Creating a new simulation

AMR: Adaptive mesh refinement

Root Grid

↓

1 Child Grid

↓

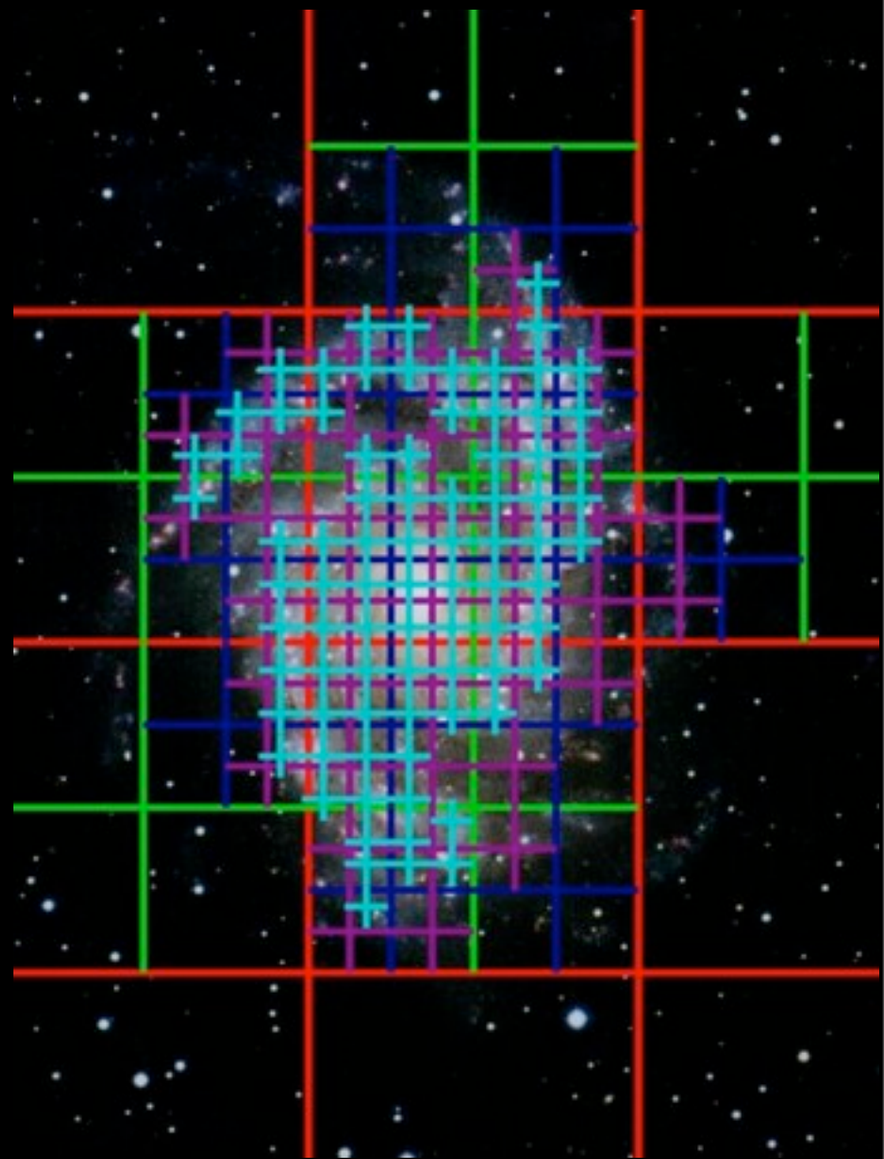
2 Child Grid

↓

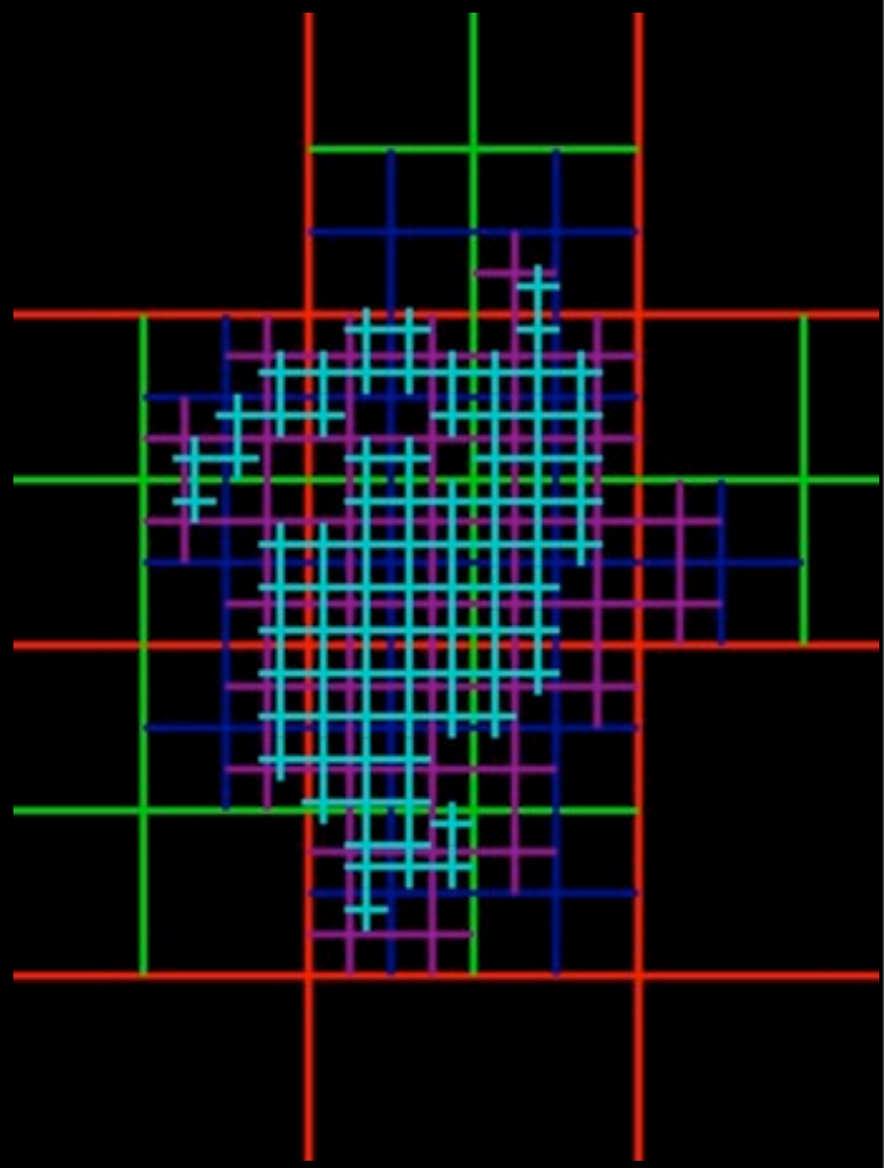
3 Child Grid

↓

4 Child Grid

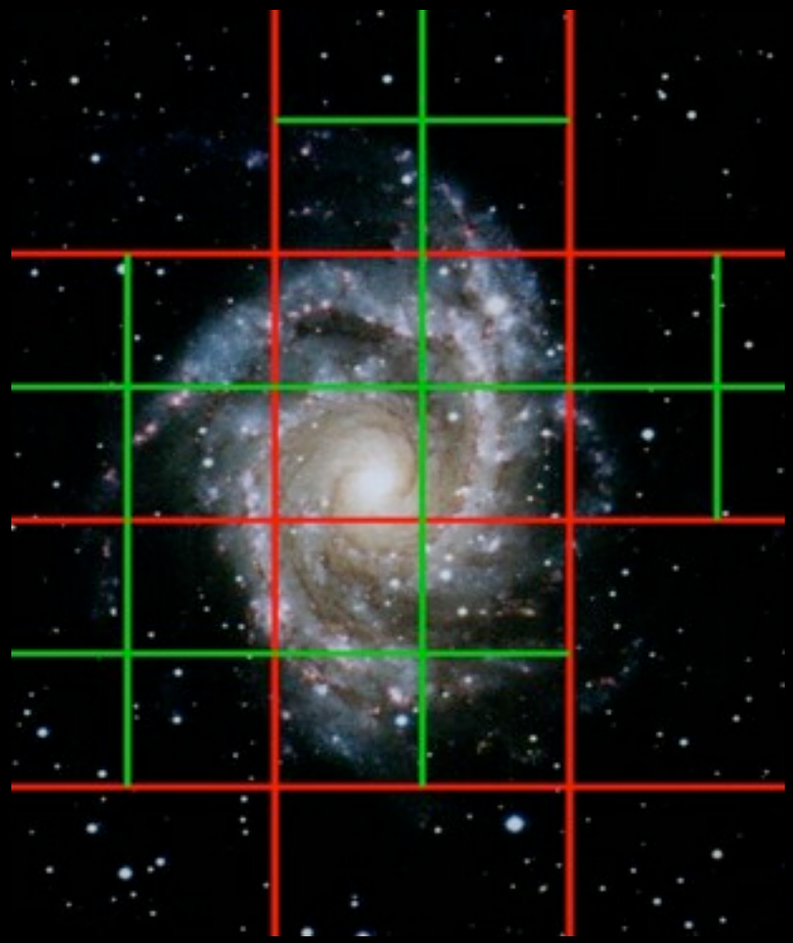


AMR: Adaptive mesh refinement

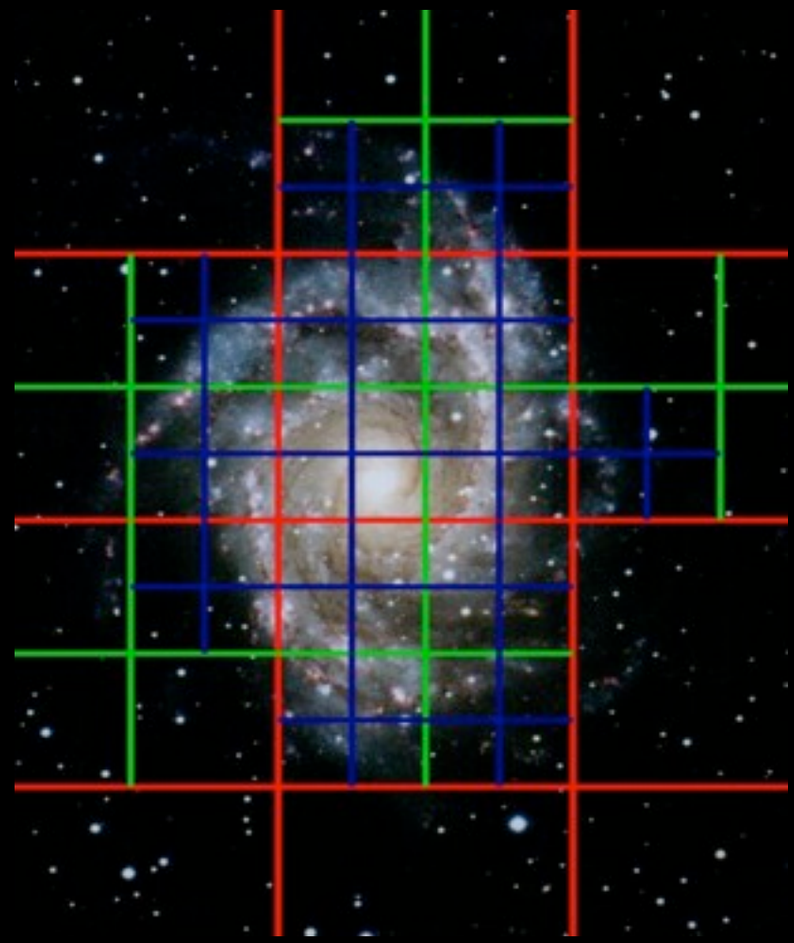


AMR: Adaptive mesh refinement

level 0



level 1



level 2

"Top Grid"

"Root Grid"

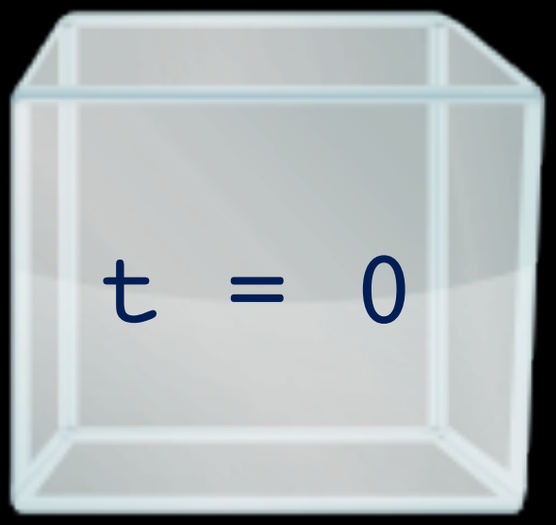
parent

"Child Grid"

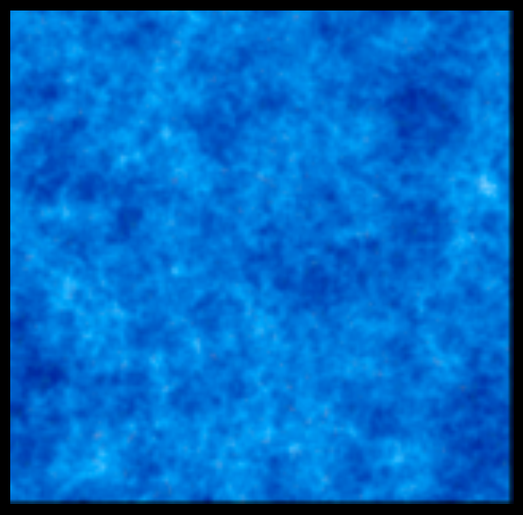
parent

"Child Grid"

Initial Conditions



Simulation box at start

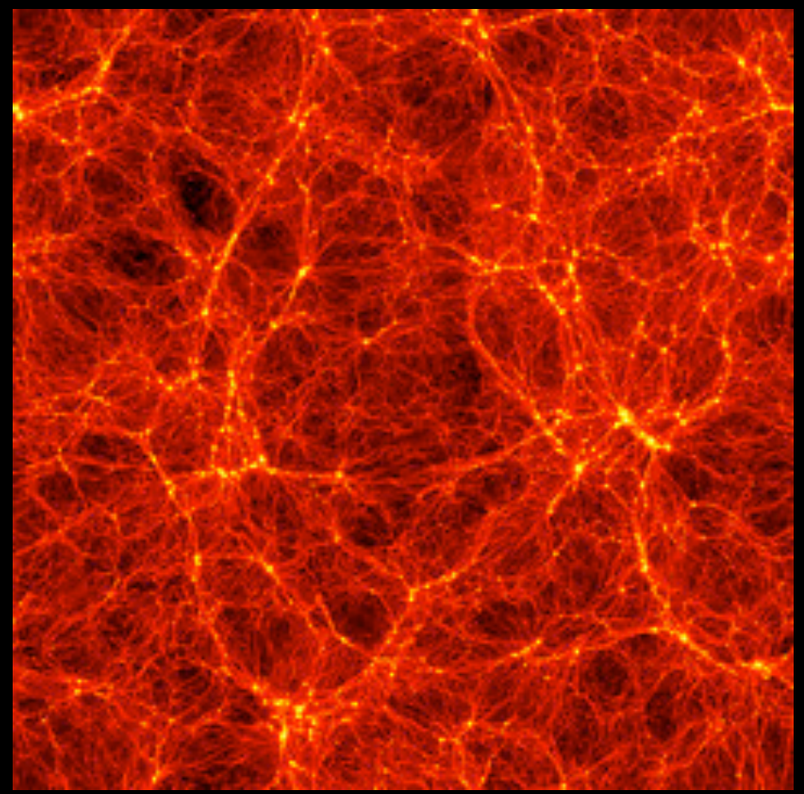


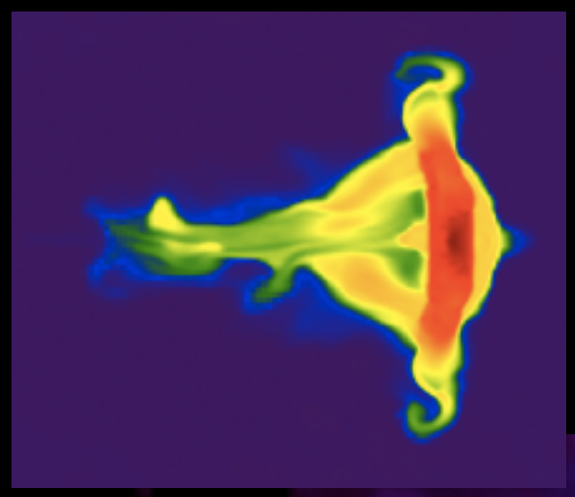
e.g. cosmology simulation

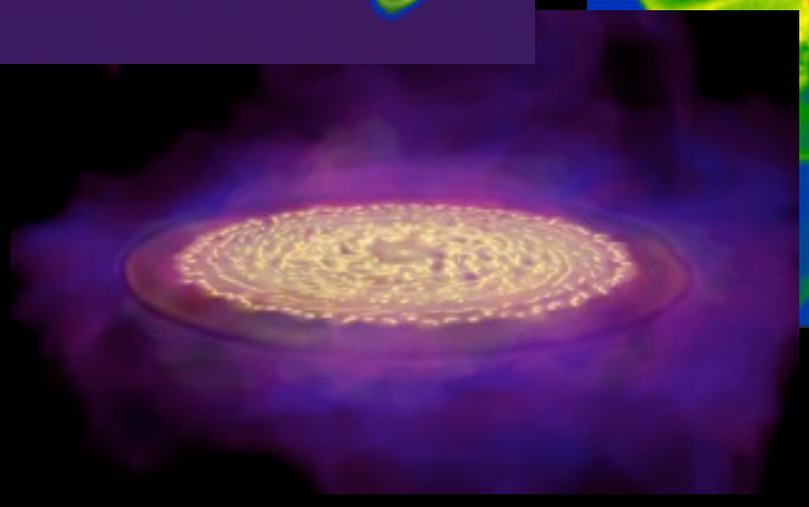
small density perturbations (changes)

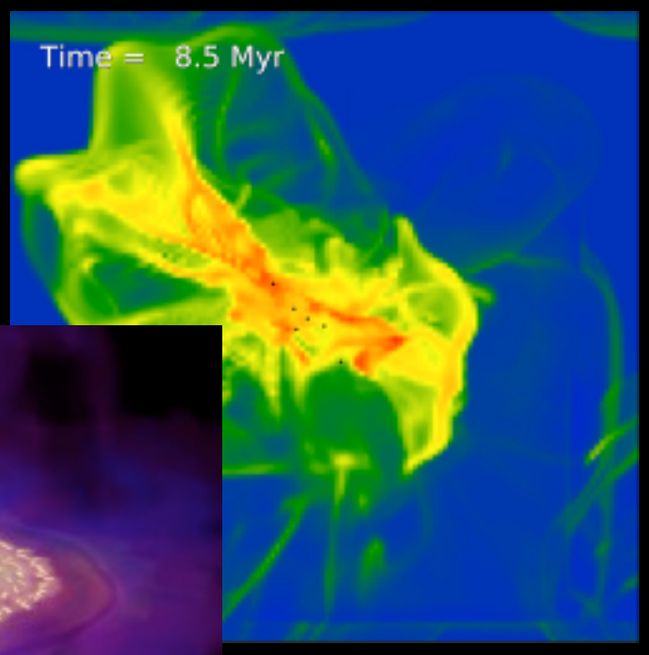
Simulations

A cosmology simulation is 1 choice...









But other simulations do not start at z = 50

e.g. galaxy discs, colliding clouds, star formation ...

Running Enzo

(1) enzo.C

Initial condition choices

(2) InitializeNew.C

ProblemType = 1

HydroShockTubesInitialize.C

(3)

ProblemType = 30

CosmologySimulationInitialize.C

(3)

ProblemType = ?

Your new simulation

(3)

Running Enzo

```
> cd enzo-stable/src/enzo

> ls *Initialize*.C
```

1. bash

```
CosmologySimulationInitialize.C
DoubleMachInitialize.C
ExternalBoundary_InitializeExternalBoundaryFaceIO.C
FOF_Initialize.C
FSMultiSourceInitialize.C
FSProb_Initialize.C
FastSiblingLocatorInitialize.C
FastSiblingLocatorInitializeStaticChainingMesh.C
FreeExpansionInitialize.C
GalaxySimulationInitialize.C
GravityEquilibriumTestInitialize.C
Grid_ClusterInitializeGrid.C
Grid_CollapseTestInitializeGrid.C
Grid_ConductionBubbleInitialize.C
Grid_ConductionCloudInitialize.C
Grid_ConductionTestInitialize.C
Grid_CoolingTestInitializeGrid.C
Grid_CosmoIonizationInitializeGrid.C
Grid_CosmologyInitializeParticles.C
Grid_CosmologySimulationInitializeGrid.C
Grid_DoubleMachInitializeGrid.C
Grid_FSMultiSourceInitializeGrid.C
Grid_FreeExpansionInitializeGrid.C
Grid_GalaxySimulationInitializeGrid.C
Grid_GravityEquilibriumTestInitializeGrid.C
Grid_HydroShockTubesInitializeGrid.C
Grid_ImplosionInitializeGrid.C
Grid_InitializeGravitatingMassField.C
Grid_InitializeGravitatingMassFieldParticles.C
Grid_InitializeRadiativeTransferFields.C
Grid_InitializeSource.C
Grid_InitializeUniformGrid.C
Grid_KHInitializeGrid.C
Grid_MHDBlastInitializeGrid.C
Grid_NestedCosmologySimulationInitializeGrid.C
Grid_NohInitializeGrid.C
Grid_OneZoneFreefallTestInitializeGrid.C
Grid_PhotonTestInitializeGrid.C
Grid_PhotonTestRestartInitializeGrid.C
Grid_PoissonSolverTestInitializeGrid.C
Grid_PressurelessCollapseInitialize.C
Grid_ProtostellarCollapseInitializeGrid.C
Grid_PutSinkRestartInitialize.C
Grid_RHIonizationClumpInitializeGrid.C
Grid_RHIonizationSteepInitializeGrid.C
Grid_RHIonizationTestInitializeGrid.C
Grid_RadHydroConstTestInitializeGrid.C
Grid_RadHydroGreyMarshakWaveInitializeGrid.C
Grid_RadHydroPulseTestInitializeGrid.C
Grid_RadHydroRadShockInitializeGrid.C
Grid_RadHydroStreamTestInitializeGrid.C
Grid_RadiatingShockInitializeGrid.C
Grid_RotatingCylinderInitialize.C
Grid_RotatingDiskInitializeGrid.C
Grid_SedovBlastInitializeGrid.C
Grid_SedovBlastInitializeGrid3D.C
Grid_ShearingBox2DInitializeGrid.C
Grid_ShearingBoxInitializeGrid.C
sleigh:enzo Elizabeth$
```

```
Grid_ZeldovichPancakeInitializeGrid.C
HydroShockTubesInitialize.C
ImplosionInitialize.C
InitializeCloudyCooling.C
InitializeEquilibriumCoolData.C
InitializeGadgetEquilibriumCoolData.C
InitializeHM12Photorates.C
InitializeLocal.C
InitializeLymanWernerTable.C
InitializeMovieFile.C
InitializeNew.C
InitializePythonInterface.C
InitializeRadiationFieldData.C
InitializeRadiativeTra
InitializeRateData.C
KHInitialize.C
MHDBlastInitialize.C
NestedCosmologySimul
NohInitialize.C
OneZoneFreefallTestI
PhotonTestInitialize.
PhotonTestRestartIni
PoissonSolverTestIni
PressurelessCollapse
ProtostellarCollapse
PutSinkRestartInitia
RHIonizationClumpIni
RHIonizationSteepIni
RHIonizationTestInitialize.C
RadHydroConstTestInitialize.C
RadHydroGreyMarshakWaveInitialize.C
RadHydroPulseTestInitialize.C
RadHydroRadShockInitialize.C
RadHydroStreamTestInitialize.C
RadiatingShockInitialize.C
RadiativeTransferInitialize.C
RotatingCylinderInitialize.C
RotatingDiskInitialize.C
SedovBlastInitialize.C
ShearingBox2DInitialize.C
ShearingBoxInitialize.C
ShearingBoxStratifiedInitialize.C
ShockInABoxInitialize.C
ShockPoolInitialize.C
SphericalInfallInitialize.C
StarParticleInitialize.C
StarParticlePopIII_IMFInitialize.C
StratifiedMediumExplosionInitialize.C
SupernovaRestartInitialize.C
TestGravityInitialize.C
TestGravitySphereInitialize.C
TestOrbitInitialize.C
TurbulenceSimulationInitialize.C
WavePoolInitialize.C
ZeldovichPancakeInitialize.C
gFLDProblem_Initialize.C
gFLDSplit_Initialize.C
```

Different simulation initial conditions

SimulationInitialize.C

Grid_SimulationInitializeGrid.C

e.g.

ZeldovichPancakeInitialize.C

Grid_ZeldovichPancakeInitialize.C

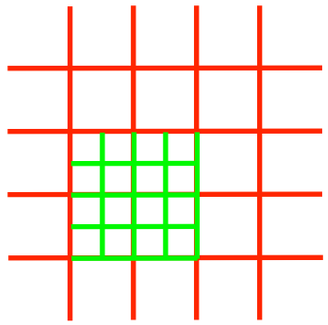
ProblemType = 12
StarParticleCreation = 1
StarParticleFeedback = 1
StaticHierarchy = 0
.
.
.

enzo.exe param.enzo

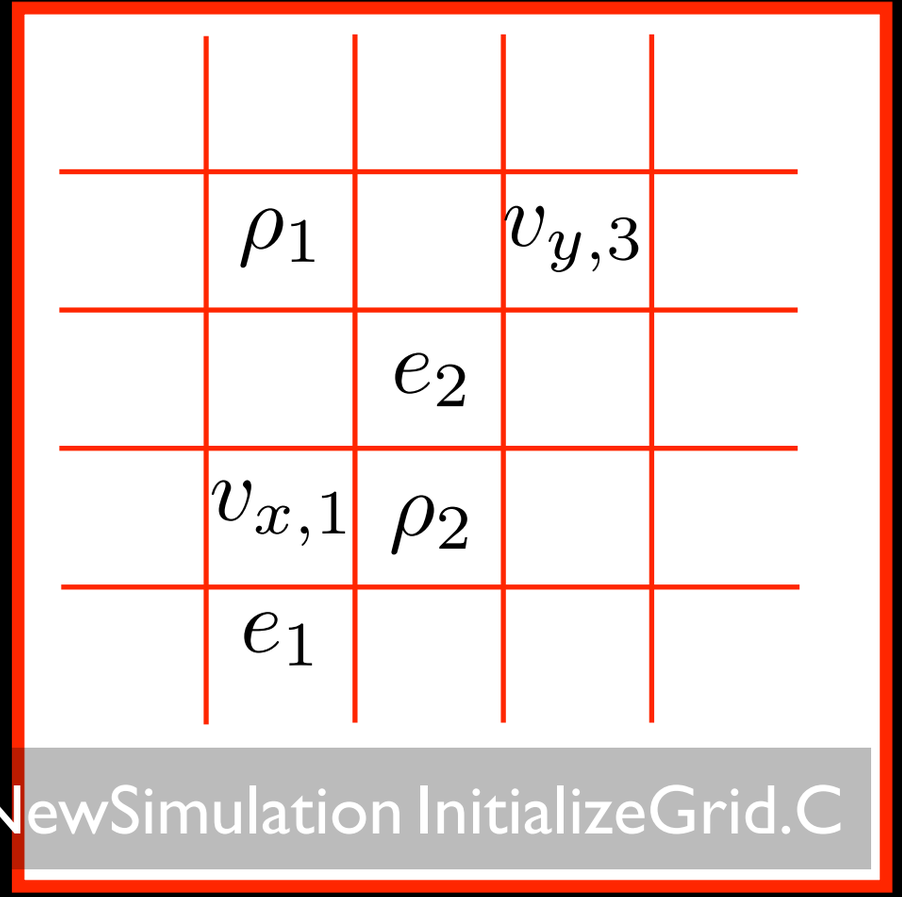
NewSimulationInitialize.C

Read parameters

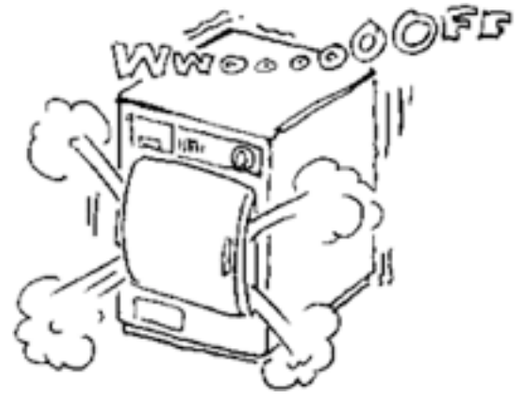
Set up
grid levels



ProblemType = 12
StarParticleCreation = 1
StarParticleFeedback = 1
StaticHierarchy = 0
.
.
.

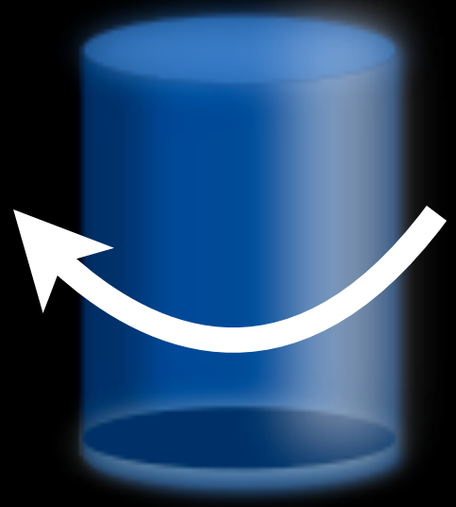


Grid_NewSimulation InitializeGrid.C



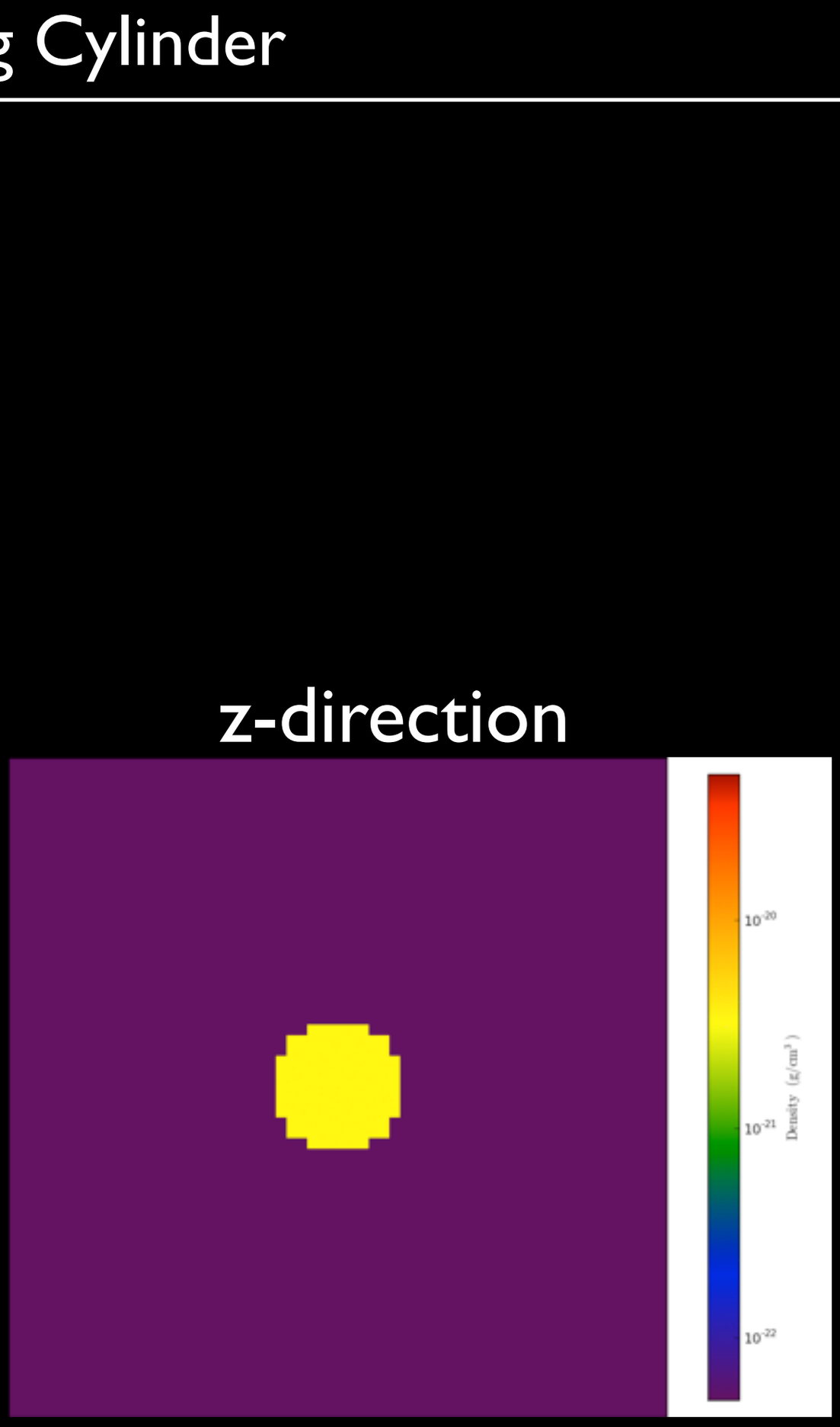
Enzo

e.g. Rotating Cylinder



x-direction

z-direction



e.g. Rotating Cylinder

on conival:

Copy:

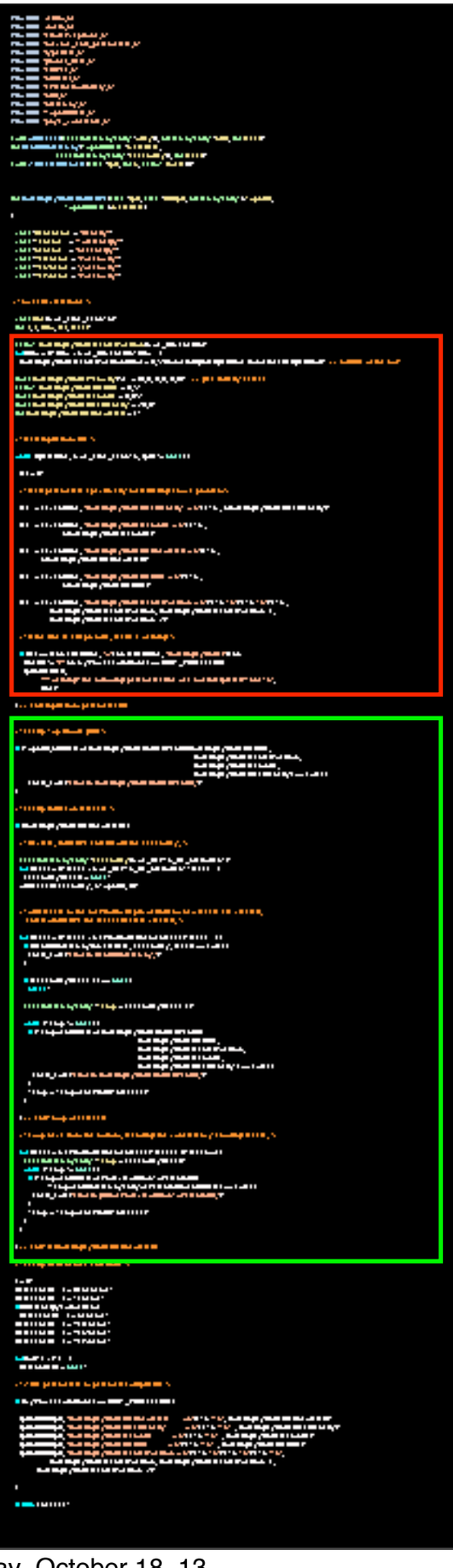
```
> cp /home/tasker/workshop2013/workshop/
NewRotatingCylinderInitialize.C .

> cp /home/tasker/workshop2013/workshop/
Grid_NewRotatingCylinderInitializeGrid.C .

> cp /home/tasker/workshop2013/workshop/
NewRotatingCylinder.enzo .
```

Read:

```
> emacs -nw NewRotatingCylinderInitialize.C
```

NewRotatingCylinderInitialize.C

(called by InitializeNew.C)



Read parameters

Set up grid levels

2 choices for setting up initial grids

Adaptive meshes

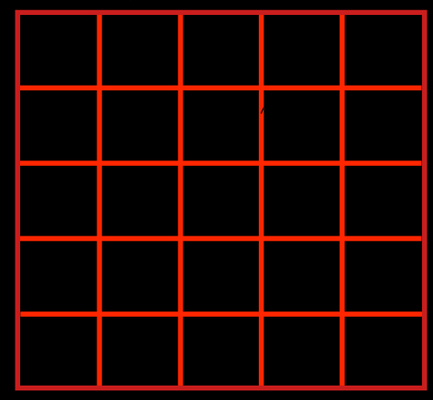
Child grid location based on parent grid properties

ρ_1 ρ_1 ρ_1 ρ_1 ρ_1
ρ_1 ρ_1 ρ_1 ρ_1 ρ_1
ρ_1 ρ_1 ρ_1 ρ_1 ρ_1
ρ_1 ρ_1 ρ_1 ρ_1 ρ_1
ρ_1 ρ_1 ρ_1 ρ_1 ρ_1

Subgrids

Child grid location based on fixed position

(x_l, y_l, z_l)



(x_r, y_r, z_r)

NewRotatingCylinderInitialize.C

```
FLOAT RotatingCylinderCenterPosition[MAX_DIMENSION];
for(dim = 0; dim < MAX_DIMENSION; dim++)
  RotatingCylinderCenterPosition[dim] = 0.5*(DomainRightEdge[dim]+DomainLeftEdge[dim]);  // middle of the box

float RotatingCylinderVelocity[3]   = {0.0, 0.0, 0.0};   // gas initally at rest
FLOAT RotatingCylinderRadius = 0.3;
float RotatingCylinderLambda = 0.05;
float RotatingCylinderOverdensity = 20.0;
int RotatingCylinderRefineAtStart = 1;
```

Set defaults

```
/* read input from file */

while (fgets(line, MAX_LINE_LENGTH, fptr) != NULL) {

  ret = 0;

  /* read parameters specifically for radiating shock problem*/

  ret += sscanf(line, "RotatingCylinderOverdensity  = %"FSYM, &RotatingCylinderOverdensity);

  ret += sscanf(line, "RotatingCylinderLambda = %"FSYM,
                &RotatingCylinderLambda);

  ret += sscanf(line, "RotatingCylinderRefineAtStart = %"ISYM,
                &RotatingCylinderRefineAtStart);

  ret += sscanf(line, "RotatingCylinderRadius = %"PSYM,
                &RotatingCylinderRadius);

  ret += sscanf(line, "RotatingCylinderCenterPosition = %"PSYM" %"PSYM" %"PSYM,
                RotatingCylinderCenterPosition, RotatingCylinderCenterPosition+1,
                RotatingCylinderCenterPosition+2);
```

Read problem-specific parameters

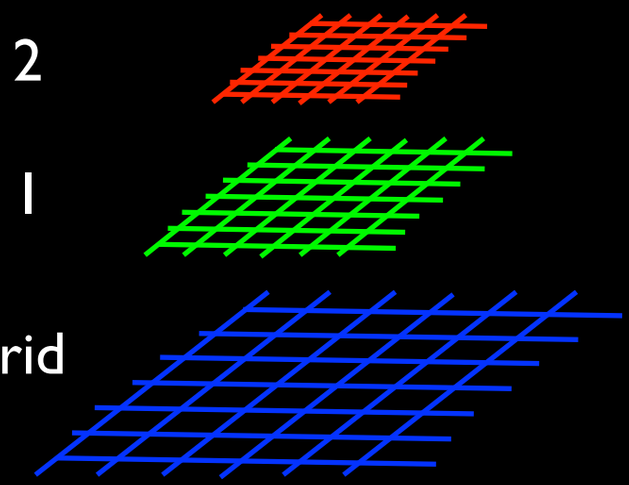
```
  /* if the line is suspicious, issue a warning */

  if (ret == 0 && strstr(line, "=") && (strstr(line, "RotatingCylinder")) &&
    line[0] != '#' && MyProcessorNumber == ROOT_PROCESSOR)
    fprintf(stderr,
            "*** warning: the following parameter line was not interpreted:\n%s\n",
            line);

} // end input from parameter file
```

Check we've not missed any

```
| /* Set up Top (Root)  grid */

  if (TopGrid.GridData->RotatingCylinderInitializeGrid(RotatingCylinderRadius,
                                                       RotatingCylinderCenterPosition,
                                                       RotatingCylinderLambda,
                                                       RotatingCylinderOverdensity) == F
      ENZO_FAIL("Error in RotatingCylinderInitializeGrid.");
  }

  /* Set up initial AMR levels */

  if (RotatingCylinderRefineAtStart) {

    /* Declare, initialize and fill out the LevelArray. */

    LevelHierarchyEntry *LevelArray[MAX_DEPTH_OF_HIERARCHY];
    for (level = 0; level < MAX_DEPTH_OF_HIERARCHY; level++)
      LevelArray[level] = NULL;
    AddLevel(LevelArray, &TopGrid, 0);


    /* Add levels to the maximum depth or until no new levels are created,
       and re-initialize the level after it is created. */

    for (level = 0; level < MaximumRefinementLevel; level++) {
      if (RebuildHierarchy(&MetaData, LevelArray, level) == FAIL) {
        ENZO_FAIL("Error in RebuildHierarchy.");
      }

      if (LevelArray[level+1] == NULL)
        break;

      LevelHierarchyEntry *Temp = LevelArray[level+1];

      while (Temp != NULL) {
        if (Temp->GridData->RotatingCylinderInitializeGrid(
                                        RotatingCylinderRadius,
                                        RotatingCylinderCenterPosition,
                                        RotatingCylinderLambda,
                                        RotatingCylinderOverdensity ) == FAIL) {
          ENZO_FAIL("Error in RotatingCylinderInitializeGrid.");
        }
        Temp = Temp->NextGridThisLevel;
      }

    } // end: loop over levels
```

Set $\rho, e, \bar{v}$ cells in top grid

Create AMR hierarchy



Set $\rho, e, \bar{v}$ cells in child grid

Largely identical for all problem types

Grid_NewRotatingCylinderInitializeGrid.C

(called by NewRotatingCylinderInitialize.C)

```
> emacs -nw Grid_NewRotatingCylinderInitializeGrid.C
```

Assign memory for $\rho, e, \bar{v}$ fields

Set $\rho, e, \bar{v}$ for each cell

Grid_NewRotatingCylinderInitializeGrid.C

```
/* create fields */

NumberOfBaryonFields = 0;
FieldType[NumberOfBaryonFields++] = Density;
FieldType[NumberOfBaryonFields++] = TotalEnergy;
if (DualEnergyFormalism)
  FieldType[NumberOfBaryonFields++] = InternalEnergy;
int vel = NumberOfBaryonFields;
FieldType[NumberOfBaryonFields++] = Velocity1;
FieldType[NumberOfBaryonFields++] = Velocity2;
FieldType[NumberOfBaryonFields++] = Velocity3;
```

Create $\rho, e, \bar{v}$ fields

```
if (ProcessorNumber != MyProcessorNumber)
  return SUCCESS;
```

Only do this on 1 processor

```
/* declarations */

FLOAT x = 0, y = 0, z = 0, radius, z_distance, x_velocity = 0.0, y_velocity = 0.0, z_velocity = 0.0;
float sintheta, costheta, omega;
float outside_density = 1.0, outside_energy = 0.5, density = 1.0, energy = 0.5;
int i, j, k, dim, cellindex;

/* compute size of fields */

int size = 1;
for (dim = 0; dim < GridRank; dim++)
  size *= GridDimension[dim];

/* allocate fields */

int field;
for (field = 0; field < NumberOfBaryonFields; field++)
  if (BaryonField[field] == NULL)
    BaryonField[field] = new float[size];
```

Assign memory for fields

```
int DensNum, GENum, TENum, Vel1Num, Vel2Num, Vel3Num, MetalNum;
if (this->IdentifyPhysicalQuantities(DensNum, GENum, Vel1Num, Vel2Num,
                                     Vel3Num, TENum) == FAIL) {
  ENZO_FAIL("Error in IdentifyPhysicalQuantities.\n");
}
```

Useful function for finding fields

```
for (k = 0; k < GridDimension[2]; k++)
 for (j = 0; j < GridDimension[1]; j++)
  for (i = 0; i < GridDimension[0]; i++) {
   cellindex = i + j*GridDimension[0] + k*GridDimension[0]*GridDimension[1];
```

Loop over cells

```
   energy = outside_energy;
   density = outside_density;
```

set background values

```
   x = CellLeftEdge[0][i] + 0.5*CellWidth[0][i];
   y = CellLeftEdge[1][j] + 0.5*CellWidth[1][j];
   z = CellLeftEdge[2][k] + 0.5*CellWidth[2][k];

   /* Find distance from center. */

   radius = POW(x-RotatingCylinderCenterPosition[0], 2.0) +
    POW(y-RotatingCylinderCenterPosition[1], 2.0);

   radius = sqrt(radius);  // ok, now it's just radius

   z_distance = fabs(z-RotatingCylinderCenterPosition[2]);
```

Cell position

```
   if ( (radius <= RotatingCylinderRadius) && (z_distance <= RotatingCylinderRadius) ) {
    // inside the cylinder

    density = outside_density * RotatingCylinderOverdensity;

    sintheta = (y-RotatingCylinderCenterPosition[1])/radius;
    costheta = (x-RotatingCylinderCenterPosition[0])/radius;

    // x,y and z velocity.
    x_velocity = -1.0*sintheta*omega*radius;
    y_velocity = costheta*omega*radius;
    z_velocity = 0.0;

    energy = outside_energy / RotatingCylinderOverdensity;

   } // if (r <= RotatingCylinderRadius)
```

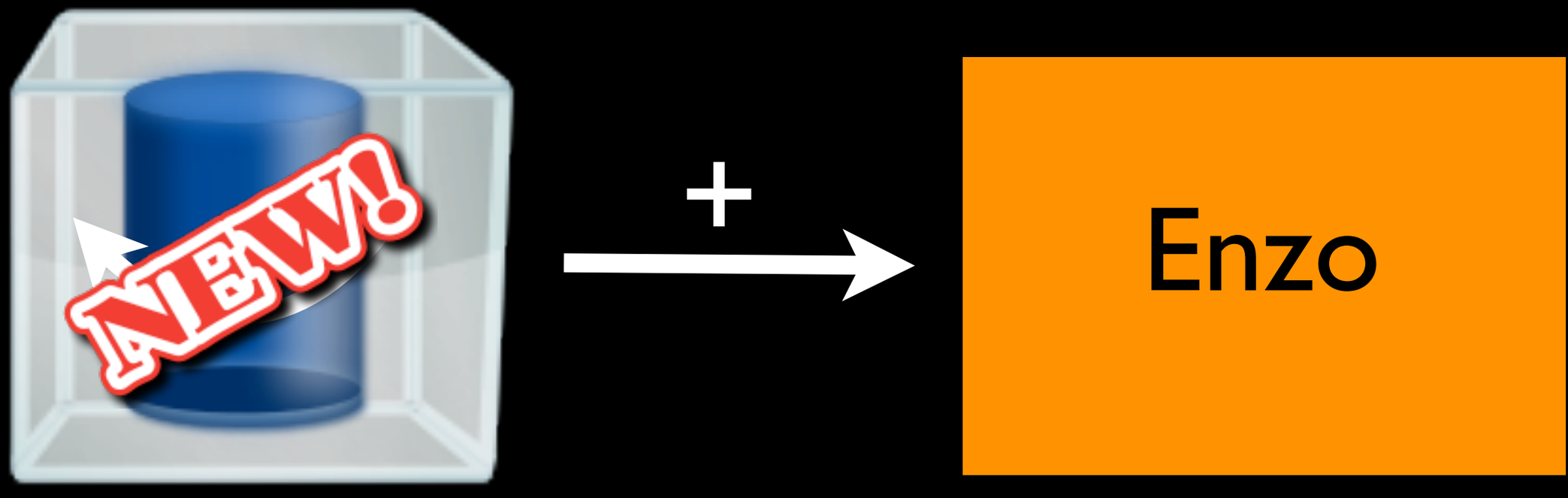
$\rho, e, \bar{v}$ inside cylinder

```
   BaryonField[DensNum][cellindex] = density;

   BaryonField[Vel1Num][cellindex] = x_velocity;
   BaryonField[Vel2Num][cellindex] = y_velocity;
   BaryonField[Vel3Num][cellindex] = z_velocity;

   BaryonField[TENum][cellindex] = energy;
```

Set final field value

Add to Enzo

Now we must add this simulation to Enzo



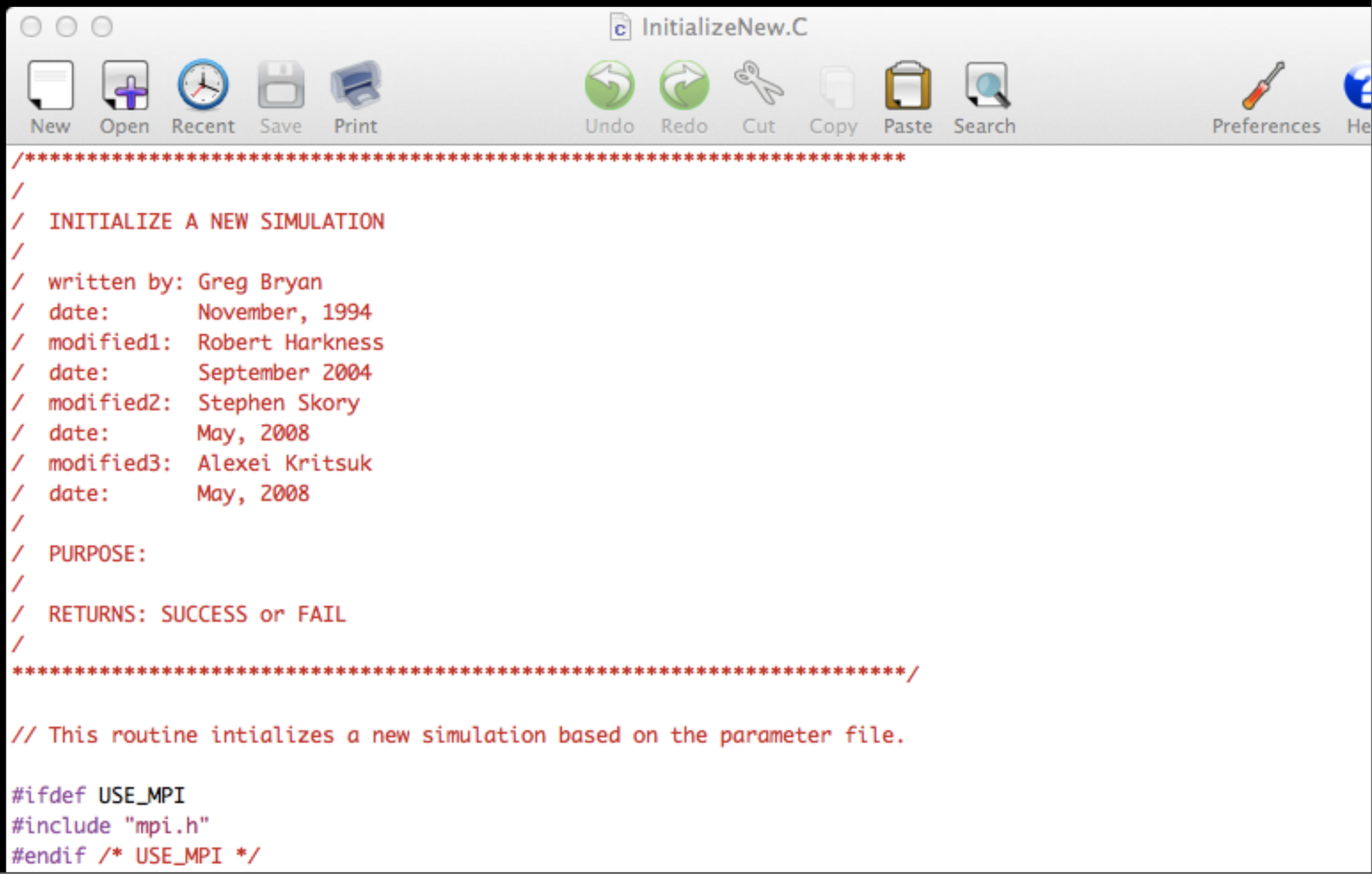
(1) put files in code directory:

```
> mv NewRotatingCylinderInitialize.C ~/enzo-stable/src/
enzo/.

> mv Grid_NewRotatingCylinderInitializeGrid.C ~/enzo-
stable/src/enzo/.
```

Add to Enzo

(2) add to InitializeNew.C

```
> cd ~/enzo-dev/src/enzo
emacs -nw InitializeNew.C
```



Add to Enzo

(2) add to InitializeNew.C

InitializeNew.C

New Open Recent Save Print Undo Redo Cut Copy Paste Search Preferences Help

```
int HydroShockTubesInitialize(FILE *fptr, FILE *Outfptr,
                              HierarchyEntry &TopGrid, TopGridData &MetaData);
int WavePoolInitialize(FILE *fptr, FILE *Outfptr, HierarchyEntry &TopGrid,
                       TopGridData &MetaData);
int ShockPoolInitialize(FILE *fptr, FILE *Outfptr, HierarchyEntry &TopGrid,
                        TopGridData &MetaData);
int DoubleMachInitialize(FILE *fptr, FILE *Outfptr, HierarchyEntry &TopGrid,
                         TopGridData &MetaData, ExternalBoundary &Exterior);
int ShockInABoxInitialize(FILE *fptr, FILE *Outfptr, HierarchyEntry &TopGrid,
                          TopGridData &MetaData, ExternalBoundary &Exterior);
int ImplosionInitialize(FILE *fptr, FILE *Outfptr, HierarchyEntry &TopGrid,
                        TopGridData &MetaData);
int RotatingCylinderInitialize(FILE *fptr, FILE *Outfptr, HierarchyEntry &TopGrid,
                               TopGridData &MetaData);
int RotatingDiskInitialize(FILE *fptr, FILE *Outfptr, HierarchyEntry &TopGrid,
                           TopGridData &MetaData);
int ConductionTestInitialize(FILE *fptr, FILE *Outfptr, HierarchyEntry &TopGrid,
                             TopGridData &MetaData);
int ConductionBubbleInitialize(FILE *fptr, FILE *Outfptr, HierarchyEntry &TopGrid,
                               TopGridData &MetaData);
int ConductionCloudInitialize(FILE *fptr, FILE *Outfptr, HierarchyEntry &TopGrid,
                              TopGridData &MetaData);
int StratifiedMediumExplosionInitialize(FILE *fptr, FILE *Outfptr, HierarchyEntry &TopGrid,
                                        TopGridData &MetaData);
int KHInitialize(FILE *fptr, FILE *Outfptr, HierarchyEntry &TopGrid,
                 TopGridData &MetaData);
int NohInitialize(FILE *fptr, FILE *Outfptr, HierarchyEntry &TopGrid,
                  TopGridData &MetaData);
int SedovBlastInitialize(FILE *fptr, FILE *Outfptr, HierarchyEntry &TopGrid,
                         TopGridData &MetaData);
```

-:--- **InitializeNew.C** 5% (52,0) (C++/l Abbrev)

"function prototypes"

List of new simulations

Add to Enzo

(2) add to InitializeNew.C

```
> emacs NewRotatingCylinderInitialize.C
```

```
#include "Hierarchy.h"
#include "TopGridData.h"
#include "phys_constants.h"

void AddLevel(LevelHierarchyEntry *Array[], HierarchyEntry *Grid, int level);
int RebuildHierarchy(TopGridData *MetaData,
                     LevelHierarchyEntry *LevelArray[], int level);
void WriteListOfFloats(FILE *fptr, int N, FLOAT floats[]);



int NewRotatingCylinderInitialize(FILE *fptr, FILE *Outfptr, HierarchyEntry &TopGrid,
                          TopGridData &MetaData)
{

  char *DensName = "Density";
  char *TEName    = "TotalEnergy";
  char *GEName    = "GasEnergy";
  char *Vel1Name = "x-velocity";
  char *Vel2Name = "y-velocity";
  char *Vel3Name = "z-velocity";


 /* local declarations */

  char line[MAX_LINE_LENGTH];
  int  i, j, dim, ret, level;

  FLOAT RotatingCylinderCenterPosition[MAX_DIMENSION];
```

copy

NewRotatingCylinderInitialize.C

Add to Enzo

(2) add to InitializeNew.C

In InitializeNew.C

```
int TurbulenceSimulationReInitialize(HierarchyEntry *TopGrid,
                                     TopGridData &MetaData);

int TracerParticleCreation(FILE *fptr, HierarchyEntry &TopGrid,
                           TopGridData &MetaData);

int ShearingBoxInitialize(FILE *fptr, FILE *Outfptr, HierarchyEntry &TopGrid,
                          TopGridData &MetaData);
int ShearingBox2DInitialize(FILE *fptr, FILE *Outfptr, HierarchyEntry &TopGrid,
                          TopGridData &MetaData);
int ShearingBoxStratifiedInitialize(FILE *fptr, FILE *Outfptr, HierarchyEntry &TopGrid,
                          TopGridData &MetaData);

int NewRotatingCylinderInitialize(FILE *fptr, FILE *Outfptr, HierarchyEntry &TopGrid,
                                  TopGridData &MetaData);

#ifdef TRANSFER
int PhotonTestInitialize(FILE *fptr, FILE *Outfptr,
                         HierarchyEntry &TopGrid, TopGridData &MetaData,
                         bool Reinitialize=false);
int PhotonTestRestartInitialize(FILE *fptr, FILE *Outfptr,
                                HierarchyEntry &TopGrid, TopGridData &MetaData,
                                ExternalBoundary &Exterior);
int FSMultiSourceInitialize(FILE *fptr, FILE *Outfptr,
```

paste

InitializeNew.C

Add to Enzo

(2) add to InitializeNew.C

InitializeNew.C

New Open Recent Save Print Undo Redo Cut Copy Paste Search Preferences Help

```
// 1) Shocktube problem

if (ProblemType == 1)
  ret = HydroShockTubesInitialize(fptr, Outfptr, TopGrid, MetaData);

// 2) Wave pool

if (ProblemType == 2)
  ret = WavePoolInitialize(fptr, Outfptr, TopGrid, MetaData);

// 3) Shock pool

if (ProblemType == 3)
  ret = ShockPoolInitialize(fptr, Outfptr, TopGrid, MetaData);

// 4) Double Mach reflection

if (ProblemType == 4)
  ret = DoubleMachInitialize(fptr, Outfptr, TopGrid, MetaData, Exterior);

// 5) ShockInABox

if (ProblemType == 5)
  ret = ShockInABoxInitialize(fptr, Outfptr, TopGrid, MetaData, Exterior);

// 6) Implosion

if (ProblemType == 6)
  ret = ImplosionInitialize(fptr, Outfptr, TopGrid, MetaData);
```

-:--- **InitializeNew.C** 44% (389,0) (C++/l Abbrev)

Give number to simulation

Add to Enzo

(2) add to InitializeNew.C

InitializeNew.C

New Open Recent Save Print Undo Redo Cut Copy Paste Search Preferences Help

untitled 1 | InitializeNew.C 2 | NewRotatingCylinderInitialize.C 3

```
// 13) RotatingDisk
if (ProblemType == 13)
  ret = RotatingDiskInitialize(fptr, Outfptr, TopGrid, MetaData);


// 14) NewRotatingCyclinder
if (ProblemType == 14)
  ret = NewRotatingCylinderInitialize(fptr, Outfptr, TopGrid, MetaData);


// 20) Zeldovich Pancake

if (ProblemType == 20)
  ret = ZeldovichPancakeInitialize(fptr, Outfptr, TopGrid, MetaData);

// 21) 1D Pressureless collapse

if (ProblemType == 21)
  ret = PressurelessCollapseInitialize(fptr, Outfptr, TopGrid, MetaData);

// 22) Adiabatic expansion

if (ProblemType == 22)
  ret = AdiabaticExpansionInitialize(fptr, Outfptr, TopGrid);

// 23) GravityTest

if (ProblemType == 23)
  ret = TestGravityInitialize(fptr, Outfptr, TopGrid, MetaData);
```

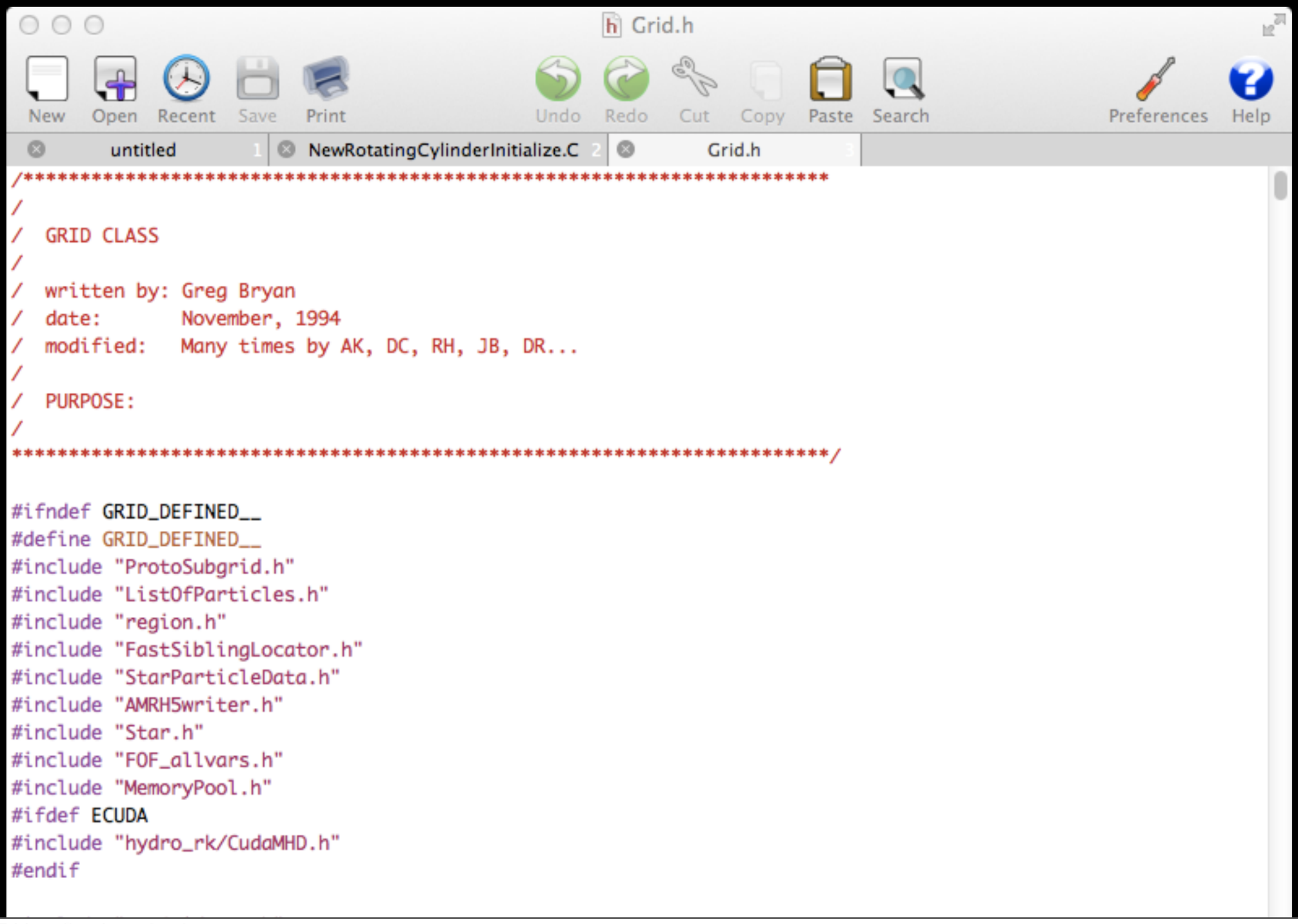
-:**- InitializeNew.C 49% (446,0) (C++/l Abbrev)

add
~ line 445

Add to Enzo

(3) add to Grid.h

```
> emacs Grid.h
```



Add to Enzo

(3) add to Grid.h

```
                                  FLOAT RadiatingShockSedovBlastRadius,
                                  float RadiatingShockEnergy,
                                  float RadiatingShockPressure,
                                  float RadiatingShockKineticEnergyFraction,
                                  float RadiatingShockRhoZero,
                                  float RadiatingShockVelocityZero,
                                  int RadiatingShockRandomSeedInitialize,
                                  FLOAT RadiatingShockCenterPosition[MAX_DIMENSION]);

/* Initialize a grid for a rotating cylinder collapse */
int RotatingCylinderInitializeGrid(FLOAT RotatingCylinderRadius,
                                   FLOAT RotatingCylinderCenterPosition[MAX_DIMENSION],
                                   float RotatingCylinderLambda,
                                   float RotatingCylinderOverdensity);
```

```
/* Initialize a new grid for a rotating cylinder collapse */
int NewRotatingCylinderInitializeGrid(FLOAT RotatingCylinderRadius,
                                      FLOAT RotatingCylinderCenterPosition[MAX_DIMENSION],
                                      float RotatingCylinderLambda,
                                      float RotatingCylinderOverdensity);
```

add ~ line 1820

```
int RotatingDiskInitializeGrid(float RDScaleRadius,
                               float RDScaleHeight,
                               float RDTemperature,
                               float RDDMConcentration,
                               float RDTotalDMMass,
                               float RDCentralDensity,
                               float RDOuterEdge);
```

Grid.h

Add to Enzo

(3) add to Make.config.objects

```
> emacs Make.config.objects
```

```
#=======================================================================
#
# FILE:        Make.config.objects
#
# DESCRIPTION: Make include file defining OBJS_MAIN
#
# AUTHOR:      James Bordner (jobordner@ucsd.edu)
#
# DATE:        2007-02-21
#
#=======================================================================

#-----------------------------------------------------------------------
# Default Enzo object files
#-----------------------------------------------------------------------

OBJS_CONFIG_LIB = \
        acml_st1.o \
        AdiabaticExpansionInitialize.o \
        AdjustRefineRegion.o \
        AdjustMustRefineParticlesRefineToLevel.o \
        AMRH5writer.o \
        AnalysisBaseClass.o \
        AnalysisBaseClass_HDF5Utils.o \
        arccosh.o \
        arcsinh.o \
        AssignGridToTaskMap.o \
        auto_show_compile_options.o \
        auto_show_config.o \
        auto_show_flags.o \
```

Add to Enzo

(3) add to Make.config.objects

```
MakeFieldConservative.o\
MemoryAllocationRoutines.o \
MemoryPoolRoutines.o \
MersenneTwister.o \
mg_calc_defect.o \
mg_prolong2.o \
mg_prolong.o \
mg_relax.o \
mg_restrict.o \
mkl_st1.o \
Mpich_V1_Dims_create.o \
multi_cool.o \
MultigridSolver.o \
mused.o \
NestedCosmologySimulationInitialize.o \
NewRotatingCylinderInitialize.o \
ngpinterp.o \
ngp_deposit.o \
NohInitialize.o \
nr_1d.o \
nr_2d.o \
nr_3d.o \
nr_st1.o \
NullProblem.o \
OneZoneFreefallTestInitialize.o \
OutputAsParticleData.o \
OutputCoolingTimeOnly.o \
OutputFromEvolveLevel.o\
OutputLevelInformation.o \
OutputPotentialFieldOnly.o \
```

add ~ line 663

Make.config.objects

Add to Enzo

(3) add to Make.config.objects

```
Grid_InterpolateStarParticlesToGrid.o \
Grid_KHInitializeGrid.o \
Grid_MagneticFieldResetter.o \
Grid_MirrorStarParticles.o \
Grid_MoveAllParticles.o \
Grid_MoveAllStars.o \
Grid_MoveParticlesFOF.o \
Grid_MoveSubgridParticlesFast.o \
Grid_MoveSubgridParticles.o \
Grid_MoveSubgridStars.o \
Grid_MultiSpeciesHandler.o \
Grid_NestedCosmologySimulationInitializeGrid.o \
Grid_NewRotatingCylinderInitializeGrid.o \
Grid_NohInitializeGrid.o \
Grid_OneZoneFreefallTestInitializeGrid.o \
Grid_OutputAsParticleData.o \
Grid_OutputStarParticleInformation.o \
Grid_ParticleSplitter.o \
Grid_PoissonSolver.o \
Grid_PoissonSolverCGA.o \
Grid_PoissonSolverTestInitializeGrid.o \
Grid_PrepareBoundaryFluxes.o \
Grid_PrepareFFT.o \
Grid_PrepareGreensFunction.o \
Grid_PrepareGridDerivedQuantities.o \
Grid_PrepareGrid.o \
Grid_PreparePeriodicGreensFunction.o \
Grid_PreparePotentialField.o \
Grid_PrepareRandomForcingNormalization.o \
Grid_PressurelessCollapseInitialize.o \
```

add ~ line 484

Make.config.objects

Add to Enzo

Let's run!

```
> make


> cd
> mkdir RotCylinder
> cd RotCylinder
> cp ~/enzo-stable/src/enzo/NewRotatingCylinder.enzo .

> ~/enzo-stable/src/enzo/enzo.exe -d NewRotatingCylinder.enzo
```

Summary

Make the 2 new files (or copy from another test problem):

MyProblemInitialize.C

Grid_MyProblemInitializeGrid.C

Add prototype and call for non-Grid initialization routine in InitializeNew (make sure it has a unique ProblemType #).

/src/enzo/InitializeNew.C

Add definition of Grid initialization routine to Grid.h

/src/enzo/Grid.h

Add the 2 new files to Make.config.objects (so they get compiled)

/src/enzo/Make.config.objects

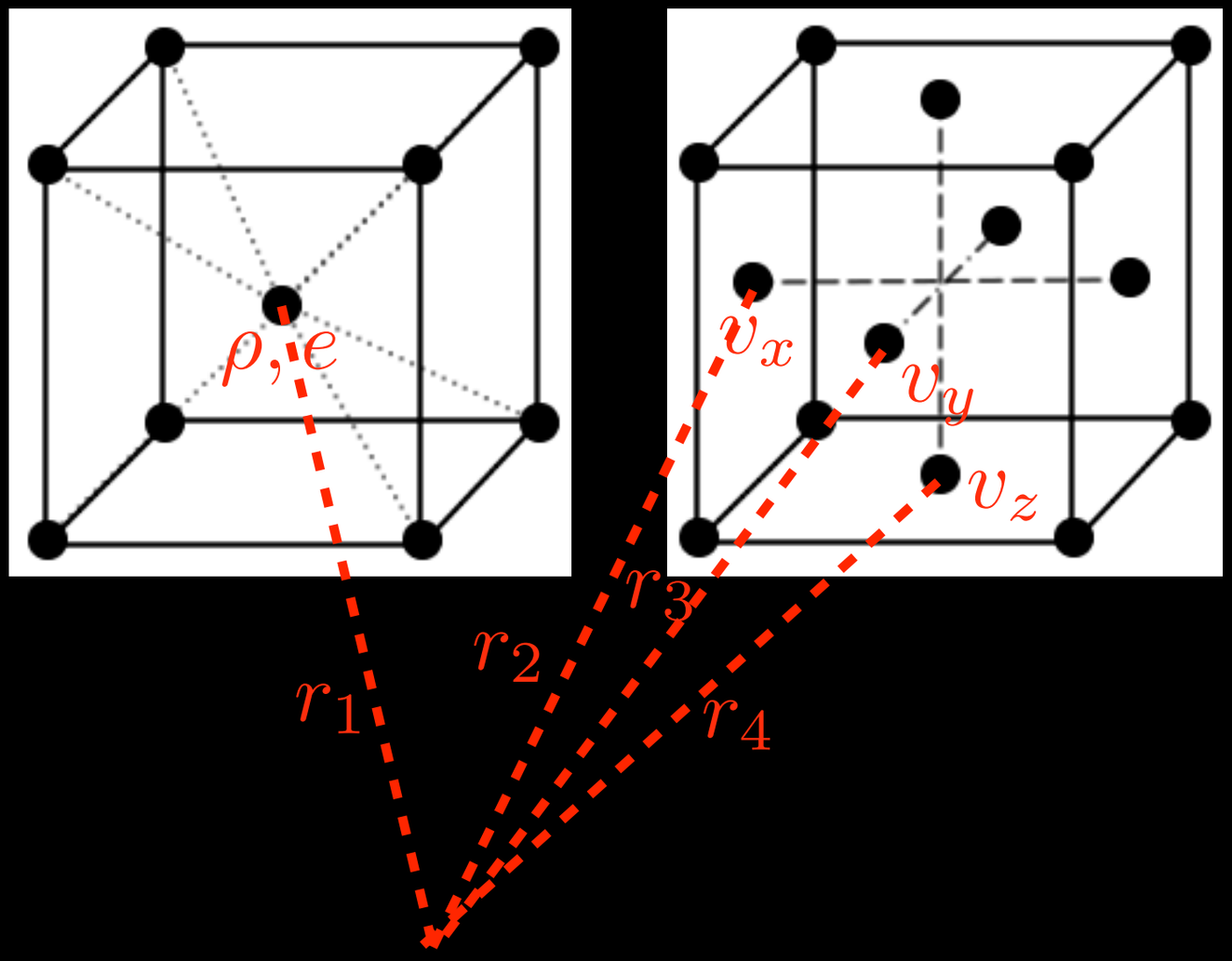
Let’s try making a change!

We’ll make a rotating sphere instead

Points to watch

Using Zeus (HydroMethod = 2)

Zeus uses a face-centered velocity




```
/* Loop over dims if using Zeus (since vel's face-centered). */

for (dim = 0; dim < 1+(HydroMethod == Zeus_Hydro ? GridRank : 0);
     dim++) {

  /* Compute position. */

  xpos = x-DiskPosition[0] -
    (dim == 1 ? 0.5*CellWidth[0][0] : 0.0);
  ypos = y-DiskPosition[1] -
    (dim == 2 ? 0.5*CellWidth[1][0] : 0.0);
  zpos = z-DiskPosition[2] -
    (dim == 3 ? 0.5*CellWidth[2][0] : 0.0);



/* Compute velocty: L x r_perp. */

if (dim == 0 || dim == 1)
  Velocity[0] = DiskVelocityMag*(AngularMomentum[1]*xhat[2] -
                                 AngularMomentum[2]*xhat[1]);
if (dim == 0 || dim == 2)
  Velocity[1] = DiskVelocityMag*(AngularMomentum[2]*xhat[0] -
                                 AngularMomentum[0]*xhat[2]);
if (dim == 0 || dim == 3)
  Velocity[2] = DiskVelocityMag*(AngularMomentum[0]*xhat[1] -
                                 AngularMomentum[1]*xhat[0]);
```

It also uses internal energy, not total energy

Points to watch

Energy

BaryonField[TENum] is energy/mass

Particle Mass

ParticleMass[i] is particle mass / cell volume

GravitationalConstant

GravitationalConstant = 4 pi G

Must be in code units if SelfGravity = 1