Embedding some `misc3d` figures in a PDF as interactive 3D objects

Luke Tierney
Statistics and Actuarial Science
University of Iowa

September 27, 2009

Recent versions of PDF and of the Acrobat reader support inclusion of 3D formats for representing interactive 3D models. This note describes how this support can be used to include interactive versions of 3D contour plots and other triangular mesh scenes produced by the `misc3d` R package. There are two file formats supported called U3D and PRC. Both are complex binary formats, so it is useful to find simple textual formats for representing 3D objects along with software for converting these to U3D or PRC. So far I have found two approaches, one that captures only the geometry and one that captures geometry along with color and transparency.

1 Geometry-only images using MeshLab and U3D.

MeshLab is an open source system for the processing and editing of unstructured 3D triangular meshes. It can read and write several formats representing such meshes. An easy format to write that MeshLab can read is the OFF format. A function to write out `misc3d` scenes in OFF format is shown in Appendix A. This function does not attempt to write out color, transparency or material properties. Color and transparency are supported in OFF format but MeshLab doesn't seem to write them properly into the U3D (or read them from OFF for that matter). The OFF file produced by this function can then be read in by MeshLab and then exported in U3D format. The `movie15` LaTeX package can then be used to include the image in a PDF document created with `pdflatex`.

Figures 1 and 2 show two images created by this approach, one shows nested contours of the density a mixture of three-dimensional normals, the other shows the volcano surface from the R distribution. The normal mixture contours were created with a variant of code from the `contour3d` help page:

```
nmix3 <- function(x, y, z, m, s) {
    0.4 * dnorm(x, m, s) * dnorm(y, m, s) * dnorm(z, m, s) +
    0.3 * dnorm(x, -m, s) * dnorm(y, -m, s) * dnorm(z, -m, s) +
    0.3 * dnorm(x, m, s) * dnorm(y, -1.5 * m, s) * dnorm(z, m, s)
}

f <- function(x,y,z) nmix3(x,y,z,.5,.5)

gs1 <- function(n = 40, k = 5, cmap = heat.colors, ...) {
    th <- seq(0.05, 0.2, len = k)
    col <- rev(cmap(length(th)))
    x <- seq(-2, 2, len=n)
    m <- function(x,y,z) x > .25 | y < -.3
    contour3d(f,th,x,x,x,color=col, mask = m, engine = "none",
              scale = FALSE, ...)
}
```

(nmix.u3d)

Figure 1: Embedded U3D image showing nested contours of the density of a mixture of three tri-variate normal distributions.

.

```
conts <- gs1(40, 5, screen=list(z = 130, x = -80),
             color2 = "lightgray", cmap=rainbow)
saveTrianglesAsOFF(conts,"nmix.OFF")
```

The volcano example was created with

```
z <- 2 * volcano
x <- 10 * (1:nrow(z))
y <- 10 * (1:ncol(z))
vtri <- surfaceTriangles(x, y, z, color="green3")
saveTrianglesAsOFF(vtri,"volcano.OFF")
```

2 Images including color and transparency using Asymptote

Asymptote is a powerful vector graphics language for 2D and 3D graphics. It is inspired by MetaPost and generalizes MetaPost path construction algorithms to three dimensions. It is possible to write out an asymptote program to represent a triangular mesh scene produced by `misc3d`. A function to do this is given in Appendix B. Color and transparency are supported, though I have not yet figured out whether it is possible to have different colors on the two sides of a facet, or how to specify other material properties—it may well be possible. As an example, the code

```
vtri <- surfaceTriangles(x, y, z, color="green3")
```

(nmix.u3d)

Figure 2: Embedded U3D image of the R volcano surface data.

.

```
cvtri <- updateTriangles(vtri, color = function(x,y,z) {
    cols <- terrain.colors(diff(range(z)))
    cols[z - min(z) + 1]
})
saveTrianglesAsASY(cvtri,"volcano.asy")
```

adds coloring to the volcano surface. The command

```
asy -inlineimage -tex pdflatex volcano.asy
```

then produces a PRC file that can be included in a LaTeX document. The result is shown in Figure 3 The code produced seems to require a fairly recent version of Asymptote; I used version 1.86. Embedding in LaTeX also requires a PCR-aware version of the `movie15` package; such a version is included in the Asymptote distribution.

Color can also be used to help distinguish multiple contour surfaces as shown in Figure 4.

Transparency can also be sued to show multiple contour surfaces. This is illustrated in Figure 5. The additional code to produce this figure is

```
g1 <- function(n = 40, k = 5, alo = 0.1, ahi = 0.5, cmap = heat.colors) {
    th <- seq(0.05, 0.2, len = k)
    col <- rev(cmap(length(th)))
    al <- seq(alo, ahi, len = length(th))
    x <- seq(-2, 2, len=n)
    contour3d(f,th,x,x,x,color=col,alpha=al, engine = "none")
}
```

(vol.prc)

Figure 3: Color version of the volcano surface

```
aconts <- g1(40,5)
saveTrianglesAsASY(aconts, "nmix-alpha.asy")
```

As final example, Figure 6 shows the contour of a brain from an MR image along with several contours of the activation level measures in a PET experiment. To keep the file size and performance reasonable the brain image data needed to be reduced in resolution by one half in each dimension.

The PCR files produced by this approach are huge. There may be some way to reduce them but I am not yet aware of any such approach. Possibly because of the file sizes the images do not immediately load when a page is shown in the reader. Also I have observed a few crashes of the reader when the file includes a number of large embedded 3D images.

3 Other approaches

One other option using Open Source tools it so write out the objects in JVX format (an XML variant) and convert with jReality, but that also doesn't seem to write out colors for U3D properly.

The Adobe 3D software available as part of Acrobat Pro Extended may be able to read OFF files and/or produce smaller PCR files. I am looking into this.

A Writing OFF Files

```
## Write out a triangle mesh scene as an OFF format file.  This only
## outputs the geometry; color and transparency are ignored for now.
## The format supports adding three or four rgb values to each face
```

(nmix-color.prc)

Figure 4: Color version of nested density contours.

```
## line, but MeshLab seems to ignore these.

saveTrianglesAsOFF <- function(scene, filename = "scene.OFF") {
    scene <- misc3d:::colorScene(scene)
    triangles <- misc3d:::canonicalizeAndMergeScene(scene, "color", "color2",
                                                    "alpha", "col.mesh",
                                                    "fill", "smooth")
    ve <- misc3d:::t2ve(triangles)
    f <- file(filename, open = "w")
    on.exit(close(f))
    write("OFF", f)
    write(c(ncol(ve$vb), ncol(ve$ib), 3 * ncol(ve$ib)), f, 3)
    write(ve$vb, f, 3)
    write(rbind(3, ve$ib - 1), f, 4)
    invisible(NULL)
}
```

B Writing Asymptote Files

```
## write an asymptote program for recreating a triangular mesh
## scene. Color and transparency are supported; mesh drawing, color2,
## and material properties are not currently supported. The loops
## could be vectorized but seem adequate for now.
```

(nmix-alpha.prc)

Figure 5: Nested density contours with colors and transparency.

```
saveTrianglesAsASY <- function(scene, filename = "scene.asy") {
    scene <- misc3d:::colorScene(scene)
    triangles <- misc3d:::canonicalizeAndMergeScene(scene, "color",
                                                    "color2", "alpha",
                                                    "col.mesh", "fill",
                                                    "smooth")
    ve <- misc3d:::t2ve(triangles)
    f <- file(filename, open = "w")
    on.exit(close(f))

    ## write out header information and vertices
    cat("//generated by saveTrianglesAsASY\n\n",
        "import three;\n\n",
        "size(20cm);\n\n",
        "//currentprojection=perspective(250,-250,250);\n",
        "currentlight=Viewport;\n\n",
        "typedef path3[] trimesh;\n\n",
        "// Vertices\n",
        "triple[] V;\n",
        sep = "", file = f)

    nv <- ncol(ve$vb)
    x <- ve$vb[1,]
```

(brain.prc)

Figure 6: MRI contour of a brain along with several contours of the activation level in a PET experiment.

```
y <- ve$vb[2,]
z <- ve$vb[3,]
for (i in 1 : nv)
    cat(sprintf("V[%d] = (%f, %f, %f);\n",
                i - 1, x[i], y[i], z[i]), file = f)

## write out the faces
cat("\n",
    "guide3 triface_(int i, int j, int k) {\n",
    "  guide3 gh; gh=V[i-1]--V[j-1]--V[k-1]--cycle;\n",
    "  return gh;\n",
    "};\n\n",
    "// Faces\n",
    "trimesh F;\n",
    sep = "", file = f)

nf <- ncol(ve$ib)
v1 <- ve$ib[1,]
v2 <- ve$ib[2,]
v3 <- ve$ib[3,]
for (i in 1 : nf)
    cat(sprintf("F[%d] = triface_(%d, %d, %d);\n",
                i - 1, v1[i], v2[i], v3[i]), file = f)
```

```
    ## write out color and transparency values
    cat("\n",
        "// Colors\n",
        "material M[];\n",
        sep = "", file = f)

    cols <- col2rgb(triangles$color)
    alpha <- triangles$alpha
    r <- cols[1,]
    g <- cols[2,]
    b <- cols[3,]
    if (any(alpha < 1))
        for (i in 1 : nf)
            cat(sprintf("M[%d] = rgb(%f, %f, %f) + opacity(%f);\n",
                        i - 1, r[i], g[i], b[i], alpha[i]),
                file = f)
    else
        for (i in 1 : nf)
            cat(sprintf("M[%d] = rgb(%f, %f, %f);\n",
                        i - 1, r[i], g[i], b[i]),
                file = f)

    cat("\ndraw(surface(F), M);\n", file = f)
    invisible(NULL)
}
```