Writing components

ESS McStas Training 2016
May 30th - June 1st

Topics

- Important runtime library macros/functions
- Important component details
- Let's write a simplistic example component

McStas internals 1) Runtime libraries

- Typically originates from one or a few (big) components
- Resides in $MCSTAS/share
- C-style library with .c and .h files
- List from 2.2a (most important listed):
  - mccode-r.c / mccode-r.h - common with McXtrace, used by all instrs
  - mcstas-r.c / mcstas-r.h - neutron specifics, used by all inserts
  - read_table-lib.c / read_table-lib.h - Used by comps reading files, e.g. samples
  - interoff-lib.c / interoff-lib.h - Used by “any shape” components
  - monitor_nd-lib.c / monitor_nd-lib.h - Used when using Monitor_nD and relatives
  - ref-lib.c / ref-lib.h - Used by reflecting optics
  - interpolation.c / interpolation.h - interpolation library
  - nexus-lib.c / nexus-lib.h - NeXus Data format
  - pol-lib.c / pol-lib.h - Polarization
  - ref-lib.c / ref-lib.h (-”- but should be generalized for e.g. Guides)
  - adapt_tree-lib.c / adapt_tree-lib.h - Adaptive source

McStas internals 1) Runtime libraries, 'share'

- Typically originates from one or a few (big) components
- Resides in $MCSTAS/share
- C-style library with .c and .h files
- List from 2.2a (most important listed):

  - mccode-r.c / mccode-r.h - common with McXtrace, used by all instrs
  - mcstas-r.c / mcstas-r.h - neutron specifics, used by all inserts

Windows - c:\mcstas-2.2a\lib\

Mac OS X - /Application/McStas-2.2.a/Contents/Resources/mcstas/2.2a/
(also as link in /usr/local/mcstas/2.2a/

Debian, Ubuntu etc. - /usr/share/mcstas/2.2a/

Other Unix - /usr/local/mcstas/2.2a/

  - pol-lib.c / pol-lib.h - Polarization
  - ref-lib.c / ref-lib.h (-"- but should be generalized for e.g. Guides)

  - adapt_tree-lib.c / adapt_tree-lib.h - Adaptive source

McStas internals 2) Important runtime libraries

- Let's look at the so-called mcstas runtime or mccode-r.h/c + mcstas-r.h / c

- Types of content:
  - Inclusion of basic c-libs e.g. math.h
  - Portability issues i.e. Linux vs. Mac vs. Win32 specifics (pathdefs, signals, blah, blah...)
  - Interoperability with other codes, e.g. DANSE
  - I/O and data output formats
  - MPI (parallel computing)
  - Various useful user macros - you will see the list shortly

Runtime - constants

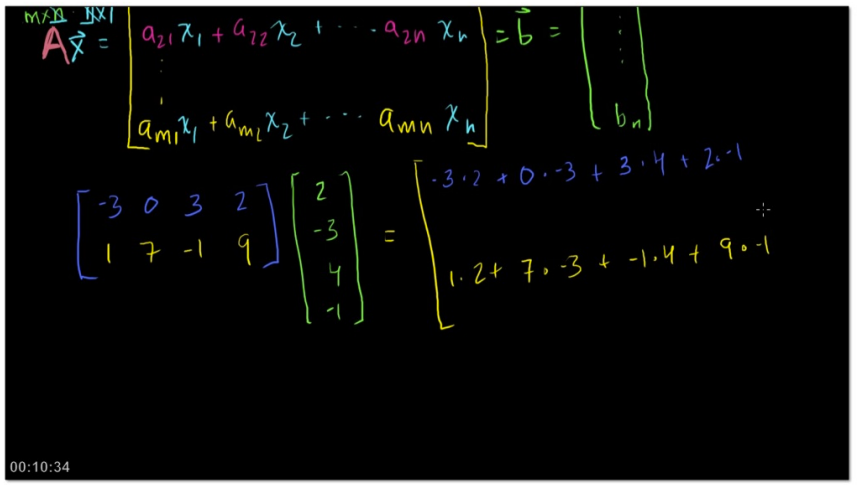
- #define RAD2MIN ((180*60)/PI)
- #define MIN2RAD (PI/(180*60))
- #define DEG2RAD (PI/180)
- #define RAD2DEG (180/PI)
- #define AA2MS   629.622368   /* Convert k[1/AA] to v[m/s] */
- #define MS2AA   1.58825361e-3   /* Convert v[m/s] to k[1/AA] */
- #define K2V   AA2MS
- #define V2K   MS2AA
- #define Q2V   AA2MS
- #define V2Q   MS2AA
- #define SE2V   437.393377   /* Convert sqrt(E)[meV] to v[m/s] */
- #define VS2E   5.22703725e-6   /* Convert (v[m/s])**2 to E[meV] */
- #define FWHM2RMS 0.424660900144   /* Convert between full-width-half-max and */
- #define RMS2FWHM 2.35482004503   /* root-mean-square (standard deviation) */
- #define HBAR   1.05457168e-34   /* [Js] h bar Planck constant CODATA 2002 */
- #define MNEUTRON 1.67492728e-27   /* [kg] mass of neutron CODATA 2002 */
- #define GRAVITY 9.81   /* [m/s^2] gravitational acceleration */
- # define PI M_PI
- # define PI 3.14159265358979323846

Runtime - Matrix/vector

- vec_prod(&x, &y, &z, x1, y1, z1, x2, y2, z2)
- double scalar_prod(x1, y1, z1, x2, y2, z2)
- NORM(&x,&y,&z)
- rotate(&x, &y, &z, vx, vy, vz, phi, ax, ay, az)
- mirror(&x,&y,&z,rx,ry,rz,nx,ny,nz)

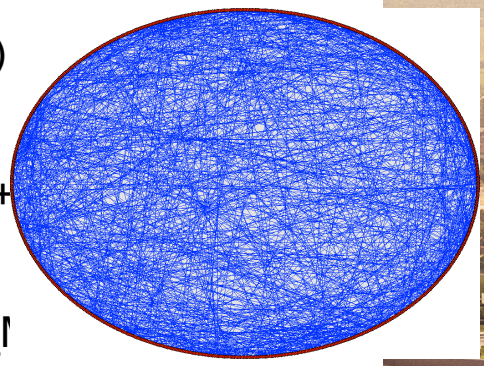
- Coords coords_set(MCNUM x, MCNUM y, MCNUM z);
- Coords coords_get(Coords a, MCNUM *x, MCNUM *y, MCNUM *z);
- Coords coords_add(Coords a, Coords b);
- Coords coords_sub(Coords a, Coords b);
- Coords coords_neg(Coords a);
- Coords coords_scale(Coords b, double scale);
- double coords_sp(Coords a, Coords b);
- Coords coords_xp(Coords b, Coords c);
- void coords_print(Coords a);

- void rot_set_rotation(Rotation t, double phx, double phy, double phz);
- int rot_test_identity(Rotation t);
- void rot_mul(Rotation t1, Rotation t2, Rotation t3);
- void rot_copy(Rotation dest, Rotation src);
- void rot_transpose(Rotation src, Rotation dst);
- Coords rot_apply(Rotation t, Coords a);
- void mccoordschange(Coords a, Rotation t, double *x, double *y, double *z,
  - double *vx, double *vy, double *vz, double *time,
  - double *s1, double *s2);
- void normal_vec(double *nx, double *ny, double *nz,
  - double x, double y, double z);



Runtime - random numbers

- #define rand01() (((double)random())/((double)MC_RAND_MAX+1))
- #define randpm1() (((double)random()) / (((double)MC_RAND_MAX+
- #define rand0max(max) (((double)random()) / (((double)MC_RAND_M
- #define randminmax(min,max) (rand0max((max)-(min)) + (min))
- double randnorm(void);
- double randtriangle(void);



10480	15011	01536	02011	81647	91646	69179	14194	62590
22368	46573	25595	85393	30995	89198	27982	53402	93965
24130	48360	22527	97265	76393	64809	15179	24830	49340
42167	93093	06243	61680	07856	16376	39440	53537	71341
37570	39975	81837	16656	06121	91782	60468	81305	49684
77921	06907	11008	42751	27756	53498	18602	70659	90655
99562	72905	56420	69994	98872	31016	71194	18738	44013
96301	91977	05463	07972	18876	20922	94595	56869	69014
89579	14342	63661	10281	17453	18103	57740	84378	25331
85475	36857	53342	53988	53060	59533	38867	62300	08158
28918	69578	88231	33276	70997	79936	56865	05859	90106
63553	40961	48235	03427	49626	69445	18663	72695	52180
09429	93969	52636	92737	88974	33488	36320	17617	30015
10365	61129	87529	85689	48237	52267	67689	93394	01511
07119	97336	71048	08178	77233	13916	47564	81056	97735
51085	12765	51821	51259	77452	16308	60756	92144	49442
02368	21382	52404	60268	89368	19885	55322	44819	01188
01011	54092	33362	94904	31273	04146	18594	29852	71585
52162	53916	46369	58586	23216	14513	83149	98736	23495
07056	97628	33787	09998	42698	06691	76988	13602	51851
48663	91245	85828	14346	09172	30168	90229	04734	59193
54164	58492	22421	74103	47070	25306	76468	26384	58151
32639	32363	05597	24200	13363	38005	94342	28728	35806
29334	27001	87637	87308	58731	00256	45834	15398	46557
02488	33062	28834	07351	19731	92420	60952	61280	50001
81525	72295	04839	96423	24878	82651	66566	14778	76797

Runtime - detector-output

- DETECTOR_OUT(p0,p1,p2)

- DETECTOR_OUT_0D(t,p0,p1,p2)
- DETECTOR_OUT_1D(t,xl,yl,xvar,x1,x2,n,p0,p1,p2,f)
- DETECTOR_OUT_2D(t,xl,yl,x1,x2,y1,y2,m,n,p0,p1,p2,f)
- DETECTOR_OUT_3D(t,xl,yl,zl,xv,yv,zv,x1,x2,y1,y2,z1,z2,m,n,p,p0,p1,p2,f)
- DETECTOR_CUSTOM_HEADER(t)

- t is a filename-string

- double mcestimate_error(double N, double p1, double p2);

Runtime - component data

- NAME_CURRENT_COMP
- INDEX_CURRENT_COMP

```
COMPONENT Mono_Cradle = Arm()
  AT (0, 0, 5.2746) RELATIVE Source ROTATED (0, A1, 0) RELATIVE Source
```

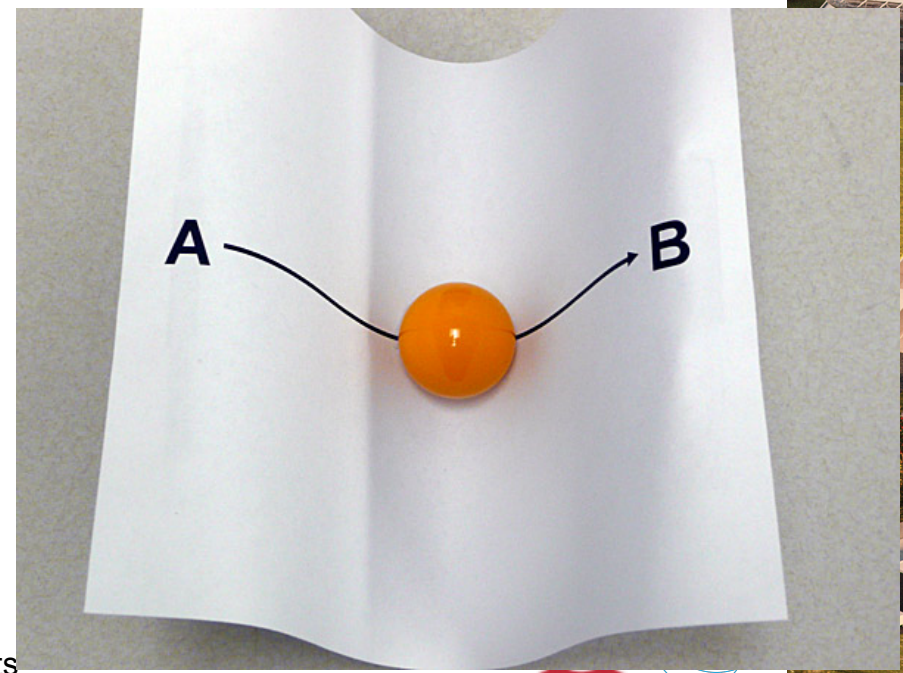
- POS_
- POS_R_CURRENT_COMP
- ROT_A_CURRENT_COMP
- ROT_R_CURRENT_COMP

Runtime - simulation flow / neutron state

- SCATTER - really means “please update the neutron state for visualization”
- ABSORB - really means “cancel further processing of this ray”
- STORE_NEUTRON(index, x, y, z, vx, vy, vz, t, sx, sy, sz, p)
- RESTORE_NEUTRON(index, x, y, z, vx, vy, vz, t, sx, sy, sz, p)
- - index should be INDEX_CURRENT_COMP

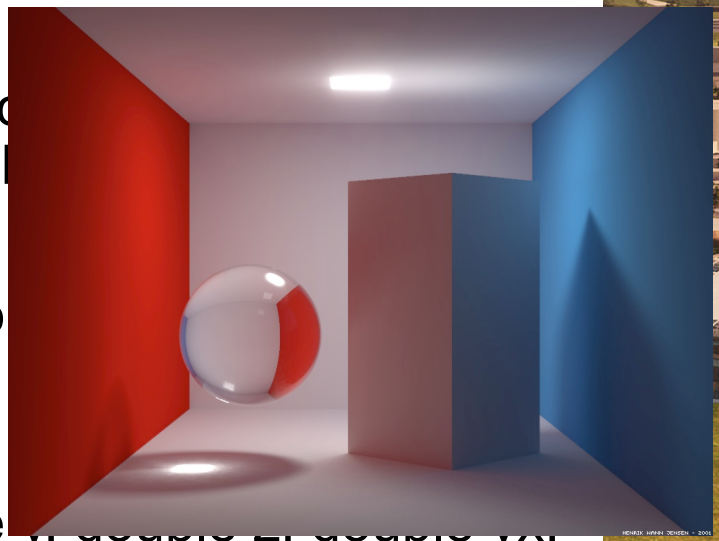
Runtime - propagation

- ALLOW_BACKPROP - allow negative time when propagating
- DISALLOW_BACKPROP - restore default behaviour
- PROP_DT(dt)
- PROP_X0 / Y0 / Z0



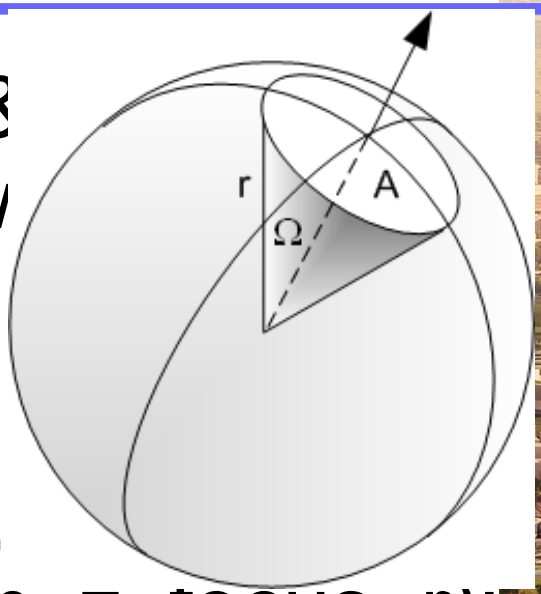
Runtime - space geometry

- int inside_rectangle(double, double, double, double)
- int box_intersect(double *dt_in, double *dt_out, double x, double y, double z, double vx, double vy, double vz, double dx, double dy, double dz)
- int cylinder_intersect(double *t0, double *t1, do
  double vx, double vy, double vz, double r, doul
- int sphere_intersect(double *t0, double *t1, do
  double vx, double vy, double vz, double r)
- int plane_intersect(double *t, double x, double
  double vy, double vz, double nx, double ny, double nz, double wx, double wy, double wz)
- int off_intersect(double*, double*, double, double, double, double, double, double, off_struct)



Solid angle “limitation” aka focusing

- randvec_target_rect(&vx, &vy, &
  &solid_angle, tx, ty, tz, focus_xw
  ROT_A_CURRENT_COMP);



- randvec_target_circle(&vx, &vy,
  &solid_angle, aim_x, aim_y, aim_z, focus_r);

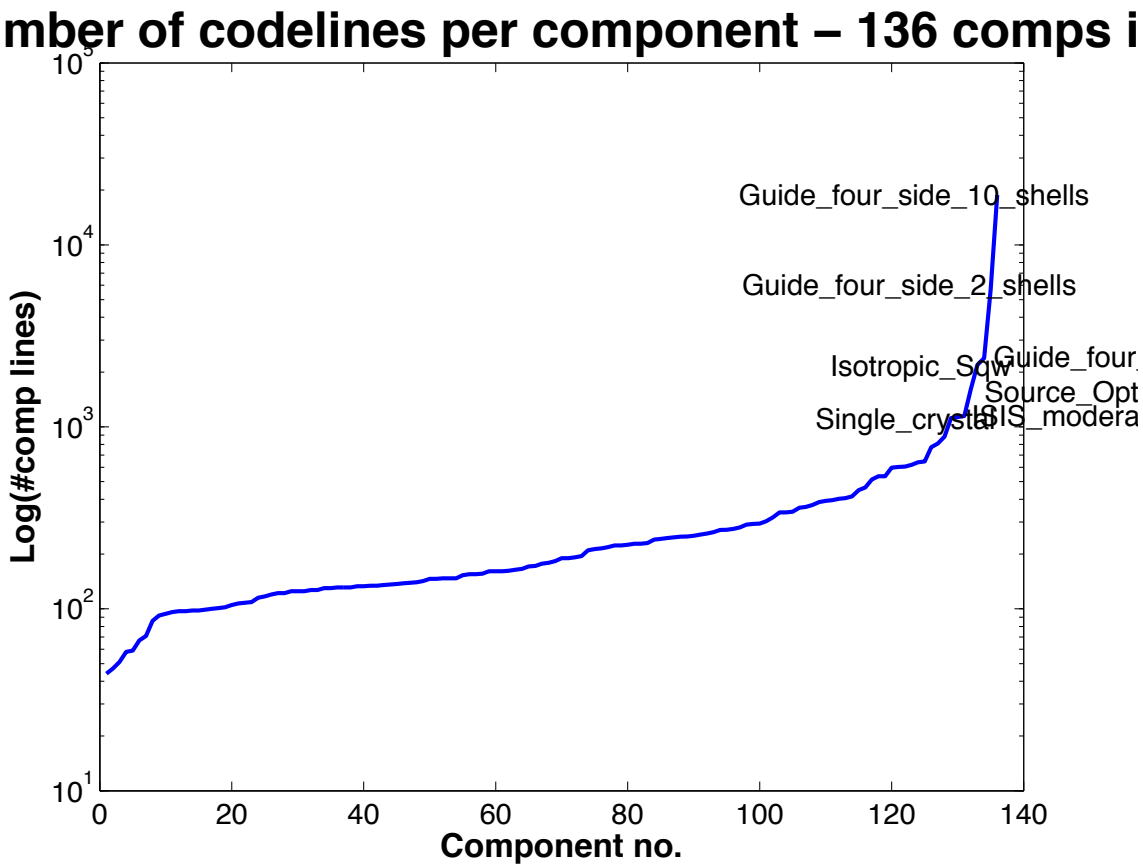
- randvec_target_rect_angular(&vx
  &solid_angle,aim_x, aim_y, aim_
  VarsV.ah, ROT_A_CURRENT_C

Various other stuff is there - all sorts...

- MC_GETPAR(comp, par)
- int solve_2nd_order(double *Idt,
  double A, double B, double C);
- Read any geometry using OFF format
- Load ascii tables
- Choose Larmor-precession algorithm
- ...

For inspiration...

- Check our long list of components and look inside... Most of them are quite simple and short... Statistics:



Shared libs are also not bad...

- 21 chopper_fermi.h
- 45 ref-lib.h
- 47 intersection.h
- 75 pol-lib.h
- 76 interoff-lib.h
- 82 adapt_tree-lib.h
- 85 ref-lib.c
- 118 vitess-lib.h
- 135 nexus-lib.h
- 156 adapt_tree-lib.c
- 174 read_table-lib.h
- 228 monitor_nd-lib.h
- 264 general.h
- 280 pol-lib.c
- 330 nexus-lib.c
- 498 vitess-lib.c
- 607 general.c
- 638 interoff-lib.c
- 699 chopper_fermi.c
- 912 intersection.c
- 991 mcstas-r.h
- 1101 read_table-lib.c
- 1688 monitor_nd-lib.c
- 5473 mcstas-r.c

McStas library - where is what?

- /usr/share/mcstas/2.2a on Unix, Linux
- /usr/local/mcstas/2.2a on Mac
- c:\McStas-2.2a\lib on Windows

- Contents:

- contrib (Contributed components)
- data (All kinds of data files)
- doc (home of the manual & component manual)
- editors (syntax highlighting for various editors)
- examples (example instrument files)
- misc (misc. components)
- monitors (monitor components)
- obsolete (obsolete components)
- optics (optics components)
- samples (sample components)
- share (shared libs, including mcstas-r.h/c)
- sources (source components)
- tools (perl scripts, matlab backend etc.)

Let's try writing a component!

- Starting point: Arm.comp
- Goal: Perfect mirror