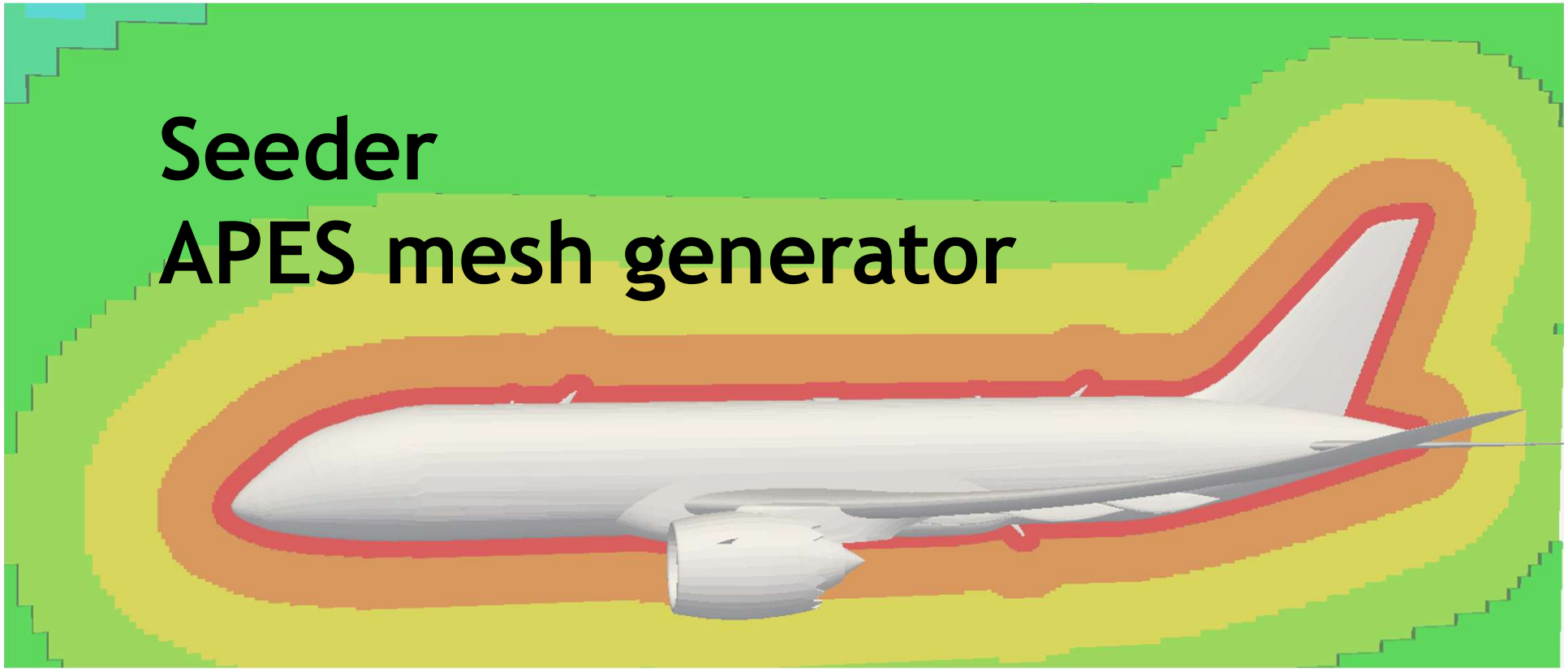
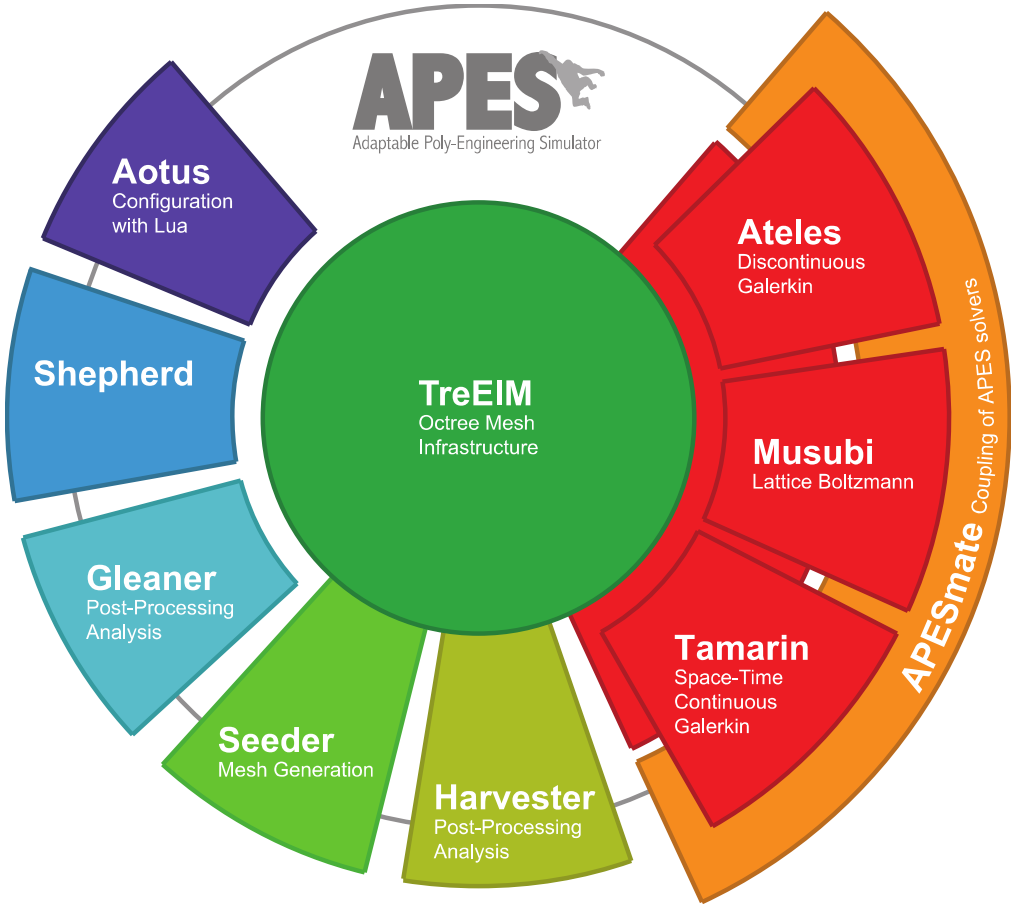
Seeder
APES mesh generator



Outline

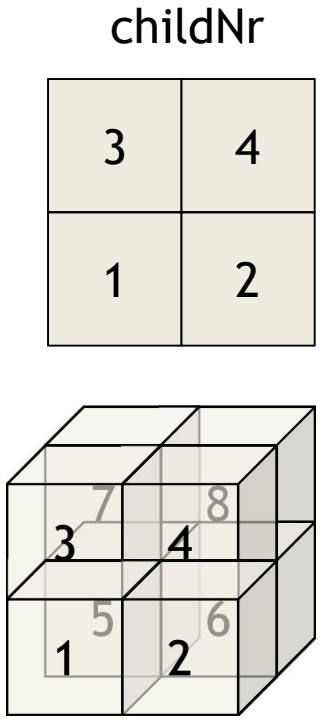
- Introduction
- Algorithm
- Build prototree
- Flooding
- Boundary identification
- qVal boundary
- How to setup configuration file?

APES framework



Introduction

- TreeID – Unique ID for every node in the tree
  - 2D:
    - ChildID = parentID * 4 + childNr
    - ParentID = int((childID - 1) / 4)
  - 3D:
    - ChildID = parentID * 8 + childNr
    - ParentID = int((childID – 1) / 8)
- Bounding cube / universe
- Space-filling curve



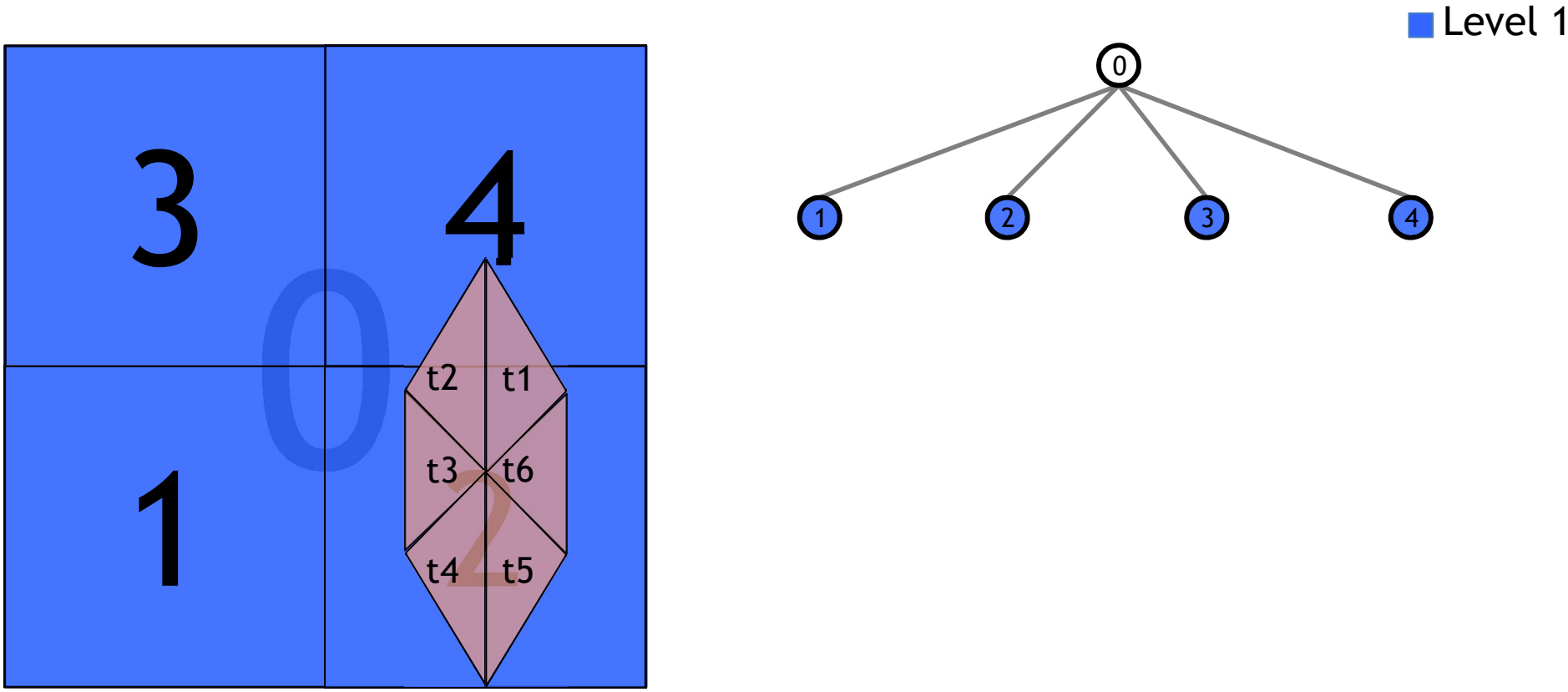
Algorithm

- Build protoTree
  - Iterative → levelwise
- Identify neighbors in 6 normal directions
- Flooding
  - Run over all nodes – Is iterative, due to levelwise sorting of the nodes in the protoTree
- Refine leaf
  - Iterative
- Proto2TreElm
  - Recursive traverse tree
- Dump the tree and the boundary info to disk

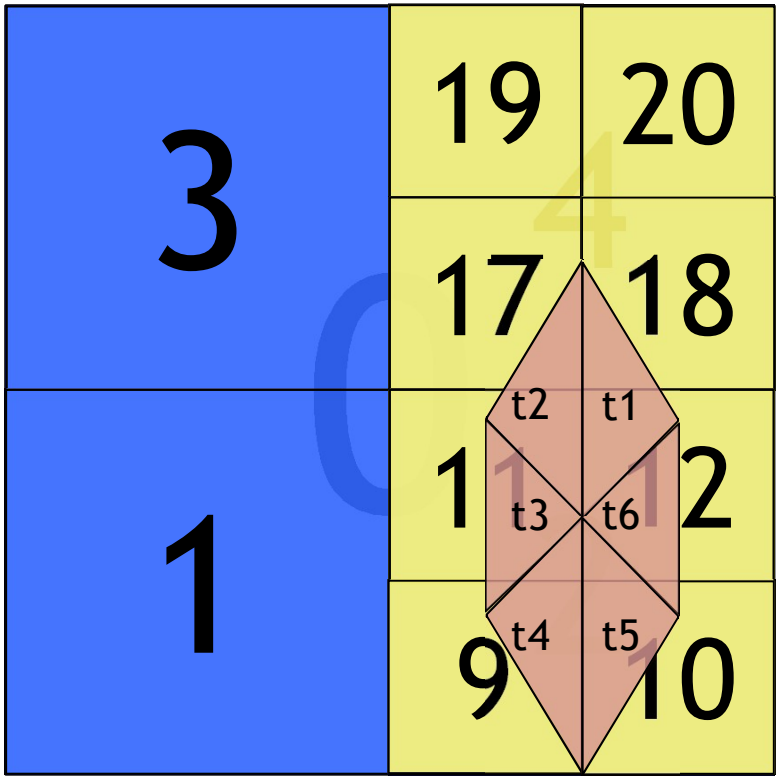
Build protoTree

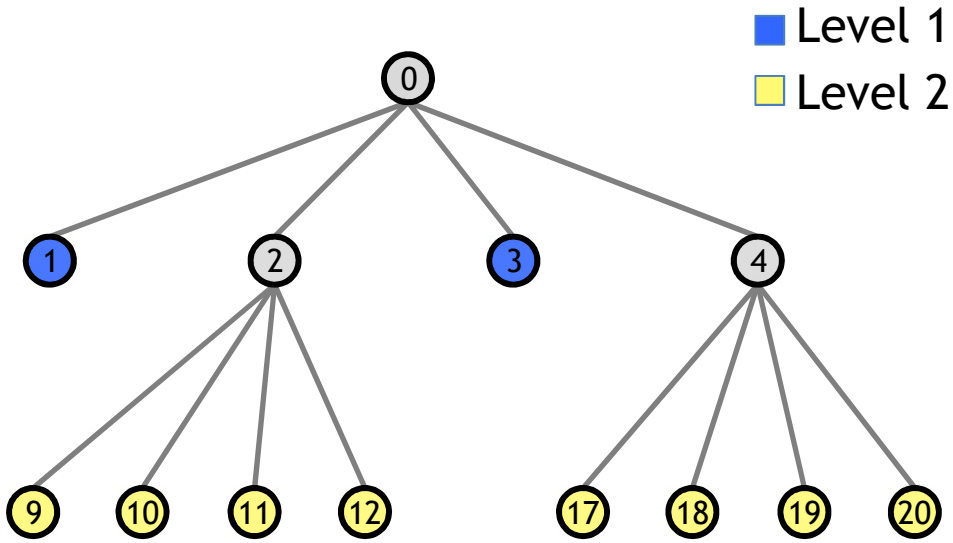
- Start with level 0 and refine nodes intersected by boundary objects, until the defined level of that boundary is reached
- If node is intersected by a seed object, set the node property bit to fluid and wet its neighbors face
- If node is intersected by a boundary object, set the node property bit to intersected boundary bit
- If node is not intersected by any object or the max. level is reached, set the node property bit to leaf

Proto tree: Level 1

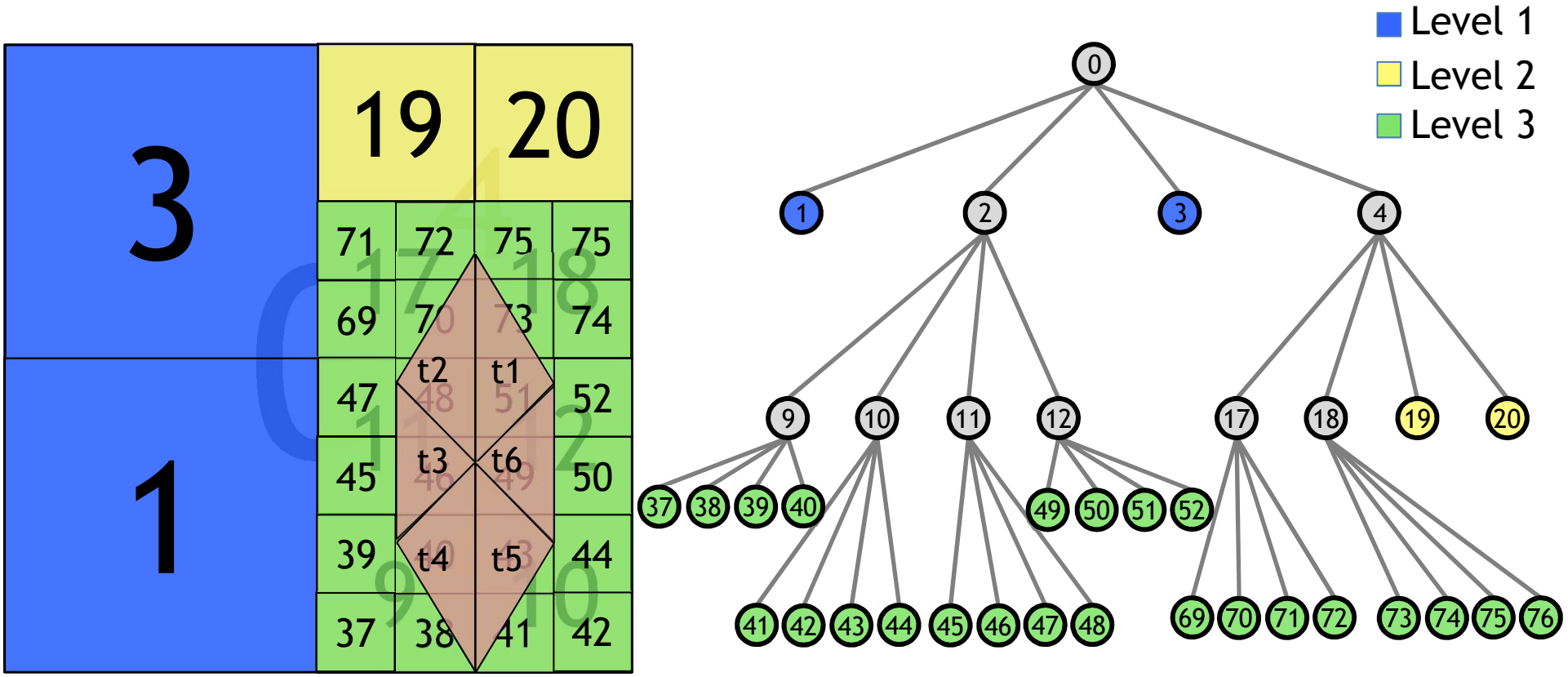


Proto tree: Level 2

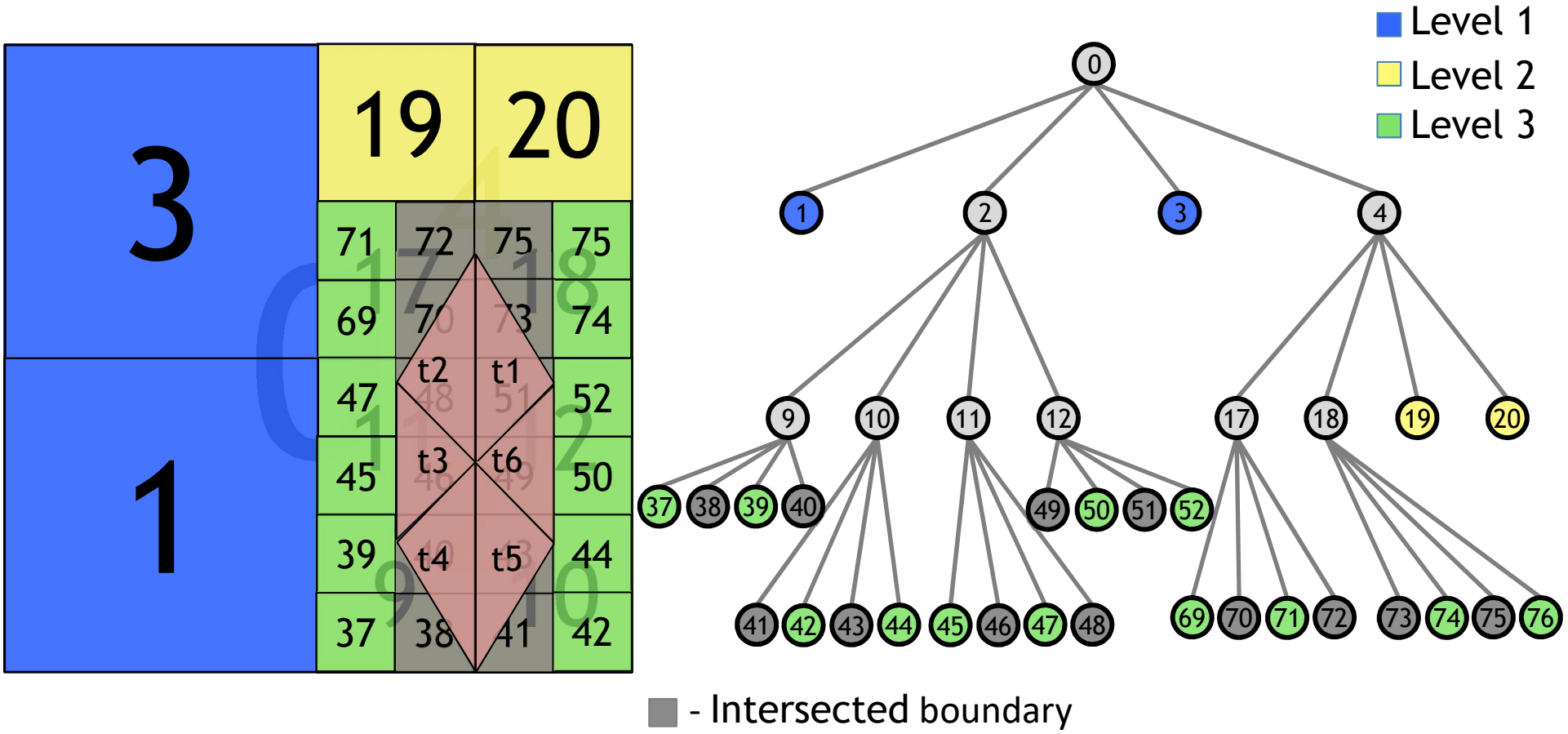




Proto tree: Level 3

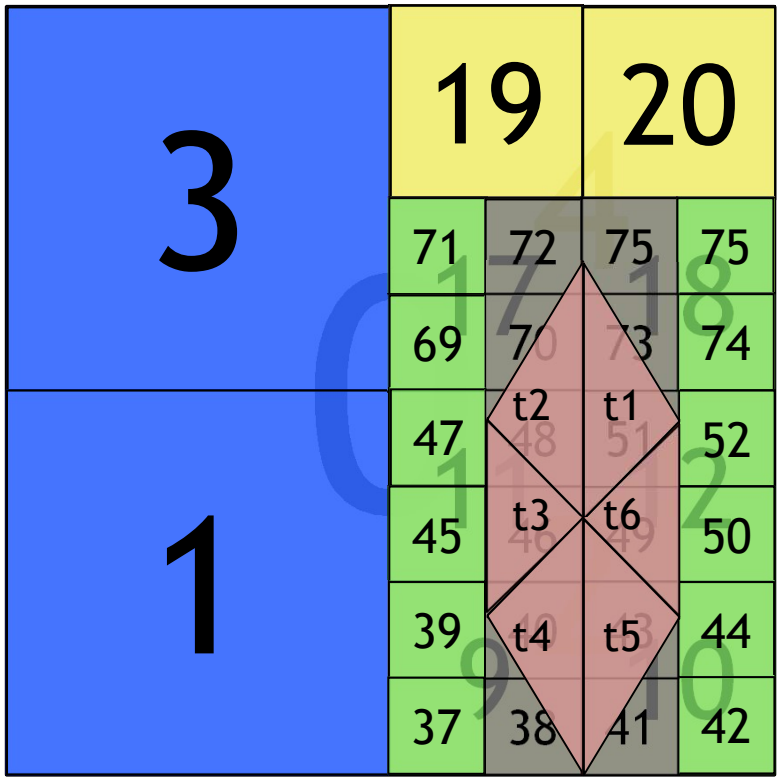


Proto tree: Level 3



Neighbor identification

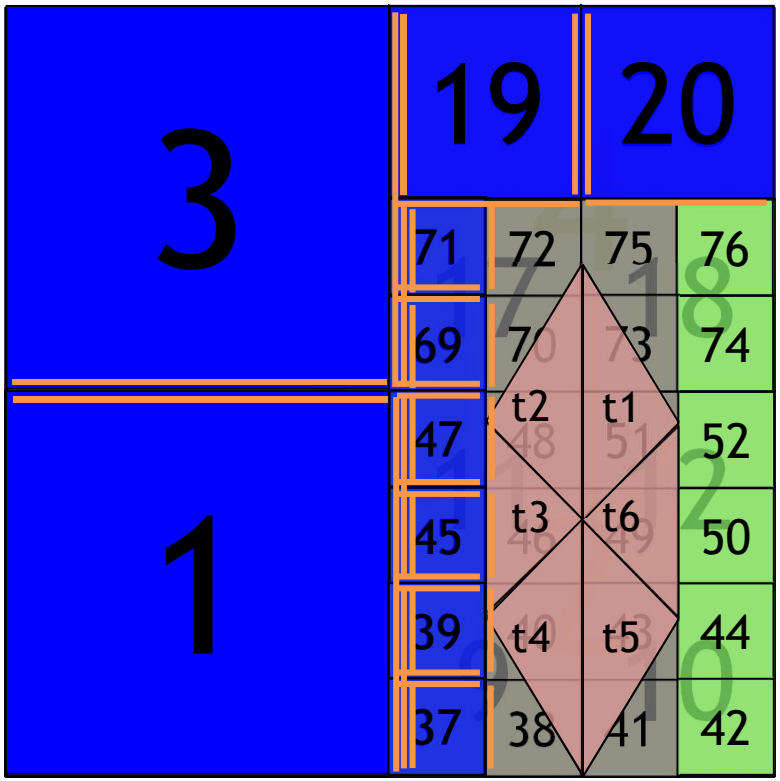
- Identify the neighbors of each node, which are connected to its faces.
- If the neighbor does not exist in the same level, then search for existing neighbor node in the coarser level.

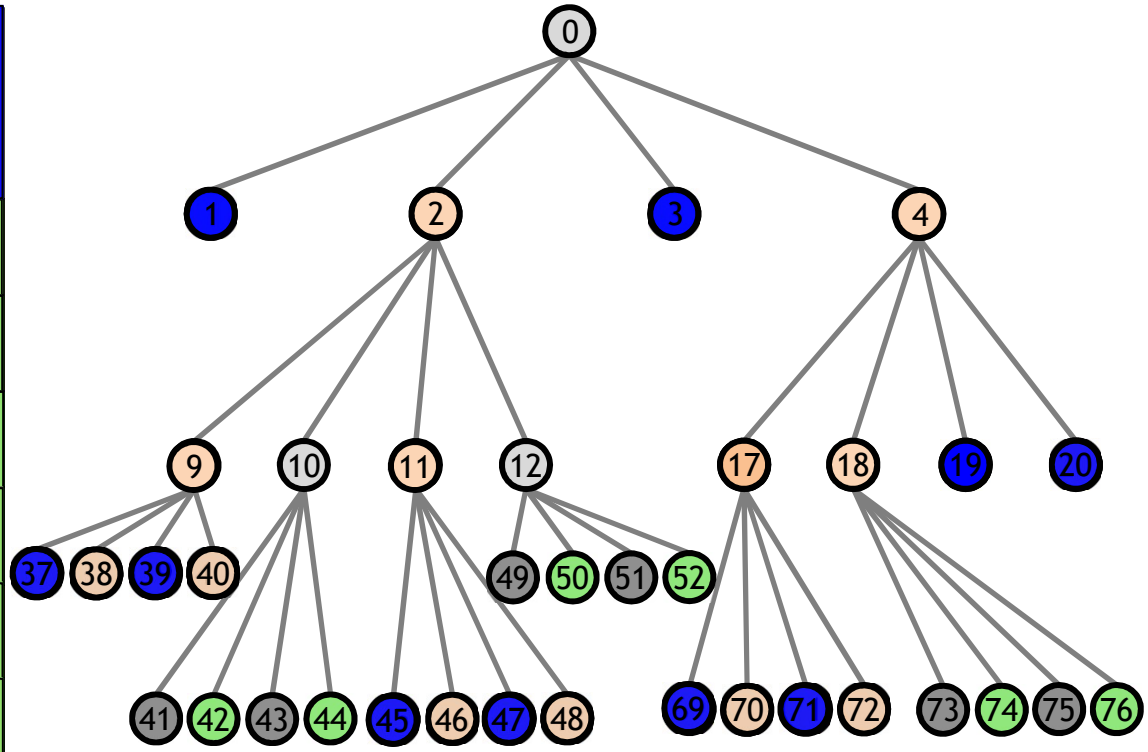


Flooding

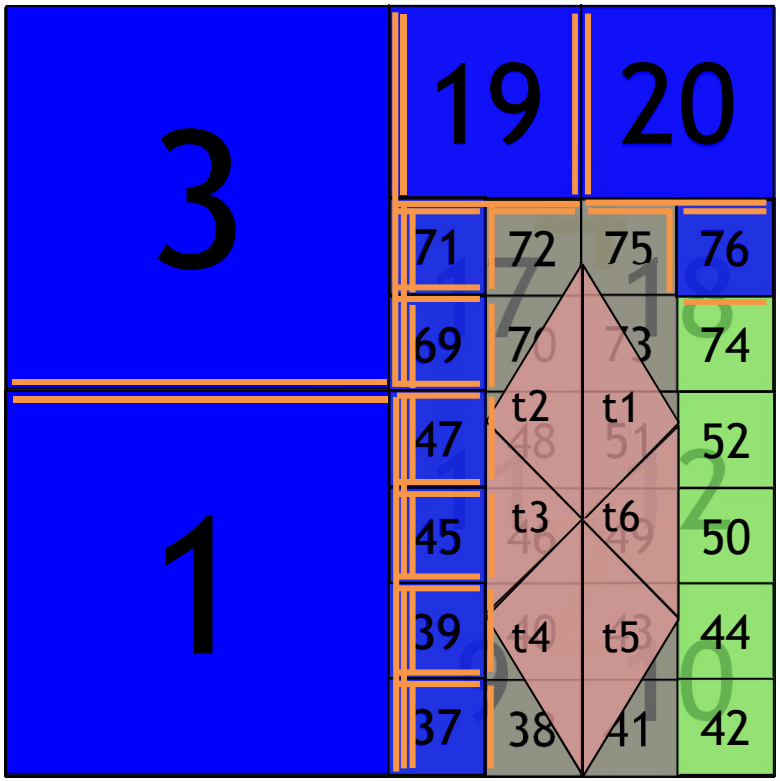
- Loop over nNodes in protoTree
- If the node is leaf and not intersected by a boundary and has a wet face, then flood this node and wet its neighbors face
- If the node is not leaf and not intersected by a boundary and has a wet face, then inherit wet face to its corresponding children faces
- Since nodes in protoTree are sorted from coarser level to finer level, inheritance of the wet face from the coarser nodes are automatically propagated down within a single wave

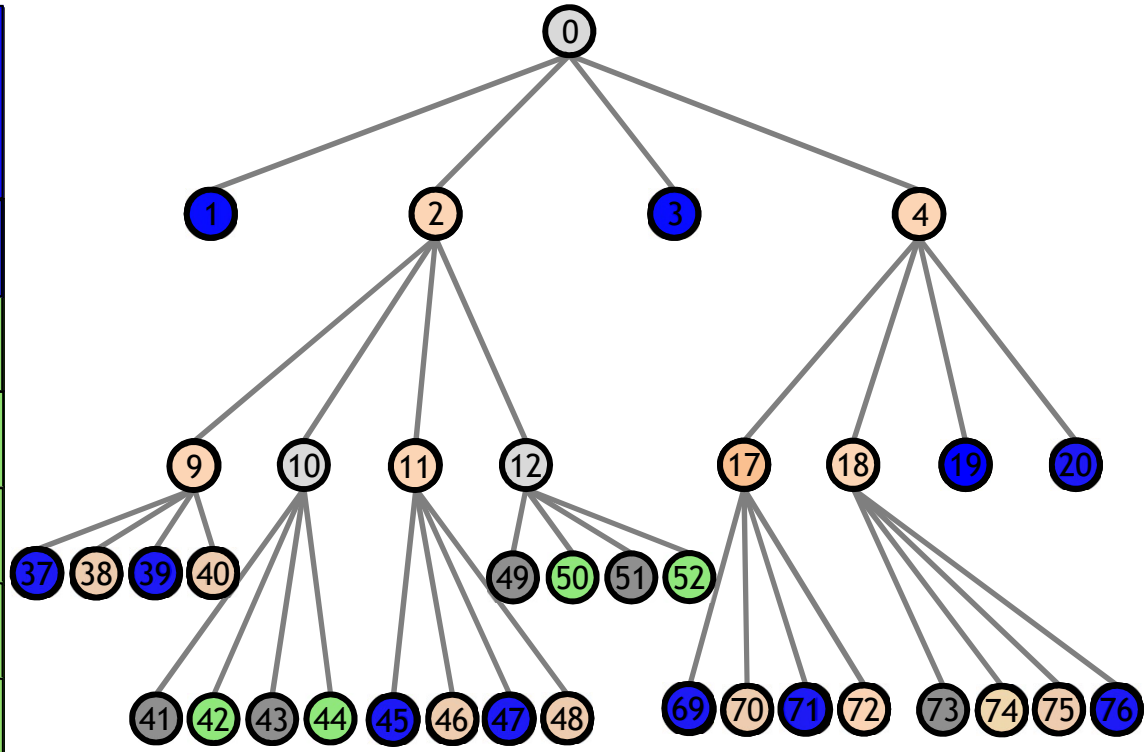
Proto tree: Flooding wave 1



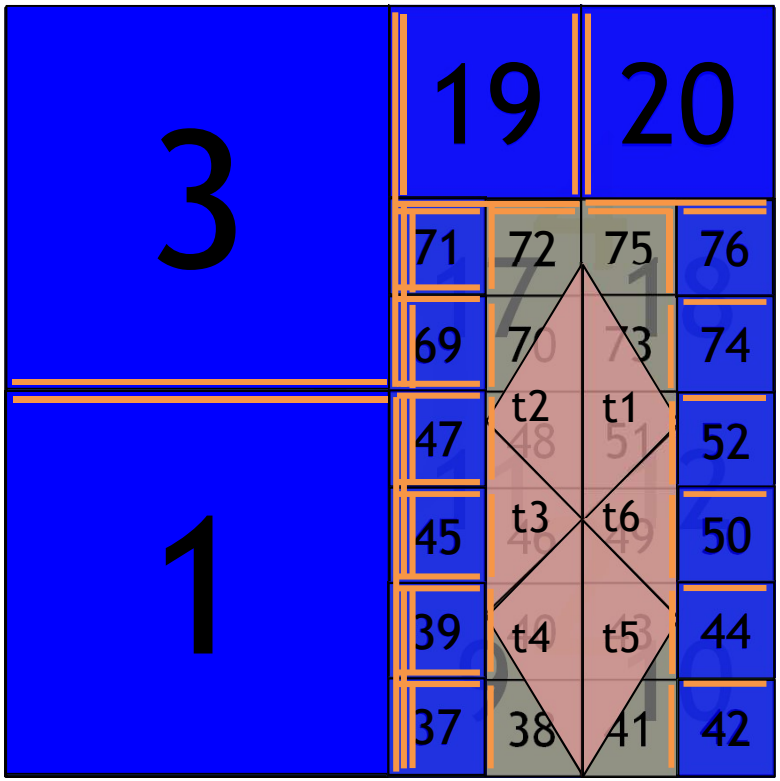


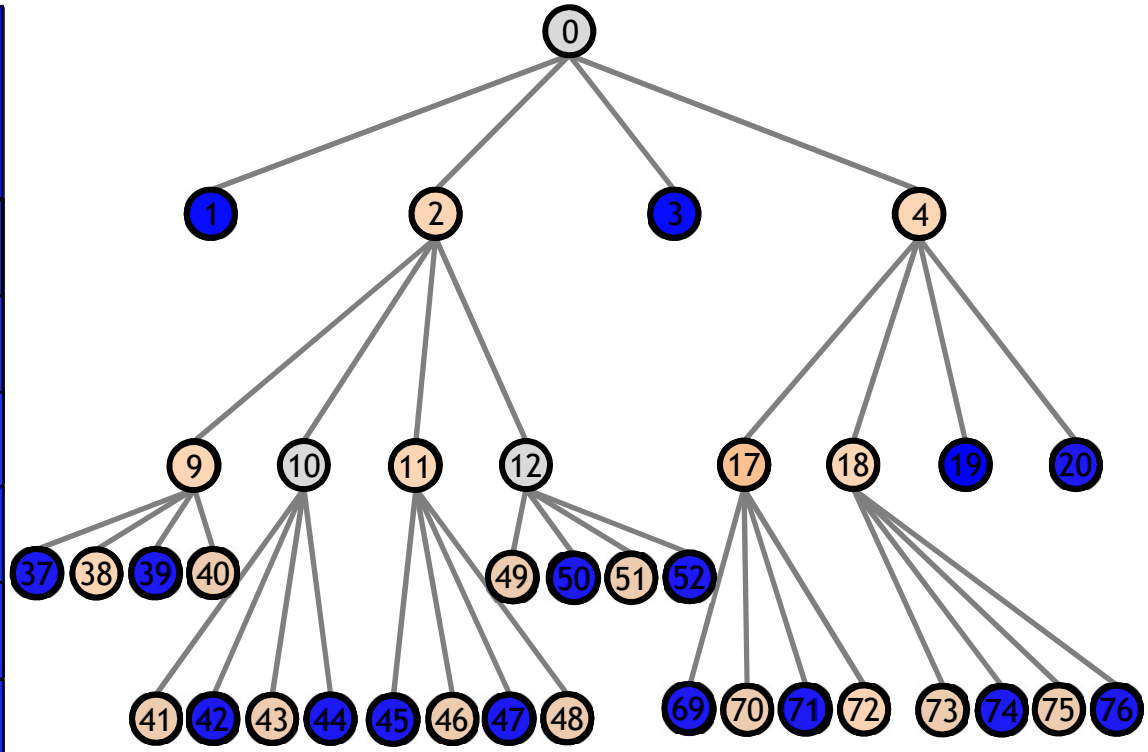
Proto tree: Flooding wave 2



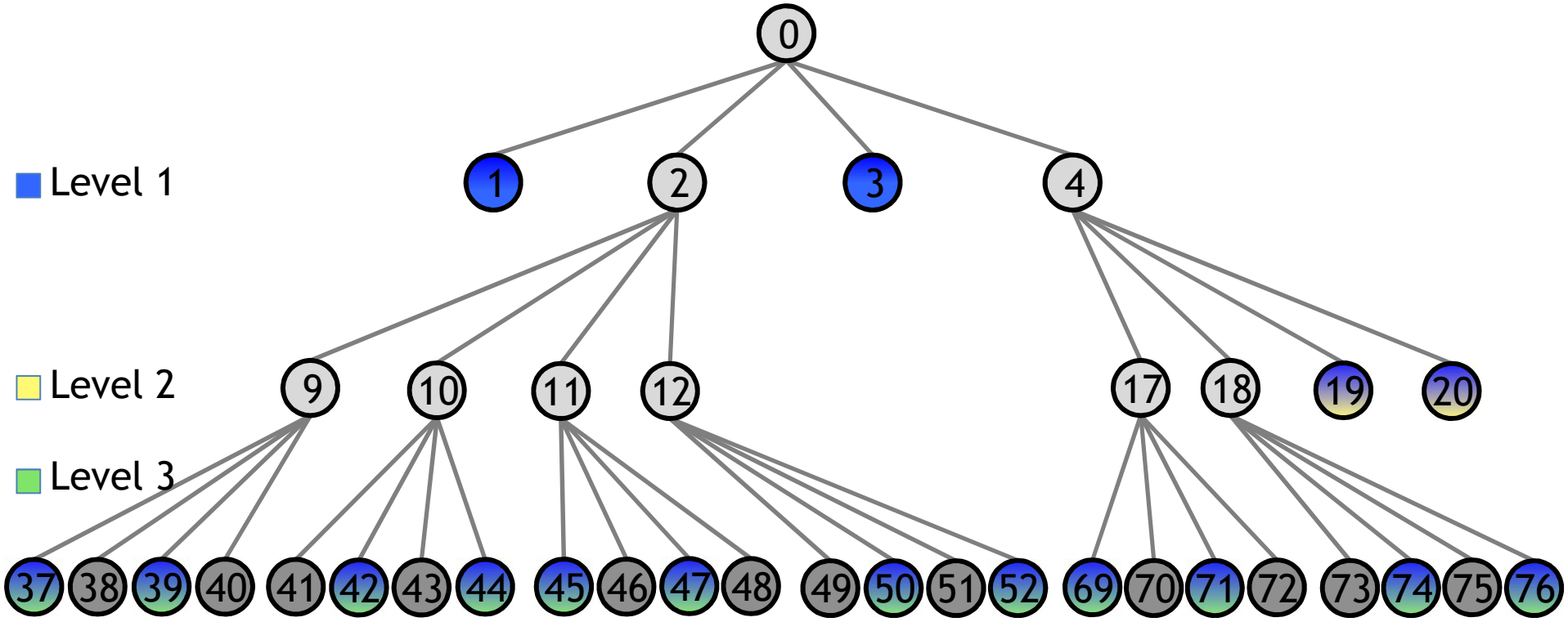


Proto tree: Flooding wave 3..7





Proto tree: After flooding



Flooded & leaf TreeIDs:

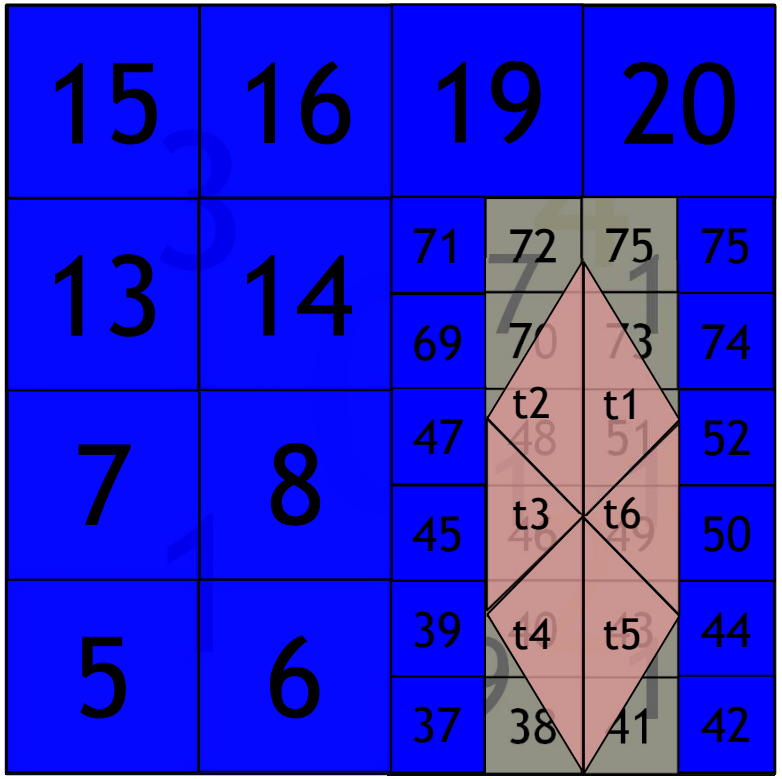
1	37	39	42	44	45	47	50	52	3	69	71	74	76	19	20

Algorithm: Iterative refine leaf

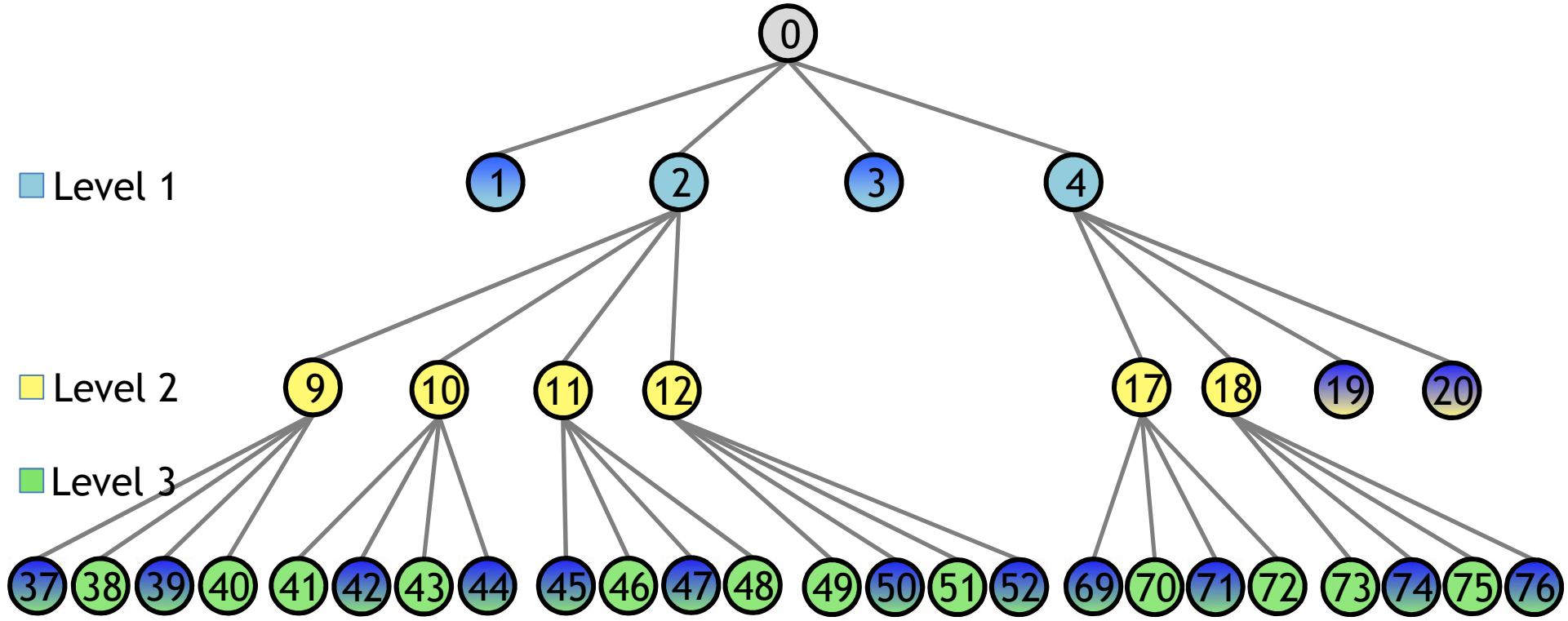
- Level loop
  - Loop over the parent nodes of the current level
    - If the parent is flooded, leaf and not intersected boundary then
      - Set maxLevel=minLevel
      - Check maxLevel of parent's intersected objects.
      - If smoothbounds is true then check in all 26 directions for intersected boundary neighbor and get max(maxLevel, bndLevel)
      - If maxLevel >= current level then create 8 children and inherit intersected objects.
  - Update lthe evelNode_last for current level. Also, Update the levelNode_first and levelNode_last for iLevel+1 up to the global max levels
  - Exit level loop, when no new nodes are created in the current level

Refine leaf

- Refine leaf nodes until the defined minlevel is reached
- E.g.: minlevel = 2

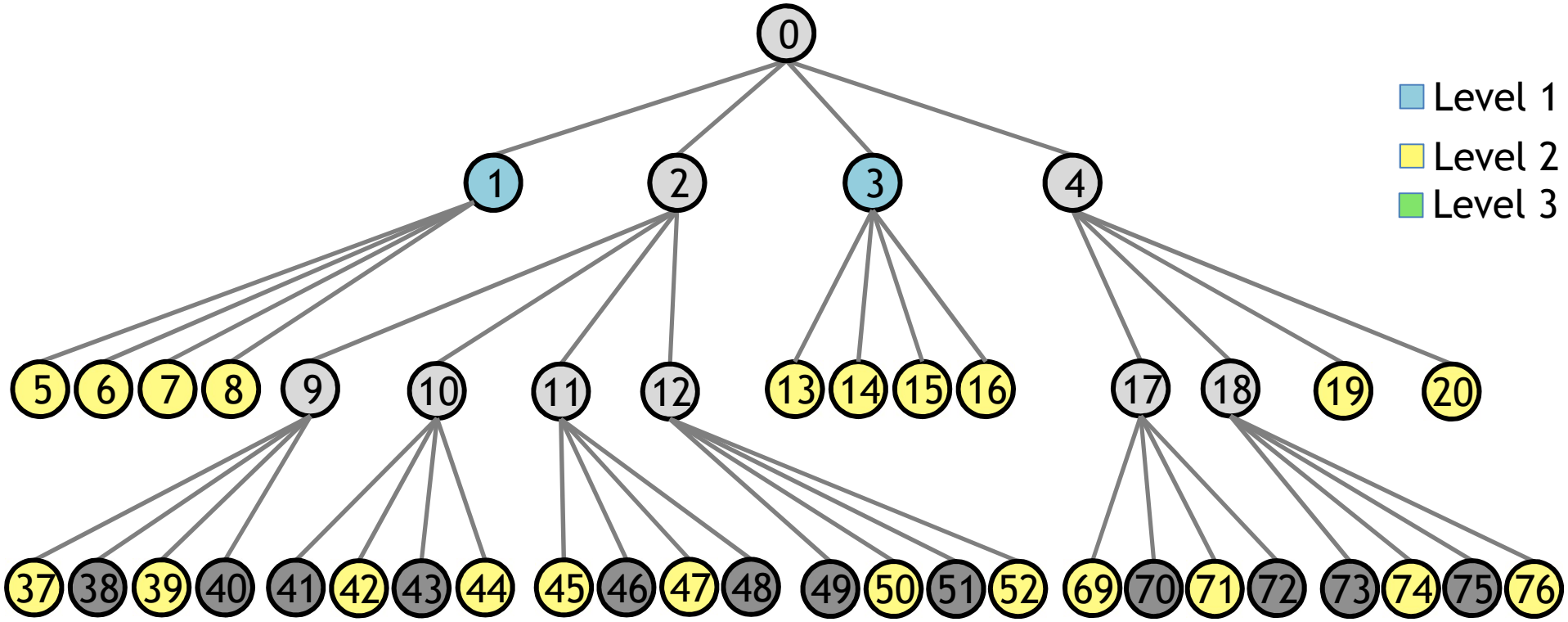


Proto tree: After flooding



TreeIDs:	0	1	2	3	4	9	10	11	12	17	18	19	20	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	69	70	71	72	73	74	75	76
Sorted IDX:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36

Iterative refine leaf

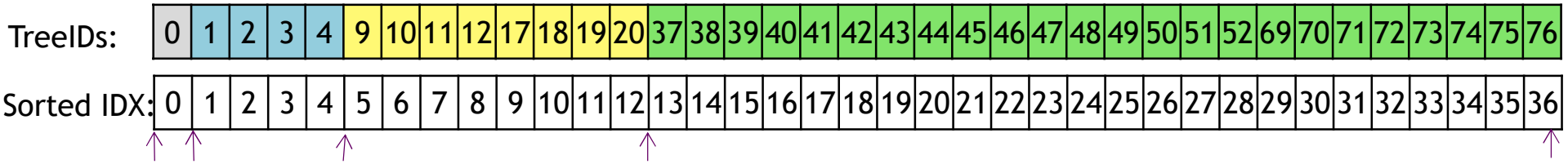


TreeIDs: 0 1 2 3 4 9 10 11 12 17 18 19 20 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 69 70 71 72 73 74 75 76 5 6 7 8 13 14 15 16

Sorted IDX: 0 1 2 3 4 37 38 39 40 5 6 7 8 41 42 43 44 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36

Level node first and last

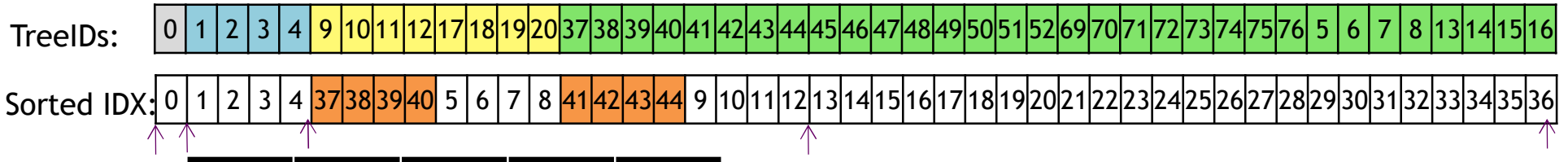
Before refine leaf



Level Node

Level	0	1	2	3
First	0	1	6	13
Last	0	5	12	36

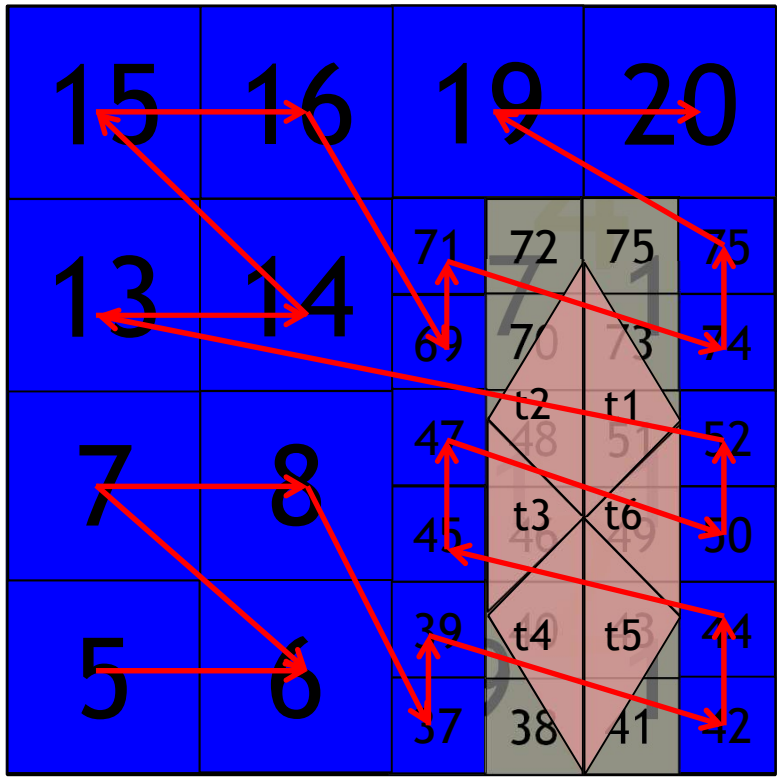
After refine leaf



Level Node

Level	0	1	2	3
First	0	1	6	21
Last	0	5	20	44

Order of TreeIDs according to the space filling curve

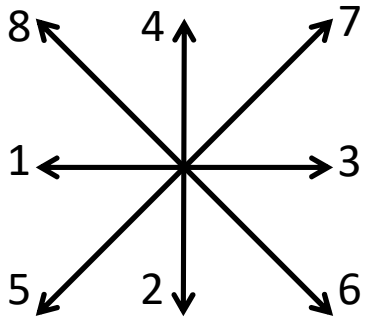


Final TreeIDs:

5	6	7	8	37	39	42	44	45	47	50	52	13	14	15	16	69	71	74	76	19	20

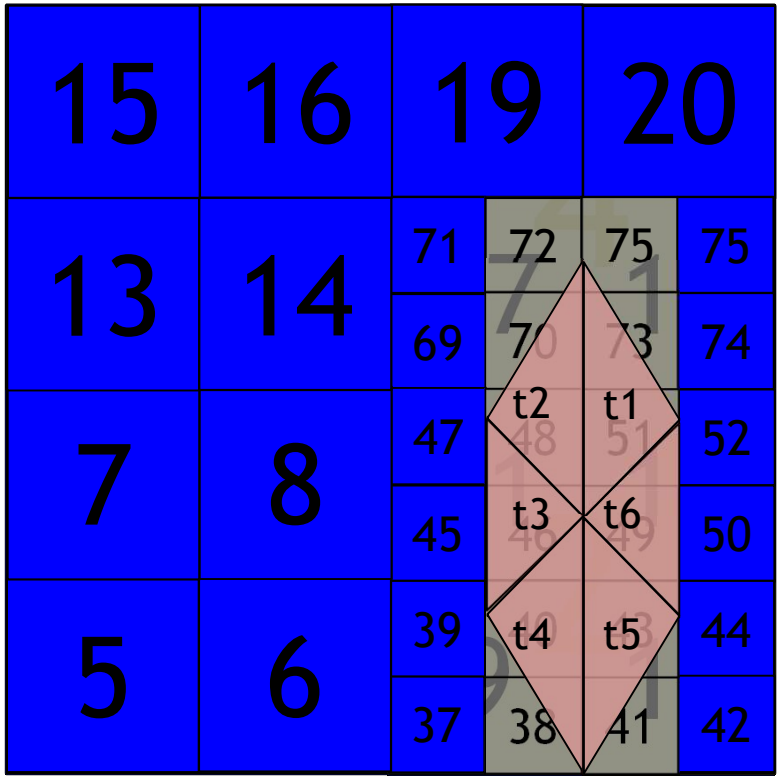
Boundary identification

- Identify the boundary Ids of each node with an intersecting boundary neighbor in all directions.



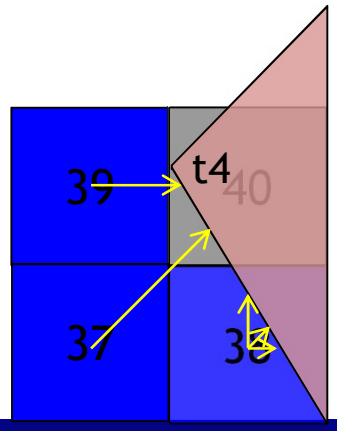
Node 39:

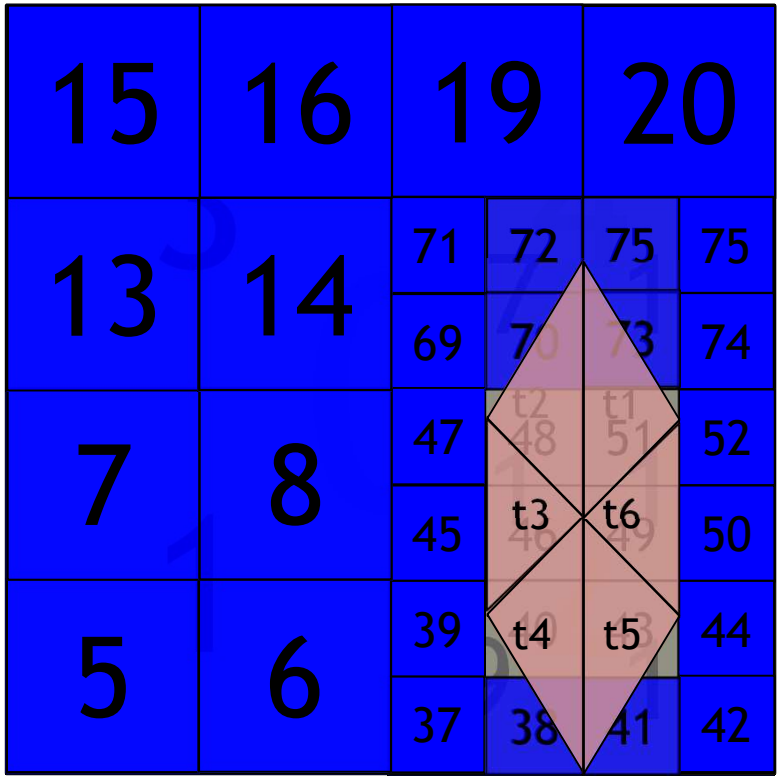
0	0	T4	0	0	T4	T3	0



qVal boundaries

- Flood intersected boundary nodes, which have no geometry intersection on the direction of flooded node from its barycenter
- E.g.: Node 38, 41, 70, 73, 72, 75



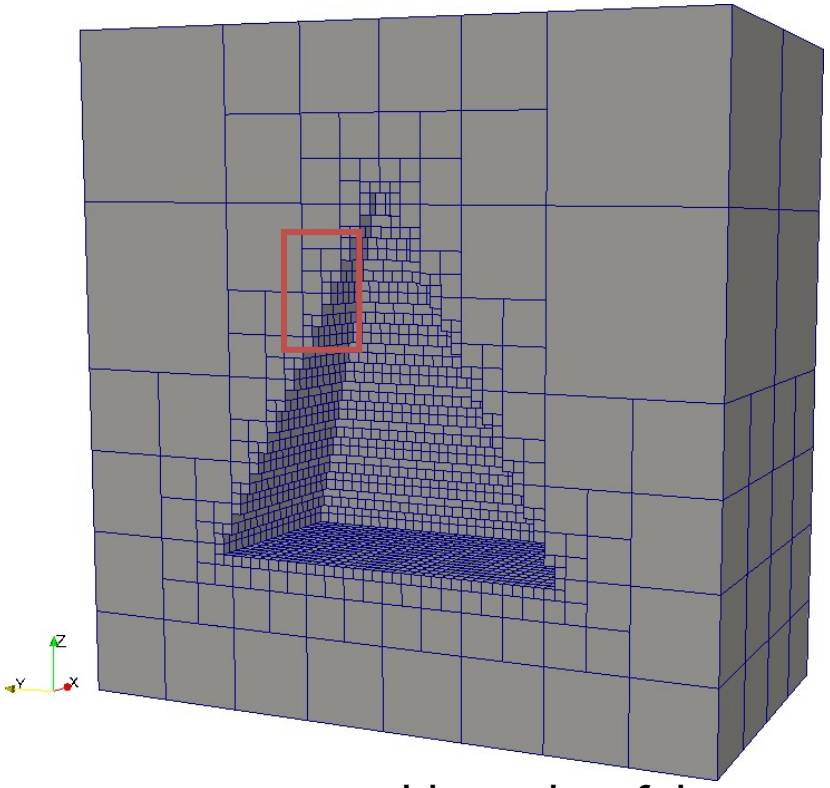


Algorithm: Smoothing

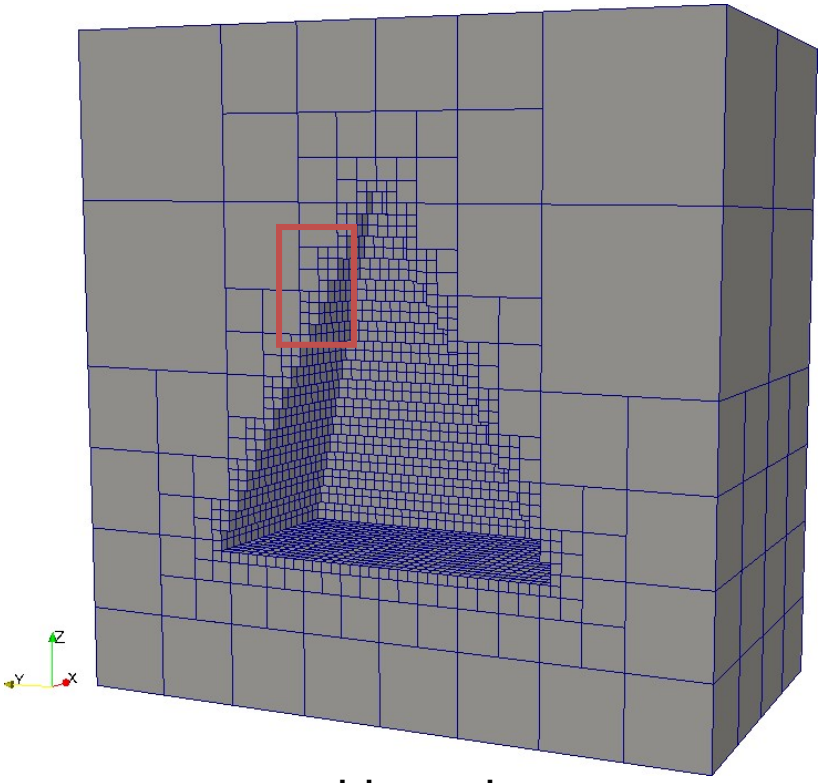
- Do iWave until no new nodes are created
  - Level loop (minLevel+1, maxLevel – iWave)
    - Loop over parent nodes of current level
      - If parent is flooded, leaf and not intersected boundary then
        - » *Loop over all 26 directions*
          - If neighbor is not a leaf
            - If any eligible child in iDir is not a leaf
              - Create 8 children
    - Update levelNode_last for current level. Also, update levelNode_first and levelNode_last for iLevel+1 up to the global max level

Smooth bound: Example

- Configurable - “smoothbounds = true/false”



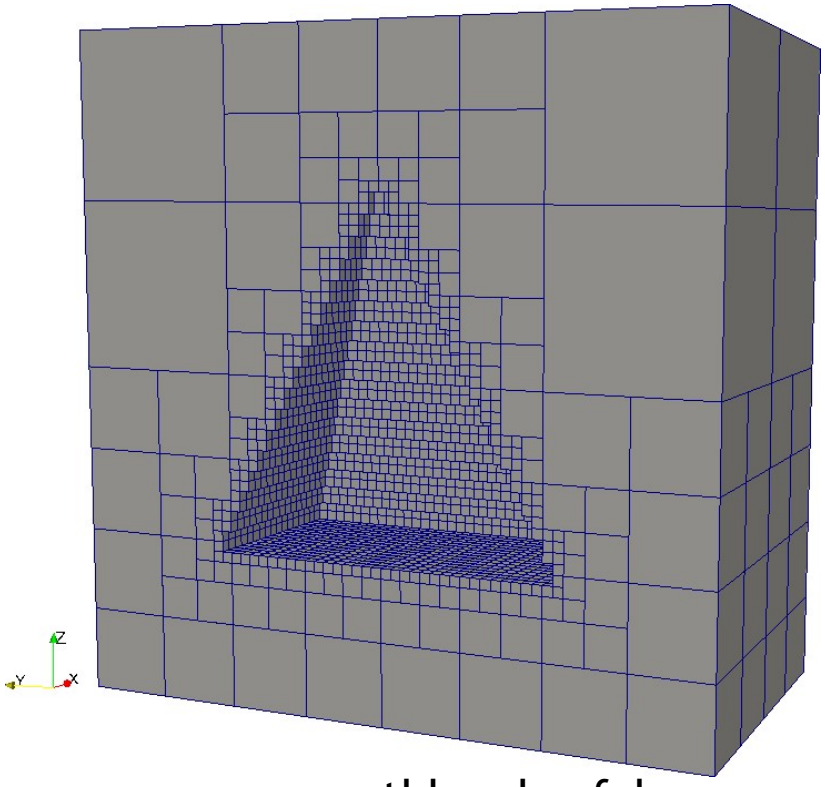
smoothbounds = false



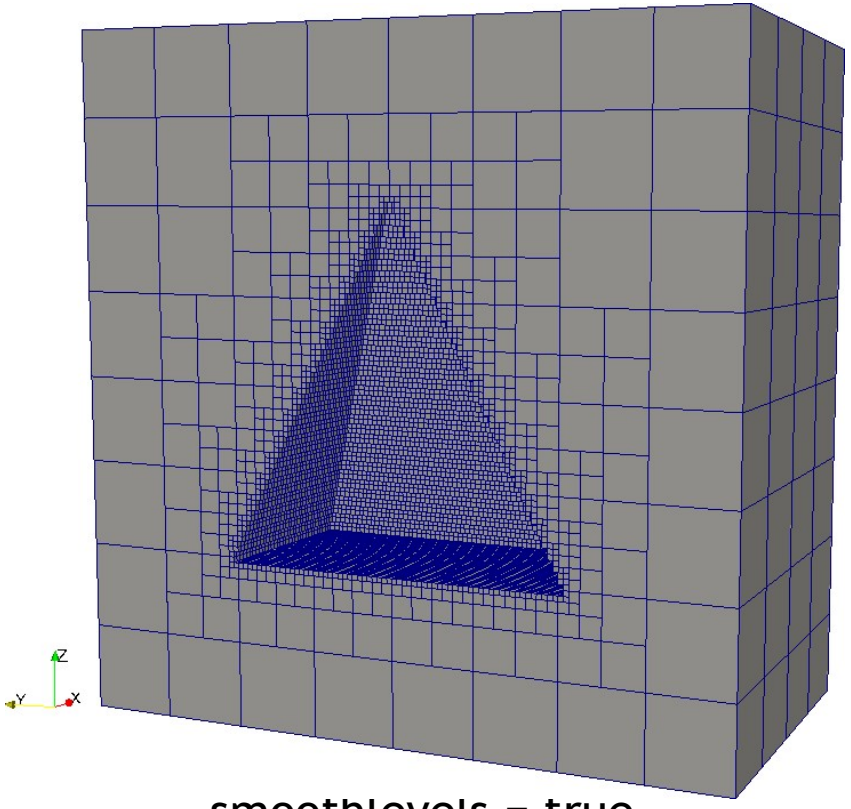
smoothbounds = true

Smooth level: Example

- Configurable - “smoothlevels = true/false”



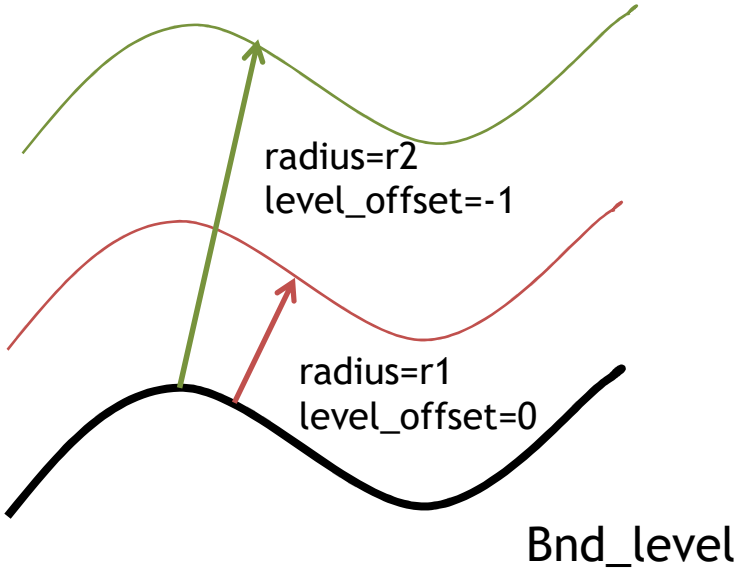
smoothlevels= false

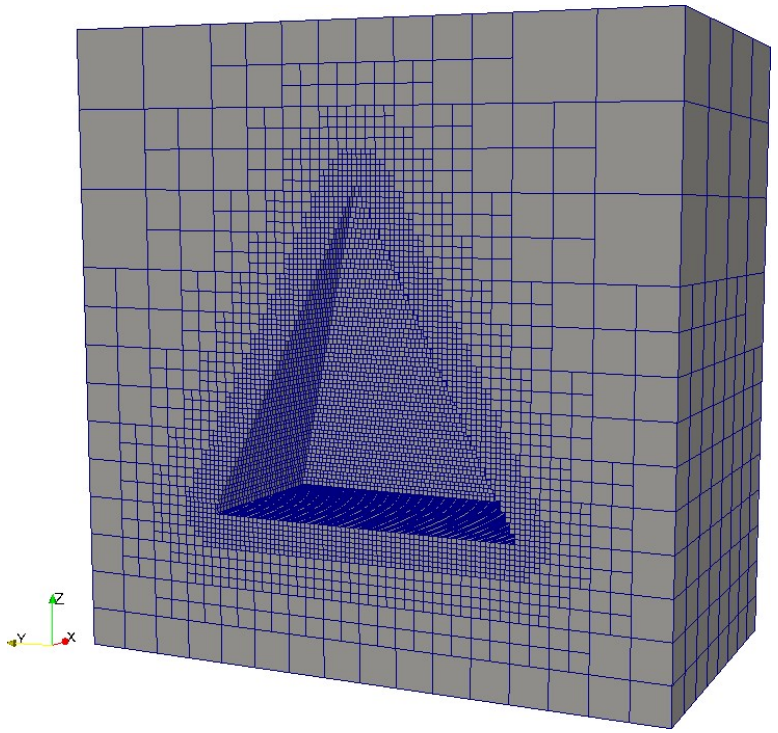


smoothlevels = true

Distance refinement

- Define the thickness of the boundary elements

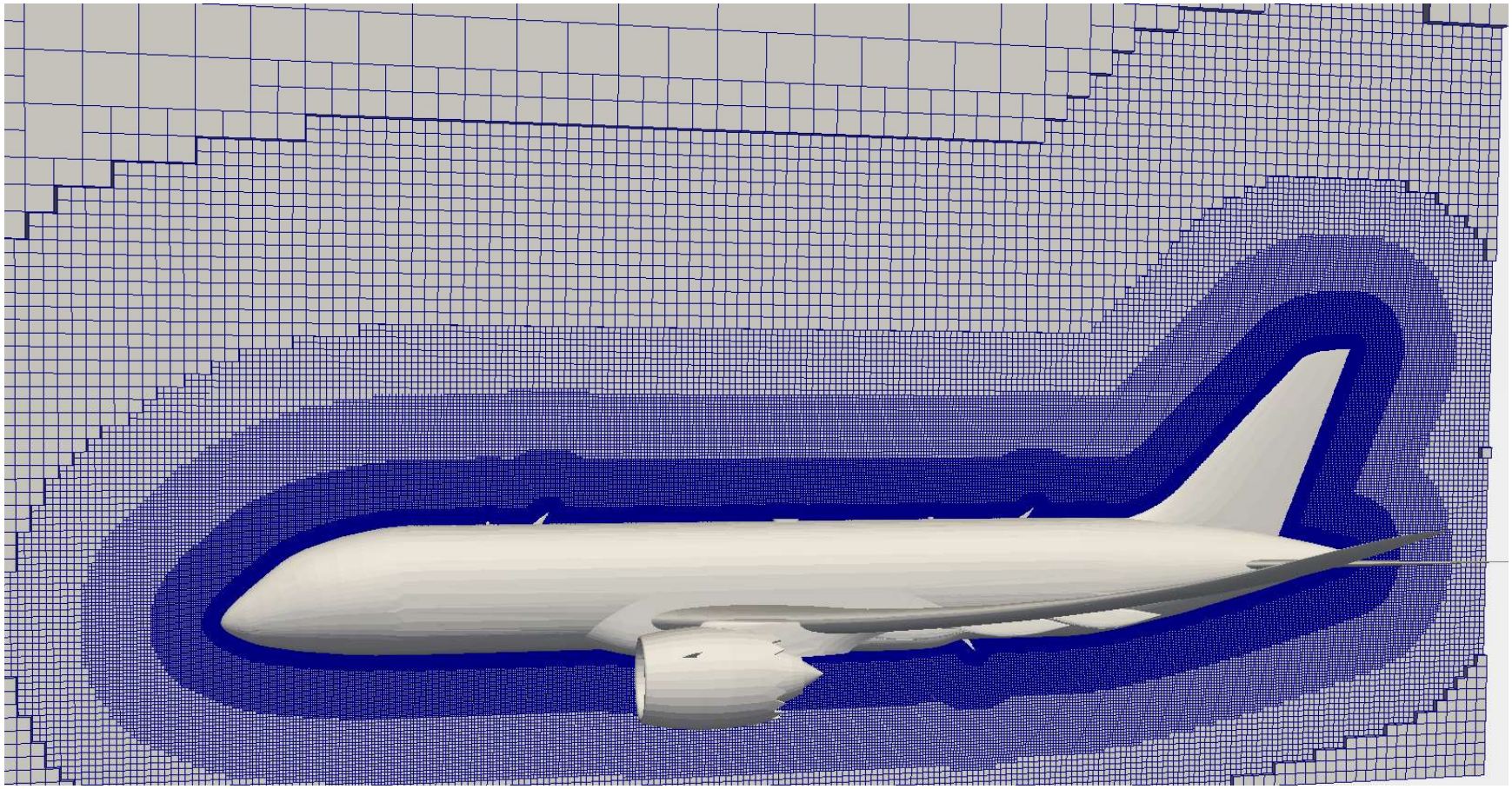




Algorithm: Build protoTree for distance refinement

- Level loop
  - Loop over the parents of the current level
    - If the parent is leaf and intersected by a boundary, then mark neighbors as has_boundary
    - If the parent is intersected by a boundary and distance refine object is defined for this boundary on the parent node, then
      - Create a spatial object with attribute “refinement” and geometry “sphere" with radius and level defined by user.
  - Loop over parents of current level
    - Refine towards boundary object (create children and inherit Intersected objects)

Example

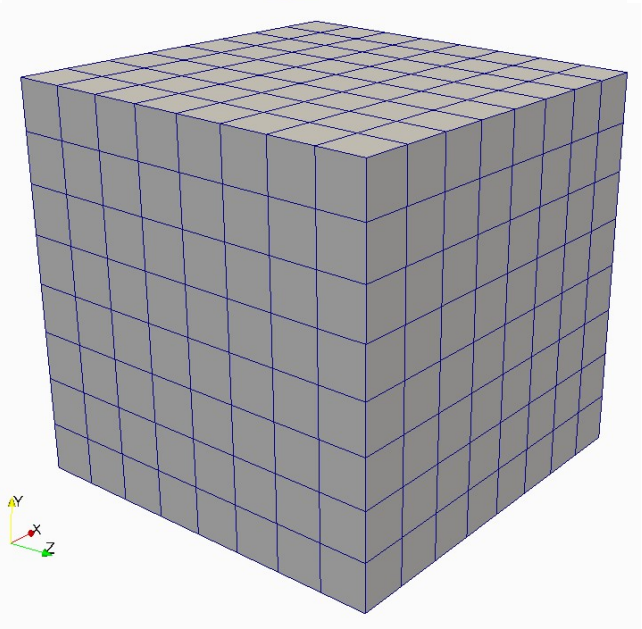


Configuration file: basic

```
-- Location to write mesh
folder = 'mesh/'

-- Bounding cube: The root node of the octree, defining the complete universe,
-- from which all elements are derived by recursive bisection.
-- The origin is the corner from which on the cube is spanned with the given
-- length in each direction.
bounding_cube = {
      origin = {-1.0, -1.0, -1.0},
      length = 2.0
}

-- A minimum level, by which all parts in the computational domain should at
-- least be resolved with. Default is 0.
minlevel = 3

-- element size
dx = bounding_cube.length/2^minlevel
```

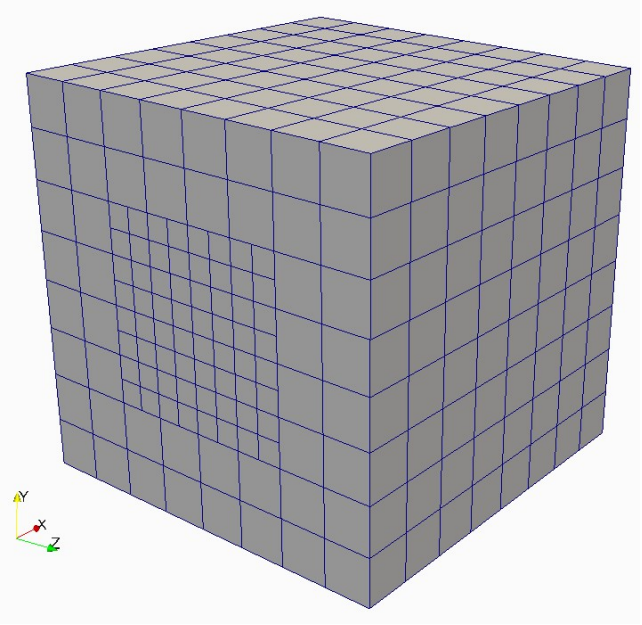
Configuration file: seed object

```
-- Spatial object: Is defined by an attribute and geometry attached to
this attribute.
-- Available attribute kinds: seed/boundary/refinement/periodic
-- Available geometry kinds: stl/canoND/sphere/cylinder/periodic

-- Define "seed" object to create fluid domain. Without this object,
seeder cannot create the mesh.
spatial_object = {
  attribute = {
    kind = 'seed'
  },
  geometry = {
    kind = 'canoND',
    object = {
      origin = { 0.0, 0.0, 0.0 }
    }
  }
}
```



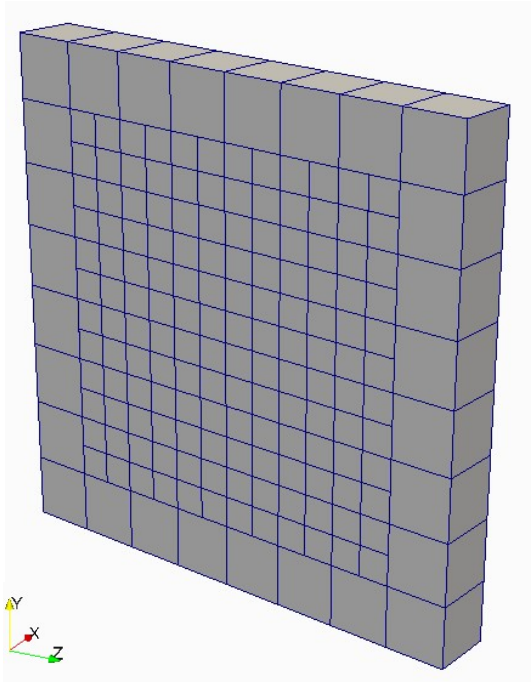
Configuration file: refinement object

```
-- Create multilevel mesh using “refinement” attribute.
-- Example to create refinement box
spatial_object = {
  attribute = {
    kind = ‘refinement’,
    level = 4
  },
  geometry = {
    kind = ‘canoND’,
    object = {
      origin = { -1.0, 0.25, 0.25 },
      vec = {
        { 2.0, 0.0, 0.0 },
        { 0.0, 0.5, 0.0 },
        { 0.0, 0.0, 0.5 }
      }
    }
  }
}
```



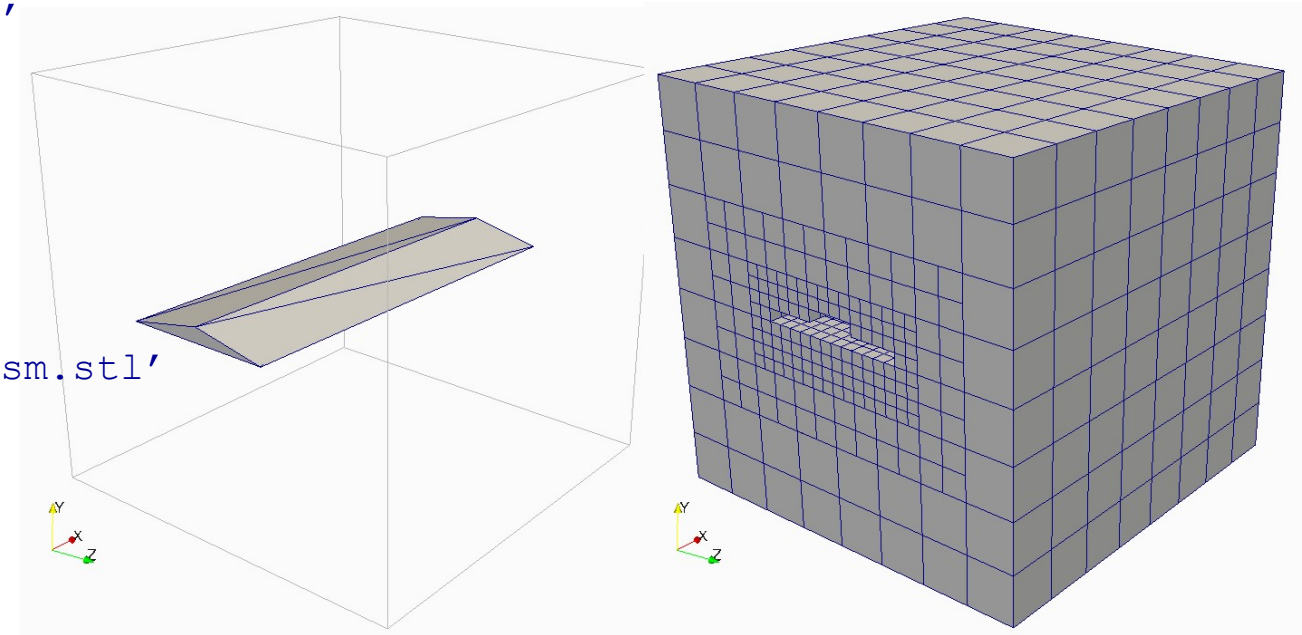
Configuration file: Periodic object

```
-- Periodic objects are used to create periodic boundaries i.e whatever comes out from
-- one side of the periodic plane enters on the opposite side of the plane.
-- Note: Ensure that plane normal is pointed away from the fluid domain.
spatial_object = {
  {
    attribute = {
    kind = 'periodic',
  },
  geometry = {
    kind = 'periodic',
    object = {
      plane1= {
        origin = { dx+dx/2, -1.0, -1.0 },
        vec = { { 0.0, 2.0, 0.0 }, { 0.0, 0.0, 2.0 } }
      },
      plane2 = {
        origin = { -dx/2., -1.0, -1.0 },
        vec = { { 0.0, 0.0, 2.0 }, {0.0, 2.0, 0.0} }
      }
    }
  },
-- NOTE: Place your seed object between the periodic planes
  {
    attribute = { kind = 'seed' },
    geometry = {
      kind = 'canoND',
      object = {
        origin = { dx/2.0, 0.0, 0.0 }
      }
    }
  }
}
```



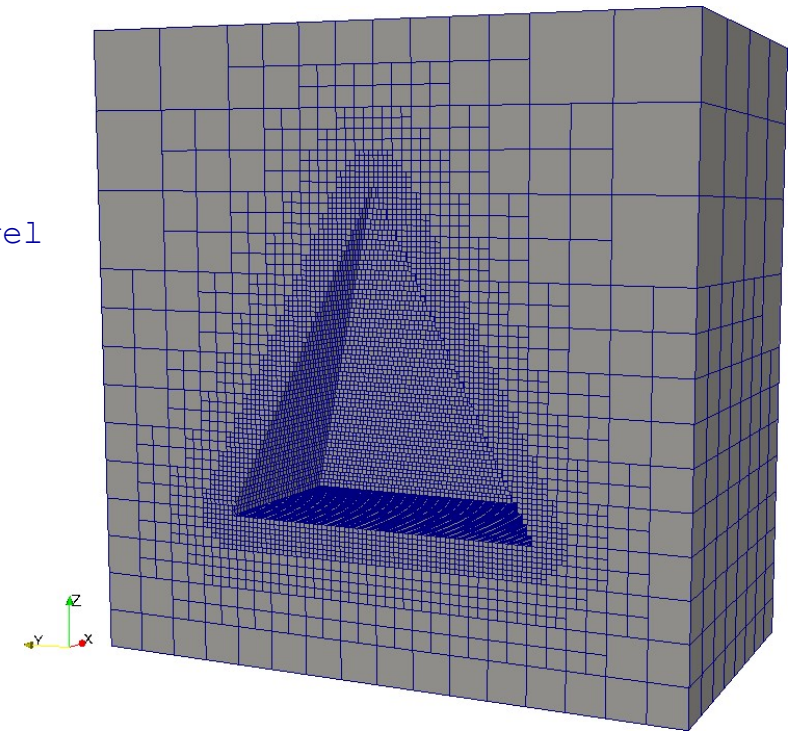
Configuration file: boundary object

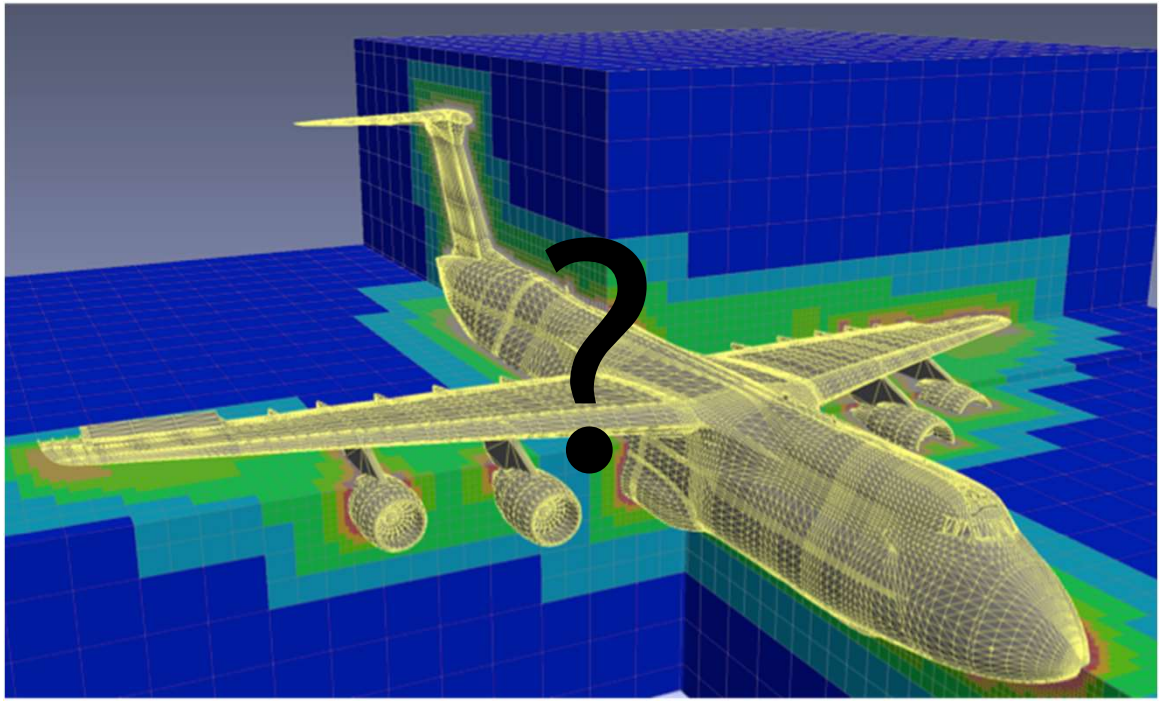
```
-- Example to add a boundary object.
-- NOTE: Label must be provided for boundary objects, to identify them
as boundaries in the solver
spatial_object = {
  attribute = {
    kind = 'boundary',
    label = 'prism',
    level = 4
  },
  geometry = {
    kind = 'stl',
    object = {
      filename = 'prism.stl'
    }
  }
}
```



Configuration file: Distance refinement

```
spatial_object = {
  attribute = {
    kind = 'boundary',
    level = bnd_level,
    distance_refine = {
      {
        -- Thinkness of the refinement
        radius = 0.1,
        -- Level offset from the attribute.level
        -- to refine elements within a given
        -- radius. Must be <=0
        level_offset = 0
      },
      {
        radius = 0.2,
        level_offset = -1
      }
    }
  }
}
```





Thanks for your attention ☺