Module - getLocalTopology

import mesh

In[2]:=
```
DumpGet["D:\\LocalData\\hashmial\\Research\\Vertex
    Model 3D\\sanity checks for functions\\meshGen_noise.mx"];
```

In[3]:=
```
yLim[[1]] = 0.;
edges = SetPrecision[edges, 10];
faceListCoords = SetPrecision[faceListCoords, 10];
(*convert faceListCoords into an association*)
indToPtsAssoc = SetPrecision[indToPtsAssoc, 10];
ptsToIndAssoc = KeyMap[SetPrecision[#, 10] &, ptsToIndAssoc];
xLim = SetPrecision[xLim, 10];
yLim = SetPrecision[yLim, 10];
faceListCoords = Map[Lookup[indToPtsAssoc, #] &, cellVertexGrouping, {2}];
```

In[11]:=
```
Clear@periodicRules;
With[{xlim1 = xLim[[1]], xlim2 = xLim[[2]], ylim1 = yLim[[1]], ylim2 = yLim[[2]]},
  periodicRules = Dispatch[{
     (*bottom right half*)
     {x_ /; x ≥ xlim2, y_ /; y ≤ ylim1, z_} :>
      SetPrecision[{x - xlim2, y + ylim2, z}, 10],
     (*right*)
     {x_ /; x ≥ xlim2, y_ /; ylim1 < y < ylim2, z_} :>
      SetPrecision[{x - xlim2, y, z}, 10],
     (*bottom*)
     {x_ /; xlim1 < x < xlim2, y_ /; y ≤ ylim1, z_} :>
      SetPrecision[{x, y + ylim2, z}, 10],
     (*bottom left half*)
     {x_ /; x ≤ xlim1, y_ /; y ≤ ylim1, z_} :>
      SetPrecision[{x + xlim2, y + ylim2, z}, 10],
     (*left half*)
     {x_ /; x ≤ xlim1, y_ /; ylim1 < y < ylim2, z_} :>
      SetPrecision[{x + xlim2, y, z}, 10],
     (*top-left half*)
     {x_ /; x ≤ xlim1, y_ /; y ≥ ylim2, z_} :>
      SetPrecision[{x + xlim2, y - ylim2, z}, 10],
     (*top*)
     {x_ /; xlim1 < x < xlim2, y_ /; y ≥ ylim2, z_} :>
      SetPrecision[{x, y - ylim2, z}, 10],
     (*top-right*)
     {x_ /; x ≥ xlim2, y_ /; y ≥ ylim2, z_} :> SetPrecision[{x - xlim2, y - ylim2, z}, 10]
    }];

  transformRules = Dispatch[{
     {x_ /; x ≥ xlim2, y_ /; y ≤ ylim1, _} :> SetPrecision[{-xlim2, ylim2, 0}, 10],
     {x_ /; x ≥ xlim2, y_ /; ylim1 < y < ylim2, _} :> SetPrecision[{-xlim2, 0, 0}, 10],
     {x_ /; xlim1 < x < xlim2, y_ /; y ≤ ylim1, _} :> SetPrecision[{0, ylim2, 0}, 10],
     {x_ /; x ≤ xlim1, y_ /; y ≤ ylim1, _} :> SetPrecision[{xlim2, ylim2, 0}, 10],
     {x_ /; x ≤ xlim1, y_ /; ylim1 < y < ylim2, _} :> SetPrecision[{xlim2, 0, 0}, 10],
     {x_ /; x ≤ xlim1, y_ /; y ≥ ylim2, _} :> SetPrecision[{xlim2, -ylim2, 0}, 10],
     {x_ /; xlim1 < x < xlim2, y_ /; y ≥ ylim2, _} :> SetPrecision[{0, -ylim2, 0}, 10],
     {x_ /; x ≥ xlim2, y_ /; y ≥ ylim2, _} :> SetPrecision[{-xlim2, -ylim2, 0}, 10],
     {___Real} :> SetPrecision[{0, 0, 0}, 10]}];
 ];
```

In[13]:=
```
(*
origcellOrient=<|MapIndexed[First[#2]→#1&, faceListCoords]|>;
boundaryCells=
 With[{ylim1=yLim[[1]],ylim2=yLim[[2]],xlim1=xLim[[1]],xlim2=xLim[[2]]},
  Union[First/@Position[origcellOrient,
      {x_/;x≥xlim2,__}|{x_/;x≤xlim1,__}|{_,y_/;y≥ylim2,_}|{_,y_/;y≤ylim1,_}]/.
    Key[x_]:> x]
 ];
*)
wrappedMat = AssociationThread[
   Keys[cellVertexGrouping] → Map[Lookup[indToPtsAssoc, #] /. periodicRules &,
     Lookup[cellVertexGrouping, Keys[cellVertexGrouping]], {2}]];
```

Miscellaneous F[x]

In[14]:=
```
triangulateFaces[faces_] := Block[{edgelen, ls, mean},
    (ls = Partition[#, 2, 1, 1];
       edgelen = Norm[SetPrecision[First[#] - Last[#], 10]] & /@ ls;
       mean = Total[edgelen * (Midpoint /@ ls)] / Total[edgelen];
       mean  = mean ~ SetPrecision ~ 10;
       Map[Append[#, mean] &, ls]) & /@ faces
  ];
```

In[15]:=
```
Clear@outerCellsFn;
outerCellsFn::Information = "the function finds the cells at the boundary";
outerCellsFn[faceListCoords_, vertexToCell_, ptsToIndAssoc_] :=
  With[{xlim1 = xLim[[1]], xlim2 = xLim[[2]], ylim1 = yLim[[1]], ylim2 = yLim[[2]]},
   Block[{boundaryCells, bcells, temp, res},
    temp = <|MapIndexed[First[#2] → #1 &, faceListCoords]|>;
    boundaryCells = Union[First /@ Position[temp,
          {x_ /; x ≥ xlim2, __} | {x_ /; x ≤ xlim1, __} |
           {_, y_ /; y ≥ ylim2, _} | {_, y_ /; y ≤ ylim1, _}] /. Key[x_] ⇢ x];
    bcells = KeyTake[faceListCoords, boundaryCells];
    res = Union@(Flatten@Lookup[vertexToCell,
           Lookup[ptsToIndAssoc,
            DeleteDuplicates@Cases[bcells,
                {x_ /; x ≥ xlim2, __} | {x_ /; x ≤ xlim1,
                  __} | {_, y_ /; y ≥ ylim2, _} | {_, y_ /; y ≤ ylim1, _}, {3}]
              /. periodicRules
           ]
          ] ~ Join ~ boundaryCells);
    res
   ]
  ];
```

In[18]:=
```
Clear@boundaryPtsPairing;
boundaryPtsPairing::Information =
  "the function pairs the points at the boundaries
    with corresponding mirror points";
boundaryPtsPairing[vertexToCell_, indToPtsAssoc_, ptsToIndAssoc_] :=
  Block[{outerpts, mirrorpairs, pt, mirror},
   outerpts = Keys@Select[vertexToCell, Length[#] ≠ 3 &];
   mirrorpairs = <|
       (pt = Lookup[indToPtsAssoc, #];
          If[
           pt[[1]] ≤ xLim[[1]] ||
            pt[[1]] ≥ xLim[[2]] || pt[[2]] ≤ yLim[[1]] || pt[[2]] ≥ yLim[[2]],
           mirror = ptsToIndAssoc[pt /. periodicRules];
           # → mirror,
           Nothing]) & /@ outerpts
       |> // KeySort;
   Map[Sort@* Flatten][List @@@ Normal@GroupBy[Normal@mirrorpairs, Last → First]]
  ];
```

getLocalTopology

In[21]:=

```
𝒟 = Rectangle[{First@xLim, First@yLim}, {Last@xLim, Last@yLim}];
getLocalTopology[ptsToIndAssoc_, indToPtsAssoc_, vertexToCell_,
    cellVertexGrouping_, wrappedMat_, faceListCoords_][vertices_] :=
  Block[{localtopology = <||>, wrappedcellList = {}, vertcellconns,
    localcellunion, v, wrappedcellpos, vertcs = vertices, rl1, rl2,
    transVector, wrappedcellCoords, wrappedcells, vertOutofBounds,
    shiftedPt, transvecList = {}, $faceListCoords = Values@faceListCoords,
    vertexQ, boundsCheck, rules, extractcellkeys, vertind,
    cellsconnected, wrappedcellsrem},
   vertexQ = MatchQ[vertices, {__?NumberQ}];
   If[vertexQ,
    (vertcellconns =
      AssociationThread[{#} , {vertexToCell[ptsToIndAssoc[#]]}] &@vertices;
     vertcs = {vertices};
     localcellunion = Flatten[Values@vertcellconns]),
    (vertcellconns = AssociationThread[# ,
        Lookup[vertexToCell, Lookup[ptsToIndAssoc, #]]] &@vertices;
     localcellunion = Union@Flatten[Values@vertcellconns])
   ];

   If[localcellunion ≠ {},
    AppendTo[localtopology,
     Thread[localcellunion →
       Map[Lookup[indToPtsAssoc, #] &, cellVertexGrouping /@ localcellunion, {2}]]
    ]
   ];
   (* condition to be an internal edge: both vertices should have 3 neighbours *)
   (* if a vertex has 3 cells in its local neighbourhood then the entire
     network topology about the vertex is known → no wrapping required *)
   (* else we need to wrap around the vertex because other cells
     are connected to it → periodic boundary conditions *)
   With[{vert = #},
      vertind = ptsToIndAssoc[vert];
      cellsconnected = vertexToCell[vertind];
      If[Length[cellsconnected] ≠ 3,
       If[(𝒟~RegionMember~Most[vert]),
          (*Print["vertex inside bounds"];*)
          v = vert;
          With[{x = v[[1]], y = v[[2]]}, boundsCheck =
            (x == xLim[[1]] || x == xLim[[2]] || y == yLim[[1]] || y == yLim[[2]])];
          extractcellkeys = If[boundsCheck,
            {rl1, rl2} = {v, v /. periodicRules};
            rules = Block[{x$},
              With[{r = rl1, s = rl2},
               DeleteDuplicates[
                HoldPattern[SameQ[x$, r]] || HoldPattern[SameQ[x$, s]]]
              ]
             ];
            Position @@ With[{rule = rules},
              Hold[wrappedMat, x_ /; ReleaseHold@rule, {3}]
```

```
    ],
   Position[wrappedMat, x_ /; SameQ[x, v], {3}]
  ];
 (* find cell indices that are attached to the vertex in wrappedMat *)
 wrappedcellpos = DeleteDuplicatesBy[
   Cases[extractcellkeys,
    {Key[p : Except[Alternatives @@ Join[localcellunion,
           Flatten@wrappedcellList]]], y__} :> {p, y}],
   First];
 (*wrappedcellpos = wrappedcellpos/.
    {Alternatives@@Flatten[wrappedcellList],__} :> Sequence[];*)
 (* if a wrapped cell has not been considered earlier (i.e. is new)
  then we translate it to the position of the vertex *)
 If[wrappedcellpos ≠ {},
  If[vertexQ,
   transVector = SetPrecision[(v - Extract[$faceListCoords, Replace[#,
           {p_, q__} :> {Key[p], q}, {1}]]) & /@ wrappedcellpos, 10],
   (* call to function is enquiring an edge and not a vertex*)
   transVector =
    SetPrecision[(v - Extract[$faceListCoords, #]) & /@ wrappedcellpos, 10]
  ];
  wrappedcellCoords = MapThread[#1 → Map[Function[x,
        SetPrecision[x + #2, 10]], $faceListCoords[[#1]], {2}] &,
    {First /@ wrappedcellpos, transVector}];
  wrappedcells = Keys@wrappedcellCoords;
  AppendTo[wrappedcellList, Flatten@wrappedcells];
  AppendTo[transvecList, transVector];
  AppendTo[localtopology, wrappedcellCoords];
 ],
 (* the else clause: vertex is out of bounds *)
 (*Print["vertex out of bounds"];*)
 vertOutofBounds = vert;
 (* translate the vertex back into mesh *)
 transVector = vertOutofBounds /. transformRules;
 shiftedPt = SetPrecision[vertOutofBounds + transVector, 10];
 (* ------------- CORE B ------------- *)
 (* find which cells the
  shifted vertex is a part of in the wrapped matrix *)
 wrappedcells = Complement[
   Union@Cases[Position[wrappedMat, x_ /;
        SameQ[x, shiftedPt] || SameQ[x, vertOutofBounds], {3}],
      x_Key :> Sequence @@ x, {2}] /. Alternatives @@
      localcellunion → Sequence[],
   Flatten@wrappedcellList];

 (*forming local topology now that we know the wrapped cells *)
 If[wrappedcells ≠ {},
  AppendTo[wrappedcellList, Flatten@wrappedcells];
  wrappedcellCoords = AssociationThread[wrappedcells,
    Map[Lookup[indToPtsAssoc, #] &,
     cellVertexGrouping[#] & /@ wrappedcells, {2}]];
  With[{opt = (vertOutofBounds /. periodicRules) | vertOutofBounds},
   Block[{pos, vertref, transvec},
    Do[
     With[{cellcoords = wrappedcellCoords[cell]},
```

```
                pos = FirstPosition[cellcoords /. periodicRules, opt];
                If[Head[pos] === Missing,
                 pos = FirstPosition[
                     Chop[cellcoords /. periodicRules, 10^-6], Chop[opt, 10^-6]];
                ];
                vertref = Extract[cellcoords, pos];
                transvec = SetPrecision[vertOutofBounds - vertref, 10];
                AppendTo[transvecList, transvec];
                AppendTo[localtopology,
                 cell → Map[SetPrecision[# + transvec, 10] &, cellcoords, {2}]];
               ], {cell, wrappedcells}]
             ];
            ];
           ];
           (* to detect wrapped cells not detected by CORE B*)
           (* ------------- CORE C ------------- *)
           Block[{pos, celllocs, ls, transvec, assoc, tvecLs = {}, ckey},
            ls = Union@Flatten@Join[cellsconnected, wrappedcells];
            If[Length[ls] ≠ 3,
             pos = Position[faceListCoords, x_ /; SameQ[x, shiftedPt], {3}];
             celllocs = DeleteDuplicatesBy[Cases[pos, Except[{Key[Alternatives @@ ls],
                   __}]], First] /. {Key[x_], z__} :→ {Key[x], {z}};
             If[celllocs ≠ {},
              celllocs = Transpose@celllocs;
              assoc = <|
                MapThread[
                 (transvec = SetPrecision[vertOutofBounds -
                      Extract[faceListCoords[Sequence @@ #1], #2], 10];
                  ckey = Identity @@ #1;
                  AppendTo[tvecLs, transvec];
                  ckey → Map[SetPrecision[Lookup[indToPtsAssoc, #] + transvec,
                      10] &, cellVertexGrouping[Sequence @@ #1], {2}]
                 ) &, celllocs]
                |>;
              AppendTo[localtopology, assoc];
              AppendTo[wrappedcellList, Keys@assoc];
              AppendTo[transvecList, tvecLs];
             ];
            ];
           ];
          ];
         ];
        ] & /@ vertcs;

       transvecList = Which[
         MatchQ[transvecList, {{{__?NumberQ}}}], First[transvecList],
         MatchQ[transvecList, {{__?NumberQ} ..}], transvecList,
         True, transvecList //. {x___, {p : {__?NumberQ} ..}, y___} :→ {x, p, y}
        ];
       {localtopology, Flatten@wrappedcellList, transvecList}
      ];
```

In[ ]:=
```
(* to fasten speed of pattern matching we
 only extract the outermost cells for the wrappedMat *)
```

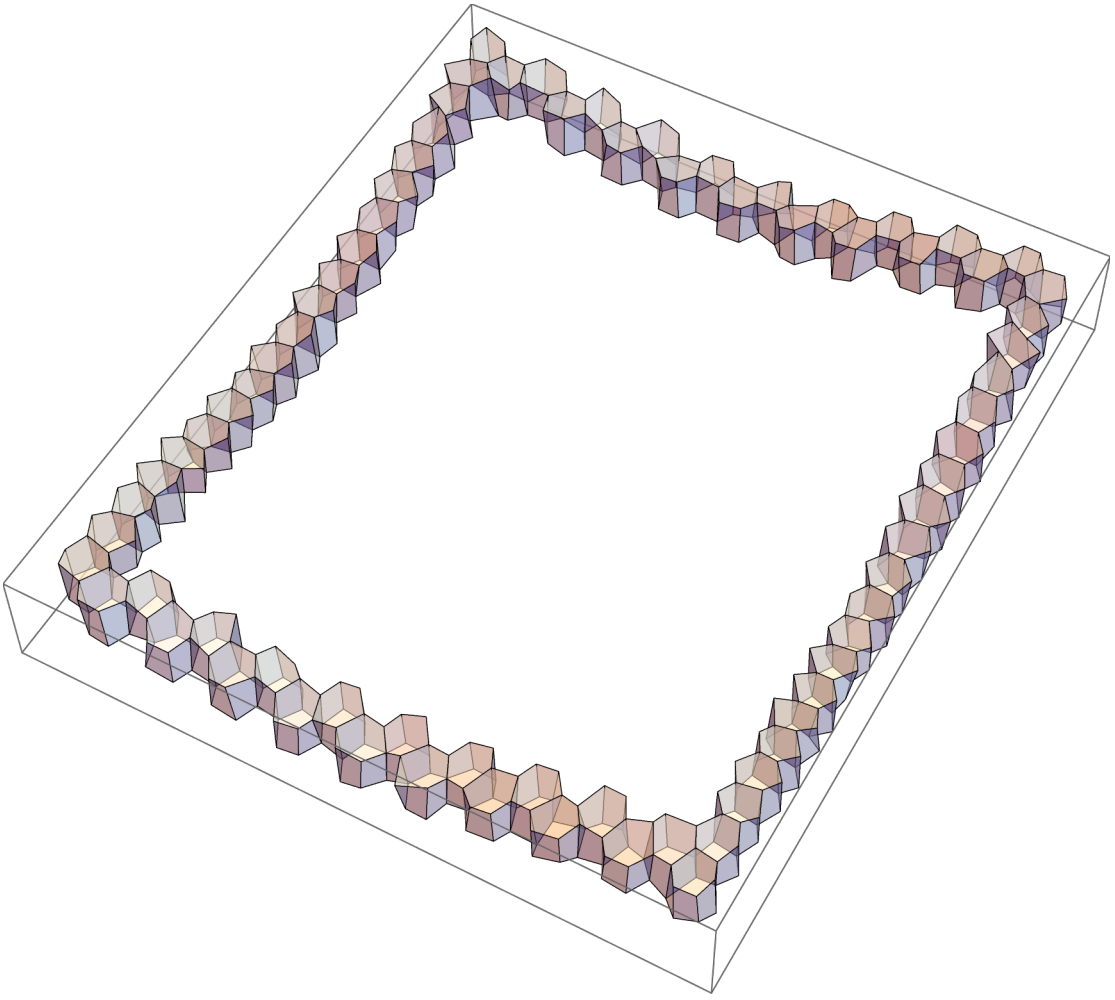
In[23]:=
```
boundarycells = outerCellsFn[faceListCoords, vertexToCell, ptsToIndAssoc]
```

Out[23]=
```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 40, 41, 60, 61,
 80, 81, 100, 101, 120, 121, 140, 141, 160, 161, 180, 181, 200, 201, 220, 221, 240,
 241, 260, 261, 280, 281, 300, 301, 320, 321, 340, 341, 360, 361, 380, 381, 382, 383,
 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 400}
```

In[24]:=
```
wrappedMatSel = wrappedMat ~ KeyTake ~ boundarycells;
```

In[25]:=
```
plt = Graphics3D[{Opacity[0.5], White, Polyhedron /@ Map[Lookup[indToPtsAssoc, #] &,
      Lookup[cellVertexGrouping, boundarycells], {2}]}, ImageSize → Large]
```

Out[25]=



In[ ]:=
```
(* neighbour-count for vertex *)
```

In[26]:=
```
Keys[getLocalTopology[ptsToIndAssoc, indToPtsAssoc, vertexToCell, cellVertexGrouping,
        wrappedMatSel, faceListCoords][indToPtsAssoc[#]] // First] & /@
   Range[Max@ptsToIndAssoc] // Counts@*Map[Length] // AbsoluteTiming
```

Out[26]=
```
{1.0666, ‹|3 → 1760|›}
```

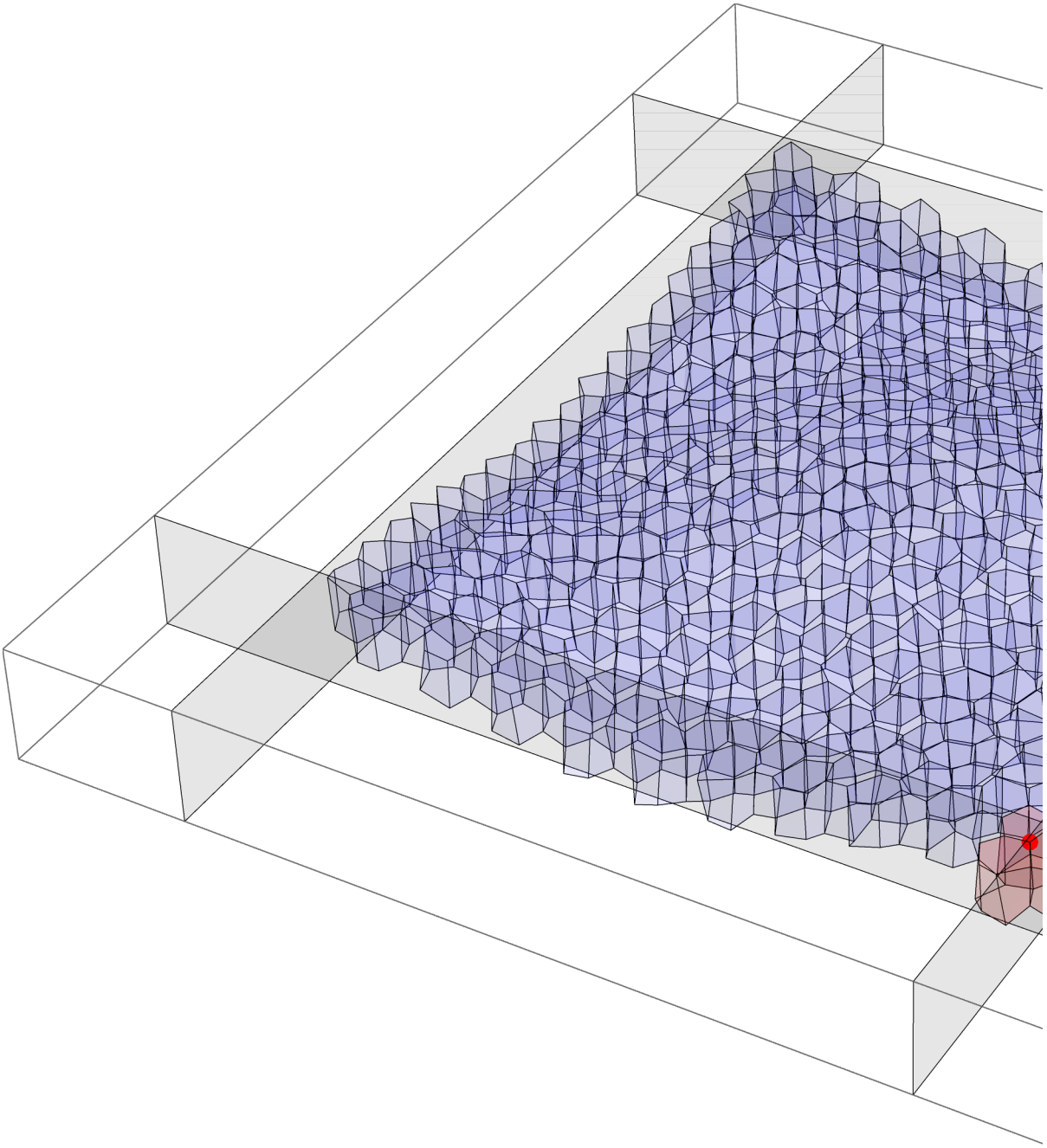
In[ ]:=
```
(* neighbour-count for vertex is slow if slightly slower if we use wrappedMat *)
```

In[27]:=
```
Keys[getLocalTopology[ptsToIndAssoc, indToPtsAssoc, vertexToCell, cellVertexGrouping,
        wrappedMat, faceListCoords][indToPtsAssoc[#]] // First] & /@
   Range[Max@ptsToIndAssoc] // Counts@*Map[Length] // AbsoluteTiming
```

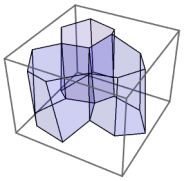
Out[27]=
```
{4.1722, ‹|3 → 1760|›}
```

selected vertex

In[28]:=
```
Block[{xx, yy, zz, edgelen, edgesel, keys, dset, id = 1680, point},
 point = indToPtsAssoc[id];
 {xx, yy, zz} = getLocalTopology[ptsToIndAssoc, indToPtsAssoc, vertexToCell,
    cellVertexGrouping, wrappedMatSel, faceListCoords][indToPtsAssoc[id]];
 keys = Keys@xx;
 Print@Graphics3D[
   {{Opacity[0.05], Blue, Polyhedron /@ Values@faceListCoords}, Red, PointSize[0.01],
    Point@point, Opacity[0.1], Red, Polyhedron /@ Values[xx],
    Black, InfinitePlane[{{0, 0, 0}, {0, xLim[[2]], 0}, {0, xLim[[2]], 1}}],
    InfinitePlane[
     {{0, yLim[[2]], 0}, {xLim[[2]], yLim[[2]], 0}, {xLim[[2]], yLim[[2]], 1}}],
    InfinitePlane[{{xLim[[2]], yLim[[2]], 0}, {xLim[[2]], yLim[[2]], 1},
      {xLim[[2]], yLim[[1]], 1}}],
    InfinitePlane[{{xLim[[2]], yLim[[1]], 0}, {xLim[[2]], yLim[[1]], 1},
      {0, yLim[[1]], 1}}]},
   ImageSize → 1024];
 <|{"vertex" → id,
   "vertices in topology" → Length@DeleteDuplicates@Flatten[Values@xx, 2],
   "cell indices" → keys,
   "translated cell indices" → yy, (*"transvec"→zz,*)
   "localtopology" →
    Graphics3D[{Opacity[0.1], RandomColor[], Polyhedron /@ Lookup[xx, keys],
      Blue, If[yy ≠ {}, Polyhedron /@ Lookup[xx, yy]]}, ImageSize → Tiny]}|>
]
```



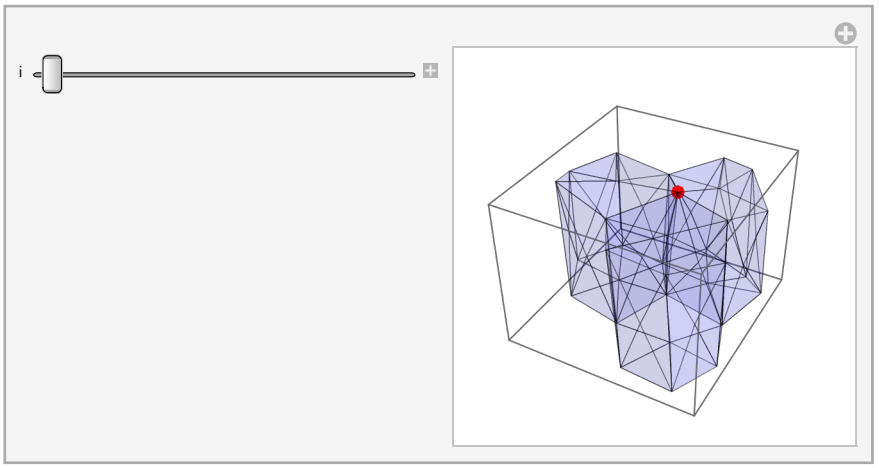
Out[28]= ⟨| vertex → 1680, vertices in topology → 26, cell indices → {381, 20, 400},

translated cell indices → {20, 400}, localtopology →  |⟩

vertex to cell topology

In[35]:=
```
Manipulate[
 Graphics3D[{{Opacity[0.1], Blue, EdgeForm[{Opacity[0.5], Black}],
    Values@Map[Polyhedron@* (Flatten[#, 1] &) @*triangulateFaces,
      First@getLocalTopology[ptsToIndAssoc, indToPtsAssoc, vertexToCell,
         cellVertexGrouping, wrappedMatSel, faceListCoords][indToPtsAssoc[i]]]},
   {Red, PointSize[0.04], Point@indToPtsAssoc[i]}}, ImageSize → Small],
 {i, 1, Max[ptsToIndAssoc], 1}, SaveDefinitions → True
]
```

Out[35]=



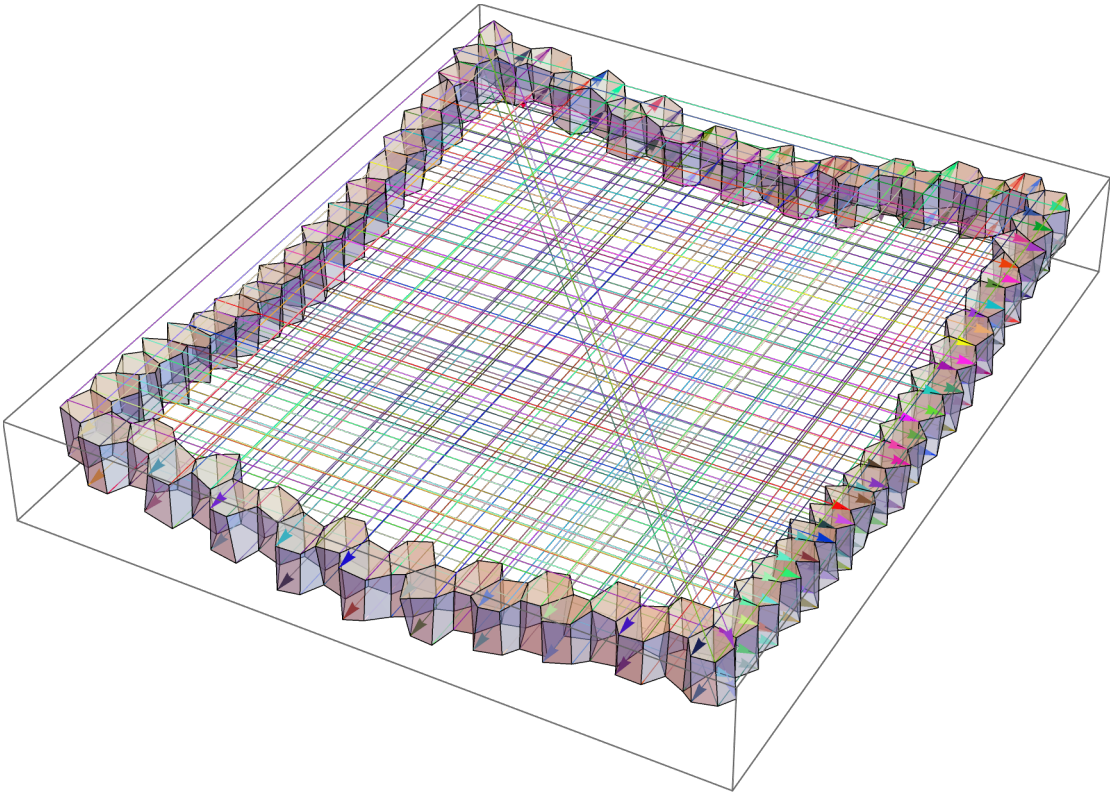
In[ ]:=
```
(* checking boundary pt pairing *)
```

In[36]:=
```
{timing, arrows} =
 boundaryPtsPairing[vertexToCell, indToPtsAssoc, ptsToIndAssoc] // AbsoluteTiming
```

Out[36]=
```
{0.0096111, {{4, 1678}, {6, 158}, {10, 1682}, {11, 163, 1681}, {12, 162}, {16, 1686},
  {32, 1694}, {40, 1698}, {52, 1704}, {56, 1706}, {72, 1714}, {84, 1720}, {88, 1722},
  {108, 1732}, {116, 1736}, {120, 1738}, {128, 1742}, {132, 1744}, {136, 1746},
  {140, 1748}, {144, 1750}, {152, 1754}, {5, 159, 1677}, {157, 167}, {168, 246},
  {169, 332}, {161, 171}, {172, 248}, {251, 328}, {165, 329}, {254, 331}, {327, 335},
  {336, 414}, {337, 500}, {330, 339}, {340, 416}, {419, 496}, {422, 499}, {333, 497},
  {495, 503}, {504, 582}, {498, 507}, {508, 584}, {587, 664}, {501, 665},
  {590, 667}, {505, 668}, {669, 833}, {663, 671}, {672, 750}, {666, 675},
  {676, 752}, {755, 832}, {758, 835}, {673, 836}, {831, 839}, {840, 918},
  {834, 843}, {844, 920}, {923, 1000}, {837, 1001}, {926, 1003}, {841, 1004},
  {999, 1007}, {1008, 1086}, {1002, 1011}, {1012, 1088}, {1091, 1168}, {1005, 1169},
  {1094, 1171}, {1009, 1172}, {1173, 1337}, {1167, 1175}, {1176, 1254}, {1177, 1340},
  {1170, 1179}, {1180, 1256}, {1259, 1336}, {1262, 1339}, {1341, 1505}, {1335, 1343},
  {1344, 1422}, {1345, 1508}, {1338, 1347}, {1348, 1424}, {1427, 1504}, {1430, 1507},
  {1509, 1673}, {1503, 1511}, {1512, 1590}, {1506, 1515}, {1516, 1592}, {1595, 1672},
  {1598, 1675}, {1513, 1676}, {1671, 1679}, {160, 1680, 1758}, {1674, 1683},
  {164, 1684, 1760}, {3, 1685}, {9, 1687}, {20, 1688}, {15, 1689}, {24, 1690},
  {19, 1691}, {28, 1692}, {23, 1693}, {27, 1695}, {36, 1696}, {31, 1697}, {35, 1699},
  {44, 1700}, {39, 1701}, {48, 1702}, {43, 1703}, {47, 1705}, {51, 1707}, {60, 1708},
  {55, 1709}, {64, 1710}, {59, 1711}, {68, 1712}, {63, 1713}, {67, 1715}, {76, 1716},
  {71, 1717}, {80, 1718}, {75, 1719}, {79, 1721}, {83, 1723}, {92, 1724}, {87, 1725},
  {96, 1726}, {91, 1727}, {100, 1728}, {95, 1729}, {104, 1730}, {99, 1731},
  {103, 1733}, {112, 1734}, {107, 1735}, {111, 1737}, {115, 1739}, {124, 1740},
  {119, 1741}, {123, 1743}, {127, 1745}, {131, 1747}, {135, 1749}, {139, 1751},
  {148, 1752}, {143, 1753}, {147, 1755}, {156, 1756}, {151, 1757}, {155, 1759}}}
```

In[37]:=
```
Show[plt, Graphics3D[{Arrowheads[Medium],
   Map[{RandomColor[], Arrow@Lookup[indToPtsAssoc, #]} &, arrows]}, ImageSize → Large]]
```

Out[37]=



checks

In[38]:=
```
indToPtsAssocC = indToPtsAssoc;
```

In[39]:=
```
indToPtsAssocC[328] = {2.025000000000000355271367880050092935 56`10.,
   15.650000000000000 3553`10., 0.280269296720683180890176799948676489 29`10.};
indToPtsAssocC[329] = {1.575000000000000017763568394002504646778`10.,
   15.650000000000000 3553`10., 0.359669334024670883653840292026870884`10.};
indToPtsAssocC[331] = {2.025000000000000355271367880050092935 56`10.,
   15.650000000000000 3553`10., -0.719730703279316763598671968793496489 52`10.};
indToPtsAssocC[332] = {1.575000000000000017763568394002504646778`10.,
   15.650000000000000 3553`10., -0.640330665975329171857310939230956137 18`10.};
indToPtsAssocC[165] = {1.575000000000000017763568394002504646778`10.,
   0, 0.359669334024670883653840292026870884`10.};
indToPtsAssocC[169] = {1.575000000000000017763568394002504646778`10.,
   0, -0.640330665975329171857310939230956137 18`10.};
indToPtsAssocC[251] = {2.025000000000000355271367880050092935 56`10.,
   0, 0.280269296720683180890176799948676489 29`10.};
indToPtsAssocC[254] = {2.025000000000000355271367880050092935 56`10.,
   0, -0.719730703279316763598671968793496489 52`10.};
```

In[47]:=
```
wrappedMatC = AssociationThread[
   Keys[cellVertexGrouping] → Map[Lookup[indToPtsAssocC, #] /. periodicRules &,
     Lookup[cellVertexGrouping, Keys[cellVertexGrouping]], {2}]];
```

In[48]:=
```
faceListCoordsC = Map[Lookup[indToPtsAssocC, #] &, cellVertexGrouping, {2}];
```

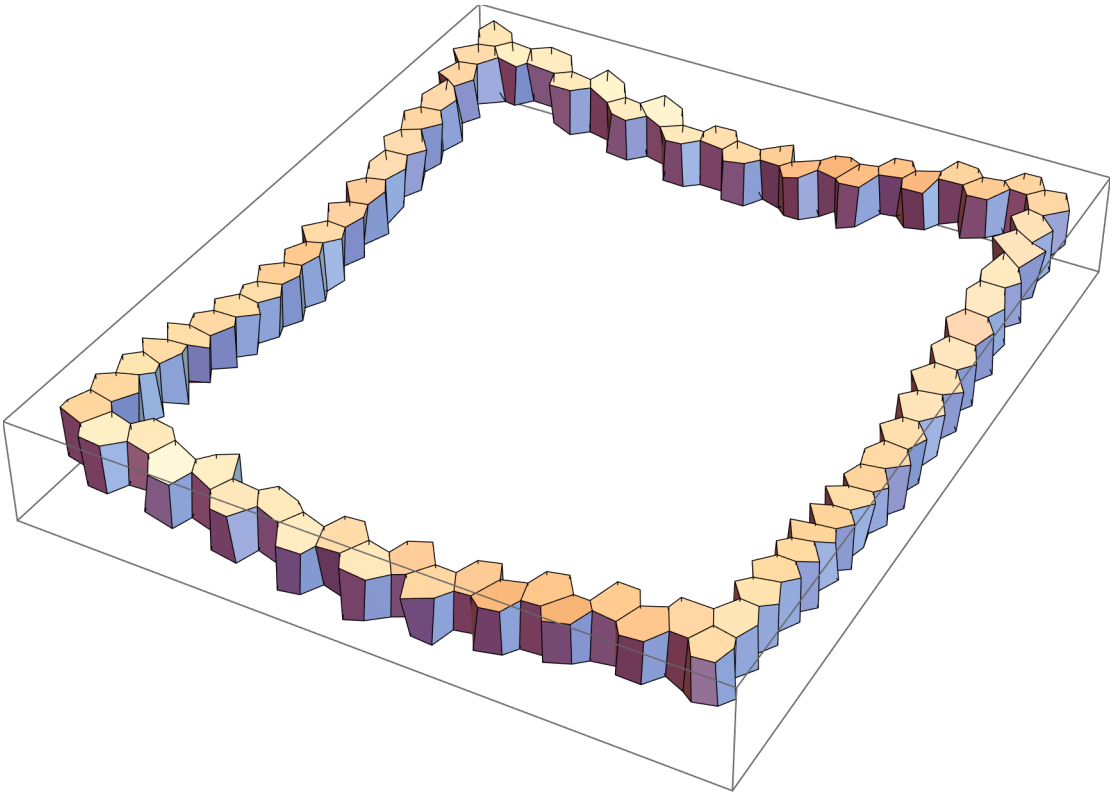
In[49]:=
```
ptsToIndAssocC = AssociationMap[Reverse, indToPtsAssocC];
```

In[50]:=
```
boundarycellsC = outerCellsFn[faceListCoordsC, vertexToCell, ptsToIndAssocC]
```

Out[50]=
```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 40, 41, 60, 61,
 80, 81, 100, 101, 120, 121, 140, 141, 160, 161, 180, 181, 200, 201, 220, 221, 240,
 241, 260, 261, 280, 281, 300, 301, 320, 321, 340, 341, 360, 361, 380, 381, 382, 383,
 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 400}
```

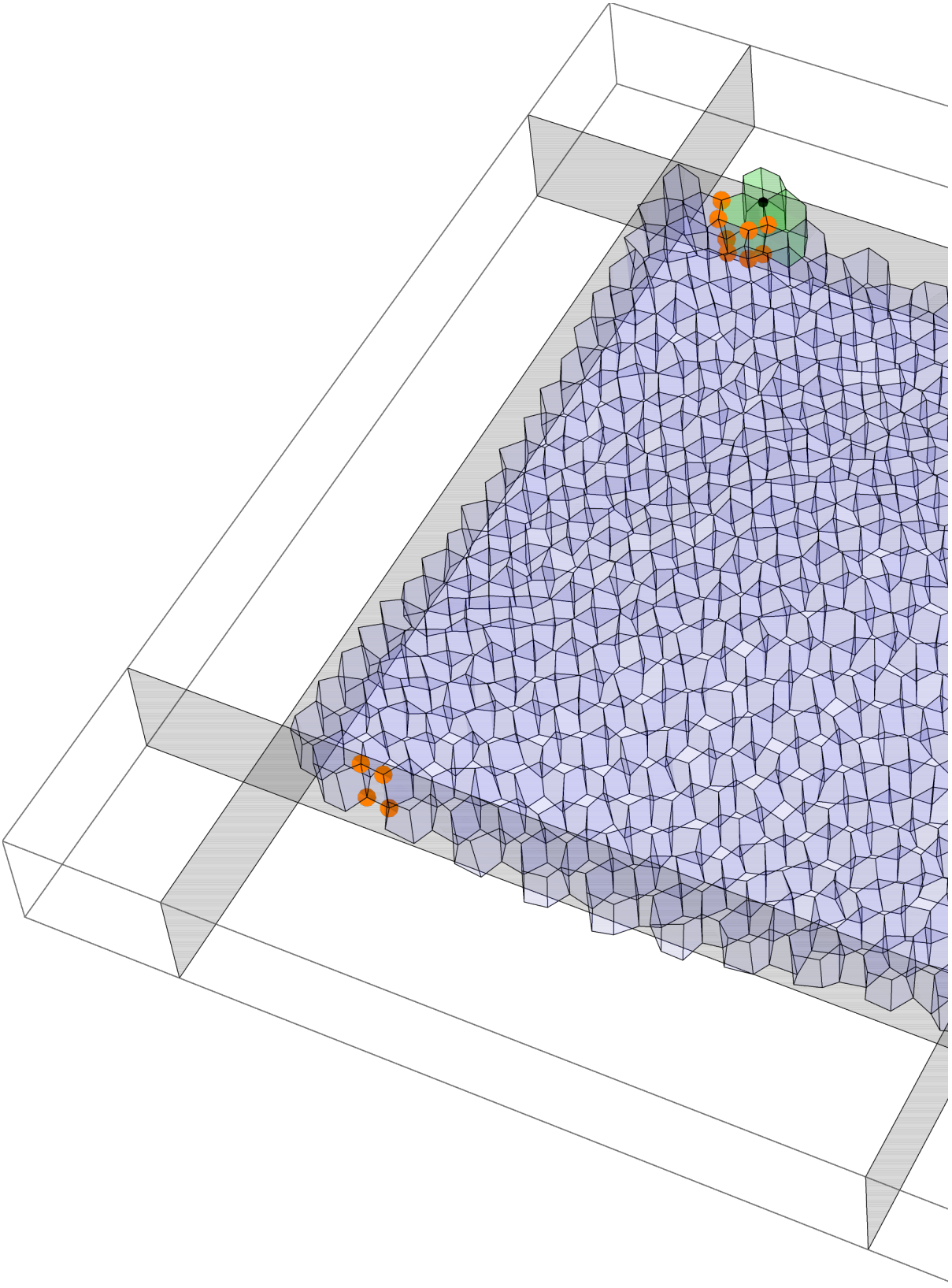
In[51]:=
```
Graphics3D[Polyhedron /@ Map[Lookup[indToPtsAssoc, #] &,
    Lookup[cellVertexGrouping, boundarycells], {2}], ImageSize → Large]
```

Out[51]=

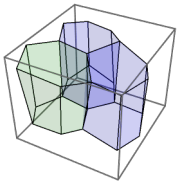


In[52]:=
```
wrappedMatCSel = KeyTake[wrappedMatC, boundarycellsC];
```

In[53]:=
```
Block[{xx, yy, zz, edgelen, edgesel, keys, dset, id = 328, point},
 point = indToPtsAssocC[id];
 {xx, yy, zz} = getLocalTopology[ptsToIndAssocC, indToPtsAssocC, vertexToCell,
    cellVertexGrouping, wrappedMatCSel, faceListCoordsC][indToPtsAssocC[id]];
 keys = Keys@xx;
 Print@Graphics3D[{{Opacity[0.05], Blue, Polyhedron /@ Values@faceListCoordsC},
    Black, PointSize[0.007], Point@point, Green, Point[point /. periodicRules],
    PointSize[0.012], Orange, Map[Point, wrappedMatC[60], {2}],
    Opacity[0.15], Green, Polyhedron /@ Values[xx],
    Black, InfinitePlane[{{0, 0, 0}, {0, xLim[[2]], 0}, {0, xLim[[2]], 1}}],
    InfinitePlane[
     {{0, yLim[[2]], 0}, {xLim[[2]], yLim[[2]], 0}, {xLim[[2]], yLim[[2]], 1}}],
    InfinitePlane[{{xLim[[2]], yLim[[2]], 0}, {xLim[[2]], yLim[[2]], 1},
      {xLim[[2]], yLim[[1]], 1}}],
    InfinitePlane[{{xLim[[2]], yLim[[1]], 0}, {xLim[[2]], yLim[[1]], 1},
      {0, yLim[[1]], 1}}]},
   ImageSize → 1024];
 <|{"vertex" → id,
   "vertices in topology" → Length@DeleteDuplicates@Flatten[Values@xx, 2],
   "cell indices" → keys, "translated cell indices" → yy, (*"transvec"→zz,*)
   "localtopology" →
    Graphics3D[{Opacity[0.1], RandomColor[], Polyhedron /@ Lookup[xx, keys],
      Blue, If[yy ≠ {}, Polyhedron /@ Lookup[xx, yy]]}, ImageSize → Tiny]}|>
]
```



Out[53]= ⟨| vertex → 328, vertices in topology → 26, cell indices → {60, 41, 61},

translated cell indices → {41, 61}, localtopology →  |⟩

In[55]:=
```
Keys[getLocalTopology[ptsToIndAssocC, indToPtsAssocC, vertexToCell, cellVertexGrouping,
        wrappedMatCSel, faceListCoordsC][indToPtsAssocC[#]] // First] & /@
   Range[Max@ptsToIndAssocC] // Counts@*Map[Length] // AbsoluteTiming
```

Out[55]= {0.817395, ‹| 3 → 1760 |›}