11.0 User Programming Tricks

- Important concepts
- Useful overall strategies
- When to use various user program segments

User Program Files

- A user program file contains one or more program segments (functions).
- Each user program file is associated with one and only one potential array.
- Each potential array can be associated with one and only one user program file.
- User program files share the name of their array (e.g. einzel.pa0 <> einzel.prg).

Legal Program Segments

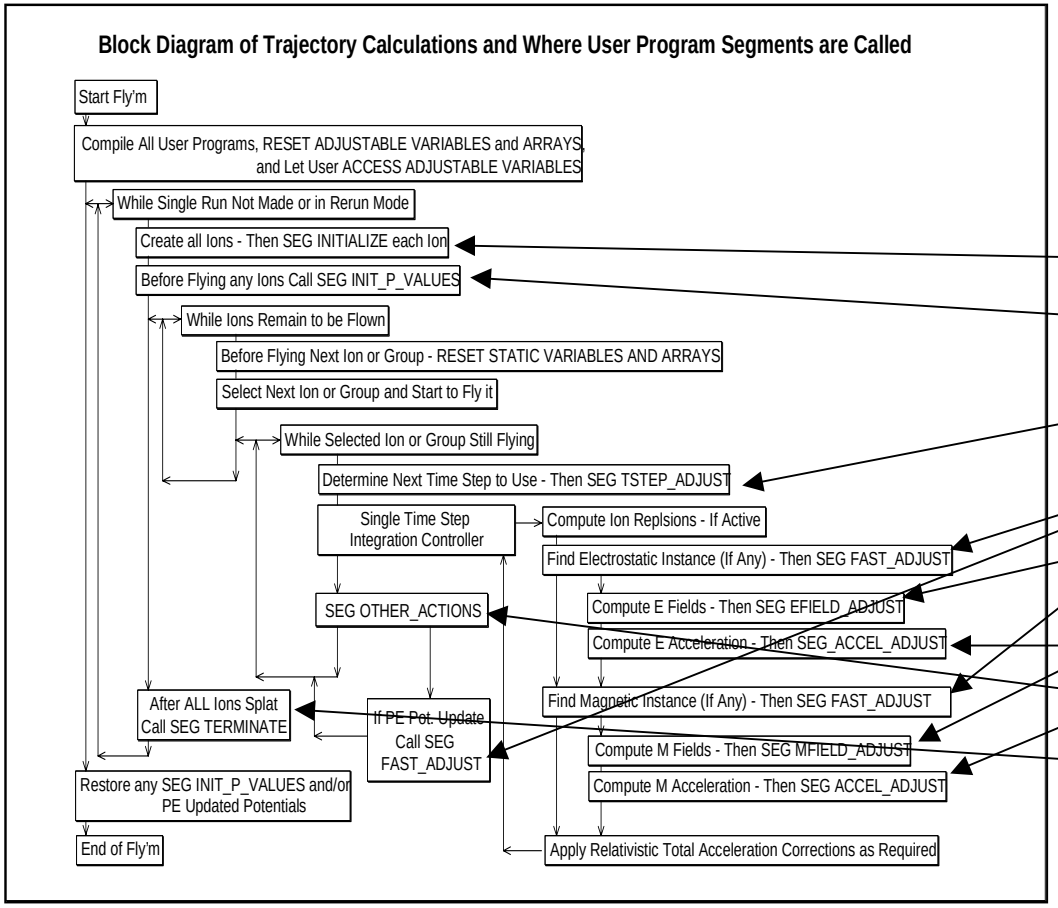
Only one segment of each type can be in a user program file

Define_Data	Defines Global variables and arrays
Init_P_Values	Pre-sets fast adjust potentials (new)
Initialize	Initializes ion's starting parameters
Tstep_Adjust	Controls time step size used
Fast_Adjust	Fast adjusts potentials as ions fly
Efield_Adjust	Adjusts electrostatic fields
Mfield_Adjust	Adjusts magnetic fields
Accel_Adjust	Adjusts ion accelerations
Other_Actions	Post tstep control of ion's parameters
Terminate	Post flight analysis and rerun control

Program Segment Access Rules

- Each program segment is ONLY accessed at specified points in the trajectory computation cycle.
- The ion MUST be in an instance that references a potential array that has a user program file containing the program segment (e.g. Initialize). The ONLY exception is the Init P Values segment.

Program Segments and Calculations



Program Segments

- Initialize
- Init_P_Values
- Tstep_Adjust
- Fast_Adjust
- Efield_Adjust
- Mfield_Adjust
- Accel_Adjust
- Other_Actions
- Terminate

Segment Access Paradigm

- Reserved Variables used for communication
- The propose and modify paradigm:
  - 1. SIMION calculates: proposed time step
  - 2. SIMION calls Tstep_Adjust (as per rules) passing value in ion_time_step
  - 3. Tstep_Adjust examines and/or modifies the value in ion_time_step
- Reserved Variable r/w access is controlled

Reserved Variables

- Reserved Variables used for communication between SIMION and program segments

Variable Name	Use	Units	Read Access	Write Access
Adj_Elect00 to Adj_Elect30	Fast Adj Electrode[1] Voltages	Volts	Fast_Adjust Init_P_Values	Fast_Adjust Init_P_Values
Adj_Pole00 to Adj_Pole30	Fast Adj Pole[1] Mag Potentials	Mags	Fast_Adjust Init_P_Values	Fast_Adjust Init_P_Values
Ion_Ax_mm Ion_Ay_mm Ion_Az_mm	Ion's current Acceleration (WB Coordinates)	mm/micro sec[2] (WB Orientation)	Accel_Adjust Other_Actions Terminate	Accel_Adjust
Ion_BfieldX_gu Ion_BfieldY_gu Ion_BfieldZ_gu	Magnetic Field at Ion's Location (PA's Orientation)	Gauss (PA's Orientation)	Mfield_Adjust	Mfield_Adjust
Ion_BfieldX_mm Ion_BfieldY_mm Ion_BfieldZ_mm	Magnetic Field at Ion's Location (WB Orientation)	Gauss (WB Orientation)	Accel_Adjust Other_Actions Terminate	None
Ion_Charge	Ion's current charge	in units of elementary charge	Any Prog Seg except Init_P_Values	Initialize Other_Actions
Ion_Color	Color of Ion	0-15	Initialize Other_Actions	Initialize Other_Actions
Ion_DvoltsX_gu Ion_DvoltsY_gu Ion_DvoltsZ_gu	Voltage Gradient at Ion's Location (PA's Orientation)	Volts/grid unit (PA's Orientation)	Efield_Adjust	Efield_Adjust
Ion_DvoltsX_mm Ion_DvoltsY_mm Ion_DvoltsZ_mm	Voltage Gradient at Ion's Location (WB Orientation)	Volts/mm (WB Orientation)	Mfield_Adjust Accel_Adjust Other_Actions Terminate	None
Ion_Instance	Current Instance	1- max instance	Any Prog Seg except Init_P_Values	None
Ion_Mass	Ion's current mass	amu	Any Prog Seg except Init_P_Values	Initialize Other_Actions
Ion_mm_Per_Grid_Unit	Min Current Scaling[2]	mm/grid unit	Any Prog Seg except Init_P_Values	None
Ion_Number	Ion's Number	1- max ion	Any Prog Seg except Init_P_Values	None

Variable Name	Use	Units	Read Access	Write Access
Ion_Px_Abs_gu Ion_Py_Abs_gu Ion_Pz_Abs_gu	Ion's current PA Array Coordinates	grid units (PA's Abs Coordinates)	Any Prog Seg except Init_P_Values	None
Ion_Px_gu Ion_Py_gu Ion_Pz_gu	Ion's current (PA's Coordinates)	grid units (PA's Coordinates)	Any Prog Seg except Init_P_Values	None
Ion_Px_mm Ion_Py_mm Ion_Pz_mm	Ion's current Workbench Coordinates	mm (WB Coordinates)	Any Prog Seg except Init_P_Values	Initialize Other_Actions
Ion_Splat	Ion Status Flag[3]	Flying = 0 Not Flying != 0	Initialize Other_Actions	Initialize Other_Actions
Ion_Time_of_Birth	Ion's Birth Time	micro seconds	Any Prog Seg except Init_P_Values	Initialize Other_Actions
Ion_Time_of_Flight	Ion's current TOF[4]	micro seconds	Any Prog Seg except Init_P_Values	Other_Actions
Ion_Time_Step	Ion's Time Step[5]	micro seconds	All but Initialize and Init_P_Values	Tstep_Adjust
Ion_Volts	Electrostatic Pot at Ion's Location	Volts	Efield_Adjust Mfield_Adjust Accel_Adjust Other_Actions Terminate	Efield_Adjust
Ion_Vx_mm Ion_Vy_mm Ion_Vz_mm	Ion's current Velocity (WB Coordinates)	mm/micro sec (WB Orientation)	Any Prog Seg except Init_P_Values	Initialize Other_Actions
Rerun_Flym	Rerun Flym Flag[6]	NO = 0 YES = 1	Initialize Other_Actions Terminate	Initialize Other_Actions Terminate
Trajectory_Image_Control	Controls Trajectory Image Viewing and Recording[7]	Value View Retain 0 YES YES 1 YES NO 2 NO YES 3 NO NO	Initialize Other_Actions Terminate	Initialize Other_Actions Terminate
Retain_Changed_Potentials	Controls restoring of changed potentials at end of Fly'm[8]	NO = 0 (default) YES = 1	None	Terminate
Update_PE_Surface	New PE Surface[9]	YES != 0	None	Other_Actions

Temporary Variables

- Visibility: Visible only in user program segment that defines it
- Lifetime: Exists only during a reference to the program segment
- User Adjustability: No
- Conserves Changes: No
- Error Trapping: Yes (for sto without rcl)

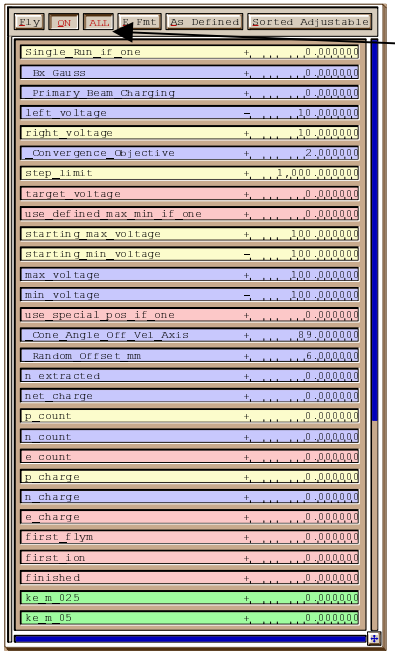
Static Variables & Arrays

- Visibility: Globally visible to most user programs except: Initialize, Init_P_values, and Terminate.
- Lifetime: Reset before each ion or group flight.
- User Adjustability: No
- Conserves Changes: No

Adjustable Variables & Arrays

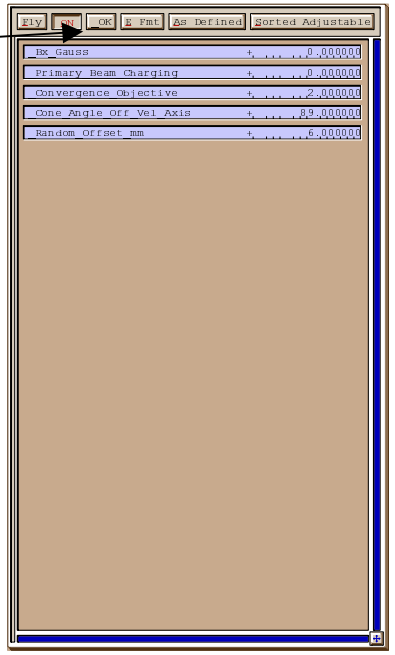
- Visibility: Globally visible to all user programs
- Lifetime: Values conserved throughout entire fly'm (including reruns).
- User Adjustable:
  - Before ion flying starts
  - During ion flying (via AdjV tabbed screen)
- Conserves Changes: YES (before fly'm)

Controlling Variable Visibility

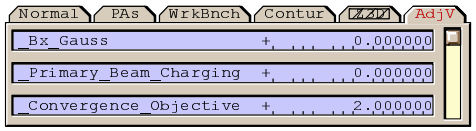


All
Adjustable
Variables

_OK
Leading
Underscore
Adjustable
Variables



AdjV runtime Screen



- Adjustable Variables
  - Use leading underscore in names to see (e.g. _Left_Electrode_Voltage)

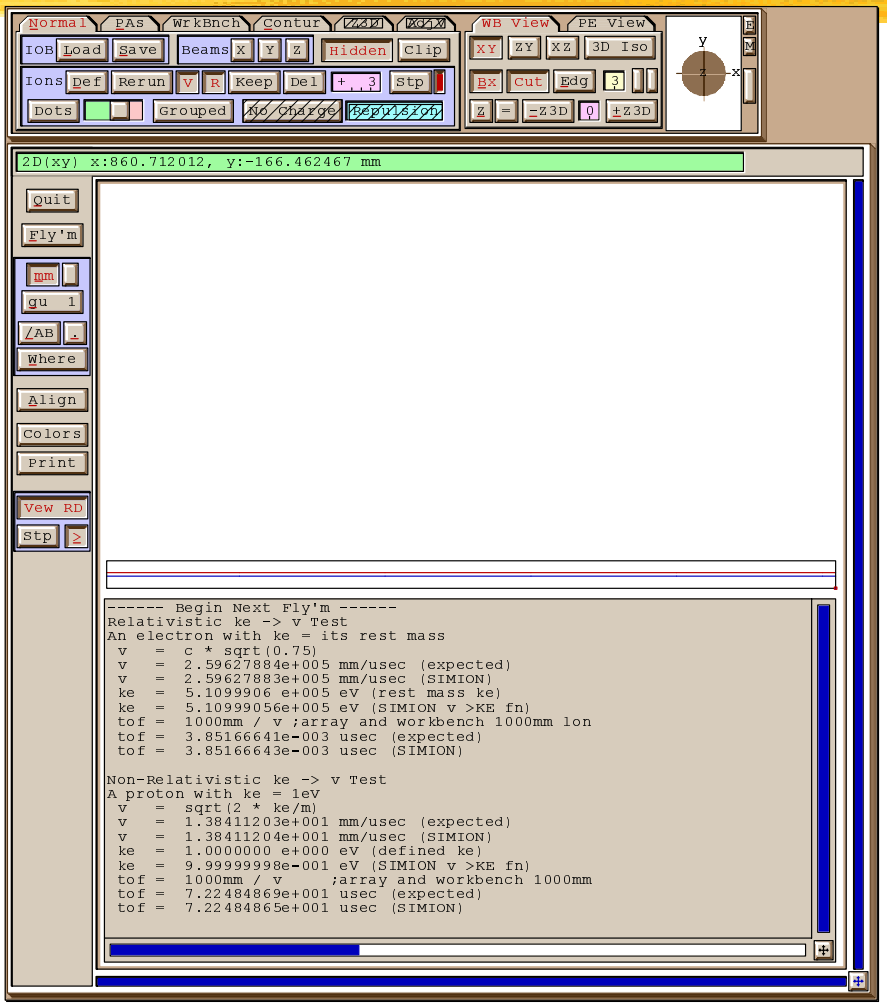
Controlling Variable Visibility

- Use Adjustable & Static Arrays for Flags

```
Adefa aflags 20 ;"defaults.dat" adjustable flags -- initialized
Adefs sflag    500  ; static array flag -- zeroed for each ion

1 arcl aflags          ; simple recall (x = aflags[1])
0 1 sto aflags         ; simple store (aflags[1] = 0)


defa flag 0     ;flag shown at start (unless _OK underscore)

rcl flag
1 sto flag
```

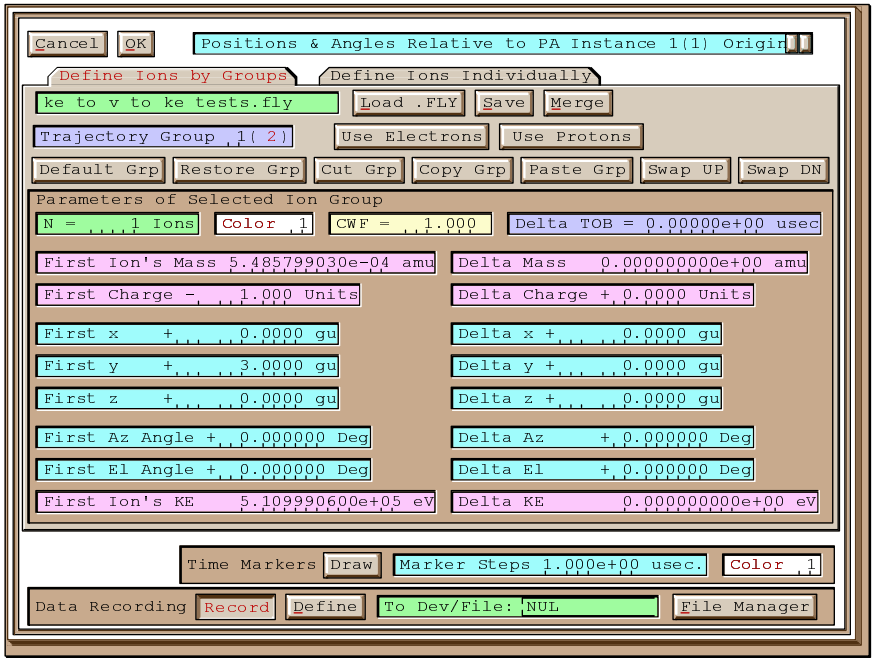
Creating Integrated Demos

1. Ke to Velocity
   (Test Demo)

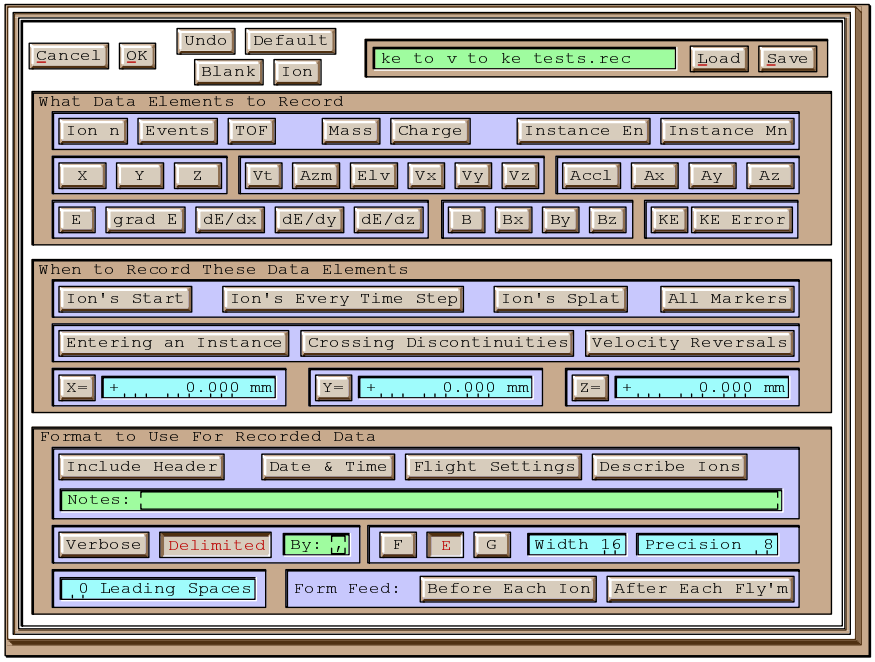


Creating Integrated Demos

1. Ke to Velocity (Test Demo)



Ion Definitions



Data Recording

Creating Integrated Demos

1. Ke to Velocity (Test Demo)

```
seg other_actions

rcl ion_splat x=0 exit                          ;exit if ion hasn't splatted
1 rcl ion_mass x>y goto ions   ;jump if proton
                                                ;else output relativistic electron data
        message ;Relativistic ke -> v Test
        message ;An electron with ke = its rest mass
                                                ;relativistic velocity from
                                                ; 2 = 1/(sqrt(1 - v*v/c*c))
        message ; v   =  c * sqrt(0.75)
                                                ; given c = 2.99792458e+005 mm/usec
        message ; v   =  2.59627884e+005 mm/usec (expected)
        rcl ion_vx_mm                           ; velocity at splat
        message ; v   = # mm/usec (SIMION)


                                                ; rest mass of electron
                                                ; and starting ke
        message ; ke  =  5.1099906 e+005 eV (rest mass ke)
        rcl ion_mass                            ; get mass of electron
        rcl ion_vx_mm                           ; get its exit velocity
        >ke                                     ; convert back to ke as check
        message ; ke  = # eV (SIMION v >KE fn)

                                                ;equation for const velocity tof
        message ; tof =  1000mm / v ;array and workbench 1000mm long

                                                ;tof assuming expected velocity
        message ; tof =  3.85166641e-003 usec (expected)
        rcl ion_time_of_flight                  ;tof computed by SIMION
        message ; tof = # usec (SIMION)
        message ;
        exit

lbl ions                                        ;proton data output
message ;Non-Relativistic ke -> v Test
message ;A proton with ke = 1eV
message ; v   =  sqrt(2 * ke/m)
                ; v   =  sqrt(2.0 * 1.60217733e-19/1.6726231e-27)
                ;where                  1eV = 1.60217733e-19 J
                ;               proton's mass = 1.6726231e-27 kg
message ; v   =  1.38411203e+001 mm/usec (expected)
rcl ion_vx_mm                                   ; velocity at splat
message ; v   = # mm/usec (SIMION)
                                                ; defined initial ke
message ; ke  =  1.0000000 e+000 eV (defined ke)
rcl ion_mass                                    ; get mass of proton
rcl ion_vx_mm                                   ; get its exit velocity
>ke                                             ; convert back to ke as check
message ; ke  = # eV (SIMION v >KE fn)

                                                ;equation for const velocity tof
message ; tof =  1000mm / v   ;array and workbench 1000mm long
                                                ;tof assuming expected velocity
message ; tof =  7.22484869e+001 usec (expected)
rcl ion_time_of_flight                          ;tof computed by SIMION
message ; tof = # usec (SIMION)
exit
```

Creating Integrated Demos

- Recommended Strategies
  - Use auto-loading FLY (or ION) and REC files
  - Control displayed number format with REC file
  - Use self-evident message output if possible via Initialize, Other_Actions, Terminate, and/or data recording (REC)
  - Hide Adjustable Variables from user (e.g. _Voltage (shown), flag (hidden), or via arrays)

Randomizing Ions

- Use Initialize Segment to Randomize Ions

_Random Directory

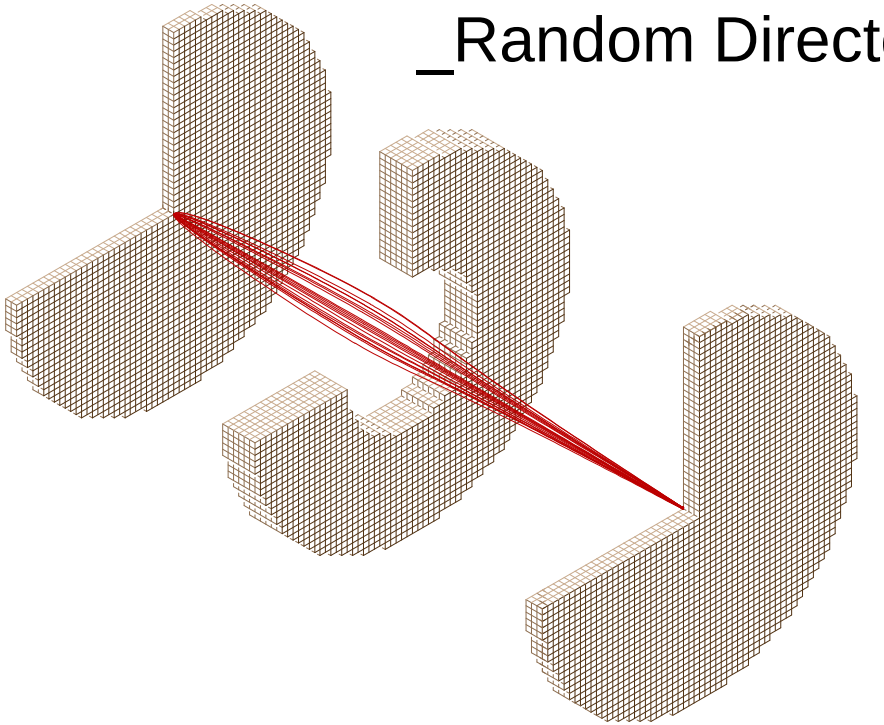
Initialize

Input Ions

Group of ions with identical starting parameters.

Adjustable Variables

Percent_Energy_Variation
Cone_Angle_Off_Vel_Axis



Randomizing Ions

- Initialize Segment Strategy

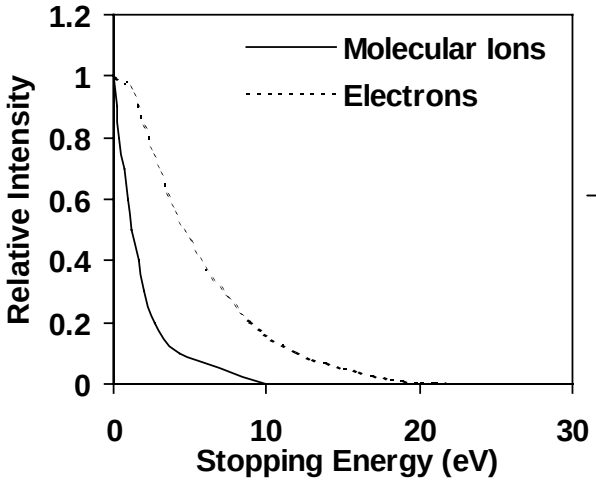
Get polar v components	Get ion's velocity components from SIMION Convert to 3D polar and save speed plus az and el angles
Compute KE	get ion_mass, recall speed, convert to ke, and save ke
Randomize KE	use rand and percent_energy_variation to randomize ke
Convert to v	get ion_mass, recall new ke, convert to v, and save speed
Create random conical pattern in vertical	assume ion flying vertical randomize el off vertical by cone angle
Convert to 3D Rectangular	randomize az to +- 180 degrees convert to rectangular 3D coords
Swing back to az = el = 0	rotate vector -90 in elevation (restores a starting point)
Rotate entry el	rotate vector initial el angle
Rotate entry az	rotate vector initial az angle
Pass V Rect. to SIMION	pass ion's velocity components back to SIMION

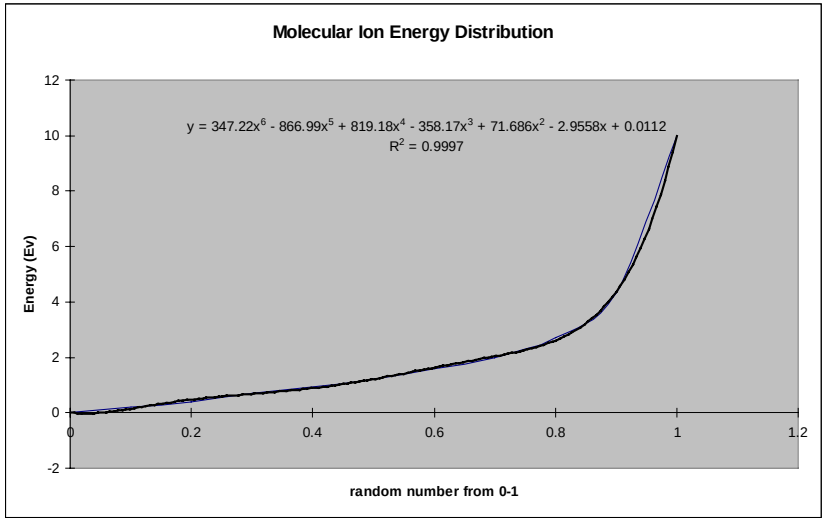
Initialize for Random Ions

```
;-------------------------------------------------------------------
; this user program randomly changes the initial ke and direction of ions
; energy is randomly changed +- Percent_Energy_Variation * ke
; ions are emitted randomly within a cone of revolution around the
; ion's defined velocity direction axis
; the full angle of the cone is +- Cone_Angle_Off_Vel_Axis
; (e.g. 90.0 is full hemisphere, 180 is a full sphere)
;-------------------------------------------------------------------

;--------- you can use it with your own lenses without modification ----------
;          (just rename user program file using your pa's name)
;------ Note: you can also modify the emission distributions as desired ------
defa Percent_Energy_Variation     50    ; (+- 50%) random energy variation
defa Cone_Angle_Off_Vel_Axis      90    ; (+- 90 deg) cone angle hemisphere


seg initialize                        ; initialize ion's velocity and direction
      ;------------------- get ion's initial velocity components -------------
      rcl ion_vz_mm                   ; get ion's specified velocity components
      rcl ion_vy_mm
      rcl ion_vx_mm

      ;------------------- convert to 3d polar coords -------------
      >p3d                            ; convert to polar 3d

      ;------------------- save polar coord values ----------------
      sto speed rlup                  ; store ion's speed
      sto az_angle rlup               ; store ion's az angle
      sto el_angle                    ; store ion's el angle

      ;------------------- make sure Percent_Energy_Variation is legal -------------
                        ; force 0 <= Percent_Energy_Variation <= 100
      rcl Percent_Energy_Variation abs
      100 x>y rlup sto Percent_Energy_Variation

      ;------------------- make sure Cone_Angle_Off_Vel_Axis is legal -------------
                        ; force 0 <= Cone_Angle_Off_Vel_Axis <= 180
      rcl Cone_Angle_Off_Vel_Axis abs
      180 x>y rlup sto Cone_Angle_Off_Vel_Axis

      ; ---------------------- calculate ion's defined ke -------------
      rcl ion_mass                    ; get ion's mass
      rcl speed                       ; recall its total speed
      >ke                             ; convert speed to kinetic energy
      sto kinetic_energy              ; save ion's defined kinetic energy

; ---------------------- compute new randomized ke -------------
                              ; convert from percent to fraction
      rcl Percent_Energy_Variation 100 /
      sto del_energy 2 * rand *       ; fac = 2 * del_energy * rand
      rcl del_energy - 1 +            ; fac += 1 - del_energy
      rcl kinetic_energy *            ; new ke = fac * ke

      ; ---------------------- convert new ke to new speed -----------
      rcl ion_mass                    ; recall ion mass
      x><y                            ; swap x any y
      >spd                            ; convert to speed
      sto speed                       ; save new speed

      ;-- compute randomized el angle change 90 +- Cone_Angle_Off_Vel_Axis -------
      ;-------- we assume elevation of 90 degrees for mean ----------
      ;-------- so cone can be generated via rotating az +- 90 -------
                              ; (2 * Cone_Angle_Off_Vel_Axis * rand)
      2 rcl Cone_Angle_Off_Vel_Axis * rand *
                              ;  - Cone_Angle_Off_Vel_Axis + 90
      rcl Cone_Angle_Off_Vel_Axis - 90 +

      ;-------------- compute randomized az angle change ------------
      ;--------- this gives 360 effective because of +- elevation angels ---
      180 rand * 90 -                 ;          +- 90 randomized az

      ;---------------------- recall new ion speed ------------------
      rcl speed                       ; recall new speed

      ;--------- at this point x = speed, y = az, z = el --------------
      ;------------ convert to rectangular velocity components ---------
      >r3d                            ; convert polar 3d to rect 3d

      ;------------ el rotate back to from 90 vertical -------------
      -90 >elr

      ;------------ el rotate back to starting elevation -------------
      rcl el_angle >elr

      ;------------ az rotate back to starting azimuth -------------
      rcl az_angle >azr

      ;------------ update ion's velocity components with new values --------
      sto ion_vx_mm                   ; return vx
      rlup
      sto ion_vy_mm                   ; return vy
      rlup
      sto ion_vz_mm                   ; return vz

      ;---------------------- done ---------------------------------------
```

Another Randomizing Example

- Monte Carlo example
  - Secondary ion energies (fitting)





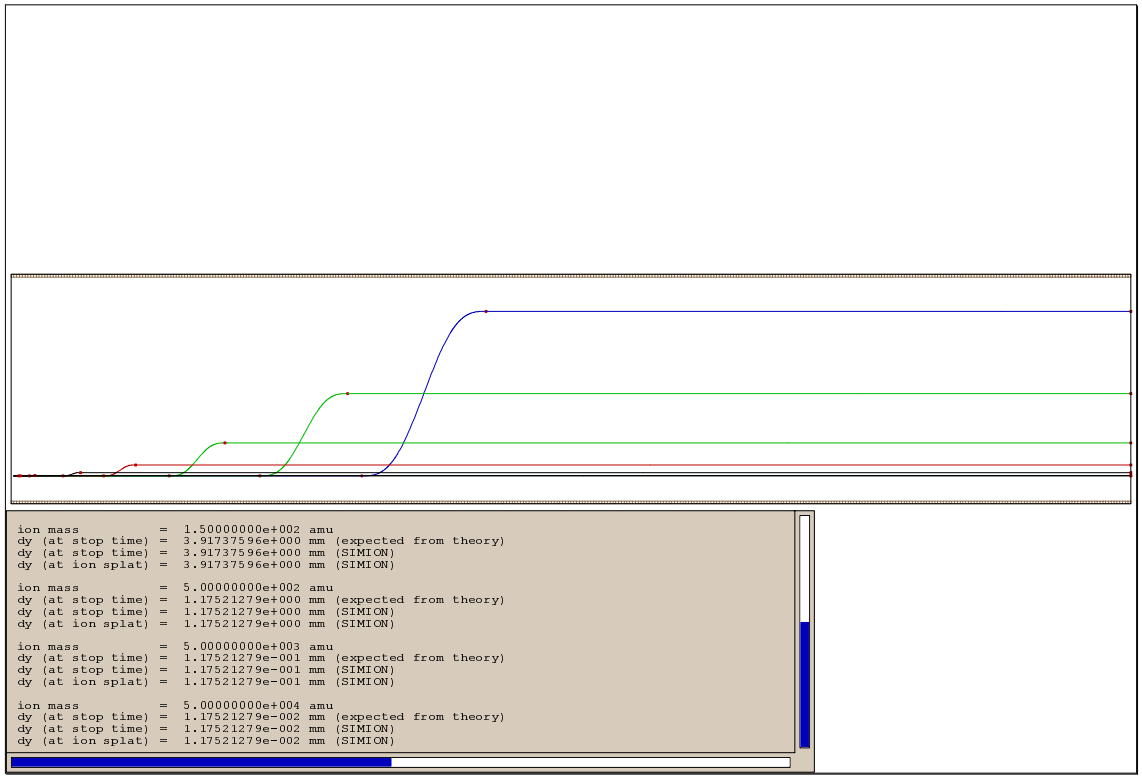
Another Randomizing Example

- Monte Carlo example cont.
  - Secondary ion energies (coding)

```
;molecular energies
rand sto x
          ;get random number x (0-1)
0.0112
          ; +0.0112
2.9558 rcl x * -                                  ; -2.9558x
71.686 rcl x        rcl x * sto xpower * +        ; +71.686x^2
358.17 rcl xpower rcl x * sto xpower * -          ; -358.17x^3
819.18 rcl xpower rcl x * sto xpower * +          ; +819.18x^4
866.99 rcl xpower rcl x * sto xpower * -          ; -866.99x^5
347.22 rcl xpower rcl x * sto xpower * +          ; +347.22x^6
abs
```

Controling Times and Time Steps

- Critical Timing Example



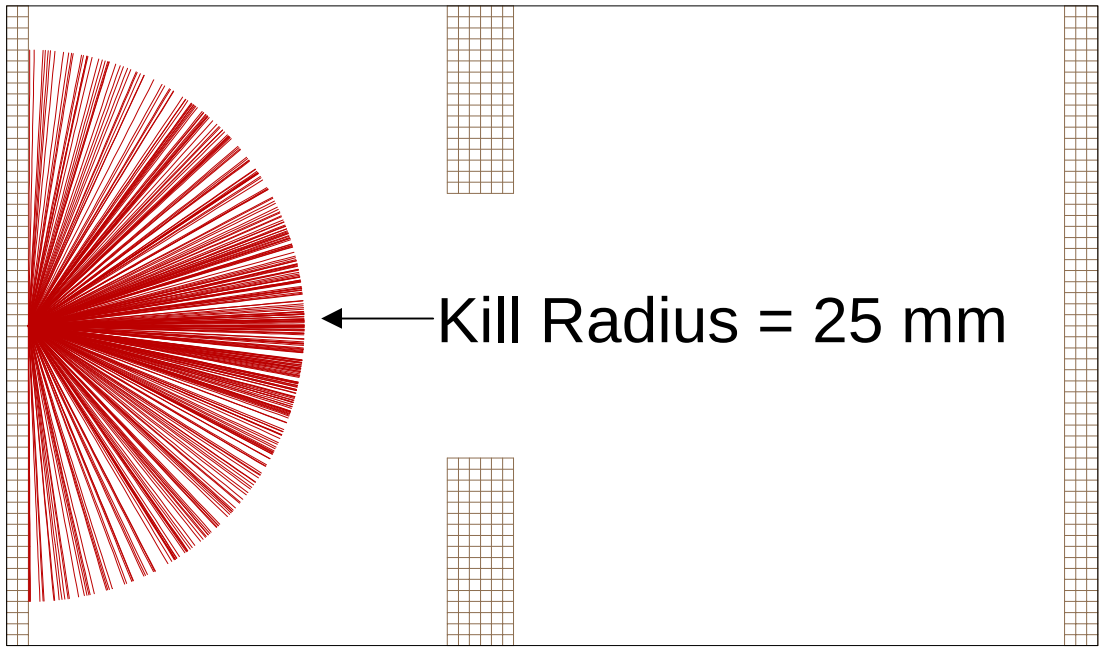
Controling Times and Time Steps

- Strategy: Tstep_Adjust via Ion_Time_Step

```
If TOF >= left  then right test
If TOF + TS < left then exit
ion_time_step = left - TOF
exit

LBL right test
If TOF >= right then exit
If TS > MAX then TS = MAX
If TOF + TS < right then exit
ion_time_step = right - TOF
exit
```

Binary Boundary Approach

- Use Tstep_Adjust and Other_Actions



Binary Boundary Approach

- Strategy

Other_Actions Control

If boundary crossed
    if not too close
    restore prior ion state
    save time_step
    set half step flag

else
    save ion's state
    (Px,Py,Pz,
    Vx,Vy,Vz,TOF)

Tstep Control

If half step flag not set
    exit

else
    get saved time_step
    halve it
    save to ion_time_step
    clear half step flag

Fast Adjusting Potentials

- Strategy: Fast_Adjust or Init_P_Values
  - Time varying fields: Fast_Adjust (only)
    - Tstep_Adjust may be needed for step size control
  - Static fields: Fast_Adjust or Init_P_Values
    - Fast_Adjust for large arrays and smaller numbers of ions.
    - Init_P_Values for smaller arrays and larger numbers of ions.
    - Conduct time trials (very useful).

Specifying Known Fields

- Strategy: Mfield_Adjust or Efield_Adjust
  - Fixed Magnetic fields (Mfield_Adjust)
    - Use field-free magnetic array

    - 0 sto BfieldZ_gu ;zero Bz
      sto BfieldY_gu ;zero By
      rcl _Bfield_gauss ;get Bx from Adj. Var.
      sto BfieldX_gu ;save user defined Bx

Specifying Known Fields

- Strategy: Mfield_Adjust or Efield_Adjust
  - Measured Magnetic fields (Mfield_Adjust)
    - Use field-free magnetic array
    - Adefa Bfield 10000 ;”mfield.dat” measured B
    - Get ion’s gu coords (abs or 3D)
      Use to get interpolated B fields at point
      Pass Bx, By, and Bz fields back to SIMION
      (see measured magnetic fields example)

Specifying Known Fields

- Strategy: Mfield_Adjust or Efield_Adjust
  - Analytic Electrostatic fields (Efield_Adjust)

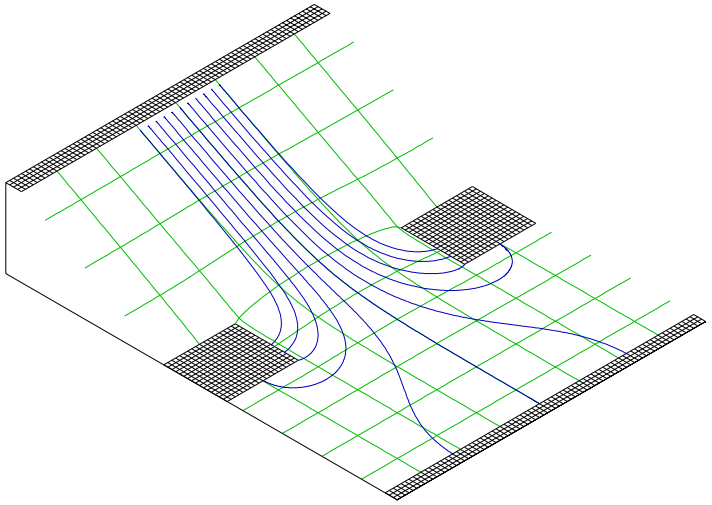
```
                    seg efield_adjust                       ;quadrupolar field
Selectively use         3 rcl ion_number x<=y exit          ;use refined field <= 3

                        rcl ion_px_mm entr *                ; x^2
Compute and store       rcl ion_py_mm entr * -              ; (x^2 - y^2)
potential               10 / sto ion_volts                  ; (x^2 - y^2)/10
                        rcl ion_px_mm 0.2 *                 ; volts/mm x gradient
Compute and store       rcl ion_mm_per_grid_unit *          ; volts/gu x gradient
x gradient              sto ion_dvoltsx_gu                  ; pass to SIMION
                        rcl ion_py_mm -0.2 *                ; volts/mm y gradient
Compute and store       rcl ion_mm_per_grid_unit *          ; volts/gu y gradient
y gradient              sto ion_dvoltsy_gu                  ; pass to SIMION
Zero z gradient         0 sto ion_dvoltsz_gu                ; no field in z
                        exit
```

Modifying Ion Accelerations

- Strategy: Use Accel_Adjust

```
seg accel_adjust                ; beginning of accel_adjust segment

   rcl ion_time_step x=0 exit   ; exit if zero time step
   rcl linear_damping x=0 exit  ; exit if damping set to zero
   abs sto damping              ; force damping term to be positive
   * sto tterm                  ; compute and save number of time constants
   chs e^x 1 x><y -             ; (1 - e^(-(t * damping)))
   rcl tterm / sto factor       ; factor = (1 - e^(-(t * damping)))/(t * damping)

   rcl ion_ax_mm                ; recall ax acceleration
   rcl ion_vx_mm                ; recall vx velocity
   rcl damping * -              ; multiply times damping and sub from ax
   rcl factor *                 ; multiply times factor
   sto ion_ax_mm                ; store as new ax acceleration

   rcl ion_ay_mm                ; recall ay acceleration
   rcl ion_vy_mm                ; recall vy velocity
   rcl damping * -              ; multiply times damping and sub from ay
   rcl factor *                 ; multiply times factor
   sto ion_ay_mm                ; store as new ay acceleration

   rcl ion_az_mm                ; recall az acceleration
   rcl ion_vz_mm                ; recall vz velocity
   rcl damping * -              ; multiply times damping and sub from az
   rcl factor *                 ; multiply times factor
   sto ion_az_mm                ; store as new az acceleration
```

Drag Demo



Arbitrary Ion Accelerations

- Strategy: Use Accel_Adjust
  - Can define arbitrary field forces (e.g. $1/r^5$)
  - Can define arbitrary field directions
    - Always normal to ion's current velocity
    - Always parallel to ion's current velocity
    - Moving points of attraction
    - Random points of attraction
    - Whatever you can imagine

Controlling Ion's Fate

- Strategy: Other_Actions
  - Position
  - Velocity
  - Mass
  - Charge
  - Color
  - TOF
  - Splat

User Programming's

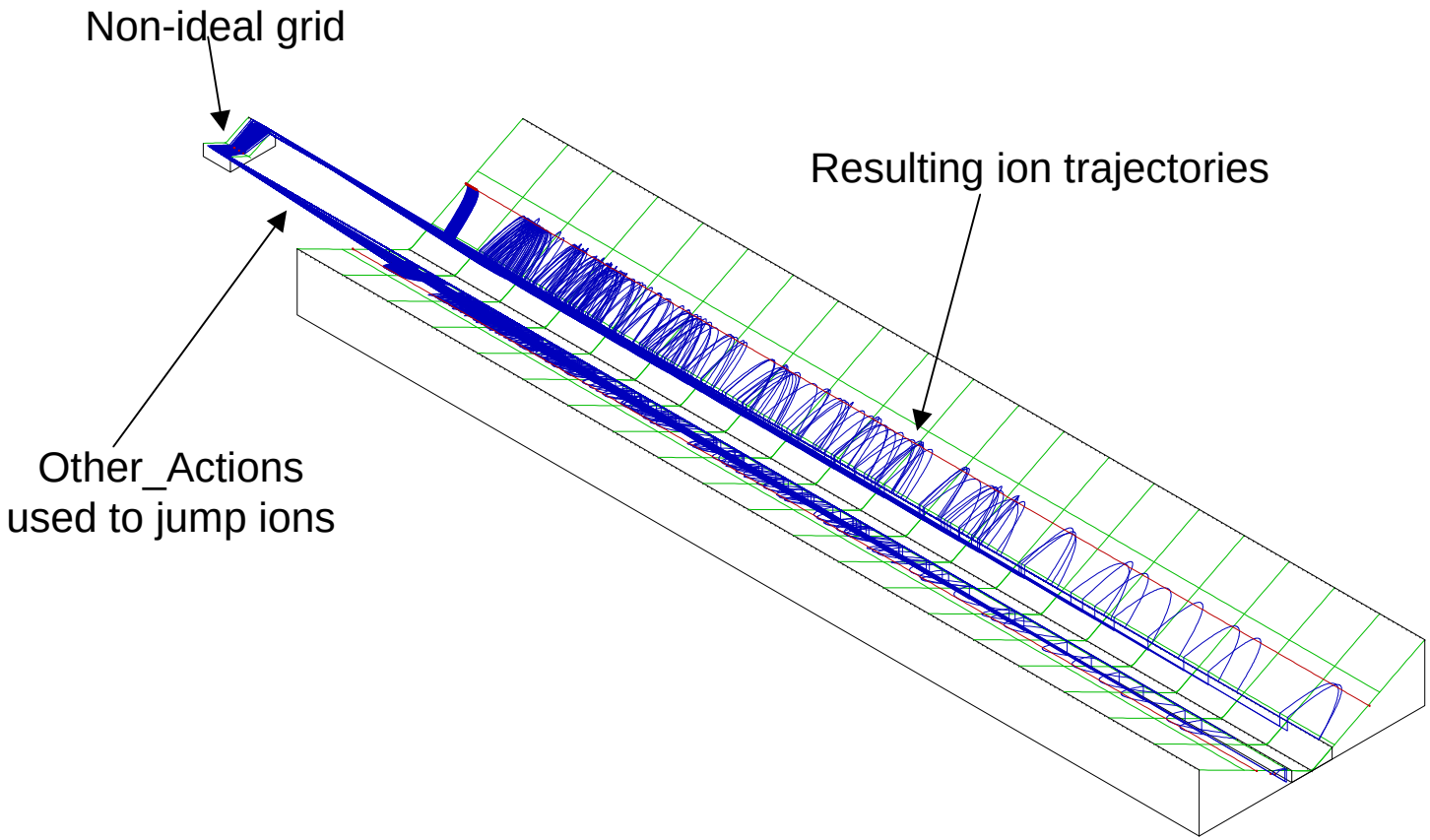
Master Controller

(Most Powerful Program Segment)

Controlling Ion's Fate

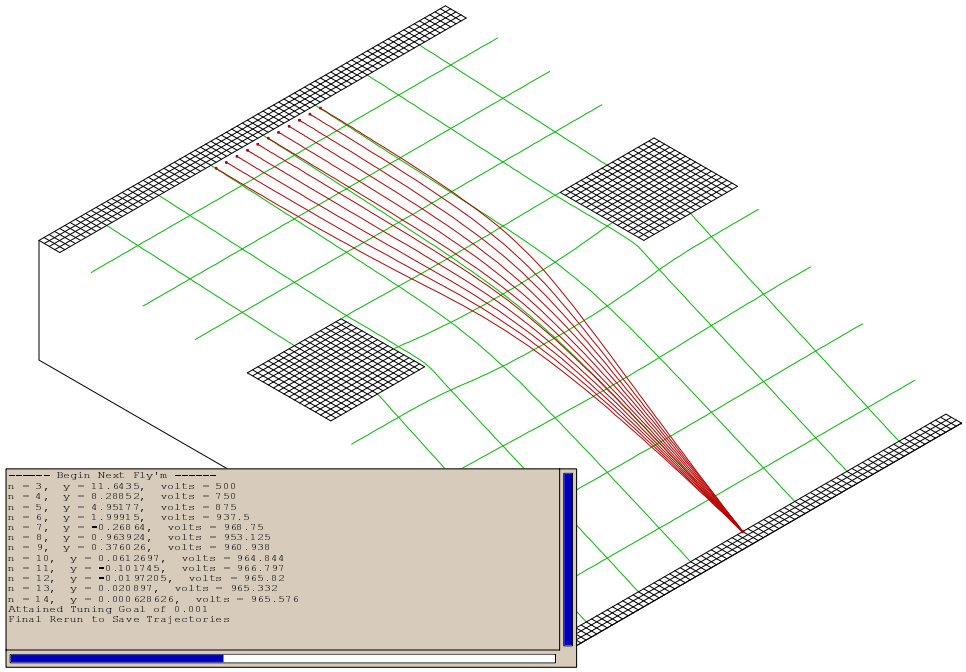
- Strategy: Other_Actions
  - Mass and/or charge
  - Energy and/or position
  - Boundary approaches
  - Collision simulations
  - Simulations involving jumps
  - Selective ion marking (e.g. via color & marks)
  - Selective ion killing

Controlling Ion's Fate



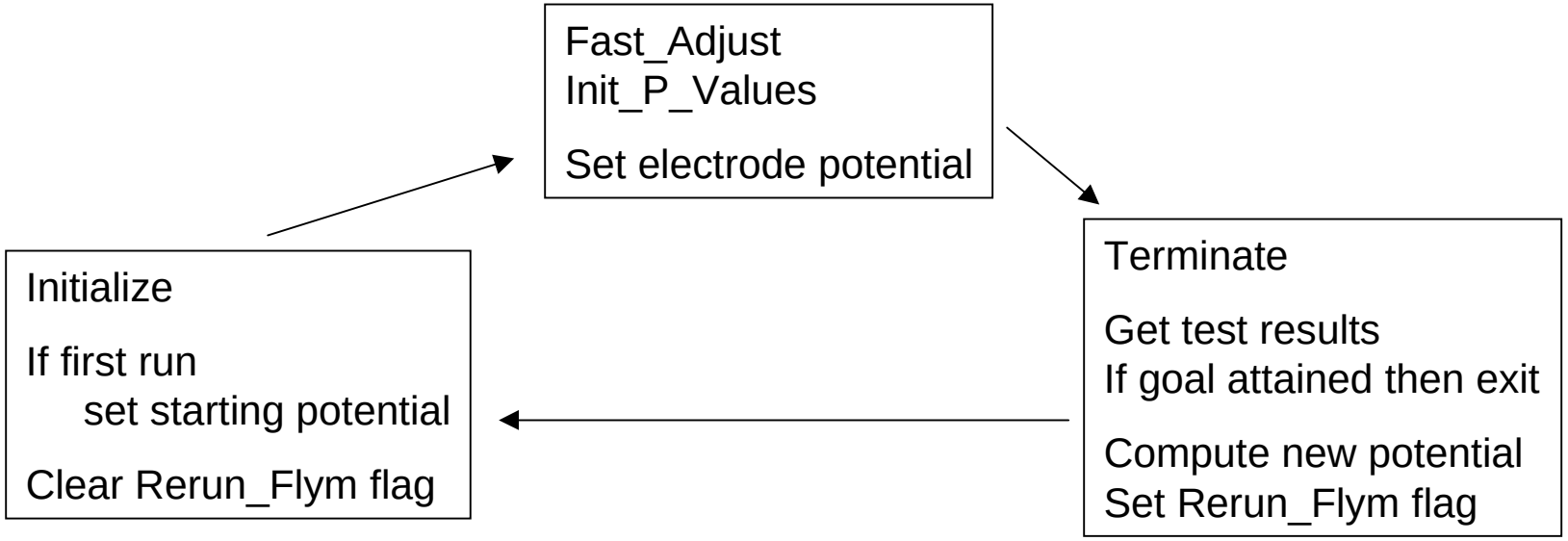
Controlling Successive Runs

- Strategy: Terminate and Initialize
  - Via the Rerun_Flym reserved variable



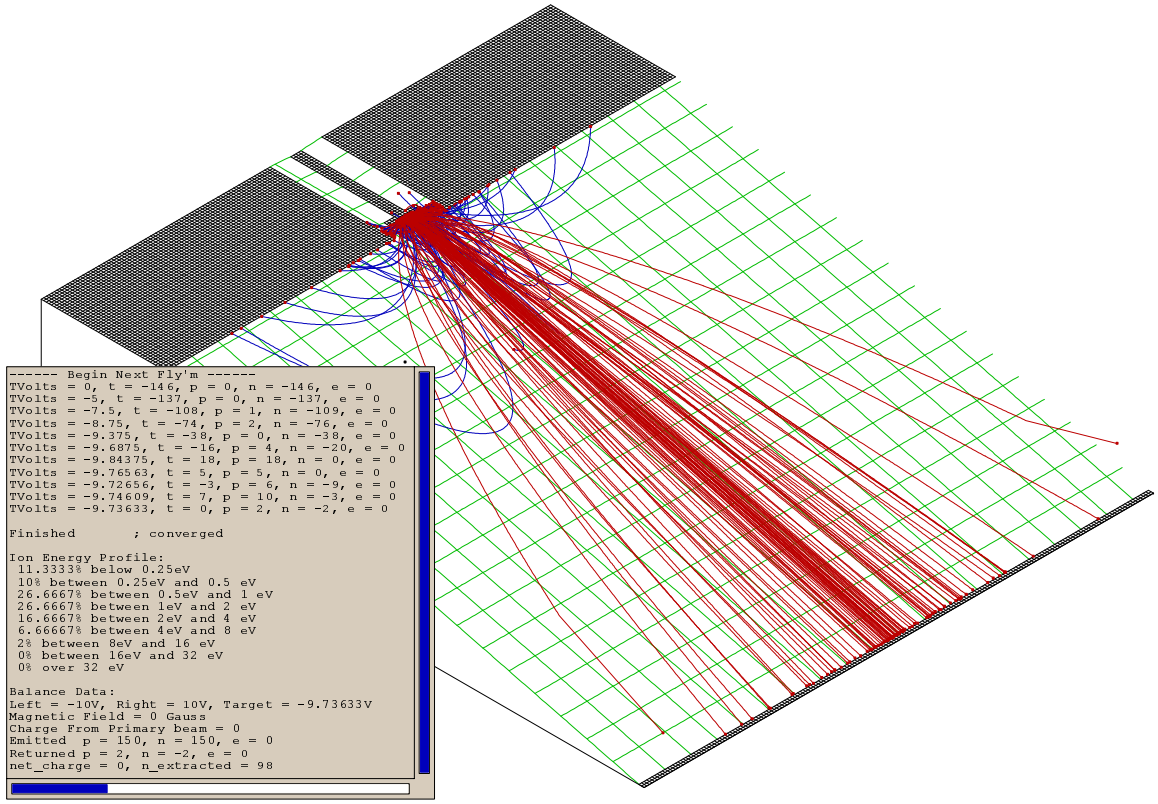
Controlling Successive Runs

- Strategy: Terminate and Initialize



Complex Multi-Run Simulations

- Monte Carlo -- Self-Charge Stabilization



Afternoon User Programming Lab

- Self-directed lab. Do what interests you.
- Options:
  - Form groups to work on a common interest
  - Explore user program demos in reference
  - Continue to explore a previous lab
  - Attack the instructors