

greysonbmiller-volunteer-test Docu- mentation

Release 1.0.0

© 2026, Greyson Miller

octubre 16, 2026

1	FreeCAD Macro	1
1.1	What it does.	1
1.2	How to run it	1
1.3	Source	2
2	Video	7
3	What I did in FreeCAD	9
4	What I did in kdenlive	11
5	Files	13
6	Music Attribution	15
7	Tips for Future Volunteers	17
7.1	Linux	17
7.2	FreeCAD	17
7.3	Music	17
7.4	kdenlive	18
7.5	Sphinx and GitHub Pages	18
7.6	PeerTube	18
8	License	19

Figura 1: Screenshot of the FreeCAD object: a 1 m x 1 m x 20 cm sheet of steel with my initials «GM» carved 5 cm deep 6

FreeCAD Macro

The model on the main page (a 1 m x 1 m x 20 cm steel sheet with the initials «GM» carved 5 cm deep) was built by this FreeCAD macro, `build_gm_sheet.FCMacro`. The macro ran inside the FreeCAD GUI while `recordmydesktop` recorded the screen, so every step appears in the video.

- Download: `build_gm_sheet.FCMacro`
- View on GitHub: [docs/build_gm_sheet.FCMacro](https://github.com/FreeCAD/FreeCAD/blob/master/Tools/build_gm_sheet.FCMacro)

1.1 What it does

1. Creates a new document and a PartDesign Body.
2. Opens a sketch on the `XY_Plane` in the **Sketcher** workbench and draws a fully constrained 1000 mm x 1000 mm square, one line at a time.
3. **Pads** the square 200 mm and sets the Body's material to *Steel*.
4. Opens a second sketch on the `XY_Plane`, offset 200 mm so it sits on the top face, and draws the outlines of block letters **G** and **M** (500 mm tall, 80 mm strokes).
5. **Pockets** the letters 50 mm deep.
6. Checks the result (valid solid, 1000 x 1000 x 200 mm bounding box, removed volume equals letter area x 50 mm), saves `gm-eco-libre-volunteer-test.FCStd`, and takes the screen-shot shown on the main page.

In the GUI it pauses between steps and opens the Pad and Pocket dialogs, so you can watch each step happen. Run headless, it skips the pauses.

1.2 How to run it

In the FreeCAD GUI, use **Macro -> Macros...**, choose the folder containing the macro, select it, and click **Execute**. You can also pass it on the command line (FreeCAD from Flathub shown):

```
# in the GUI
flatpak run org.freecad.FreeCAD build_gm_sheet.FCMacro

# headless
flatpak run --command=FreeCADCmd org.freecad.FreeCAD build_gm_sheet.FCMacro
```

It saves its output next to the macro, or in `$GM_OUT_DIR` if set. It was tested with FreeCAD 1.1.4.

Nota The lines that set the document's license metadata (CC BY-SA 4.0) and the default output folder

were added after the recording. The committed `.FCStd` had its license set to match.

1.3 Source

```
# Eco-Libre Volunteer Test: 1 m x 1 m x 20 cm metal sheet with the initials
# "GM" carved 5 cm deep using a Sketcher sketch + PartDesign Pocket.
#
# Run in the FreeCAD GUI (visible, paced for a screen recording):
#   freecad build_gm_sheet.FCMacro      (or Macro -> Macros... -> Execute)
# or headless:
#   FreeCADCmd build_gm_sheet.FCMacro
#
# Optional environment variables:
#   GM_OUT_DIR      where to save the .FCStd and screenshot (default: next to
this macro)
#   GM_START_TRIGGER wait until this file exists before modelling (to start a
screen recorder first)
#   GM_DONE_MARKER  file to create when finished
#
# SPDX-License-Identifier: GPL-3.0-or-later
import os
import time

import FreeCAD as App
import Part
import Sketcher

try:
    _HERE = os.path.dirname(os.path.abspath(__file__))
except NameError: # __file__ isn't always set when run from the Macro dialog
    _HERE = os.path.expanduser("~")
OUT_DIR = os.environ.get("GM_OUT_DIR", _HERE)
FCSTD = os.path.join(OUT_DIR, "gm-eco-libre-volunteer-test.FCStd")
RENDER = os.path.join(OUT_DIR, "gm-eco-libre-volunteer-test-render.png")
SCREENSHOT = os.path.join(OUT_DIR, "gm-eco-libre-volunteer-test-screenshot.png")
DONE_MARKER = os.environ.get("GM_DONE_MARKER", "")

GUI = App GuiUp
if GUI:
    import FreeCADGui as Gui
    from PySide import QtGui

def pause(seconds):
    """Let the GUI repaint for a while so each step is visible in the video."""
    if not GUI:
        return
    end = time.time() + seconds
    while time.time() < end:
        QtGui.QApplication.processEvents()
        time.sleep(0.02)

def fit(seconds=1.0):
    if GUI:
        Gui.SendMsgToActiveView("ViewFit")
        pause(seconds)

def steel_look(*objs):
```

```

    if not GUI:
        return
    look = App.Material(DiffuseColor=(0.72, 0.74, 0.77), AmbientColor=(0.35, 0.36,
0.38),
                        SpecularColor=(0.55, 0.55, 0.55), Shininess=0.35)

    for o in objs:
        o.ViewObject.ShapeAppearance = [look]

def oblique(view, spin=0):
    """Tilted front view that keeps the initials readable."""
    tilt = App.Rotation(App.Vector(1, 0, 0), 50)
    view.setCameraOrientation(App.Rotation(App.Vector(0, 0, 1),
spin).multiply(tilt))
    Gui.SendMsgToActiveView("ViewFit")

def add_polygon(sketch, pts, x0, y0, step_pause):
    """Add a closed polygon of line segments, constraining shared endpoints."""
    first = None
    prev = None
    for i in range(len(pts)):
        a = pts[i]
        b = pts[(i + 1) % len(pts)]
        gid = sketch.addGeometry(Part.LineSegment(
            App.Vector(x0 + a[0], y0 + a[1], 0), App.Vector(x0 + b[0], y0 + b[1],
0)), False)
        if prev is not None:
            sketch.addConstraint(Sketcher.Constraint("Coincident", prev, 2, gid,
1))
        else:
            first = gid
            prev = gid
        if GUI:
            App.ActiveDocument.recompute()
            pause(step_pause)
    sketch.addConstraint(Sketcher.Constraint("Coincident", prev, 2, first, 1))

# Block-letter outlines in mm, origin at each letter's lower-left corner.
# Letters are 500 mm tall with 80 mm strokes.
LETTER_G = [
    (0, 0), (350, 0), (350, 250), (175, 250), (175, 170), (270, 170),
    (270, 80), (80, 80), (80, 420), (350, 420), (350, 500), (0, 500),
]
LETTER_M = [
    (0, 0), (80, 0), (80, 330), (200, 180), (320, 330), (320, 0),
    (400, 0), (400, 500), (320, 500), (200, 330), (80, 500), (0, 500),
]

# Optionally wait (with FreeCAD already on screen) until the screen recorder
# has started and created this trigger file.
START_TRIGGER = os.environ.get("GM_START_TRIGGER", "")
if GUI:
    mw = Gui.getMainWindow()
    mw.showMaximized()
    while START_TRIGGER and not os.path.exists(START_TRIGGER):
        pause(0.2)

# --- New document -----
doc = App.newDocument("GM_metal_sheet")
doc.CreatedBy = "Greyson Miller"
doc.License = "Creative Commons Attribution-ShareAlike 4.0 International (CC BY-SA

```

```
4.0)"
doc.LicenseURL = "https://creativecommons.org/licenses/by-sa/4.0/"
if GUI:
    Gui.activateWorkbench("PartDesignWorkbench")
    pause(2.0)

body = doc.addObject("PartDesign::Body", "Body")
if GUI:
    Gui.ActiveDocument.ActiveView.setActiveObject("pdbody", body)
    Gui.ActiveDocument.ActiveView.viewIsometric()
pause(1.0)

xy_plane = body.Origin.OriginFeatures[3] # XY_Plane
assert xy_plane.Role == "XY_Plane", xy_plane.Role

# --- Sketch 1: 1000 x 1000 mm outline of the metal sheet -----
sheet_sketch = body.newObject("Sketcher::SketchObject", "SheetSketch")
sheet_sketch.AttachmentSupport = (xy_plane, [""])
sheet_sketch.MapMode = "FlatFace"
doc.recompute()
if GUI:
    Gui.ActiveDocument.setEdit(sheet_sketch) # switches to the Sketcher workbench
    Gui.ActiveDocument.ActiveView.viewTop()
    pause(1.5)

add_polygon(sheet_sketch, [(0, 0), (1000, 0), (1000, 1000), (0, 1000)], -500,
-500, 2.0)
# Fully constrain the square: horizontal/vertical edges, 1000 mm sides, centred.
sheet_sketch.addConstraint(Sketcher.Constraint("Horizontal", 0))
sheet_sketch.addConstraint(Sketcher.Constraint("Horizontal", 2))
sheet_sketch.addConstraint(Sketcher.Constraint("Vertical", 1))
sheet_sketch.addConstraint(Sketcher.Constraint("Vertical", 3))
sheet_sketch.addConstraint(Sketcher.Constraint("DistanceX", 0, 1, 0, 2, 1000.0))
sheet_sketch.addConstraint(Sketcher.Constraint("DistanceY", 1, 1, 1, 2, 1000.0))
sheet_sketch.addConstraint(Sketcher.Constraint("Symmetric", 0, 1, 1, 2, -1, 1))
doc.recompute()
fit(1.5)
if GUI:
    Gui.ActiveDocument.resetEdit()
pause(1.0)

# --- Pad 200 mm: the 1 m x 1 m x 20 cm sheet -----
pad = body.newObject("PartDesign::Pad", "Sheet")
pad.Profile = sheet_sketch
pad.Length = 200.0 # 20 cm
doc.recompute()
if GUI:
    Gui.ActiveDocument.setEdit(pad)
    Gui.ActiveDocument.ActiveView.viewIsometric()
    fit(4.0)
    Gui.ActiveDocument.resetEdit()
    sheet_sketch.Visibility = False

# Steel material on the body, so it is literally a sheet of metal.
try:
    import Materials
    mm = Materials.MaterialManager()
    steel = next(m for m in mm.Materials.values() if m.Name in ("Steel-Generic",
"Steel"))
    body.ShapeMaterial = steel
except Exception as e: # material library differs between FreeCAD versions
    App.Console.PrintWarning("Could not assign steel material: {}\n".format(e))
doc.recompute()
```

```

steel_look(body, pad)
if GUI:
    Gui.ActiveDocument.ActiveView.viewIsometric()
fit(2.5)

# --- Sketch 2: initials "GM" on the top face (z = 200 mm) -----
letters = body.newObject("Sketcher::SketchObject", "InitialsSketch")
letters.AttachmentSupport = (xy_plane, [""])
letters.MapMode = "FlatFace"
letters.AttachmentOffset = App.Placement(App.Vector(0, 0, 200), App.Rotation())
doc.recompute()
if GUI:
    Gui.ActiveDocument.setEdit(letters)
    Gui.ActiveDocument.ActiveView.viewTop()
    fit(1.5)

add_polygon(letters, LETTER_G, -420, -250, 1.5)
add_polygon(letters, LETTER_M, 30, -250, 1.5)
doc.recompute()
pause(1.5)
if GUI:
    Gui.ActiveDocument.resetEdit()
    pause(1.0)

# --- Pocket 50 mm: carve the initials 5 cm deep -----
pocket = body.newObject("PartDesign::Pocket", "CarvedInitials")
pocket.Profile = letters
pocket.Length = 50.0 # 5 cm
doc.recompute()
if GUI:
    Gui.ActiveDocument.setEdit(pocket)
    Gui.ActiveDocument.ActiveView.viewIsometric()
    fit(4.0)
    Gui.ActiveDocument.resetEdit()
    letters.Visibility = False
steel_look(body, pad, pocket)
if GUI:
    oblique(Gui.ActiveDocument.ActiveView)
    pause(2.0)

# --- Sanity checks -----
shape = body.Shape
bb = shape.BoundingBox
sheet_volume = 1000.0 * 1000.0 * 200.0
removed = sheet_volume - shape.Volume
App.Console.PrintMessage(
    "GM_RESULT valid={} bbox=({:.1f} x {:.1f} x {:.1f}) mm removed={:.0f} mm^3 "
    "pocket_depth={:.1f} mm\n".format(
        shape.isValid(), bb.XLength, bb.YLength, bb.ZLength, removed,
        pocket.Length.Value))

doc.saveAs(FCSTD)

if GUI:
    # Rotate a little so the 5 cm depth of the carving is visible, then capture.
    view = Gui.ActiveDocument.ActiveView
    Gui.Selection.clearSelection()
    oblique(view, spin=-20)
    pause(2.0)
    view.saveImage(RENDER, 1920, 1080, "White")
    shot = QtGui.QApplication.primaryScreen().grabWindow(mw.winId())
    shot.save(SCREENSHOT, "PNG")

```

```
pause(1.0)

# Slow orbit around the finished part to show the 5 cm deep carving.
for angle in range(-20, 341, 2):
    oblique(view, spin=angle)
    pause(0.05)
pause(3.0)

if DONE_MARKER:
    with open(DONE_MARKER, "w") as f:
        f.write("done\n")
```

This documentation is the final deliverable of the [Eco-Libre Volunteer Test](#), performed by Greyson Miller.

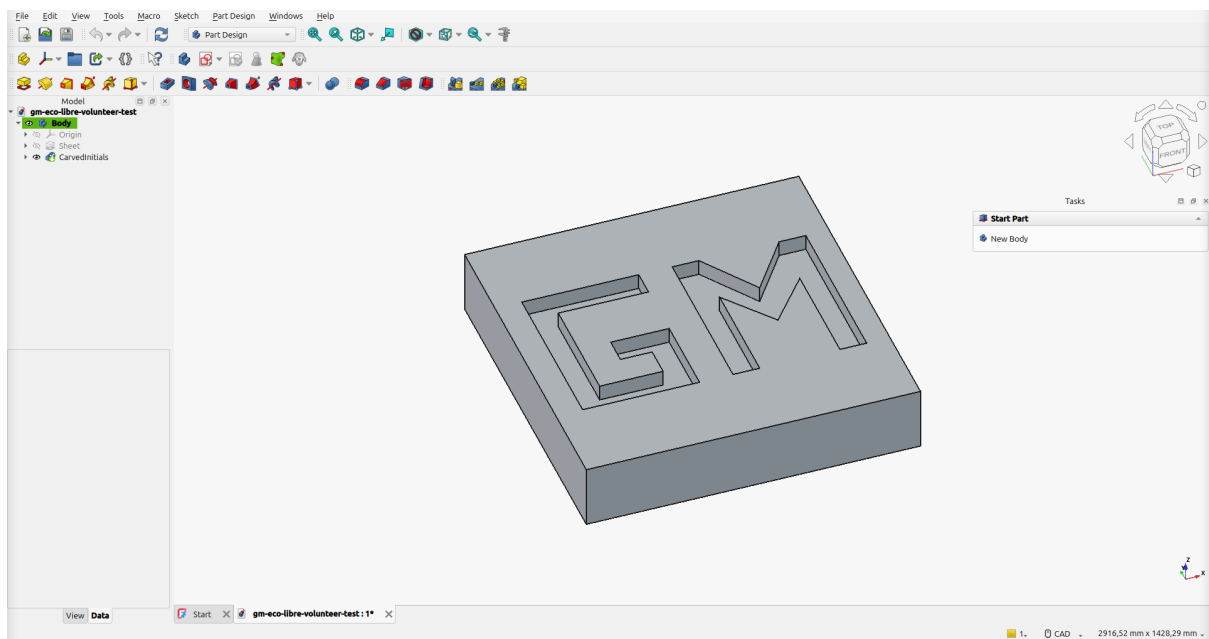


Figura 1. Screenshot of the FreeCAD object: a 1 m x 1 m x 20 cm sheet of steel with my initials «GM» carved 5 cm deep

The purpose of this test is to demonstrate experience with the tools needed to be successful as a volunteer with Eco-Libre: GNU/Linux, git, FreeCAD, recordmydesktop, kdenlive, PeerTube, and Sphinx. It also demonstrates familiarity with open-source licensing, specifically the [Creative Commons Attribution-ShareAlike 4.0 International](#) license, under which all of Eco-Libre's works are licensed.

Video

The video below shows the sheet of metal being created and the initials being carved in FreeCAD. It was recorded with `recordmydesktop`, sped up to 30 seconds in `kdenlive`, given a CC BY-SA background soundtrack, rendered in `kdenlive`, and uploaded to PeerTube under CC BY-SA.

Watch it on PeerTube: <https://peertube.watch/w/a8wNhiH1Hc4WimVL8e41Ck>

What I did in FreeCAD

The model was built in FreeCAD 1.1 (Flatpak) on Linux Mint 22.2, using the **PartDesign** workbench for the solid features and the **Sketcher** workbench for the sketches.

1. Created a new document and a PartDesign Body.
2. **Sheet outline.** Created a sketch on the `XY_Plane` and, in the Sketcher workbench, drew a square. I constrained it to be fully defined: horizontal/vertical edges, 1000 mm sides, and symmetric about the origin.
3. **Sheet of metal.** Closed the sketch and used **Pad** with a length of 200 mm, giving a 1 m x 1 m x 20 cm plate. I set the Body's material to *Steel* so it really is a sheet of metal.
4. **Initials.** Created a second sketch on the `XY_Plane`, offset 200 mm in Z (`AttachmentOffset`) so it sits on the top face of the plate. In the Sketcher workbench I drew the outlines of block letters **G** and **M** (500 mm tall, 80 mm strokes) as closed polylines with coincident constraints at every corner.
5. **Carving.** Closed the sketch and used **Pocket** with a length of 50 mm, carving the initials 5 cm deep into the metal.
6. **Checking the result.** The final solid is valid, its bounding box is 1000 x 1000 x 200 mm, and the volume removed is 11,140,000 mm³. That is exactly the area of the two letters (104,400 + 118,400 mm²) times the 50 mm depth.

The model in the video was built by running the FreeCAD macro `build_gm_sheet.FCMacro` in the FreeCAD GUI. It performs all of the steps above, and you can use it to reproduce the model. See [FreeCAD Macro](#) for the full source and how to run it.

What I did in kdenlive

1. Opened a kdenlive 26.08 project (1080p, 30 fps) with the `recordmydesktop` capture and the soundtrack in the Project Bin.
2. Put the 2 min 22 s recording on the video track, then used **Change Speed** on the clip and typed a duration of 00:00:30:00. kdenlive calculated the speed as 473.56%.
3. In the Clip Monitor, set the soundtrack's out point at 00:00:29:29 and used **Overwrite** to put those first 30 seconds on the audio track under the video.
4. Dragged the audio clip's fade handle to add a ~3 second fade-out at the end.
5. Rendered with the **MP4-H264/AAC** preset (*Render -> Render to File*), giving a 30 second, 1920 x 1080 video.

Files

All source files are in the `docs/` directory of this repo:

- `gm-eco-libre-volunteer-test.FCStd`: the FreeCAD model
- `build_gm_sheet.FCMacro`: the FreeCAD macro that builds the model ([source and instructions](#))
- `gm-recordmydesktop.ogv`: the source screen recording from `recordmydesktop`
- `gm-eco-libre-volunteer-test.kdenlive`: the kdenlive project
- `John_Harrison_-_Vivaldi_Spring_Mvt_1_Allegro.mp3`: the background soundtrack
- `gm-eco-libre-volunteer-test.mp4`: the final 30 second video rendered by kdenlive

Music Attribution

The background soundtrack is «[Spring, Mvt. 1: Allegro](#)» from Antonio Vivaldi's *The Four Seasons*, performed by **John Harrison with the Wichita State University Chamber Players**, via the [Free Music Archive](#) ([archive.org mirror](#)). The recording is licensed under [CC BY-SA 3.0](#), which is compatible with the CC BY-SA 4.0 license of the video. The composition itself is in the public domain.

Tips for Future Volunteers

This section documents what I learned while completing the Eco-Libre Volunteer Test, to help the next person.

7.1 Linux

I used Linux Mint 22.2 (Cinnamon), which is based on Ubuntu 24.04. Like Michael Altfield found with Linux Mint, `freecad` is **not** available in its apt repos. Instead of switching distros, I installed FreeCAD and kdenlive from [Flathub](#), which also gives much newer versions than apt (FreeCAD 1.1 and kdenlive 26.08 instead of kdenlive 23.08):

```
flatpak install --user flathub org.freecad.FreeCAD org.kde.kdenlive
sudo apt install recordmydesktop python3-sphinx python3-sphinx-rtd-theme
python3-git inkscape gimp
```

Flatpak apps can't see files in `/tmp`, so keep your work somewhere in your home directory.

7.2 FreeCAD

- Draw letters as **closed outlines**, not single strokes. Pocket needs closed wires that don't cross themselves. Block letters made from straight segments are the easiest.
- Put the initials sketch on the **top face** of the plate (or on the XY plane offset by the plate thickness). A Pocket cuts in the direction opposite the sketch normal, down into the metal.
- In FreeCAD 1.x the sketch attachment property is called `AttachmentSupport` (it was `Support` in 0.x), which matters if you follow older tutorials or Python examples.

7.3 Music

Only CC0, CC BY, and CC BY-SA music can be used in a CC BY-SA video. **NC (NonCommercial) and ND (NoDerivatives) licenses are not compatible.**

Check the license in more than one place. The track I first picked was listed as CC BY-SA on both its Free Music Archive page and archive.org, but the copyright tag inside the MP3 said *Attribution-Non-commercial-Share Alike*. I switched to a track where the album page, archive.org, and the MP3 tag all agree. You can read the tag with:

```
ffprobe -v error -show_entries format_tags=copyright -of default=nw=1 song.mp3
```

7.4 kdenlive

- **Change Speed** (right-click the clip in the timeline) lets you type the duration you want, and kdenlive calculates the speed. That's easier than guessing a percentage.
- The *Overwrite* (B) and *Insert* (V) shortcuts edit **every active track**. Turn off the video track's target (click its V1 label so it goes grey) before overwriting an audio-only clip. Otherwise kdenlive blanks the video track under it.
- Home jumps to the start of the *clip*. Ctrl+Home jumps to the start of the *project*.

7.5 Sphinx and GitHub Pages

I forked [Michael Altfeld's volunteer test repo](#), which already has a working Sphinx + Read the Docs theme + GitHub Actions setup (see [this article](#)). After forking:

1. **Settings -> Actions -> General -> Workflow permissions**: choose «Read and write permissions», so the CI can push the gh-pages branch.
2. **Actions tab**: GitHub disables workflows on forks. Click «I understand my workflows, go ahead and enable them».
3. **Settings -> Pages**: deploy from the gh-pages branch, / (root).
4. In docs/conf.py, change github_user to your username and github_version to main/docs/ so the «Edit on GitHub» link works.

You can also build the site locally before pushing: `make -C docs html` and then open docs/_build/html/index.html.

Two more things I ran into:

- On my fork, pushes didn't start the workflow, but running it manually did. I added `workflow_dispatch` to `.github/workflows/docs_pages_workflow.yml`, so you can start a build from the Actions tab with **Run workflow** (or `gh workflow run docs_pages_workflow.yml`).
- The pdf download link 404'd, in the template too. Debian bookworm's pip refuses to install packages system-wide (PEP 668), so rinohtype was never installed. Simply adding `--break-system-packages` made pip upgrade Sphinx as well, which broke the HTML build. The fix in docs/buildDocs.sh installs rinohtype constrained to the versions apt already installed.

7.6 PeerTube

Most PeerTube instances require an admin to approve new accounts. The [JoinPeerTube instance list](#) lets you filter for instances with open registration. The video embed above uses the «Embed» code from PeerTube's *Share* dialog, passed through a [raw directive](#).

License

This site and the media in this repo are licensed [CC BY-SA 4.0](#) (code is GPLv3). It is based on [Michael Altfield's Eco-Libre Volunteer Test](#), also CC BY-SA.

