Using Rhinoceros and Python to produce 3D printed mathematical models - workshop notes

Henry Segerman

segerman@math.okstate.edu

Abstract

This workshop will be a hands-on introduction to producing physical 3D printed mathematical models using computer software. 3D printing is an accessible way to produce physical objects, for use in outreach, teaching or research. In addition to being excellent visualisation aids, physical objects go further, allowing for a tactile understanding. We will use the 3D design program "Rhinoceros", and the Python scripting interface to Rhinoceros to produce 3D files ready to be sent to a 3D printer. These notes are written for the Mac version of the software. The Windows version is very similar – substitute the Ctrl key instead of the Command key. The materials from this workshop are available at `http://math.okstate.edu/people/segerman/talks/3d_printing_workshop_files.zip`.

1 3D printing file format

One of the most common formats for sending a model to a 3D printer is the OBJ format. This consists of a list of vertex positions in $\mathbb{R}^3$, and a list of faces, each of which is a triangle or quadrilateral. Each face is given by a set of three or four indices into the list of vertices. The triangles and quadrilaterals make up a *mesh*. Current 3D printers take such a mesh, and interpret it as a closed orientable embedded surface in $\mathbb{R}^3$, from which they can determine which voxels (3-dimensional analogues of pixels) are inside the surface (and so should be solid, filled with plastic or whatever material the printer is using), and which are outside the surface (and so should be empty).

2 Rhinoceros

To get Rhinoceros, go to `http://www.rhino3d.com/download/` for a 90 day trial version, available for both Mac and Windows. Educational discounts are available for the full versions of the software, costing around $200.

2.1 Navigating the viewpoints

- Open the Rhinoceros program and start a new model. Four viewpoints are shown: three orthogonal views, the top, front and right views, and a perspective view.
- On a Mac with a touchpad, on any window: pinch with two fingers to zoom, or drag with two fingers to rotate the perspective view, or Shift+drag with two fingers to translate.
- On both Windows and Mac with a mouse: Shift+hold right click and drag on any of the windows to translate that view. Shift+Ctrl+hold left click does the same thing. Scroll wheel on any of the windows zooms the view. Hold right click and drag to rotate the perspective view. Ctrl+hold left click does the same thing.

2.2 Building an object: right angle tetrahedron

- File → New or Command+N starts a new file.
- We will be building the edges of a tetrahedron with vertices at $(0,0,0), (20,0,0), (0,20,0), (0,0,20)$.
- If it isn't already on, turn on snap to grid, using the button "Grid Snap" just above the top left viewpoint window (near the bottom of the window on Windows).
- Build the three lines along the axes first, using the orthogonal viewpoints. Use Curve → Line → Single Line, then left click on one of the viewpoints at the start and endpoints to define the line. (Space bar repeats the last used command, which may be useful here.)
- Now turn on Object Snap, using the button "OSnap" at the top left of the screen (again, below in Windows), then check the "End" box in the "Object Snaps" window that appears.
- For the other three lines, use the perspective view. When drawing the new lines, the mouse will snap to the endpoints of the existing lines.
- To correct an error, either Edit → Undo, or Command+Z to undo, or alternatively left click to select an object and Backspace to delete. Left click and drag a box to select multiple objects. Shift+left click adds objects to the selection and Command+left click removes objects from the selection.
- Other things to try: Transform → Move, Transform → Rotate, Transform → Scale → Scale 3-D. Follow the instructions in the pop-up window that appears.

We now need to thicken these lines up so that we can print the tetrahedron.

- Edit → Select Objects → All Objects or Command+A to select all of the lines.
- Solid → Pipe then Enter to make pipes around the lines (of radius 1, this can be altered in the dialog at the top left of the window).
- View → Wireframe, or Shaded, Rendered etc. changes the view style. Shaded is better for seeing surfaces than Wireframe.
- (Optional) use Solid → Sphere → Center, Radius to put a sphere at each of the vertices of the tetrahedron, to round off the corners.
- Select all with Command+A.
- Use Solid → Union to boolean union the pipes and spheres together into a single, non-overlapping object.

Rhino uses *NURBS surfaces* as well as meshes to describe surfaces. ("NURBS" stands for "Non-uniform rational B-spline". NURBS surfaces are generalisations of Bézier curves.) At the moment we have a *polysurface* made from NURBS surface patches, but we need to convert it to a mesh before sending the data to a 3D printer.

- Select all with Command+A.
- Do Mesh → From NURBS Object, then Enter to convert the polysurface to a mesh.
- The mesh and the polysurface now occupy the same space. To see only the mesh, select it, and then either move it away from the polysurfaces, or put it on a different *layer*. For either strategy, use Edit → Select Objects → Polygon Meshes (or similar) to easily select one kind of object.

- To move selected objects, either use Transform → Move, or turn on Gumball, using the button "Gumball" at the top of the screen, then use the handles that appear when you select an object.
- To change the layer of a selected object, first do Window → Show Layers Panel, unless the panel is already open. Then right click on the desired layer in the layers panel, and choose "Move objects to this layer". Toggling the lightbulb icon changes the visibility of objects on that layer.
- To export the mesh as an OBJ file, select only the mesh, then File → Export Selected. Change the file format at the bottom of the save dialog to OBJ. Choose a filename, and then click Export.

3 Rhinoceros and Python

3.1 Make a parametric curve

- Command+N to start a new file.
- To run a python script, type "RunPythonScript" (just start typing, it starts writing in the "Command" box in the top left; also the command should autocomplete after you type a few letters) then Enter. Choose "parametric_curve_example.py". This will draw one turn of a helix.
- Open "parametric_curve_example.py" using a text editor to see the code. (The Windows version of Rhino has its own built in script editor.)
- The function "rhino.AddInterpCurve" in the script is one of many functions used to interface with Rhinoceros, see `https://developer.rhino3d.com/api/RhinoScriptSyntax/` for documentation on other functions.
- You can alter the parametric function in the script. See `https://docs.python.org/2/library/math.html` for a list of basic mathematical functions included in the Python math library. There are many other libraries and modules available online that can be used for more complicated functions.
- To convert a curve into something that can be printed, Pipe then Mesh it as before (using Union if there are multiple curves involved).
- If a NURBS object self-intersects (e.g. if you Pipe a self-intersecting curve), then Union will not do anything, and 3D printers may have trouble printing the resulting mesh. To solve this kind of problem, subdivide the curve into non-intersecting pieces before applying Pipe. Use the Round Cap option in the Pipe dialog – this will make the resulting NURBS objects overlap transversely, so that Union will work.

3.2 Make a parametric surface

- Use RunPythonScript to run "parametric_surface_example.py". This draws part of a hyperbolic parabola. Again you can open the file with a text editor to see and edit the code.
- To thicken the surface up to make it printable, use Solid → Extrude Surface. This will make a closed NURBS object, made out of six NURBS patches. In contrast, applying Pipe to a curve will usually produce a closed NURBS object made out of three NURBS patches: the two end caps and the tube around the curve.

3.3 Joining and exploding NURBS patches

- Delete the thickened hyperbolic parabola (or move it to a different layer). You should still have the two-dimensional hyperbolic parabola surface. Use Transform → Mirror to reflect the surface across the line $x = 10$, making a copy of it (you have to draw the mirror plane, use the top viewpoint to do this).

- Now select both NURBS surface patches and Edit → Join. This joins the two surfaces together along the edge they meet at. My usual strategy is to build complicated shapes up out of NURBS in this way. You may need to change the viewpoint from "Wireframe" to "Shaded" to see the surface properly.

- With the joined surface selected, do Surface → Edge Tools → Show Edges, then click the "Naked Edges" button in the pop-up window. This highlights the boundary ("naked") edges of the surface. To make a surface that we can turn into a mesh and then print, all edges or surface patches must be matched up properly to give a closed surface.

- With the joined surface selected, do Edit → Explode. This is the opposite of Join, and you should see that the edges between the two surface patches are now naked.

- When you Mesh a surface made from NURBS patches that are properly joined together, Rhino produces a mesh that is connected across the join. Without proper joins, the mesh surface will think it has non-empty boundary, and the 3D printer will not know what is inside and outside of the mesh. To check the status of a mesh, do Edit → Show Object Properties Panel, and then select the mesh. In the Details tab, under the Geometry heading, it will tell you if the mesh is open (with non-empty boundary) or closed. Surface → Edge Tools → Show Edges also works for meshes, to tell you where the mesh has boundary.

3.4 Make a parametric solid

- The final example script in these notes takes input of an existing mesh. We will make a simple mesh before proceeding.

- Use Mesh → Polygon Mesh Primitives → Box to draw a cuboid in the first octant of space, slightly away from the coordinate planes. Set the "X Faces", "Y Faces" and "Z Faces" variables to 1, so that we get only one mesh rectangle on each face. You may want to build the box on the x-y plane, then use the Gumball tool to lift it up off of the x-y plane.

- Use RunPythonScript to run "parametric_transform_mesh.py". It will ask you to select a mesh (unless you already have your mesh selected when you run the script), and a scaling factor (you can hit Enter to accept the default value). The script should generate the image of your mesh under the transformation from spherical to cartesian coordinates, outputting a closed NURBS object as the result. Each face of the mesh is transformed into a NURBS patch in the result.

- The last operation that the script performs is to join its generated surface patches together. This may fail if any of the faces of the input mesh are on a coordinate plane (in the case of conversion between spherical and cartesian coordinates), since there may be degenerate NURBS patches in the output.

- Try building other collections of meshes to use as input for the script, or changing the parametric transformation function in the script. Note that the script may run slowly if there are a large number of mesh faces.