

2D VERTEX MODEL

geometrical f(x)

```
In[1]:= Clear[getCounterClockwise];  
getCounterClockwise[vertex_, vertices_] := Block[{pos, v},  
  pos = First@@Position[vertices, vertex];  
  If[pos == Length[vertices], pos = 1, pos += 1];  
  vertices[[pos]]  
];
```

```
In[3]:= Clear[getClockwise];  
getClockwise[vertex_, vertices_] := Block[{ls, pos},  
  pos = First@@Position[vertices, vertex];  
  If[pos == 1, pos = Length[vertices], pos -= 1];  
  vertices[[pos]]  
];
```

```
In[5]:= getCounterClockwise[{xi, yi}, {{xi-1, yi-1}, {xi, yi}, {xi+1, yi+1}, d, e}]
```

```
Out[5]= {x1+i, y1+i}
```

```
In[6]:= getClockwise[{xi, yi}, {{xi-1, yi-1}, {xi, yi}, {xi+1, yi+1}, d, e}]
```

```
Out[6]= {x-1+i, y-1+i}
```

```
In[7]:= Clear[areaOfPolygon];  
areaOfPolygon[cells_ /; Head[cells] === Association] := Map[Area@*Polygon, cells];
```

```
In[9]:= Clear[areaPolygon];  
areaPolygon[vertices_] := Block[{edges},  
  edges = Partition[vertices, 2, 1, 1];  
  0.5 Abs@Total[(#[[1, 1]] * #[[2, 2]]) - (#[[2, 1]] * #[[1, 2]]) & /@ edges]  
];
```

```
In[11]:= Clear[perimeterOfPolygon];  
perimeterOfPolygon[cells_ /; Head[cells] === Association] :=  
  (Perimeter@*Polygon) /@ cells;
```

```
In[13]:= Clear[perimeterPolygon];  
perimeterPolygon[vertices_] := Block[{edges},  
  edges = Partition[vertices, 2, 1, 1];  
  Total[Apply[EuclideanDistance] /@ edges]  
];
```

```
In[15]:= Clear[centroidPolygon];
centroidPolygon[vertices_] := Mean@vertices
```

```
In[17]:= Clear[polyCounterClockwiseQ];
polyCounterClockwiseQ[poly_] := Block[{area = 0, j, vertLength = Length[poly]},
  Do[
    j = Mod[i, vertLength] + 1;
    area += poly[[i, 1]] * poly[[j, 2]];
    area -= poly[[j, 1]] * poly[[i, 2]],
    {i, vertLength}];
  (area/2) > 0
]
```

```
In[19]:= Clear[sortCC];
sortCC[polyinds_, indTopts_, ptsToInds_] := Block[{cent, poly},
  poly = Lookup[indTopts, polyinds];
  Lookup[ptsToInds,
    DeleteDuplicates@
      Flatten[MeshPrimitives[ConvexHullMesh[poly], 1] /. Line -> Sequence, 1]
  ];
```

```
In[21]:= Clear[sortPointsCC];
sortPointsCC[polyinds_, indTopts_, ptsToInds_] := Block[{cent, ordering, polyPoints},
  polyPoints = Lookup[indTopts, polyinds];
  cent = Mean@polyPoints;
  ordering = Ordering[ArcTan[#[[1]], #[[2]]] &@ (# - cent) & /@ polyPoints];
  Lookup[ptsToInds, Part[polyPoints, ordering]]
]
```

mesh restructuring operations

T1 transition (neighbour swapping)

```
In[23]:= (* T1 transition: neighbour switching *)
```

```
In[24]:= Clear@edgesforT1;
edgesforT1[edges_, indTopts_, threshLength_ : 0.0015] := Block[{edges, dist},
  edges = Lookup[indTopts, #] & /@ edges_;
  dist = EuclideanDistance @@ # & /@ edges;
  {Pick[edges, Thread[dist ≤ threshLength], True],
   Pick[edges_, Thread[dist ≤ threshLength], True]}
];
```

```
Clear@T1transitionFn;
T1transitionFn[edges_, indToptsAssoc_, vertexToCellG_, cellToVertexG_, dSep : 0.01] :=
```

```

Block[{findEdges, edgeind, connectedcellKeys, edge,
  newpts, cellvertIndices, cellvertices, pos, cellpolys, memF, keyscellP,
  selcellKeys, ptToCell, newptsindices, indToPts = indToPtsAssoc, ptsToInds,
  PtIndToCell, keysToMap, cellindicesAssoc, f1, otherkeys, f2, polysharingEdge,
  bag = CreateDataStructure["DynamicArray"], vertToCellG = vertexToCellG,
  cellToVertG = cellToVertexG, testpts, edgechanged},
{edgechanged, findEdges} = edgesforT1[edges, indToPts];
(* finding all possible edges for T1 transition *)
If[findEdges != {},
Scan[
  (edgeind = #;
  If[ContainsAll[Keys[indToPts], edgeind],
    (* should be an edge not
    connected to an edge that has already undergone a T1 *)
    connectedcellKeys = DeleteDuplicates[Flatten@Lookup[vertToCellG, edgeind]];
    cellvertIndices = Lookup[cellToVertG, connectedcellKeys];
    edge = Lookup[indToPts, edgeind];
    If[Length[connectedcellKeys] == 1,
      (*edge that is exposed to the void to be merged as a single vertex*)
      newpts = Mean[edge];
      newptsindices = Max[Keys@indToPts] + 1;
      KeyDropFrom[indToPts, edgeind];
      AppendTo[indToPts, newptsindices -> newpts];
      bag["Append", edgeind];
      ptsToInds = AssociationMap[Reverse, indToPts];
      cellToVertG = MapAt[
        DeleteDuplicates[# /. (Alternatives@@ edgeind) -> newptsindices] &,
        cellToVertG, Key[connectedcellKeys /. {z_Integer} -> z]
      ],
      (*else proceed with T1 transition*)
      newpts = With[{midPt = Mean[edge]},
        midPt + dSep Normalize[(# - midPt)] & /@
          Flatten[RotationTransform[- $\frac{\pi}{2}$ , midPt] /@ {edge}, 1]
      ];
      testpts = With[{midPt = Mean[edge]},
        midPt + 0.000001 Normalize[(# - midPt)] & /@ newpts
      ];
      pos = Position[cellvertIndices, {OrderlessPatternSequence[
        ___, First@edgeind, ___, Last@edgeind, ___]}, {1}];
      polysharingEdge = Extract[cellvertIndices, pos];
      (* the edge should not be part of any  $\Delta$  *)
      If[(AllTrue[polysharingEdge, Length[#] != 3] &)&
        ContainsNone[edgeind, Union@* Flatten@* Normal@bag],
        cellvertices = Map[Lookup[indToPts, #] &, cellvertIndices];
        cellpolys = Polygon /@ cellvertices;
        memF = Function[x, RegionMember@x, Listable][Extract[cellpolys, pos]];
        keyscellP = Extract[connectedcellKeys, pos];
        selcellKeys = Thread[keyscellP -> memF];

```

```

ptToCell = Quiet[#, First @@ Select[selcellKeys, Function[x,
    Last[x][#]]] & /@ testpts /. HoldPattern[_ -> First[]] -> Nothing];
(* pt to cell *)
ptToCell = ptToCell /. Thread[testpts -> newpts];
newptsindices = Range[#+1, #+2] &[Max[Keys@indToPts]];
KeyDropFrom[indToPts, edgeind];
AppendTo[indToPts, Thread[newptsindices -> newpts]];
ptsToInds = AssociationMap[Reverse, indToPts];
bag["Append", edgeind];
PtIndToCell = MapAt[ptsToInds, ptToCell, {All, 1}] /. Rule -> List;
(*index to cell*)
keysToMap = MapAt[Key, PtIndToCell, {All, 2}];
cellindicesAssoc = AssociationThread[connectedcellKeys, cellvertIndices];

f1 = Fold[MapAt[Function[x, DeleteDuplicates[x /. Thread[
    edgeind -> #2[[1]] ]]], #1, #2[[2]]] &, cellindicesAssoc, keysToMap];
otherkeys = List@*Key /@ Complement[connectedcellKeys, keysToMap];
f2 = MapAt[(# /. (Alternatives @@ edgeind) -> Splice[newptsindices] //
    sortPointsCC[#, indToPts, ptsToInds] &) &, f1, otherkeys];
AppendTo[cellToVertG, f2];
];
];
vertToCellG = GroupBy[
    Flatten[(Reverse[#, 2] &) @* Thread /@ Normal@cellToVertG], First -> Last];
]) &, findEdges]
];
{edgechanged, indToPts, cellToVertG, vertToCellG}
];

```

T2 transition

In[28]:= (* T2 transition: removal of cell *)

```

In[29]:= Clear@cellsforT2;
cellsforT2[areaAssoc_, cellVertexG_, thresh_ : 10^-5] := Block[{keys, ls, inds},
    keys = Keys@Select[areaAssoc, # < thresh &];
    ls = Lookup[cellVertexG, keys];
    inds = Flatten@Position[ls, x_ /; (3 ≤ Length[x] ≤ 6), {1}];
    (* 3 ≤ cell edges ≤ 6 *)
    If[inds ≠ {}, {keys[[inds]], ls[[inds]]}, {{}}, {}]] (*cell inds, vertices*)
];

```

In[31]:=

```

Clear@T2TransitionFn;
T2TransitionFn[{cellsToRemove_, vertindsRemove_}, indTopts_, cellToVertexG_,
  areaPolygonAssoc_, periPolygonAssoc_] := Block[{newVertices, maxkey, newindices,
  newentries, indTopts = indTopts, ruleDisp, removeentries,
  CVG = cellToVertexG, notaCell, VertCellGrouping},
  newVertices = Mean@Lookup[indTopts, #] & /@ vertindsRemove;
  maxkey = Max@*Keys@indTopts;
  newindices = Range[maxkey + 1, maxkey + Length[newVertices]];
  newentries = Thread[newindices → newVertices];
  KeyDropFrom[indTopts, Union@Flatten[vertindsRemove]];
  AppendTo[indTopts, newentries];
  ruleDisp =
    Dispatch@Flatten[MapThread[Thread[#1 → #2] &, {vertindsRemove, newindices}]];
  removeentries = Union@Flatten@cellsToRemove;
  KeyDropFrom[CVG, removeentries];
  CVG = DeleteDuplicates /@ Replace[CVG, ruleDisp, {2}];
  notaCell = Keys@Select[Length /@ CVG, # < 3 &];
  KeyDropFrom[CVG, notaCell];
  VertCellGrouping =
    GroupBy[Flatten[(Reverse[#, 2] &)@*Thread /@ Normal@CVG], First → Last];
  {indTopts, CVG, VertCellGrouping, KeyDrop[areaPolygonAssoc,
    removeentries ~ Join ~ notaCell],
    KeyDrop[periPolygonAssoc, removeentries ~ Join ~ notaCell]}
] /; vertindsRemove ≠ {};

```

cell division

In[33]:= (* probability of division based on the cell area *)

In[34]:=

```

Clear[selectDivCells];
selectDivCells[areaPolygon_, areathresh_ : 2.2, thresh_ : 0.0025] :=
  Block[{candidates, pos},
    candidates = Normal@Select[areaPolygon / Mean[areaPolygon], # > areathresh &];
    pos = Position[0.1 RandomReal[1, Length@candidates], x_ /; x < thresh];
    Keys@Extract[candidates, pos]
  ];

```

In[36]:= (* division more random *)

In[37]:=

```

Clear[pickcellsDiv];
pickcellsDiv[cellToVertG_, areaAssoc_] := Block[{pickcells, selcells, pos},
  pickcells = Keys@Select[Pick[areaAssoc,
    Thread[RandomReal[{0, 1}, Length[areaAssoc]] < 0.001, True], # > 0.005 &];
  pos = Position[Lookup[cellToVertG, pickcells], x_ /; Length[x] > 3, {1}];
  Extract[pickcells, pos]
];

```

In[39]:=

```

Clear[cellDivision];
cellDivision[polygonind_, indToPoints_, areaAssoc_, perimAssoc_, cellToVertG_] :=

```

```

Block[{x, y, num, matrix, xx, xy, yy, eigvals, eigVecs, maxeigpos, cent, edges,
  edgesL, intersects, intersectionPts, posIntersections, repPart,  $\alpha$ ,  $\beta$ ,
  polygonPts, newkeys = Range[# + 1, # + 2] &[Max@Keys[indToPoints]], newPtToInds,
  indtoPtAssoc = indToPoints, ptToIndAssoc, edgeinds, contour, poly1, poly2, res, seq,
  newcells = Range[# + 1, # + 2] &[Max@Keys[areaAssoc]],
  CVG = cellToVertG, addcellsRule, polygonPtsInds, VCG},
VCG = GroupBy[Flatten[(Reverse[#, 2] &)*Thread/@Normal@CVG], First → Last];
polygonPtsInds = CVG[polygonind];
num = Length@polygonPtsInds;
ptToIndAssoc = AssociationMap[Reverse, indToPoints];
polygonPts = Lookup[indToPoints, polygonPtsInds];
Evaluate[Table[{xi, yi}, {i, num + 1}]] = Append[polygonPts, First@polygonPts];


$$I_{xx} = \left(\frac{1}{12}\right) \sum_{i=1}^{num} (x_i y_{i+1} - x_{i+1} y_i) (y_i^2 + y_i y_{i+1} + y_{i+1}^2);$$



$$I_{yy} = \left(\frac{1}{12}\right) \sum_{i=1}^{num} (x_i y_{i+1} - x_{i+1} y_i) (x_i^2 + x_i x_{i+1} + x_{i+1}^2);$$



$$I_{xy} = \left(\frac{1}{24}\right) \sum_{i=1}^{num} (x_i y_{i+1} - x_{i+1} y_i) (x_i y_{i+1} + 2 x_i y_i + 2 x_{i+1} y_{i+1} + x_{i+1} y_i);$$


Table[
  {Unevaluated[Subscript[x, j]] = ., Unevaluated[Subscript[y, j]] = .}, {j, num + 1}];
matrix =  $\begin{pmatrix} I_{xx} & -I_{xy} \\ -I_{xy} & I_{yy} \end{pmatrix}$ ;
{eigvals, eigVecs} = Eigensystem[matrix];
maxeigpos = Position[eigvals, Max@eigvals];
{edges, edgeinds} = Partition[#, 2, 1, 1] &/@ {polygonPts, polygonPtsInds};
edgesL = Line /@ edges;
cent = centroidPolygon[polygonPts];
intersects = RegionIntersection[
  InfiniteLine[{cent, cent + Extract[eigVecs, maxeigpos][[1]]}], #] &/@ edgesL;
intersectionPts = Cases[intersects, {(_Real | _Integer) ..}, {3}];
newPtToInds = Thread[intersectionPts → newkeys];
posIntersections = Flatten@Position[intersects, _Point, {1}];
MapThread[
  (res = Complement[Intersection@@Lookup[VCG, #2], {polygonind}];
  If[res ≠ {},
    seq = Partition[CVG[First@res], 2, 1, 1];
    AppendTo[CVG,
      First@res → DeleteDuplicates@
        Flatten@SequenceSplit[seq, {x___, p : {OrderlessPatternSequence[
          #2[[1]], #2[[-1]]}], y___} → {x, Insert[p, #1, 2], y}}
    ];
  ] &, {newkeys, edgeinds[[posIntersections]]}];

repPart =
  Thread[{Thread[{ReverseSort@posIntersections, 2}], Reverse[intersectionPts]}];
{ $\alpha$ ,  $\beta$ } = intersectionPts;
AppendTo[ptToIndAssoc, newPtToInds];
AppendTo[indtoPtAssoc, Reverse[newPtToInds, 2]];
contour =

```

```

DeleteDuplicates@Flatten[Fold[Insert[#1, #2[[2]], #2[[1]]] &, edges, repPart], 1];
poly1 = Join @@ SequenceCases[contour, {___,  $\alpha$ } | { $\beta$ , ___}];
poly2 = Join @@ SequenceCases[contour, { $\alpha$ , __,  $\beta$ };
KeyDropFrom[CVG, polygonind];
addcellsRule = Thread[newcells  $\rightarrow$  {poly1, poly2}];
AppendTo[CVG, addcellsRule /. ptToIndAssoc];
{indtoPtAssoc, CVG, Append[KeyDrop[areaAssoc, polygonind],
  MapAt[Area@*Polygon, addcellsRule, {All, 2}]]},
Append[KeyDrop[perimAssoc, polygonind],
  MapAt[Perimeter@*Polygon, addcellsRule, {All, 2}]]}
];

```

force computation

In[41]:= $ka = 1$; $A0 = 1$; $\gamma = 0.04 * ka * A0$; $\delta t = 0.02$; $P0 = 0$; $\kappa = 0.025$;

In[42]:=

```

FAreaElasticity[indTopts_, vertexToCellG_, cellToVertexG_, areaPolygonAssoc_] :=
Block[{cellinds, temp, vertKeys = Keys[indTopts],
  vertLs, vertex, gc, gcc, diffVec, grad, coeff},
First@*Last@Reap@Do[
  cellinds = Lookup[vertexToCellG, i];
  temp = {0, 0};
  vertex = indTopts[i];
  Do[
    vertLs = Lookup[indTopts, Lookup[cellToVertexG, j]];
    gcc = getCounterClockwise[vertex, vertLs];
    gc = getClockwise[vertex, vertLs];
    diffVec = gcc - gc;
    grad =  $0.5 * \{\{0, 1\}, \{-1, 0\}\} \cdot \text{diffVec}$ ;
    coeff =  $ka * (\text{areaPolygonAssoc}[j] - A0)$ ;
    temp += grad * coeff, {j, cellinds}
  ];
  Sow@temp, {i, vertKeys}
]

```

In[43]:= $\text{MatrixForm}[\{\{0, 1\}, \{-1, 0\}\} \cdot (\{x_{i+1}, y_{i+1}\} - \{x_{i-1}, y_{i-1}\})]$

Out[43]//MatrixForm=

$$\begin{pmatrix} -y_{-1+i} + y_{1+i} \\ x_{-1+i} - x_{1+i} \end{pmatrix}$$

```

In[44]:= FPerimeterElasticity[indTopts_, vertexToCellG_, cellToVertexG_, periPolygonAssoc_] :=
Block[{cellinds, temp, vertKeys = Keys[indTopts], vertLs,
  vertex, gc, gcc, v1, v2, coeff, grad},
First@*Last@Reap@Do[
  cellinds = Lookup[vertexToCellG, i];
  temp = {0, 0};
  vertex = indTopts[i];
  Do[
    vertLs = Lookup[indTopts, Lookup[cellToVertexG, j]];
    gc = getClockwise[vertex, vertLs];
    gcc = getCounterClockwise[vertex, vertLs];
    v1 = Normalize[vertex - gc];
    v2 = Normalize[vertex - gcc];
    grad = v1 + v2;
    coeff =  $\gamma$  (periPolygonAssoc[j] - P0);
    temp += grad * coeff, {j, cellinds}];
  Sow@temp, {i, vertKeys}]
]

```

```

In[45]:= MatrixForm@Normalize[{xi, yi} - {xj, yj}]

```

```

Out[45]//MatrixForm=

```

$$\begin{pmatrix} \frac{x_i - x_j}{\sqrt{\text{Abs}[x_i - x_j]^2 + \text{Abs}[y_i - y_j]^2}} \\ \frac{y_i - y_j}{\sqrt{\text{Abs}[x_i - x_j]^2 + \text{Abs}[y_i - y_j]^2}} \end{pmatrix}$$

```

In[46]:= FLineTension[indTopts_, vertexToCellG_, cellToVertexG_] :=
Block[{cellinds, temp, vertKeys = Keys@indTopts, vertLs,
  vertex, gc, gcc, v1, v2},
First@*Last@Reap@Do[
  cellinds = Lookup[vertexToCellG, i];
  temp = {0, 0};
  vertex = indTopts[i];
  Do[
    vertLs = Lookup[indTopts, Lookup[cellToVertexG, j]];
    gc = getClockwise[vertex, vertLs];
    gcc = getCounterClockwise[vertex, vertLs];
    v1 = Normalize[vertex - gc];
    v2 = Normalize[vertex - gcc];
    temp +=  $\kappa$  v1 +  $\kappa$  v2, {j, cellinds}];
  Sow@temp, {i, vertKeys}]
]

```



```
In[47]:= FActiveContraction[indTopts_, vertexToCellG_, cellToVertexG_, areaPolygonAssoc_] :=
Block[{cellinds, temp, vertKeys = Keys@indTopts, vertLs,
  vertex, gc, gcc, diffVec, grad, coeff},
First@*Last@Reap@Do[
  cellinds = Lookup[vertexToCellG, i];
  temp = {0, 0};
  vertex = indTopts[i];
  Do[
    vertLs = Lookup[indTopts, Lookup[cellToVertexG, j]];
    gcc = getCounterClockwise[vertex, vertLs];
    gc = getClockwise[vertex, vertLs];
    diffVec = gcc - gc;
    grad = 0.5 * {{0, 1}, {-1, 0}}.diffVec;
    coeff = 0.1 ka * (areaPolygonAssoc[j]);
    temp += grad * coeff, {j, cellinds}];
  Sow@temp, {i, vertKeys}]
]
```

```
In[48]:= FT[indTopts_, vertexToCellG_,
  cellToVertexG_, areaPolygonAssoc_, periPolygonAssoc_] := - (
  FAreaElasticity[indTopts, vertexToCellG, cellToVertexG, areaPolygonAssoc] +
  FPerimeterElasticity[indTopts, vertexToCellG, cellToVertexG, periPolygonAssoc] +
  FLineTension[indTopts, vertexToCellG, cellToVertexG] +
  FActiveContraction[indTopts, vertexToCellG, cellToVertexG, areaPolygonAssoc]
);
```

create mesh and run simulation

```
In[49]:= SeedRandom[3];
mesh = VoronoiMesh[RandomReal[1, {200, 2}], {{0, 1}, {0, 1}}, ImageSize → Medium];

In[51]:= pts = MeshPrimitives[mesh, 0] /. Point → Sequence;

In[52]:= cornerpts = pts[[-4 ;;]];
pts = pts[[1 ;; -5]];

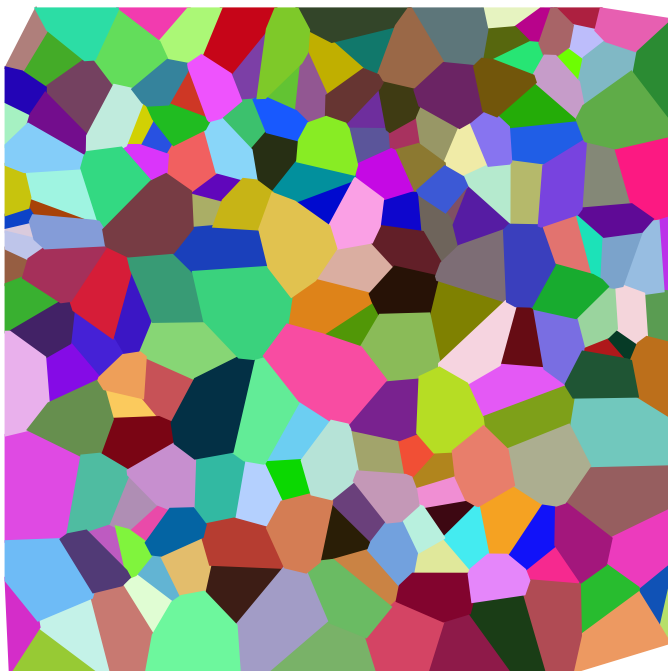
In[54]:= $ptsToInd = ptsToInd = AssociationThread[pts → Range@Length@pts];
$indTopts = indTopts = AssociationMap[Reverse][ptsToInd];

In[56]:= cellmeshprim = MeshPrimitives[mesh, 2];
cells = (MeshPrimitives[#, 0] & /@ cellmeshprim) /. Point → Sequence /.
  Thread[cornerpts → Nothing];

In[58]:= $cellToVertexG =
  cellToVertexG = AssociationThread[Range[Length@cells] → Map[ptsToInd, cells, {2}]];
$vertexToCell = vertexToCell =
  GroupBy[Flatten[(Reverse[#, 2] &) @* Thread /@ Normal@cellToVertexG], First → Last];
```

```
In[60]:= Graphics[Map[{RandomColor[], Polygon@Lookup[indTopts, #]} &, Values@cellToVertexG],
  ImageSize -> Medium]
```

Out[60]=

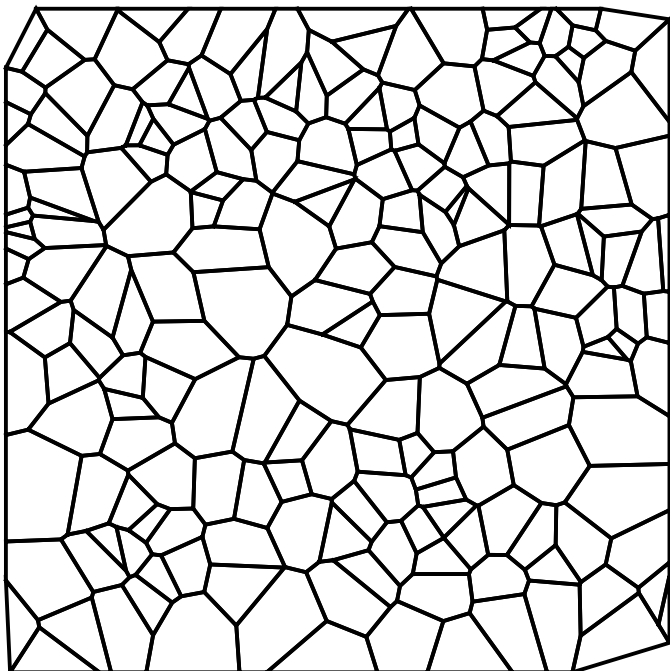


```
In[61]:= (*edges=Flatten[Map[Partition[#,2,1,1]&,Values[cellToVertexG]],1];*)
$cellToPts = cellToPts = Lookup[indTopts, #] & /@ cellToVertexG;
$periPolygonAssoc = periPolygonAssoc = perimeterPolygon /@ cellToPts;
$areaPolygonAssoc = areaPolygonAssoc = areaPolygon /@ cellToPts;
```

```
In[64]:= Clear[plt, indTopts, ptsToInd, vertexToCell,
  cellToVertexG, periPolygonAssoc, areaPolygonAssoc, cellToPts, edges];
```

```
In[65]:= pltOriginal = Graphics[{Black, Thick,
  Values@Map[Line[Join[##, {First@#}]] &@Lookup[$indTopts, #] &, $cellToVertexG]}]
```

Out[65]=



```
In[66]:= t =  $\delta t$ ;
indTopts = $indTopts;
ptsToInd = $ptsToInd;
vertexToCell = $vertexToCell;
cellToVertexG = $cellToVertexG;
periPolygonAssoc = $periPolygonAssoc;
areaPolygonAssoc = $areaPolygonAssoc;
cellToPts = $cellToPts;
edges = DeleteDuplicatesBy[
  Flatten[Map[Partition[#, 2, 1, 1] &, Values@$cellToVertexG], 1], Sort];
```

In[75]:=

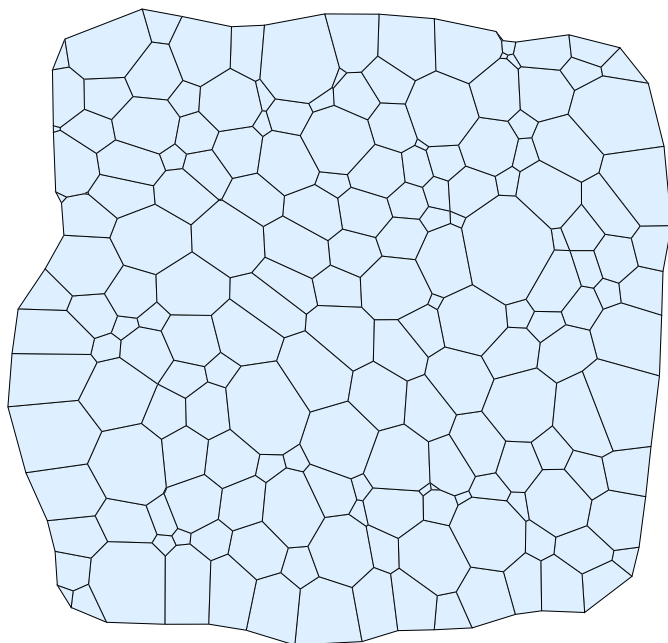
```

Module[{cellsToRemove, vertsToRemove, edgechanged, polydiv},
  saveres = First@Last@Reap@Monitor[
    While[t ≤ 100 δt,
      (* T2 transitions *)
      {cellsToRemove, vertsToRemove} = cellsforT2[areaPolygonAssoc, cellToVertexG];
      If[cellsToRemove ≠ {},
        {indTopts, cellToVertexG, vertexToCell, areaPolygonAssoc, periPolygonAssoc} =
          T2TransitionFn[{cellsToRemove, vertsToRemove}, indTopts,
            cellToVertexG, areaPolygonAssoc, periPolygonAssoc]
      ];
      (* T1 transitions *)
      edges = DeleteDuplicatesBy[
        Flatten[Map[Partition[#, 2, 1, 1] &, Values[cellToVertexG]], 1], Sort];
      {edgechanged, indTopts, cellToVertexG, vertexToCell} =
        T1transitionFn[edges, indTopts, vertexToCell, cellToVertexG];
      cellToPts = Lookup[indTopts, #] & /@ cellToVertexG;
      areaPolygonAssoc = areaPolygon /@ cellToPts;
      periPolygonAssoc = perimeterPolygon /@ cellToPts;
      (* Divisions *)
      polydiv = selectDivCells[areaPolygonAssoc];
      (*polydiv=pickcellsDiv[cellToVertexG,areaPolygonAssoc];*)
      If[polydiv ≠ {},
        Scan[
          ({indTopts, cellToVertexG, areaPolygonAssoc, periPolygonAssoc} = cellDivision[
            #, indTopts, areaPolygonAssoc, periPolygonAssoc, cellToVertexG]) &,
            polydiv];
        vertexToCell = GroupBy[
          Flatten[(Reverse[#, 2] &) @* Thread /@ Normal@cellToVertexG], First → Last];
      ];
      (* update positions *)
      indTopts = AssociationThread[Keys[indTopts] → (Values[indTopts] + FT[indTopts,
        vertexToCell, cellToVertexG, areaPolygonAssoc, periPolygonAssoc] δt)];
      cellToPts = Lookup[indTopts, #] & /@ cellToVertexG;
      areaPolygonAssoc = areaPolygon /@ cellToPts;
      periPolygonAssoc = perimeterPolygon /@ cellToPts;
      (*plt=Graphics[{ColorData[1][1],Thick,Values@Map[Line[Join[##,{First@#}]]&@
        Lookup[indTopts,#]&,cellToVertexG]},ImageSize→Medium];*)
      plt = Graphics[{FaceForm[LightBlue], EdgeForm[{Black}], Values[
        Polygon@Lookup[indTopts, #] & /@ cellToVertexG]}, ImageSize → Medium];
      Sow[plt];
      t += δt;
    ], plt
  ];
];

```

In[76]:= **plt**

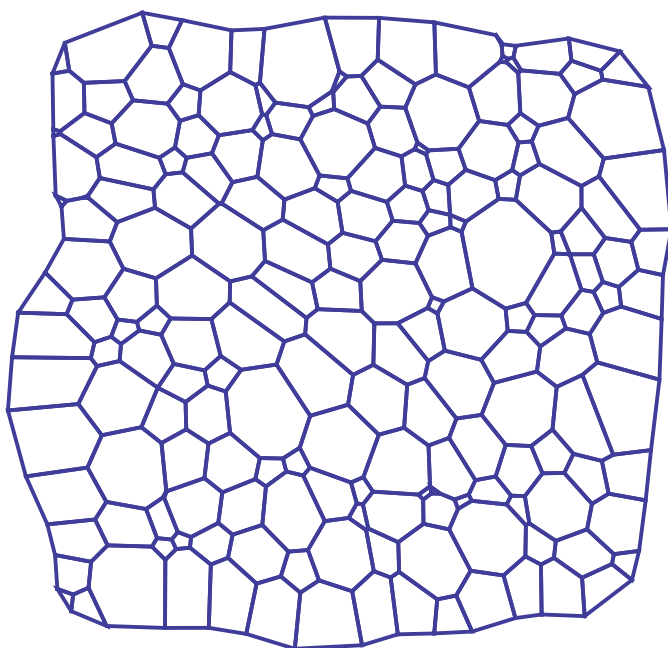
Out[76]=

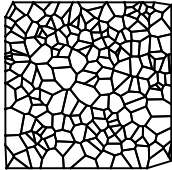



In[77]:= (***Export**["C:\\Users\\aliha\\Desktop\\result.gif",saveres,AnimationRepetitions→ ∞]*)

In[80]:= **Graphics**[{■, **Thick**,
Values@Map[**Line**[**Join**[##, {**First**@#}]] &@**Lookup**[indTopts, #] &, **cellToVertexG**]],
ImageSize → **Medium**]

Out[80]=



In[81]:=  Show[/. Black → Red, Graphics[{, Thick, Values@Map[Line[Join[##, {First@#}]] &@Lookup[indTops, #] &, cellToVertexG]]]

