```
FindNeighbours[IList_, JList_, Start_, nextI_, nextJ_, ConvexHull_] :=
  Module[{SortedPointsKeys = <||>, SortedPoints = {}, proportionJ,
  |Modul
    proportionI, jtemp, itemp, PotNextJDir, distanceNextPotPointJ, PotNextIDir ,
    distanceNextPotPointI, NeighbourNumberJ, NeighbourNumberI, distanceJ,
    distanceI, NextJDir, NextIDir, StartPropJForFirstCompleteGridJ},

   Print["
   |gib aus

Detect I and J rows of
        |imaginäre Einheit I
       Points_____________________________________________________________

"];
   Clear[CheckPoints];
   |lösche
   CheckPoints = {};
   StartPoint = Start;
   StartPointI = Start;
   StartPointJ = Start;
   NextPointI = nextI;
   NextPointJ = nextJ;

   Print["Start = ", Start];
   |gib aus
   Print[" nextI = ", nextI];
   |gib aus
   Print["nextJ = ", nextJ];
   |gib aus
   IDirStart = {nextI[CoordI] - Start[CoordI], nextI[CoordJ] - Start[CoordJ]};
   JDirStart = {nextJ[CoordI] - Start[CoordI], nextJ[CoordJ] - Start[CoordJ]};

   Print["IDir = ", IDirStart];
   |gib aus
   Print["JDir = ", JDirStart];
   |gib aus
   StartProportionI = nextI[CoordJ] - Start[CoordJ];
   StartProportionJ = nextJ[CoordI] - Start[CoordI];

   StartDistanceI = Sqrt[Abs[IDirStart[[1]]^2 + IDirStart[[2]]^2]];
                    |Qua·· |Absolutwert
   StartDistanceJ = Sqrt[Abs[JDirStart[[1]]^2 + JDirStart[[2]]^2]];
                    |Qua·· |Absolutwert

   AssociateTo[StartPoint, {NeighbourJ → 1, NeighbourI → 1}];
   |assoziiere mit
   AssociateTo[NextPointI , { NeighbourJ → 1, NeighbourI → 2}];
   |assoziiere mit
   AssociateTo[NextPointJ, { NeighbourJ → 2, NeighbourI → 1}];
   |assoziiere mit
   AppendTo[SortedPoints, {StartPoint, NextPointI, NextPointJ}];
   |hänge an bei
```

```
AppendTo[CheckPoints, Start];
hänge an bei
AppendTo[CheckPoints, nextJ];
hänge an bei
AppendTo[CheckPoints, nextI];
hänge an bei

NeighbourNumberJ = 2;
NeighbourNumberI = 2;
aI = 2;
aJ = 2;
NextJDir = JDirStart;
distanceJ = StartDistanceJ;

proportionJ = StartProportionJ;
StartPropJForFirstCompleteGridJ = StartProportionJ;

For[pp = 1, pp ≤ Length[JList] * 2, pp++,
For–Schleife        Länge
 For[tt = 1, tt ≤ Length[JList], tt++,
 For–Schleife        Länge
   If[JList[[tt]][CoordJ] ≠ StartPointJ[CoordJ] &&
   wenn
      JList[[tt]][CoordJ] ≥ NextPointJ[CoordJ] && JList[[tt]][CoordI] ≥
       NextPointJ[CoordI] || JList[[tt]][CoordI] ≤  NextPointJ[CoordI],

    PotNextJDir = {JList[[tt]][CoordJ] - NextPointJ[CoordJ],
      JList[[tt]][CoordI] - NextPointJ[CoordI]};
    distanceNextPotPointJ = Sqrt[Abs[PotNextJDir[[1]]^2 + PotNextJDir[[2]]^2]];
                            Qua·· Absolutwert
   ];

   If[JList[[tt]][CoordJ] ≠ NextPointJ[CoordJ] &&
   wenn
     JList[[tt]][CoordJ] ≠ StartPointJ[CoordJ] &&
     (JList[[tt]][CoordI]) ≤ NextPointJ[CoordI] + proportionJ + 0.3 &&
     (JList[[tt]][CoordI]) ≥ NextPointJ[CoordI] + proportionJ - 0.3 &&
     distanceNextPotPointJ ≤ (distanceJ + 0.5) &&
     JList[[tt]][CoordJ] ≥ NextPointJ[CoordJ] + (distanceJ / 2),

    StartPointJ = NextPointJ;
    NextPointJ = JList[[tt]];

    Print["Test J = ", JList[[tt]]];
    gib aus

    NextJDir = {NextPointJ[CoordJ] - StartPointJ [CoordJ],
      NextPointJ[CoordI] - StartPointJ [CoordI]};
    distanceJ = Sqrt[Abs[NextJDir[[1]]^2 + NextJDir[[2]]^2]];
                Qua·· Absolutwert
    proportionJ = NextPointJ[CoordI] - StartPointJ[CoordI];
    aJ = aJ + 1;
    AppendTo[CheckPoints, JList[[tt]]];
    hänge an bei
    hänge an bei
    AssociateTo[SortedPointsKeys,
    assoziiere mit
     { JList[[tt]], NeighbourJ  → aJ, NeighbourI → 1}];
    AppendTo[SortedPoints, SortedPointsKeys];
    hänge an bei
    LastJPointsCellI = JList[[tt]][CellI];
    CheckLastPointJ = JList[[tt]];
    AssociateTo[CheckLastPointJ, {NeighbourJ  → aJ, NeighbourI → 1}];
    assoziiere mit
   ];
  ];
];
(*];*)

AppendTo[SortedPoints, SaftyListJ[Start, CheckLastPointJ,
hänge an bei
  proportionJ, LastJPointsCellI, ConvexHull, distanceJ, NextJDir]];

AppendTo[SortedPoints, CompleteJGrid[ nextI, ConvexHull, StartDistanceJ,
hänge an bei
  StartPropJForFirstCompleteGridJ, Start , NeighbourNumberJ, aI]];

proportionI = StartProportionI;
distanceI  = StartDistanceI;
NextIDir = IDirStart;

For[pp = 1, pp ≤ Length[IList] * 2, pp++,
For–Schleife      Länge
 For[tt  = 1, tt ≤ Length[IList], tt++,
 For–Schleife         Länge
   If[IList[[tt]][CoordI] ≠ StartPointI[CoordI] &&
   wenn
      IList[[tt]][CoordJ] ≥ NextPointI[CoordJ] || IList[[tt]][CoordJ] ≤
       NextPointI[CoordJ] && IList[[tt]][CoordI] ≥ NextPointI[CoordI],

    PotNextIDir  = {IList[[tt]][CoordJ] - NextPointI[CoordJ],
      IList[[tt]][CoordI] - NextPointI[CoordI]};
    distanceNextPotPointI =  Sqrt[Abs[PotNextIDir[[1]]^2 + PotNextIDir[[2]]^2]];
                             Qua⋯ Absolutwert
   ];

   If[IList[[tt]][CoordI] ≠ NextPointI [CoordI] &&
   wenn
     IList[[tt]][CoordI] ≠ StartPointI[CoordI]  &&
     (IList[[tt]][CoordJ]) ≤ NextPointI[CoordJ] + proportionI + 0.3 &&
     (IList[[tt]][CoordJ]) ≥  NextPointI[CoordJ] + proportionI  - 0.3 &&
     distanceNextPotPointI  ≤ distanceI + 0.5 &&
     IList[[tt]][CoordI] ≥ NextPointI[CoordI] + (distanceI / 3),

    Print["Test I = ", IList[[tt]]];
    gib aus        imaginäre Einheit I
    StartPointI  = NextPointI;
    NextPointI = IList[[tt]];
    NextIDir = {NextPointI[CoordJ] - StartPointI[CoordJ],
      NextPointI[CoordI] - StartPointI[CoordI]};
    distanceI = Sqrt[Abs[NextIDir[[1]]^2 + NextIDir[[2]]^2]];
                |Qua·· |Absolutwert

    proportionI = NextPointI[CoordJ] - StartPointI[CoordJ];
    (*propJForGrind =Abs[NextPointI[CoordJ]-StartPointI[CoordJ]];*)
                     |Absolutwert
    propJForGrind = Abs[StartProportionJ];
                    |Absolutwert
    aI = aI + 1;

    AppendTo[CheckPoints, IList[[tt]]];
    |hänge an bei
    AssociateTo[SortedPointsKeys,
    |assoziiere mit
     { IList[[tt]], NeighbourJ → 1, NeighbourI → aI}];
    AppendTo[SortedPoints, SortedPointsKeys];
    |hänge an bei

    LastIPointsCellJ = IList[[tt]][CellJ];
    CheckLastPointI = IList[[tt]];
    AssociateTo[CheckLastPointI, {NeighbourJ → 1, NeighbourI → aI}];
    |assoziiere mit

    NeighbourNumberJ = NeighbourNumberJ;
    AppendTo[SortedPoints, SaftyListI[Start, CheckLastPointI,
    |hänge an bei
      proportionI, LastIPointsCellJ, ConvexHull, distanceI, NextIDir]];

    AppendTo[SortedPoints, CompleteJGrid[IList[[tt]],
    |hänge an bei
      ConvexHull, distanceJ, propJForGrind, Start, NeighbourNumberJ , aI]];


   ];
  ];
 ];



 Print["Länge ConvexHull = ", Length[ConvexHull]];
 |gib aus                    |Länge
 Print["Länge SortedPoints = ", Length[Flatten[SortedPoints]]];
 |gib aus                      |Länge  |ebne ein
 If[Length[Flatten[SortedPoints]] ≠ Length[ConvexHull] && splits ≤ 8,
 |··· |Länge  |ebne ein             |Länge
  splits = splits + 1;
  Clear[CheckPoints];
  |lösche
  CheckPoints = {};
  Print["
  |gib aus
Another
    round_____________________________________________________________
```

```
"];
     FindMinMax[PointList],
     Print["
     gib aus
End and
beende Kontext
         Result________________________________________________________________


"];
     Print["SortedPoints = ", Flatten[SortedPoints]];
     gib aus                            ebne ein
     DrawGraph[SortedPoints];
     splits = 2;
    ];
   ];


SaftyListJ[Start_, CheckLastPointJ_, proportionJ_,
    LastJPointsCellI_, ConvexHull_, distanceJ_, NextJDir_] :=
   Module[{SaftyList = {}, SaftyKeys = <||>, SaftyKeysList = {},
   Modul
     propJ, lastDirJ, lastdistanceJ , PotNextJDir, tempj,
     distanceNextPotPointJ , nextNeighbourNumber, StartAtThisJPoint},

    If[Length[SaftyKeysList] ≠ 0,
    ⋯  Länge
     Clear[SaftyKeysList];
     lösche
    ];
    StartAtThisJPoint = CheckLastPointJ;
    nextNeighbourNumber = CheckLastPointJ[NeighbourJ] + 1;
    propJ = proportionJ;
    lastDirJ = NextJDir;
    lastdistanceJ = distanceJ;

    For[ii = 1, ii ≤ Length[ConvexHull], ii++,
    For-Schleife       Länge

     If[ConvexHull[[ii]][CellI] == CheckLastPointJ[CellI] + 1 ||
     wenn
        ConvexHull[[ii]][CellI] == CheckLastPointJ[CellI] - 1 ||
        ConvexHull[[ii]][CellI] == CheckLastPointJ[CellI] &&
         ConvexHull[[ii]][CellJ] ≥ CheckLastPointJ[CellJ],

       AppendTo[SaftyList, ConvexHull[[ii]]];
       hänge an bei
      ];
    ];

    Print["SaftyList J = ", SaftyList];
    gib aus
gib aus
For[ll = 1, ll ≤  Length[SaftyList], ll++,
For–Schleife          Länge
 If[SaftyList[[ll]][CoordJ] ≠ Start[CoordJ] &&
 wenn
    SaftyList[[ll]][CoordJ] ≥  StartAtThisJPoint[CoordJ] &&
    SaftyList[[ll]][CoordI] ≥   StartAtThisJPoint[CoordI] ||
   SaftyList[[ll]][CoordI] ≤   StartAtThisJPoint[CoordI] ,

  PotNextJDir  = {StartAtThisJPoint[CoordJ] - SaftyList[[ll]][CoordJ],
    StartAtThisJPoint[CoordI] - SaftyList[[ll]][CoordI]};
  distanceNextPotPointJ  = Sqrt[Abs[PotNextJDir[[1]]^2 + PotNextJDir[[2]]^2]];
                           Qua⋯ Absolutwert
 ];
 If[SaftyList[[ll]][CoordJ] ≠ Start[CoordJ] &&
 wenn
   SaftyList[[ll]][CoordJ] ≠ StartAtThisJPoint[CoordJ]  &&
   SaftyList[[ll]][CoordJ] ≥  StartAtThisJPoint[CoordJ] &&
   SaftyList[[ll]][CoordI] ≤ StartAtThisJPoint[CoordI] + propJ + 0.3 &&
   SaftyList[[ll]][CoordI] ≥  StartAtThisJPoint[CoordI] + propJ - 0.3 &&
   distanceNextPotPointJ  ≤  lastdistanceJ  + 0.5 &&
   SaftyList[[ll]][CoordJ] ≥  StartAtThisJPoint[CoordJ] + lastdistanceJ / 3
   (**(3/4)*)  && MemberQ[CheckPoints, SaftyList[[ll]]] == False,
                 enthalten?                               falsch
  Print["SaftyListPoint  = ", SaftyList[[ll]]];
  gib aus
  AppendTo[CheckPoints, SaftyList[[ll]]];
  hänge an bei
  AssociateTo[SaftyKeys, { SaftyList[[ll]],
  assoziiere mit
    NeighbourJ  → nextNeighbourNumber , NeighbourI → CheckLastPointJ[NeighbourI]}];
  AppendTo[SaftyKeysList, SaftyKeys];
  hänge an bei

  lastDirJ  = {StartAtThisJPoint [CoordJ] - SaftyList[[ll]]  [CoordJ],
    StartAtThisJPoint[CoordI] - SaftyList[[ll]][CoordI]};
  lastdistanceJ  = Sqrt[Abs[lastDirJ [[1]]^2 + lastDirJ [[2]]^2]];
                   Qua⋯ Absolutwert
  propJ = SaftyList[[ll]][CoordI] - StartAtThisJPoint[CoordI];

  StartAtThisJPoint = SaftyList[[ll]];
  Print["StartAtThisJPoint  = ", StartAtThisJPoint];
  gib aus
  nextNeighbourNumber = nextNeighbourNumber + 1;


  For[ii = 1, ii ≤  Length[ConvexHull], ii++,
  For–Schleife         Länge
   If[ConvexHull[[ii]][CellI] == StartAtThisJPoint[CellI] + 1 ||
   wenn
      ConvexHull[[ii]][CellI] == StartAtThisJPoint[CellI] - 1 ||
      ConvexHull[[ii]][CellI] == StartAtThisJPoint[CellI] &&
       ConvexHull[[ii]][CellJ] ≥ StartAtThisJPoint[CellJ],

     AppendTo[SaftyList, ConvexHull[[ii]]];
     hänge an bei
```

```
            hänge an bei
           ];
         ];
        ];
       ];

       Clear[SaftyList];
       lösche
       Return[SaftyKeysList];
       gib zurück
      ];


   SaftyListI[Start_, CheckLastPointI_, proportionI_,
       CheckCellJForI_, ConvexHull_, distanceI_, NextIDir_] :=
      Module[{SaftyList = {}, SaftyKeys = <||>, SaftyKeysList = {}, propI,
      Modul
        lastDirI, lastdistanceI , PotNextIDir, tempi, distanceNextPotPointI ,
        nextNeighbourNumber, StartAtThisIPoint},

       If[Length[SaftyKeysList] ≠ 0,
       ⋯  Länge
        Clear[SaftyKeysList];
        lösche
       ];
       StartAtThisIPoint = CheckLastPointI;
       nextNeighbourNumber = CheckLastPointI[NeighbourI] + 1;

       propI = proportionI;
       lastDirI = NextIDir;
       lastdistanceI = distanceI;

       For[kk = 1, kk ≤ Length[ConvexHull], kk++,
       For-Schleife      Länge

        If[ConvexHull[[kk]][CellJ] == CheckLastPointI[CellJ] + 1 ||
        wenn
            ConvexHull[[kk]][CellJ] == CheckLastPointI[CellJ] - 1 &&
             ConvexHull[[kk]][CellI] ≥ CheckLastPointI[CellI],

          AppendTo[SaftyList, ConvexHull[[kk]]];
          hänge an bei
         ];
       ];

       Print["SaftyList I = ", SaftyList];
       gib aus                  imaginäre Einheit I
       For[uu = 1, uu ≤ Length[SaftyList], uu++,
       For-Schleife      Länge
        (*Print["Start At This JPoint = ",StartAtThisJPoint];*)
          gib aus
        If[SaftyList[[uu]][CoordI] ≠ Start[CoordI] &&
        wenn
            SaftyList[[uu]][CoordI] ≥ StartAtThisIPoint[CoordI] &&
            SaftyList[[uu]][CoordJ] ≥  StartAtThisIPoint[CoordJ] ||
     SaftyList[[uu]][CoordJ] ≤  StartAtThisIPoint[CoordJ] ,

    PotNextIDir = {StartAtThisIPoint[CoordI] - SaftyList[[uu]][CoordI],
      StartAtThisIPoint[CoordJ] - SaftyList[[uu]][CoordJ]};
    distanceNextPotPointI = Sqrt[Abs[PotNextIDir[[1]]^2 + PotNextIDir[[2]]^2]];
                            ⌊Qua⋯⌊Absolutwert
   ];
   If[SaftyList[[uu]][CoordI] ≠ Start[CoordI] &&
  ⌊wenn
     SaftyList[[uu]][CoordI] ≠ StartAtThisIPoint[CoordI]  &&
     SaftyList[[uu]][CoordI] ≥ StartAtThisIPoint[CoordI] &&
     SaftyList[[uu]][CoordJ] ≤ StartAtThisIPoint[CoordJ] + propI + 0.3 &&
     SaftyList[[uu]][CoordJ] ≥  StartAtThisIPoint[CoordJ] + propI - 0.3 &&
     distanceNextPotPointI ≤ lastdistanceI + 0.5 &&
     SaftyList[[uu]][CoordI] ≥ StartAtThisIPoint[CoordI] + lastdistanceI / 3
     (**(3/4)*) && MemberQ[CheckPoints, SaftyList[[uu]]] == False,
                   ⌊enthalten?                               ⌊falsch
    Print["SaftyListPoint I = ", SaftyList[[uu]]];
    ⌊gib aus              ⌊imaginäre Einheit I
    AppendTo[CheckPoints, SaftyList[[uu]]];
    ⌊hänge an bei
    AssociateTo[SaftyKeys,
    ⌊assoziiere mit
     { SaftyList[[uu]], NeighbourJ → 1 , NeighbourI → nextNeighbourNumber}];
    AppendTo[SaftyKeysList, SaftyKeys];
    ⌊hänge an bei

    lastDirI = {StartAtThisIPoint [CoordI] - SaftyList[[uu]] [CoordI],
      StartAtThisIPoint[CoordJ] - SaftyList[[uu]][CoordJ]};
    lastdistanceI  = Sqrt[Abs[lastDirI [[1]]^2 + lastDirI [[2]]^2]];
                     ⌊Qua⋯⌊Absolutwert

    propI = SaftyList[[uu]][CoordJ] - StartAtThisIPoint[CoordJ];
    StartAtThisIPoint = SaftyList[[uu]];
     nextNeighbourNumber = nextNeighbourNumber + 1;

    AppendTo[SaftyKeysList, CompleteJGrid[StartAtThisIPoint,
    ⌊hänge an bei
      ConvexHull, lastdistanceI, propI, Start, 2 , nextNeighbourNumber - 1]];

    For[ii = 1, ii ≤ Length[ConvexHull], ii++,
    ⌊For-Schleife  ⌊Länge
     If[ConvexHull[[ii]][CellJ] == StartAtThisIPoint[CellJ] + 1 ||
     ⌊wenn
        ConvexHull[[ii]][CellJ] == StartAtThisIPoint[CellJ] - 1 ||
        ConvexHull[[ii]][CellJ] == StartAtThisIPoint[CellJ] &&
         ConvexHull[[ii]][CellJ] ≥ StartAtThisIPoint[CellJ],

       AppendTo[SaftyList, ConvexHull[[ii]]];
       ⌊hänge an bei
      ];
    ];
   ];
  ];
```

```
   Clear[SaftyList];
   lösche
   Return[SaftyKeysList];
   gib zurück
  ];

CompleteJGrid[StartPointI_, ConvexHull_, StartDistanceJ_, proportionJ_,
   Start_, NeighbourNumberJ_, aI_] := Module[{PossiblePointsListJ = {},
                                      Modul
    SortedPointsKeys = <||>, SaftyPossiblePointsListJ = {}, propJ, StartPointForJGrid ,
    distanceJ , NextNeighbourNumbrtJ, distanceNextPotGridPointJ , tempj,
    NextPointJDir, PotNextJDir , NextJDir, CheckPointJ, CheckCellForJ},

   propJ = proportionJ;
   StartPointForJGrid = StartPointI;
   distanceJ = StartDistanceJ;
   NextNeighbourNumbrtJ = NeighbourNumberJ;

   Print["proportionJ Complete Grid = ", proportionJ];
   gib aus                         Gitter

   For[aa = 1, aa ≤ Length[ConvexHull], aa++,
   For-Schleife   Länge
    If[ConvexHull[[aa]][CellI] == StartPointForJGrid[CellI] + 1 ||
    wenn
       ConvexHull[[aa]][CellI] == StartPointForJGrid[CellI] - 1 ||
       ConvexHull[[aa]][CellI] == StartPointForJGrid[CellI] && ConvexHull[[aa]][
          CellJ] ≤ splits && MemberQ[CheckPoints, ConvexHull[[aa]]] == False,
                             enthalten?                                 falsch
      AppendTo[PossiblePointsListJ, ConvexHull[[aa]]];
      hänge an bei
     ];
   ];

   Print["PossiblePointList = ", PossiblePointsListJ];
   gib aus

   For[pp = 1, pp ≤ Length[PossiblePointsListJ], pp++,
   For-Schleife   Länge

    If[PossiblePointsListJ[[pp]][CoordJ] ≠ StartPointForJGrid[CoordJ] &&
    wenn
       PossiblePointsListJ[[pp]][CoordJ] ≥ StartPointForJGrid[CoordJ] &&
       PossiblePointsListJ[[pp]][CoordI] ≤ StartPointForJGrid[CoordI] ||
      PossiblePointsListJ[[pp]][CoordI] ≥ StartPointForJGrid[CoordI],

     PotNextJDir = {PossiblePointsListJ[[pp]][CoordJ] - StartPointForJGrid[CoordJ],
       PossiblePointsListJ[[pp]][CoordI] - StartPointForJGrid[CoordI]};
     distanceNextPotGridPointJ = Sqrt[Abs[PotNextJDir[[1]]^2 + PotNextJDir[[2]]^2]];
                                 Qua··· Absolutwert
    ];
    If[PossiblePointsListJ[[pp]][CoordJ] ≠ StartPointForJGrid[CoordJ] &&
    wenn
      PossiblePointsListJ[[pp]][CoordJ] ≠ Start[CoordJ] &&
      PossiblePointsListJ[[pp]][CoordJ] ≥ StartPointForJGrid[CoordJ] &&
      PossiblePointsListJ[[pp]][CoordI] ≤  StartPointForJGrid[CoordI] + propJ + 0.3 &&
      PossiblePointsListJ[[pp]][CoordI] ≥
       StartPointForJGrid[CoordI] + propJ - (*1.0*) 3.0 &&
      PossiblePointsListJ[[pp]][CoordJ] ≥ StartPointForJGrid[CoordJ] + distanceJ / 3 &&
      distanceNextPotGridPointJ  ≤  (distanceJ + 0.9) &&
      MemberQ[CheckPoints, PossiblePointsListJ[[pp]]] == False,
      enthalten?                                        falsch
     Print["[PossiblePointsListJ[[pp]]]; = ", PossiblePointsListJ[[pp]]];
     gib aus
     NextJDir = { PossiblePointsListJ[[pp]][CoordJ] - StartPointForJGrid[CoordJ],
       PossiblePointsListJ[[pp]][CoordI] - StartPointForJGrid[CoordI]};
     distanceJ = Sqrt[Abs[NextJDir[[1]]^2 + NextJDir[[2]]^2]];
                 Qua·· Absolutwert

     propJ = PossiblePointsListJ[[pp]][CoordI] - StartPointForJGrid[CoordI];

     StartPointForJGrid = PossiblePointsListJ[[pp]];

     AppendTo[CheckPoints, PossiblePointsListJ[[pp]]];
     hänge an bei
     AssociateTo[SortedPointsKeys, {PossiblePointsListJ[[pp]],
     assoziiere mit
       NeighbourJ → NextNeighbourNumbrtJ, NeighbourI → aI}];
     AppendTo[SaftyPossiblePointsListJ, SortedPointsKeys];
     hänge an bei
     AssociateTo[StartPointForJGrid,
     assoziiere mit
      {NeighbourJ → NextNeighbourNumbrtJ, NeighbourI → aI}];
     NextNeighbourNumbrtJ = NextNeighbourNumbrtJ + 1;
    ];

   ];

   CheckPointJ = StartPointForJGrid;
   CheckCellForJ = StartPointForJGrid[CellJ];

   AppendTo[SaftyPossiblePointsListJ, SaftyListJ[Start, StartPointForJGrid,
   hänge an bei
     propJ, StartPointForJGrid[CellJ], ConvexHull, distanceJ, NextJDir]];

   Clear[PossiblePointsListJ];
   lösche
   Print["Checklist = ", CheckPoints];
   gib aus
   Print["NEXT"];
   gib aus
   Return[SaftyPossiblePointsListJ];
   gib zurück
  ];

CreateSyntheticPointsForFurtherChecks[] := Module[{},
                                           Modul
```

```
];
```