

4. Geometry Files

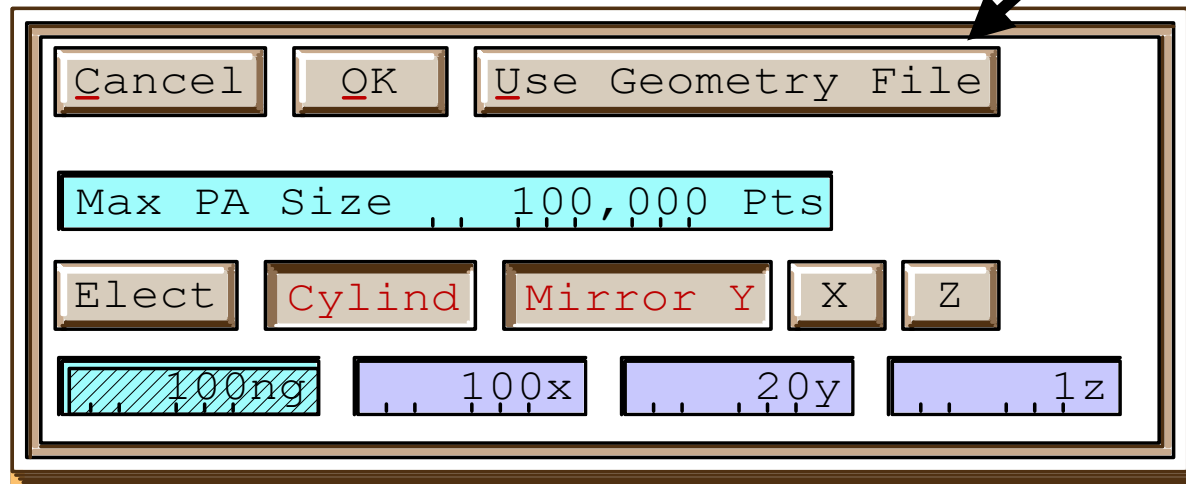
- A geometry file is an ASCII file with the .GEM filename extension that contains the electrode/pole definitions for a potential array.
- It also may contain the definition of the potential array:

```
pa_define(100,20)
```



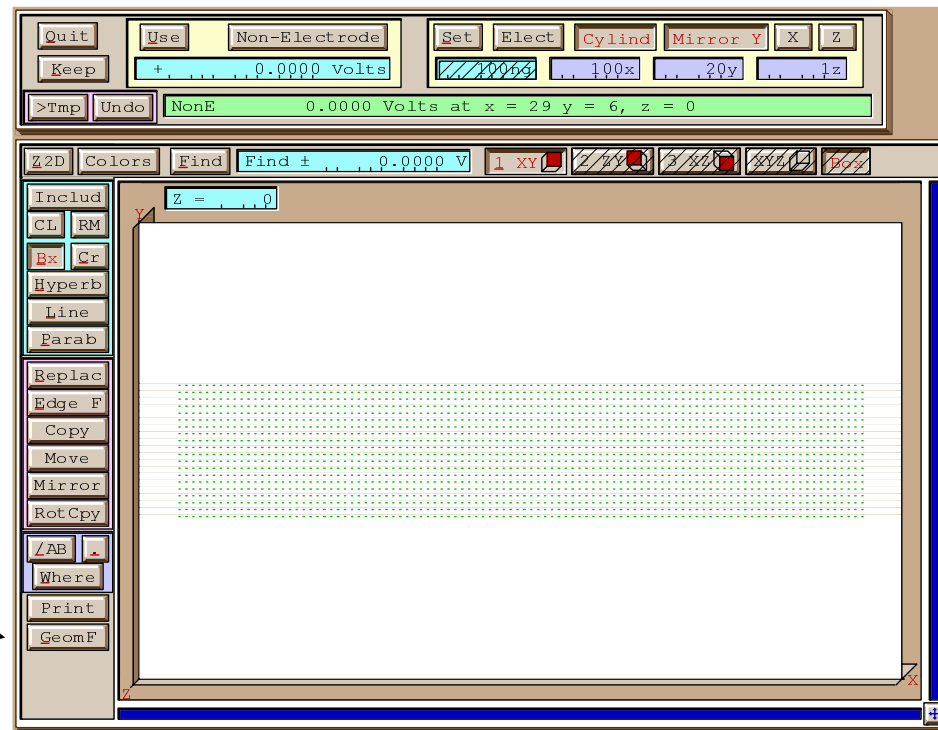
SIMION Functions that Use Geometry Files

- The New function via the Use Geometry File button (*makes use of pa_define commands*).



SIMION Functions that Use Geometry Files

- The Modify function via the **GeomF** button (*ignores pa_define commands for an existing array*).



How Geometry Files Work

- The geometry files contain a collection of fill commands.
- Fill commands define complex areas/volumes using:
 - inclusion/exclusion commands (e.g. within and notin),
 - orientation commands (e.g. locate), and
 - basic shape commands (e.g. circle).



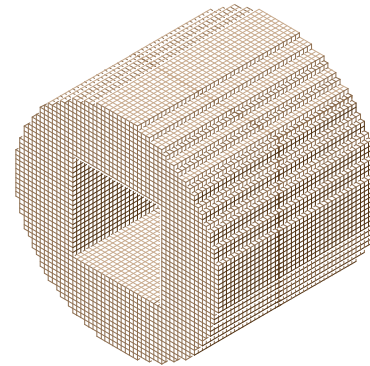
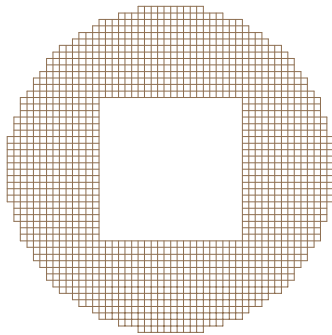
How Geometry Files Work

- Each successive fill command has increasing priority. Later fill commands override earlier fill commands.
- SIMION uses a geometry compiler to create an ordered fill search tree.
- Each point within the array is checked to see if it is affected by a fill command (from last fill toward first fill).



Example

```
e(1)                                ; electrode of one volt
{
  fill                                ; fill command
  {
    within{ circle(0,0,5)}           ; within a circle at 0,0 r = 5
    notin{ box(0,0,2,2)}             ; notin a box with corners of
  }                                    ; x,y: 0,0 and 2,2
}
```



Geometry File Language

- The Language is nested - commands fit inside each other.

```
e(1)                ; electrode of one volt
  {
    fill            ; fill command
      {
        within{    circle(0,0,5)} ; within a circle at 0,0 r = 5
          notin{   box(0,0,2,2)}  ; notin a box with corners of
        }          ; x,y: 0,0 and 2,2
      }
    }
  }
```



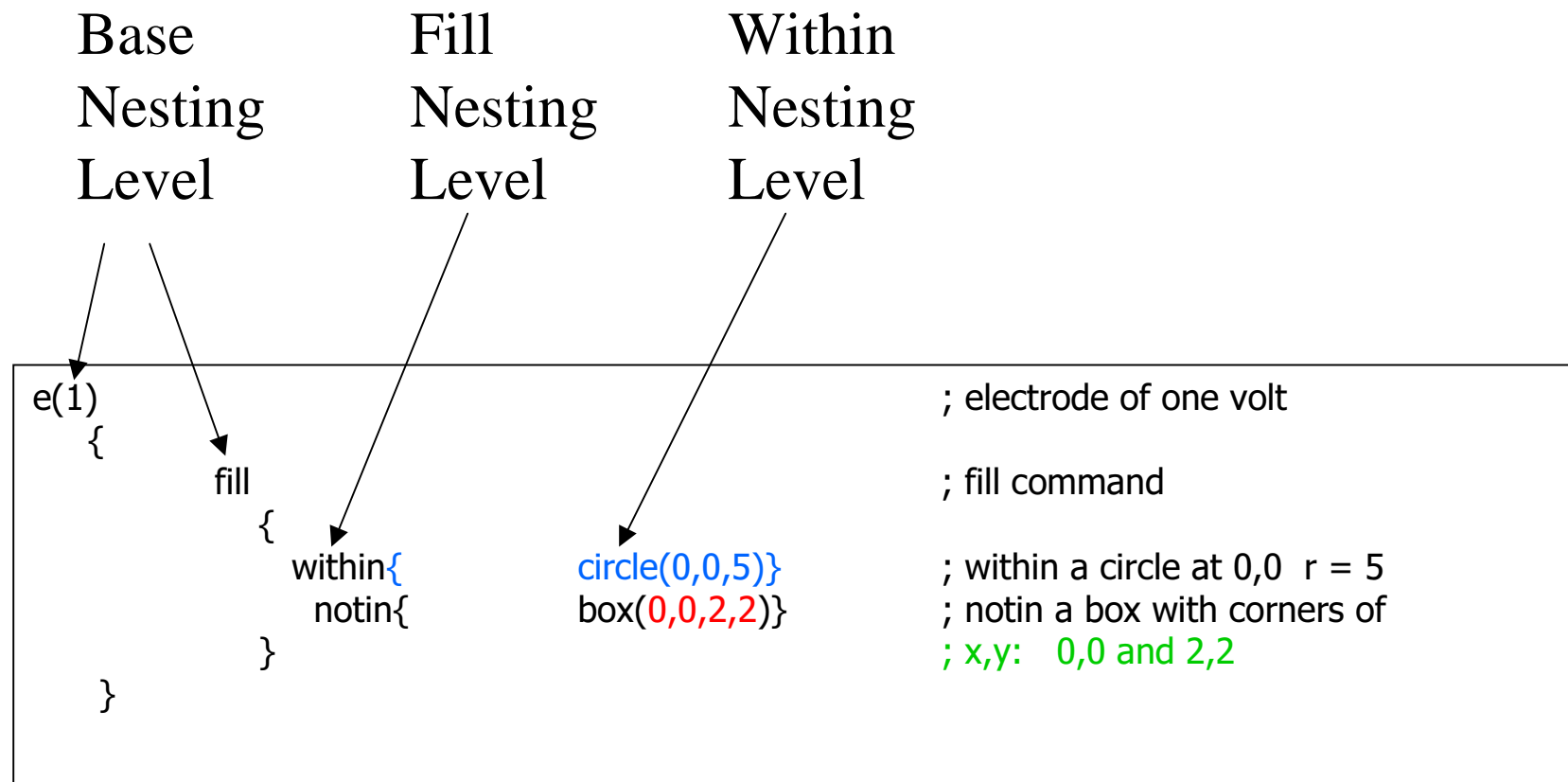
Geometry File Language

```
e(1) ; electrode of one volt
{
  fill ; fill command
  {
    within{ circle(0,0,5)} ; within a circle at 0,0 r = 5
    notin{ box(0,0,2,2)} ; notin a box with corners of
  } ; x,y: 0,0 and 2,2
}
```

- Each command has its **{scope}**: The region of commands that are under its influence.
- Commands have **(parameters)** and/or scope.
- Inline comments begin with a **semicolon**.



Geometry File Nesting Rules



Geometry File Development System

- Accessed from Modify with the GeomF button.
- Geometry files are edited, compiled and debugged.
- Potential array can be erased.
- Geometry definitions can be inserted into the potential array.
- Provides full manual control over translation, scaling and orientation via locate parameter panels (serves as the outermost Locate command).



Geometry File Development System

Quit egun2d.gem

Insert into PA Select Edit

Erase Entire PA Compile Xref

View .ger File

Default Position Scale 1.00000

X Off + 0.0000 Az + 0.0000 Deg

Y Off + 0.0000 El + 0.0000 Deg

Z Off + 0.0000 Rt + 0.0000 Deg

SIMION 7.0 Geometry File Compiler
Tue Apr 25 16:51:05 2000
Errors for: egun2d.gem

No Compiling Errors Encountered

Lines Processed
Main File: 41
Included: 0

In Total: 41

Number of Fills Intersecting PA's Volume: 4

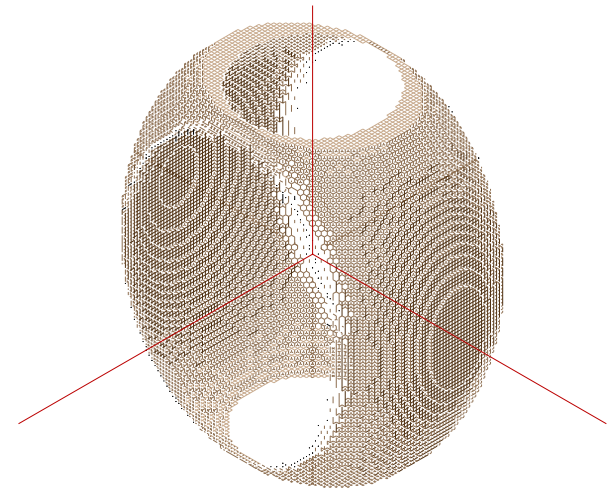
Volume of PA to Scan:
0 <= x <= 134
0 <= y <= 42
0 <= z <= 0



Location, Scaling, Orientation

- Locate commands are used to locate, scale, and orient anything at any level. They even can be nested within themselves:

```
locate(100)
{
  fill
  {
    within{sphere(0,0,0,50,60,40)}
    notin{sphere(0,0,0,45,55,45)}
    locate(0,0,0,1,0,0,-90)
    {
      notin{circle(0,0,10,5)}
    }
  }
}
```



Geometry File Lab



- Modifying the Trap Geometry File into a Stretched Trap
- Modifying the Trap Geometry File to Include End Cap Apertures

