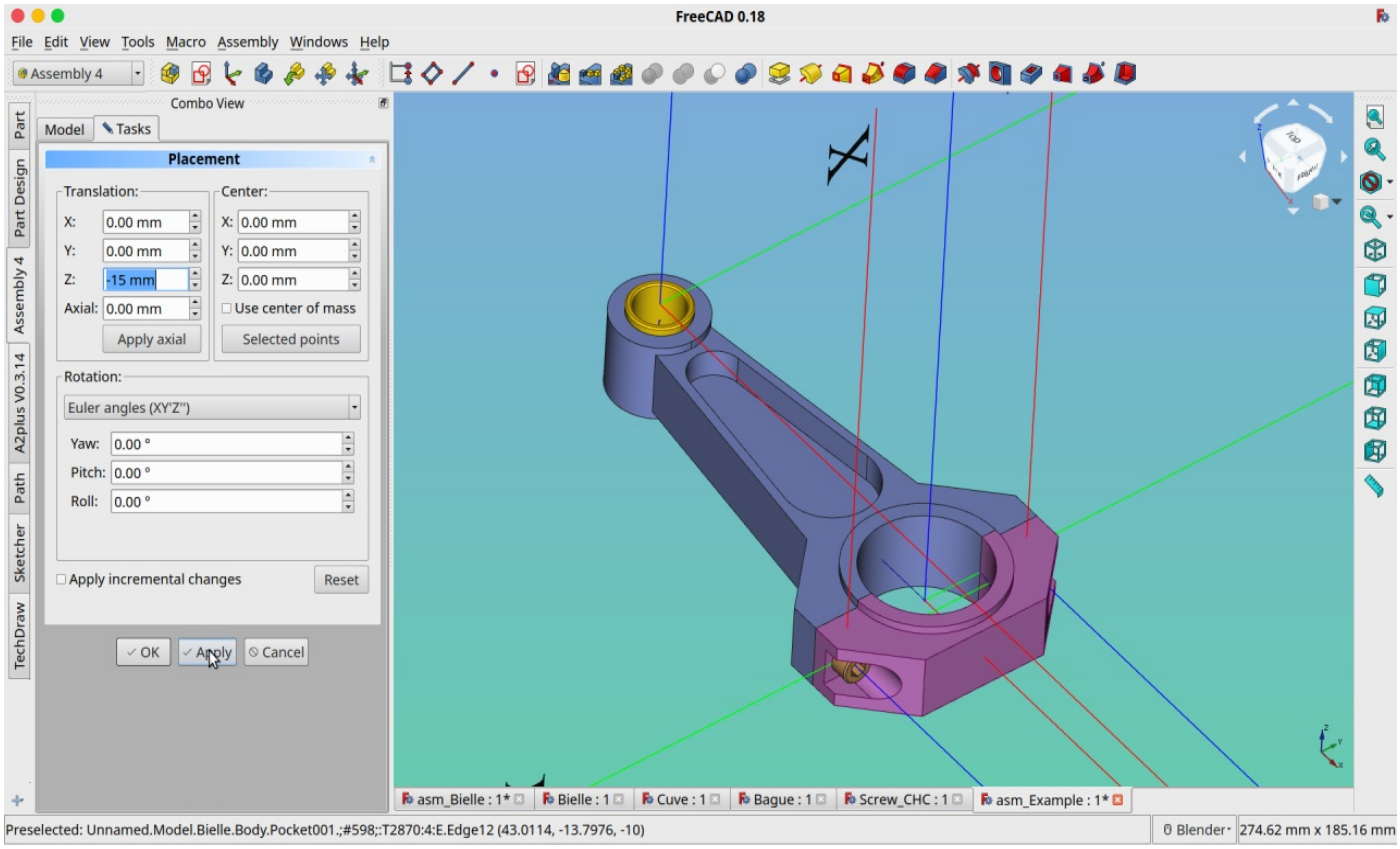
The 2 LCS - the one in the linked part and the one in the assembly, be it from the parent assembly or a sister part - are superimposed using an ExpressionEngine, with the following syntax:

- **If the LCS belongs to the parent assembly:**
  `<<attLCS>>.Placement.multiply(<<constr_Link>>.Offset).multiply(.<<partLCS.>>.Placement.inverse())`
- **If the LCS belongs to a sister part:**
  `<<attPart>>.Placement.multiply(<<attPart>>.<<attLCS.>>.Placement).multiply(<<constr_Link>>.Offset).multiply(.<<partLCS.>>.Placement.inverse())`
- **Constraints:**
  To each part inserted into an assembly is associated an `App::FeaturePython` object, placed in the 'Constraints' group. This object contains information about the placement of the linked object in the assembly. It also contains an `App::Placement`, called 'AttachmentOffset', which introduces an offset between the LCS in the part and the LCS in the assembly. The main purpose of this offset is to correct bad orientations between the 2 matching LCS.

  These constraints are not really constraints in the traditional CAD sense, but since `App::FeaturePython` objects are very versatile, they could be expanded to contain real constraints in some (distant) future.



Taking a closer look at the fields contained in an `App::FeaturePython` object associated with the part 'Bague'. The small button under the cursor opens the dialog that allows to edit the parameters of the Attachment Offset



Dialog that opens when clicking the previous small button, and permitting to edit the parameters of the `App::Placement` *called* 'AttachmentOffset' *in the constraint associated with a link, and allowing relative placement of the link -vs- the attachment LCS*

Workflow

Part

The basic workflow for creating a part is the following:

- create a new document, create a new 'Model' with the macro 'new_Model'. This will create a new App::Part called 'Model' with the default structure
- create or import geometries in bodies (`PartDesign::Body`)
- create LCS with the macro 'new_ModelLCS', and place them wherever you feel they're useful. Use the MapMode to attach the LCS
- save part (only saved parts can be used currently). If necessary, close and re-open the file
- repeat

Assembly

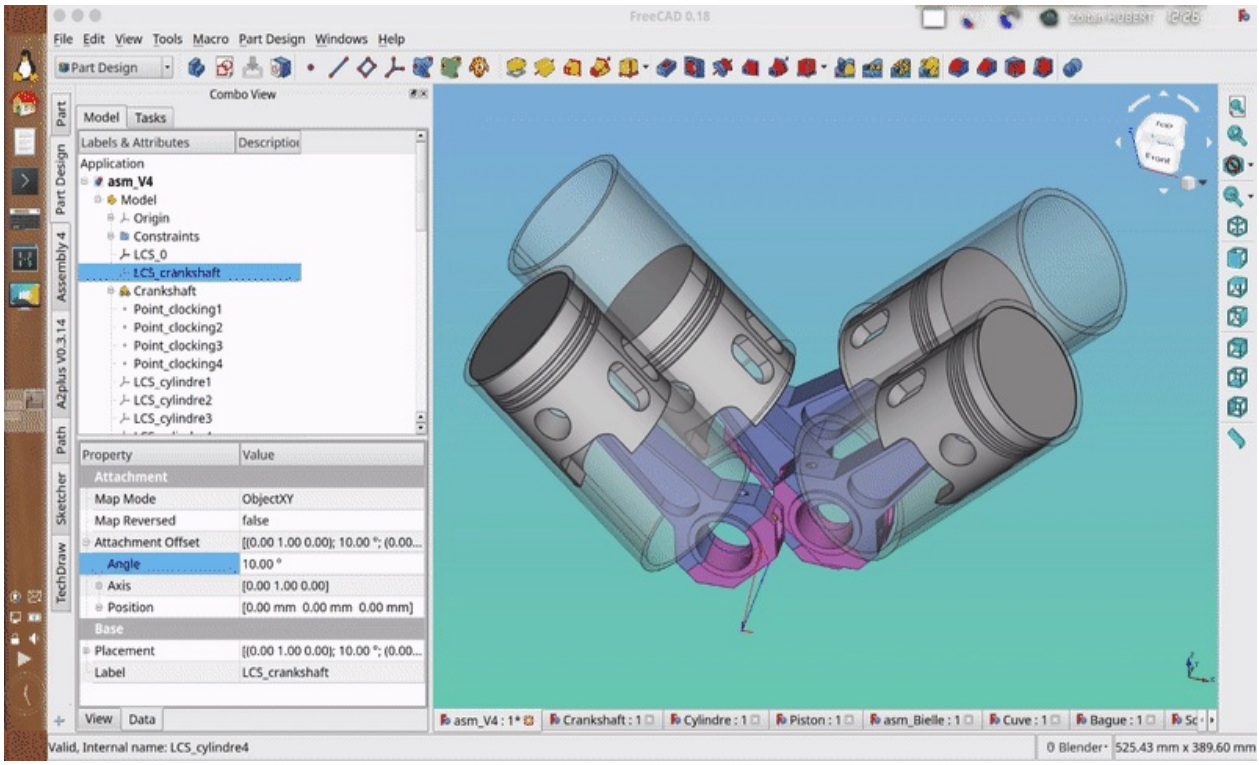
The basic workflow for creating a part is the following:

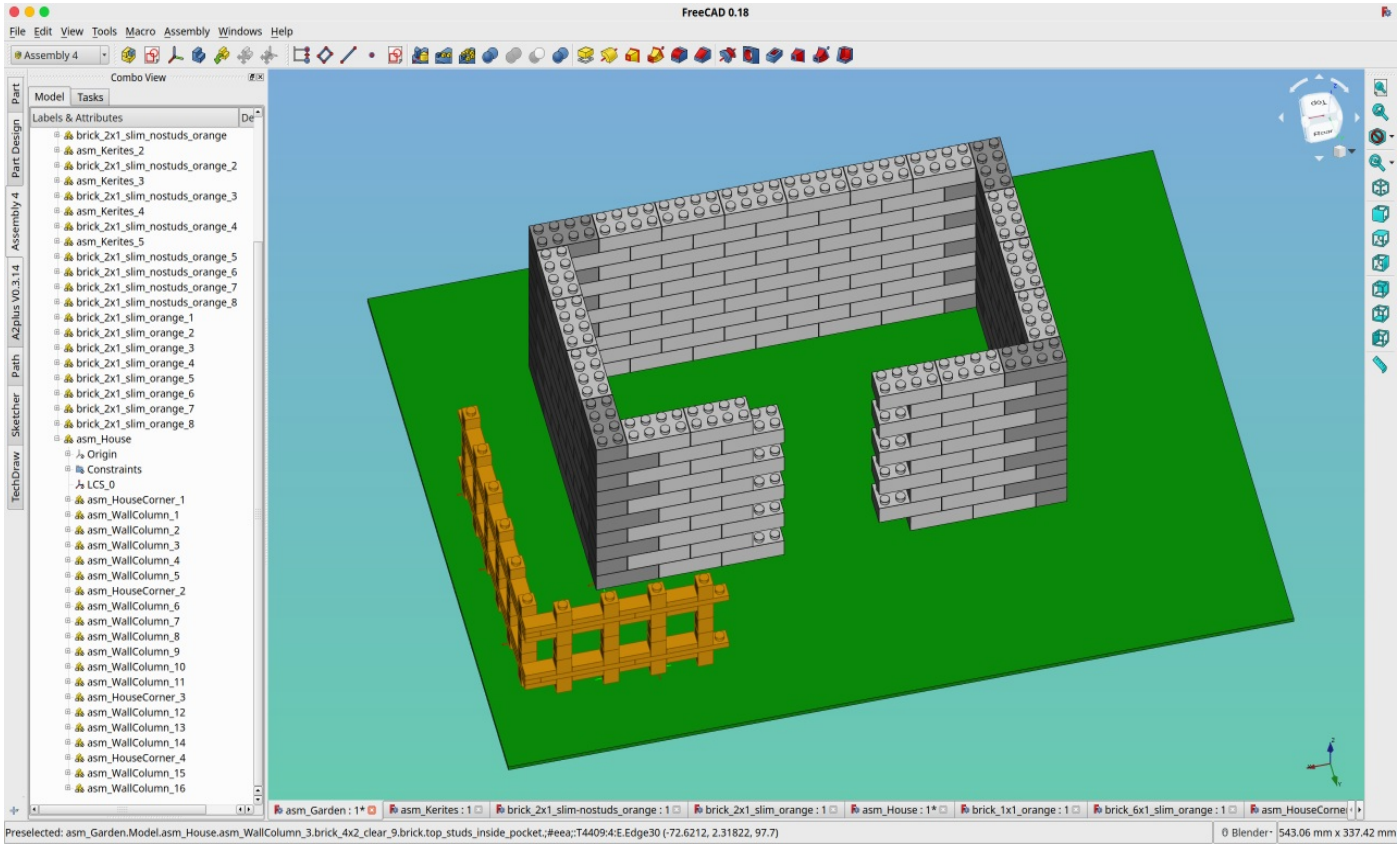
- create a new document, create a new 'Model'
- create a new sketch with the 'new_ModelSketch' macro. Attach it to whatever is useful, and draw the skeleton of the assembly, placing vertices and lines where useful
- create new LCS (with the new_ModelLCS macro) and place them on the correct vertices of the sketch (using MapMode)
- save document
- import a part using the 'link_Model' macro. It will place your part with its LCS_0 to the LCS_0 of the assembly. You should give it a recognisable name.
- select the link in the tree, and execute the macro 'place_Link'. Select the LCS of the part that you want to use as attachment point, select the part that you want it to be attached, and select the LCS in that part where you want the part to be attached. Click 'Apply'.
- if the placement corresponds to your needs, click 'Ok', else select other LCS / Part until satisfied
- if the part is correctly paced but badly oriented, select the corresponding constraint in the 'Constraints' folder, select the property 'Offset', and modify its parameters. Click 'Apply'. Repeat until satisfied

Nested assemblies

The previous method allows to assemble parts within a single level.
But this workbench also allows the assembly of assemblies: since there is no difference between parts and assemblies, the 'Insert External Part' allows to chose a part that has other parts linked to it. The only difference will be for the coordinate systems in the inserted assemblies: in order to be used with Assembly 4, a coordinate system must be directly in the root 'Model' container, meaning that a coordinate system inside a linked part cannot be used to attach the assembly to a higher-level assembly.

Therefore, in order to re-use a coordinate system of a part in an assembly, a coordinate system must be created at the root of the 'Model', and the placement of this coordinate system must be 'copied' over from the coordinate system that the user wants to use. This is done by inserting a coordinate system and using the 'Place LCS' command, which allows to select a linked part in the assembly and one of it's coordinate systems: the 2 coordinate systems — the one at the root of 'Model' and the one in the linked part — will always be superimposed, even if the linked part is modified, allowing the placement of the assembly in a higher level assembly using a linked part as reference. It sounds more complicated than it actually is.





Release notes

Release notes

- 2019.07.23 (**version 0.5.5**) :
  Fixed a bug in partLCSlist.findItems

- 2019.07.18 (**version 0.5.4**) :
  A cosmetic update to fix a 25 year old Windows bug: some UTF-8 characters in the comments were not accepted on some Windows 10 machines

- 2019.06.15 (**version 0.5.3**) :
  Now the LCS can be renamed, and they show up in the LCS list in the command placeLink as such. It's only visual, the ExpressionEngine still uses the LCS.Name though

- 2019.05.07 (**version 0.5.2**) :
  added insertDatumCmd

- 2019.03.18 (**version 0.5.1**) :
  Part can now be linked without being placed: this is then a raw interface with App::Link
  The instance can be moved manually with the 'Transform' dragger

- 2019.03.12 (**version 0.5**) :
  moved the actual code to Mod_Asm4

- 2019.03.11 (**version 0.4.1**) :
  Added placement of Datum Point

- 2019.03.09 (**version 0.4**) :
  FreeCAD now imports as App
  insert_Link launches place_Link

- 2019.03.05 (**version 0.3.1**) :
  added the RotX-Y-Z buttons

- 2019.02.20 (**version 0.3**)
  mostly working version

- 2019.02.18 (**version 0.1**) :
  initial release of Assembly 4 WB