In [5]:

```
using SparseArrays

# A data structure to store 3d-dose matrices
struct Dose3D
    s::String
    r::String
    nPlanes::Int
    nRows::Int
    nCols::Int
    planeOffsets::Array{Float32,1}
    rowSpacing::Int
    colSpacing::Int
    origin::Tuple{Float32,Float32,Float32}
    doses::SparseVector
end
```

In [6]:

```
# Reads all data files in the directory 'dat/RTDose' and constructs an array of dose struct
# phuonglh, December 24, 2019
function read(path::String)
    xs = Array{Dose3D, 1}()
    foreach(readdir(path)) do fileName
        m = match(r"(s\d{3})(rot\d{3})", fileName)
        if (m != nothing)
            println(fileName)
            s, t = m.captures
            f = open(string(path, fileName))
            lines = readlines(f)
            close(f)
            nPlanes = parse(Int, split(lines[1], ":")[2])
            nRows = parse(Int, split(lines[2], ":")[2])
            nCols = parse(Int, split(lines[3], ":")[2])
            po = match(r"\[(.+)\]", lines[4])
            planeOffsets = map(x -> parse(Float32, x), split(po.captures[1], ","))
            rowSpacing = parse(Int, split(lines[5], ":")[2])
            colSpacing = parse(Int, split(lines[6], ":")[2])
            oo = match(r"\((.+)\)", lines[7])
            parts = split(oo.captures[1], ",")
            origin = parse.(Float32, (strip(parts[1]), strip(parts[2]), strip(parts[3])))
            doseString = strip(split(lines[9], ":")[2])
            ds = map(x -> parse(Float32, x), split(doseString, " "))
            doses = sparsevec(ds)
            push!(xs, Dose3D(s, t, nPlanes, nRows, nCols, planeOffsets, rowSpacing, colSpac
        end
    end
    xs
end
```

Out[6]:

```
read (generic function with 1 method)
```

In [7]:

```
# Reads a file in the directory 'dat/RTDose' and constructs a dose structure.
# dunghn
function readFile(shotPosition::String, rotation::String)
    fileNamePrefix = "dat/RTDose2/RTDose_ImatrexPRTv2_16cmCyl_d80mm_s";
    fileNamePostfix = ".dcm.txt"
    fileName = fileNamePrefix * shotPosition * "rot" * rotation * fileNamePostfix
    println(fileName)
    f = open(fileName)
    lines = readlines(f)
    close(f)
    nPlanes = parse(Int, split(lines[1], ":")[2])
    nRows = parse(Int, split(lines[2], ":")[2])
    nCols = parse(Int, split(lines[3], ":")[2])
    po = match(r"\[(.+)\]", lines[4])
    planeOffsets = map(x -> parse(Float32, x), split(po.captures[1], ","))
    rowSpacing = parse(Int, split(lines[5], ":")[2])
    colSpacing = parse(Int, split(lines[6], ":")[2])
    oo = match(r"\((.+)\)", lines[7])
    parts = split(oo.captures[1], ",")
    origin = parse.(Float32, (strip(parts[1]), strip(parts[2]), strip(parts[3])))
    doseString = strip(split(lines[9], ":")[2])
    ds = map(x -> parse(Float32, x), split(doseString, " "))
    doses = sparsevec(ds)
    return Dose3D(shotPosition, rotation, nPlanes, nRows, nCols, planeOffsets, rowSpacing,
end
```

Out[7]:

```
readFile (generic function with 1 method)
```

In [8]:

```
#xs = @time read("dat/RTDose/")
#println("Number of data files = ", length(xs))
##println(xs[100])
#println(length(xs[100].doses))
#println(nnz(xs[100].doses))
#println(nnz(xs[200].doses))
```

In [9]:

```
# (plane, row, col)
function coordinate2index(dose3d::Dose3D,i::Int,j::Int,k::Int)
    return i + j*dose3d.nPlanes + k*dose3d.nRows*dose3d.nPlanes
end
```

Out[9]:

```
coordinate2index (generic function with 1 method)
```

In [10]:

```
function index2coordinate(dose3d::Dose3D, i::Int)
    col = div(i, dose3d.nRows*dose3d.nPlanes)
    i -= col*dose3d.nRows*dose3d.nPlanes
    row = div(i,dose3d.nPlanes)
    plane = i - row*dose3d.nPlanes
    return (plane,row, col)
end
```

Out[10]:

```
index2coordinate (generic function with 1 method)
```

In [11]:

```
numCols = 80
numRows = 80
numPlanes = 80
cs = collect(Int, range(0, numCols - 1, step=1))
rs = collect(Int, range(0, numRows - 1, step=1))
ps = collect(Int, range(0, numPlanes - 1, step=1))
positions = collect(Iterators.product(ps, rs, cs))
println("#(positions) = ", length(positions))
```

```
#(positions) = 512000
```

In [12]:

```
using Plots

cmap = Plots.ColorGradient(:inferno)
cmap.colors[1] = RGBA(0.,0.0,0.0,0.0)
```

Out[12]:

```
RGBA{Float64}(0.0,0.0,0.0,0.0)
```

In [13]:

```
function plotSurface(dose3d::Dose3D, planeIndex::Int, colorMap)
    filteredPositions = filter(p -> p[1] == planeIndex, positions)
    values = map(t -> dose3d.doses[coordinate2index(dose3d, t[1], t[2], t[3])+1], filteredP
    u = map(x -> x[2], filteredPositions)
    v = map(x -> x[3], filteredPositions)
    surface(u, v, values, color=colorMap)
end
```

Out[13]:

```
plotSurface (generic function with 1 method)
```

In [14]:

```
function plotContourFixPlane(dose3d::Dose3D, planeIndex::Int, filled=true)
    filteredPositions = filter(p -> p[1] == planeIndex, positions)
    f(c, r) = dose3d.doses[coordinate2index(dose3d, planeIndex, r, c)+1]
    c = sort(unique(map(x -> x[3], filteredPositions)))
    r = sort(unique(map(x -> x[2], filteredPositions)))
    p = contour(c, r, f, fill=filled, aspect_ratio=1, xlims = (0,90), xticks = 0:10:90, xla
                ylims = (0,90), yticks = 0:10:90, ylabel = "rows", title="layer="*string(pl
    plot(p)
end
```

Out[14]:

```
plotContourFixPlane (generic function with 2 methods)
```

In [15]:

```
function plotContourFixRow(dose3d::Dose3D, rowIndex::Int, filled=true)
    filteredPositions = filter(p -> p[2] == rowIndex, positions)
    f(c, p) = dose3d.doses[coordinate2index(dose3d, p, rowIndex, c)+1]
    p = sort(unique(map(x -> x[1], filteredPositions)))
    c = sort(unique(map(x -> x[3], filteredPositions)))
    pl = contour(c, p, f, fill=filled, aspect_ratio=1, xlims = (0,90), xticks = 0:10:90, xl
        yticks = 0:10:90, ylabel = "planes", title="row="*string(rowIndex))
    plot(pl)
end
```

Out[15]:

```
plotContourFixRow (generic function with 2 methods)
```

In [39]:

```
shotPosition = "041"
rotation = "000"
dose3d = readFile(shotPosition,rotation);
```

```
dat/RTDose2/RTDose_ImatrexPRTv2_16cmCyl_d80mm_s041rot000.dcm.txt
```

In [38]:

```
dose3d_2 = readFile("016",rotation);
```

```
dat/RTDose2/RTDose_ImatrexPRTv2_16cmCyl_d80mm_s016rot000.dcm.txt
```

In [32]:

```
dose3d_3 = readFile("051",rotation);
```

```
dat/RTDose2/RTDose_ImatrexPRTv2_16cmCyl_d80mm_s051rot000.dcm.txt
```

In [33]:

```
# compare 3d dose matrices
nnz(dose3d_2.doses - dose3d.doses);
```

In [34]:

```
nnz(dose3d_3.doses - dose3d.doses);
```

In [35]:

```
plotSurface(dose3d, 64, cmap);
```

In [36]:

```
# max dose
maxDose = maximum(dose3d.doses);
```

In [37]:

```
# index with max dose
maxIndex = findlast(x -> x==maxDose, dose3d.doses);
```

In [40]:

```
#finding voxel with max dose
index2coordinate(dose3d, maxIndex);
```

In [ ]:

```
# Note: plane is from 0 to 79, 0 corresponds to -80mm at the bottom of phantom,
# while index 79 corresponds to 78mm on the top of phantom
# max dose at plane index 64, i.e., depth = (79-64)*2 = 3cm

# plotContourFixPlane(dose3d, 79)

L1 = collect(0:10:40)
L2 = collect(45:5:60)
L3 = collect(61:1:79)
layer2display = vcat(L1,L2,L3)
gr()
# thông thường kết quả hiện ra trong 1 screen nhỏ và phải scroll nếu kết quả quá dài.
# Nếu muốn hiển thị hết trên màn hình phải chọn Cell -> Current Outputs -> Toggle Scrolling

for i in reverse!(layer2display)
  display(plotContourFixPlane(dose3d,i));
end
```

In [28]:

```
plotContourFixRow(dose3d,40);
```

In [29]:

```
plotContourFixRow(dose3d,30);
```

In [ ]:

```

```