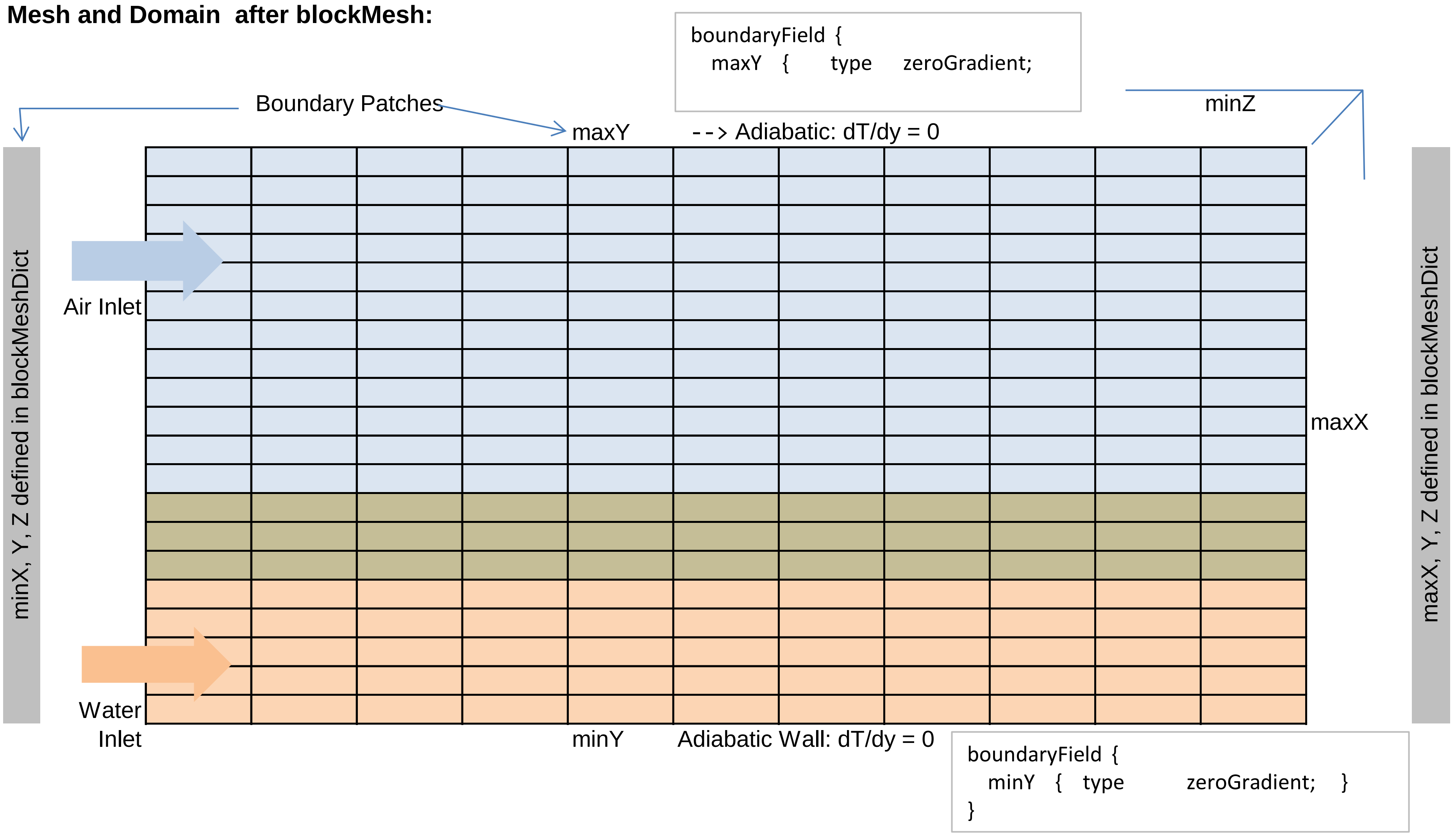
Conjugate Heat Transfer Set-up

1606+ -> v2.4.0.

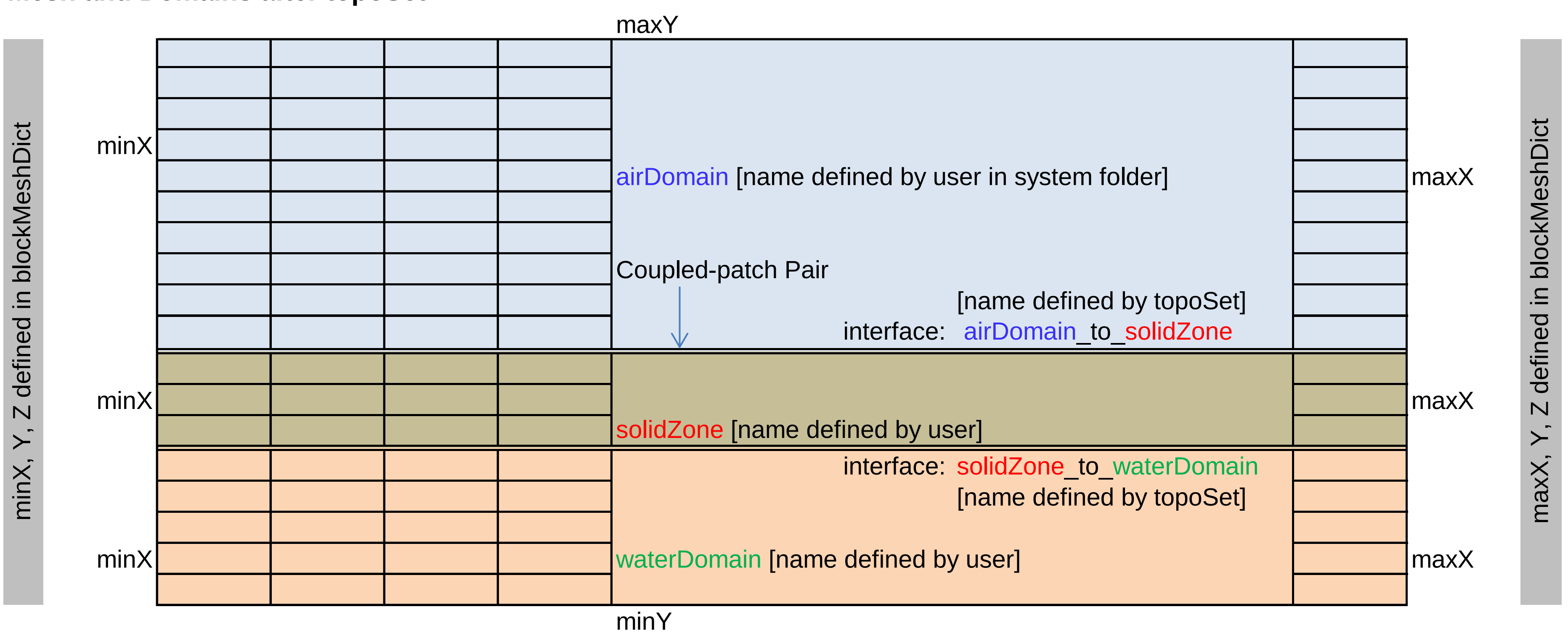
Reference: chtMultiRegionFoam\multiRegionHeater

http://openfoamwiki.net/index.php/Getting_started_with_chtMultiRegionSimpleFoam_-_planeWall2D

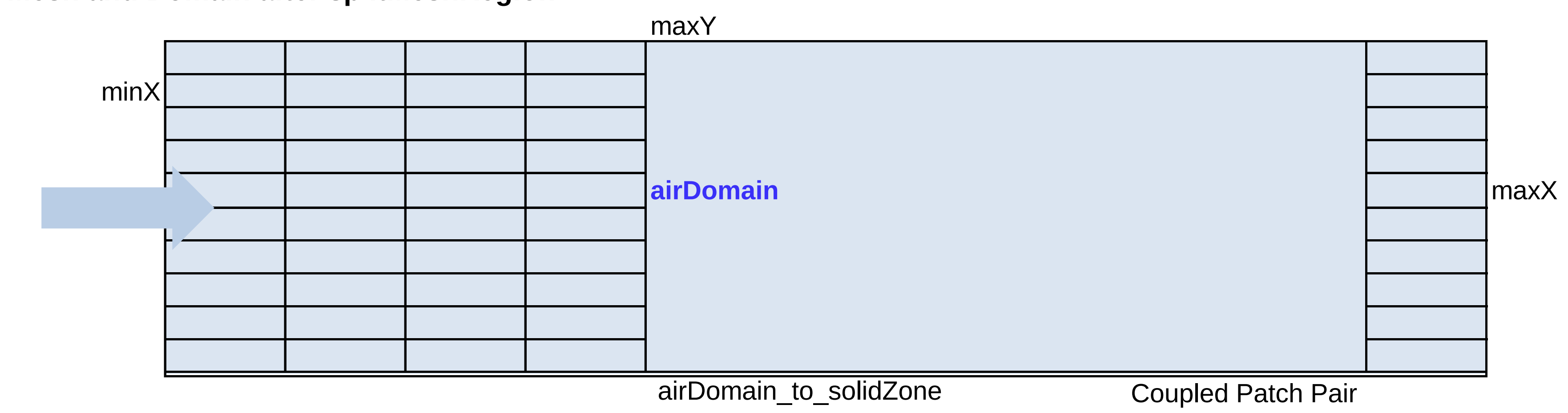
Mesh and Domain after blockMesh:



Mesh and Domains after topoSet

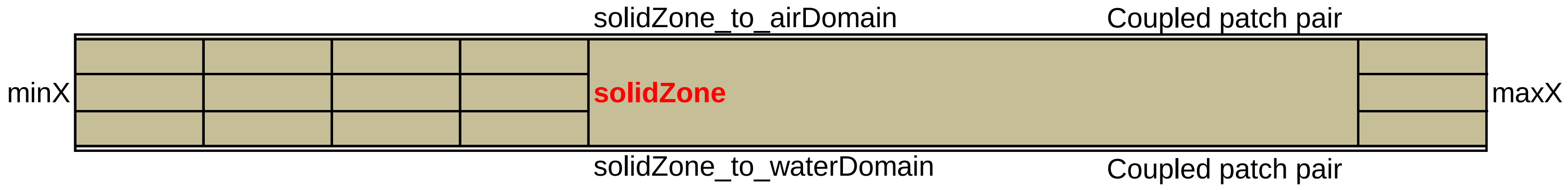


Mesh and Domain after splitMeshRegion



Defined in constant/**airDomain**/polyMesh/boundary file - -

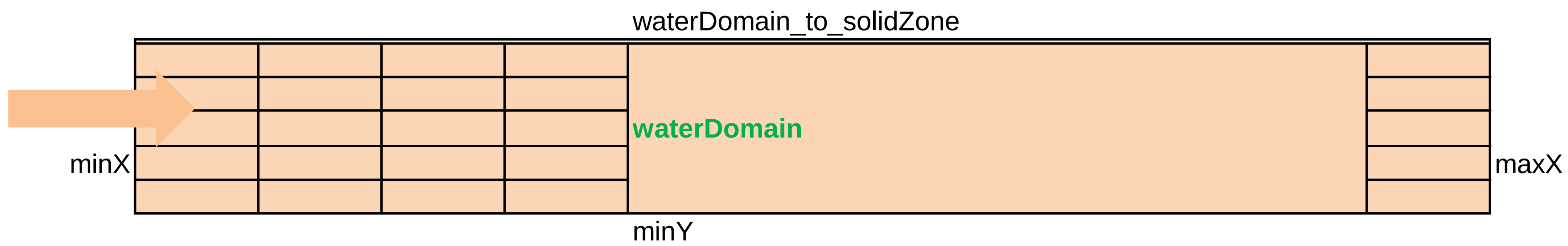
```
airDomain_to_solidZone   {
  type             mappedWall;
  inGroups         1(wall);
  nFaces           500;
  startFace        23400;
  sampleMode       nearestPatchFace;
  sampleRegion     solidZone;
  samplePatch      solidZone_to_airDomain;
}
```



Defined in constant/**solidZone**/polyMesh/boundary file - - >

```
solidZone_to_waterDomain  {
    type            mappedWall;
    inGroups        1(wall);
    nFaces          500;
    startFace       15100;
    sampleMode      nearestPatchFace;
    sampleRegion    waterDomain;
    samplePatch     waterDomain_to_solidZone;
}

solidZone_to_airDomain  {
    type            mappedWall;
    inGroups        1(wall);
    nFaces          500;
    startFace       15600;
    sampleMode      nearestPatchFace;
    sampleRegion    airDomain;
    samplePatch     airDomain_to_solidZone;
}
```



Defined in constant/**waterDomain**/polyMesh/boundary file - - >

This file contains boundaries:

minX, minY, minZ, maxX, maxZ, waterDomain_to_solidZone
type same as defined in blockMeshDict.

```
waterDomain_to_solidZone   {
    type            mappedWall;
    inGroups        1(wall);
    nFaces          500;
    startFace       23400;
    sampleMode      nearestPatchFace;
    sampleRegion    solidZone;
    samplePatch     solidZone_to_waterDomain;
}
```

Specifying fluid and solid regions

constant/regionProperties

```
regions  (
  fluid     (waterDomain airDomain)
  solid     (solidZone)
);
```

Setting Boundary Conditions

splitMesh creates 0/region_names/ directories for all regions and copies all the field files existing in the 0 directory into the 0/region_names/ directories. Hence, the solid region will contain files related to k, p, epsilon, U, T ... These files not applicable to solid domains need to be deleted manually. Though, files U, p, p_rgh, epsilon, k are required to be present in solid domain also.

Similarly, files applicable to solid field need to be deleted from folders applicable to fluid fields.

Setting Initial Conditions

Option-1

changeDictionary

changeDictionary uses changeDictionaryDict files located in system/region_names/ folder to create initial, boundary and coupling boundary conditions for all fields existing in 0/region_names/ directory for all regions. changeDictionaryDict has to be edited to suit the needs of each simulation case. The content of changeDictionaryDict is like any other field file such as U, T, p... but not associated to any specific field. In other words it may contain field information for U, p, T, epsilon ... in same file.

The power of changeDictionary lies in the usage of wild card characters '.' and '*' as explained below.
"patch1_to_.*" can be used to access all patches that start with patch1_to_, here '.' means any single character, '*' is called a wild-card character which can be used to access any set of continuous characters (not containing white spaces).

Thus, an entry like shown below in changeDictionaryDict is equivalent to:

```
".*"       {
  type       fixedValue;
  value      uniform (0 0 0);     }
```

≡

```
patch1     {
  type       fixedValue;
  value      uniform (0 0 0);     }
patch1     {
  type       fixedValue;
  value      uniform (0 0 0);     }
...
patch1     {
  type       fixedValue;
  value      uniform (0 0 0);     }
```

Any setting for specific patch can be set as:

```
patchX     {
  type       fixedValue;
  value      uniform (0 0 0);     }
```

The advantage of changeDictionaryDict is the fact that in case you need to update the mesh by using utilities -blockMesh, topoSet and splitMeshRegion - the boundary and initial conditions has to be set again. Manual method would be a bit tedious and prone to error & omissions. Running changeDict every time mesh domain is changes is faster and easier.

Option-2

Alternatively, the field files U, p, T ... Can be modified manually one by one without running changeDictionary utility.

In case mesh and cell regions are changed, all the files in 0/ folder including sub-folders need to replaced with files used initially. This excludes files cellToRegion which was created by splitMeshRegions utility on the new mesh and cell zones.

Solver Setting

The fvSchemes and fvSolution file inside the system folders are dummy ones. The actual setting files should inside the respective folders for solid domain and fluid domains.
Run: chtMultiRegionFoam > logFile & - rung the solver in background (&) and send output to log file named 'logFile'

Serial Run: No extra operation
Parallel Run: Use utility 'decomposePar',edit decomposeParDict as per the number of processors available on your machine.

Solver Monitoring

Monitoring the solution progress: can be done using **foamLog** utility and tail -f logFile where 'logFile' is the name of log file specified in previous step.

Serial Run: No extra operation
Parallel Run: Use utility "reconstructPar -allRegions"

Post-processing:

CHT{solidZone}.foam	Each one of these files will render in ParaView the respective region.
CHT{airDomain}.foam	The file "CHT.foam" will show the whole original mesh.
CHT{waterDomain}.foam	