```
/*---------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     |
    \\  /    A nd           | www.openfoam.com
     \\/     M anipulation  |
-------------------------------------------------------------------------------
    Copyright (C) 2011-2016 OpenFOAM Foundation
    Copyright (C) 2017 OpenCFD Ltd.
-------------------------------------------------------------------------------
License
    This file is part of OpenFOAM.

    OpenFOAM is free software: you can redistribute it and/or modify it
    under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
    ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
    FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
    for more details.

    You should have received a copy of the GNU General Public License
    along with OpenFOAM.  If not, see <http://www.gnu.org/licenses/>.

Application
    chtMultiPhaseInterFoam

Group
    grpHeatTransferSolvers


\*---------------------------------------------------------------------------*/

#include "fvCFD.H"
#include "subCycle.H"
#include "multiphaseSystem.H"
#include "turbulentFluidThermoModel.H"
#include "pimpleControl.H"
#include "CombustionModel.H"
#include "fixedGradientFvPatchFields.H"
#include "regionProperties.H"
#include "alphaCourantNo.H"
#include "ddtAlphaNo.H"
#include "compressibleCourantNo.H"
#include "solidRegionDiffNo.H"
#include "solidThermo.H"
#include "radiationModel.H"
#include "fvOptions.H"
#include "coordinateSystem.H"
#include "loopControl.H"
#include "pressureControl.H"
#include "CorrectPhi.H"
```

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

int main(int argc, char *argv[])
{
    argList::addNote
    (
        "Transient solver for buoyant, turbulent fluid flow and solid heat"
        " conduction with conjugate heat transfer"
        " between solid and fluid regions."
    );

    #define NO_CONTROL
    #define CREATE_MESH createMeshesPostProcess.H
    #include "postProcess.H"
    #include "setRootCaseLists.H"
    #include "createTime.H"
    #include "createMeshes.H"
    #include "createFields.H"
    #include "createFieldRefs.H"
    #include "initContinuityErrs.H"
    #include "createTimeControls.H"
    #include "readFluidTimeControls.H"
    #include "readSolidTimeControls.H"
    #include "alphaCourantMultiRegionNo.H"
    #include "ddtAlphaMultiRegionNo.H"
    #include "compressibleMultiRegionCourantNo.H"
    #include "solidRegionDiffusionNo.H"
    #include "setInitialMultiRegionDeltaT.H"
    #include "validateTurbulenceModel.H"
    // #include "incompInitCorrectPhi.H"

    Info<< "\nStarting time loop\n" << endl;

    while (runTime.run())
    {
        #include "readTimeControls.H"
        #include "readFluidTimeControls.H"
        #include "readSolidTimeControls.H"
        #include "readPIMPLEControls.H"

        #include "alphaCourantMultiRegionNo.H"
        #include "ddtAlphaMultiRegionNo.H"
        #include "compressibleMultiRegionCourantNo.H"
        #include "solidRegionDiffusionNo.H"
        #include "setMultiRegionDeltaT.H"


        ++runTime;

        Info<< "Time = " << runTime.timeName() << nl << endl;

        // --- PIMPLE loop
        for (int oCorr=0; oCorr<nOuterCorr; ++oCorr)
        {
            const bool firstIter = (oCorr == 0);
```

```
        const bool finalIter = (oCorr == nOuterCorr-1);

        forAll(fluidRegions, i)
        {
            Info<< "\nSolving for fluid region "
                << fluidRegions[i].name() << endl;
            #include "setRegionFluidFields.H"
            #include "readFluidMultiRegionPIMPLEControls.H"
            #include "solveFluid.H"
        }

        forAll(solidRegions, i)
        {
            Info<< "\nSolving for solid region "
                << solidRegions[i].name() << endl;
            #include "setRegionSolidFields.H"
            #include "readSolidMultiRegionPIMPLEControls.H"
            #include "solveSolid.H"
        }

                // Additional loops for energy solution only
        if (!oCorr && nOuterCorr > 1)
        {
            loopControl looping(runTime, pimpleCHT, "energyCoupling");

            while (looping.loop())
            {
                Info<< nl << looping << nl;

                forAll(fluidRegions, i)
                {
                    Info<< "\nSolving for fluid region "
                        << fluidRegions[i].name() << endl;
                    #include "setRegionFluidFields.H"
                    #include "readFluidMultiRegionPIMPLEControls.H"
                    frozenFlow = true;
                    #include "solveFluid.H"
                }

                forAll(solidRegions, i)
                {
                    Info<< "\nSolving for solid region "
                        << solidRegions[i].name() << endl;
                    #include "setRegionSolidFields.H"
                    #include "readSolidMultiRegionPIMPLEControls.H"
                    #include "solveSolid.H"
                }
            }
        }
    }

    runTime.write();

    runTime.printExecutionTime(Info);
}
```

```
    Info<< "End\n" << endl;

    return 0;
}


// ************************************************************************* //
```